HOVEDPROSJEKT:

HELSEKORT I OMSORGSSEKTOREN

FORFATTERE: Kai Hallingstad
Frode Andersen
Anders Helling

Dato: 22.mai 2001

Sammendrag av hovedprosjekt

Tittel:	Helsekort for omsorgssektoren	Nr. :	
		Dato : 21.05.2001	
Deltakere:	Kai Hallingstad		
	Frode Andersen		
	Anders Helling		
Veileder:	Ivar Farup		
Oppdragsgiver:	ErgoSolutions		
	Hans Mustadsgate 31, 2821 Gjøvik		
Kontaktperson:	Asbjørn Hovstø		
Stikkord (4 stk)	smartkort, PDA, java, omsorgssektor		
Antall sider: 95	Antall bilag:	Tilgjengelighet (åpen/konfidensiell): åpen	

Kort beskrivelse av hovedprosjektet:

Denne oppgaven tar for seg utviklingen av et smartkortsystem som skal lette hverdagen for omsorgssektoren. Hensikten er å få pasientenes hjemmejournal, som i dag er på papirform, over på et smartkort. Pleieren skal benytte en håndholdt PC for å lese og oppdatere data på kortet.

Vi har utviklet to applikasjoner. Den ene initierer et nytt pasientkort, og den andre kjører på den håndholdte PC'en som pleieren har med seg ved alle pasientbesøk. Pleieren kan med denne lese pasientens smartkort og på denne måten hente ut og oppdatere data i pasientens journal.

FORORD

Denne oppgaven er utarbeidet som et hovedprosjekt ved Høgskolen i Gjøvik. Arbeidet nedlagt i dette prosjektet tilsvarer fire av totalt seksti vekttall ved dataingeniør studiet.

Vi er tre studenter som har jobbet sammen om dette prosjektet. Arbeidet startet rett før julen 2000 og har gjennom siste semester gitt oss et omfattende innblikk i hvordan et større prosjekt arter seg. Dette har vært en fin anledning til å benytte seg av de fleste fagene i en større sammenheng.

Valg av oppdragsgiver falt på ErgoSolutions, et firma som lenge har jobbet med å utvikle ulike smartkortløsninger. Vi fant smartkortoppgaven interessant fordi det var helt ny teknologi, samtidig som at dette er noe vi tror vil bli en del av fremtiden. Vi ønsker å rette stor takk til Asbjørn Hovstø ved ErgoSolutions på Gjøvik. Han har vært til stor hjelp gjennom hele prosjektet. Vi vil også takke Sissel Slettum Bjerke ved Gjøvik Kunnskapspark. Hun har vært behjelpelig med å sette oss i kontakt med fagpersonell i omsorgssektoren.

Ivar Farup har vært veileder for oss under hele prosjektetperioden. Han har alltid satt av tid til oss når vi har hatt behov for det. Vi har satt stor pris på samarbeidet. Takk for hjelpen!

Vi retter også en kjempetakk til Stabburet og Coca Cola. De har vært til stor hjelp i en vanskelig tid.

Gjøvik, 22.mai 2001

Kai Hallingstad	Frode Andersen	Anders Helling

INNHOLD:

FIGURLISTE

1 INNLEDNING

1.1 Oppgavedefinisjon

Slik omsorgstjenesten fungerer i dag er den preget av mye muntlig informasjon. Hver pasient har sin egen hjemmemappe som inneholder de mest sentrale pasientopplysningene. Hjemmemappen befinner seg hjemme hos pasienten. Det finnes også en elektronisk journal som befinner seg på kontoret til omsorgstjenesten. Det er ikke mulig for hjemmesykepleierne å sette seg inn i hele denne før hvert besøk.

Fra politisk hold er det kommet ønsker om at det skal brukes elektroniske journaler i helsetjensten [9]. I 1997 ble det skrevet en utredning hvor elektroniske pasientkort ble satt i fokus [10]. I løpet av en 4-5 års periode er det snakk om at det skal lages et helsekort som inneholder hele helsejournalen. Meningen er at folk skal ha med seg et slikt kort akkurat som du i dag har med deg et minibankkort. På denne måten vil det bli enklere for alle instanser å ha journalen tilgjengelig til enhver tid.

Formålet med prosjektet er å få fram smartkortbaserte personkortløsninger for flere områder, både i primærhelsetjenesten og for sykehusenes behov. Informasjonen mellom sykepleierne slik den er i dag øker risikoen for feilinformasjon. Nå håndskrives to nesten identiske journaler, en journal som ligger hjemme hos hver pasienten og en tilsvarende kopi av denne journalen som ligger inne på hovedkontoret til hjemmesykepleien. Det ligger et stort potensial i å effektivisere dette, og dermed frigjøre tid til pleie, og sikre at opplysninger er korrekte.

Oppgaven går ut på og lette arbeidet til sykepleierne ved hjemmebesøk hos pasienter. Dette kan gjøres ved hjelp av smartkort, som skal inneholde deler av pasientens journal.

Administrativt kan pasientkort brukes for å forenkle registrering og tilrettelegge pasientbehandling. Tidligere har ei hovedprosjektgruppe laget kort som kan brukes for trygdeopplysninger, ved betaling av helsetjenester og som kvittering for egenandel. Vår oppgave blir å lage et system for hjemmesykepleien som skal gjøre kommunikasjonen enklere mellom pasient og pleier, og pleierne seg imellom.

Oppgaven går ut på å bruke en ferdig databaseapplikasjon, GemDB [5] og lage et databasedesign for optimal plassutnyttelse av helsejournal på kortet. Programmere en java-applikasjon som kan tilpasses ODBC-grensesnitt på PC og håndholdt PC. Vi skal lage et demonstrasjonsprogram som skal vise mulighetene, men ikke et ferdig produkt. Vi skal også utføre en EAL-trusselanalyse. En slik analyse vil si at vi skal se på mulig angrep mot kort og applikasjon. Trusselanalysen vil være en forholdsvis liten del av oppgaven vår, og vi vil kun se på enkle angrep mot smartkort og applikasjon. Skulle vi gjennomført en komplett trusselanalyse vil det være et helt prosjekt i seg selv.

Løsningen skal fungere slik at når en hjelpe-/sykepleier kommer til en pasient skal kortet til pasienten leses av. Pleieren får da informasjon om deler av pasientens journal, medikamentdosering og eventuelt hva som ble gjort av den pleieren som var der sist. Ved besøkets slutt lagres alle endringer som pleieren har gjort på kortet. Disse dataene blir nå tilgjengelige for neste pleier som kommer på besøk.

Vår hovedoppgave blir å finne ut hva slags data det er fornuftig å ha på kortet, og kode java-applikasjonen som lagrer og henter data. Vi skal også lage en applikasjon som oppretter en tom database på kortet slik at det er klart til bruk.

Vi må begrense oppgaven kraftig for at vi skal ha mulighet til å komme fram til et fornuftig resultat. Vi prioriterer applikasjonen som skal lagre data på kortet, og skal kun konsentrere oss om hjemmesykepleien. Det vil si når hjelpepleier/sykepleier reiser på besøk til en pasient. Lege- /sykehusbesøk skal ikke være med.

1.2 Målgruppe

Rapporten er beregnet på ErgoSolutions som er oppdragsgiveren vår. Vi prøver å gjøre rapporten leselig for andre også. I deler av rapporten blir det lagt stor vekt på å forklare faguttrykk. I mer tekniske deler er faguttrykk brukt mer eller mindre uten forklaring.

Applikasjonene som vi utvikler er først og fremst beregnet på omsorgssektoren. Databaseoppretter applikasjonen(DBO) er beregnet på de som jobber med administrasjon. Den skal kun brukes for å opprette databasen. Hjemmejournal-applikasjonen(HJA) skal brukes av hjemmesykepleierne i det daglige arbeidet.

1.3 Faglig bakgrunn

Prosjektgruppen består av tre studenter ved Høgskolen i Gjøvik. Alle går siste året på dataingeniørstudiet. Vi har i løpet av disse årene hatt en god del forskjellige fag som hjelper oss godt på vei mot målet. Spesielt kan vi nevne programmeringsfaget ”Programmering mot www” som er det eneste javakurset vi har. Systemutvikling er også et fag som bør nevnes. Dette faget har vært til stor hjelp i planleggingsfasen.

1.4 Arbeidsformer

Vi valgte å samarbeide mest mulig i starten. Det er viktig at alle får være med på planleggingen av prosessen. I planleggingsfasen hadde vi også meget god kontakt med veileder og oppdragsgiver. De ga oss masse gode råd og tips. Siden vi hadde forholdsvis liten greie på hjemmehjelpstjenesten og hvordan den fungerer, hadde vi kontakt med flere personer for å lære mer om denne sektoren.

Når vi kom i gang med koding ble arbeidet i større grad fordelt mellom oss. Vi forstod raskt at skulle vi ha noen mulighet til å gjennomføre et slikt prosjekt kunne vi ikke sitte alle tre og samarbeide om alt. Selv om vi har jobbet hver for oss, har det vært hyppige ”møter” der vi har oppsummert arbeidet. Det ble som regel slik at en eller to satt med

koding mens resten av gruppen drev med rapporten. I denne fasen ble det lenger mellom møtene med veileder og oppdragsgiver. Avslutningsfasen har vært preget av hyppigere møter med veileder. Vi samarbeider mer igjen for å få en "rød tråd" gjennom hele rapporten.

1.5 Rapportens organisering

Rapporten er delt opp i 10 hovedkapitler.

Kapittel 1 - Innledning:

Inneholder organisering av rapporten. Oppgaveformulering og målgruppe. Om prosjektgruppen, arbeidsmåter, terminologibruk og forkortelser.

Kapittel 2 - Kravspesifikasjon:

Her er det en detaljert beskrivelse av hvordan systemet skal fungere. Kravspesifikasjonen er oppdelt i tre deler. Hvor første delen er ment for ikke datakyndige mens del to og tre blir mer detaljert og flere faguttrykk blir brukt.

Kapittel 3 – Design:

Her blir det nøye beskrevet hvordan vi har tenkt å lage løsningen. Det blir brukt en god del skjermbilder for å illustrere hva vi mener. Pseudokode hører også hjemme her.

Kapittel 4 – Implementering:

Her gjør vi rede for hvilke valg som er gjort med hensyn på verktøybruk og utviklingsmiljø. Standarder og prinsipper vi har valgt, samt oversikt over klasser og ansvarsforhold mellom dem blir også beskrevet her. Det blir lagt vekt på å få med de erfaringer vi har gjort i implementeringsfasen.

Kapittel 5 – Kvalitetssikring og testing:

Her beskrives hva som er gjort for å kvalitetssikre arbeidet vårt. Det beskrives også hva som er gjort for at resultatet skal inneholde så lite feil som mulig og hvilke måter vi har valgt å gjennomføre testingen på, samt testresultat.

Kapittel 6 – Trusselanalyse:

Her vil vi prøve å beskrive mulige trusler mot smartkort og applikasjonen vi lager.

Kapittel 7 – Diskusjon:

Her skal vi gjøre rede for hvordan det ferdige produktet har blitt sammenlignet med kravspesifikasjonen. Avvik vil bli drøftet og forslag til forbedringer vil også være med. Dette kapittelet vil også inneholde våre erfaringer og hvilke meninger vi sitter inne med etter endt prosjekt. Hver gruppedeltager vil også skrive en kortfattet egenvurdering av prosjektet som plasseres i dette kapittelet.

Kapittel 8 – Konklusjon:
Slutninger som kan trekkes fra resultatene blir presentert i dette kapittelet.

Kapittel 9 – Litteraturliste:
Oversikt over litteratur/web-sider som er benyttet.

Kapittel 10 – Vedlegg:
Her legges det med sentrale dokumenter vi mener er av betydning for prosjektet. For eksempel forprosjektrapporten og statusrapporter.

CD:
Her ligger alt vi har produsert av dokumenter og kode. Alle biblioteker og klasser som trengs for å kjøre applikasjonen vil også ligge her.

1.6 Definisjoner

.class	Kjørbart java program, eventuelt deler av et program
.jar	Pakket fil som består av mange .class filer
DBO	Databaseoppretter applikasjonen.
EID	Elektronisk ID. En unik ID som ligger på hvert kort og brukes for å identifisere eieren.
GemDB	Databaseapplikasjonen som ligger på smartkortet.[4],[5],[6]
HJA	Helsejournalapplikasjon.
IT	Informasjonsteknologi.
Java VM	Java virtual machine. Brukes for å kunne kjøre java programmer.
Nøkkel	I denne oppgaven brukes nøkkel om den funksjonen som låser opp smartkortet slik at det er mulig å få aksess til dataene som allerede ligger der samt å kunne legge inn nye.
PCMCIA-kort	Utvidelseskort til bærbar PC/PDA
PDA	Håndholdt PC.
Pocket PC	Operativsystemet på PDA
Smartkort	Formet som et kredittkort. Brukes til å lagre data om pasienten.
Smartkortleser	Hardware som gjør det mulig å kommunisere med smartkortet.

2 KRAVSPESIFIKASJON

2.1 Del 1 - Introduksjon

2.1.1 Bakgrunn

Hjemmesykepleiere sin jobb er å besøke pasientene hjemme. Før de kan dra på besøk, må sykepleieren sette seg inn i eventuelle endringer i helsejournalen. Endringer kan være ny medisin, nye sår på kroppen, det kan hende at en pasient ikke klarer å stå opp av sengen lenger, eller endring i daglige rutiner. Selvfølgelig må de pasientene som ikke klarer å stå opp av sengen besøkes først.

Rutinene kan variere fra kommune til kommune. Vi har valgt å ta for oss et distrikt i Gjøvik. Her finnes en hovedjournal som er på kontoret. Denne inneholder data om alle pasientene. Det er også en hjemmejournal hos hver pasient. Hjemmejournalen inneholder de siste behandlingene som har vært utført. Samt medisindosering og pasienttilstand.

Når sykepleieren er ferdig med pasientbehandlingen, må dette føres inn i hjemmejournalen. Neste sykepleier som kommer til pasienten kan se hva som er blitt gjort. Når besøkrunden er ferdig, må alle data om hver pasient skrives inn i hovedjournalen. Dette medfører at all informasjon blir skrevet ned to ganger for hver pasient. Slik det er i dag, kan man spare mye tid ved å innføre et nytt system. Det er her smartkort/omsorgskort kan forbedre rutinene. Tanken er at man slipper å ha en hjemmejournal i papirform. All informasjon om en pasient skal da være lagret på et omsorgskort. Dette kortet ligger hjemme hos hver enkelt pasient. For å få tilgang til informasjonen på kortet må man være sykepleier/ hjelpepleier eller lege. Sikkerheten blir mye bedre, og mer sensitiv data kan lagres på kortet. Hjemmejournalen ligger i dag tilgjengelig for alle sykepleiere, men også for alle andre som ønsker å lese denne.
Vi skal lage en applikasjon som erstatter hjemmejournalen med et omsorgskort, men skal ikke forandre på hovedjournalen.

