HOVEDPROSJEKT:

Evig Elev

Electronic Election System

FORFATTER(E):

Kent Elverum
Frode Elverum
Stian Frydenlund Lereng
Tom Fladsrud

Dato: 23. mai 2002

SAMMENDRAG AV HOVEDPROSJEKT

Tittel:	EvigElev	Nr. : 4
	Elektronisk valgsystem for Høgskolen i Gjøvik	Dato : 23.05.02
Deltaker(e):	Stian Lereng Kent Elverum	
	Frode Elverum	
	Tom Fladsrud	
Veileder(e):	Tom Røise	
Oppdragsgiver:	Høgskolen i Gjøvik	
Kontaktperson:	Hans Engenes	
Stikkord (4 stk)	Valgsystem, Java, JDBC , RUP	

Antall sider: 222	Antall bilag: 9	Tilgjengelighet (åpen/konfidensiell): Åpen

Kort beskrivelse av hovedprosjektet:

Oppgaven har gått ut på å utvikle et elektronisk valgsystem for Høgskolen i Gjøvik, med alle de utfordringer det har ført med seg. Disse utfordringene dreier seg i stor grad om sikkerhet under avstemmingen, og at anonymitet og demokrati blir ivaretatt.

Vi har valgt å utvikle systemet i Java, siden dette gir et fleksibelt system. Vi mener dessuten at Java er ”in”, og tilhører framtiden innen objektorientert programmering.

Det som er spesielt for vår løsning er at vi har lagt stor vekt på at den skal være åpen og generell, med mulighet for selv å konfigurere systemet etter eget behov. Dette vil føre til at systemet vil være levedyktig selv om det skulle forekomme endringer i dets omgivelser og brukermiljø i framtiden.

Systemet er utviklet med RUP som rammeverk. I løpet av utviklingsprosessen har det derfor blitt en god del dokumentasjon i tillegg til det ferdige systemet.

Forord

Det elektroniske va lgsystemet EvigElev® er utviklet av fire tredjeårsstudenter ved datalinjen på høgskolen i Gjøvik. Oppgaven ble valgt fordi den forespeilet mulighetene til å ta i bruk de kunnskaper vi har skaffet oss gjennom valg av hovedretning for inneværende studieår, og dessuten satte krav til tilegnelse av ny kunnskap. Problemene ved dagens valgsituasjon var trekk vi kjente oss godt igjen i og hadde et ønske om å gjøre noe med.

Vi ønsker å rette en stor takk til kontaktperson Hans Engenes, som under hele prosjektperioden har tatt seg tid til å engasjere seg i oppgaven og komme med nyttige innspill og konstruktiv kritikk. Som leder for valgstyret har Engenes meget god oversikt over rutinene ved dagens valgordning og dermed god innsikt i problemene dette fører med seg. Vi vil også gi ros til Birgith Børthus og resten av valgstyret som har tatt oppgaver på strak arm når prosjektgruppen har spurt om tjenester.

En takk rettes også til Tom Røise, som har vært en engasjert og kritisk veileder. Utover systemutviklingskompetanse har Røise god erfaring fra veilederrollen i tidligere hovedprosjekter og har dermed kunnet komme med generelle råd som angår hovedprosjektet. Dette er erfaringer som vi håper har gitt oss et fortrinn i den overordnede strukturen av systemet og hovedprosjektrapporten. IT -tjenesten har også vært behjelpelig; spesielt vil vi takke Tom Seljeflot som har vært den personen fra IT-tjenesten som gruppen har hatt mest kontakt med. Han har vist god vilje og godt initiativ til å løse oppgaver som har berørt problema tikken rundt manntallslister. Øivind Kolloen har vært en stor ressurs hva Java angår og har svart på spørsmål via e-mail, tatt en tur på grupperommet eller kommet med muntlige råd og anbefalinger ved flere anledninger.

Sist, men ikke minst vil vi få takke Anders Rindal, som omsatte våre skisser og tanker rundt utformingen av applikasjonens logo og overskrifter til et vellykket resultat.

Gjøvik, 23. mai 2002

Frode Elverum
loggansvarlig

Kent Elverum

Tom Fladsrud

Stian Frydenlund Lereng
prosjektleder

Innholdsfortegnelse

1 Innledning

1.1 Oppgavebeskrivelse

Prosjektet har gått ut på å utvikle et elektronisk valgsystem for Høgskolen i Gjøvik, som skal kunne brukes for å avholde alle typer valg ved høgskolen.

I dag utføres det flere ganger årlig valg av roller, styrer og råd ved høgskolen. Eksempler på organer og roller som det avholdes valg på er studentparlamentet, avdelingsstyre, høgskolestyre, dekanus, prodekanus, rektor og prorektor.

Et av hovedproblemene med dagens manuelle system er lav valgdeltakelse. De som ønsker å stemme må møte opp personlig i avstemmingslokalet og kunne legitimere seg. Et elektronisk valgsystem er i så måte interessant for å gjøre avstemmingsprosessen enklere og mer motiverende for velgerne, og dermed forsøke å øke valgoppslutningen.

Valgstyret har som oppgave å skaffe nok kandidater til de forskjellige valg, noe det ofte slites med å få til. Det var det en interessant utvidelse av valgsystemet å åpne for mulighet for nominasjon av kandidater ved forskjellige valg.

Når et valg skal avholdes, er det mange oppgaver som skal gjøres. Kandidater skal klargjøres, stemmesedler ska l trykkes opp, valget skal annonseres, det skal være vakter ved stemmeurnene hele valgperioden, og stemmene skal telles opp. De administrative oppgavene rundt valget er altså mange, tungvinte og ressurskrevende. Valgstyrets leder, som også har vært vår kontaktperson, har ansvaret for mange av disse oppgavene i dag. Et elektronisk valgsystem er derfor interessant for å kunne lette denne jobben, samtidig som det vil være ressursbesparende. Når det gjelder opptellingen av stemmesedlene vil et valgsystem også være av interesse for å kunne gi et korrekt valgresultat, uten mulighet for menneskelig svikt i opptellingen.

Administrator vil være ansvarlig for konfigurasjonen av valget, via nominasjons- og valgperioder, og til slutt opptelling av valgresultatet og publisering av dette resultatet på internett. Nominator på sin side vil kunne benytte systemet til å nominere kandidater, mens velger kan avgi sin stemme ved de forskjellige valg vedkommende er stemmeberettiget ved.

Et sentralt mål har vært å utvikle en løsning med brukervennlig brukergrensesnitt, slik at det skal bli motiverende for både velgerne og administrator å benytte valgsystemet. Det har også vært viktig å få systemet stabilt og pålitelig, slik at systemets brukere får tillit til systemet. Dette er helt nødvendig for et system som har de sikkerhetsmessige aspekter et valgsystem har.

1.2 Målgruppe (for rapporten og for oppgaven)

Målgruppen for oppgaven er Høgskolen i Gjøvik, herunder valgstyret, som vil fungere som administratorer av systemet, samt alle stemmeberettigede, som innbefatter studenter, lærere og andre ansatte.

Rapportens målgruppe er prosjektets veileder, oppdragsgiver og sensor, studenter som eventuelt skal videreutvikle systemet, og andre interesserte.

1.3 Systemets omverden

Høgskolen i Gjøvik er en statlig høgskole med 195 ansatte og 1400 studenter. Studentene og de ansatte er hovedsakelig fordelt på henholdsvis avdeling teknologi og avdeling helsefag. De ansatte er videre delt inn etter hvorvidt de har faglige eller tekniske stillinger. Disse studenter og ansatte er spredd utover et bygg på Brandbu og tre bygg på Gjøvik.

Høgskolens nettverk består i dag av mellom 400 og 500 arbeidsstasjoner lokalisert på disse fire byggene. Det er et tjuetalls Mac-maskiner, rundt 30 Linux-stasjoner, og resten Windows NT og Windows 2000 arbeidsstasjoner. Disse er koblet opp mot høgskolens intranett, og både studenter og ansatte kan logge seg på med sine personlige brukernavn og passord.

1.4 Faglig bakgrunn

Alle prosjektdeltagerne nærmer seg slutten av ingeniørutdanningen ved Høgskolen i Gjøvik, og har således en del faglig bakgrunn fra diverse fag som kunne dras inn i hovedprosjektarbeidet.

Ved prosjektets oppstart hadde et flertall av gruppedeltagerne godt innblikk i og erfaring med systemutvikling ved bruk av ”Rational Unified Process”, som ble benyttet som rammeverk for prosjektet.

Alle gruppedeltagerne har databasekunnskaper fra kursene ”Databaser I” og ”Databaser II”, samt grunnleggende javakunnskaper enten fra kurset ”Programmering mot WWW” eller fra ”Programvareutvikling”. Programmering i Java Servlets hadde ingen i gruppen vært borti tidligere, men hele gruppen fulgte kurset ”Klient- og serversideprogrammering” parallelt med hovedprosjektet og fikk gjennom det innføring på dette området.

1.5 Arbeidsformer

Ved prosjektets oppstart ble det tildelt grupperom på høgskolen, hvor vi etter hvert fikk etablert utviklingsmiljø. Dette grupperommet ble benyttet under bortimot hele prosjektperioden, med unntak av noe hjemmejobbing underveis.

Omtrent annenhver uke har det vært avholdt møter med veileder og kontaktperson, samt internmøter hver mandag. Videre har vi hatt annen møtevirksomhet når det har vært behov for det.

Under hele prosjektperioden ble det kontinuerlig skrevet og deretter lagt ut møtereferater, statusrapporter og andre relevante dokumenter på web, slik at kontaktperson, veileder og andre interesserte kunne holde seg oppdatert på prosjektet og dets status.

1.6 Organisering av rapporten

Vi har valgt å organisere rapporten i 10 kapitler på følgende måte:

- **Kapittel 1 - Innledning**
  Gir en helt kort innføring i prosjektet og prosjektarbeidet.

- **Kapittel 2 – Systembeskrivelse**
  Fokuserer på å vise helheten i systemet uten å gå inn på krav.

- **Kapittel 3 - Kravspesifikasjon**
  Kravspesifikasjonsdokumentet, som er tilpasset rapporten.

- **Kapittel 4 – Designdokument**
  Består av systemarkitekturbeskrivelse, utdrag fra designmodell, databasebeskrivelse og sentrale beslutninger for hvordan designet er bygd opp.

- **Kapittel 5 – Implementering, koding og produksjon**
  Beskriver systemets oppbygning og hvordan kodingen er implementert, samt hvordan de enkelte filer fungerer og hva de gjør.

- **Kapittel 6 – Testing og kvalitetssikring**
  Inneholder beskrivelser av prosedyrer for datalagring og beskrivelse av testing av systemet.

- **Kapittel 7 – Beskrivelse av utviklingsprosessen**
  Forteller om valg av utviklingsmodell, og anvendelsen av RUP i vårt prosjekt.

- **Kapittel 8 – Arbeidsmetoder og resultater**
  Dette er avslutningskapitlet. Her drøftes og diskuteres oppgaven, samt prosessen i prosjektet frem til det endelige resultatet. Videre kommer egenevaluering og en konklusjon til slutt.

- **Kapittel 9 - Litteraturliste**
  Litteraturliste for prosjektet.

- **Kapittel 10 - Vedlegg**
  Oversikt over rapportens vedlegg.

1.7 Terminologi

Begrep	*Definisjon*
Administrator	Brukeren som skal administrere systemet, sette opp valg og sørge for at opptelling og beregning av valgresultat blir gjort
Aktiv-modus	Betyr at en valgtype er aktiv og dermed i bruk. Har attributtet ErAktivt lik 1 i databasen
Avstemmingsperiode	Perioden det er mulig å avlegge stemme ved et valg. Perioden har en startdato og en sluttdato
Domene	Påloggingsgruppe mot NT-domene
Dropdown-liste	En nedtrekksliste hvor det er mulig å velge blant en rekke elementer
ER-modell	Entitet-Relasjon-modell. Modell som viser alle tabellene i databasen og sammenhengene mellom disse
Forhåndsstemme	En stemmeseddel som er avgitt manuelt på papir "utenfor" valgsystemet og må legges inn i systemet for den stemmeberettigede velgeren av administrator
Generator	Et kall som lager autonummer i databasetabellen når en ny post blir lagt til.
GUI	Grafisk brukergrensesnitt
Hovedadministrator	Hovedadministratoren er den av administratorene som mottar e-mailer som blir automatisk utsendt av systemet i forbindelse med kandidatnominasjon
JBuilder	Utviklingsverktøy for Java
Nominasjonsperiode	Perioden det er mulig å nominere kandidater til et valg. Perioden har en startdato og en sluttdato
Nominator	Bruker av systemet som kan nominere kandidater til valg
NT-domene	Domene som inneholder alle ansattes og studenters påloggingsnavn og passord.
NULL	Betyr ingenting. Brukes i forbindelse med databaser. NULL betyr at et felt ikke er fylt ut, og at verdien vil være tom.
Opptellingsresultat	Resultatet fra opptellingen av stemmesedlene ved et valg. Dette resultatet må godkjennes av valgstyret
Piksel	Brukes for å definere skjermoppløsning. Antall piksler er antall fargepunkter skjermen består av
Preferanse	Benyttes som en rangeringsmetode av kandidatene på stemmeseddelen ved valget
Redundans	Dobbeltlagring av data i database
Splash	Introduksjonsbilde mens applikasjonen lastes
Statusattributter	Attributter i databasen som forteller noe om status til et valg eller en valgtype (ErOpptalt, ErFerdig, ErAktivt)
Stemmeseddel	Innebefatter navnet på kandidatene til valget med tilhørende felt for valgt preferanse. Skjermbildet som blir presentert for brukeren idet han skal velge kandidater visualiserer den fysiske stemmeseddelen ved manuelle valg
Trigger	Et kall som blir automatisk utført når en definert handling blir gjort mot databasen

URL	Internettadresse
Valgregler	En kortfattet, tekstlig beskrivelse av det enkelte valg og hvilke regler som gjelder ved avstemming og beregning av de totale stemmene
Valgresultat	Det endelige valgresultatet som blir offentliggjort etter at det er godkjent av valgstyret og verv er fordelt.
Valgtype	Betegnelse på hva slags type valg som avholdes. Dette kan for eksempel være ”Rektorvalg”
Velgergruppe	En stemmeberettiget gruppe velgere ved et valg

Vi har brukt en del ord og begreper om hverandre for å forbedre og variere språket.

- Nedtrekksliste og kombinasjonsboks er også brukt i stedet for dropdown-liste.
- Bruker og administrator er brukt om hverandre for administrasjonsmodulen.
- Stemmeberettiget og velger brukes om hverandre for avstemmingsmodulen.

Generelt kan brukeren av systemet være kalt for en bruker, uansett hvilken modul det er snakk om.

2 Systembeskrivelse

Gjennom dette prosjektet har vi utarbeidet en detaljert kravspesifikasjon for systemet under utvikling. Dette kapittelet skal prøve å vise helheten i systemet, og ikke gå inn på direkte krav. Disse vil komme i kravspesifikasjonen i neste kapittel.

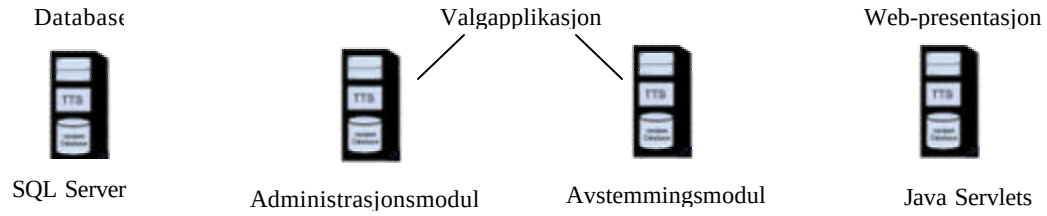
Ved å lese systembeskrivelsen i dette kapittelet håper vi at leseren raskt vil forstå systemets struktur uten at det blir for "tungt fordøyelig". Dette er altså ment å være en innledning til resten av kravspesifikasjonen.

Vi har også valgt å legge kravspesifikasjonen i sin helhet på vedlagt CD for interesserte lesere. Dette har vi gjort fordi man da får et mer helhetlig og sammenhengende bilde av kravspesifikasjonen.

2.1 Visjonsdokument

2.1.1 Introduksjon

Vi har som mål å utvikle et elektronisk valgsystem for Høgskolen i Gjøvik som kan tas i bruk etter endt hovedprosjekt. Systemet skal utvikles i to moduler, en administrasjonsmodul og en avstemmingsmodul. Administrasjonsmodulen skal gjøre de administrative oppgavene knyttet til valg enklere og mindre ressurskrevende. Disse rutinene vil i stor grad bli automatisert. Avstemmingsmodulen skal gjøre det lettere og sikrere å avlegge stemme ved valgene på skolen. Vi ønsker gjennom dette prosjektet å gjøre det enklere og mer motiverende for studenter og ansatte å avlegge stemme, og på denne måten bidra til økt valgdeltakelse.



Figur 2.1 Valgsystemets oppbygning

Systemet er forutsatt å fungere på den etablerte IT-infrastrukturen ved skolen, slik at ansatte og studenter kan avgi stemme herfra, uansett hvor på skolen de sitter og hvilken plattform de bruker. Målgruppen for systemet er alle stemmeberettigede, herunder studenter, lærere og andre ansatte, samt representanter fra valgstyret, som kommer til å administrere systemet. Da gruppen av stemmeberettigede er nokså sammensatt og kjennskapen til datakunnskaper må forventes å være variabel, ser vi det som en viktig del av prosjektet å utforme en brukervennlig og intuitiv løsning. Det samme gjelder for administrasjonsmodulen, da man heller ikke kan forvente at administratorene er spesielt datakyndig.

2.1.2 Posisjonering

Forretningsmulighet
Våre undersøkelser tyder på at markedet for elektroniske valgsystemer er voksende, noe som trolig skyldes at fordelene er mange og tungtveiende. Det eksisterer firmaer som selger valgsystemer til andre bedrifter og institusjoner. Vi tar ikke sikte på å være en konkurrent til disse, men legger vekt på å utvikle et system som er mest mulig tilpasset og kan være til god nytte for høgskolen. Følgelig vil vår løsning bli preget av at valgansvarlig skal få muligheten til å konfigurere valgene selv. Dette gjør valgsystemet fleksibelt og levedyktig, siden man har mulighet til å legge til nye valgtyper med tilhørende regler, sette opp nye valg med aktuelle datoer, og ellers klargjøre hele valgene elektronisk. Vi ser for oss at systemet åpner for en utvidelse til en nettbasert løsning i fremtiden, slik at stemmeberettigede ikke trenger å befinne seg på høgskolens område for å avgi stemmer.

2.1.3 Problemer ved dagens system
Dagens system baserer seg på manuelle valgrutiner hvor de som ønsker å stemme må møte opp personlig i avstemningslokalet på valgdagen og legitimere seg. Dette har ført til at høgskolen sliter med lav valgdeltakelse og liten interesse fra både velgere og kandidater. Potensielle velgere har liten eller ingen kjennskap til eller tilgjengelig informasjon om de aktuelle kandidatene, og det er trolig at også dette er med på å redusere motivasjonen hos velgerne. De administrative oppgavene rundt valget er mange, tungvinte og ressurskrevende. Ordningen innebærer mye opptellingsarbeid, papirarbeid, stemmesedler kan bli borte og det er også et problem med feilutfylte og blanke stemmesedler og behandlingen av disse.

2.1.4 Produktfortrinn

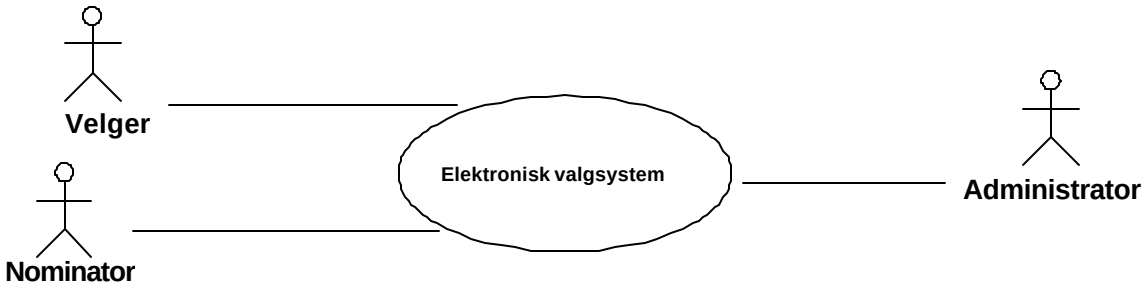
Et elektronisk valgsystem vil lette det administrative arbeidet rundt valget, da dette i stor grad vil automatiseres. Opptelling av stemmesedler og utarbeiding av rapporter vil bli gjort av systemet. Feilutfylte og blanke stemmesedler vil bli avvist av systemet og eliminerer dermed problemer man har hatt på dette området. Det nye systemet vil være billigere og mindre tids- og ressurskrevende. Enklere prosedyrer vil sannsynligvis føre til økt velgeroppslutning og dermed større sannsynlighet for at en ”rettferdig” kandidat blir valgt. Systemet vil også sørge for at valgjuks blir en tilnærmet umulighet. Et elektronisk valgsystem vil gi velgerne mulighet til å få informasjon om kandidatene slik at man har bedre grunnlag for å velge hvilken kandidat som fortjener stemmen. Systemet vil også gjøre det enklere å foreslå kandidater til de ulike valgene.

2.1.5 Brukermål

Velger:	avgi stemme, logge inn, se valgresultat, lese kandidatinformasjon
Nominator:	nominere kandidat
Administrator:	logge inn, opprette velgergruppe, opprette valgtype, legge til verv, endre valgtype, angi valgtype, legge inn valgbeskrivelse, angi stemmevekting, sette tidsfrister, legge inn manntallsliste, legge inn kandidater og informasjon om disse, legge inn ferdigutfylte stemmer, avlyse valg, igangsette stemmeopptelling, se oversikt over nominerte kandidater, hente ut opptellingsresultat og godkjenne dette, hente ut stemmesedler, hente ut opptellingsrapport, hente ut valgresultat, angi avanserte innstillinger for systemet, legge til nye administratorer

2.1.6 Produktoversikt

Systemet vil utføre tjenester for velger, nominator og administrator som brukere.



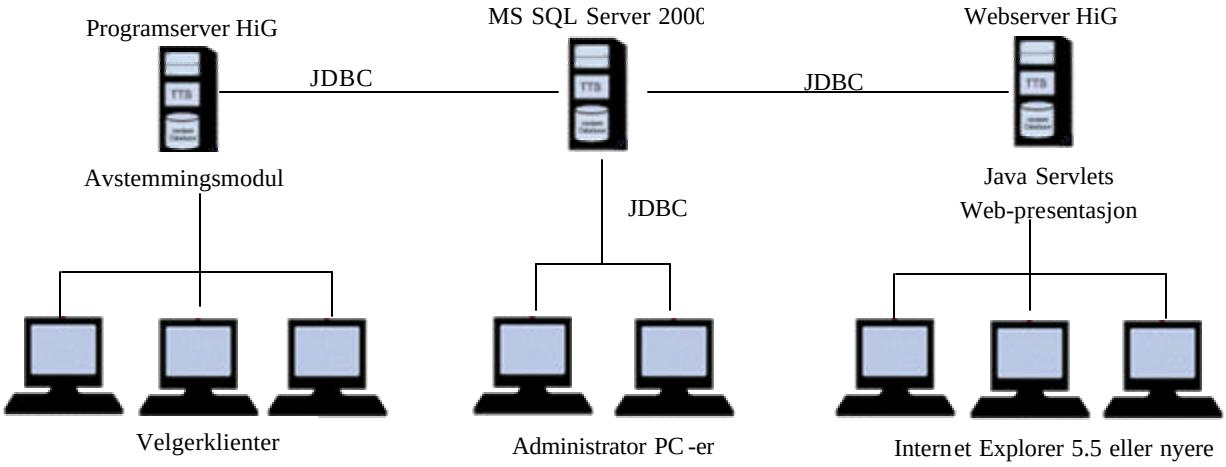
2.2 Systembeskrivelse for bruker

2.2.1 Systemets omgivelser

Omgivelsene til systemet vil være tredelt. Den ene delen er MS SQL Server databasen som vil ligge på en av IT-tjenestens servere. Denne delen vil kun kunne aksesseres via de to modulene systemet består av. Det trengs gyldig brukernavn og passord for å få tilgang til databasen, slik at det kun er IT-tjenesten, databaseansvarlig ved høgskolen, og systemet som vil kunne operere mot eller gjøre noe med databasen.

Den andre delen, avstemmingsmodulen, vil kjøres fra en nettverksressurs som alle på HiG skal ha tilgang til. Dette vil være en dedisert server hos IT-tjenesten. Den vil kunne aksesseres via alle PC-ene i HiG sitt nettverk.

Den tredje delen, administrasjonsmodulen, vil ligge lokalt på administratorenes arbeidsstasjoner (PC-er). Dette er av sikkerhetsmessige aspekter. Da den ligger lokalt vil kun de innlagte administratorene få mulighet til å bruke denne delen av systemet. De må i tillegg benytte riktig brukernavn og passord for å logge seg inn på systemet.



Figur 2.2 Systemoversikt

2.2.2 Systemets brukere

Avstemmingsmodulen vil bli brukt av både stemmeberettigede ved høgskolen i Gjøvik og skolens studenter og ansatte som ønsker å nominere kandidater, uavhengig av om disse har stemmerett eller ikke. En lettfattelig brukermanual i kombinasjon med hjelpmeny i systemet skal være nok til å hjelpe brukeren gjennom modulens funksjonalitet.

Administrasjonsmodulen vil bli brukt av administratorene. Denne vil nødvendigvis være mer kompleks enn avstemmingsmodulen siden den er mer omfattende. Også her vil det finnes både hjelpmeny i systemet og en brukermanual som kan hjelpe administratorene til å forstå og

bruke systemet. Det er viktig at administratorene får en grundig innføring i systemet og lærer seg å bruke det skikkelig, slik at de kan stole på det.

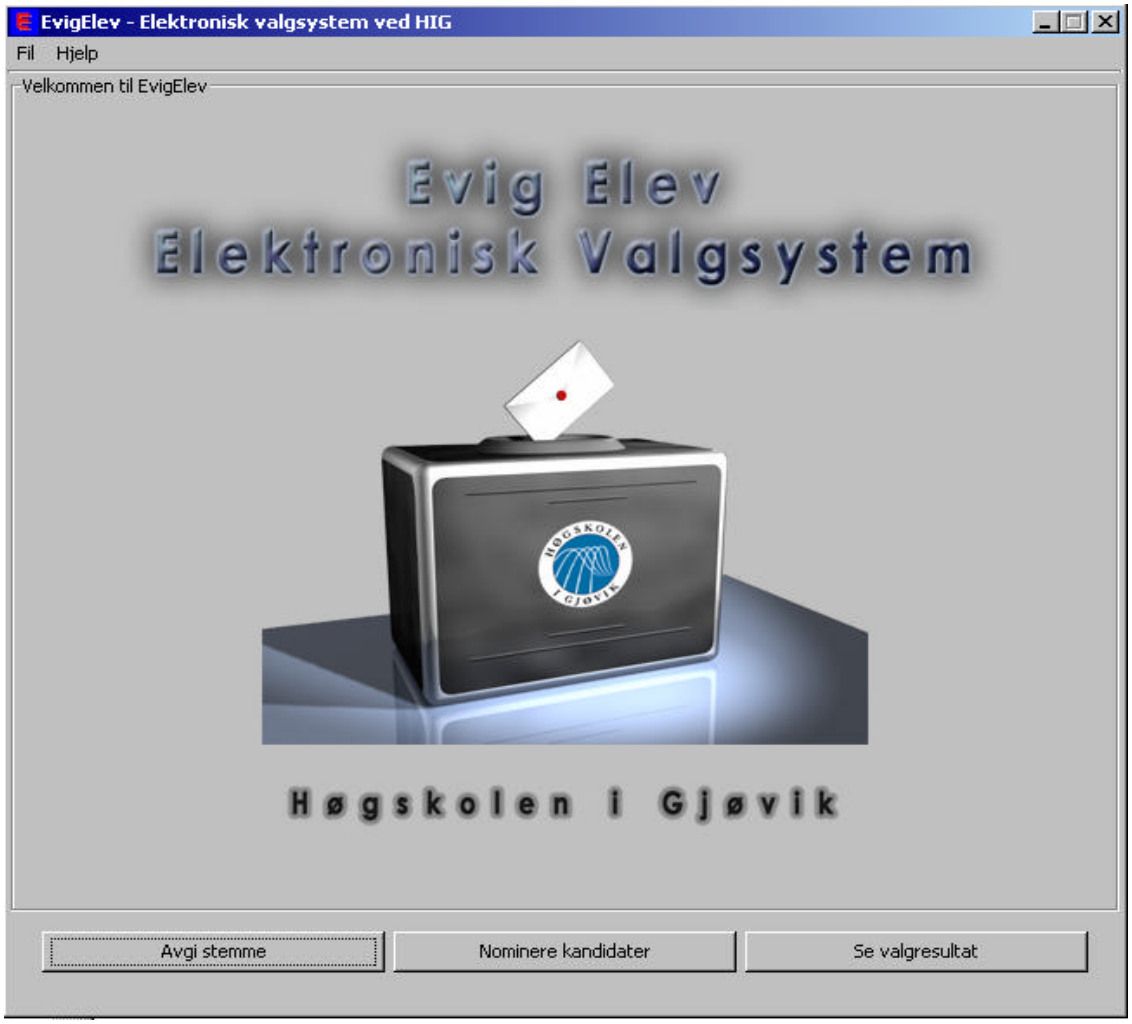
Det vil ikke være noen inndeling i brukernivåer i systemet, siden vi har utviklet administratordelen i en egen modul. Brukermålene til systemets brukere, velgere og administratorer, er beskrevet under punkt 2.1.5 Brukermål, i visjonsdokumentet.

2.2.3 Modulbeskrivelser

Siden systemet skal håndtere to så vidt forskjellige områder som å stemme ved et valg og å administrere valget har vi valgt å dele det opp i to atskilte deler. Vi mener at dette har ført til at hver modul er raskere og enklere enn om de skulle vært slått sammen til en modul. Sikkerhetsmessige hensyn er også en medvirkende årsak til at vil har valgt å skille de to delene av systemet.

Avstemmingsmodul

Fra avstemmingsmodulen vil brukerne av systemet kunne avgi stemmer anonymt ved valg hvor vedkommende har stemmerett, nominere kandidater til valg som skal avholdes, og se valgresultat for valg som er ferdige.



Figur 2.3 Hovedmeny for avstemmingsmodul

Under stemmeavgivingen og utfyllingen av stemmeseddelen vil velgeren måtte preferere kandidatene i ønsket rekkefølge, akkurat som ved dagens manuelle valg. Systemet vil sjekke at stemmeseddelen er riktig utfylt, og man slipper da problemet med feilutfylte stemmesedler. Derfor vil kun lovlige verdier, og korrekt utfylte stemmesedler skrives til databasen. Noe som er viktig med tanke på demokrati og anonymitet er at ikke velgerens ID eller navn blir knyttet opp mot en bestemt stemmeseddel slik at man kan finne ut hvem som har stemt hva. Dette blir *ikke* mulig å finne ut i vårt system.

I nominasjonsdelen av modulen kan brukeren velge å nominere en kandidat for første gang til et aktuelt valg, eller gi en nominasjon til en kandidat som er nominert av andre. Man kan kun

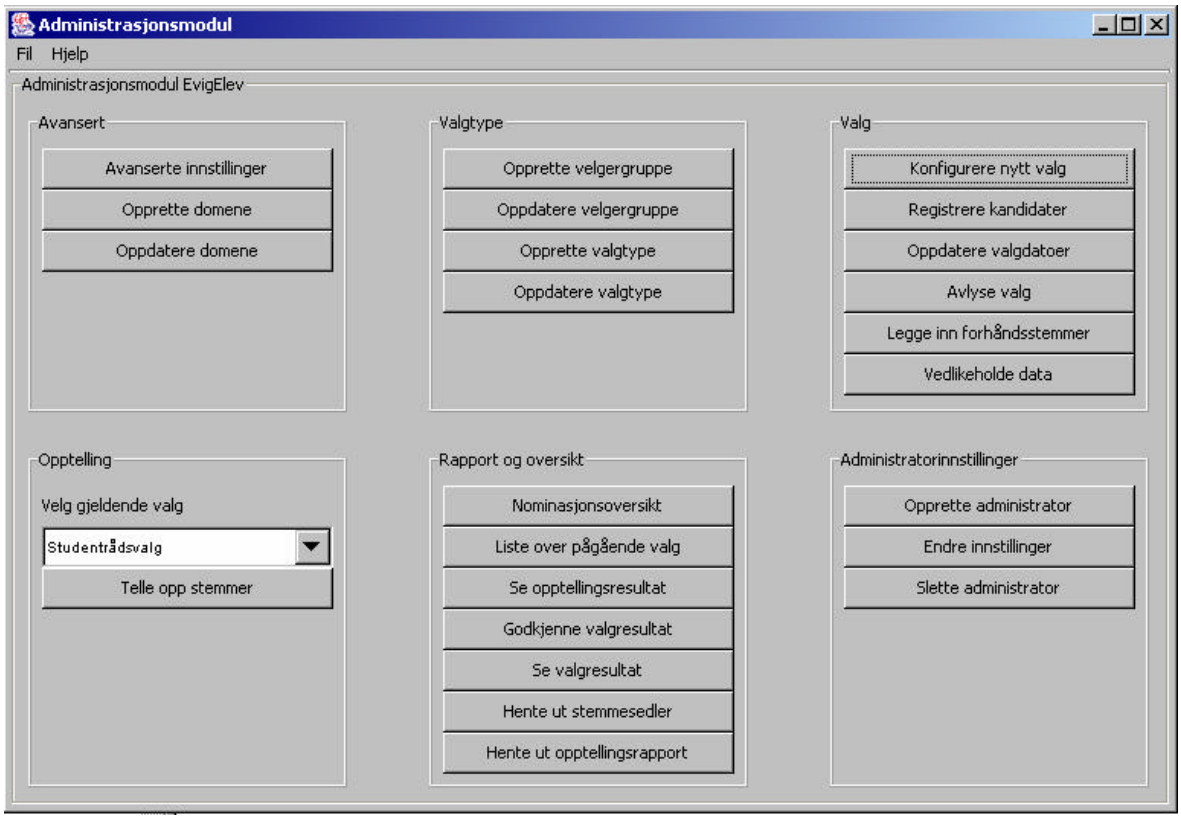
nominere den samme kandidaten én gang ved det samme valget. Dersom en nominert kandidat oppnår fem ulike nominasjoner, regnes denne som en reell kandidat til valget og blir varslet om dette per e-mail.

Den siste delen i denne modulen er å se valgresultater. Brukeren blir da sendt inn på en webside hvor valgresultater ligger.

Avstemmingsmodulen vil jobbe mot en SQL Server-database, samt foreta passordsjekking mot NT-domene ved innlogging og benytte mailserveren til HiG ved automatisk utsendelse av e-mail ved nominasjon.

Administrasjonsmodul

I denne modulen kan administratoren foreta alle nødvendige operasjoner som trengs for å avholde et valg. Han vil kunne opprette nye valgtyper, starte valg av eksisterende typer, registrere kandidater, legge inn forhåndsstemmer, avlyse valg og telle opp stemmene elektronisk. Modulen innehar også funksjonalitet for rutiner som er aktuelle etter at avstemmingsperioden er over, slik som å godkjenne opptellingsresultat, hente ut rapporter og oversikter samt å slette gamle stemmesedler og valg.



Figur 2.4 Hovedmeny for administrasjonsmodul

Det vil også være mulighet for å legge til nye administratorer, slette administratorer og endre personlige innstillinger.

Under alle nevnte operasjoner vil systemet sjekke at korrekte opplysninger blir fylt inn til enhver tid, at alle påkrevde opplysninger blir angitt, og at kun de valg som skal kunne behandles vil bli vist i nedtrekkslistene til enhver tid. På den måten skal ikke administratoren kunne skrive feil data eller mangelfull data til databasen.

Denne modulen vil i likhet med avstemmingsmodulen jobbe mot den samme SQL Server-databasen.

2.2.4 Bruk av database

Etter hvert som tiden går og flere og flere valg har blitt avholdt vil databasen vokse betydelig i størrelse. Det er derfor av stor betydning av administratoren kan slette data som ikke er relevant å ta vare på, eller at disse automatisk blir slettet når de ikke trengs lenger.

Det er også viktig at viktige data blir tatt vare på så lenge som dette er påkrevd.

Ellers kan databasen brukes til historikk i fremtiden. Man kan hente ut valgresultater, måle valgoppslutning, og hente ut statistikk man ønsker fra databasen.

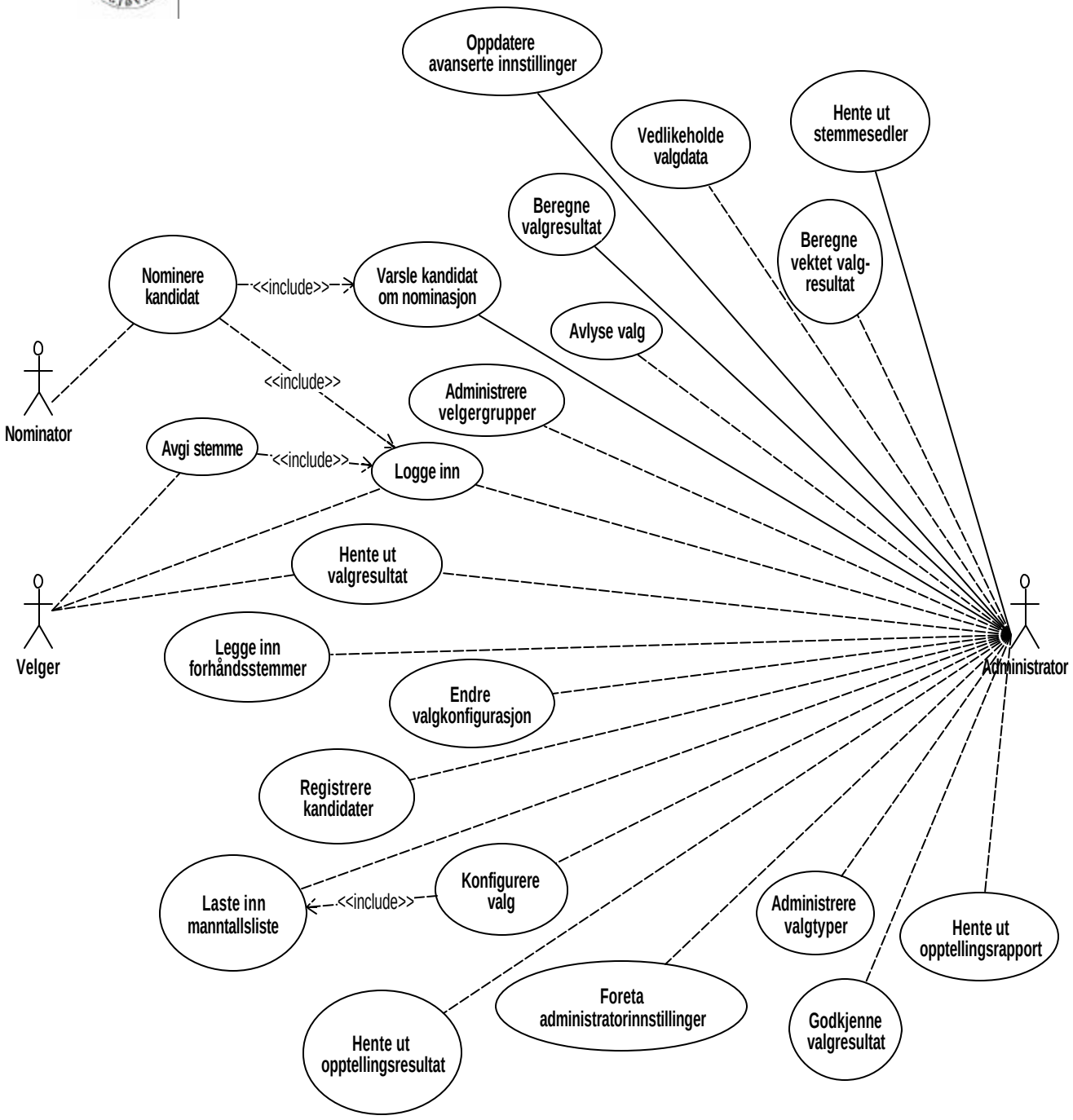
3 Kravspesifikasjon

3.1 Innledning

I dette kapittelet har vi valgt å ta med utvalgte deler av kravspesifikasjonen som vi synes belyser sentrale emner og gir et innblikk i hvordan vi har jobbet med å avdekke krav og detaljere ned disse. Vi har valgt å vise tre use cases som er fullt detaljert og high-level-beskrivelser for resten for å gi leseren en god oversikt over systemets krav. Use casene 'Avgi stemme', 'Konfigurere valg' og 'Nominere kandidat' er tre sentrale use cases som representerer den generelle strukturen vi har benyttet ved use case-detaljering. Vi har også valgt å legge ved system sekvensdiagrammer til use casene fordi dette er et artifakt som hører inn under use case-modellen.

Ytterligere low-level-beskrivelser med system sekvensdiagrammer vil finnes i vedlegg H.

I use case-diagrammet har vi tre aktører. I utgangspunktet hadde vi bare administrator og velger, men siden vår definisjon av velger tilsier at personen er stemmeberettiget til et valg, og kravet til systemet er at en hvilken som helst ansatt eller student skal ha mulighet til å nominere kandidater til valg, valgte vi å skille mellom disse rollene. En nominator trenger med andre ord ikke å være en velger. Ellers gir use case-modellen et riktig inntrykk av at administrasjonsmodulen blir tyngre funksjonalitetsmessig enn avstemmingsmodulen.



Figur 3.1 Use Case-diagram

3.2 Use Case -prioritering

A) Risk – teknisk kompleksitet
B) Usability – betydningen av brukervennlighet
C) Coverage – krav til berøring i tidlige iterasjoner, bred implementering over flere komponenter (informasjon og innsikt i design) - use cases som bør berøres tidlig/grunnlaget for løsningen
D) Critic ality – funksjoner med høy forretningsmessig verdi
E) Betydning for systemets arkitektur

Vi bruker en skala fra 1 til 3, hvor 3 tilsvarer det viktigste/mest kritiske.

Kategori	*Vekt*	*Range*
A	3	1-3
B	3	1-3
C	1	1-3
D	1	1-3
E	2	1-3

Usability er vektlagt såpass høyt siden det er av stor betydning at produktet skal gjøre valgprosessen enklere og mer motiverende.

Use Case	**A**	**B**	**C**	**D**	**E**	**Total**
Nominere kandidater	2	3	1	2	2	22
Avlyse valg	1	1	1	1	1	10
Avgi stemme	3	3	3	3	3	30
Administrere valgtyper	2	3	2	2	2	23
Hente ut opptellingsresultat	1	2	1	2	1	14
Varsle om nominasjon	2	1	1	2	2	16
Konfigurere valg	3	3	3	2	3	29
Hente ut stemmesedler	2	1	1	1	1	13
Logge inn	2	1	1	1	2	15
Hente ut valgresultat	1	2	1	2	1	14
Beregne vektet valgresultat	3	1	2	3	2	21
Laste inn manntallsliste	2	1	2	1	2	16
Endre valgkonfigurasjon	1	2	1	1	1	13
Godkjenne valgresultat	2	3	1	2	1	20
Registrere kandidater	1	2	2	1	1	14
Beregne valgresultat	3	1	3	3	2	22
Oppdatere avanserte innstillinger	1	1	1	1	1	10
Foreta administratorinnstillinger	1	2	1	1	1	13
Administrere velgergrupper	1	1	2	1	1	11
Legge inn forhåndsstemmer	2	2	1	2	2	19
Vedlikeholde data	2	2	1	1	2	18
Hente ut opptellingsrapport	2	2	1	2	2	19

Prioritert liste over use cases

1. Avgi stemme
2. Konfigurere valg
3. Nominere kandidater
4. Beregne valgresultat
5. Administrere valgtyper
6. Beregne vektet valgresultat
7. Godkjenne valgresultat
8. Legge inn forhåndsstemmer
9. Hente ut opptellingsrapport
10. Vedlikeholde data
11. Laste inn manntallsliste
12. Varsle om nominasjon
13. Logge inn
14. Registrere kandidater
15. Hente ut opptellingsresultat
16. Hente ut valgresultat
17. Hente ut stemmesedler
18. Endre valgkonfigurasjon
19. Foreta administratorinnstillinger
20. Administrere velgergrupper
21. Avlyse valg
22. Oppdatere avanserte innstillinger

3.2.1 Begrunnelse for prioritering

Listen over prioriterte use cases stemmer godt overens med de verdier som de enkelte use cases er tilegnet i risikoanalysen. Vi har ikke byttet om rekkefølge på noen av use casene fra prioriteringen, fordi vi etter å ha vurdert det mener at det i dette tilfellet gir det beste bildet på viktigheten av hvert use case. Vi har begrunnet hvorfor vi har foretrukket enkelte use cases foran andre med samme totalverdi i prioriteringen.

Vi har prioritert ”Nominere kandidat” foran ”Beregne valgresultat” av den grunn at førstnevnte use case er innledningen i selve valgprosessen og dermed trolig er kurant å få på plass tidlig, mens opptellingsproblematikken vil være avhengig av at enkelte deler av systemet må være på plass og følgelig vil bli vanskelig å teste såpass tidlig i prosessen.

Vi har også valgt å foretrekke ”Legge inn forhåndsstemmer” før ”Hente ut opptellingsrapport”, siden det å legge inn forhåndsstemmer angår selve avstemmingen og derfor er viktigere enn å se en rapport over opptellingen.

”Laste inn manntallsliste” er prioritert foran ”Varsle om nominasjon”, siden førstenevnte er kritisk for at man kan avholde et valg, og den sistenevnte er en litt mer ”kjekt å ha”-funksjon.

Videre har vi valgt å sette ”Registrere kandidater” foran ”Hente ut valgresultat” og ”Hente ut opptellingsresultat” fordi de sistnevnte i utstrakt grad baserer seg på funksjonaliteten til use

caset ”Beregne valgresultat”, mens ”Registrere kandidater” tar fatt i et område av systemet som ikke berøres i andre use case.

Vi valgte derfor i første omgang å ta tak i de tre-fire høyest prioriterte use casene, siden disse både er arkitekturmessig tunge og vil sette fokus på ulike og vitale deler av det ferdige systemet.

Use casene “Legge inn forhåndsstemmer”, ”Hente ut opptellingsrapport” og ”Vedlikeholde data” er prioritert på 8. 9. og 10. plass, men har likevel ikke fått low-level beskrivelser. Grunnen til dette er at disse, sammen med et par andre use cases, kom til sent i prosessen. Vi så da ikke nytteverdien av å bruke tid og ressurser på å skrive low-level for disse. Vi har likevel oppdatert use case-modellen, prioritert dem og skrevet high-level beskrivelser for dem etter å ha konferert veileder angående problemstillingen. Disse use casene har stort sett blitt til i forbindelse med at oppdragsgiver har kommet med nye ønsker og innspill under brukertesting og demonstrasjoner av pilotversjoner av systemet. Noen av disse use casene er “kjekt å ha”-funksjonalitet, mens andre er litt mer vesentlig for systemet.

3.3 Use Case-beskrivelser

Vi har valgt å ta inn hele use case-beskrivelsene for tre av de viktigste use casene for å vise eksempler på low-level-beskrivelser. Disse berører vitale deler av systemet, og har vært sentrale i utviklingsprosessen. Videre har vi tatt med high-level beskrivelser for resten av use casene, for å gi leseren en kort innføring i systemet samt et innblikk i hva de forskjellige use cases omfatter.

USE CASE: Avgi stemme

High-level

Aktør:	Velger
Type:	Primær
Beskrivelse:	Velgeren skal kunne lese informasjon om og gi ønskede preferanser til en eller flere kandidater ved et bestemt valg. Hver velger har kun mulighet til å stemme én gang. Stemmen vil være anonym, slik at ingen kan spore denne tilbake til en bestemt velger. Det vil ikke være mulighet for å levere en feilutfylt stemmeseddel.

Low-level

Aktør:	Velger
Hensikt:	Sørge for at velgeren gis mulighet til enkelt og sikkert å avgi sin stemme med ønskede preferanser på de nominerte kandidatene, lese relevant informasjon om hver kandidat, og at disse blir registrert i database uten mulighet for å spore stemmen tilbake til tilhørende velger.
Oversikt:	Velgeren kan lese informasjon om kandidatene, og så bestemme seg for hvem av kandidatene som skal få førstepreferanse, andrepreferanse, tredjepreferanse og så videre. Når velgeren har gjort de riktige prioriteringer avlegges stemmen. Denne blir så lagret i en database og velgeren blir satt til å ha avlagt sin stemme.
Prebetingelse:	Velgeren er innlogget i systemet. Valget er aktivt. Velgeren har ikke stemt ved dette valget tidligere.
Postbet.:	Stemmeseddel er korrekt utfylt og registrert i databasen. Velgeren settes til å ha avlagt stemme.

Main Success Scenario

Aktørhendelser	*Systemhendelser*
	1. Use caset "Logge inn" kjøres
2. Velger angir ved hvilket valg han ønsker å avgi stemme	
	3. Systemet henter stemmeseddel for ønsket valg
4. Velger angir ønsket preferanse på de nominerte kandidatene	
5. Velger avgir sin stemme	
	6. Systemet kontrollerer at stemmeseddel er korrekt utfylt
	7. Systemet genererer forhåndsvisning av kandidatene etter angitte preferanser
8. Velger bekrefter at han/hun ønsker å avgi stemmeseddelen slik den er fylt ut	
	9. Systemet registrerer at velger har avgitt stemme og stemmeseddel lagres i database, og gir velger bekreftelse på at stemmeseddelen ble avgitt

Extensions

2

a) Velgeren har ikke stemmerett ved noen valg for øyeblikket
   1) Velger gis beskjed om at han ikke kan stemme ved noen valg

3

a) Systemet kan ikke opprette kontakt med database
   1) Velger gis beskjed om at stemmeseddel for ønsket valg ikke kan hentes, og at han må prøve igjen senere

6

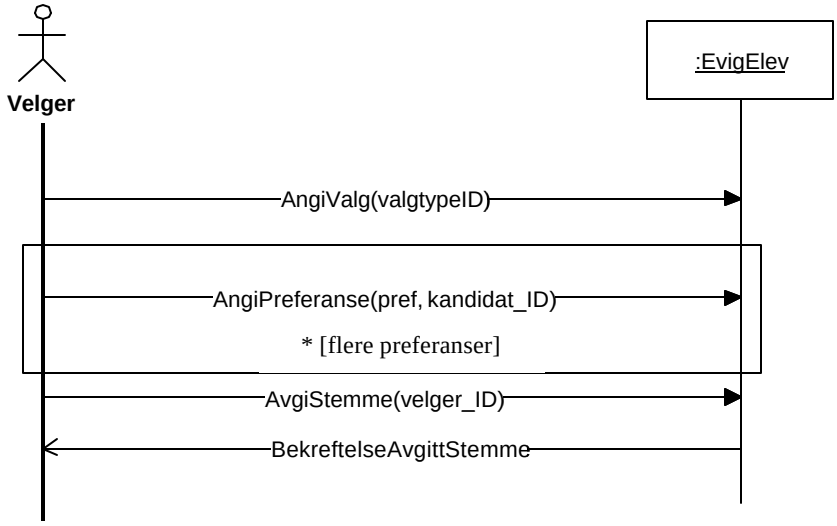
a) Systemet kan ikke godkjenne avgitt stemmeseddel
   1) Velger blir bedt om å korrigere feilene på stemmeseddelen

8

a) Velger sier at han/hun ikke ønsker å avgi stemmeseddelen slik den er fylt ut
   1) Systemet sender velgeren tilbake til stemmeseddelen slik at velgeren kan korrigere stemmeseddelen etter eget ønske

9

a) Systemet kan ikke opprette kontakt med database
   1) Velgeren gis beskjed om at stemmen ikke kan avgis for øyeblikket, og at han må prøve på nytt


Velger
:EvigElev
AngiValg(valgtypeID)
AngiPreferanse(pref, kandidat_ID)
* [flere preferanser]
AvgiStemme(velger_ID)
BekreftelseAvgittStemme

USE CASE: Konfigurere valg

High-level

Aktør:	Administrator
Type:	Primær
Beskrivelse:	Før hvert valg må administrator konfigurere det forestående valget. Dette innebærer å velge blant innlagte valgtyper hvilket valg som skal avholdes, sette opp datoer for valget, og laste inn lister over hvem som er stemmeberettiget. Innstillingene og informasjonen vil forhåndsvises underveis, slik at opplysningene kan dobbeltsjekkes og eventuelt justeres.

Low-level

Aktør:	Administrator
Hensikt:	For at et forestående valg skal kunne arrangeres av systemet, må valget konfigureres riktig for akkurat dette valget. Ferdig opprettede valgtyper skal gjøre konfigurasjonsjobben raskere og enklere for administratoren. Bare nødvendige valgdatoer, manntallslister og kandidater for det aktuelle valget trenger å registreres for hver gang et valg av en bestemt valgtype avholdes, mens valgregler og annen statisk informasjon om valget ligger lagret i valgtypen som hentes frem.
Oversikt:	Før et valg skal avholdes må administratoren klargjøre valget for nominasjon og avstemming. Administratoren må velge riktig valgtype for det forestående valget, og dessuten laste inn lister over hvem som er stemmeberettiget ved det aktuelle valget. Videre må datoer for nominasjons-, avstemmings- og vervperiode angis, sammen med dato for nominasjonsfrist.
Prebetingelse:	Administrator er innlogget i systemet. Den aktuelle valgtypen er registrert og informasjon om denne er riktig.
Postbet.:	Riktig utvalg av stemmeberettigede er gjort i henhold til valgt valgtype. Valg er opprettet.

Main Success Scenario

Aktørhendelser	*Systemhendelser*
1. Administrator velger valgtype	
	2. Systemet henter innstillingene for den valgte valgtypen
	3. Use case "laste inn manntallsliste" kjøres
4. Administrator godkjenner innstillingene	
5. Administrator angir gyldig tidsramme for nominasjons-, avstemmings- og vervperiode samt dato for nominasjonsfrist	
	6. Systemet kontrollerer at angitte datoer er riktig utfylt
	7. Systemet generer forhåndsvisning av valginformasjon
8. Administrator godkjenner valginformasjon	
	9. Systemet lagrer valgkonfigurasjon
	10. Systemet gir administrator bekreftelse på at valget er lagret

Extensions

1
a) Den ønskede valgtypen eksister ikke
   1) Administrator må opprette ny valgtype

2
a) Systemet kan ikke opprette kontakt med database
   1) Administrator gis beskjed om at innstillingene ikke kan hentes, og at han må prøve igjen senere

4
a) Administrator oppdager at innstillingene ikke er korrekte
   1) Administrator avbryter valgkonfigurasjonen og velger å oppdatere valgtypen

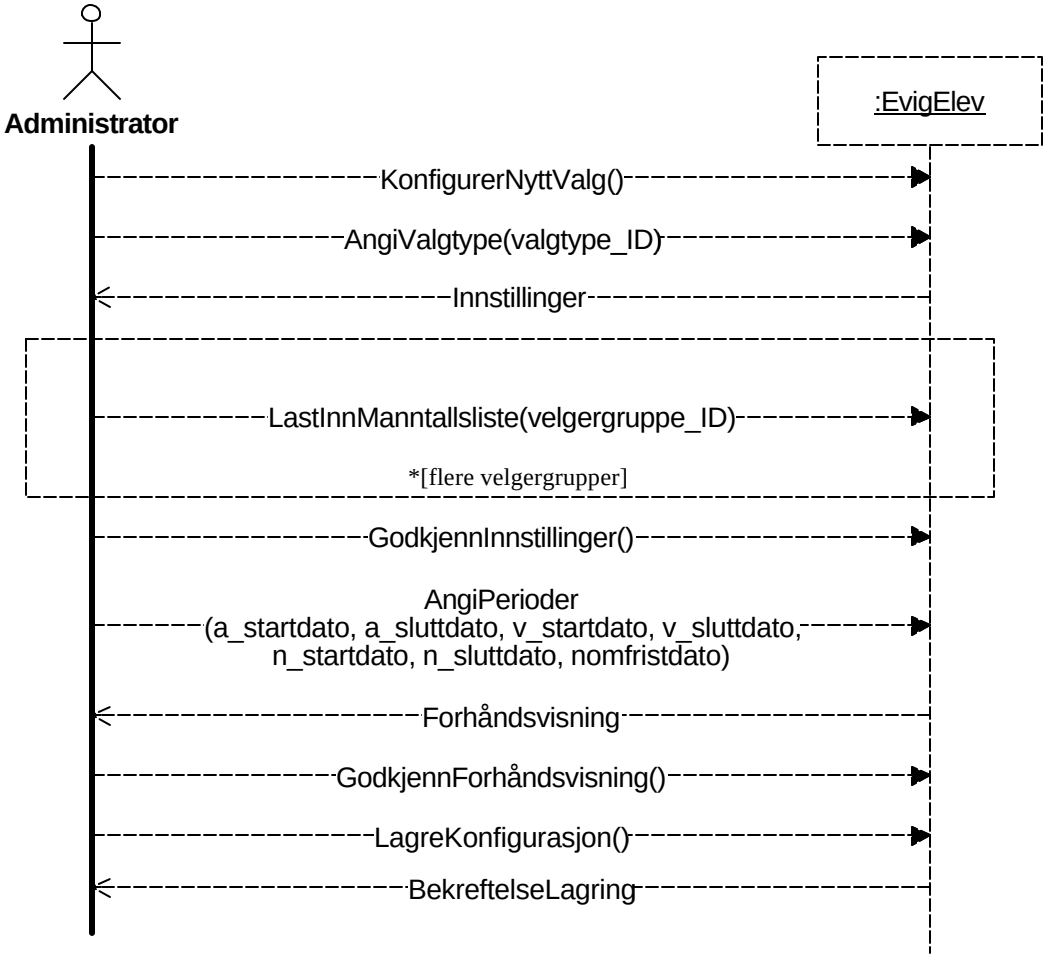
6
a) Administrator har valgt datoer som ikke er i riktig rekkefølge i forhold til hverandre
   1) Administrator får beskjed om å korrigere de aktuelle datoene

8

a) Administrator oppdager feil i opplysningene som er utfylt
   1) Administrator må gå tilbake til det stedet hvor han ønsker å gjøre endringer og fortsette konfigurasjonen som normalt derfra

9

a) Systemet kan ikke opprette kontakt med database
   1) Administrator gis beskjed om at valgkonfigurasjon ikke kan lagres, og at han må prøve igjen senere



USE CASE: Nominere kandidat

High-level

Aktør:	Nominator
Type:	Primær
Beskrivelse:	En gitt tidsperiode før et valg vil alle stemmeberettigede bli gitt muligheten til å kunne nominere kandidater ved dette valget. En kandidat vil trenge nominasjon fra fem ulike personer for å være reell kandidat til valget. Da vil e-mailer bli sendt ut for å varsle både hovedadministratoren og kandidaten om nominasjonen. Nominator vil kunne nominere så mange kandidater denne vil, men kan nominere hver enkelt kandidat kun én gang. En nominasjon vil være anonym i den forstand at det ikke blir opplyst om hvem som har nominert en kandidat, men det må nødvendigvis lagres i databasen.