2.1.2 Kort om krav til systemet

Omsorgskort skal ved hjelp av smartkortteknologi kunne lagre helseopplysninger om en pasient inn i en database som ligger på kortet. Systemet består av to applikasjoner - helsejournalapplikasjon(HJA) og databaseoppretterapplikasjon(DBO). HJA skal kunne kommunisere med smartkortet for å legge data inn på kortet. En passordkontroll skal hindre at uautoriserte personer kan få tilgang til databasen på kortet. Applikasjonen skal kunne kjøres både på en vanlig PC og en PDA. Programmet må være funksjonelt, og ha et intuitivt brukergrensesnitt. Systemet må være så brukervennlig, at ikke datakyndige personer kan lære seg systemet etter en kort innføring. Det må også lages en DBO som oppretter databasen på kortet, lagrer personinformasjon og pårørende. Dette blir en

applikasjon som inneholder det mest nødvendige for å opprette databasen. Og må nok videreutvikles etter behov.

2.1.3 Kort om systemets omgivelser.

Systemet vi skal utvikle er i første omgang tiltenkt som en demonstrasjonsutgave for å vise hva smartkort teknologien kan tilføre omsorgssektoren i dag. Systemet skal kjøre på en PDA som er tilkoblet en kortleser. For få dette systemet til å fungere er vi avhengig av følgende hardware:

- PDA
- kortleser
- smartkort

2.1.4 Dokumentets struktur

Vi har valgt å dele denne kravspesifikasjonen inn i tre hoveddeler. Dette er gjort med tanke på at vår applikasjon og dokumentasjon er tiltenkt en gruppe der flertallet ikke har gode IT-kunnskaper. Her vil det bli gitt en kort oversikt over hva de enkelte delene omhandler og deres relevans i forhold til de ulike lesergruppene.

Del 1

I denne delen av dokumentet skal det fokuseres på helheten i systemet. En kort og lettfattet innføring som skal være tilrettelagt for alle som vil komme i kontakt med system i sin fremtidige hverdag.

Del 2

Dette avsnittet er rettet mer mot de fremtidige brukerne av systemet, det skal her forklares mer inngående om hvordan systemet skal fungere. Dette vil bygge videre på det som står i del 1 og utbrodere den helhetlige oversikten til å gå litt mer spesifikt inn i hvordan det ferdige systemet vil arte seg.

Del 3

I del tre ligger en langt mer detaljert beskrivelse av hvordan systemet faktisk virker. Denne delen av dokumentet er tiltenk som et grunnlag for designfasen i prosjektet og det er her valgt en objektorientert metode for å beskrive hvordan systemet er bygget opp.

2.2 Del 2 - brukerbeskrivelse

2.2.1 Omgivelser

I dag skjer all kommunikasjon via papirskjemaer og muntlige beskjeder. Det finnes enkelte datasystemer, men det er bare administrasjonen som benytter disse. Systemene brukes blant annet når pasienten skal til sykehus eller flyttes. Den fysiske plasseringen

av vårt system vil bli ute hos hver enkelt pleietrengende, og hos hjemmesykepleieren. Hver pasient skal ha sitt personlige smartkort, og hver pleier sin PDA. Pleieren bruker PDA'en til å legge inn data på kortet.

Smartkortleseren vil bli koblet til PDA'en via et PCMCIA kort. Kortleseren trenger ingen ekstern strømkilde, men henter strøm fra PDA'ens batteri. Kommunikasjonen mellom applikasjonen og smartkortet vil foregå via et ODBC-grensesnitt.

DBO skal kjøres kun på PC. Det vil være mer datakyndige mennesker som bruker denne. Vi legger derfor ikke så mye vekt på hjelpefunksjoner. Dette vil foreløpig kun bli en applikasjon hvor de nødvendige tabeller opprettes og personinformasjon lagres på kortet.

2.2.2 Brukere

De fleste hjemmesykepleierne har minimal datakunnskap. Applikasjonen må tilrettelegges for slike brukere. Det skal legges vekt på at det grafiske brukergrensesnittet skal være selvforklarende. Slik at ikke tid og ressurser tas bort fra brukeren.

2.2.3 Funksjon

Input fra brukeren

Når en pleier kommer hjem til en pasient skal pleieren sette kortet i kortleseren. Vår applikasjon startes og vil be pleieren om å identifisere seg med brukernavn og passord. Hvis det ikke er noe kort i kortleseren, kommer det en feilmelding. Har inntastingen gått greit, er kortet klar for innlesing av data. Brukeren vil legge til eller endre de data som er nødvendig. Applikasjonen vil så lagre disse på kortet.

Output fra kortet

Kortet er nå åpent og de data som brukeren har rettigheter til å se vil kunne bli vist. De mest sentrale data vil være:

- Personinformasjon
- Pårørende
- Behandling
- Diagnose
- Medisinering
- Dagsrapporter

Input til kortet

I denne applikasjonen skal det kun tas hensyn til pleieren og funksjonalitet rundt hjemmebesøkene. Når en pleier er ferdig med besøket, skal de nye dataene om pasientens tilstand skrives inn på PDA'en og lagres på kortet.

I en utvidet utgave av applikasjonen skal også pasientens lege kunne legge inn data på kortet. Det er viktig at bare legen skal kunne endre pasientens medisinering.

Output fra systemet

Applikasjonen skal gi korte og lettfattede meldinger når noe er galt.

- Applikasjonen klarer ikke å opprette kontakt med kortet.
- Feil passord eller brukernavn.
- Smartkortet sitter igjen i kortleseren når sykepleieren forsøker å skru av PDA'en.
- Det tastes inn ugyldige verdier.

2.2.4 Operasjon

Systemet forventes å fungere så lenge applikasjonen er kjørende. Det forutsettes også at det ikke er fysiske skader på smartkortet, smartkortleser eller på annen hardware. Når pasienter skal få omsorgskort, vil det være administrasjonen ved hjemmesykepleien som utgir dette. Det er meget viktig og fange alle mulige feil som kan oppstå ved registrering. For eksempel at fødselsnummer mangler et siffer eller at ikke alle navnfelt er fylt ut. Det skal da ikke lagres eller endres noe på kortet. Hvis smartkortet blir fjernet uten at det ble avsluttet på riktig måte, skal det komme frem en feilmelding om dette.

2.2.5 Vedlikehold

Systemet krever ikke mye vedlikehold. Applikasjonen må selvfølgelig oppdateres hvis det skal bli aktuelt å legge inn nye ting på smartkortet. På grunn av at PDA-er bruker mange forskjellige skjermoppløsninger er applikasjonen vårt tilpasset en bestemt type PDA – Compaq IPAQ 3600 serien. Bytting av PDA vil måtte medføre endringer på brukergrensesnittet. Feilretting på applikasjonen er også en selvfølge. I tillegg så kreves det rensing av kortlesere og nødvendig vedlikehold på PDA'ene.

2.2.6 Aspekter omkring livssyklus

Videreutvikling

Dette systemet blir langt fra ferdig. Viktige ting som gjenstår før systemet eventuelt kan brukes er muligheten legen har til å gå inn på kortet og legge inn data. For eksempel forandring i medisinering. Siden det kun er leger som har lov å forandre medisinene vil det også være nødvendig å legge inn ulike aksessrestriksjoner. Vi har i vårt system kun lagt vekt på hjemmesykepleierne og de data som trengs i den forbindelse.

Det må være en funksjon som sjekker hvor mye plass det er igjen på kortet. Er det liten plass igjen, slettes de eldste dagsrapportene.

Når det gjelder aksessrestriksjoner er en mulig løsning på det problemet at hver enkelt helsearbeider får sitt eget kort. Deres EID bestemmer om de får tilgang til et kort eller ikke. PIN-kode er også en mulig løsning, men det gjør det vanskeligere å skille mellom de ulike instansene i helsevesenet.

Alle data som lagres på kortet, må også lagres lokalt på PDA'en slik at de kan lagres i det sentrale systemet. En naturlig utvidelse vil være at når PDA'en blir koblet opp mot det sentrale systemet så blir dataene lagt inn på rette plass. Dette vil sannsynligvis kreve samarbeid med de som utvikler slike systemer.

DBO bør videreutvikles slik at den også oppretter databasen på kortet. Den bør også være koblet opp mot det sentrale systemet noe som vil gjøre det mulig å legge inn eksisterende helseopplysninger på kortet. Et kort kan bli ødelagt, sperret eller borte på annen måte.

4 til 5 år fram i tid er det meningen av hver nordmann skal få sitt egen helsekort. Helsekortet er et smartkort hvor hele helsejournalen er lagret. På denne måten får alle som trenger det adgang til journalen. Om dette kan sees på som en videreutvikling av vårt system er vel tvilsomt, men vårt omsorgskort vil kunne bli en del av hele helsekortet.

Dokumentasjon

Denne rapporten som inkluderer kravspesifikasjon, designdokument, kildekode, forutsetninger og begrensninger vil være tilgjengelig for offentligheten. Vi vil så langt det er hensiktsmessig bruke Javadoc for å lage dokumentasjon til koden vår, men også komme med egen dokumentasjon ved behov. HJA vil også inneholde en hjelpefunksjon som er beregnet på brukerne av systemet.

2.2.7 Ytelse

Det som tar tid på systemet vårt er inntasting av data. Dette er umulig for oss å gjøre noe med ettersom det kommer an på brukerne. Vi må legge til rette for at kommunikasjonen med smartkortet blir så rask som mulig. Det er vesentlig at systemet blir så raskt og enkelt å bruke at papirbaserte løsninger ikke vil være raskere.

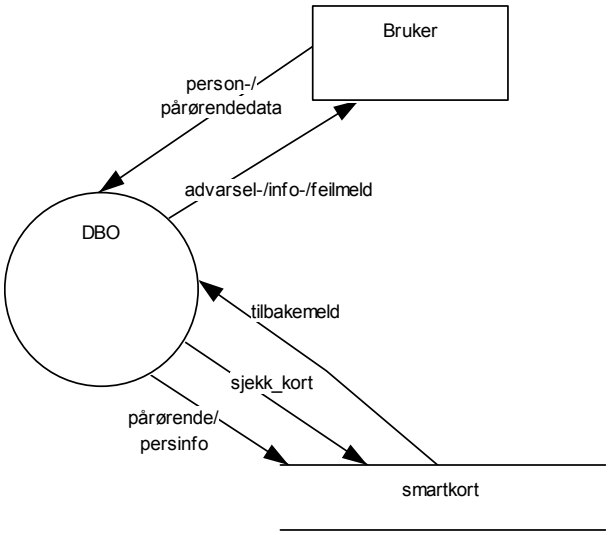
2.2.8 Begrensninger

Det viktigste begrensningen i systemet vårt er smartkortet. Siden vi ikke har mer enn knappe 5 måneder på oss og har andre fag i tillegg så må vi begrense oppgaven kraftig.

- Databasen får en maksimal størrelse på 10KB. Etter hvert så kommer smartkort med større lagringskapasitet [6].
- Eneste variabeltypen i databasen er VARCHAR. Et slikt felt kan være fra 1 til 34 tegn langt.
- En tabell kan inneholde maks 15 felter.
- Hver brukergruppe har kun et brukernavn og passord som alle deler.
- Ingen kan slette dagsrapporter som er lagt inn.
- Vårt system kommuniserer ikke med hjemmesykepleiens eksisterende datasystem.
- Vi antar at det er nok plass på kortet.
- Databasen er opprettet på kortet allerede. Vi oppretter kun tabellene og legger data inn i dem.

2.3 Del 3 - detaljert kravspesifikasjon -DBO

2.3.1 Funksjonell struktur



Figur 2-1 Dataflytdiagram

2.3.2 Dataspesifikasjon og dataordliste

Data input

Input fra brukeren skjer i et grafisk brukergrensesnitt. Disse sendes fra applikasjonen til kortet når brukeren trykker på lagre.

Data output

Output til brukeren vil stort sett være feilmeldinger. Disse dukker opp som meldingsbokser. Det vil ikke gis noen meldinger om at data er lagret på kortet.

2.3.3 Databasen på kortet

Begrensninger

Navn på tabeller og felter kan ikke bestå av mer enn 8 tegn i GemDB. De kan ikke inneholde æ, ø, å eller spesialtegn. Eneste datatypen er varchar som gir maksimum lengde på 34 tegn. Dette begrenser muligheten for å lagre data, spesielt tall, på en effektiv måte.
GemDB er ennå i utviklingsfasen og det er ikke laget databasedrivere for alle vanlige operativsystem.

Tabellene

Som tidligere nevnt har vi valgt å gjøre visse begrensninger for at ikke oppgaven vår skal bli for stor for tiden vi har til rådighet. Dette er også ment som et første utkast og ikke noe ferdig produkt som skal brukes. Vi mener det er viktigste er å vise mulighetene og få fram helheten i systemet. Vi vil her vise hva vi har lagt inn i tabellene og forklare litt hvorfor.

Persoppl	Her lagres personopplysninger om pasienten.
Fnr	Fødselsnummer.
Fnavn	Fornavn.
Enavn	Etternavn.

Tabell 2-1 Personopplysninger

Her kunne vi nok med fordel hatt med telefonnummeret til pasienten, ellers er det ikke så mye som skal lagres her.

Tilknytn	Her lagres opplysninger om pasientens pårørende
Fnavn	Fornavn.
Enavn	Etternavn.
Tilknytn	Slektsforhold.
Tlfnr	Telefonnummer.

Tabell 2-2 Tilknytning

En viktig del av informasjonen på kortet er hvem som er pårørende. Skulle det skje noe med pasienten er det viktig at hjemmesykepleieren vet hvem de kan ta kontakt med. Vi har for enkelhets skyld valgt å bare ta med et telefonnummer til hver pårørende.

Behandl	Her lagres data om hvilke behandlinger pasienten skal ha.
Id	Behandlingsidentifikasjon.
Start	Når behandlingen skal starte.
Slutt	Når behandlingen skal avsluttes.
Oppdrart	Hva som skal gjøres.
Hvem	Hvem utfører behandlingen.
Frekvens	Hvor ofte skal behandlingen utføres.

Tabell 2-3 Behandling

For at pleierne skal vite hva de skal gjøre når de kommer til en pasient er det vesentlig at alle data om en behandling er tilstede.

Medisin	Her lagres informasjon om hvilke medisiner pasienten skal ha.
Navn	Medisinnavn.
Styrke	Styrken på medisinen.
Dosering	Hvor stor dose skal pasienten ha.
Ordinert	Navnet på legen som har foreskrevet medisinen.
Fra	Når skal medisinering starte.
Til	Når skal medisinering slutte.

Tabell 2-4 Medisin

Denne tabellen er meget viktig fordi feilmedisinering kan forårsake alvorlige konsekvenser. En eventuell utvidelse her vil være at medisin lagres med sin medisinkode.

Dagsrapp	Her lagres data om pasienten når sykepleierne er på besøk.
Pleier	Navnet til pleieren har vært på besøk.
Datotid	Når er pleieren på besøk.
Koder	Her lagres dagsrapporten ved hjelp av koder.
Koder2	
Koder3	
Diverse	Her kan pleieren skrive inn en egen tekst.
Diverse2	
Diverse3	
Diverse4	
Diverse5	

Tabell 2-5 Dagsrapport

Dette er den tabellen som skal inneholde hjemmesykepleierens daglige rapport. Her skrives det om forandringer som har skjedd i pasientens tilstand. En post i denne tabellen vil også fungere som en ”kvittering” på at pleieren har besøkt pasienten. Pleieren kan også lese hva som har skjedd tidligere.