Low-level

Aktør:	Nominator
Hensikt:	Tradisjonelt har det vært et problem med dårlig kandidatoppslutning ved de fleste valg ved høgskolen. For å øke muligheten til å få flere kandidater nominert til valgene, skal systemet gjøre det enklere for brukerne å nominere kandidater til et valg på en uformell måte, samt lette administratorens jobb med å sjekke om kandidater virkelig vil stille til det valget han/hun er nominert til.
Oversikt:	I et gitt tidsintervall før et valg vil høgskolens ansatte og studenter få muligheten til å nominere kandidater til valget. For å kunne nominere kandidater ved dette valget må velgeren logge seg inn i systemet. Han/hun vil så bli gitt muligheten til å enten nominere en ny kandidat, eller gi nominasjon til en kandidat som allerede er blitt nominert av noen andre (dette er hensiktsmessig siden en kandidat trenger fem nominasjoner for å være aktuell kandidat til et valg). Når nominator har foreslått en kandidat, vil systemet sjekke at alle opplysninger er fylt inn, blant annet kandidatens e-mailadresse. Når en kandidat har blitt nominert fem ganger, vil det automatisk sendes ut en e-mail hvor kandidaten får spørsmål om han/hun virkelig vil stille til valget, sammen med en del andre relevante spørsmål angående kandidaturet. Hovedadministrator vil også bli tilsendt en e-mail når en kandidat er nominert fem ganger. Denne administratoren vil også være mottaker av svarmailen fra den nominerte kandidaten.
Prebetingelse:	Nominasjonen foregår mellom start- og sluttdato for nominasjonsperioden.
Postbet.:	Brukeren har fylt inn påkrevde opplysninger.

Main success scenario

Aktørhendelser	*Systemhendelser*
	1. Use caset ”Logge inn” kjøres
2. Velger angir ved hvilket valg han ønsker å nominere	
3. Velger oppgir informasjon om kandidaten han vil nominere	
4. Velger sender sin nominasjon	
	5. Systemet sjekker at påkrevde opplysninger er fylt inn
.6. Velger bekrefter nominasjonen av kandidaten	
	7. Systemet sjekker at den nominerte kandidatens antall nominasjoner er lik fem
	8. Use Caset ”Varsle om nominasjon” overtar
	9. Systemet viser en bekreftelse på kandidatnominasjon

Extensions

2

a) Det foregår ingen valgnominasjoner for øyeblikket
    1) Velger gis beskjed om at ingen valg hvor vedkommende er stemmeberettiget er aktive for nominasjon for øyeblikket

5

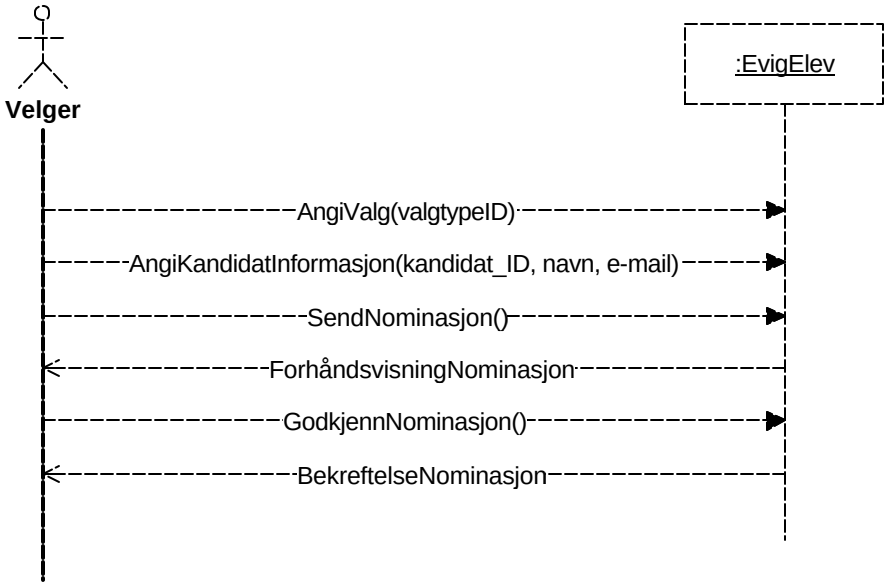
a) Systemet finner ut at velgeren ikke har angitt alle påkrevde opplysninger
    1) Velger blir bedt om å fylle inn nødvendige opplysninger

6

a) Velgeren godkjenner ikke nominasjonen.
    1) Systemet avbryter nominasjonen og ingen e-mailer blir utsendt

7

a) Kandidatens antall nominasjoner ved det aktuelle valget er ikke lik fem.
    1) Systemet hopper over steg 8. Systemet lar altså være å sende mail til administrator og kandidat om nominasjonen


Velger
:EvigElev
AngiValg(valgtypeID)
AngiKandidatInformasjon(kandidat_ID, navn, e-mail)
SendNominasjon()
ForhåndsvisningNominasjon
GodkjennNominasjon()
BekreftelseNominasjon

USE CASE: Administrere valgtyper

High-level

Aktør:	Administrator
Type:	Primær
Beskrivelse:	Administrator skal kunne opprette en ny valgtype, legge inn en passende valgbeskrivelse, overskrift og valgregler, angi eventuell stemmevekting for forskjellige grupper stemmeberettigede. Det vil være mulig å legge inn de forskjellige typer verv som hører inn under en valgtype. Videre vil det være mulighet til å endre eller slette valgtyper som er lagt inn.

USE CASE: Beregne valgresultat

High-level

Aktør:	Administrator
Type:	Primær
Beskrivelse:	Etter endt avstemming for valg uten vektingsordning, må administrator aktivere opptelling av avgitte stemmer. Systemet vil da telle opp stemmene for de ulike kandidatene. På bakgrunn av stemmeopptellingsreglementet vil systemet generere et korrekt opptellingsresultat som vil bli presentert for administrator.

USE CASE: Beregne vektet valgresultat

High-level

Aktør:	Administrator
Type:	Primær
Beskrivelse:	Etter endt avstemming for et vektet valg, må administrator aktivere opptelling av avgitte stemmer. Systemet vil da telle opp stemmene for de ulike kandidatene og beregne hvor mye de ulike stemmene teller. På bakgrunn av disse beregningene vil systemet generere et korrekt opptellingsresultat som vil bli presentert administrator.

USE CASE: Godkjenne valgresultat

High-level

Aktør:	Administrator
Type:	Primær
Beskrivelse:	Etter at administrator har hentet og skrevet ut opptellingsresultatet, må dette resultatet godkjennes av valgstyret. Det kan for eksempel være at det i ettertid oppdages at en eller flere kandidater ikke er akseptert som kandidat, eller at det aktuelle valget skal kjønnskvoteres. I slike tilfeller må resultatrekkefølgen forandres før valgresultatet offentliggjøres. Det må også fordeles verv til aktuelle kandidater før man har et endelig valgresultat. Når valgresultatet er godkjent av valgstyret, vil det bli godkjent i systemet og være tilgjengelig på ei dedisert internettside.

USE CASE: Legge inn forhåndsstemmer

High-level

Aktør:	Administrator
Type:	Primær
Beskrivelse:	Når det kommer inn forhåndsstemmer på papir fra folk som vil stemme før avstemmingsperioden eller fra folk som er bortreist, må disse også registreres i systemet. Administratoren skal da kunne gå inn og mate disse stemmesedlene inn i systemet sammen med brukernavnet til de som avla forhåndsstemmen, slik at også disse stemmene blir med i stemmeopptellingen.

USE CASE: Hente ut opptellingsrapport

High-level

Aktør:	Administrator
Type:	Primær
Beskrivelse:	På samme måte som valgresultatet kan hentes ut etter at et valg er gjennomført, opptalt og godkjent, vil det også være mulighet til å få se opptellingsrapporten fra valgopptellingen. Når opptellingen foregår vil data bli lagret i databasen underveis, slik at det kan lages en rapport som viser alle interessante tall fra opptellingen. Rapporten vil presenteres i en Java Servlet-side, som administratoren vil ha tilgjengelig i sin modul.

USE CASE: Vedlikeholde data

High-level

Aktør:	Administrator
Type:	Primær
Beskrivelse:	Etter hvert som tiden går vil databasen øke i omfang. Derfor kan administratoren slette stemmesedler når disse ikke lenger må oppbevares, og valg som ikke lenger er interessante å ta vare på. Når et valg blir slettet vil alle data knyttet til det valget bli slettet fra databasen.

USE CASE: Laste inn manntallsliste

High-level

Aktør:	Administrator
Type:	Primær
Beskrivelse:	For å konfigurere et valg må administratoren hente inn korrekte manntallslister til systemet. Disse kan hentes fra filer med mulighet for å legge til eller fjerne velgere fra listen. Når listene er klare og har blitt innlastet, kan man kan holde rede på hvem som har stemmerett til det aktuelle valget og hvem som allerede har avgitt sin stemme.

USE CASE: Varsle om nominasjon

High-level

Aktør:	Administrator
Type:	Primær
Beskrivelse:	Hvis en kandidat har blitt nominert til et valg fem ganger, skal systemet sende ut en beskjed til kandidaten, hvor det blir spurt etter diverse informasjon, og eventuelt om vedkommende ønsker å stille til valg dersom han/hun er student som er nominert. Er en ansatt nominert, vil han ikke ha mulighet til å reservere seg fra kandidaturet. Beskjeden til kandidaten vil automatisk bli utsendt som e-mail. Systemet skal også sende ut en e-mail til hovedadministrator med informasjon om at den aktuelle kandidaten er blitt nominert fem ganger.

USE CASE: Logge inn

High-level

Aktør:	Velger, administrator, nominator
Type:	Primær
Beskrivelse:	Både velger, nominator og administrator må av sikkerhetsmessige hensyn skrive inn brukernavn og passord ved pålogging på systemet. Administratorens brukernavn og passord vil sjekkes mot en tabell i MS SQL Server-databasen, mens velgerens og nominatorens brukernavn og passord vil bli sjekket opp mot NT-domene ved høgskolen.

USE CASE: Registrere kandidater

High-level

Aktør:	Administrator
Type:	Primær
Beskrivelse:	Før avstemmingen kan starte må administrator på bakgrunn av hvilke ansatte som er nominert, og eventuelle svarmailer fra studenter som er nominert, sammen med valgstyret gjøre en utvelgelse av hvem som er aktuelle for å stille som kandidater til valg. Administratoren må legge inn kandidatene sammen med relevant informasjon om hver av dem.

USE CASE: Hente ut opptellingsresultat

High-level

Aktør:	Administrator
Type:	Primær
Beskrivelse:	Etter endt valg og etter at stemmeopptellingen er fullført kan administrator hente og skrive ut opptellingsresultatet som en rangert liste over de aktuelle kandidater, og hvilke verv som skal fordeles på dem. Dette vil være et uoffisielt opptellingsresultat som kun administrator har tilgang til, og som først blir offisielt når administrator i samråd med valgstyret har godkjent resultatet.

USE CASE: Hente ut valgresultat

High-level

Aktør:	Velger, Administrator
Type:	Primær
Beskrivelse:	Etter at valget er gjennomført og godkjent, vil systemet lagre det endelige valgresultatet i databasen. Dette resultatet vil kunne hentes ut i en Java Servlets-side, som skal være tilgjengelig for både velgere og administrator. Siden vil vise hver kandidat og hvilket verv den enkelte er valgt til.

USE CASE: Hente ut stemmesedler

High-level

Aktør:	Administrator
Type:	Primær
Beskrivelse:	I tilfelle resultatet av et valg må etterprøves, eller telles opp manuelt grunnet klager på stemmeopptellingen, må administrator kunne gå inn i systemet og hente ut alle stemmesedlene som er avgitt ved det aktuelle valget. Administratoren vil også få mulighet til å skrive ut disse, slik at man enklere kan kontrolltelle manuelt.

USE CASE: Endre valgkonfigurasjon

High-level

Aktør:	Administrator
Type:	Primær
Beskrivelse:	Etter at et valg er konfigurert vil administrator ha mulighet til å endre datoer for perioden hvor det er mulig for brukerne å nominere kandidater, svarfristdato for nominerte kandidater, når det er mulig for stemmeberettigede å avgi stemmer, og datoer for perioden når selve vervet gjelder. På denne måten kan et valg forskyves eller man kan forandre tidsrammene for et valg.

USE CASE: Foreta administratorinnstillinger

High-level

Aktør:	Administrator
Type:	Primær
Beskrivelse:	Når det blir forandringer i hvilke administratorer som skal administrere systemet, kan administratorene legge til nye administratorer med relevante opplysninger, endre sine egne innstillinger og passord, og slette administratorer som ikke lenger skal ha administratortilgang. Det skal ikke være mulig å slette seg selv eller hovedadministratoren, siden systemet må ha en hovedadministrator til enhver tid.

USE CASE: Administrere velgergrupper

High-level

Aktør:	Administrator
Type:	Primær
Beskrivelse:	Når det blir opprettet nye grupper av stemmeberettigede eller når noen av de eksisterende blir forandret eller fjernet, skal administratoren kunne gå inn og gjøre endringer her. Han skal da kunne opprette nye, slette eller endre velgergrupper.

USE CASE: Avlyse valg

High-level

Aktør:	Administrator
Type:	Primær
Beskrivelse:	Etter at et valg er konfigurert og publisert, kan det oppstå forskjellige situasjoner som gjør at valget må avlyses. Dette kan for eksempel være mangel på kandidater, kandidater som har trukket seg, eller det kan være at et valg har blitt feilaktig konfigurert. Det vil da være nødvendig for administrator å kunne trekke tilbake det aktuelle valget. Dette må gjøres før perioden for å avgi stemmer har begynt. Da vil det være for sent å avlyse valget.

USE CASE: Oppdatere avanserte innstillinger

High-level

Aktør:	Administrator
Type:	Primær
Beskrivelse:	For at systemet skal være fullt brukbart ved eventuelle forandringer gjort av IT-tjenesten i fremtiden, har administratoren mulighet til å kunne endre på noen innstillinger for systemet. Disse innstillingene er hvilken mailserver og hvilke internettsider som brukes for publisering av valgresultater og valgrapporter som skal brukes. I tillegg kan administratoren legge til nye og slette eller endre eksisterende domener for pålogging mot NT-domenet, slik at velgere og nominatorer får logget seg på systemet hvis det blir endringer på dette området.

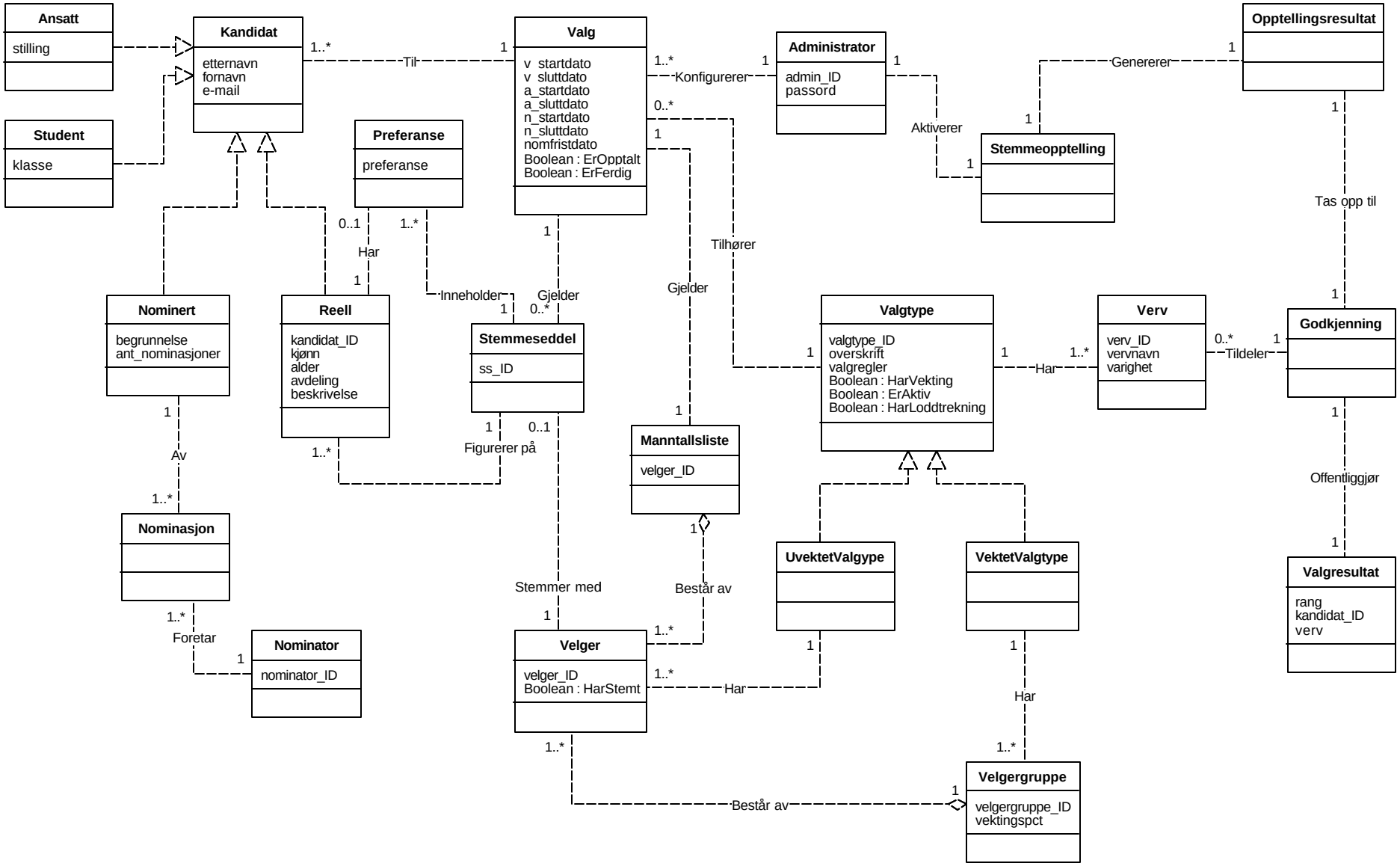
3.4 Konseptuelt klassediagram

Kommentar til klassediagrammet

Klassediagrammet gir et konseptuelt bilde av systemet og er med på å klargjøre gangen i det utviklede produktet. Fra use caset 'Konfigurere valg' og 'Administrere valgtyper' kommer klassene valgtype, verv, manntallsliste og valg. Administrator konfigurerer et valg av ønsket valgtype, for eksempel rektorvalg, og laster inn manntallslister bestående av velgere med stemmerett til aktuelt valg. En valgtype kan enten være en uvektet valgtype eller en valgtype med vektingsordning, derav generaliseringsstrukturen. Med bruk av en slik struktur får man modellert betydningen av velgergruppene, som kun har reell verdi ved vektede valg, da hver velgergruppe har en egen vektingsprosent som sier noe om hvor mye de enkelte velgernes stemmer teller.

Ansatte og studenter ved høgskolen kan nominere kandidater og har da rollen som nominator. Klassene 'Nominator' og 'Nominasjon' kommer fra use caset 'Nominere kandidat'. Etter at nominasjonsperioden er over vil administrator registrere det antall kandidater som kreves for å fylle valgets verv, eventuelt legge til flere dersom det fortsatt skulle finnes nominerte kandidater med tilstrekkelig mange nominasjoner. Dette fører til en generaliseringsstruktur for kandidat; nominert kandidat er en foreslått kandidat under nominasjonsperioden, mens en reell kandidat er en kandidat som er blitt godkjent av valgstyret og dermed offisielt kan stille til valg. En kandidat kan enten være en ansatt eller en student. Under valgets avstemmingsperiode kan stemmeberettigede fylle ut stemmeseddelen ved å gi preferanser til en eller flere kandidater etter eget ønske. Klassene 'Stemmeseddel' og 'Preferanse' stammer fra use caset 'Avgi stemme'.

Når avstemmingsperioden er over vil administrator aktivere stemmeopptellingen som resulterer i et opptellingsresultat. Klassene 'Stemmeopptelling' og 'Opptellingsresultat' berører dette området og er inspirert fra use casene 'Beregne valgresultat' og 'Beregne vektet valgresultat'. Administrator, eventuelt i samråd med en eller flere representanter fra valgstyret, tar opptellingsresultatet opp til godkjenning, hvor eventuell kjønnskvotering utføres og tilhørende verv fordeles på kandidatene, og man får et offisielt valgresultat som er tilgjengelig for alle.



Figur 3.2 Konseptuelt klassediagram

3.5 Supplementsspesifikasjon

Inneholder diverse krav som ikke inngår i use casene eller som krever forklaring utover det use casene gir.

3.5.1 Funksjonalitet

Sikkerhet
Avstemmingsdelen av systemet krever at brukeren logger på med riktig passord for å få avgi stemme. Velgeren kan kun avlegge én stemme for hvert valg han/hun er stemmeberettiget til. Den administrative delen av systemet krever at systemets administrator logger på med gyldig brukernavn og passord.

Utvidet funksjonalitet
Se punkt 7.2.

3.5.2 Menneskelige krav

Systemet setter visse krav til administrator, som vil trenge en viss opplæring i å bruke systemet, siden denne delen av systemet kan være litt omfattende da man må gjennom flere steg for å komme i mål. Dette kan gjøres ved hjelp av en lettfattelig manual, hjelpemenyer i systemet og demonstrasjoner. Brukerne av systemet, altså velgere og nominatorer, vil ikke trenge særskilt opplæring i bruk av systemet siden brukergrensesnittet for å avgi stemme er såpass lettfattelig og intuitivt som det er. Vi satte ned en brukergruppe for å teste vårt utkast til GUI, og dette testpanelet hadde ingen problemer med å bruke verken administrator- eller avstemmingsmodulen enkelt på en fullverdig måte.

3.5.3 Pålitelighet

Datalagring
Krav til oppbevaringstid av stemmesedler etter et valg er en uke med det manuelle systemet. Med det elektroniske systemet bør stemmesedlene lagres minst like lenge, i tilfelle klager på valgresultatet. Etter denne perioden vil det være mulig å fjerne stemmesedlene fra systemet. Rapporter og valgresultater vil måtte lagres over en lengre periode enn dette.

Transaksjonshåndtering
Databasen og applikasjonen må kunne sørge for at ikke samme person kan avgi stemme mer enn én gang. Videre må databasen og applikasjonen sammen håndtere problematikken med at to personer kan stemme på nøyaktig samme tidspunkt. Da vil begge disse transaksjonene jobbe mot samme rad i databasen samtidig. Her må det implementeres transaksjoner i applikasjonen, slik at ikke feilaktige eller ufullstendige data lagres.

3.5.4 Ytelse

Det vil være spesielt i brukerens interesse at avstemmingen foregår så raskt og enkelt som mulig. Man bør også kunne forlange en stabil oppkobling mellom klient og database, slik at man ikke opplever problemer med å få kontakt med database når stemme skal avgis. Det er også frustrerende for velgeren å oppleve dette, og det svekker dennes tillit til systemet. Man må også stille krav vil at flere brukere kan aksessere applikasjonen på samme tidspunkt.

3.5.5 Implementasjonsbegrensninger

Prosjektdeltakerne planlegger bruk av Java for utvikling av systemet, med JDBC mot databasen. Videre skal det brukes Java Servlets for presentasjon av diverse data på internett. Disse Servlet-sidene vil også jobbe mot databasen. Til utvikling av GUI for testpanel og administrator som skal teste resultatet, vil Delphi bli brukt, siden dette er et mye raskere og enklere verktøy å sette opp GUI i enn Java. Som database skal Microsoft SQL Server benyttes, etter samråd med IT-tjenesten ved høgskolen. Vi ser ikke for oss at dette vil legge noen begrensninger på løsningen.

3.5.6 Grensesnitt

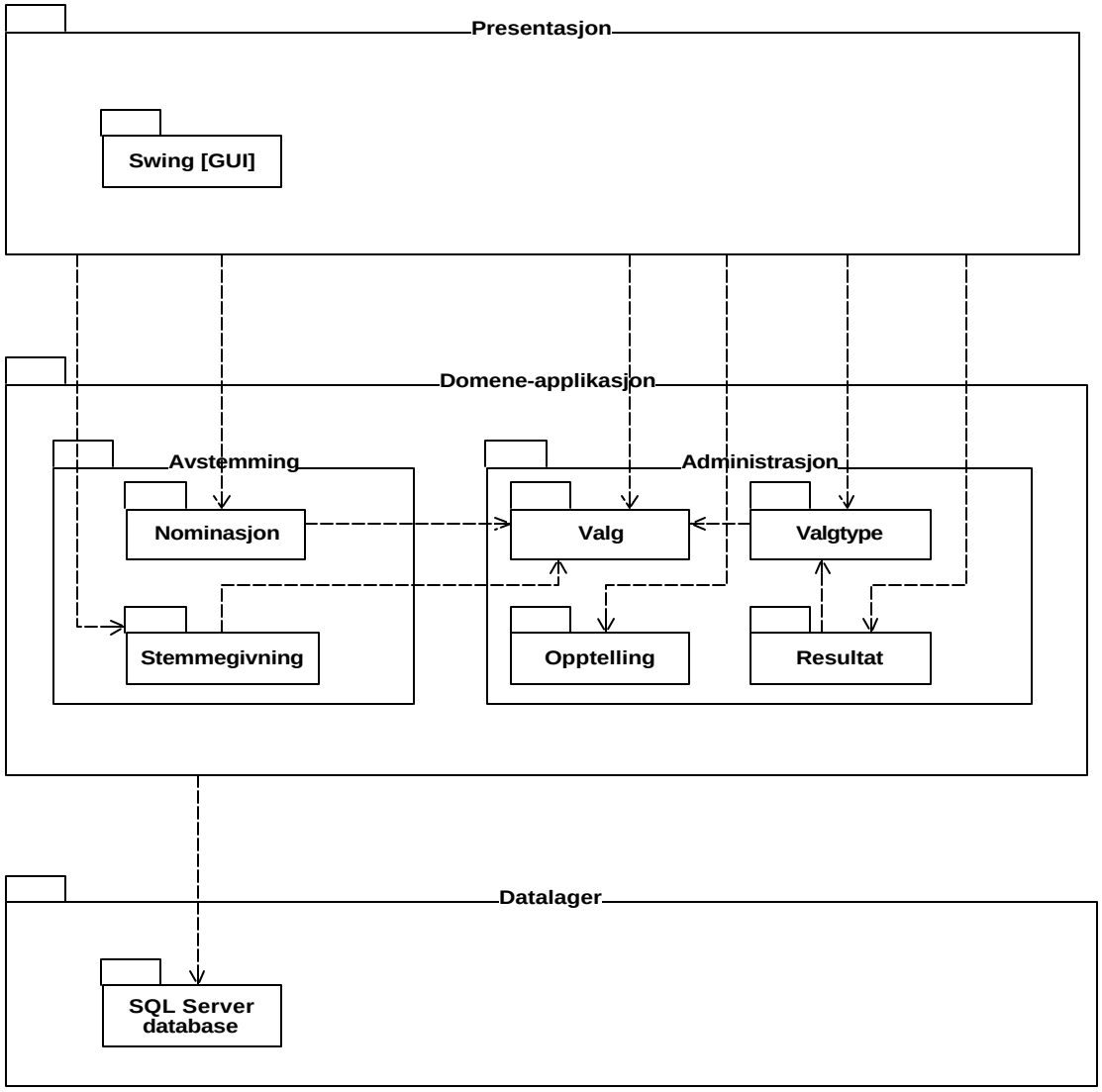
Vi ser for oss en sentralisert løsning, der avstemmingsmodulen ligger på en vilkårlig partisjon på en av høgskolens servere, hvor den kan aksesseres via en snarvei fra alle arbeidsstasjonene ved HiGs nettverk. Det vil da være mulig for stemmeberettigede velgere å gå inn og avgi stemme ved valgene. Administrasjonsmodulen skal legges på de stasjonære PC-ene til administratorene, siden det kun er administratorene som skal benytte denne delen av systemet. For beskrivelse av brukergrensesnittet henvises til punkt 4.5.

4 Design

Designdokumentet i sin helhet vil finnes som vedlegg på CD. Årsaken til at vi har valgt å legge det på CD i stedet for som papirvedlegg er at mye av informasjonen som står i kapittel 4 i hovedprosjektrapporten finnes i designdokumentet. I tillegg vil designdokumentet inneholde ytterligere eksempler på sekvensdiagrammer og funksjonalitetsbeskrivelser for hele applikasjonen. Designdokumentet vil være systemdokumentasjonen for det utviklede produktet.

4.1 Systemarkitektur

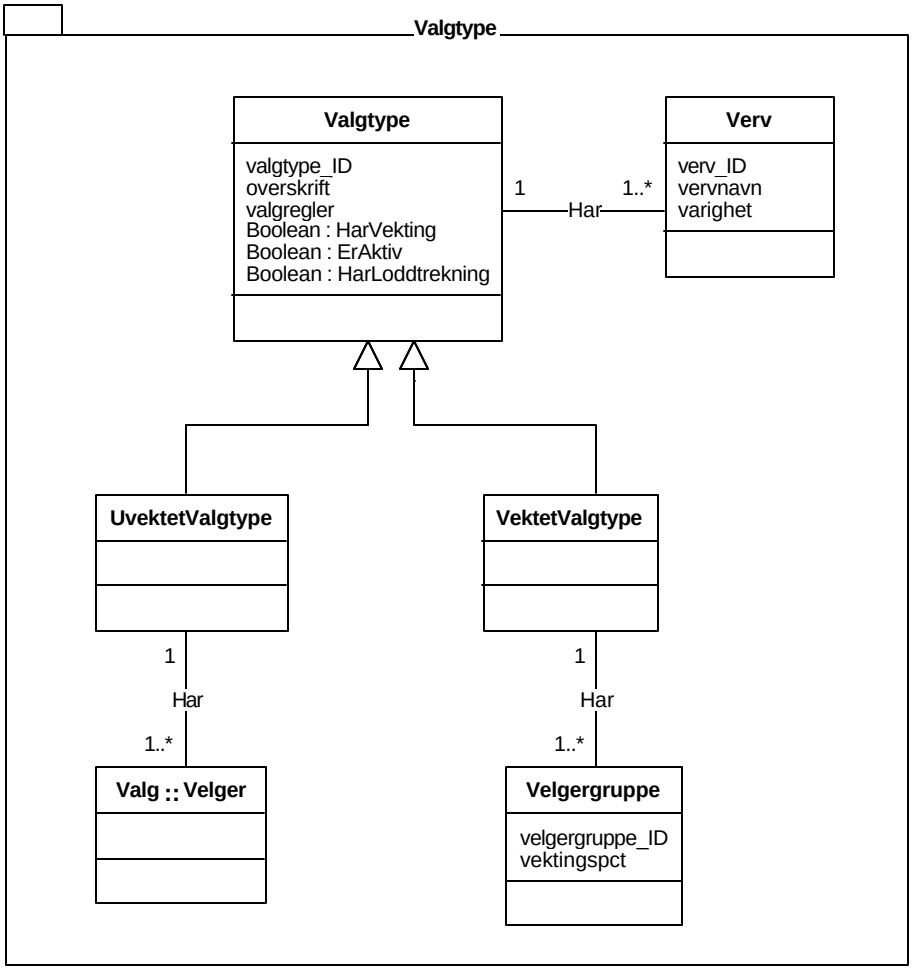
Når man skal utvikle et elektronisk valgsystem vil det være mye informasjon som skal lagres, både i forbindelse med valg som pågår og fra avholdte valg. Derfor var det en forutsetning at det ble brukt en database. Med JDBC (Java Database Connectivity) og SQL som spørrespråk blir mulighetene for uthenting, oppdatering og skriving til database gode.



Figur 4.1 Logical View

Vår valgapplikasjon bygger på en tre-lags struktur med et lag for presentasjon, et lag for domene og applikasjon og et lag for datalager (figur 4.1). Presentasjonslaget vil inneholde brukergrensesnittet som består av en rekke skjermbilder programmert i Java. På domene- og applikasjonslaget vil selve applikasjonen ligge, fordelt på to moduler: Avstemmingsmodulen og administrasjonsmodulen. Disse modulene inneholder igjen pakker. Under avstemming finnes pakkene 'Nominasjon' og 'Stemmegivning', som i sin tur inneholder klasser som deltar i use casene 'Nominere kandidat' og 'Avgi stemme'.

Administrasjonspakken inneholder pakkene 'Valgtype', 'Valg', 'Opptelling' og 'Resultat'. Hver av disse pakkene inneholder klasser som stammer fra henholdsvis valgkonfigurasjons-, opptellings- og godkjenningsproblematikk. Figur 4.2 viser valgtypepakkens innhold. Hver pakkes klasser definerer områder som er konseptuelt relaterte og dermed har naturlig tilknytning; eksempelvis er klassene 'Verv', 'Velgergruppe' og 'Valgtype' knyttet til use caset 'Administrere valgtyper'.



Figur 4.2 Klasseinndeling i pakke "Valgtype"

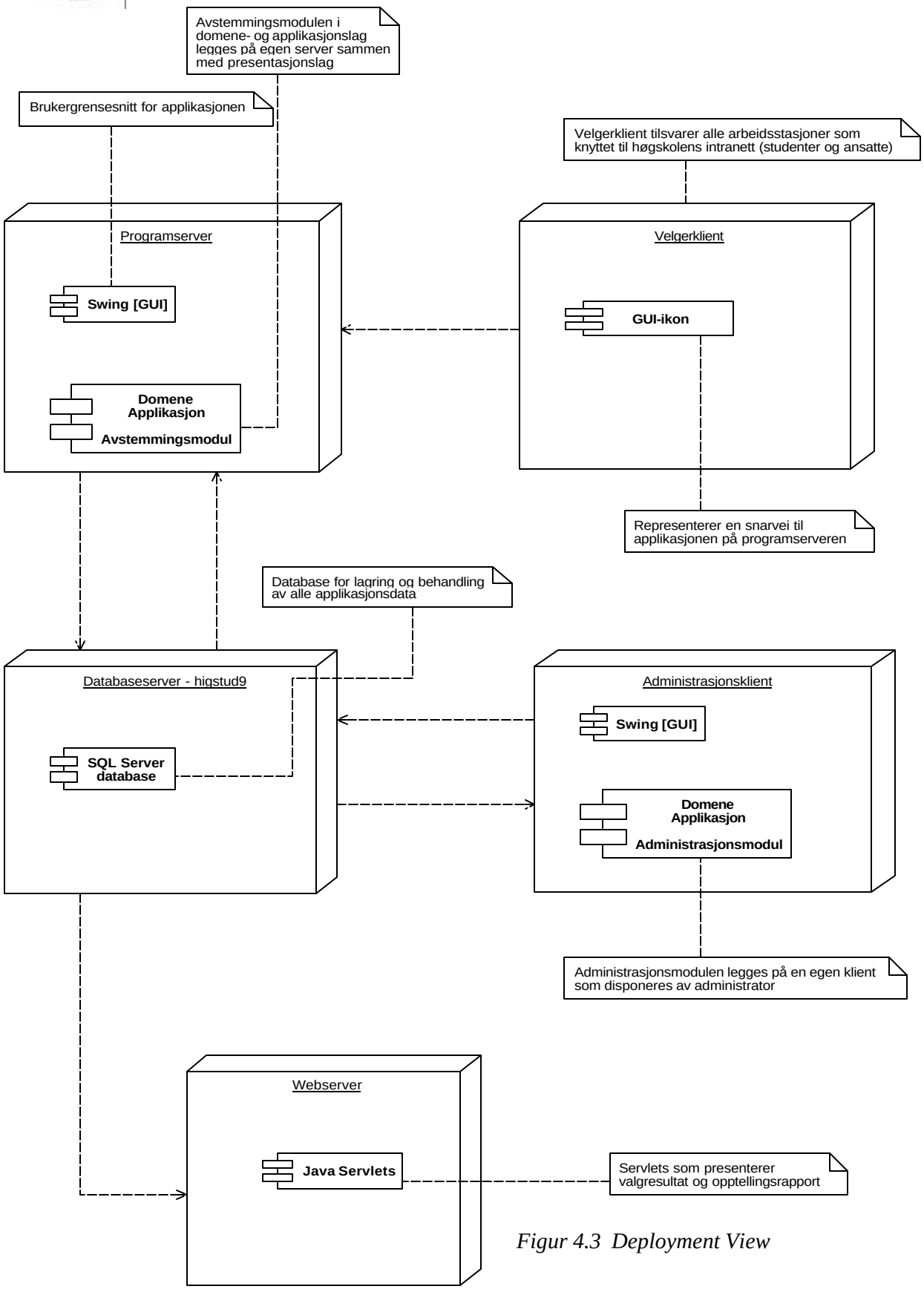
Enkelte av klassene, som for eksempel ’Kandidat’, ’Velger’ og ’Valg’, finnes i flere pakker. Dette er illustrert med notasjonen ’Pakkenavn :: Klassenavn’.

Datalager inneholder en SQL Server-database som er en relasjonsdatabase. Valget av DBMS (Database Management System) ble gjort i samråd med IT-tjenesten, som hadde et ønske om at SQL Server ble brukt, siden flere hovedprosjekter ved høgskolen skal benytte denne.

Som man ser av figur 4.3 har vi valgt å legge brukergrensesnittet for avstemmingsmodulen sammen med applikasjonslaget for samme modul på en egen server. Dette ble gjort med tanke på at IT-tjenesten skulle slippe å installere modulen på hver eneste arbeidsstasjon på skolen. Dessuten har vi forhørt oss med ressurspersoner med Java-kompetanse som mener at en Java-applikasjon skal fungere helt greit når flere klienter aksesserer applikasjonen på en server. Fra denne programserveren vil applikasjonen kommunisere med SQL Server-databasen på higstud9. Brukere av avstemmingsdelen av systemet vil starte applikasjonen ved hjelp av et ikon på skrivebordet på arbeidsstasjonen hvor de er pålogget. Dette ikonet representerer en snarvei til programmet på programserveren.

Administrasjonsmodulen vil ligge på administratorens arbeidsstasjon og kommunisere med SQL Server-databasen på higstud9. Årsaken til at vi har valgt å legge administrasjonsmodulen på klient i stedet for på en server, er av sikkerhetsmessige hensyn. Siden det vil være snakk om et fåtall administratorer, mest sannsynlig en eller to, er det lite å tjene på å legge modulen på en egen server på samme måte som avstemmingsmodulen. Med tanke på at administrasjonsmodulen inneholder store konfigurasjonsmuligheter og mulighet for å legge inn forhåndsstemmer fra andre, er det ønskelig at ingen får mulighet til å prøve å logge seg på som administrator bortsett fra registrerte administratorer. Vi mener at vår løsning er med på å ivareta sikkerheten i større grad enn dersom denne hadde ligget på samme programserver som resten av applikasjonen. Mer om denne problematikken i punkt 7.1.2

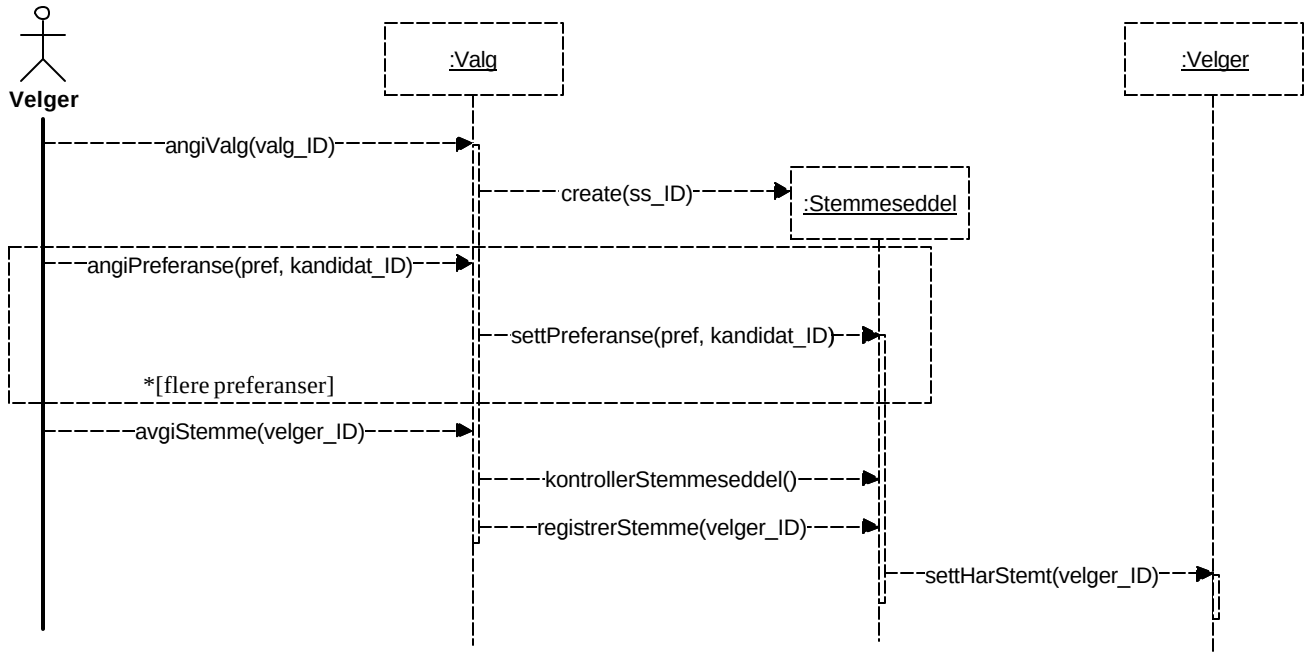
Java Servlets som henter ut informasjon fra databasen vedrørende valgresultater og opptellingsrapporter og presenterer dette, vil ligge på en av høgskolens webservere. For mer informasjon om hvorfor vi har valgt å legge disse delene av systemet på nett, refereres det til 7.1.2.



Figur 4.3 Deployment View

4.2 Designmodell

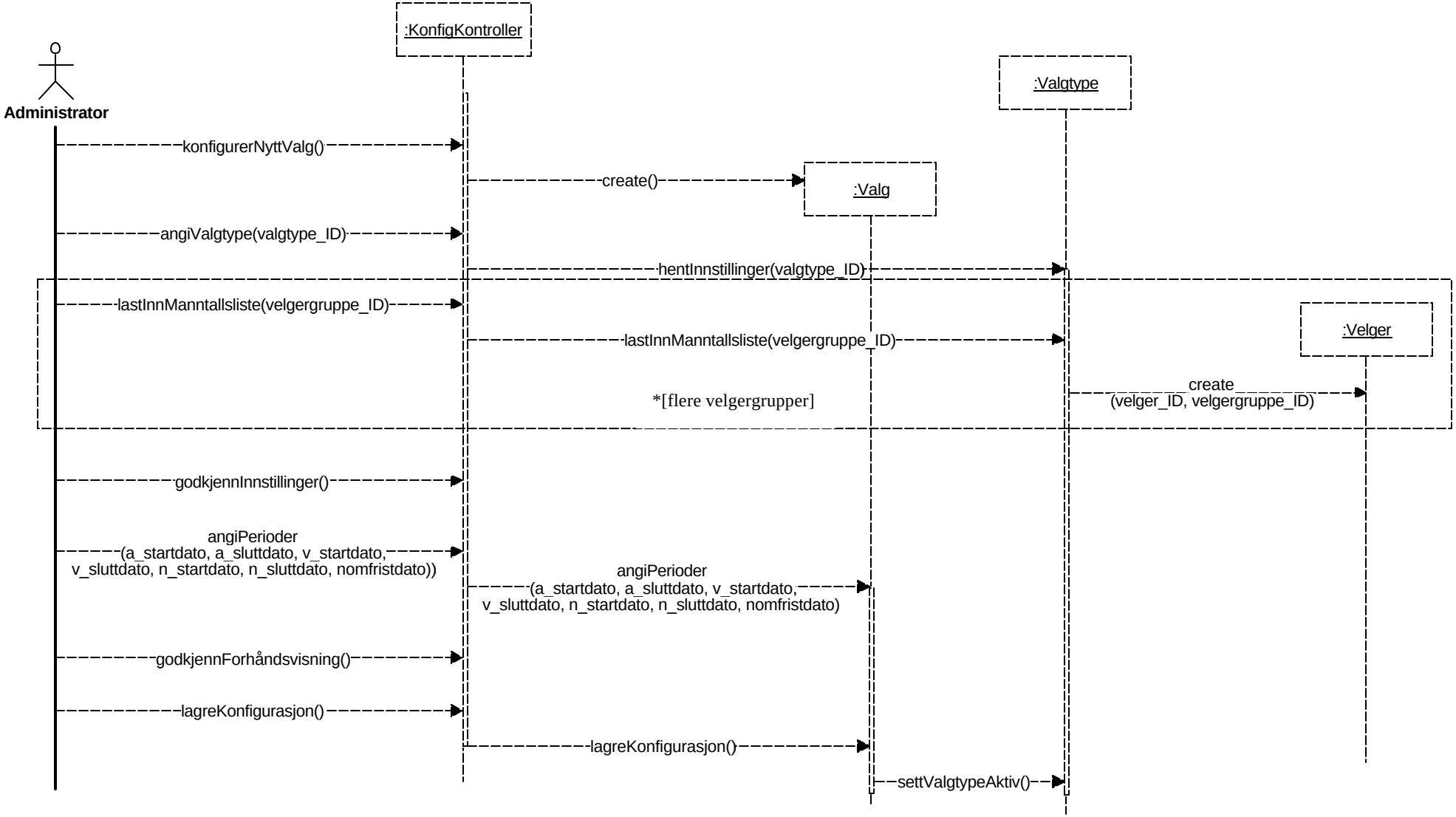
For use caset 'Avgi stemme' er 'Valg' valgt som kontrollerklasse.
Velgeren angir valget han ønsker å stemme ved, og et stemmeseddelobjekt opprettes.
Velgeren setter ønsket preferanse på kandidatene og avgir stemmen. Stemmeseddelobjektet kontrollerer deretter om stemmeseddelen er korrekt utfylt og registrerer deretter stemmen.
Velgerobjektet setter at velgeren har avlagt stemme. Se figur 4.4.



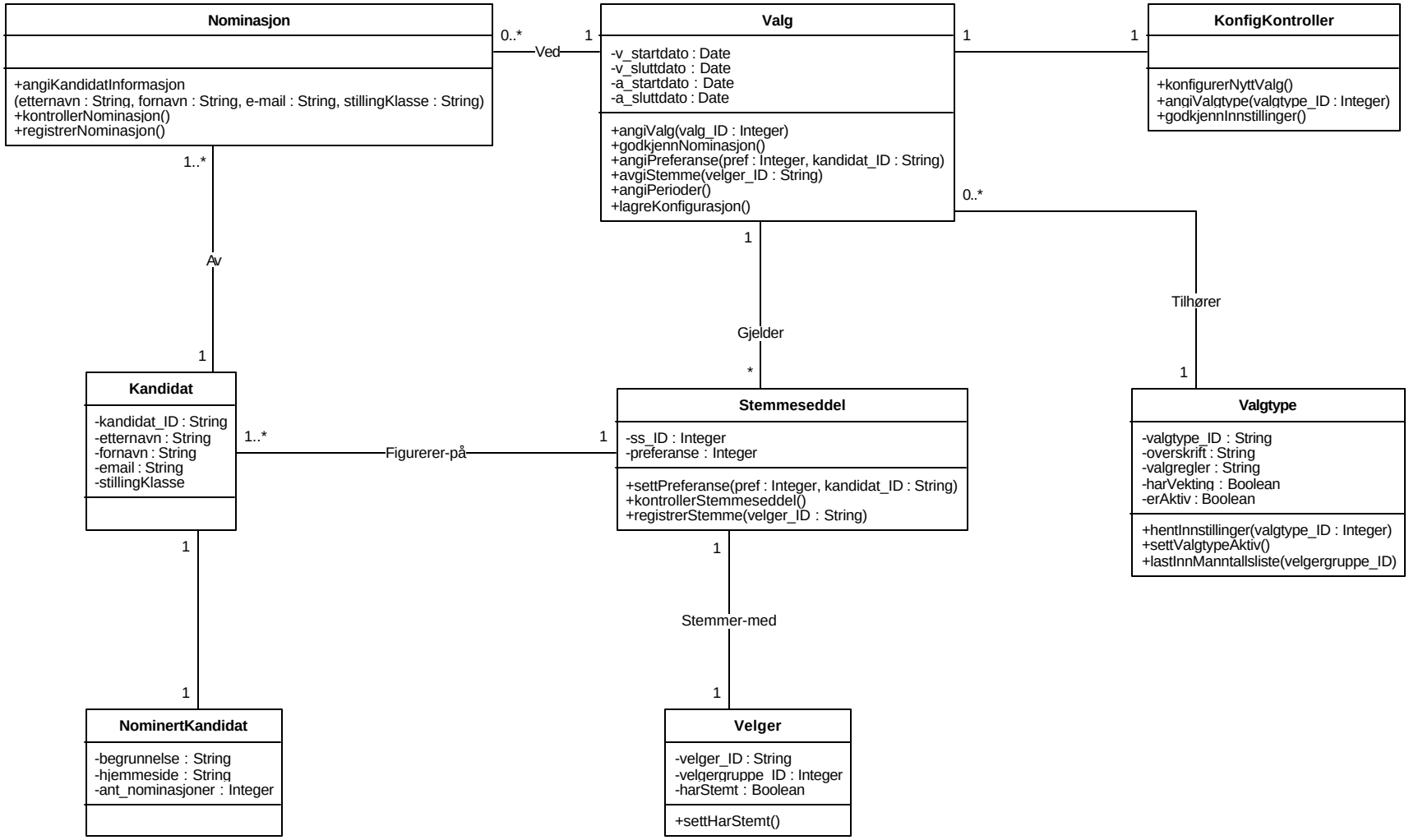
Figur 4.4 Sekvensdiagram "Avgi Stemme"

Ved konfigurasjon av valg har vi valgt å lage en egen kontrollerklasse, 'KonfigKontroller', som har ansvaret for å motta og håndtere innkommende systemhendelser. Begrunnelsen for dette er at vi ønsker å unngå stor "belastning" på valgklassen, som ville hatt lavere modulstyrke og kanskje bidratt til høyere kobling dersom denne hadde blitt valgt til kontrollerklasse.

Administrator starter ny konfigurasjon og et valgobjekt blir opprettet. Deretter angir han hvilken type valg han ønsker å konfigurere, og et valgtypeobjekt henter velgergruppe-innstillinger for den valgte valgtypen. Administrator laster inn manntallsliste for velgergruppe, og det opprettes et velgerobjekt for hver velger som finnes i denne listen. Administrator fortsetter innlastingsprosessen for hver velgergruppe som er angitt i valgtypen. Når han er ferdig med manntallslistene angis verv, avstemmings- og nominasjonsperiode i tillegg til dato for nominasjonsfrist. I neste steg vises en forhåndsvisning av informasjon om dette valget som administrator godkjenner, og til slutt lagrer valgobjektet konfigurasjonen, mens valgtypeobjektet setter valgtypen til aktiv. Se figur 4.5



Figur 4.5 Sekvensdiagram"Konfigurere valg"



Figur 4.6 Design klassediagrammet viser softwareklasser fra use casene 'Avgi stemme', 'Konfigurere valg' og 'Nominere kandidat'.

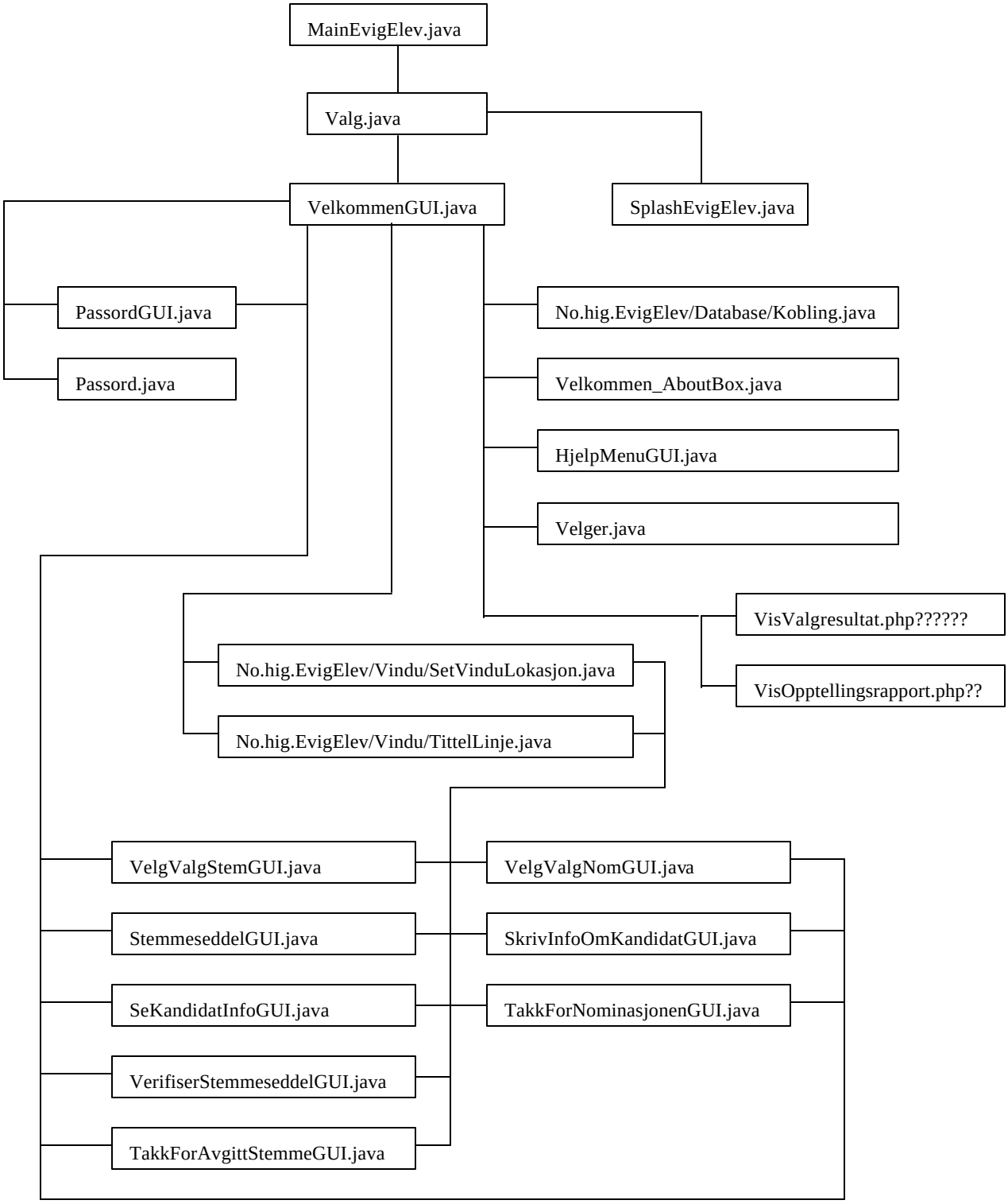
4.3 Navigasjonsstruktur

For å vise navigasjonsstrukturen mellom de forskjellige filene i systemet, har vi laget et navigasjonskart for hver av de to modulene. Disse navigasjonskartene skisserer hvilke filer som tilkaller hverandre.

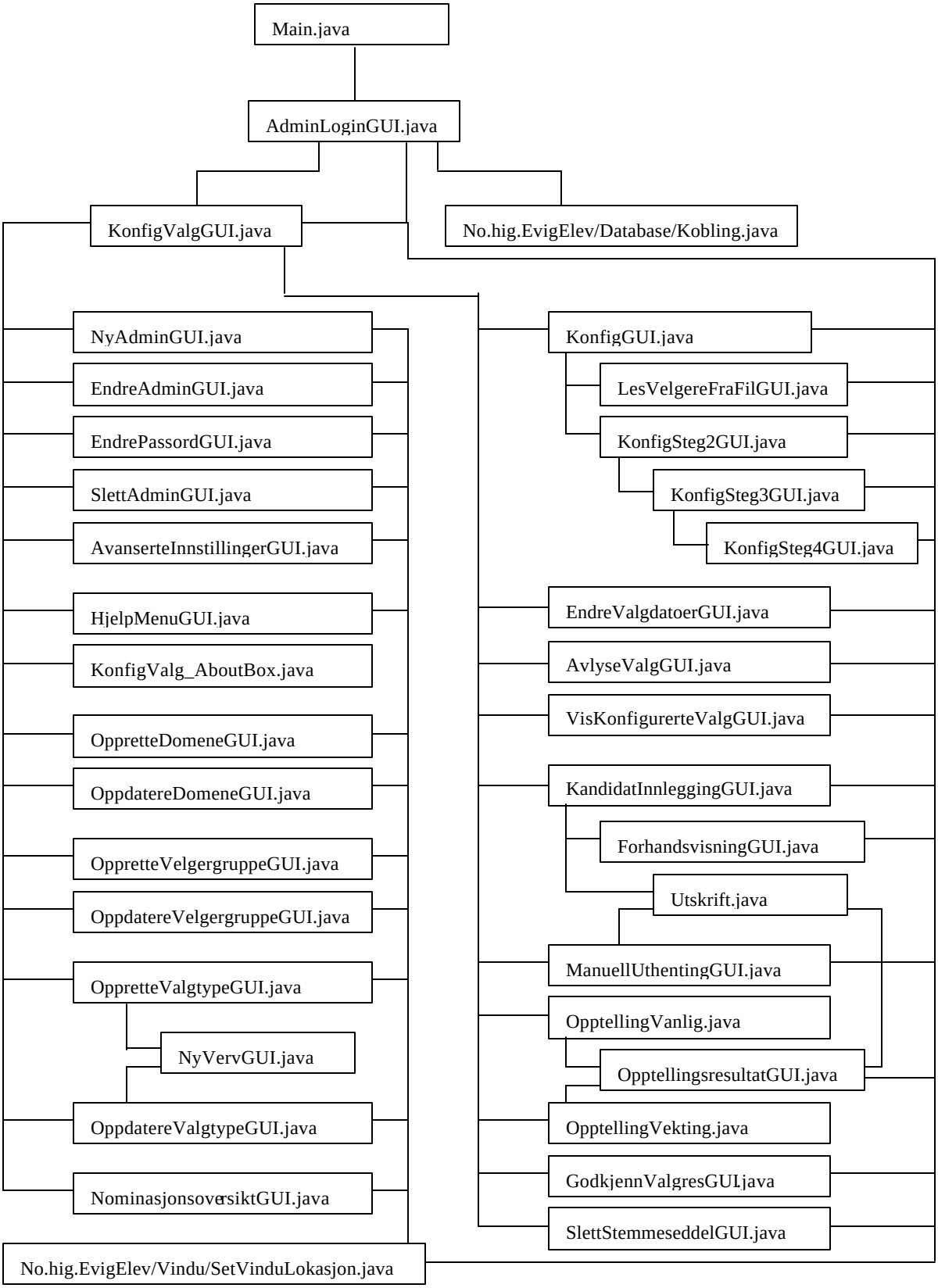
I avstemmingsmodulen vil alle filer med endelsen GUI.java kalle de to filene *"No.hig.EvigElev/Vindu/SetVinduLokasjon.java"* og *"No.hig.EvigElev/Vindu/TittelLinje.java"*, som sentrerer skjermbildet og setter tittellinjen for hver frame.