I feltene koder skal det lagres koder for faste prosedyrer som gjentas ofte. Det er kun kodene som lagres i databasen. Sammenhengen mellom prosedyrer og kode lagres i applikasjonen. For en oversikt over hvilke koder som brukes se avsnitt koder på side 20.

Videreutvikling

Følgende to tabeller er ikke med i systemet, men ved en eventuell videreutvikling bør de nok være med.

Allergi	Hva er pasienten allergisk mot.
Allergitype	Navn på allergien.
Beskrivelse	Liste over hva pasienten reagerer på og hva skjer hvis .

Tabell 2-6 Allergi

Diagnose	Fylles ut av lege ved legebesøk
Diagnose	
Dato	
Lege	Navn på legen.

Tabell 2-7 Diagnose

Koder

Alle kodene skal bestå av to tegn. Vi har kun konsentrert oss om noen få utvalgte for å vise hvordan det fungerer. Det er ikke noe problem og legge inn flere og kanskje mer relevante koder senere.

Kode	Prosedyre
Va	Vask
Vh	Vask av hjem
Du	Dusj
Sa	Stell av sår
Me	Har fått medisin
Mf	Frokost
Ml	Lunch
Mm	Middag
Mk	Kveldsmat

2.3.4 Operasjonelle systemkrav

a) Normal operasjon

Modus og kontroll

Systemet kan kjøres på en windowsbasert PC. For at applikasjonen skal kunne kjøre trengs en smartkortleser og et GemDB smartkort. På smartkortet må det allerede være opprettet en tom database.

- Brukermodus: Her venter DBO på input fra brukeren. For eksempel når brukeren taster inn data i tekstfeltene.
- Kort-kommunikasjon: Applikasjonen kommuniserer med smartkortet. For eksempel når tabellene lagres.

Ytelse

DBO skal brukes av få personer som alle har kjennskap til data. Vi legger derfor ikke vekt på en stor hjelpefunksjon, men heller lage applikasjonen enkel og intuitiv i bruk.

Krav til maskinvare er at PC'en kan kjøre Java VM. Selvfølgelig må det også være koblet til en kortleser. Applikasjonen vil være så lite ressurskrevende at det stilles ingen andre krav til maskinvare.

Sikkerhet

DBO er ikke passordbeskyttet og hvem som helst kan starte applikasjonen. Applikasjonen gir ingen tilgang til noe på kortet, det er kun lov å lagre nye data. Databasen på kortet er tom slik at sikkerhet rundt systemet er ikke så veldig høyt prioritert.

Oppstart og nedtagning

DBO startes og avsluttes av brukeren når han har behov for å utstede nye kort.

Innebygde tester

a) Om kortet er satt i kortleseren og kommunikasjonen fungerer.
b) Om brukeren taster riktig verdier i tekstfelter.

b) Operasjoner i feilsituasjoner

Feilrapportering

Hvis de innebygde testene slår negativt ut gis tilbakemelding til brukeren med beskjed om hva som er feil.

Hva skjer ved feil

a) Brukeren får beskjed om å sette kortet i kortleser og hvis feilen gjentar seg kontakte systemansvarlig.
b) Brukeren får beskjed om at han har tastet en ulovlig verdi. Han må trykke ok.

Sikkerhet

Det lagres ikke noen kopi av data som brukeren skriver inn. Skulle det oppstå feil slik at data blir borte må brukeren legge inn på nytt.

2.3.5 Funksjonelle krav

Input

Brukeren taster inn personnummer, fornavn og etternavn på pasienten. Når alle feltene er fylt inn trykker brukeren på lagre personopplysninger som lagrer data på kortet og sende brukeren til et nytt skjermbilde. Nå skal brukeren taste inn fornavn, etternavn, telefonnummer til pårørende. Den pårørendes tilknytning til pasienten velges i en ”combobox”. Brukeren trykker på lagre pårørende. Det er mulig å legge inn flere pårørende. Brukeren har så mulighet til å lage nytt kort eller avslutte.

Prosessering

Applikasjonen oppretter tabeller og lagrer data på kortet. Når brukeren trykker på ”nytt kort”-knappen så sendes han tilbake til første skjermbildet for å opprette et nytt pasientkort.

Output

Når brukeren registrerer pårørende vil navnet på disse dukke opp i et tekstfelt etter hvert som de blir registrert, all annen output vil være feilmeldinger/veiledning.

Kontroll

Verdiene brukeren taster inn i tekstfeltene blir kontrollert slik at de ikke inneholder ulovlige tegn.

- Fornavn, etternavn: bare bokstaver
- Personnummer: gyldig personnummer
- Telefonnummer: kun 8 tall er gyldig

2.3.6 Begrensninger

a) Software begrensninger

Standarder og språk

Applikasjonen skal utvikles i java. For kommunikasjon med databasen på kortet brukes SQL gjennom et ODBC-grensesnitt.

Pakker/verktøy

For å utvikle applikasjonen skal vi bruke Jbuilder 4 fra Borland. Dette vil lette arbeidet med det grafiske brukergrensesnittet betraktelig. For å lage dokumentasjon av koden skal vi bruke Javadoc som lager html-sider. I systemutviklingsbiten har vi brukt UML-suite fra Telelogic. For å opprette selve databasen på kortet bruker vi GemDB administrative tools.

Operativsystemet

På grunn av databasen som ligger på kortet trenger vi en databasedriver. Denne fungerer kun i Windows 98 og på Windows NT. Det kreves også at operativsystemet kan kjøre java, noe både windows 98 og Windows NT kan.

Smartkortet

Eneste variabeltypen i databasen er VARCHAR. Et slikt felt kan være fra 1 til 34 tegn langt. En tabell kan inneholde maks 15 felter.

b) Hardware begrensninger

Se punkt 2.1.3 - Kort om systemets omgivelser.
Se punkt 2.2.8 - Begrensninger

2.3.7 Aspekter om livssyklus

Dokumentasjon

All kode kommentertes på en slik måte at det er enkelt for andre datakyndige å sette seg inn i hva som skjer. Det vil ikke bli lagt stor vekt på noen hjelpefunksjon i programmet, men utarbeidet en liten forklaring på hva som skjer. Denne vil være tilgjengelig fra hjelp-menyen i applikasjonen.

Modul og integrasjonstesting

Vi vil teste for følgende feil/mangler:

- Smartkort er satt i kortleseren
- Kommunikasjonen med GemDB fungerer.
- Tabellene opprettes og korrekte data lagres i dem.

I tillegg vil vi teste at brukervennligheten er ok. Det vil si at en litt datakyndig person enkelt skal kunne bruke applikasjonen.

Krav til support, service og vedlikehold

ErgoSolutions er de som vil stå for support, service og vedlikehold. ErgoSolutions får muligheten til å utvikle løsningen videre etter eget ønske.

2.3.8 Aspekter omkring installasjon

Installasjon

For at systemet skal fungere kreves en PC som kan kjøre Java VM. Det kreves også at en kortleser er koblet til PC'en.

For å installere software må man kopiere .jar fila over til PC'en. Denne kjøres ved hjelp av Java VM.

Opplæring

Siden programmet er såpass enkelt og intuitivt vil det ikke være behov for noe særlig opplæring. Det er selvfølgelig en fordel at brukerne får en kort opplæring i hvordan smartkortteknologien fungerer.

De som skal videreutvikle programmet vil få tilgang til denne rapporten og all kildekode inkludert kommentarer. I tillegg har vi også web-sidene laget med Javadoc.

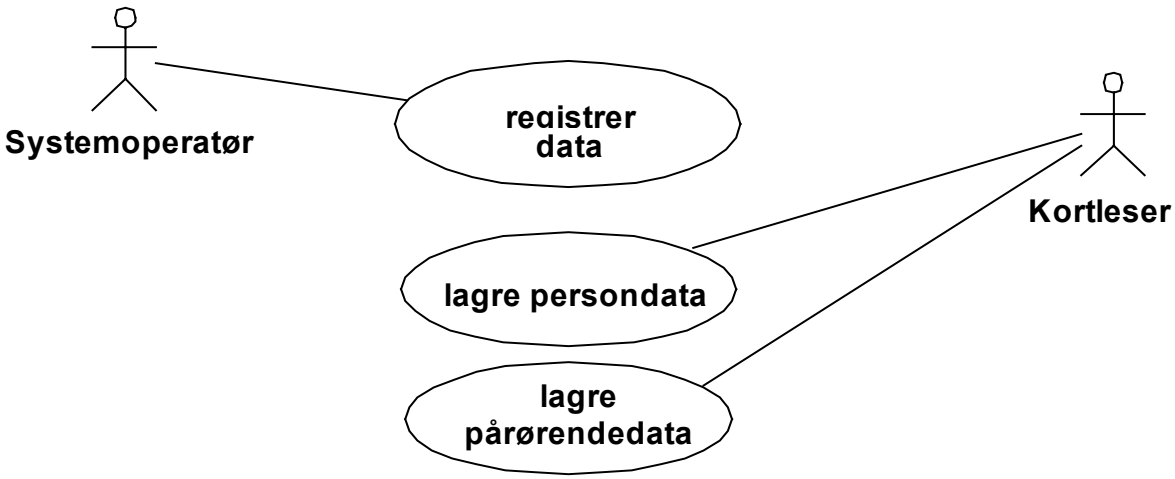
2.3.9 Use-case

Use case:	registrer data
Aktør:	administrasjon
Type:	primært
Beskrivelse:	Brukeren setter kortet i kortleseren. Taster inn nødvendig person-/pårørendeopplysninger i tekstfeltene. Når brukeren registrert ferdig

trykkes det på lagre. Hvis det fylles inn ugyldige data vil det bli gitt feilmelding og brukeren må trykke ok. Det må fylles inn gyldige verdier i alle felter før brukeren kommer videre.

Use case: lagre persondata
Aktør: kortleser
Type: primært
Beskrivelse: Når brukeren trykker lagre personinformasjon blir dette use caset kjørt. Tabellene opprettes på kortet og data fra tekstfeltene blir lagret på kortet. Hvis data ikke blir lagret vil det komme en feilmelding.

Use case: lagre pårørendedata
Aktør: kortleser
Type: primært
Beskrivelse: Dette use caset blir kjørt når brukeren trykker på lagre pårørende. Data fra tekstfeltene blir lagret på kortet. Hvis data ikke blir lagret vil det komme en feilmelding. Når en pårørende er lagret vil denne vises i et eget felt.



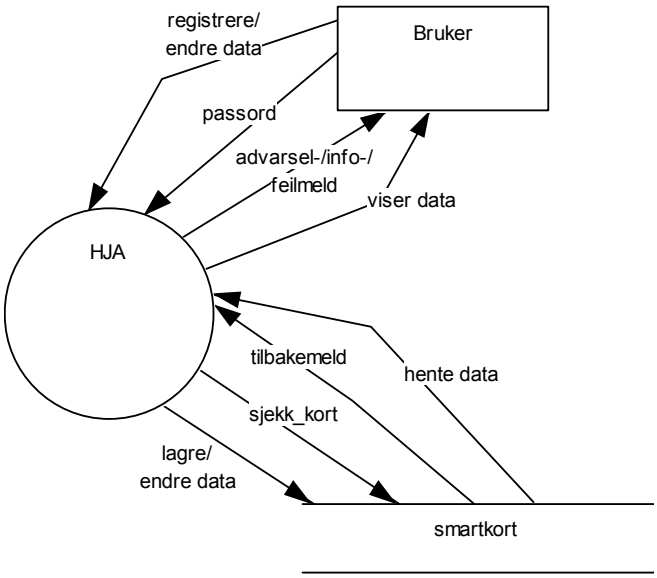
Figur 2-2 Use case diagram

2.4 Del 3 - detaljert kravspesifikasjon – HJA

2.4.1 Innledning

Denne applikasjonen er på mange måter veldig lik DBO-applikasjonen både hard- og softwaremessig. For at vi ikke skal få et veldig oppstykket dokument velger vi å ta med momenter som også er nevnt i DBO-kravspesifikasjonen.

Funksjonell struktur



Figur 2-3 Dataflytdiagram

2.4.2 Dataspesifikasjon og dataordliste

Data input

Input fra brukeren skjer i et grafisk brukergrensesnitt. Input sendes fra applikasjonen til kortet når brukeren trykker på de ulike lagre-knappene.

Data output

Output til brukeren vil være data som ligger lagret på kortet samt enkelte feilmeldinger. De dataene som ligger lagret i kortet vil komme opp i ulike felter og tekstområder i applikasjonen. Det vil bli brukt ”tabsheet” for bedre å utnytte den lille plassen som er tilgjengelig på en PDA. Feilmeldingene vil dukke opp som meldingsbokser. Det vil ikke gis noen meldinger om at data er lagret på kortet.

2.4.3 Operasjonelle systemkrav

a) Normal operasjon

Modus og kontroll

Denne applikasjonen er tiltenkt å skulle kjøre på en PDA og alt fra funksjonalitet til grafikk er derfor tilrettelagt for dette. Det kreves en PDA med kortleser for smartkort, samt et smartkort fra GemDB med en initiert database. Initiering av databasen vil si at DBO kjøres for å lage tabeller på kortet.

Hente data

Når HJA startes vil det være en sjekk som kontrollerer om kortet er i leseren. Hvis det ikke er noe kort der vil det komme en melding om å sette i kortet. Etterpå blir data hentet fra kortet. Kommunikasjonen med smartkortet er ganske treg og for å unngå unødvendig venting vil kun nødvendige data blir lest inn. Det vil være fullt mulig å hente mer data hvis brukeren ønsker dette. Data som blir hentet i første omgang vil være personinformasjon, siste dagsrapporter, navn på medisiner, dosering og behandling. Vil bytte til brukermodus så snart innhenting av data er ferdig.

Brukermodus

Her venter HJA på brukeren skal komme med input. Brukeren kan velge mellom de ulike menyene, lese inn mer data fra kortet, skrive inn data som skal på kortet.

Lagremodus

Hit kommer vi når brukeren har endret eller skrevet inn nye data som skal lagres på kortet. Data blir satt inn i de riktige tabellene og systemet går tilbake til brukermodus. Hvis det skulle oppstå feil under lagring vil det bli generert feilmelding.

Sjekk smartkort

Når applikasjonen kjører vil det hele tiden være en sjekk på at smartkortet sitter i leseren. Hvis kortet fjernes skal det genereres en feilmelding.

Innebygde tester

1. Passordsjekk ved oppstart av applikasjon.
2. Om kortet er satt i kortleseren og kommunikasjonen fungerer.
3. Kommunikasjonen med kortet brytes.
4. Om brukeren taster riktig verdier i tekstfelter.
5. Om data lagres korrekt.

b) Operasjoner i feilsituasjoner

Feilrapportering

Hvis de innebygde testene slår negativt ut gis tilbakemelding til brukeren med beskjed om hva som er feil.