Når en bruker starter applikasjonen vil en innledende splash kjøres mens hovedmenyen (VelkommenGUI) lastes. Fra VelkommenGUI vil brukeren måtte logge på (PassordGUI.java) for å komme inn på nominere- og avstemmingsdelene av systemet. Den nettbaserte delen av systemet vil kunne aksesseres uten pålogging.

I administrasjonsmodulen må pålogging skje før man kommer inn i selve menyen. Herfra styres det meste som skjer i modulen. Alle GUI filer i modulen kaller filen *"No.hig.EvigElev/Vindu/SetVinduLokasjon.java"*.



Figur 4.7 Navigasjonskart for avstemmingsmodul



Figur 4.8 Navigasjonskart for administrasjonsmodul

4.4 Databasestruktur

4.4.1 Oppbygning og utvikling av datab asen

Systemet har en del betingelser og krav til databasemodellen som har gjort at vi ikke kan følge 3. normalform fullt ut under normaliseringen av databasen. Hovedpunktene her er å ivareta anonymitet og sikkerhet, samt forhindre dobbeltstemming. Disse punktene har vært retningsgivende under utformingen av databasemodellen og dens relasjoner, tabeller og attributter.

Blant annet vil en database normalisert på 3NF knytte velgerens VelgerID opp mot stemmeseddelen denne velgeren avgir. Dette vil utvilsomt komme i konflikt med systemets krav om at velgernes identitet skal være anonym. Og siden man videre ved vektede valg må vite hvor en stemmeseddel kommer fra men ikke kan knytte den opp mot velgeren, må stemmeseddelen derfor knyttes til denne velgerens velgergruppe i stedet.

Da 3NF ikke ville gitt en slik databasestruktur, har de forskjellige forutsetningene gjort at vi har sett oss nødt til å se delvis bort fra en slik normalisering av databasen. Databasen er derfor normalisert så langt det lar seg gjøre, og deretter modifisert og tilpasset valgsystemets krav og behov. Dette er gjort både for å unngå redundans, flerfunksjonalitet, og transitiv og partiell avhengighet i databasen, samt for å tilpasse valgsystemets lovmessige og systemmessige krav.

Statusattributter er lagt til i valg-, valgtype- og velger-tabellene for å sikre systemets krav til databasen. Et eksempel på dette er en attributt i velgertabellen som indikerer hvorvidt en stemmeberettiget har avgitt stemme eller ikke ved det aktuelle valget. På denne måten vil det forhindres at en velger får avgitt stemme ved et valg mer enn én gang.

Strukturen og tabellene med de tilhørende relasjoner og attributter i databasen er vist i ER-modell (figur 4.9). Denne databasemodellen er implementert i databasen for systemet.

En del steder er det valgt å bruke tall som identifikatorer for hvert enkelt tuppel i tabellen. Og enkelte av disse identifikatorene settes som autonummer. Et alternativ er å bruke en generator som genererer et løpenummer som øker for hvert tuppel som insertes i databasen. Vi har valgt å ikke bruke slike generatorer i prosjektet vårt, da slike autonummer i SQL Server 2000 kan settes i column-innstillingene for det aktuelle attributtet.

Ved uthenting av hvilke valg som til enhver tid er i hvilken periode, for eksempel hvilke valg som er til avstemming, brukes en funksjon som heter GetDate(). Denne funksjonen sammenligner dagens dato med datoinnstillingene for valget, og finner således ut om det er mulig å stemme ved valget på det gitte tidspunkt. Grunnen til at vi har valgt å bruke GetDate() er at den sjekker opp mot datoen på serveren hvor SQL Server ligger, og ikke mot klokka på hver enkelt klient. Det vil derfor ikke være mulig å ”jukse” seg til å kunne stemme utenfor tidsrammene ved å endre klokkeinnstillingene på den PC-en hvor man sitter.

Databasens struktur er nøye vurdert og bevisst valgt, både når det gjelder databasens tabeller, dens attributter og deres datatyper, men også når det gjelder hvorvidt vi har brukt triggere,

generatorer og lignende. Applikasjonen håndterer sjekk på at felter som ikke kan være NULL, fylles ut og settes steder hvor dette er nødvendig.

Et mulig problemområde for stemmegivningen er at to velgere forsøker å stemme akkurat samtidig, og at de henter ut MAX-verdien for StemmeseddelID og forsøker å legge inn sin stemmeseddel med samme stemmeseddelID (MAX+1). Vi vurderte flere løsninger for å løse denne problematikken. En mulighet hadde vært å utvikle en trigger som utførte en rollback dersom dette skulle skje. Vi valgte i stedet å kjøre både MAXverdi-uthentingen og insert av stemmeseddelen som én samlet transaksjon. Dette gjøres i applikasjonen og ikke i databasen. På denne måten vil hele transaksjonen avbrytes dersom en annen velger skulle benytte samme MAX-verdi og allerede har lagt inn sin stemmeseddel på dette MAX+1 nummeret. Selv om sannsynligheten for at en slik samtidighet skulle inntreffe er svært liten, er det noe vi har valgt å ta høyde for, fordi konsekvensene vil være alvorlige.

4.4.2 Databasemodell

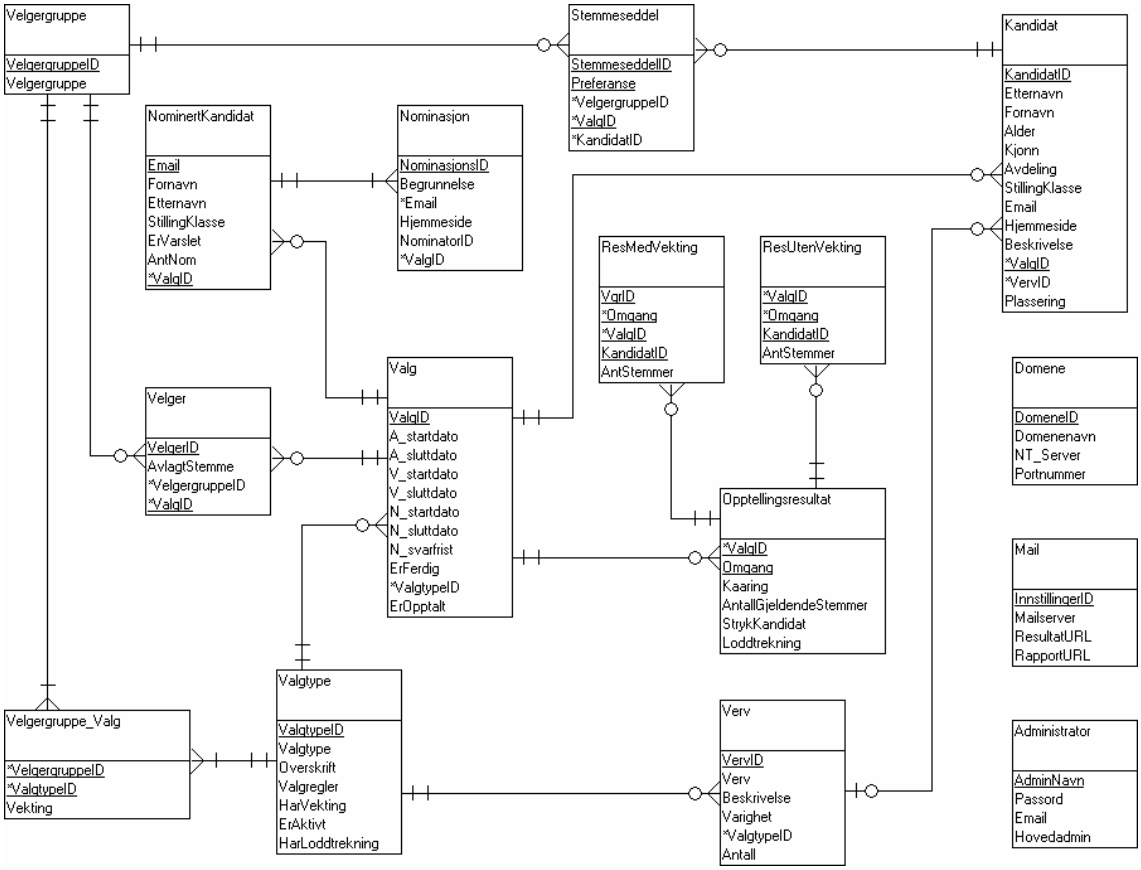


Fig. 4.9 ER-modell for databasen

4.4.3 Tabellforklaringer

Under utviklingen av databasen ble det tatt hensyn til at løsningen skal være generell, altså at nye typer valg skal kunne opprettes etter behov. Disse typene valg blir lagt i valgtype-tabellen, og hentes fram for hvert valg som skal avholdes.

Den helt sentrale tabellen i databasen er 'Valg', og en kandidat knytter sammen stemmeseddelen og det aktuelle valget. Stemmeseddel er videre knyttet mot velgergruppe for å ivareta anonymitet og samtidig vite stemmeseddelens vekting ved valg som har vektingsordning.

Under opptellingen av stemmesedlene for et valg, vil hver opptellingsrunde legges inn i Opptellingsresultat, med hver enkelt kandidat sin stemmefordeling i enten ResMedVekting eller ResUtenVekting.

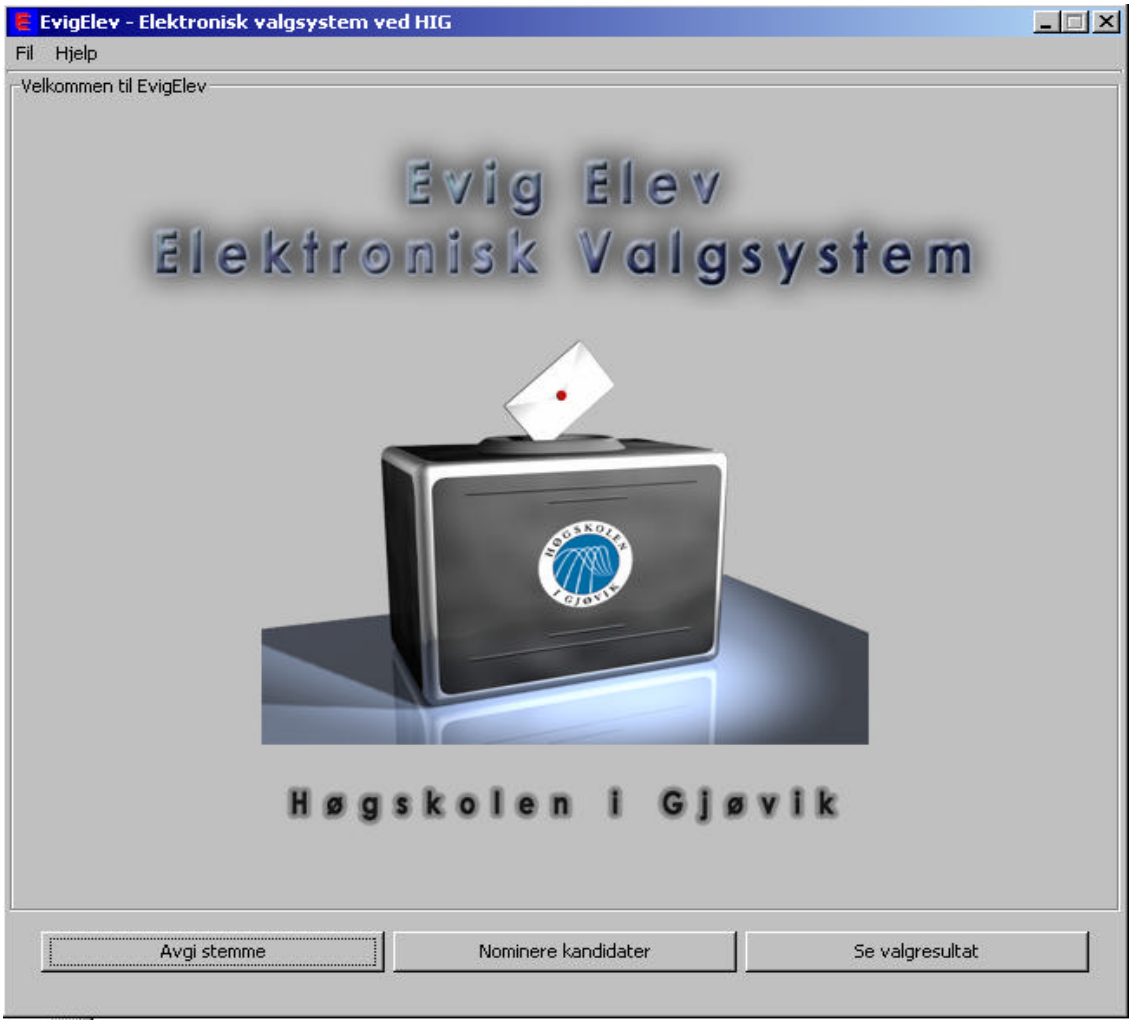
Detaljert tabell- og attributtbeskrivelse er lagt i vedlegg F.

4.5 Brukergrensesnitt

Brukergrensesnittet har hatt høy prioritet under utviklingen av dette systemet siden det er viktig med tanke på at uerfarne databrukere skal kunne benytte systemet uten særlig opplæring, eller ved hjelp av en brukermanual. Derfor har vi prøvd å dra nytte av lærdommen vi fikk av faget "Ergonomi i digitale medier", og følge de retningslinjer vi lærte der.

Brukergrensesnittet håndterer grafikk, slik som vinduer, knapper, tekstfelter, menyer og så videre, og er også ansvarlig for å oppdage og håndtere brukerens aktiviteter inne i systemet. Grensesnittet er vindusbasert, og bygd opp etter "point and click"-metoden. Dette betyr at brukeren kan peke på, og klikke seg fram i syste met.

Man kan gå inn på en funksjon i systemet ved å klikke på ønsket knapp, for eksempel "avgi stemme"-knappen (figur 4.10).



Figur 4.10 Hovedmeny avstemmingsmodul

4.5.1 Standarder

Generelt

Vi har hatt noen grunnleggende prinsipper i bunnen når vi har utviklet brukergrensesnittet i systemet slik at alle skjermbilder følger en standard. Systemet vil dermed få en bedre helhet og en "rød tråd" som går igjen i alle menyvalg og skjermbilder. Dette er viktig siden alle prosjektdeltakerne har deltatt i programmeringen. Systemet vil på denne måten bli enklere og raskere å sette seg inn i.

Brukergrensesnittet skal sørge for at brukeren har minimale muligheter til å gjøre feil. I tillegg har vi også laget hjelpemenyer for alle skjermbilder slik at brukeren raskt kan få hjelp hvis han trenger det. Alle steder hvor noe skal slettes fra databasen vil brukeren bli bedt om å bekrefte at han vil gjøre dette. Slik unngår man at brukeren sletter noe ved en feiltagelse. Alle skjermbilder har vanlig grå bakgrunn siden dette er naturlig og behagelig for øynene å se på.

Alle skjermbilder vil ha ei menylinje der det vil finnes mulighet til å lukke applikasjonen, få hjelp i hjelpemenyen, eller lese om produktet i "About"-boksen.
Til slutt kan det nevnes at alle skjermbilder er "non-resizable". Det vil si at størrelsen på skjermbildene er låst og dermed ikke er mulig å forandre.

Skjermoppløsning

Systemet og skjermbildene er designet med tanke på bruk i skjermoppløsning på 1024x768 piksler. I denne oppløsningen vil alle skjermbilder få naturlig og fin størrelse, og man vil ikke oppleve noen problemer med å få plass til skjermbildene på skjermen.

Meny

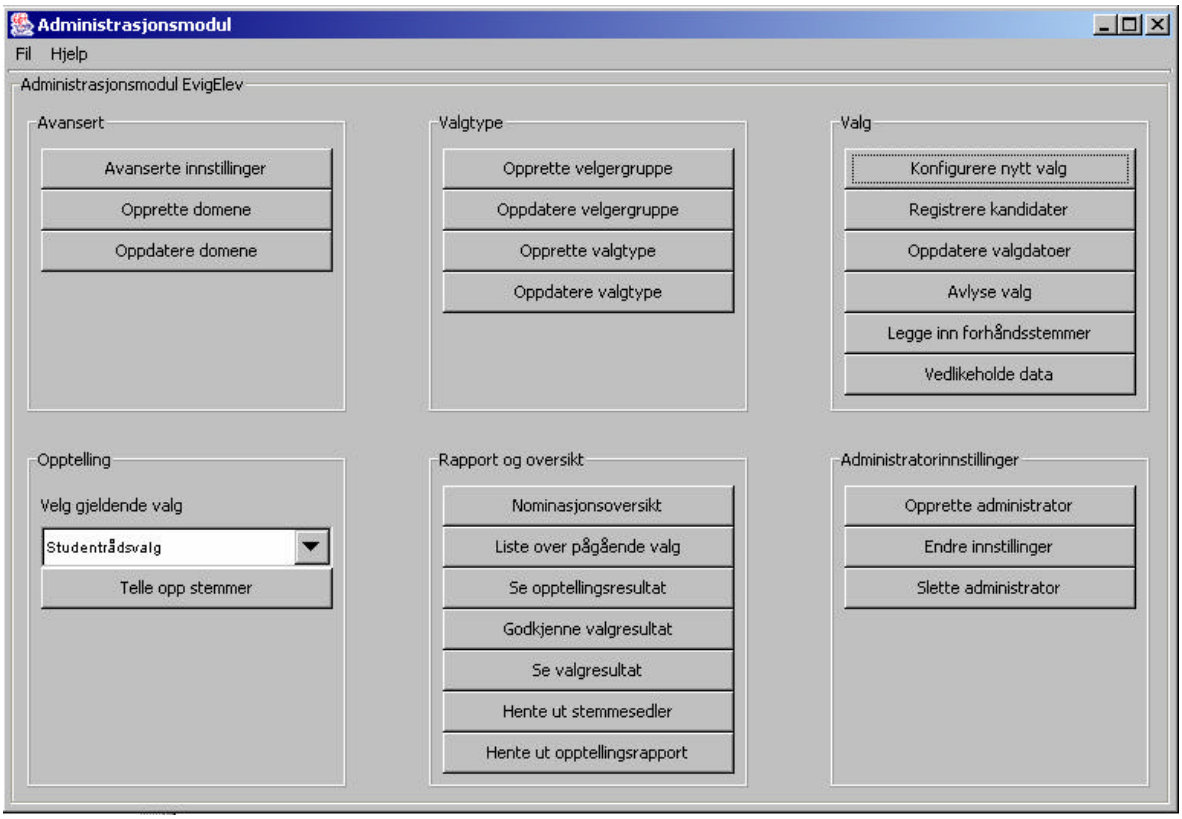
Fra brukerens ståsted er hovedmenyene selve hjertet i systemet. Dette er det første som møter brukeren når han starter programmet. Derfor er det ekstra viktig at menyene er oversiktlige og lett forståelige. Hvis ikke kan det lett ta fra brukeren motet og lysten til å bruke applikasjonen.

Det vil være to hovedmenyer i systemet, en for hver modul. Menyene vil være utgangspunkt for alle handlinger som skal kunne utføres i systemet, og brukeren vil kunne navigere seg herfra og til de menyvalgene han vil benytte. Menyene vil aldri forandres, og vil dermed være statiske.

Når man velger et menyvalg på hovedmenyen vil skjermbildet for det menyvalget han har valgt dukke opp på skjermen, mens hovedmenyen *ikke* vil vises i bakgrunnen bak det nye skjermbildet. Grunnen til at vi har valgt å lukke hovedmenyen når brukeren velger et valg er at det vil kunne virke forvirrende eller forstyrrende med skjermbildet av menyen i bakgrunnen av det man jobber med. Dessuten er det med på å sikre at brukeren lukker vinduene og dermed objektene før han går tilbake.

I menyen for administrasjonsmodulen (*figur 4.11*) har vi valgt å samle de menyvalgene som har mest med hverandre å gjøre i forskjellige bordere (rammer), slik at for eksempel alle rapporter ligger i en og samme border. I tillegg vil både borderne og menyvalgene innad i hver border være ordnet i kronologisk rekkefølge, det vil si i samme rekkefølge som de skal utføres når et valg skal avholdes. Dette er for å gjøre det mer intuitivt og lettere å finne fram, og dessuten skjønne hva han skal gjøre og hvilke menyvalg han skal velge til enhver tid.

Vi vurderte å lage knapper med undermenyer på administrasjonsmodulen, for eksempel en knapp som heter administrator, med valgene "administratorinnstillinger, slette administrator, og opprette ny administrator", og for eksempel en knapp som heter rapporter, med undermeny for "nominasjonsoversikt", "liste over pågående valg", "se opptellingsresultat", "se valgresultat", "hente ut stemmesedler" og så videre. Vi kunne ha fulgt denne prosedyren for nesten alle knappene på menyen, og således fått en mer oversiktlig meny. Vi har derimot vurdert det dit hen at det vil være unødvendig og bare bli irriterende for administrator å måtte inn i alle mulige undermenyer etter hvert som han/hun blir vant til å bruke systemet. Ved å la brukeren se alle valg på hovedmenyen vil det gi en bedre oversikt over funksjonalitet som systemet har å tilby.



Figur 4.11 Hovedmeny administrasjonsmodul

Menyen for avstemmingsmodulen (figur 4.10) vil være enklere, og kun inneholde tre menyvalg. Struktureringen av disse trenger ingen særskilt forklaring.

Visning, innlegging, endring og sletting av data

Skjermbilder som viser informasjon til endring eller sletting av data eller som inneholder tekstfelter for innlegging av data, har forklarende ledetekster som forteller hva brukeren skal gjøre og hvilken informasjon som skal fylles inn i tekstfeltene. I tillegg vil det finnes knapper for de forskjellige ting man kan gjøre. For å unngå problemet med at vi ikke får plass til alle felter på et skjermbilde og dermed skape uoversiktlighet, har vi i noen menyvalg valgt å lage flere steg før man er "i mål" med hele menyvalget.

Plassering av felter

Plassering av felter er gjort mest mulig med tanke på hvordan brukerne skal fylle ut disse, og med tanke på fornuftig plassbruk på skjermbildet. Feltene er plassert i den rekkefølgen det vil være mest naturlig å skrive inn/lese opplysningene. For eksempel hvis det skal registreres en kandidat kommer fornavn før etternavn, og etternavn før alder.

Der det er mulig må brukeren velge data fra dropdown-lister eller radioknappgrupper, slik at han ikke har mulighet for å fylle inn feil data. Her må det velges et element i stedet for å skrive inn opplysningen selv.

Knapper

Vi har valgt en plassering av knappene som gjør det mest mulig naturlig hvilke knapper bruke ren skal velge til enhver tid.

Siden det er mange menyvalg som har flere skjermbilder og dermed flere steg man må gå gjennom, har vi valgt å gjøre det på den måten som er beskrevet her.

I de skjermbildene hvor knappene ”<< Forrige” og ”Neste >>” finnes, vil disse være plassert henholdsvis til venstre og til høyre. Grunnen til dette er at ”<< Forrige” betyr at man vil gå tilbake, og bør derfor stå lengst tilbake, altså mot venstre. På samme måte vil ”Neste >>” stå til høyre siden dette er mest naturlig, da ”Neste >>” indikerer at man skal videre til neste skjermbilde.

Der det finnes knapper som ”Godkjenn”, ”Lagre”, ”Opprett”, ”Slett”, ”Oppdater” og så videre, vil disse plasseres til høyre, mens ”Forrige” eller ”Avbryt” eller ”Tilbake” vil plasseres til venstre. Dette er av samme grunn som for ”<< Forrige” og ”Neste >>” knappene.

Det vil være standard i hele systemet at knapper som ikke skal kunne klikkes på et gitt tidspunkt er ”disabled”, det vil si at det ikke er mulig å trykke på knappen fordi den er inaktiv. Dette er av den grunn at brukerne skal skjønne at de nå ikke kan utføre handlingen som knappen indikerer. For eksempel hvis brukeren vil oppdatere en velgergruppe vil knappen for å slette velgergruppe være inaktiv hvis det ikke er noen velgergrupper å slette.

Tilbakemeldinger og feilmeldinger

Tilbakemeldinger og feilmeldinger vil dukke opp midt på skjermen i et nytt vindu. Det er viktig at de er så informative som mulig, siden dette er brukernes eneste respons og tilbakemelding fra systemet. Dette er viktig for at brukeren skal stole på systemet og på at det han gjorde ble riktig utført. Meldingene skal kort fortelle hva som ble gjort, og eventuelt hva som gikk galt.

Brukere skal motta kvitteringer i form av tilbakemeldinger når de har utført en endring på systemet, og feilmeldinger når de har gjort noe de ikke har lov til eller når det oppstår feil. Disse endringene kan for eksempel være å avgi stemme, telle opp stemmesedler, eller å legge til, endre eller slette data i systemet. Ting brukerne ikke får lov til gjøre kan være å prøve å lagre uten å ha fylt inn alle nødvendige felter, eller skrive inn feil datatyper. Eksempel på feil som kan oppstå er at programmet ikke får kontakt med databasen.

4.5.2 Beskrivelse av utvalgte menyvalg for applikasjonen

Under beskrivelse av menyvalgene for de to modulene har vi valgt å plukke ut andre deler av systemet enn de som er berørt i low-level use case-beskrivelser i kapittel 3. Dette er gjort for å belyse flere deler av systemet, slik at leseren kan få best mulig innsikt og kjennskap til produktet.

Dette vil være en kort beskrivelse av hvert menyvalg for de to modulene. En mer detaljert beskrivelse med screenshots vil finnes i brukermanualen som er vedlagt på CD. For en beskrivelse av de øvrige menyvalgene refererer vi til vedlagt designdokument på CD.

Avgi stemme

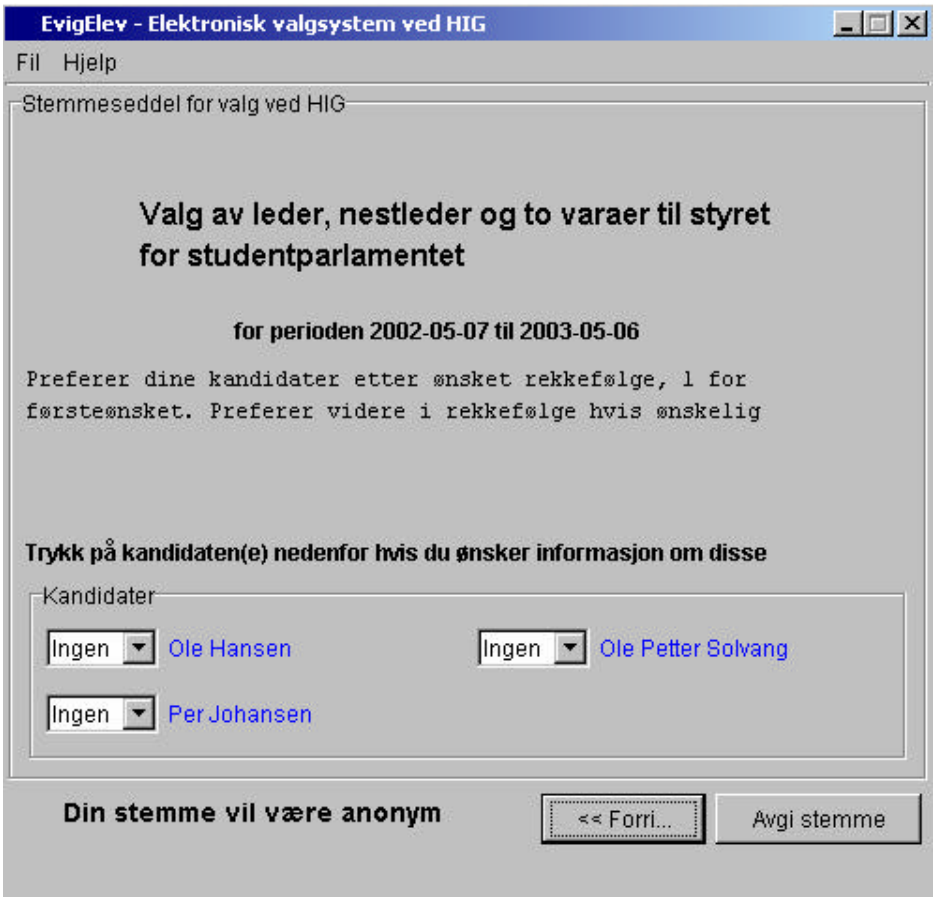
Her vil brukeren få mulighet til å avgi stemme ved de valgene han har stemmerett for.

Først må brukeren oppgi et brukernavn og et passord og hvilket domene han tilhører. Dette blir sjekket opp mot NT-domenet for at brukeren skal få logge seg inn i systemet.

Så vil han bli presentert et skjermbilde med en dropdown-liste, hvor han må velge hvilket valg han ønsker å stemme ved. I denne listen vil kun de valg hvor vedkommende bruker har stemmerett, og ikke allerede har avlagt stemme, komme. Dette er av sikkerhetsmessige årsaker. Brukeren vil på denne måten ikke ha mulighet til å forsøke å avlegge stemme ved et valg han ikke har stemmerett til.

Når ønsket valg er valgt, og brukeren har gått videre, vil stemmeseddelen presenteres i skjermbildet (Figur 4.12). Her vil overskrift, periode som valget gjelder for, og valgreglene bli vist, sammen med kandidatene for valget. Det vil være mulig å få informasjon om hver av kandidatene ved å klikke på den enkeltes navn. Et nytt skjermbilde vil da komme opp. Dette vil inneholde all relevant informasjon som er lagt inn om kandidaten.

Før stemmen kan avlegges må brukeren angi preferansene til kandidatene. Når han forsøker å avlegge stemmen vil systemet sjekke at stemmeseddelen er riktig utfylt.



Figur 4.12 Stemmeseddelen

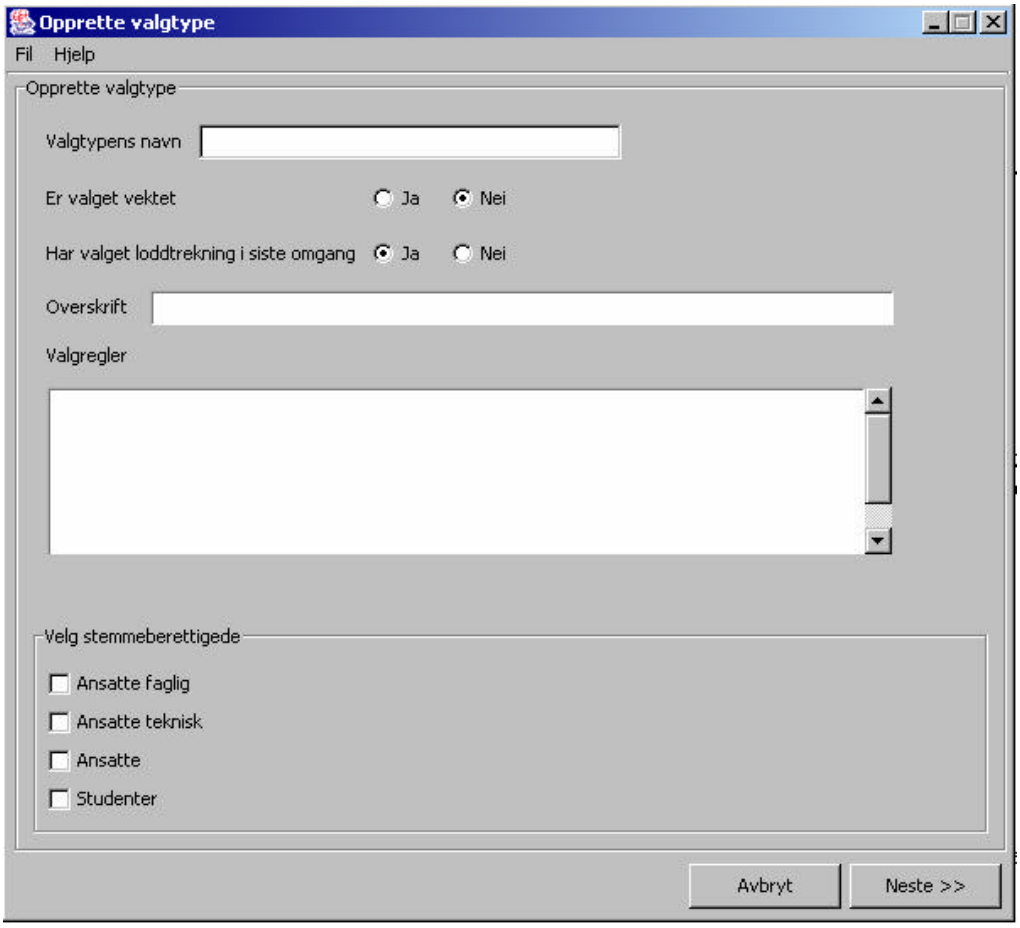
Når stemmeseddelen er kontrollert, vil systemet lage en forhåndsvisning av kandidatene med tilhørende preferanser, slik at brukeren får se hvordan han har prioritert. Her blir brukeren bedt om å bekrefte stemmeavgivningen sin. Når det skjer vil systemet sjekke at ikke velgeren har avlagt stemme ved dette valget.

Opprette valgtype

I dette menyvalget kan brukeren opprette en ny valgtype som skal brukes når bestemte valg skal konfigureres. Her vil brukeren måtte klikke seg gjennom to skjermbilder før han kommer i mål.

I det første skjermbildet må valgtypens navn skrives inn. Så må det angis om det er en vektet eller uvektet valgtype og om det skal foretas loddtrekning ved stemmelikhet i valgets siste omgang. Dette angis ved hjelp av to radioknappgrupper. Deretter må brukeren skrive inn overskrift og valgregler for valgtypen. Disse vil bli satt inn i stemmeseddelen i avstemmingsmodulen. Til slutt i det første skjermbildet ka n det angis hvilke velgergrupper som har stemmerett ved denne typen valg ved å hake ut sjekkbokser. Hvis valgtypen har vekting vil det komme opp tekstfelter hvor det må angis vektingsprosent for hver velgergruppe. Brukeren er ikke nødt til å angi hvilke ve lgergrupper og eventuelt vektingsprosent, i tilfelle dette er usikkert på det tidspunktet valgtypen blir opprettet.

Administrator kan benytte menyvalget ”Oppdatere valgtype” for å gjøre dette på ved en senere anledning.



Figur 4.13 Opprette valgtype

I steg 2 vil brukeren bli forespeilet et nytt skjermbilde hvor han må angi hvilke verv som hører til den valgtypen han holder på å legge inn. Her må vervets navn, hvor mange kandidater som skal velges til dette vervet, om disse eventuelt skal rangeres hvis det er flere enn én, varighet for vervet og til slutt en beskrivelse av vervet legges inn. Rangering eller ikke angis ved å hake ut en sjekkboks. Det er verdt å merke seg at rangering bare gjelder innad for vervet som opprettes, for eksempel at det skal kåres to vararepresentanter, men at disse deles i 1. vara og 2. vara.

Etter hvert som man legger til vervene vil de bli presentert i en ramme nederst i skjermbildet slik at man kontinuerlig får se hvilke verv som er lagt til så langt. Når administrator har lagt til alle vervene for valgtypen kan han opprette denne. Valgtypen vil da umiddelbart bli klar til bruk.

Hver valgtype trengs kun å bli lagt til én gang. For eksempel hvis administratoren har lagt inn valgtypen ”Rektorvalg”, vil denne valgtypen kunne brukes ved alle fremtidige rektorvalg.

Registrere kandidater

Dette menyvalget brukes for å legge inn kandidater til de valg som allerede er konfigurert, men hvor avstemmingsperioden ikke har begynt. Brukeren må først velge hvilket valg han ønsker å legge inn kandidater for. Dette gjøres ved å velge ønsket valg i nedtrekkslisten øverst. I denne listen vil i tillegg de valgene hvor avstemmingsperioden har begynt ligge. Grunnen til det er at det kan være ønskelig for administrator å skrive ut kandidatlister for valg som er under avstemming. Når et slikt valg er valgt vil det *kun* være mulig å skrive ut kandidatlisten eller gå tilbake til hovedmenyen. De andre knappene vil være inaktive.

Når administrator skal legge inn kandidatinformasjon skal brukernavnet til kandidaten, navn, alder, kjønn, stilling eller klasse, avdeling og e-mail registreres. Kjønnet angis ved å velge en radioknapp. I tillegg kan hjemmesideadresse og en beskrivelse av kandidaten legges inn hvis det er ønskelig.

Administrator har mulighet til å nullstille alle feltene, lagre kandidaten eller skrive ut kandidatlisten for det valgte valget. Når han har lagret en kandidat vil alle feltene automatisk bli blanket ut igjen slik at alt er klart for å legge inn neste kandidat. Hvis han vil skrive ut kandidatlisten, vil den bli presentert i Notepad, med mulighet for å skrive ut derfra.

Etter at alle kandidatene er lagt inn kan administratoren gå videre og se en forhåndsvisning av stemmeseddelen med alle kandidatene som er lagt inn for det angitte valget. I dette skjermbildet kan han slette kandidater hvis dette skulle være ønskelig, gå tilbake og legge inn flere kandidater eller avslutte kandidatinnleggingen. Hvis han ønsker å slette en kandidat må denne først velges fra en nedtrekksliste med alle kandidatene, ved siden av slett-knappen.

Vedlikeholde data

Her kan administrator gå inn og slette gamle stemmesedler når disse har vært ”oppbevart” tilstrekkelig lenge, eller slette gamle valg som ikke lenger er interessante å beholde. Skjermbildet her vil være todelt, øverst kan han slette stemmesedler og nederst kan gamle valg slettes.

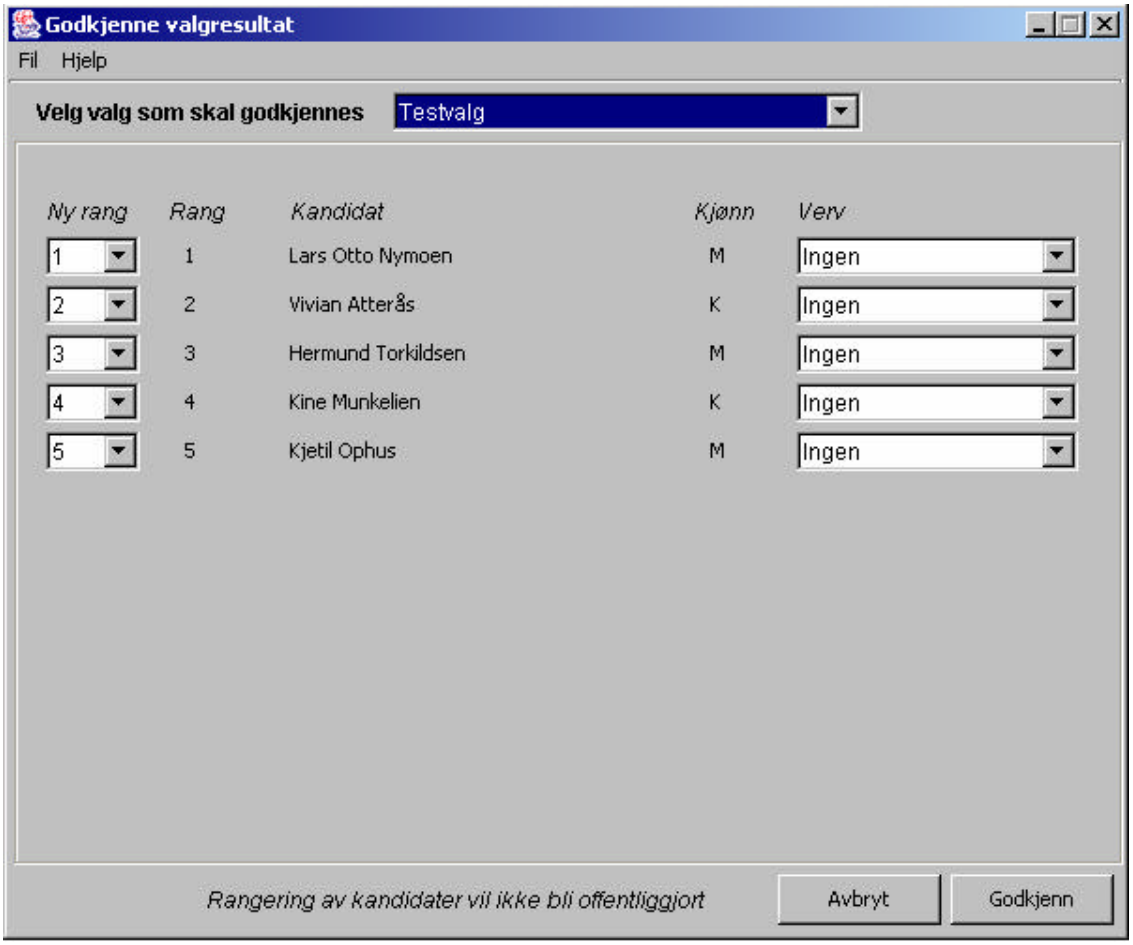
I den øverste dropdown-listen velges hvilket valg stemmesedlene skal slettes fra. I denne lista vil de valgene som er opptalt og godkjent finnes. Når et valg er valgt her, vil avstemmingsdatoene for valget vises, slik at administratoren kan være sikker på at det er riktig valg han sletter stemmesedler for.

I dropdown-listen i den nederste delen av skjermbildet vil de valgene som er godkjent og opptalt, men som enda ikke er slettet, finnes. Etter at et valg er valgt her, må brukeren også velge gjeldende periode for valget han vil slette. Dette gjøres i nok en dropdown-liste som ligger nedenfor. Grunnen til at også datoene må velges er at det kan ligge inne flere valg av samme valgtype som enda ikke er slettet. Når et valgt valg blir slettet vil samtidig dets kandidater, stemmesedler, opptellingsresultat og valgresultat bli slettet.

Godkjenne valgresultat

Her kan administratoren gå inn og godkjenne og eventuelt endre på opptellingsresultatet hvis dette skulle være ønskelig, før det blir et endelig valgresultat.

Aktuelt valg må velges fra nedtrekkslisten øverst. Her vil valgene som er opptalt men ikke godkjent komme. Når dette er gjort vil opptellingsresultatet komme opp på skjermen, med mulighet for å endre på rekkefølgen på kandidatene, og mulighet for å fordele vervene som er lagret for denne valgtypen. Disse endringene og fordelingene gjøres ved hjelp av nedtrekkslister som inneholder plasseringstall og verv. Når administratoren skal godkjenne valgresultatet, vil systemet sjekke at kandidatene er i rekkefølge fra en og oppover, og at hvert verv er fordelt maksimum det antall ganger som er spesifisert for dette vervet.

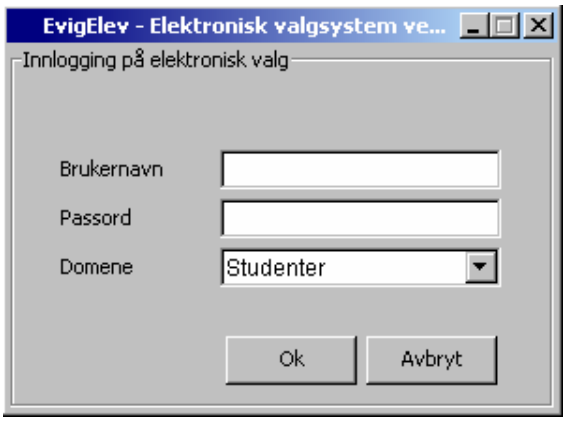


Figur 4.14 Godkjenne valgresultat

Når valgresultatet er godkjent vil dette være tilgjengelig for alle på internett. Valget vil bli satt til å være ferdig godkjent, valgtypen vil frigis og bli klar til ny bruk igjen, stemmeberettigede til valget blir slettet og nominasjonene til valget blir slettet.

4.5.3 Pålogging

Pålogging må skje uansett hvilken modul brukeren ønsker å bruke. I administrasjonsmodulen vil påloggingsvinduet dukke opp midt på skjermen når applikasjonen startes, og gyldig administratornavn og passord må angis for å komme inn. I tillegg må administratoren logge seg inn hvis han ønsker å se opptellingsrapport fra et valg. Denne ligger på ei internettside hvor han må logge seg på. I avstemmingsmodulen vil også påloggingsvinduet dukke opp midt på skjermen når brukeren vil stemme eller nominere kandidater. Her må gyldig brukernavn, passord og domene angis (figur 4.15). Dette blir sjekket mot NT-domene, altså må man logge på med samme brukernavn og passord som man benytter på arbeidsstasjonene ved høgskolen.



Figur 4.15

4.6 Grensesnitt mellom applikasjonene og SQL Server

Som grensesnitt for kommunikasjon mellom applikasjon og database trengs en databasemotor. Mot SQL Server 2000 blir det brukt ADO (Microsoft ActiveX Data Objects). Denne databasemotoren er utviklet av Microsoft og fungerer derfor godt mot SQL Server. Kommunikasjonen mellom applikasjon og database består i oppkobling og SQL-spørringer fra applikasjonen mot databasen.

Operasjonene mot SQL server fra programmet vårt foregår med standard SQL-spørringer. Vi har brukt SELECT-, DELETE-, UPDATE- og INSERT-setninger. Til orientering er SELECT et kall som henter ut data fra databasen, DELETE sletter data, UPDATE oppdaterer data, og INSERT skriver til databasen. Hvis det er flere SQL spørringer som hører sammen slik at enten alle eller ingen skal utføres, har disse blitt knyttet sammen i en transaksjon.

5 Implementering, koding og produksjon

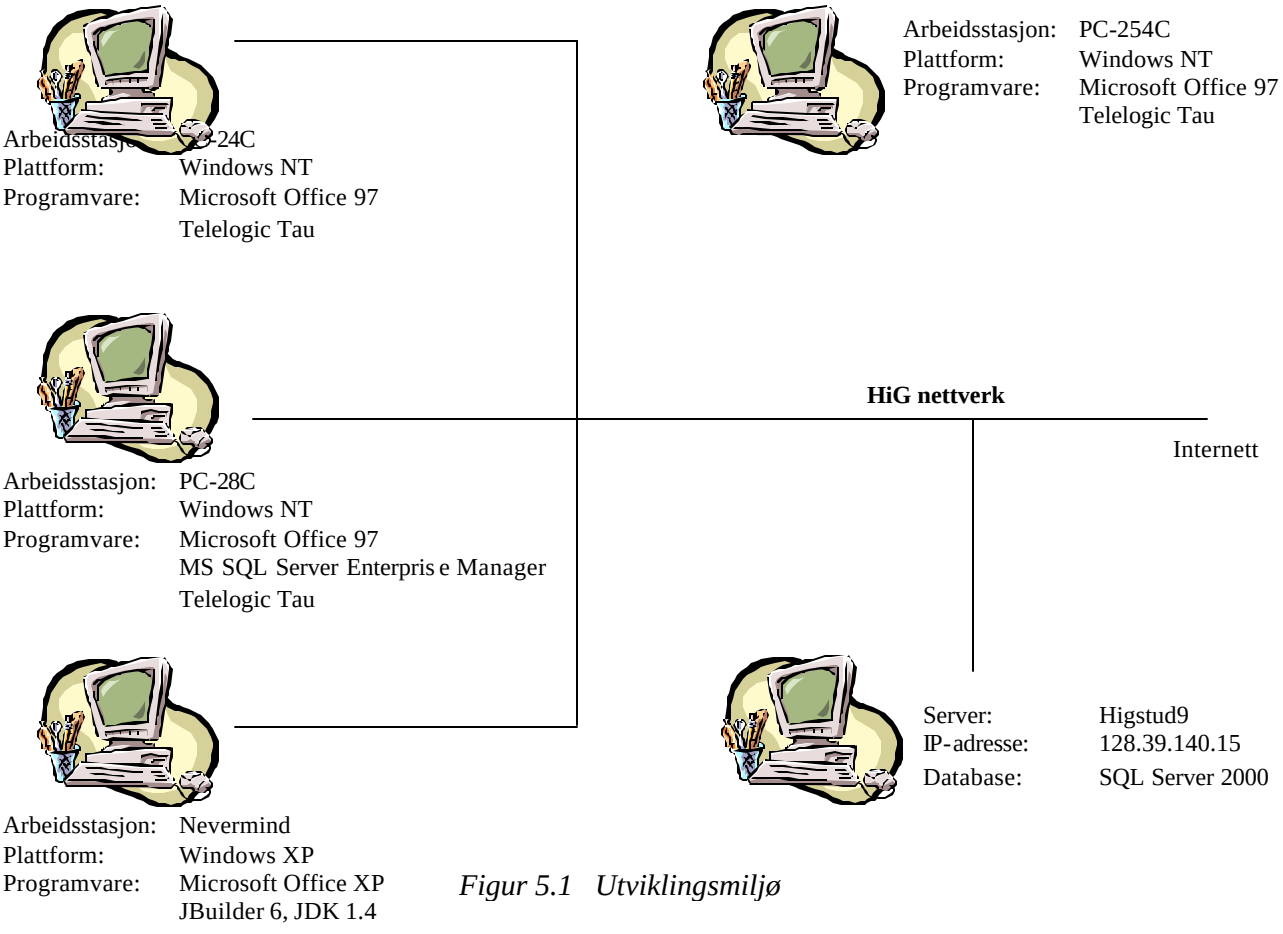
5.1 Plattform

Systemet som skal utvikles er en modulinndelt databaseapplikasjon med grensesnitt som skal fungere uavhengig av hvilken plattform arbeidsstasjonen benytter. Systemet skal derfor kunne kjøres på høgskolens intranett fra både Linux- og Windows-labene, og ansattes egne arbeidsstasjoner.

5.2 Utviklingsmiljø

I hovedprosjektets oppstartsfase ble gruppen tildelt tre PC-er som kunne benyttes i hovedprosjektarbeidet. Disse ble koblet opp mot høgskolens nettverk. Etter hvert som behovet meldte seg ble egne PC-er tatt med for å erstatte de vi hadde fått tildelt, dette fordi de tildelte PC-ene ikke var kraftige nok til å kjøre nødvendig programvare. Ikke alle maskiner var koblet opp mot nettverket.

Alle arbeidsstasjonene hadde installert Microsoft Office. Videre ble Microsoft SQL Server Enterprise Manager, Telelogic Tau, JBuilder 6, og JDK 1.4 installert på enkelte maskiner.



Figur 5.1 Utviklingsmiljø

5.3 Valg av verktøy

5.3.1 Programmeringsspråk

Java

Valget av programmeringsspråk som skulle benyttes i prosjektet falt tidlig på Java, siden dette er et programmeringsspråk som er fleksibelt, fungerer godt mot databaser, og alle gruppemedlemmene hadde litt erfaring med språket fra før. En annen faktor for å velge Java, er at det er plattformuavhengig og vil kunne kjøres fra alle datalaber og arbeidsstasjoner ved høgskolen. Videre ble det uttrykt ønske fra IT-tjenesten om at applikasjonen skulle utvikles i Java, blant annet fordi en del andre hovedprosjekter med høgskolen som oppdragsgiver denne våren også utvikles i dette miljøet.

Vi hadde også lyst til å benytte Java som programmeringsspråk på grunn av personlige utfordringer og egenutvikling, siden Java er et programmeringsspråk som er i vinden for tiden, det er ”trendy”, spennende, og det representerer fremtiden.

Som utviklingsverktøy ønsket vi å benytte JBuilder. Etter å ha undersøkt litt om de forskjellige versjonene og deres muligheter og ulemper, samt konsultert med høgskolen om hva som ville være mulig å skaffe av lisenser, kom vi fram til at JBuilder 6 var et godt valg.

Java Servlets

I utgangspunktet ønsket vi å bruke PHP for utvikling av websidene hvor rapporter, statistikk og valgresultater skal presenteres. Dette ble vurdert opp mot bruk av ASP og Java Servlets, men PHP ble foretrukket av ulike grunner. Alle gruppemedlemmene har lært PHP godt gjennom tidligere fag og prosjekter. Hver gang en ny side kalles, startes en ny prosess i PHP, mens det i Java Servlets og ASP startes nye tråder. Dette fører til at PHP krever mindre ressurser enn de to andre, samtidig som det gir mindre kode enn Java Servlets.

Men bruk av PHP gikk likevel ikke så greit som først antatt. MS SQL-driveren for PHP er ustabil, og vil ikke fungere som ønsket. Derfor ville alternativet vært Sybase dersom vi ønsket å bruke PHP, men da var vi avhengig av at IT-tjenesten installerte drivere, og i tillegg skaffet lisenser, så dette var uaktuelt. Valget falt derfor i stedet på Java Servlets. En fordel med Java Servlets er at det er raskere enn PHP og ASP, og det er dessuten mer fleksibelt ved skifte av database. Derfor var Java Servlets et greit valg for oss.

For å utvikle serversidene i Java Servlets benyttet vi JBuilder 6 og resultatene ble vist i Microsoft Internet Explorer 5.5

5.3.2 Database

Etter et møte med IT-tjenesten ble det klarlagt at det var ønskelig at vi skulle benytte MS SQL Server 2000, som allerede er en etablert databaseserver ved høgskolen. Java fungerer greit mot SQL Server, og alle gruppemedlemmene hadde på forhånd noe erfaring med denne typen database gjennom faget ”Databaser II”.

5.3.3 Prosjektstyringsverktøy

Valg av verktøy for prosjektstyring var relativt enkelt og vi kom raskt fram til at vi skulle benytte gantt-skjemaer. For å sette opp disse valgte vi å bruke Microsoft Project. Dette er et utbredt prosjektstyringsverktøy, som gruppemedlemmene hadde erfaring med.

Bruk av gantt-skjemaer ser vi som en effektiv og god måte å få satt opp oversikt over tidsplanlegging, hvilke aktiviteter som skal være ferdigstilt til hvilke tidspunkter, og videre hvilke aktiviteter som avhenger av og/eller kommer etter hverandre.

5.3.4 Systemutviklingsverktøy

Som verktøy for å sette opp diagrammer og modeller for systemanalyse og arkitektur, valgte vi å bruke Telelogic Tau UML Suite 4.5. Dette er et bra verktøy som gruppemedlemmene har erfaring fra gjennom tidligere prosjekt. Tau er relativt enkelt å bruke, og gir gode muligheter for tildeling av rettigheter til flere brukere, slik at alle gruppemedlemmene kan aksessere prosjektet.

5.3.5 Andre verktøy

Når det skulle tas backup av data var det hensiktsmessig å komprimere disse dataene først. Her benyttet vi WinZip 8.0, som enkelt lar oss ta backup både til disk og diskett, og eventuelt lager multispanned diskset dersom det ikke er plass til dataene på én diskett. WinZip ble også brukt til å lage en self-extracting .exe fil til installasjon av valgsystemet fra CD.

For å kunne jobbe opp mot SQL Server 2000, ble programmet MS SQL Server Enterprise Manager brukt. I dette programmet kan opprette database i SQL Server, og man logge på en aktuell database, og videre legge inn data eller gjøre endringer i denne databasens design. Spørringer mot databasen ble kjørt i Query Analyzer som er et tilleggsverktøy til SQL Server Enterprise Manager.

Databasemodeller er utviklet i Modelator.

5.4 Kode

5.4.1 Prinsipper for koding

Siden flere gruppemedlemmer til enhver tid skulle sitte og jobbe med koding av applikasjonen, kunne det lett bli vanskelig for hver enkelt å holde oversikten over hva og hvor ting er gjort. Det var derfor viktig at så mye som mulig ble gjort for at koden skulle bli lettlest og enkel å sette seg inn i. Derfor har vi har utarbeidet noen standarder for bruk i selve kodingen. Dette vil også kunne lette kommenteringen av koden noe; ikke alt trenger å kommenteres dersom det er brukt intuitive og fortellende navn på variable, funksjoner og lignende. Navngiving er vist i prefikslisten i punkt 5.4.3.

Koden er strukturert ned i mindre filer. Det er satt som standard at det skal lages egne filer for hver enkelt frame i applikasjonen. På denne måten vil det bli lettere å finne fram til de delene av koden man ønsker.

Det er forsøkt lagt vekt på god struktur og oversikt i koden, slik at den vil bli lettere å lese, og man vil lettere kunne se hvilke deler av koden som hører sammen.

5.4.2 Standarder

Navngivning
- Alle filer som inneholder GUI har endelsen GUI.java
- kildekodefiler gis navn etter funksjonalitet, for eksempel OppretteValgtypeGUI.java
- filer og prosjekter organiseres i kataloger etter versjonsnummer

Kildekode
- kildekodefiler skal inneholde dato, navn på fil, versjonsnummer og gruppenavn (se figur 5.2)
- variabelnavn gis beskrivende, logiske navn

```
/*##############################################################
#       Dato:           19.04.2002                             #
#       Filnavn:        StemmeseddelGUI.java                   #
#       Versjonsnr:     1.0                                    #
#       Gruppenavn:     Evig Elev                              #
#                                                              #
#       Kommentarer: ----------                                #
##############################################################*/
```

Figur 5.2

5.4.3 Standarder for navngiving i Java- kode

Siden alle gruppemedlemmene er involvert i kodingsprosessen har vi funnet det nødvendig å utarbeide noen standarder for navngiving i Java-koden.