Hva skjer ved feil

Brukeren får beskjed om:

1. brukernavn eller passord er tastet inn feil. Brukeren får mulighet til å prøve på nytt eller avslutte programmet.
2. Sette inn kortet. Mulighetene er prøv på nytt eller avslutt.
3. Smartkortet er tatt ut, sett det inn igjen. Muligheter vil være prøv på nytt eller avslutt.
4. Innhold i tekstfelt er ugyldig. Skriv på nytt.
5. Data ble ikke lagret. Prøv på nytt.

Sikkerhet

Det lagres ikke noen kopi av data som brukeren skriver inn. Skulle det oppstå feil slik at data blir borte må brukeren legge inn på nytt.

2.4.4 Ytelse

Lett å bruke

HJA skal i motsetning til DBO benyttes av langt flere og mindre datakyndige personer. På grunn av dette er det derfor lagt vekt på at programvaren skal være enkel i bruk, og at brukergrensesnittet skal være enkelt og selvforklarende.

Pålitelighet og robusthet

Programmet må kunne håndtere brukerfeil. Når brukerfeil oppstår skal det dukke opp gode, informative feilmeldinger. Det er viktig at systemet ikke oppgir feil verdier da dette kan få fatale konsekvenser. Systemfeil som at applikasjonen henger seg skal heller ikke forekomme.

Sikkerhet

Det vil under oppstart av applikasjonen komme en innloggingsboks hvor brukernavn og passord må oppgis. Resten av applikasjonen starter ikke før passord og brukernavn er godkjent. Det vil i dette programmet være en bruker. I fremtiden er det tenkt at hver person som skal bruke applikasjonen må identifisere seg med sitt personlige ansattkort. I dette førsteutkastet til et system vil det ikke være fokusert veldig på sikkerhet, men et systemet hvor alle ansatte i helsesektoren har et ansattkort basert på smartkort systemet vil i fremtiden kunne gjøre dette til et meget sikkert system.

Oppstart og nedtagning

HJA startes og avsluttes som et hvilket som helst annet program i windows. Det vil ha sitt eget ikon.

2.4.5 Funksjonelle krav

Input

I denne applikasjonen er det mulighet for å legge inn data etter hvert besøk hos en pasient. Det skal registreres hva som har blitt utført under besøket slik at dette er tilgjengelig for neste pleier som kommer. Det skal være mulig å legge inn en skriftlig formulering om besøket på 170 tegn. De vanligste arbeidsoppgavene finnes i avkrysningsbokser slik at skrivearbeidet minimaliseres.

Output

Brukeren kan velge mellom ulike ”tabsheets”. Dette åpner for å kunne hente ut data som ligger lagret i kortet. Det vil bli en ”tabsheet” for hver av punktene 1 til 4 under.

1. Personalia om pasienten
   - pasientens for- og etternavn
   - fødselsnummer

2. Informasjon fra tidligere besøk av pleiere/leger
   - liste over besøkene
   - avkrysningsbokser for de mest vanlige handlinger under et besøk

3. en oversikt over medisinering
   - medikamentnavn
   - styrke
   - dosering
   - tidsrom for inntak

4. hvilke behandling pasienten er under
   - tidsrom
   - type behandling
   - hvem skal behandle
   - frekvens

Kontroll

Verdiene brukeren taster inn i tekstfeltene blir kontrollert slik at de ikke inneholder ulovlige tegn.

- Data om pårørende som for- og etternavn: bare bokstaver.
- Telefonnummer: kun 8 tall er gyldig.
- Smartkort sitter i leseren mens applikasjonen kjører.
- Data blir lagret riktig.
- Brukernavn og passord.

2.4.6 Begrensninger

a) Software begrensninger

Standarder og språk

Applikasjonen skal utvikles i java. For kommunikasjon med databasen på kortet kjøres SQL gjennom et ODBC-grensesnitt.

Pakker/verktøy

For å utvikle applikasjonen skal vi bruke Jbuilder 4 fra Borland. Dette vil lette arbeidet med det grafiske brukergrensesnittet betraktelig. For å lage dokumentasjon av koden skal vi bruke Javadoc som lager html-sider. I systemutviklingsbiten har vi brukt UML-suite fra Telelogic. For å opprette selve databasen på kortet bruker vi GemDB administrative tools.

Operativsystemet

På grunn av databasen som ligger på kortet trenger vi en databasedriver. Denne fungerer kun i Windows 98 og på Windows NT. Det kreves også at operativsystemet kan kjøre java, noe både windows 98 og Windows NT kan.

Smartkortet

Eneste variabeltypen i databasen er VARCHAR. Et slikt felt kan være fra 1 til 34 tegn langt. En tabell kan inneholde maks 15 felter.

b) Hardware begrensninger

Se punkt 2.1.3 - Kort om systemets omgivelser.
Se punkt 2.2.8 - Begrensninger

2.4.7 Aspekter om livssyklus

Dokumentasjon

All kode skal bli kommentert på en slik måte at det er enkelt for andre datakyndige å sette seg inn i hva som skjer. Applikasjonen trenger en god hjelpefunksjon, da brukergruppen ikke innehar mest datakunnskaper.
Den skal være intuitiv for brukeren, slik at det er enkelt å finne frem til aktuelt problem.

Modul og integrasjonstesting

Her beskrives testing av de ulike modiene. Se punkt Modus og kontroll på side 26.

Hente data

Det må testes på om applikasjonen leser inn de korrekte dataene. At det ikke leses inn mer enn det er bruk for slik at hastigheten blir optimalisert.

Brukermodus

Vi må teste om det er mulig å gjøre feil slik at feil vil oppstå og rette på disse. Programmet må være så robust at det tåler alle mulige feilsituasjoner som kan oppstå når brukeren taster inn noe feil.

Lagre data

Her må det testes at riktige data blir lagret på kortet og at de blir lagret på rett sted. Det må testes på at data som forsøkes lagret er på en slikt format at databasefeil ikke oppstår.

Sjekk smartkort

Vi må teste at funksjonen kommer med en feilmelding hvis kortet fjernes eller det settes i et ugyldig kort.

Krav til support, service og vedlikehold

Support, service og vedlikehold vil være ErgoSolutions AS sitt ansvar. Hvis de ønsker å videreutvikle systemet står de fritt til det.

2.4.8 Aspekter omkring installasjon

Installasjon

Systemet skal kjøres på en PDA tilkoblet en kortleser. Det er et krav at denne PDA'en kan kjøre java-applikasjoner.

Applikasjonen er pakket til en .jar fil. Det er nok å kopiere denne over på PDA'en og kjøre den. Det må også opprettes ODBC-kobling mot databasen. For ytterligere informasjon om installasjon se egen installasjonsguide i vedlegg 1.

Opplæring

Dette programmet skal brukes av ei gruppe der flertallet ikke er datakyndige. Det kreves en grundig opplæring, men vi har lagt vekt på at brukergrensesnittet skal være så intuitivt som mulig. Det er også lagt vekt på at det ikke skal være mulig å gjøre store feil. Applikasjonen vil også inneholde en hjelpefunksjon som vil svare på de mest grunnleggende spørsmål.

Hvis dette produktet skal brukes til kommersielt bruk, må det videreutvikles. De som eventuelt skal videreutvikle vil ha tilgang til denne rapporten og all kildekode inkludert kommentarer.

2.4.9 Use-case

Use case: sjekk passord

Aktør: pleier
Type: primært
Beskrivelse: Applikasjonen starter opp og brukeren får opp et vindu til å skrive brukernavn og passord i. Bruker skriver inn brukernavn og passord, systemet sjekker om det er korrekt. Er det ikke korrekt får bruker nytt forsøk.

Use case: legg inn dagsrapport
Aktør: pleier
Type: primært
Beskrivelse: Brukeren får mulighet til å legge inn et sammendrag av hva som har skjedd på dette besøket i tekstfelt. Det er avkrysningsbokser for de mest brukte setningene.

Use case: endre pårørendedata
Aktør: pleier
Type: sekundært
Beskrivelse: Bruker får opp et vindu med personopplysninger om den pårørende. Brukeren kan så endre, slette eller legge til ny pårørende. Telefonnummer og etternavn er eneste som kan forandres.

Use case: legg inn medisinering
Aktør: pleier
Type: primært
Beskrivelse: Bruker får opp en liste med all medisin som pasienten bruker. Når det skal legges inn ny fyller pleieren inn de nødvendige opplysningene og medisinen lagres.

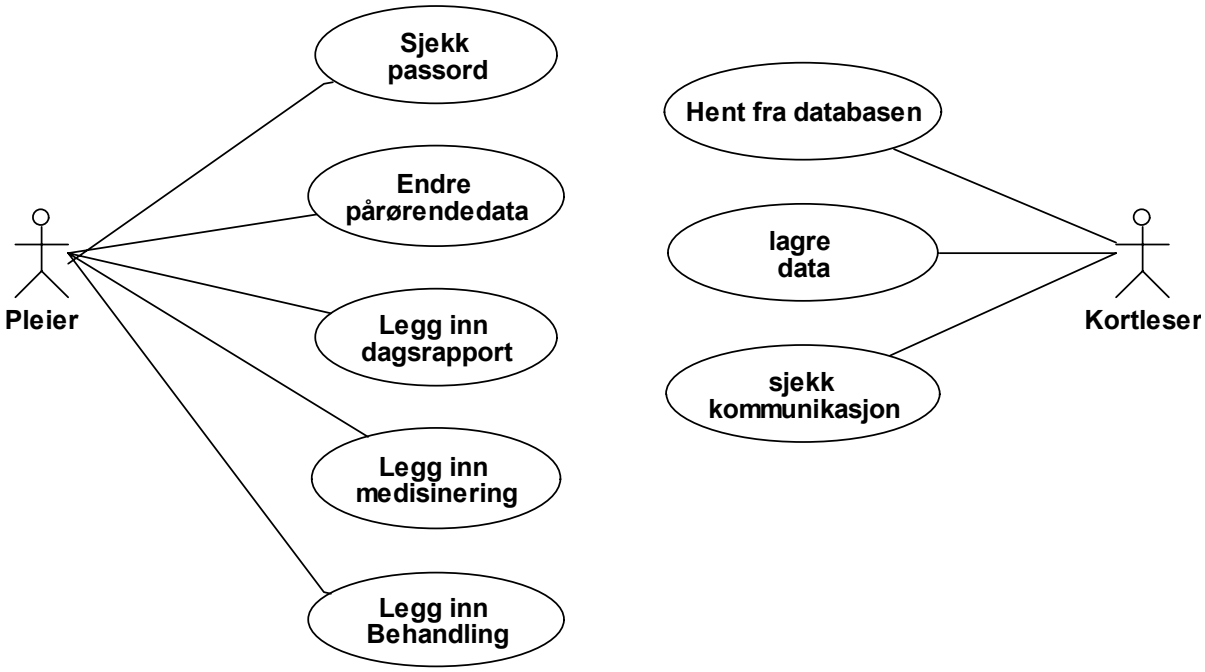
Use case: legg inn behandling
Aktør: pleier
Type: primært
Beskrivelse: Hvis pasienten trenger endringer i behandlingsopplegget har pleieren mulighet til å gjøre dette her. For eksempel at en pasient trenger hjelp til å stå opp om morgenen.

Use case: Hent fra databasen
Aktør: kortleser
Type: primært
Beskrivelse: Data hentes fra databasen ved hjelp av SQL kommandoer. Kortleseren returnerer data til applikasjonen som plasserer den i de rette feltene.

Use case: lagre data
Aktør: kortleser
Type: primært

Beskrivelse: Brukeren har gjort endringer i dataene og ønsker å lagre dette. Data sendes fra applikasjon til kortleser som lagrer de i databasen på kortet. Det foretas også en test for å sjekke at data lagres korrekt.

Use case: sjekk kommunikasjon
Aktør: kortleser
Type: sekundært
Beskrivelse: Kjøres med jevne mellomrom for å kontrollere at smartkortet ikke er fjernet fra leseren. Blir smartkortet fjernes kommer det en feilmelding. Brukeren får da beskjed om å sjekke at kortet sitter i kortleseren.



Figur 2-4 Use-case diagram

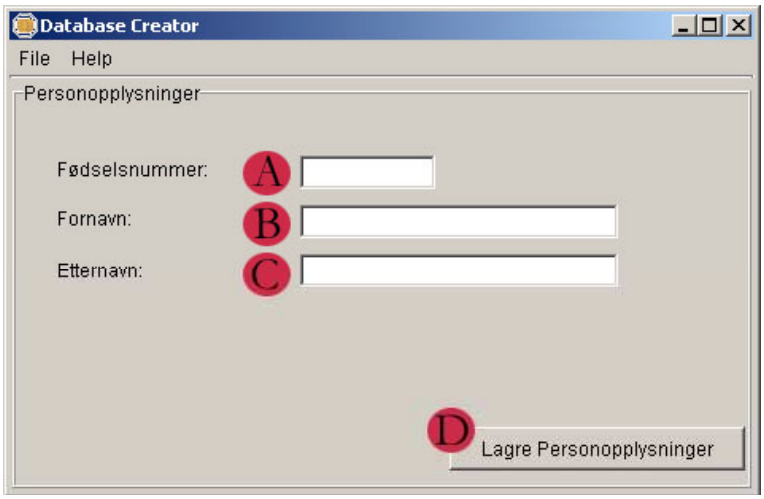
3 DESIGN

3.1 Innledning

Designdokumentet for DBO og HJA er bygd opp slik at de kan leses helt uavhengig av hverandre. Dette medfører at det er enkelte punkt som er ganske like. De er begge bygd opp på en slik måte at skjermbildene med beskrivelse av disse kommer først. Etterpå vil vi gå nærmere inn på de forskjellige klasser og funksjoner som skal brukes. Pseudokode som viser hvordan de ulike funksjonene virker vil også være en viktig del av dette dokumentet.

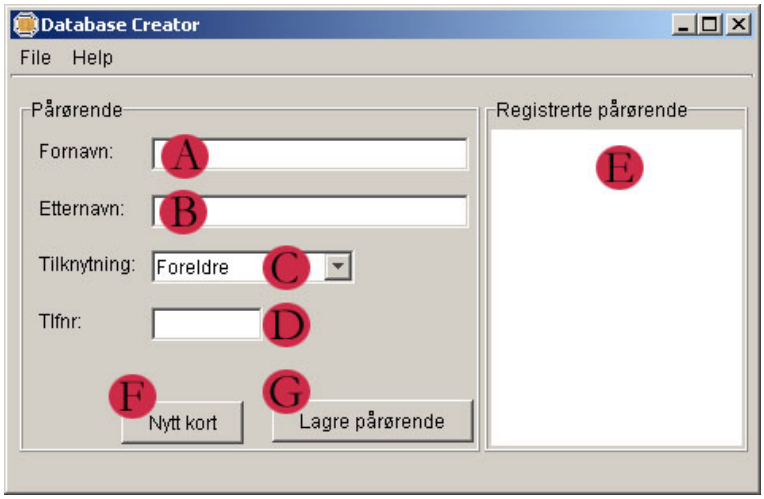
3.2 Design - DBO

3.2.1 Skjermbilder



Figur 3-1 Personopplysninger

Dette er hovedsiden som dukker opp når programmet startes. Brukeren fyller ut feltene **A**, **B**, **C** og trykker på **D** for å lagre. Når det trykkes på **D** opprettes også alle tabeller på kortet.