Prefikser	Komponenter
btn	JButton
chk	JCheckBox
cmb	JComboBox
lbl	JLabel
pnl	JPanel
pwd	JPasswordField
rbt	JRadioButton
rbtgrp	RadioButtonGroup
scr	JScrollPane
tbl	JTable
txt	JTextField
txta	JTextArea

- Alle navn på metoder skal alltid starte med liten bokstav
- Klassenavn skal begynne med stor bokstav
- Navn på Java -filer skal ha samme navn som klassen i hver fil.
- Komponenter skal gis en fast prefiks for hver type. Prefiksene for de ulike komponentene skal oppdateres i tabellen første gang de blir tatt i bruk. Disse skal brukes konsekvent gjennom kodingen for å fremme oversikten
- Konstanter skal skrives med store bokstaver

5.5 Eksempler fra kildekode

Under følger utdrag fra kildekoden som vi mener gir et godt inntrykk av valgt struktur for koden og som belyser funskjonalitet som er hyppig brukt og som representerer særskilte situasjoner. Dersom mer kildekode ønskes refererer vi til vedlagt kildekode på CD under mappen 'Kildekode'.

5.5.1 Oppkobling mot database

Figur 5.3 viser kildekoden for koblingsklassen som brukes for å opprette forbindelse til databasen. Et koblingsobjekt blir opprettet i henholdsvis AdminLoginGUI og VelkommenGUI, som er første skjermbilde i administrasjonsmodulen respektive avstemmingsmodulen. Dette koblingsobjektets Connection-objekt sendes videre til alle klasser som enten skriver til eller henter ut data fra databasen, slik at hver enkelt modul benytter samme kobling.

```
package no.hig.EvigElev.Database;

import java.sql.*;
import java.util.*;
import javax.swing.JOptionPane;

/*#################################################################
#                                                                 #
#   Dato:        20.05.2002                                       #
#   Filnavn:     Kobling.java                                     #
#   Versjonsnr:  3.0                                              #
#   Gruppenavn:  EvigElev                                         #
#                                                                 #
#   Kommentarer: klasse/pakke for opp- og nedkobling av           #
#                database                                         #
#                                                                 #
#################################################################*/

public class Kobling
 {
 private String URL = "jdbc:odbc:EvigElevSQLServer";
 private String BRUKERNAVN = "EvigElev";
 private String PASSORD = "*********";
 private Connection kobling;

 public Kobling()
  {
  try
   {
   Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   System.out.p rintln("Databasedriver er lastet");
   setConnection(kobling = DriverManager.getConnection(URL, BRUKERNAVN, PASSORD));
   System.out.println("Databasen er tilkoblet");
   }
  catch (ClassNotFoundException cnfe)
   {
   JOptionPane.showMess ageDialog(null, "Fikk ikke lastet opp JDBC/ODBC driver");
   cnfe.printStackTrace();
   System.exit(1);
   }
  catch (SQLException sqle)
   {
   JOptionPane.showMessageDialog(null, "Klarte ikke å koble opp mot databasen");
   sqle.printStackTrace();
   }
 }

 public void setConnection(Connection kobling)
  { this.kobling = kobling; }

 public Connection getConnection()
  { return kobling; }
```

```
 public void lukkDB(Connection kobling)
  {
  this.kobling = kobling;
  try
   {
   kobling.close();
   System.out.println("Databaseforbindelse lukket");
   }
  catch (SQLException sqle)
   {
   JOptionPane.showMessageDialog(null, "Fikk ikke lukket databasen");
   sqle.printStackTrace();
   }
 }
}
```

Figur 5.3 Kildekode for koblingsklasse

5.5.2 Kombinasjonsboks

Kombinasjonsbokser er en komponent vi har brukt mye. De fleste skjermbilder inneholder en kombinasjonsboks øverst med relevante valg for de enkelte skjermbildene. Med dette mener vi at SQL-spørringen som henter ut valgene i den enkelte kombinasjonsboks vil variere fra menyvalg til menyvalg, siden det eksempelvis er forskjell på hvilke valg man kan registrere kandidater til og hente ut stemmesedler fra. Siden kombinasjonsboksens innhold hentes fra database blir metoden for å initialisere denne litt mer komplisert enn dersom man har en vanlig array med stringer. Figur 5.4 viser et eksempel på implementeringen av lagcmb-metoden som setter innholdet i nedtrekkslisten.

```
 public void lagcmb()
 {
 String sql = "SELECT Valgtype.ValgtypeID, Valgtype, ValgID FROM Valgtype, Valg WHERE " +
 "ErAktivt=1 AND ErOpptalt=0 AND Valg.A_sluttdato >= (GetDate()-1) AND " +
 "Valg.ValgtypeID=Valgtype.ValgtypeID";

 try
    {
    uttrykk = kobling.createStatement();
    resultat = uttrykk.executeQuery(sql);
    }
  catch (SQLException sqle)
    { sqle.printStackTrace(); }
 try
  {
  if (resultat.next())
    {
    do
     {
      String valgtypeID = resultat.getString("ValgtypeID");
      String valgtype = resultat.getString("Valgtype");
      String valgID = resultat.getString("ValgID");
      AktivValgtypeID.addElement(valgtypeID);
      AktivValgtype.addElement(valgtype);
      AktivValgID.addElement(valgID);
     }
```

```
    while (resultat.next());
    }
  uttrykk.close();
  }
catch (SQLException sqle)
  { sqle.printStackTrace(); }

int size = AktivValgtypeID.size();
strAktivValgtype = new String[size];

if (size == 0)
  {
  strAktivValgtype = new String[1];
  strAktivValg type[0] = "Ingen aktive valg";
  btnOppdater.setEnabled(false);
  }

else
  for (int i = 0; i < AktivValgtypeID.size(); i++)
    { strAktivValgtype[i] = AktivValgtype.elementAt(i).toString(); }
cmbGjeldendeValg = new JComboBox(strAktivValgtype);
cmbGjeldendeValg.addItemListener(new ItemListener()
  {
  public void itemStateChanged(ItemEvent e)
    { visDatoer(); }
  });
}
```

Figur 5.4 Kildekode for metoden lagcmb

5.5.3 Transaksjon ved uthenting av StemmeseddelID for velger

Siste kodeutdrag viser hvordan vi har gruppert flere SQL-setninger slik at disse kjøres som en transaksjon, og dermed oppnår at alle eller ingen SQL-uttrykk kjøres. Denne problematikken er nærmere beskrevet i kapittel 4.4.1.

De viktige kallene i denne sammenhengen er 'kobling.setAutoCommit(false)', som sørger for at ingen SQL-setninger kjøres før 'kobling.commit()' kalles. Deretter settes koblingens AutoCommit til true igjen. Dersom transaksjonen ikke ble gjennomført vil 'kobling.rollback()' sørge for at databasen beholder sin konsistente tilstand. Se figur 5.5 for kildekode.

```
try
  {
  kobling.setAutoCommit(false);
  stOppdaterStemmeseddel = kobling.createStatement();
  while (enum.hasMoreElements())
    {
    Vekt = enum.nextElement().toString().trim();
    if (Vekt.equals("Ingen"))
      Vekt = "0";
    KandidatID = enum.nextElement().toString().trim();
    if (!Vekt.equals("0"))
      if (feilrapport.equals(""))
        {
        if (!(max >= 0))
          max = 0;
        String sqlOppdaterStemmeseddel =
                  "INSERT INTO Stemmeseddel (StemmeseddelID, Preferanse, VelgergruppeID, " +
                  "ValgID, KandidatID) VALUES (" +max+ ", "+Integer.parseInt(Vekt)+", +
                  "velgergruppeID+","+hentValgtValgID() +","'+KandidatID+"')";
        stOppdaterStemmeseddel.executeUpdate(sqlOppdaterStemmeseddel);
        }
    }
  kobling.commit();
  stOppdaterStemmeseddel.close();
  kobling.setAutoCommit(true);
  }
catch (SQLException sqle)
  {
  feilrapport = "Det skjedde en feil under registreringen av din stemme.\n"+
                "Vennligst prøv igjen";
  sqle.printStackTrace();
  try
    { kobling.rollback(); kobling.setAutoCommit(true);}
  catch (Exception e)
    { e.printStackTrace(); }
  }
```

Figur 5.5 Kildekode med bruk av transaksjon

6 Kvalitetssikring

6.1 Kvalitetssikring

6.1.1 Sikring av data og backup

For å sikre oss mot tap av data på best mulig måte har vi hatt backuprutiner som vi har fulgt. Disse rutinene er beskrevet ble beskrevet i forprosjektrapporten, og lyder som følger:

- Backup-rutiner
  - prosjektarbeid vil fortløpende bli lagret på dedisert område på hovedprosjektserver k: (higstud1), slik at samtlige gruppemedlemmer har tilgang på hverandres arbeid.
  - etter hver dags arbeid vil vi ta manuell backup til lokal disk på tildelte maskiner (A204)
  - ved ukeslutt tas backup til CD eller diskett, avhengig av datamengde.

Samtlige gruppemedlemmer har fulgt disse standardene på en tilfredsstillende måte. Vi har fortløpende lagret data vi har jobbet med på hovedprosjektserveren k:. I visse perioder har det vært problemer med at denne serveren har vært full, slik at vi ikke har kunnet lagret data der. Vi har da måttet lagre data lokalt eller på våre personlige områder på serveren h:, uten å kunne benytte k:. Stort sett har det likevel gått greit å lagre data på hovedprosjektdisken. I tillegg til å lagre dataene der har vi tatt backup som ble lagret lokalt hver dag, og vi har tatt med backup på disketter hjem flere ganger i uka. Det har ikke vært nødvendig å legge data på CD-er, siden datamengdene ikke har vært så store at dette har vært nødvendig.

6.1.2 Prosjektmodellens innvirkning

Med å ta i bruk RUP som systemutviklingsmodell mener vi å ha fått et raskt overblikk over svakhetene ved dagens manuelle valgordning, og tidlig kunnet komme i gang med å kravspesifisere og detaljere ned kritiske use cases som berører stemmeavgivnings- og valgkonfigurasjonsproblematikk. Vi mener at muligheten til å ta fatt på de største utfordringene tidlig i utviklingsfasen har vært med på å redusere risikoen for å ende opp med et ufullstendig valgsystem og samtidig sikre at kvaliteten på det totale produktet har blitt tilfredsstillende.

En annen fordel med denne utviklingsmodellen er at vi har hatt muligheten til å teste for eksempel kode for stemmeavgivning tidlig i utviklingsforløpet, slik at vi kunne dra nytte av den erfaringen før vi gikk løs på nominasjonsdelen av avstemmingsmodulen. På den måten kunne vi gjøre korrigeringer for andre deler av systemet som involverte samme problematikk og spare verdifull tid som kom testingen av det ferdigutviklede systemet til gode.

RUP åpner for mye kundekontakt, eller i vårt tilfelle kontakt med valgstyrets leder. Det at kontaktperson har hatt anledning til og krav på å prøve systemet på ulike tidspunkt og kunnet komme med konstruktiv kritikk, har helt sikkert bidratt til en stødig kurs mot et valgsystem som vil være et tilnærmet fullverdig alternativ til dagens manuelle ordning. Og det at

valgstyret får et system som har nytteverdi og som innfrir deres krav er jo essensen av kvalitet.

6.2 Testing

6.2.1 Testing av Java-applikasjonen

Under testingen av administrasjons- og avstemmingsmodulen, har vi valgt å benytte oss av to ulike måter for testing av systemet. I tillegg har vi kjørt brukertesting av GUI og testkjøring av applikasjonen.

Den første av disse testmetodene er White Box-testing. Denne formen for testing tester at utdata i applikasjonen er den samme som inndataene. Her bør det testes med alle mulige datatyper, for å sjekke hvorvidt systemet kun godtar de datatypene som skal godtas. Denne typen testing ble utført kontinuerlig under utviklingen av systemet. Etter hvert som en frame var utviklet, ble den testet for både om den godtok riktige typer data, og hvorvidt den hentet ut riktig utdata fra databasen i forhold til hva som var meningen. Fordelen med å utføre denne testingen kontinuerlig under utviklingen, er at utvikleren har koden friskt i minne. Det vil derfor gå raskere å rette eventuelle feil og mangler ved funksjonaliteten etter hvert enn om man skal ta det igjen på et senere tidspunkt. Videre vil man få luket bort mange feil underveis i utviklingsprosessen slik at sjansen for et stort behov for omfattende feilretting til slutt minker vesentlig.

White Box-testing ble også foretatt både på gamle og nye deler av systemet etter hver iterasjon, da nye deler ble integrert i systemet, samt etter at hele systemet var ferdig utviklet. Som et redskap for å undersøke verdier og data i systemet til enhver tid, benyttet vi System.out.println(verdi). Dette hjalp oss til å kontrollere at systemet benyttet riktige verdier fra databasen forskjellige steder i koden.

Underveis i utviklingsprosessen ble valgsystemet kjørt mot Microsoft Access-database. Dette ble gjort av praktiske grunner. Testing av systemet etter databasekonverteringen avslørte at deler av systemet som hadde fungert mot Access databasen, fungerte ikke uten videre mot SQL Server databasen. Omskriving av spørringer mot databasen, samt omstrukturering av koden måtte derfor til. Dette gjaldt spesielt lukking av statement-objekter, attributtrekkefølgen i spørringer hvor data skal hentes ut, samt å gjøre om funksjonen NOW() til GetDate() der hvor dagens dato skal hentes ut (se punkt 7.3.3 for mer om dette).

Den andre testmetoden vi brukte i testingen av systemet, er Black Box-testing. Denne metoden innebærer å sjekke hvordan systemet er oppbygd og hvordan det fungerer sammen. Denne formen for testing ble foretatt etter hver iterasjon, når nye deler ble tatt inn i systemet. Det ble da kjørt datamengder inn i systemet og testet hvordan systemet fungerte sammen, om riktige verdier ble medsendt til riktige steder og lignende.

Det ble også kjørt Black Box-testing til slutt, etter at systemet var ferdig utviklet. Denne gang med en større mengde data.

Vi valgte å ikke integrere nye deler av systemet oftere enn etter hver iterasjon, siden vi kjørte relativt korte iterasjoner, spesielt i construction-fasen. En annen årsak til at vi valgte å ikke integrere oftere, er arbeidsformen som ble benyttet under construction-fasen, da hvert enkelt gruppemedlem måtte jobbe hjemme og gruppa således var sjeldnere samlet.

Under utviklingsfasen, før systemet var ferdig utviklet, ble det også kjørt en liten form for brukertesting, hvor blant annet kontaktperson og veileder prøvekjørte systemet med den funksjonalitet som allerede var utviklet. Målet med denne testingen var ikke bare å finne feil, men også å få tilbakemeldinger, innspill og forslag til hva som kunne gjøres annerledes i løsningen.

Etter at systemet var ferdig utviklet, ble det kjørt tester på stemmeopptellingsrutinene, både for vektede og uvektede valg. Dette har vært den største og viktigste rutinen å få testet i valgsystemet, siden den er helt avgjørende for om systemet genererer et korrekt valgresultat og dermed også er forsvarlig å bruke. Stemmesedlene fra to tidligere avholdte valg ved høgskolen ble matet inn i systemet, og opptelling på disse ble kjørt. Resultatet fra disse testene var over positive, og dokumenterte til og med feil i den manuelle opptellingen ved et av de aktuelle valgene. Resultatene fra denne testingen er dokumentert i vedlegg G – testdokumentasjon, sammen med annen testing. Vi mener å tro at disse resultatene bekrefter behovet for et elektronisk valgsystem ved høgskolen.

Etter hver oppdatering av ny funksjonalitet, samt etter testing og feilretting, har systemet fått nytt versjonsnummer. Systemet endte til slutt på versjon 3.0.

6.2.2 Testing av SQL Server databasen

For å teste databasen valgte vi å foreta White Box-testing. Dette ble gjort i begynnelsen av mai, etter at en konsistent databasestruktur var utarbeidet. Testingen ble gjennomført ved å kjøre aktuelle SQL-spørringer direkte mot databasen. I SQL Server 2000 kan dette gjøres i et verktøy som heter SQL Query Analyzer. Her skrives spørreuttrykket inn, og resultatdataene fra databasen returneres.

Spørringene som ble kjørt, var først og fremst de aktuelle spørreuttrykkene som er brukt i de forskjellige delene av valgsystemet, men også spørringer for å sjekke funksjonalitet av diverse egenskaper databasen skal ha, slik som autonummerering, påkrevde attributter, samt relasjonsforbindelser (avhengighet mellom tabeller).

6.2.3 Feil som oppstod under slutt-testen, og rettelser av disse

Et problem som dukket opp i slutt-testen var når vi opprettet et valg av en valgtype som allerede var ferdig avholdt og lå lagret i databasen. Siden majoriteten av kombinasjonsboksene som benyttes i mange av menyvalgene henter ut valgtypenavnet, og referer til dette med en valgtypeID, kunne det være to eller flere valg som hadde samme valgtypeID. Dermed måtte vi bruke valgID til å referere elementene i nedtrekkslistene, siden denne er unik for hvert valg.

Hadde vi kjørt alle tester med bare valg av ulike valgtyper, er det ikke sikkert at vi hadde oppdaget denne feilen, som eksempelvis ville ført til at feil valg ble slettet eller at en endring av valgkonfigurasjonen for et aktivt dekanusvalg hadde påvirket samtlige dekanusvalg.

7 Beskrivelse av utviklingsprosessen

7.1 Valg av systemutviklingsmodell

Vi valgte Rational Unified Process som utviklingsmodell for prosjektet. Grunnen til at vi foretrakk denne framfor den tradisjonelle fossefallsmodellen er at vi hadde liten innsikt i prosjektets problemområde på forhånd, og regnet med at nye krav ville komme til underveis i prosessen etter hvert som forståelsen for den manuelle ordningen ble større. RUP er et systemrammeverk som tar opp i seg endringer og gjorde det mulig for oss å gå tilbake til ulike artifakter og oppdatere disse.

Det mest realistiske alternativet til RUP var eXtreme Programming, også kalt XP, som er mindre dokumentasjonsorientert og legger mer ressurser på å kontinuerlig utvikle nye deler av systemet og integrere disse. XP stiller enda større krav til engasjement hos kunden/oppdragsgiver enn det RUP gjør, og dette var noe vi så for oss som et mulig problem, siden vår kontaktperson har begrenset innsikt i datasystemer generelt og heller ikke har en timeplan som tillater den tilgjengelighetsom et XP-prosjekt ville krevd. I tillegg hadde tre av gruppemedlemmene benyttet RUP på et systemutviklingsprosjekt før jul som gjorde at vi hadde både erfaring og lyst til å kjøre et systemutviklingsforløp fullt ut med dette rammeverket.

Kanskje den mest tungtveiende årsaken til at RUP ble foretrukket framfor XP er at vi mener det er i alles interesse at det foreligger en grundig systemdokumentasjon for et valgsystem slik at en eventuell videreutvikling av applikasjonen har noe å støtte seg til.

7.2 Planlegging og gjennomføring av prosjektet

Før prosessen ble satt i gang satte vi opp et gantt-skjema for hele prosjektperioden. Siden dette er det første full-skala-prosjektet vi har gjennomført brukte vi oppgitte tall for ressurs- og tidsbruk fra forelesninger før jul. Vi valgte å la inception- og transition-fasen bestå av én iterasjon hver, mens elaboration og construction ble inndelt i to iterasjoner hver. Etter hver milepælssjekk gikk vi i gang med å lage fase- og iterasjonsplan for neste fase, samt skrive noen ord om hvorvidt milepælen ble nådd og eventuelt årsakene til at vi ikke nådde målet vi hadde satt oss. Dette ble tilfellet med elaboration-fasen, hvor et prosjekt i et fag som gikk parallelt med hovedprosjektet fram til påske, tok mer tid og ressurser enn hva vi hadde beregnet. Dermed kom vi litt på etterskudd og måtte kjøre en mer intensiv construction-fase med tre korte iterasjoner. Med tapper innsats fra samtlige personer på gruppen klarte vi å komme ajour og kunne ta fatt på transition-fasen til planlagt tid. Selv om det ble hektisk den siste måneden synes vi at gjennomføringen av prosjektet var vellykket, og det at enkelte momenter forstyrrer inn på den opprinnelige planen er en nyttig erfaring som setter krav til omstilling.

7.2.1 Kort om RUP

Rational Unified Process er en iterativ systemutviklingsmodell som deler arbeid og iterasjoner i fire faser: Inception, Elaboration, Construction og Transition. I stedet for å fokusere på store mengder papirdokumentasjon, vektlegger RUP utvikling og vedlikehold av modeller og diagrammer som representerer systemet under utvikling. RUP tillater bruk av

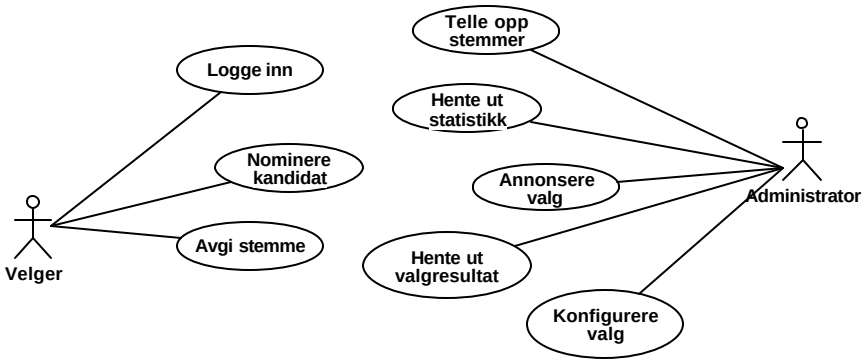
UML (Unified Modelling Language) på en effektiv måte og passer både store og små utviklingsmiljøer. For mer detaljer rundt RUP refererer vi til:
http://www.rational.com/products/whitepapers/100420.jsp

7.3 Vår anvendelse av RUP

7.3.1 Inception

Det første vi gjorde i inception-fasen var å arrangere et tidlig møte med oppdragsgiver for å få et innblikk i valgprosessen og rutinene rundt dette, siden ingen av gruppemedlemmene hadde særlig kjennskap til dette fra før. Vi fikk utdelt valgregler, eksempler på stemmesedler og annen informasjon som gjorde oss i stand til å forstå det forestående systemets oppgaver og utfordringer bedre. Selv om vi var enige om å dele applikasjonen inn i to moduler, så vi flere fordeler enn ulemper ved å kravspesifisere begge modulene som ett system i stedet for å stykke opp. Vi mente at denne måten å gjøre det på ville gjøre det lettere å se helheten i systemet, noe veileder også ga uttrykk for på et av de første veiledermøtene.

Vi satte i gang med å skrive et første utkast til visjonsdokument, hvor vi identifiserte aktørene 'Velger' og 'Administrator' og deres brukermål. Dette ble utgangspunktet for use case-diagrammet, som i sitt første utkast inneholdt åtte use cases (se figur **xx**). Når dette var på plass foretok vi en risikoanalyse, og prioriterte use casene etter vurdering av denne. Samtidig som Stian, Kent og Frode startet arbeidet med å skrive high level-beskrivelser for de use casene vi hadde funnet, satte Tom i gang med utvikling av brukergrensesnitt i Borland Delphi, siden dette er et raskere verktøy å lage brukergrensesnitt i enn verktøy som støtter Java.



Figur 7.1 Førsteutkast use case-modell

Kent og Stian gikk i gang med low level-detaljering av use casene 'Avgi stemme', 'Nominere kandidat' og 'Konfigurere valg', siden dette var use cases som oppnådde høy score og dessuten berørte ulike problemområder. Frode gikk i gang med å undersøke lovligheten av elektroniske valg, spesielt i høgskolesammenheng, og var på jakt etter informasjon om lignende applikasjoner ved andre høgskoler. Dessverre var elektroniske valgsystemer lite utbredt i Norge, men enkelte utenlandske firmaer hadde slike systemer til salgs, vel å merke nettbaserte løsninger. Vi prøvekjørte noen av disse uten å bli verken særlig overrasket eller imponert.

Etter hvert som detaljeringen av de tre utvalgte use casene nærmet seg slutten, laget vi system sekvensdiagrammer til disse og startet supplementsspesifikasjon og terminologi som senere skulle oppdateres fortløpende.

Inception-fasen ble en periode med mye innledende møteaktivitet. Vi holdt et møte med Harald Liodden i forbindelse med hvilken databasestruktur vi skulle velge og hvordan vi på enklest mulig måte kunne sikre velgers anonymitet og at en velger kun kan avlegge stemme én gang ved et valg. Liodden hadde samme forslag til sistnevnte problemer som vi hadde tenkt oss, nemlig at velger-tabellen ikke kobles direkte opp mot stemmeseddel-tabellen, og at et flagg settes i velger-tabellen når velgeren har stemt. Vi hadde også et uformelt møte med IT-tjenesten angående driftsmessige krav til systemet. Magne Reinsborg ga oss frie tøyler med hensyn til det meste, med den forutsetning at vi brukte Microsoft SQL Server. Dette var et databasehåndteringssystem som samtlige gruppemedlemmer hadde kjennskap til gjennom Databaser II-kurset fra før jul.

Parallelt med de innledende aktivitetene laget Stian en forenklet prototype av stemmeseddelen med mulighet for å avlegge stemme til en Microsoft Access-database med bruk av Java og JDBC. Dermed følte vi at noe av usikkerheten rundt JDBC ble borte og gjorde at vi kunne se mer optimistisk på den senere implementeringen.

Her følger notatet som ble skrevet ved milepælssjekken for LCO:
"Vi mener å ha nedlagt et innledende arbeid som indikerer en god kurs for det videre prosjektet. Vi tror at både veileder og oppdragsgiver har fått en pekepinn på hvor prosjektet står og hvilke formål som det ferdigutviklede systemet skal oppfylle. Vi har kartlagt de ulike aktørenes krav til systemet og gjort en prioritering av disse for å komme i gang med de mest risikofylte delene av prosjektet tidligst mulig. Videre har vi holdt møter med andre ressurspersoner på høgskolen for å få etablert et utviklingsmiljø og kommet til enighet om valg av database og annen programvare. Fase- og iterasjonsplan for elaboration er utarbeidet i plenum og setter tidsrammer som vi skal prøve å forholde oss til."

7.3.2 Elaboration

Elaboration-fasen bød på mange utfordringer. Spesielt begynte nå systemet å vokse seg større, og stadig nye idéer og krav til løsningen kom på plass i use case-diagrammet. Dette kom som et resultat av at både oppdragsgiver presenterte nye sider ved valgprosessen som vi ikke hadde tatt høyde for, og at detaljeringen av use cases i kombinasjon med at gruppemedlemmene gradvis modnet i forhold til emnet, inspirerte nye use cases. Noen av disse use casene var 'Avlyse valg', 'Endre valgkonfigurasjon' og 'Varsle kandidat om nominasjon'. Dette førte til at enkelte av oss måtte skrive high level-beskrivelser for disse, mens andre tok for seg nye use cases fra prioriteringsliste og detaljerte disse. Dette gjaldt i første rekke use casene 'Beregne valgresultat' og 'Beregne vektet valgresultat', som bar bud om at en omfattende og krevende implementasjonsjobb ventet senere i prosessen.

I samarbeid med Birgith Børthus fikk vi i stand et møte med leder av valgstyret og to representanter derfra, samt to representanter fra IT-tjenesten. Her ble det gitt en kort briefing av valgsystemet og kartlagt hvilke rutiner som ble brukt i forbindelse med manntallslisteproblematikk og IT-tjenestens rolle i dette. I korte trekk ble det bestemt at et Excel-regneark med liste over stemmeberettigede til det enkelte valg skulle sendes til IT-

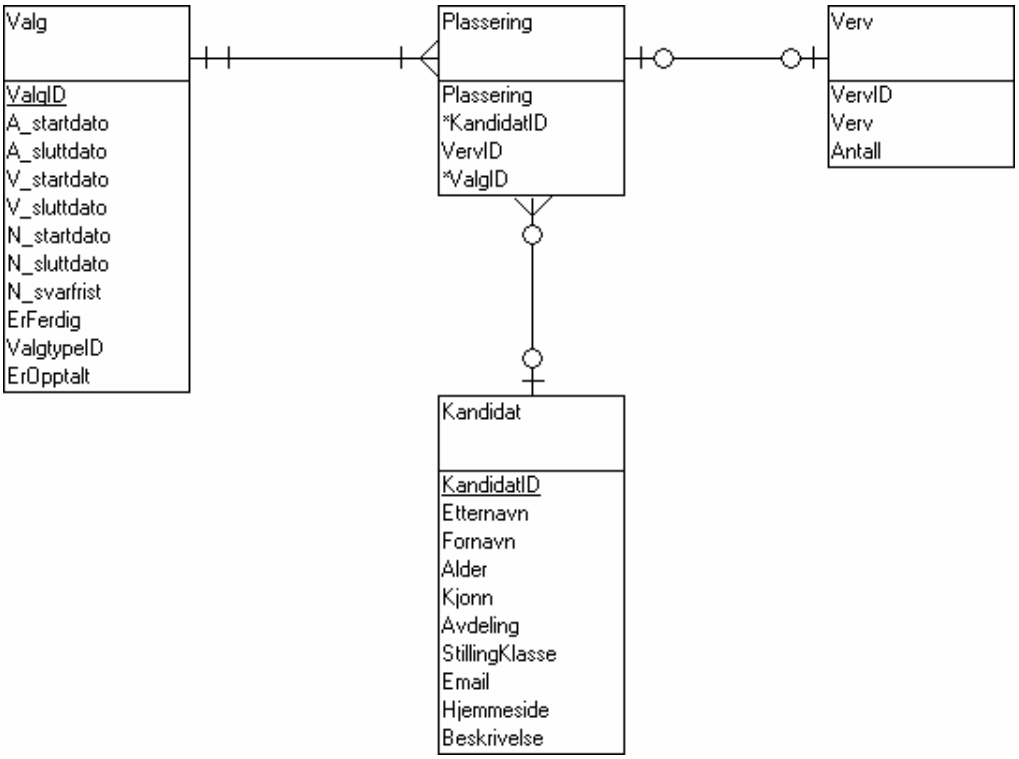
tjenesten, som ved hjelp av et script genererte en tekstfil med velgernes brukernavn som administrator kan laste inn under valgkonfigurasjonen.