Figur 3-2 Pårørendeopplysninger

Hvis tabellene blir opprettet og personinformasjon blir lagret blir brukeren sendt videre hit. Bruker legger inn data i feltene **A**, **B**, **D** og velger en tilknytning fra **C**. Når brukeren trykker på **G** for å lagre, blir data lagret i pårørende-tabellen på kortet. De registrerte pårørende vil vises i **E**. Når brukeren er ferdig med å legge inn alle pårørende avslutter han enten applikasjonen eller trykker **F** for å lage et nytt kort.

Feilmeldinger/Tilbakemeldinger

Her er et utvalg av advarsler/feilmeldinger som blir vist når bruker skriver inn feil data eller andre feil oppstår.

Figur 3-3 Feilmelding

Figur 3-4 Feilmelding

Figur 3-5 Feilmelding

Figur 3-6 Feilmelding

Figur 3-7 Feilmelding

Figur 3-8 Feilmelding

3.2.2 Pseudokode

RegPersoninfo(){
while(!sjekkKortILeser()){
info(”sett kort i leser”);
if(cancel)
break;
}
	Systemet startes.
	Bruker skriver data inn i tekstfeltene og trykker lagre.
	
	Hvis bruker trykker cancel kommer han tilbake til registreringsvinduet

if(tekstfelt.innhold=ok){
  kortbase1=new database();
  opprettTabell(SQL);
  insertData(Personopplysn);
  regPårørende();
 	
 }else{
  advarsel(”tekstfelt.innhold er feil”);
  break;
 }
 if(lagretPåKort!=ok){
  feil(”Feil på kort, ingen data lagret”);
 }
}
	Bruker har trykket lagre, kortet er i leseren.
	Oppretter databaseoppkobling
	Tabellene opprettes
	Lagrer data på kortet.
	Bytter til neste skjermbilde.
	
	Får opp advarsel og mulighet til å endre innholdet i tekstfeltene.

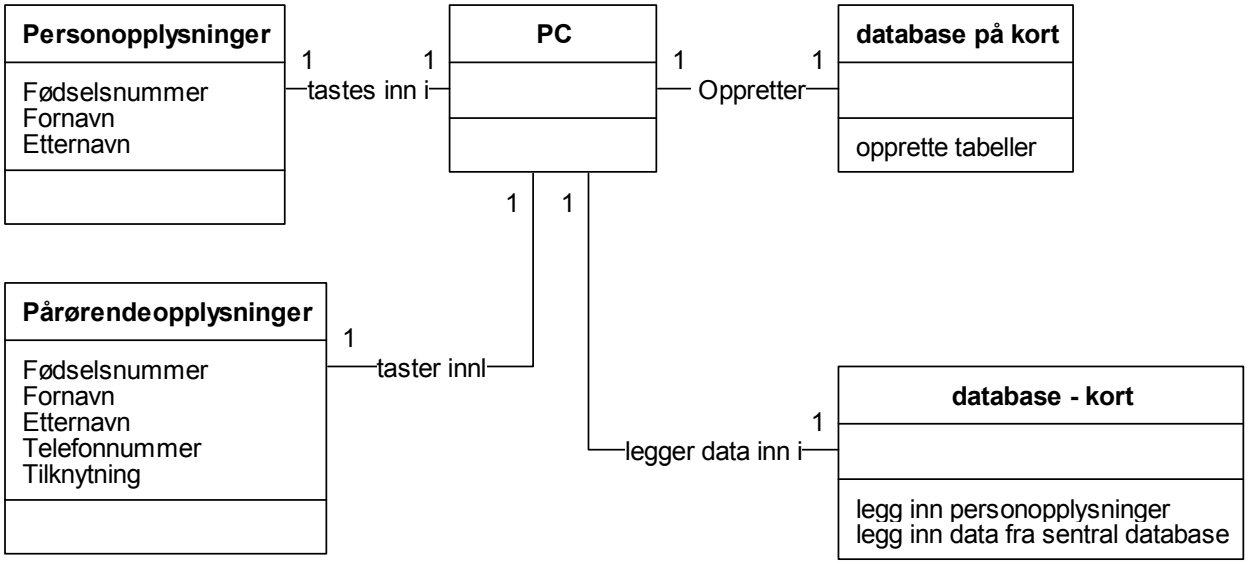
regpårørende(){
 if(tekstfelt.innhold=ok){
  insertData(pårørende);
  if(lagretPåKort!=ok)
  feil(”Feil på kort, ingen data lagret”);
 }
 else{
  advarsel(”tekstfelt.innhold er feil”);
  break;
 }
 visPårørendeITekstarea();
}
	Bruker skriver i tekstfeltene og trykker lagre.
	
	Navnet til pårørende som er registrert vises i tesktområdet.

if(RegNyPårørende) {
  regpårørende();
}elseif(NyttKort) {
  regNy();
}
else{
 exit(0);
}
	Registrerer flere pårørende.
	
	Opprett nytt kort
	
	Avslutt applikasjon

regNy(){
 tekstfelt.innhold=””;
 regPersoninfo();
}
	Nullstiller alle tekstfelt

3.2.3 Systemutviklingsmodeller

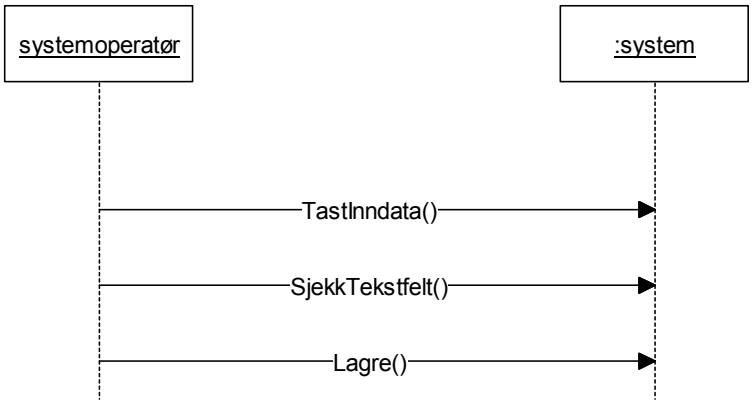
Konseptuelt klassediagram [8]



Figur 3-9 Konseptuelt klassediagram

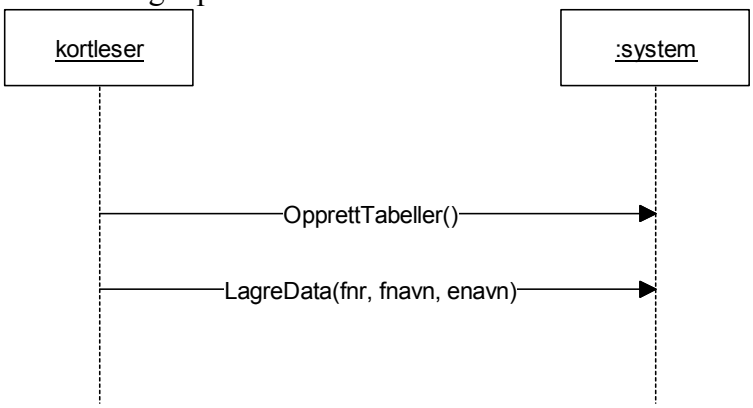
Systemsekvensdiagram [8]

Use-case: tast inn data



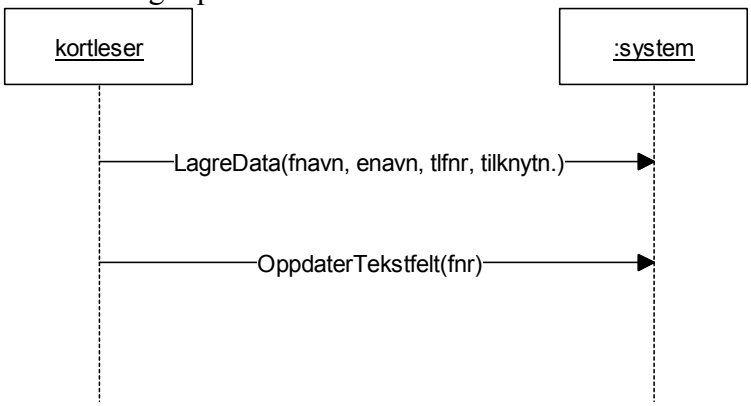
Figur 3-10 Systemsekvensdiagram

Use-case:lagre persondata



Figur 3-11 Systemsekvensdiagram

Use-case:lagre pårørende



Figur 3-12 Systemsekvensdiagram

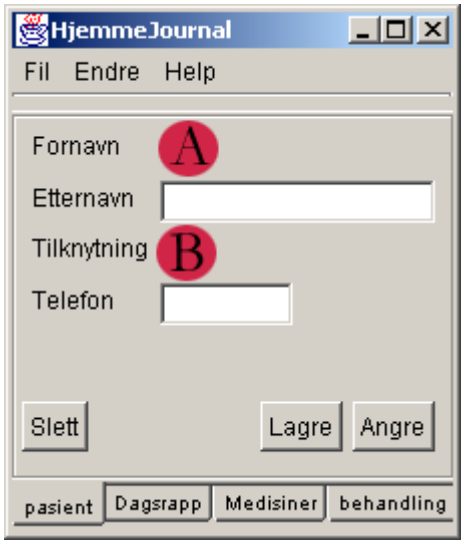
3.3 Design – HJA

3.3.1 Skjermbilder

Pasient

Figur 3-13 Pasientopplysninger

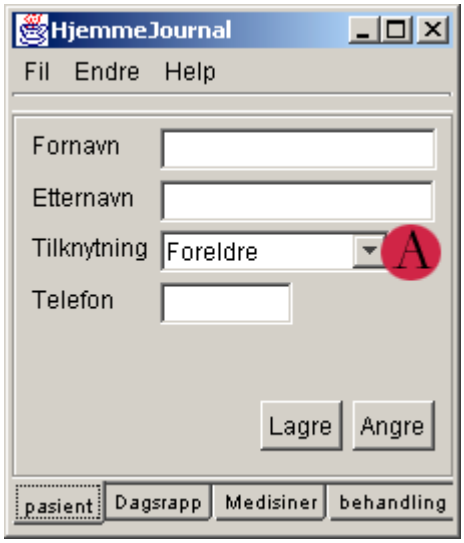
Dette er hovedsiden som dukker opp når programmet startes. Ved bokstav **A** skal det dukke opp navn på pasienten. Ved **B** står personnummeret. Tekstfeltet(**C**) inneholder navn på de pårørende som er lagt inn. Når en pårørende velges fra **C** vises det hvilken tilknytning den valgte pårørende har til pasienten og telefonnummeret hans.



Figur 3-14 Endre pårørende

Det er mulig å endre opplysninger om en pårørende. Brukeren skal da komme inn i dette skjermbildet. Ved **A** vil fornavnet dukke opp og ved **B** slektsforholdet den

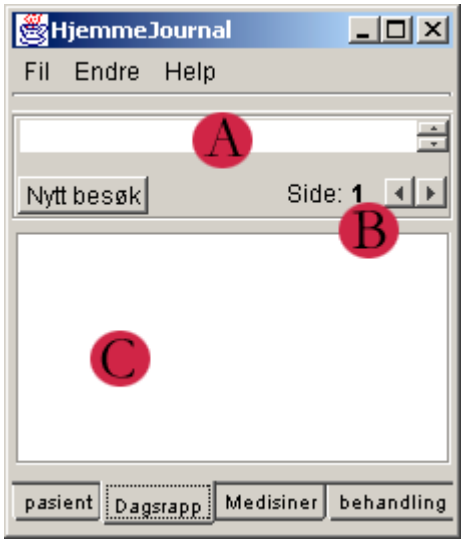
pårørende har til pasienten. Det er ikke mulig å endre disse to opplysningene. Den pårørende kan slettes med slett knappen, endringer lagres med lagre. Når brukeren er ferdig, enten ved at han trykker slett, lagre eller angre sendes han tilbake til pasientopplysninger(Figur 3-13).



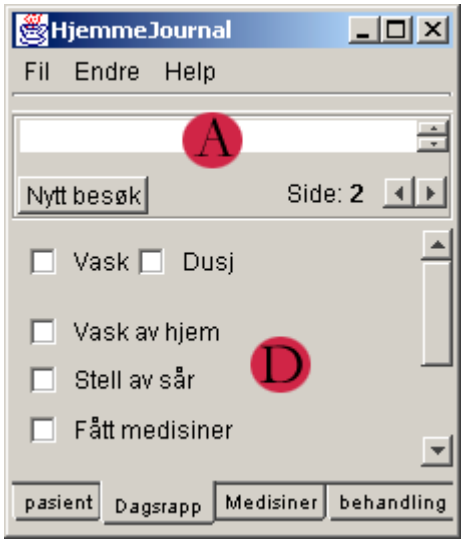
Figur 3-15 Ny pårørende

Brukeren har også mulighet for å legge inn nye pårørende. Brukeren fyller ut tekstfeltene. I ”comboboxen”(**A**) velges hvilket slektsforhold pasienten har til den pårørende. Bruker avslutter ved å trykke enten lagre eller angre. Når det blir trykt på en av disse knappene, vises pasientopplysninger igjen.

Dagsrapport



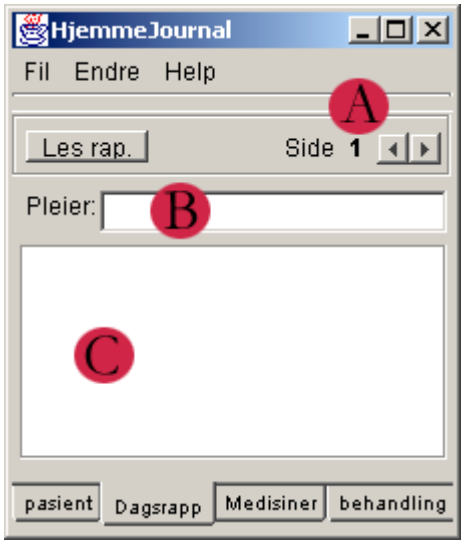
Figur 3-16 Les dagsrapport side 1



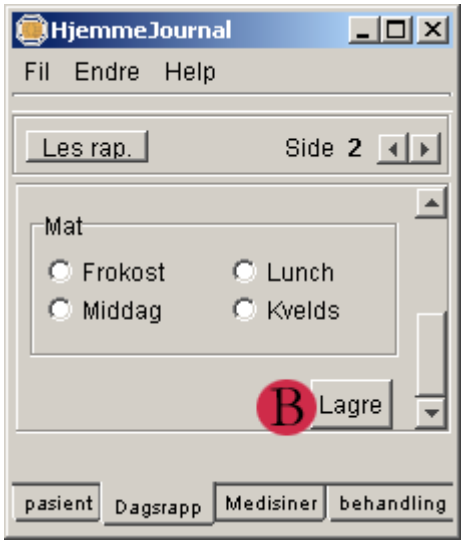
Figur 3-17 Les dagsrapport side 2

Her kan pleieren lese alle dagsrapportene som er lagt inn tidligere. Denne funksjonaliteten ligger i ”spinboxen”(**A**). I tekstfeltet her ligger dato og klokkeslett for når besøket ble lagt inn samt navnet på pleieren. En dagsrapport består av to sider. Det

kan velges mellom de to sidene med pilene(**B**). Tekstfeltet (**C**) inneholder den frie teksten som en pleier kan skrive inn. Avkrysningsboksene (**D**) gjør det mulig å kutte ned på den frie teksten. Det skal lages avkrysningsbokser for oppgaver som ofte utføres. Det vil ikke være mulig å forandre verken på teksten eller avkrysningsboksene – kun lese.



Figur 3-18 Ny dagsrapport side 1

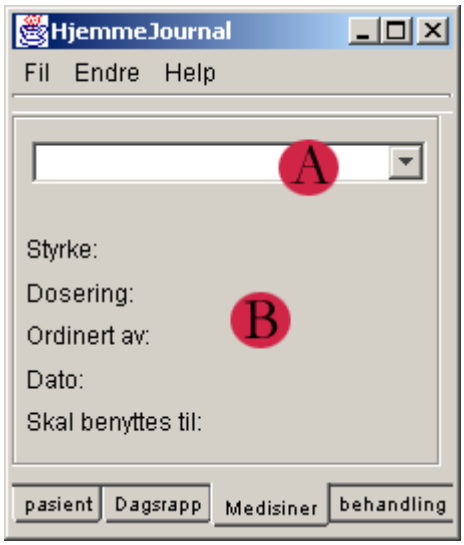


Figur 3-19 Ny dagsrapport side 2

Når brukeren trykker på ”ny dagsrapport”-knappen i Figur 3-16 kommer han automatisk til side 1 her. Brukeren må fylle ut navn på pleier(**B**) og eventuelt tekstfeltet(**C**). I tekstfeltet er det meningen av brukeren skal fylle inn en fritekst om hvordan pasientens tilstand er. Denne teksten kan maks være på 170 tegn. Brukeren kan bytte side med pilene (**A**). Side 2 består kun av avkrysningsbokser. Dagsrapporten lagres når bruker trykker på lagreknappen(**B**) på side 2.

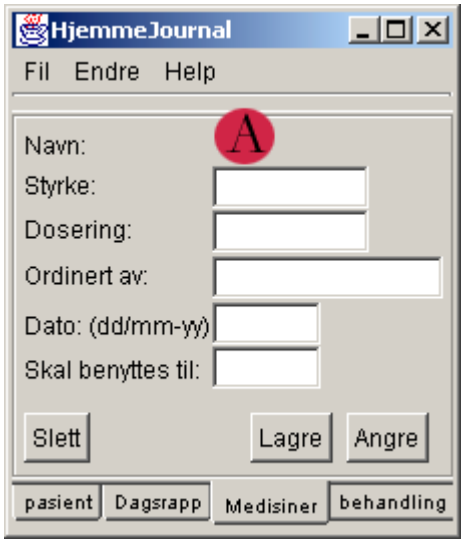
Ny dagsrapport har ingen angreknapp. Knappen ”les dagsrapport” gir brukeren en mulighet til å gå tilbake for å se på gamle dagsrapporter. For å komme seg tilbake for å fylle inn mer data, må brukeren trykke ”ny dagsrapport” igjen. Data som allerede er tastet inn vil fremdeles ligge i komponentene. Plasseringen av lagre-knappen(**B**) er gjort med hensikt, nederst på side 2, slik at brukeren skal tvinges til å se gjennom hele dagrapporten før den kan lagres. En lagret dagsrapport kan ikke endres. Grunnen til at det ikke er noen angre-knapp er at pleieren skal fylle ut en dagsrapport for hver pasient.

Medisin



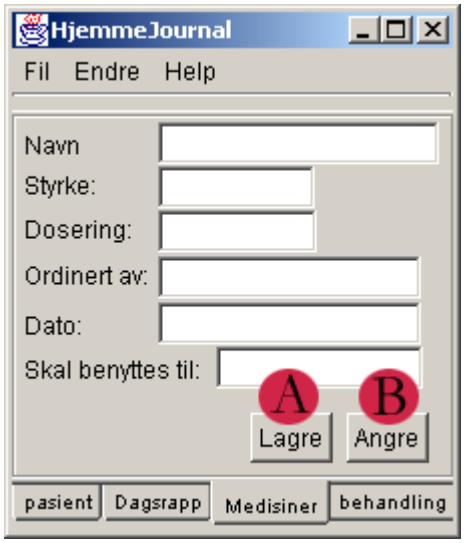
Figur 3-20 Medisin

Her kan pleieren se hvilke medisiner en pasient skal ha. I "comboboxen" (**A**) velges det mellom de ulike medisinene. Det kommer opp mer informasjon om medisinen i **B**.



Figur 3-21 Endre medisin

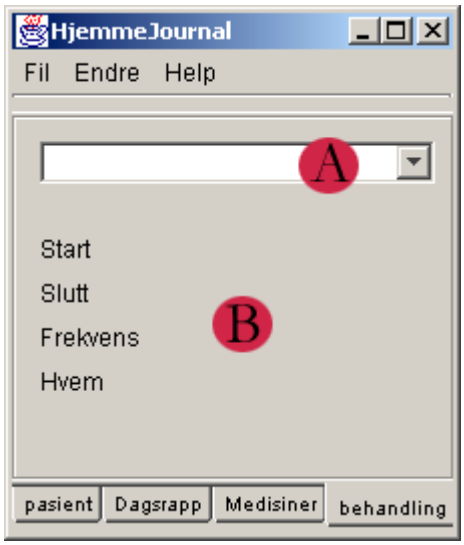
Når en medisin velges i Figur 3-20 og menyen endre velges kommer brukeren hit. Der bokstaven **A** står vil navnet på medisinen komme opp. Dette kan ikke endres. De andre opplysningene om medisinen vil vises i tekstfeltet og brukeren kan fritt endre disse så lenge verdien holdes på et gyldig format.



Figur 3-22 Ny medisin

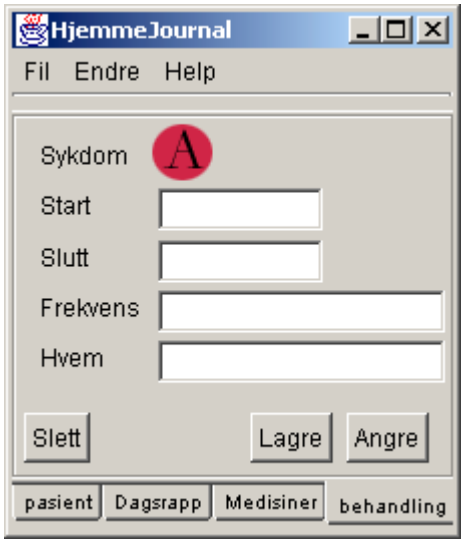
Det er også mulighet for å legge inn ny medisiner som pasienten har fått. Her må alle data fylles ut. Brukeren lagrer medisinen i databasen ved å trykke på lagre(**A**). Angre(**B**) tar brukeren tilbake til å se på de allerede innlagte medisinene.

Behandling



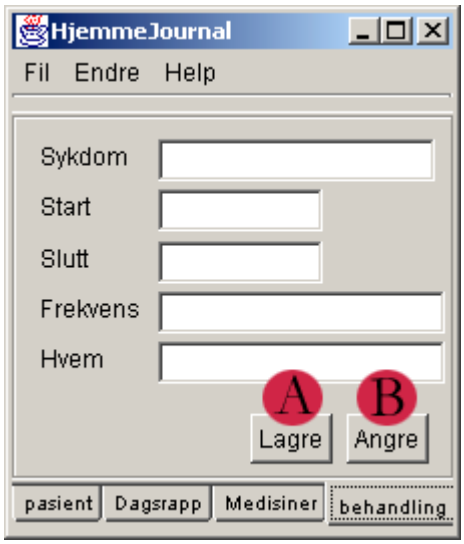
Figur 3-23 Behandling

Behandling er omtrent akkurat lik medisin, det er bare dataene som er forskjellig. De ulike behandlingene velges i ”comboboxen”(**A**). Mer data om den valgte medisinen vil komme opp ved bokstaven **B.**



Figur 3-24 Endre behandling

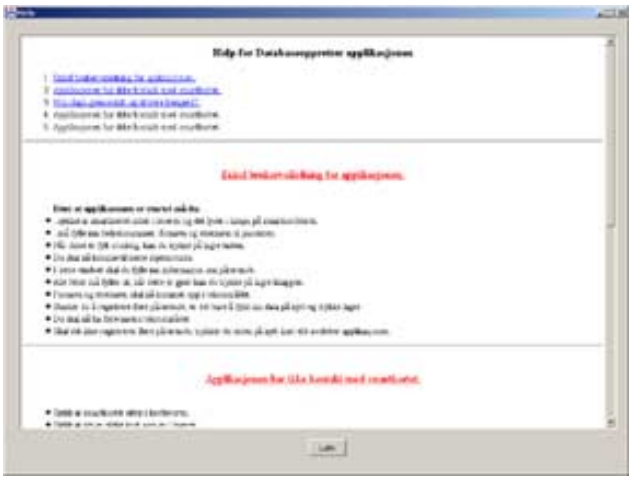
Endre behandling er helt lik endre medisin. Eneste forskjellen er at det er ulike data som lagres. Se Figur 3-21.



Figur 3-25 Ny behandling

Ny behandling er helt lik ny medisin. Se Figur 3-22.

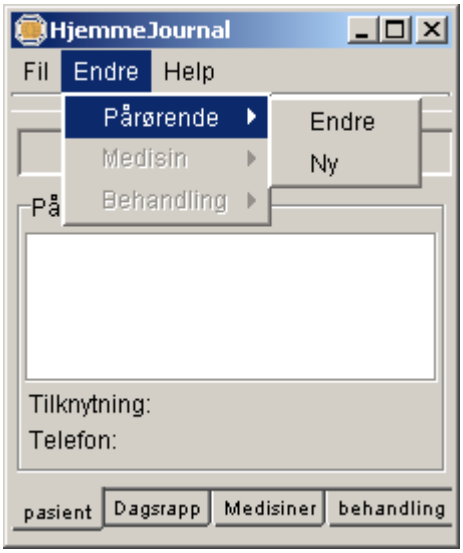
Hjelp



Figur 3-26 Hjelp

Hvis brukeren velger å bruke hjelpefunksjonen kommer dette skjermbildet opp. Skjermbildet tilpasser seg automatisk størrelsen på skjermen. Hjelpefunksjonen er bygd opp på en slik måte at en html-side vises. Når hjelpefunksjonen lukkes kommer brukeren tilbake til skjermbildet han var i.

Meny



Figur 3-27 Meny

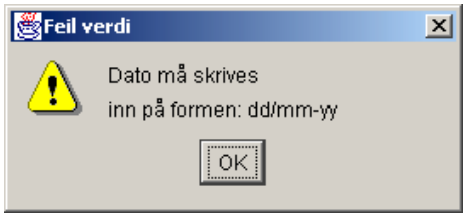
Strukturen i menyen er oppbygd slik:

- Fil:
  - Avslutt
- Endre:
  - Pårørende
    - Endre
    - Ny
  - Medisin
    - Endre
    - Ny
  - Behandling
    - Endre
    - Ny
- Hjelp
  - Hjelp
  - Om

Alle valg på fil og hjelp menyen skal være tilgjengelig hele tiden. På endre menyen er det annerledes. Pårørende skal kun være tilgjengelig i når "tabsheetet" pasient er valgt. Det er da mulighet for å legge inn en ny pårørende. Kun når en pårørende er valgt fra lista skal menyen endre gjøres aktiv. Det samme gjelder for medisin og behandling.

Feilmeldinger/Tilbakemeldinger

Feilmeldingene som blir vist vil være av samme type som i DBO-applikasjonen (se side 34). For at feilmeldingene skal få plass på skjermen på en PDA vil teksten være plassert på flere linjer enn i DBO. Selve dialogboksen vil bli litt for stor til å få plass på skjermen, men java kutter automatisk både tekst og dialogboks på høyre side slik at ingenting kommer utenfor skjermen.



Figur 3-28 Feilmelding

3.3.2 Pseudokode

if(brukernavn && passord ==OK) {
  startApplikasjon()
}
else if (avbryt) {
  exit(0)
	Systemet startes og passordkontroll vises. Hvis passord/brukernavn er ok startes selve applikasjonen opp.
	
	Hvis bruker trykker avbryt, avsluttes programmet.

}
else {
  prøv på nytt
}
switch(tab)
case(tabPersonopplysn){
  SQLSelect(Personoppl);
  SQLSelect(Pårørende);
  while (pårørende){
  leggIListe();
  }
 	
  if(Pårørende == selected) {
  SQLSelect(pårørende);
  }
  if(menyLeggTilNy) {
  visLeggInnPårørende();
  }
  if(MenyEndre) {
  visEndrePårørede();
  }
}
visLeggInnPårørende(){
  visSide();
  if(btnLagre){
  feilsjekk()
  SQLInsertData(pårørende);
  }
}
visEndrePårørende(){
  visSide();
  leggInnTekst(fornavn, tilknytning);
  if(btnEndre){
  feilsjekk()
  SQLUpdate(pårørende);
  }
  if(btnSlett){
  SQLDelete(pårørende)
  }
}
 	
case(tabdagsrapport){
  SQLSelect(dagsrapport);
  Spinbox.add(dagsrapport);
  if(spinbox forandres){
	Hvis passord/brukernavn ikke er godkjent får man spørsmål om brukernavn og passord på nytt.
	Alt ettersom hvilken tabsheet som velges.
	Tabsheet: Personopplysninger
	
	Henter personopplysninger fra database.
	Henter data om pårørende.
	Så lenge det finnes pårørende
	Legger inn navnet i velgbar liste.
	
	Hvis det velges en pårørende utføres en Database-spørring og flere data om den pårørende vises.
	Hvis Legg til ny velges i menyen.
	
	Hvis endre velges fra menyen.
	
	Viser skjermbilde for å legge inn pårørende.
	Bruker fyller inn i tekstfelt og trykker lagre.
	Data sjekkes for feil og lagres i databasen hvis alt er ok.
	
	Viser skjermbilde for å endre/slett pårørende.
	Fornavn og tilknytning kan ikke endres og legges inn som labeler. Resten i tekstfelt. Når bruker trykker endre sjekkes data for feil og de lagres i databasen.
	
	Hvis bruker trykker slett slettes den pårørende fra databasen.
	
	Tabsheet:dagsrapport
	Hent dagsrapporter fra databasen.
	Legg dagsrapportene i en spinbox. Hvis spinboxen forandres skal den valgte dagsrapporten vises i resten av vinduet

visValgtDagsrapport();
}
if(btnSkrivNy){
if(btnLagre){
feilsjekk()
SQLInsert(dagsrapport);
}
if(btnLes){
tilbakeTilLes();
}
}
}
	dagsrapporten vises i resten av vinduet.
	