Parallelt med at Tom sluttførte GUI-utviklingen i Delphi fikk gruppen ordnet rettigheter på egen SQL Server-database, slik at denne var klar til bruk når design og utvikling av database var ferdig. Use case-diagrammet, risikoanalysen og visjonsdokumentet ble fortløpende oppdatert etter hvert som manntallslisteordningen ble avklart og det ble bestemt å skille på bruker som nominerer og bruker som stemmer ved et valg. Use casene vi nå hadde gjorde at vi hadde et brukbart utgangspunkt for å lage et førsteutkast til konseptuelt klassediagram (domenemodell).

Videre i fasen forsatte vi å detaljere ned ytterligere use cases parallelt med at vi begynte på designmodellen. Sekvensdiagrammer ble laget for use casene 'Avgi stemme' og 'Konfigurere valg', og metodene og klassene som ble brukt i disse ble satt inn i design klassediagrammet (DCD). Samtidig begynte Stian å utarbeide figurer for logical view, deployment view og implementation view. Disse skissene ble skrevet ut slik at alle kunne komme med innspill til systemarkitekturen og eventuelle endringer som burde gjøres i diagrammene for å representere arkitekturen på best mulig måte. Tom var ferdig med brukergrensesnittet og vi lot et utvalgt testpanel få prøve de to modulene og komme med fortløpende kommentarer og tilbakemeldinger underveis. For mer informasjon fra GUI-testing refereres det til vedlegg G.

I denne perioden ble også PC-er fordelt på de ulike prosjektgruppene, og dette gjorde at en dag eller to ble brukt til å etablere et utviklingsmiljø på A204. Dette forløp relativt problemfritt.

Helt i slutten av elaboration-fasen begynte vi å notere ned hvilken informasjon som måtte lagres i databasen, og Kent gikk i gang med å normalisere tabellene. Dette ga oss en initiell ER-modell, som ble grundig diskutert. Det oppstod en liten diskusjon rundt hvorvidt det var mer hensiktsmessig å gå tilbake en normalform i tabeller som inneholdt få attributter, og som i tillegg kunne slås sammen med andre tabeller med relatert informasjon for å gjøre databasen mer effektiv og forenkle spørringene i applikasjonen.



Figur 7.2 Utdrag fra en tidlig versjon av ER-modellen

Figur 7.2 viser deler av modelleringen på et tidlig tidspunkt. Denne løsningen ble begrunnet med at en kandidat kunne stille til flere valg, og at det dermed ville være fornuftig å ha denne kandidatens informasjon lagret én gang. Tabellen 'Plassering' tar vare på plasseringen til kandidaten ved et spesielt valg og hvilket verv han er tildelt.

Etter hvert ble vi enige om at det var mer hensiktsmessig å legge attributtene 'Plassering', 'ValgID' og 'VervID' i kandidattabellen, og la denne tabellen være direkte knyttet til henholdsvis valg- og vervtabellen. Dermed kom de to sistnevnte attributtene inn som fremmednøkkel i 'Kandidat', og 'ValgID' og 'KandidatID' ble sammensatt primærnøkkel (se figur XX for fullstendig ER-modell). Begrunnelsen for hvorfor vi endte opp med denne løsningen er at den i tillegg til å gjøre SQL-spørringer enklere og mindre ressurskrevende (færre tabeller som eventuelt må joines) også gir muligheten for at en kandidat ved et bestemt valg kan ha en annen beskrivelse enn ved et annet valg. Hva som er relevant å ha med i en kandidatbeskrivelse vil trolig variere fra valg til valg.

Her følger notatet som ble skrevet ved milepælssjekken for LCA:
"Gruppen er kommet fram til at vi ikke kan si å ha nådd LCA-milepælen til fastsatt dato. Vi mener at vi har skaffet oss en rimelig god oversikt over prosjektet, men har ikke detaljert ned en så stor andel av use cases som ønsket, hovedsakelig fordi nye krav har framkommet gjennom kravspesifiseringsprosessen og gjort arbeidsmengden større. Gruppen har heller ikke lykkes i å fullføre arbeidet med en stabil arkitektur på nåværende tidspunkt. Derimot er både visjonsdokument og de fleste kravene stabilisert. Utvikling av database og prototyper for kommunikasjon mellom Java-utviklingsmiljø og database (JDBC) er fullført. Gruppen

har tatt tidsproblemene som har oppstått til følge og modifisert planene fremover slik at vi raskest mulig kan være på riktig kurs igjen."

7.3.3 Construction

Construction-fasen ble inndelt i tre intensive iterasjoner. Opprinnelig var tanken å ha to lengre iterasjoner, men siden et prosjekt i faget klient- og serversideprogrammering sammen med eksamen i samme fag tok en del mer tid enn beregnet, og påskeferien kom den påfølgende uken, bestemte vi oss for å ta med en iterasjon ekstra. Begrunnelsen for dette var at vi dermed hadde muligheten til å stykke opp arbeidet mer og dermed ha klare definerte mål som måtte nås ved slutten av hver iterasjon, noe som ble viktig å kontrollere på dette stadium da prosjektet lå noen uker etter opprinnelig skjema. Siden maskinene vi hadde fått utdelt til hovedprosjektet ikke hadde ressurser nok til å kjøre JBuilder 6, måtte vi ta i bruk en utfordrende arbeidsform hvor hver enkelt jobbet hjemme på sine private PC-er. Hver og én av oss fikk utdelt use cases som skulle implementeres, og etter tre-fire dager møttes vi på grupperommet og evaluerte det utførte arbeidet. Vi prøvde å holde så god kontakt som mulig slik at alle var på rett sti. Hjemmearbeid medførte at vi måtte bruke en Microsoft Access-database, siden ingen av oss hadde SQL Server hjemme.

De ferdigkodede delene av systemet ble integrert og testet og nye arbeidsoppgaver ble fordelt. Med få unntak hadde Tom hovedansvaret for koding av avstemmingsmodulen, mens Stian, Kent og Frode hadde fokus på administrasjonsdelen av systemet. Under arbeidet benyttet vi også litteratur fra skolens bibliotek, siden vi hadde liten erfaring med bruk av JDBC fra før av. Dessuten hadde vi uformelle møter med Øivind Kolloen som var svært behjelpelig med Java-relaterte spørsmål som gruppen hadde.

Underveis i kodingsprosessen ble nye behov for lagring av informasjon oppdaget, og enkelte nye use case kom også fram. Dette førte til at både ER-modell, database, use case-, domene- og designmodell måtte oppdateres. Vi valgte bevisst å oppdatere artifakter som ble påbegynt tidligere i prosessen. Selv om RUP har en retningslinje som tilsier at oppdateringer kun trengs å gjøres for artifakter som har nytteverdi for den videre utviklingen, mener vi at det er en fordel for de som skal lese systemdokumentasjonen og eventuelt videreutvikle løsningen at eksempelvis klassediagrammet er oppdatert og speiler et korrekt bilde av konseptet.

Tom hadde også ansvaret for mailutsendingsmetoder og innlogging mot NT-domene, som er grunnleggende rutiner for å få systemet til å fungere etter intensjonen. Innimellom kodingsarbeidet jobbet Kent og Frode med å detaljere ned use casene 'Hente ut valgresultat' og 'Hente ut opptellingsresultat'. Stian hadde hovedansvaret for algoritmer rundt stemmeopptellingen, som har vært en tidkrevende arbeidsoppgave med krav til testing utover det vanlige.

Etter hvert som fasens siste iterasjon nærmet seg slutten begynte det totale systemet å nærme seg komplett, selv om en del finpuss gjenstod. Vi oppdaterte SQL Server-databasen på skolen og forsøkte å koble opp mot SQL Server med en driver vi lastet ned fra internett. Denne driveren fikk problemer når vi skulle hente ut data fra databasen i et Java-resultatsett. Vi fikk tak i Kolloen som heller ikke klarte å løse problemet. Vi valgte derfor å opprette en SQL Server-datakilde gjennom Windows' kontrollpanel. Dette fungerte bra. Derimot viste det seg at SQL-funksjonen NOW() som vi hadde brukt mot Access-databasen, ikke eksisterte

i SQL Server. Etter å ha lest igjennom en del dokumentasjon fant vi ut at SQL Server har en GetDate()-funksjon som returnerer dagens dato (og klokkeslett).

Her følger notatet som ble skrevet ved milepælssjekken for IOC:
"Vi mener å ha nådd IOC-milepælen til fastsatt dato. Gjennom intensivt og målrettet arbeid har vi utviklet majoriteten av komponentene og integrert disse i tilhørende moduler. All produsert kode er blitt testet underveis, spesielt etter integreringen av de ulike delene som er utviklet, slik at vi føler oss trygge på at systemets komponenter fungerer sammen. Applikasjonen fungerer godt sammen med SQL Server-databasen. Vi har også utarbeidet et utkast til brukerveiledning som vi skal bygge videre på i neste fase. Vi mener at produktet nå er klart for grundigere testing og dokumentasjon."

7.3.4 Transition

Transition-fasen ble en arbeidsom periode, hvor mange aktiviteter ble kjørt parallelt for å komme i mål med både applikasjonen og dokumentasjonen. All funksjonalitet ble testet under flest tenkelige scenarier, noe som resulterte i en endelig versjon som fungerer stabilt og gir fornuftig og informativ respons til brukeren. Parallelt med at Tom og Stian ferdigutviklet og testet modulene, startet Kent og Frode med innholdsfortegnelse til hovedprosjektrapporten. Denne ble vurdert av veileder, som hadde forslag til elementer som burde flyttes og påpekte essensielle punkter og mindre viktige avsnitt. Blant annet ble det foreslått å flytte systemarkitekturen inn i designdokumentet og flytte beskrivelse av prosessen ut i eget kapittel.

Kent har hatt ansvaret for brukerveiledningen og brukt mye tid på den, mens Tom brukte deler av fasen til å lage Java Servlets for presentasjon av opptellingsrapport og valgresultat.

Siden IT-tjenesten ikke hadde planer om å installere JDK 1.4 og valgsystemet før i sommer, har vi ikke hatt mulighet til å få satt systemet i en reell driftssituasjon. Derimot har vi simulert miljøet ved å legge avstemmingsmodulen på en av PC-ene på grupperommet (hvor nødvendig programvare var installert), og aksessert denne fra PC-er på datalab A209. Dette har fungert fint og i henhold til intensjonen. Når det gjelder webpresentasjonen, vil heller ikke den bli lagt ut før i sommer. Det må settes opp en server med driver for Servlets hvor denne presentasjonen kan ligge. Dette har ikke IT-tjenesten mulighet til å gjøre før etter prosjektets slutt. Dermed får man kun kjørt opptellingsrapport og valgresultat gjennom JBuilder inntil videre.

Her følger notatet som ble skrevet ved milepælssjekken for PR:
"Gruppen har testet både administrasjonsmodulen og avstemmingsmodulen grundig og dokumentert dette. Systemet er klart til drift så fort IT-tjenesten har installert nødvendig programvare. Vi har utarbeidet en brukerveiledning som skal sørge for at brukerne av systemet får en god forståelse for hva de ulike menyvalgene har å tilby av funksjonalitet."

Oppsummering

Gjennom prosessen har vi fått god anledning til å benytte UML. Vår erfaring tilsier at det er en fordel å ha et rammeverk som støtter UML-bruk. Notasjonen byr på uttrykksfulle diagrammer som utfyller tekstlige beskrivelser rundt use case-diagram, klassediagram,

sekvensdiagram og arkitekturskisser. Vi mener at figurbruk sier vel så mye som selve teksten rundt et tema.

Prosessen har også gitt oss muligheter til å bruke ulike arbeidsformer. Vi har jobbet mest gruppevis, men også en god del parvis og enkeltvis hvor dette har vært nødvendig. Spesielt etter hvert som alle har fått et godt innblikk i hvilke oppgaver og utfordringer systemet skal løse har det vært aktuelt å jobbe mer parvis og alene for å utnytte ressursene best mulig. Mens gruppejobbing har satt krav til kommunikasjon og evne til å komme fram til gode løsninger gjennom drøftinger og diskusjoner, har jobbing på egenhånd satt krav til struktur og selvstendighet hos den enkelte. Vi synes at det å lære seg å takle de ulike arbeidsformene har vært lærerikt og mener at dette har gitt oss et utbytte utover det rent faglige.

Det at vi har valgt å dele applikasjonen i to moduler har hatt størst innvirkning under construction-fasen. Siden både use case- og domenemodellen representerer systemet som helhet har ikke modulinndelingen hatt særlig mye å si for tidligere faser i prosjektet.

Gjennom arbeidet med hovedprosjektet har vi erfart nytten i å ha lagt et godt systemutviklingsarbeid til grunn. Spesielt har arbeidet med å kravspesifisere gitt oss en verdifull innsikt i systemet som har ført til at majoriteten av koden vi har utviklet finnes igjen i det ferdige systemet.

Hvis vi skulle trekke fram en erfaring som har vært spesielt nyttig må det være at utvikling av en stabil database har tatt en god del mer tid enn planlagt. Databasen har først blitt komplett i begynnelsen av transition-fasen. For prosjekter i framtiden har vi lært at det lønner seg å koble opp mot databasen som faktisk skal brukes på et tidlig stadium, fordi Access og SQL Server i vårt tilfelle har små forskjeller i SQL-syntaks som tross alt kan lage problemer og ta tid å løse.

8 Arbeidsmetoder og resultater (Avslutning)

8.1 Drøftinger og diskusjoner

8.1.1 Vurdering av resultater

Vi er alle enige om at vi har nedlagt et betydelig arbeid i å utvikle en applikasjon som er en tilnærmet fullverdig erstatning for dagens manuelle valgordning. I tillegg til å utvikle en intuitiv løsning har hovedmålet vært å lage et system som åpner for at det meste av valgkonfigurasjonen gjøres av administrator, slik at nye typer valg og nye grupper av velgere kan tas med dersom slike skulle oppstå i framtiden. Et annet punkt vi har vektlagt er å gjøre systemet så levedyktig som mulig ved å gi muligheter for å legge til påloggingsdomener, endre adresser for valgresultat- og opptellingsrapportsider på internett, endre mailserver, NT-domene (for eksempel 'studenter.hig.no') og portnummer gjennom administrasjonsmodulen. Det ville vært kjedelig dersom IT-tjenesten bestemte at høgskolen skulle benytte en ny mailserver og på den måten kunne lamme deler av nominasjonsprosessen. Vi håper at vårt valgsystem har styrt unna en del av fallgruvene som kan sette et produkt ute av spill.

Selv om vi i forprosjektrapporten foreslo å utvikle en nettbasert avstemmingsmodul i tillegg til en applikasjon, var det alltid tanken at dette var en mulig utvidelse av systemet dersom det skulle bli tid til overs. Vi mener at kravene vi stilte til løsningen i forprosjektrapporten samsvarer godt med det faktiske resultat, og at kriterier for anonymitet, demokrati og nøyaktighet er oppfylt.

8.1.2 Alternativer, muligheter og valg underveis

Applikasjon kontra internettløsning
I begynnelsen av utviklingsforløpet måtte vi ta en avgjørelse om vi skulle utvikle en Java-applikasjon til bruk på høgskolens arbeidsstasjoner eller om vi skulle lage en internettløsning. Oppgavebeskrivelsen sa i utgangspunktet at det skulle utvikles en valgapplikasjon, men de undersøkelser vi gjorde på internett angående elektroniske valgsystemer avslørte at nettbaserte løsninger nok var den mest brukte, selv om den elektroniske måten å holde valg på fortsatt må anses å være i startgropen.

Den store fordelen ved en internettløsning er selvsagt at man ikke trenger å være på høgskolens område for å kunne nominere kandidater eller avgi stemme. Spesielt er dette en aktuell problemstilling for sykepleierstudentene som ofte er på ekskursjoner og dermed ikke befinner seg på høgskolen i enkelte valgperioder. Dessuten ville et slikt system innebære et minimum av installasjonsproblemer og driftsrelatert arbeid for IT-tjenesten.

Den store ulempen med ovennevnte alternativ er i første rekke sikkerheten. Man må raskt innfinne seg med å ta andre hensyn når systemet ligger "tilgjengelig" for hele verden sammenlignet med en applikasjon som brukes på skolens intranett. Vi setter "tilgjengelig" i anførselstegn fordi man selvsagt hadde sørget for at kun høgskolens studenter og ansatte kunne logge inn, men samtidig er det vanskelig å gjøre en internettløsning "innbruddssikker".

Et annet element som gikk i applikasjonsalternativets favør er muligheten for å kunne benytte dagens rutine for forhåndsstemming, hvor velgerne leverer sine forhåndsstemmer til

administrator eller en annen representant fra valgstyret, og administrator legger inn disse stemmene i administrasjonsmodulen. Dermed ble vi enige om å utvikle en elektronisk valgapplikasjon i Java.

Stopp av funksjonalitet
Et godt stykke ut i elaborationfasen fant vi ut at mengden av funksjonalitet begynte å nå en grense for hva som var realistisk å få gjennomført. Vi valgte derfor å sette en stopp for videre forslag til utvidelser av systemet, slik som valgannonsering, for å kunne fokusere på de utfordringer vi stod overfor og ha en mulighet til å ende opp med en applikasjon som kunne tas i bruk. Vi fant ut at det er veldig enkelt å love ut mye, men ikke fullt så greit å realisere alle ønskene. Heldigvis ble denne avgjørelsen tatt tidsnok slik at systemet ikke ble skadelidende.

Passord
Av åpenbare sikkerhetsmessige aspekter ble vi tidlig enige om at brukere av avstemmingsmodulen måtte logge seg på før man eventuelt avgir sin stemme ved et valg. Det er et vanlig scenario at enkelte PC-er på datalabene er pålogget, men hvor brukeren har gått et annet sted for en liten stund. I et slikt tilfelle ville det vært mulig for hvem som helst å starte opp valgapplikasjonen og avgi stemme for denne brukeren, noe som selvfølgelig er uakseptabelt.

Etter at avgjørelsen om innloggingsbehov var tatt hadde vi to alternativer
1) Brukeren logger seg på med brukernavn og passord som er eget for valgapplikasjonen
2) Brukeren benytter samme brukernavn og passord som ved pålogging på høgskolens maskiner

Vårt valg falt på alternativ 2. En fordel er at man slipper å lagre egne brukernavn og passord for hver velger i databasen, mens det mest tungtveiende fortrinnet ligger i at brukere slipper å gå rundt å huske på enda et brukernavn og passord. Vi tror at bruk av samme brukernavn og passord som ved vanlig pålogging vil gjøre det litt mer attraktivt å bruke systemet. Dette er en beslutning som vi mener er i tråd med målsetningen om å øke valgoppslutningen ved skolen.

For å sikre at en velger bare kan avgi stemme én gang ved samme valg, var vi først inne på tanken om å registrere når velgeren var pålogget på systemet, ved å sette et ErInnlogget-attributt i databasen til 1. Et graverende problem med denne måten å gjøre ting på er hvis applikasjonen skulle låse seg som følge av at andre programmer som kjøres parallelt på brukerens maskin får problemer, eller at databaseserveren krasjer. Da vil aldri ErInnlogget-attributtet bli tilbakestilt (til 0), og dermed får ikke velgeren mulighet til å prøve å stemme igjen ved aktuelt valg, fordi han er registrert som allerede innlogget. Derfor måtte vi gå vekk fra denne løsningen og angripe problemstillingen på en annen måte.

Den endelige løsningen ble å kjøre alle INSERT- og UPDATE-setningene i forbindelse med stemmeavgivning i én transaksjon. Dette i kombinasjon med SQL Servers transaksjonshåndtering sikrer at en velger som prøver å stemme samtidig fra to ulike arbeidsstasjoner kun får avgitt stemmen én gang. Den transaksjonen som kommer først av de to vil utføres i sin helhet før den neste påbegynnes, og dermed vil databasen allerede har registrert at samme velger har avlagt stemme og dermed avbryte transaksjon nummer to.

Valgresultat og rapporter

I samråd med oppdragsgiver diskuterte vi hvor tilgjengelig valgresultat og rapport for opptelling av valg skulle være. Vi ble enige om at valgresultat, både nye og gamle, skulle være tilgjengelig for alle, mens opptellingsrapporten er forbeholdt administratoren(e). Derimot vil den framtidige ordningen være som dagens, nemlig at kandidater ved et valg kan be administrator om å få se opptellingsrapporten for valget de stilte til.

Når dette var avklart hadde vi valget mellom å presentere valgresultat og opptellingsrapporter i selve applikasjonen, eller på en dedisert nettside for valg ved høgskolen. Vi fant ut at resultater og rapporter blir mer oversiktlige, penere og enklere å skrive ut på web enn i et vindu på applikasjonen, og at menyvalgene i systemet som angår valgresultat og opptellingsrapport fungerer som en link til nettsidene for disse.

8.2 Fremtidige utvidelser og forbedringsmuligheter

En mulig utvidelse og forbedring av systemet vi har utviklet vil være en mer avansert rapportutforming for administrasjonsmodulen. Dette er en del av systemet som i første rekke har blitt utviklet med henblikk på ren funksjonalitet, altså at administrator skal ha mulighet til å skrive ut kandidatlister, stemmeseddellister og opptellingsresultater på en enkel og grei måte. Hadde vi hatt mer tid til rådighet ville nok en del av ressursene blitt satt på å gjøre utskriftene penere og mindre avhengig av eksterne programmer som Notepad i vårt tilfelle. I tillegg kan man lage flere oversikter som kan være av interesse enn de som dagens system har.

En annen utvidelse av systemet kan være å lage og presentere mer statistisk tallmateriale enn bare valgresultater fra avholdte valg. For eksempel kan man lage oversikter som sammenligner valgoppslutningen fra år til år, om økning eller nedgang i oppslutning gjelder for alle velgergrupper eller bare for enkelte, hvorvidt enkelte valg engasjerer flere studenter og ansatte til å stemme enn andre. Bare fantasien setter grenser på dette området.

Ser vi litt lengre fram i tid vil et valgsystem over internett bli en aktualitet. Som tidligere nevnt var en internettløsning et alternativ vi vurderte, men av ulike årsaker slo fra oss. Men med fordelene som en internettløsning fører med seg vil dette kunne være et mulig hovedprosjekt noen år fram i tid. Da vil man i tillegg ha dokumentasjon og erfaringer fra en elektronisk valgapplikasjon, slik at den neste løsningen skulle ha gode muligheter til å favne om de fleste krav og forhåpninger man stiller til et elektronisk valgsystem.

8.3 Evaluering av eget arbeid

Innledning

Med en gruppe på fire personer vil det alltid være en utfordring å få prosjektdeltakerne til å fungere best mulig sammen slik at alle drar i samme retning og bidrar til at humøret og motivasjonen holdes oppe. Derfor har vi lagt oss på en linje der alle respekterer alle og hvor kravet om å bli hørt innebærer at man forventes å høre på andre gruppemedlemmers synspunkter. Dette har fungert veldig godt og har resultert i et arbeidsmiljø det har vært morsomt og faglig stimulerende å ta del i.

Organisering

Prosjektgruppen har hatt en prosjektleder som har hatt ansvaret for å fordele arbeidet, ta endelige beslutninger, holde internmøter, påminne om tidsfrister og tidspunkter for skriving av statusrapport. I tillegg har gruppen hatt en loggansvarlig, som har tatt notater under veileder- og oppdragsgivermøter og ført møtereferater fra disse og ukentlige internmøter.

Fordeling av arbeidet

Så langt det har vært mulig har vi prøvd å fordele arbeidsoppgaver slik at hver prosjektdeltaker har hatt omtrentlig samme arbeidsmengde som resten av gruppemedlemmene. Noen ganger har vi fordelt oppgaver på ukentlig basis, andre ganger har arbeidsoppgavene blitt fordelt daglig. Vi har i utstrakt grad forsøkt å velge utfordringer etter eget ønske, i den tro at jo mer engasjert personen som utfører oppgaven er, desto mer fruktbart blir resultatet. Stort sett har dette gått hånd i hånd med å bruke den enkeltes kompetanse der hvor den har gjort størst nytte for seg.

Tom har hatt utvikling av brukergrensesnitt i Delphi til bruk på testpanel og utvikling av avstemmingsmodul som hovedansvarsområder. Frode, Kent og Stian har hatt hovedansvaret for systemutviklingsdelen av prosjektet, mens alle tre har bidratt på programmering av administrasjonsmodul. Etter hvert som prosjektet nærmet seg slutten tok Frode og Kent tak i prosjektrapporten mens Stian fullførte uferdige oppgaver i administrasjonsmodulen. Alle fire har deltatt i utvikling av databasestruktur og testing av begge programmoduler.

Den endelige timeloggen ble slik:

Tom Fladsrud	472,5 timer
Kent Elverum	489,5 timer
Frode Elverum	506,0 timer
Stian F. Lereng	573,0 timer
Totalt for EvigElev	2041,0 timer

Prosjekt som arbeidsform

Å arbeide sammen i et prosjekt setter store krav til samarbeidsvilje og evne til å løse problemer sammen. Vi synes at denne arbeidsformen er spennende, kravstor og framtidsrettet. Den har satt krav til planlegging og fordeling av ressurser og har satt oss i stand til å vurdere andre synspunkter enn våre egne og komme fram til løsninger som er til sluttproduktets beste.

Subjektiv opplevelse av hovedprosjektet

Alle gruppemedlemmer er enige om at arbeidet med hovedprosjektet har gitt hver enkelt av oss en lærerik og fullverdig avslutning på ingeniørstudiet. Selv om vi er fire ulike personer med forskjellig bakgrunn og relativt lite kjennskap til hverandre på forhånd, har vi blitt en sammensveiset gjeng som har høstet både faglige og sosiale gevinster gjennom hovedprosjektet. Å arbeide såpass intenst med samme overordnede problemstilling i et halvt år er en erfaring vi ser verdien i, og som har gitt oss et annerledes utbytte enn den

forelesningsbaserte undervisningen vi er vant med. Dessuten har oppgaven med å utvikle et elektronisk valgsystem vært en interessant utfordring.

8.4 Konklusjon

Gjennom et halvt års arbeid har vi hatt muligheten til å gjennomføre et interessant utviklingsprosjekt fra idéstadiet til en fullverdig og dokumentert applikasjon. Prosjektet har dermed satt krav til planleggingsevne, koordinasjon av arbeidsoppgaver og veloverveide beslutninger. I tillegg har det satt både motivasjon og læringslyst på prøve. Det å måtte se et halvt år framover i tid er et perspektiv som er nytt for oss som studenter i prosjektsammenheng.

Vi mener å ha kommet fram til en gjennomtenkt og brukervennlig løsning som letter arbeidet med gjennomføringen av valg betraktelig. I tillegg til å være arbeidsbesparende vil løsningen eliminere feil som kan forekomme i den manuelle ordningen, herunder feilutfylte stemmesedler og opptellingsfeil. Vi er godt fornøyde med å ha utvidet systemet til å innlemme nominasjonsproblematikken, siden vår oppfatning er at et vellykket valg må ha egnete kandidater som har støtte blant ansatte og studenter. Det har vært motiverende å arbeide med en problemstilling hvor fortrinnene ved et datasystem har vært så synlige, og hvor elektronisk stemmeopptelling av utvalgte valg har vist at det har forekommet feil i den manuelle opptellingen som kunne ha vist seg utslagsgivende.

Avslutningsvis må vi nevne at Hans Engenes' og valgstyrets glød og engasjement under utviklingsforløpet har gjort at viljen til å stå på for å komme i mål med en applikasjon som kan tas i bruk, har vokst i takt med systemet. Det har vi satt pris på!

9 Litteraturliste

Deitel & Deitel: *Java How To Program*, 3rd Edition, 1999.

Y. Daniel Liang: *Rapid Java application development using JBuilder 4/5/6*, 2nd Edition, 2002.

Craig Larman: *Applying UML and Patterns*, 2nd Edition, 2002

George Reese: *Database programming with JDBC and Java*, 2nd Edition, 2000.

Anthony T. Mann: *Microsoft SQL Server 2000 for Dummies*, 2001

Johnsen, Jørdre & Andenæs: *Softcross*, Hovedprosjekt våren 2001 ved HiG

SUN Java Documentation
http://www.hig.no/at/data/progwww/jdk1.2/docs/api/index.html

10 Vedlegg