	Hvis bruker trykker skriv ny. Brukeren må fylle inn med data. Hvis han trykker lagre blir data kontrollert for feil og lagret i databasen hvis alt er ok.
	
	Hvis bruker trykker les rapport kommer han tilbake og får mulighet til å lese rapporter. Han kan gå tilbake til den han skriver med ny rapport knappen.

case(tabMedisinering){
SQLSelect(medisin);
combobox.add(medisin);
if(combobox forandres){
visValgtMedisin();
}
if(menyLeggTilNy) {
visLeggTilNy();
if(btnLagre){
feilsjekk()
SQLInsert(medisin);
}
if(btnAngre){
BlankTekstfelt();
TilbakeTilLes();
}
}
if(menyEndre) {
if(btnLagre){
feilsjekk()
SQLUpdate(medisin);
}
if(btnSlett){
SQLDelete(medisin);
}
if(btnAngre){
blankTekstFelt();
tilbake();
}
}
}
	Tabsheet: Medisinering
	Henter data fra databasen.
	Legger inn medisinene i en combobox
	Hvis comboboxen forandres skal den medisinen som vises oppdateres med en ny.
	
	Hvis bruker trykker på legg til ny så byttes det til legg inn ny siden. Brukeren må skrive inn i tekstfelt.
	Innhold i tekstfelt sjekkes for feil og data lagres i databasen.
	
	Hvis bruker trykker angre byttes det tilbake til se på medisin og tekstfelt tømmes for innhold.
	
	Hvis bruker velger endre medisin. Byttes til denne siden og brukeren må skrive inn i tekstfeltene. Hvis data er ok oppdateres databasen.
	
	Hvis bruker trykker slett.
	Medisinen slettes fra databasen.
	
	Hvis bruker trykker på angreknappen
	Tekstfelt tømmes og
	bruker sendes tilbake til se på medisin.

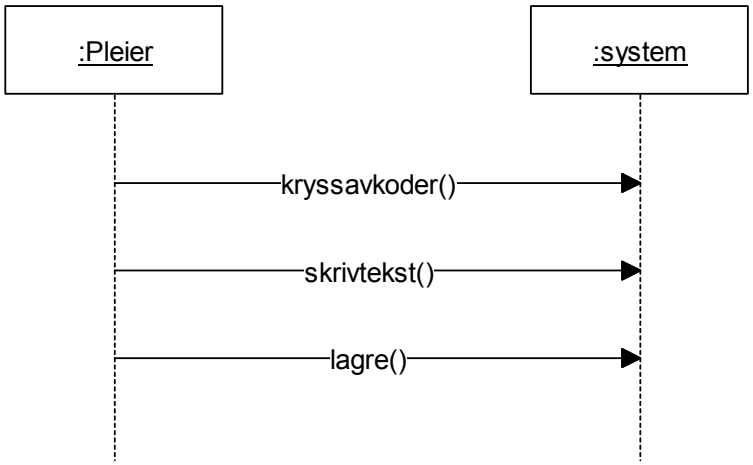
case(tabBehandling){ SQLSelect(behandling); combobox.add(behandling);	Tabsheet:Behandling Henter behandlinger fra databasen. Legger inn behandlinger i databasen

if(combobox forandres){
visValgtBehandling();
}
if(menyLeggTilNy) {
if(btnLagre){
feilsjekk()
SQLInsert(behandling);
}
if(btnAngre){
tømFelter();
tilbake();
}
}
if(menyEndre) {
visendre()
if(btnLagre){
feilsjekk()
SQLUpdate(behandling);
}
if(btnSlett){
SQLDelete(behandling);
}
}
}
}
	Hvis comboboxen forandres skal behandlingen som vises oppdateres.
	
	Hvis legg til ny behandling velges skal tekstfelter fylles ut av brukeren. Hvis brukeren trykker lagre blir innholdet i feltene kontrollert og data lagret i databasen hvis innholdet er ok.
	Hvis bruker trykker angre skal innholdet i tekstfeltene slettes og bruker sendes tilbake til å se på behandling.
	
	Hvis bruker trykker endre sendes han til endrebehandlingssiden.
	Hvis bruker trykker lagre sjekkes data for feil og databasen oppdateres.
	
	Hvis bruker trykker på slett, slettes den valgte behandlingen fra databasen.

3.3.3 Systemutviklingsmodeller

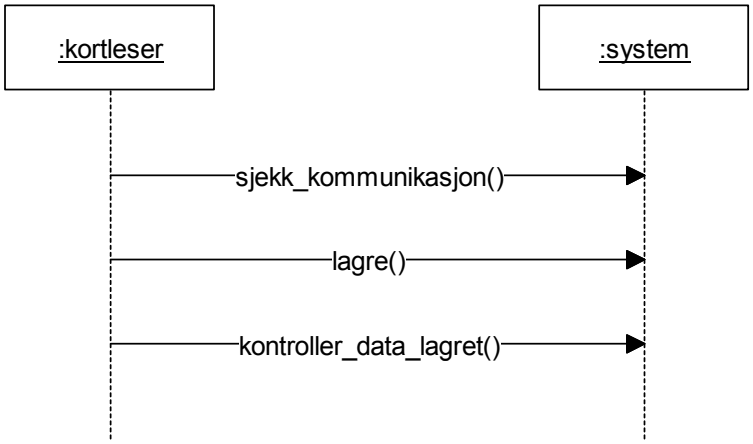
Systemutviklingsmodellene [8] vil bli såpass enkle og like at vi har ikke laget systemsekvensdiagram for mer enn to av use-casene.

Use-case:legg inn dagsrapport



Figur 3-29 Systemsekvensdiagram

Use-case:lagre data



Figur 3-30 Systemsekvensdiagram

4 IMPLEMENTERING

All kode som har blitt produsert ligger i sin helhet på den vedlagte CD'en. Det vil i avsnitt 4.5 bli gitt eksempler på enkelte deler av kildekoden.

4.1 Valg av utviklingsverktøy

Oppgaven fra oppdragsgiver gikk ut på at vi skulle kode en applikasjon i java. Vi har valgt å bruke Jbuilder 4. Valget var egentlig enkelt ettersom Jbuilder er det utviklingsverktøyet som brukes hos ErgoSolutions. Ved å bruke dette har vi også hatt mulighet til å spørre de ansatte på ErgoSolutions om hjelp. Mer om hvordan det var å bruke JBuilder finnes i kapittel 7.1.

4.2 Koding - prinsipper og erfaringer

Siden java er et objektorientert språk [3] har vi delt programmet opp i flere klasser. Dette gjør at flere har kunnet jobbe på forskjellige deler samtidig. Flere små klasser gjør det enklere å finne feil og mangler i koden. På grunn av alle de visuelle komponentene har den ene klassen blitt en god del større enn de andre. Vi oppdaget for sent at vi burde ha jobbet mer med å dele opp denne klassen, for det ble vanskelig å holde oversikten når fila ble såpass stor.

Delingen i flere klasser gjør at vi har mulighet for å bruke de samme funksjonene flere ganger. Dette gjelder spesielt funksjonene i feilsjekk-klassen. Funksjonene her brukes hver gang data lagres i databasen. Klassen dbaccess blir også flittig brukt og den brukes for å utføre databaseoperasjonene.

Vi satte oss tidlig som et mål at all kode skulle dokumenteres på en god måte. Hensikten er at andre skal kunne sette seg inn i koden på en rask måte. Dette gjelder også for oss, ettersom vi hver for oss har laget funksjoner de andre på gruppa har måtte sette seg inn i. Vi valgte å skrive kommentarer for hver funksjon isteden for å kommentere enkeltlinjer. På denne måten får vi oversikten uten å bruke alt for masse arbeid på kommentering. For å få en rask og grei oversikt over koden har vi brukt JavaDoc [3] til å lage dokumentasjon. Html-filene som JavaDoc genererer finnes på den vedlagte CD'en.

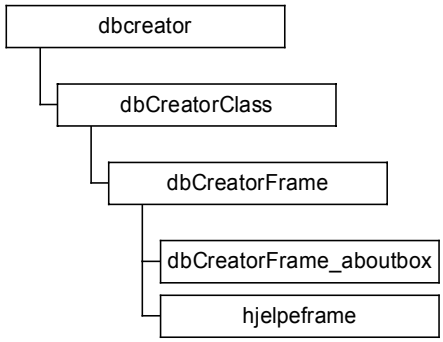
Når vi satte i gang med kodingen oppdaget vi raskt at javakunnskapene våre er temmelig begrenset. Et av våre første problem var å hente data fra databasen og legge disse inn i velgbare lister/"comboboxer". For å finne ut av dette brukte vi internettsidene til Borland [1]. Når vi fant ut av hvordan vi skulle legge inn i lister og velge ut noe derfra, fikk vi problemer med få data ut fra databasen på en fornuftig måte. Data fra databasen blir lagret i "resultset". Standard "resultset" går det bare an å lese forover i, ikke bla tilbake å finne mer om en allerede innlest post. Etter å ha brukt hjelpefunksjonen iherdig klarte vi å komme oss rundt dette problemet også.

”Spinboxen” som finnes på dagsrapport var også en lei nøtt å knekke. Det finnes ikke noen slik komponent i swing-pakka. Vi endte opp med å bruke ei liste der det kun vises ei og ei linje i gangen.

Selv om vi har hatt enkelte problemer, synes vi selv at det har vært overraskende lite problemer i selve implementasjonsdelen. Det viste seg jo at vi skulle få en del andre problemer som fikk følger for selve kodingen. Mer om disse finnes i kapittel 7.1.

4.3 Overordnet struktur

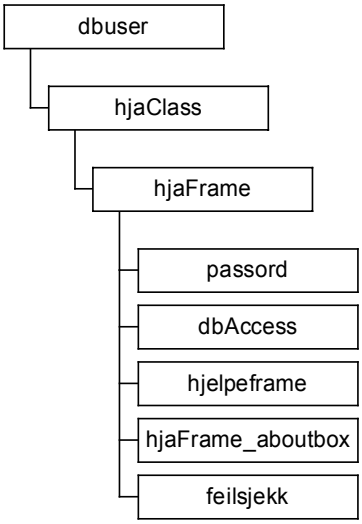
DBO



Figur 4-1 Struktur – DBO

dbcreator: Alle filene i DBO tilhører prosjektet dbcreator. dbcreator er ingen egen klasse og har ikke noe ansvar.
dbcreatorClass: Når programmet skal startes er det denne klassen som kjøres. Denne klassen gjør ikke så mye annet enn å starte dbCreatorFrame.
dbCreatorFrame: Dette er klassen som inneholder de fleste av de visuelle komponentene og det aller meste skjer. DBO gjør ikke så mye, og det vil være lite å spare på å splitte opp denne i flere klasser.
dbCreatorFrame_aboutbox: Viser aboutbox.
hjelpeframe: Ansvaret for alt som har med hjelp å gjøre. Denne klassen har vi funnet i et tidligere hovedprosjekt [7]. Vi har gjort visse endringer for at den skal passe inn i applikasjonen vår.

HJA



Figur 4-2 Struktur – HJA

dbuser: Alle filene i HJA applikasjonen hører til prosjektet dbuser. dbuser er ingen egen klasse og har ikke noe ansvar.
hjaClass: Hovedklassen. Det er denne som kjøres når programmet skal startes. Har ansvaret for å starte hjaFrame.
hjaFrame: Klassen som har ansvaret for nesten alle GUI-komponentene. Unntaket er enkelte dialogbokser og passordkontrollen. Alt det andre finnes i denne klassen. Klassen tar seg av det aller meste, de underliggende klassene er kun ”hjelpe-klasser”.
passord: Har ansvar for at passord kontrolleres.
dbAccess: Tar seg av all kommunikasjon mot databasen.
hjelpeframe: Har alt ansvaret for at hjelpefunksjonen vises. Hentet fra et tidligere hovedprosjekt [7].
hjaFrame_aboutbox: Viser aboutboksen.
feilsjekk: Har ansvaret for å kontrollere at data fra brukeren er tastet inn på et gyldig format.

4.4 Oversikt over funksjoner

Her følger en oversikt over funksjoner i de mest sentrale klassene. For å få en oversikt over alle viser vi til dokumentasjonen som medfølger på CD.

4.4.1 DBO

dbCreatorFrame

`btnLagre_actionPerformed(java.awt.event.ActionEvent e)` Kaller funksjon som oppretter tabeller.
`btnParorende_actionPerformed(java.awt.event.ActionEvent e)` Bytter side i card-layouten

btnPrLagre_actionPerformed(java.awt.event.ActionEvent e) Lagrer pårørende i databasen
feil(java.lang.String f) Viser feilmelding
jbInit() Component initialization
jMenuFileExit_actionPerformed(java.awt.event.ActionEvent e) File | Exit action performed
jMenuFileNy_actionPerformed(java.awt.event.ActionEvent e) Gjør klart for at et nytt kort kan lagres.
jMenuHelpAbout_actionPerformed(java.awt.event.ActionEvent e) Help | About action performed
jMenuItem1_actionPerformed(java.awt.event.ActionEvent e) Hjelp| Hjelp action performed Viser hjelp i et eget vindu
lagtabell() Oppretter tabellene på kortet.
parorendeSjekk() Skjekker om verdiene bruker har tastet inn om en pårørende har lovlig verdi
personinfoSjekk() Sjekker om verdiene bruker har fylt ut om en pasient har lovlig verdi
processWindowEvent(java.awt.event.WindowEvent e) Overridden so we can exit when window is closed
sjekkNummer(java.lang.String temp) Sjekker at den medsendte stringen bare består av siffer
sjekkstring(java.lang.String temp) Sjekker at den medsendte stringen bare består av bokstaver
this_windowOpened(java.awt.event.WindowEvent e) programmet startes
updatetaPrReg() Oppdaterer taPrReg med de pårørende som er registrert på denne personen

dbAccess

close() Avslutter forbindelsen med databasen.
dbConnect() Oppretter forbindelse med databasen.
executeSQL(java.lang.String sql) Kjører SQL mot databasen. Returnerer et resultset.
executeUpdateSQL(java.lang.String sql) Kjører SQL mot databasen. Returnerer ingenting.

4.4.2 HJA

HjaFrame

BNlagre() Lagrer en ny behandling i databasen.
btnBEangre_actionPerformed(java.awt.event.ActionEvent e) Går tilbake til vis behandlinger
btnBElagre_actionPerformed(java.awt.event.ActionEvent e) Kjøres når bruker prøver å endre data om en behandling.
btnBEslett_actionPerformed(java.awt.event.ActionEvent e) Sletter den valgte behandlingen fra databasen
btnBNangre_actionPerformed(java.awt.event.ActionEvent e) Går tilbake til vis behandlinger
btnBNlagre_actionPerformed(java.awt.event.ActionEvent e) Kalles når bruker prøver å legge til en ny behandling.
btnEMEangre_actionPerformed(java.awt.event.ActionEvent e) Viser endre medisin siden
btnEMEendre_actionPerformed(java.awt.event.ActionEvent e) Når bruker trykker på lagre kjøres denne funskjonen som kaller en funksjon i feilsjekk.
btnEMEslett_actionPerformed(java.awt.event.ActionEvent e) Sletter den valgte medisinen fra databasen
btnEMNangre_actionPerformed(java.awt.event.ActionEvent e) Går tilbake til vis medisinering siden
btnEMNlagre_actionPerformed(java.awt.event.ActionEvent e) Kalles når bruker trykker lagre ny medisin.
btnEPEendre_actionPerformed(java.awt.event.ActionEvent e) Kaller funksjon i feilSjekk som kommer med feilmelding om noe er feil.
btnEPEslett_actionPerformed(java.awt.event.ActionEvent e) Sletter pårørende fra tilknytn-tabellen
btnEPNlagre_actionPerformed(java.awt.event.ActionEvent e) Kaller funksjon i feilSjekk som kommer med feilmelding om noe er feil.
btnLDRTnyttBesok_actionPerformed(java.awt.event.ActionEvent e) Bytter til skriv dagsrapport i cardlayouten
btnPEPEtilbake_actionPerformed(java.awt.event.ActionEvent e) Går tilbake til vis pårørende
btnPEPNtilbake_actionPerformed(java.awt.event.ActionEvent e) Går tilbake til Vis pårørende
btnSDRlagrer_actionPerformed(java.awt.event.ActionEvent e) Når bruker trykker på lagre.
btnSDRTlesdagsrap_actionPerformed(java.awt.event.ActionEvent e) Tilbake til les dagsrapport
cmbBehandling_actionPerformed(java.awt.event.ActionEvent e) Ved valg av behandling i comboboxen vises mer data i teksfeltene.
cmbMedisinNavn_actionPerformed(java.awt.event.ActionEvent e) Ved valg av medisin i comboboxen vises mer data i tekstfeltene.

EBlagre() Oppdaterer data om en behandling i databasen
EMEendre() Oppdaterer databasen med endringene som brukeren har gjort.
EMNlagre() Lagrer medisindata i databasen.
EPEendre() Oppdaterer data i databasen
EPNlagre() Lagrer nye pårørende i databasen
fetchdata() Oppdater tekstfelt og avkrysningsbokser med data fra databasen.
jbInit() Component initialization
jMenuFileExit_actionPerformed(java.awt.event.ActionEvent e) Fil | Avslutt action performed
jMenuHelpAbout_actionPerformed(java.awt.event.ActionEvent e) Hjelp | Om action performed
lstTilknytn_valueChanged(javax.swing.event.ListSelectionEvent e) Legger de pårørendes telefonnummer og tilknytning ut i tekstfeltene når en pårørende velges i lista.
MainTabSheet_stateChanged(javax.swing.event.ChangeEvent e) Viser den valgte tabsheeten.
menEndreBehandling_actionPerformed(java.awt.event.ActionEvent e) Endre| Behandling| Endre action performed
menEndreMedisinering_actionPerformed(java.awt.event.ActionEvent e) Endre| Medisin| Endre action performed
menEndrePaarorende_actionPerformed(java.awt.event.ActionEvent e) Endre| Pårørende| Endre action performed
menHjelp_actionPerformed(java.awt.event.ActionEvent e) Hjelp| Hjelp action performed
menLeggTilBehandling_actionPerformed(java.awt.event.ActionEvent e) Endre| Behandling| Endre action performed
menLeggTilMedisinering_actionPerformed(java.awt.event.ActionEvent e) Endre| Medisin| Ny action performed
menLeggTilPaarorende_actionPerformed(java.awt.event.ActionEvent e) Endre| Pårørende| Ny action performed
oppdaterEndreMed() Setter inn tekst i tekstfelt i endre medisin siden.
pgsLDRTbyttSide_adjustmentValueChanged(AdjustmentEvent e) Bytter mellom tekstfelt og avkrysningsbokser i les dagsrapport
pgsSDRTbyttSide_adjustmentValueChanged(AdjustmentEvent e) Bytter mellom tekstfelt og avkrysningsbokser i skriv dagsrapport
processWindowEvent(java.awt.event.WindowEvent e) Overridden so we can exit when window is closed

SDRlagre() Lagrer dagsrapport i databasen.
spspin_adjustmentValueChanged(java.awt.event.AdjustmentEvent e) Når verdien i spinboksen forandres kjøres denne funksjonen som finner ut hvilken dagsrapport som skal vises.
Tab0Valgt() Kjøres når tabsheetet pasient blir valgt.
Tab1Valgt() Kjøres når tabsheetet dagsrapport velges.
Tab2Valgt() Kjøres når tabsheet medisin velges. fyller comboboxen med verdier fra databasen.
Tab3Valgt() Kjøres når tabsheet behandling velges.
taSDR1div_keyPressed(java.awt.event.KeyEvent e) Sjekker at inntastet dagsrapport ikke blir lenger enn 170 tegn

dbAccess

Lik dbAccess for DBO se side 54.

Feilsjekk

EBlagretest(java.lang.String sykdom, java.lang.String start, java.lang.String slutt, java.lang.String frekvens, java.lang.String hvem) kontrollerer at strengen som er medsendt er et gyldig
EMElagretest(java.lang.String type, java.lang.String dosering, java.lang.String ordDato, java.lang.String ordinert, java.lang.String styrke, java.lang.String varighet) kontrollerer at strengen som er medsendt er gyldig.
EPNlagretest(java.lang.String tlf, java.lang.String fnavn, java.lang.String enavn) kontrollerer at strengen som er medsendt er et gyldig telefonnummer, fornavn og etternavn
feil(java.lang.String f) Viser feilmelding
SDRlagretest(java.lang.String pleier) kontrollerer at strengen som er medsendt er et gyldig navn
sjekkDato(java.lang.String temp) Sjekker at den medsendte stringen er en en dato på et gyldig format
sjekkLengde(java.lang.String temp, int lengde, int sjekk) Sjekker at medsendt streng er så lang som den har lov til å være
sjekkNummer(java.lang.String temp) Sjekker at den medsendte stringen bare består av siffer
sjekkstring(java.lang.String temp) Kontrollerer at den medsendte stringen bare består av bokstaver.

4.5 Eksempler på kode

Viser hvordan tabellene opprettes på kortet.

```
/**Oppretter tabellene på kortet. Alle feltene er av typen varchar.*/
  void lagtabell(){
    try {
      String sql="create table tilknytn(fnavn varchar,"
        + "enavn varchar, tilknytn varchar, tlfnr varchar)";
      dbAccess.executeUpdateSQL(sql);
      sql="create table persoppl(fnavn varchar, enavn varchar,"+
            "fnr varchar)";
      dbAccess.executeUpdateSQL(sql);
      sql="create table behandl(start varchar, slutt varchar,"+
            "oppdrart varchar,frekvens varchar, hvem varchar)";
      dbAccess.executeUpdateSQL(sql);
      sql="create table dagsrapp(pleier varchar, datotid varchar,"+
            "diverse varchar, diverse2 varchar, diverse3 varchar,"+
            "diverse4 varchar, diverse5 varchar)";
      dbAccess.executeUpdateSQL(sql);
      sql="create table medisin(navn varchar, styrke varchar,"+
            "dosering varchar, ordinert varchar, dato varchar,"+
            "til varchar)";
      dbAccess.executeUpdateSQL(sql);
    }catch(SQLException sqle) {
      int ok = JOptionPane.showConfirmDialog(this,
        "Kunne ikke opprette tabeller.\n "
        + "Trykk OK for å prøve igjen eller avbryt for å avslutte"
        + "programmet.",
        "Kan ikke opprette tabeller",
        JOptionPane.OK_CANCEL_OPTION, JOptionPane.ERROR_MESSAGE);
      if (ok == JOptionPane.OK_OPTION)
        lagtabell();
      else
        System.exit(1);
    }
}
```

Viser hvordan teksten som kan lagres i dagsrapport deles opp i flere stringer på maks 34 tegn hver.

```
for (int i=0; i<4; i++){
  if(taSDR1div.getText().length() - ((1+i)*34) > 0 )
    div[i]=taSDR1div.getText().substring((i*34),(i*34+34));
  else{
    div[i]=taSDR1div.getText().substring(i*34,
      taSDR1div.getText().length());
    break;
  }
}
```

Viser hvordan vi kontrollerer at en dato har korrekt format.

```
  /**Sjekker at den medsendte stringen er en en dato på et gyldig
format
  * @param temp Stringen som skal kontrolleres.
  */
  public boolean sjekkDato(String temp){
    String a1, a3, a5;
    char a2,a4;
    Date dato = new Date();

    if(temp.length()==8){
      a1=temp.substring(0,2);
      a2=temp.substring(2,3).charAt(0);
      a3=temp.substring(3,5);
      a4=temp.substring(5,6).charAt(0);
      a5=temp.substring(6,8);
    }
    else {
      feil("Dato må skrives \ninn på formen: dd/mm-yy");
      return false;
    }
    if (a2!='/'){
      feil("Dato må skrives \ninn på formen: dd/mm-yy");
      return false;
    }
    else if (a4!='-'){
      feil("Dato må skrives \ninn på formen: dd/mm-yy");
      return false;
    }
    else {
      DateFormat datoform = new SimpleDateFormat("dd/MM-yy");
      datoform.setLenient(false);
      try{
        dato=datoform.parse(temp);
      }
      catch(IllegalArgumentException iae){
        feil("Ugyldig dato");
        return false;
      }
      catch(ParseException pe){
        feil("Ugyldig dato");
        return false;
      }
      return true;
    }
  }
```

5 KVALITETSSIKRING OG TESTING

5.1 Kvalitetssikring av prosessen

Det ble i analysefasen satt opp et ganttskjema. Dette har hele tiden vært veiledende for hvordan vi har jobbet. Det ble også satt opp en arbeidsfordeling som fordelte arbeidsoppgaver hos de ulike gruppedeltakerne. Vi har under hele prosessen hatt løpende kontakt med veileder. Dette har vært til god hjelp ved uklarheter og tvilstilfeller i oppgaven. Problemer har blitt tatt tak i på et tidlig tidspunkt og unødig tap av tid er unngått. ErgoSolution har bistått oss på en god måte. Der vår kunnskap ikke har strukket til har de kommet med råd og innspill. I programmeringsfasen har vi fått hjelp ved konkrete problemer. Momentene over har bidratt til å kvalitetssikre arbeidet vårt gjennom hele prosjektet.

5.2 Testfasen

I prosjektet har det ikke vært noen klar skillelinje mellom implementasjonsfasen og testfasen. Det har vært kontinuerlig testing hele tiden. Kodingsfasen er delt inn i tre iterasjoner noe som også har stykket arbeidet naturlig opp. Intensjonen har hele tiden vært at hver utvidelse av programmet skulle testes separat, og at vi til slutt skulle sitte med en testbar totalløsning. Vi skulle utvikle ett inkrement ferdig, før vi går videre til neste. Systemet skulle gradvis bygges opp av en og en komponent som allerede virket og var ferdig testet.

Da vi begynte å programmere på de to applikasjonene ble alle data lagret i en access database. Dette var ment som en midlertidig løsning som bare skulle brukes under testingen av programmene. Senere skulle denne byttes ut med smartkortet. På grunn av problemene med smartkortet(se kapittel 7.1), faller det automatisk bort mange mulige feil. En stor del av testfasen er blitt borte.

Vi bestemte tidlig at den som lager en funksjon ikke tester den. Vi har derfor stadig byttet på å programmere og teste. Nå mot slutten av kodingen har vi også presentert programmene for både oppdragsgiver og veileder. De har begge fungert som testpersoner.

5.3 Testingen

Vi har først valgt å teste de to applikasjonene hver for seg, for deretter å teste de sammen. I den siste testfasen vil DBO lage databasen først og deretter vil HJA jobbe på den samme.

5.3.1 Test av DBO

Test nr.	Test	Resultat
1	Sjekke standard Windowsfunksjoner for DBO	OK
2	Sjekke alle menyfunksjonalitet	OK
3	Alle feilsjekker på inntastingsverdier virker som de skal	OK
4	”Lagre personopplysninger” knappens funksjonalitet virker mot databasen	OK
5	Sjekke at alle tabeller blir opprettet riktig	OK
6	Sjekke at alle data blir skrevet korrekt i databasen	OK
7	”Lagre pårørende” knappens funksjonalitet virker mot databasen	OK

Vi fant ingen graverende feil under testing av DBO. Etter at det ble gjort en liten justering av databasen under arbeidet med HJA ble det nødvendig at de samme tilpasningene ble gjort i DBO. Dette var vi imidlertid klar over og denne jobben ble gjort før testingen kom skikkelig i gang.

5.3.2 Test av HJA

I denne delen valgte vi å dele testingen opp slik at vi først ser på de vanlige windowsfunksjonalitetene, og deretter ser på hver av de fire ”tabsheetene”.

Sjekk av de generelle funksjonene.

Test nr.	Test	Resultat
1	Sjekke standard Windowsfunksjoner for passord inngang	OK
2	Sjekke at passordinngangen virker som den skal	OK
3	Sjekke standard Windowsfunksjoner for HJA	OK

Funksjonalitetstest av ”Pasient-tabsheetet”

Test nr.	Test	Resultat
1	Siden initieres riktig når ”tabsheetet” blir valgt	OK
2	Ved valg av en pårørende skal tilknytning og tlf.nr vises.	OK
3	Menyvalg ”endre pårørende” virker som de skal. (1)	~~FEIL~~ Rettet
4	Alle feilsjekker på inntastingsverdier virker som de skal	OK
5	”Slette” knappens funksjonalitet virker mot databasen	OK
6	”Endre” knappens funksjonalitet virker mot databasen	OK
7	Begge ”Lagre” knappenes funksjonalitet virker mot databasen	OK
8	Begge ”Angre” knappene tar deg riktig tilbake til ”vis pårørende”	OK

1) Etter at en pårørende er slettet, skal det ikke være mulig å endre pårørende før en ny er valgt i lista. Dette er utbedret.

Funksjonalitets test av "Dagsrapport-tabsheetet"

Test nr.	Test	Resultat
1	Siden initieres riktig når "tabsheetet" er valgt	OK
2	Sidetall vises og funksjonalitet for bytte av side i "Les dagsrapport" virker som den skal.	OK
3	Dato og pleier for besøket vises. Funksjonalitet for å bla frem- og bakover i rapporten virker som den skal.	OK
4	"Nytt besøk" knappen tar deg til "skriv dagsrapport".	OK
5	Sidetall vises og funksjonalitet for bytte av side i "skriv dags rapport" virker som den skal.	OK
6	"Lagre" knappenes funksjonalitet virker mot databasen.	OK
7	"les dagsrapport-knappen" tar deg til "les dagsrapport".	OK

Funksjonalitets test av "Medisin-tabsheetet"

Test nr.	Test	Resultat
1	Siden initieres riktig når tabsheetet blir valgt	OK
2	Funksjonalitet for oppdatering av teksten når en annen medisin blir valgt	OK
3	Menyvalg "endre pårørende" virker som den skal.	OK
4	Alle feilsjekker på inntastingsverdier virker som de skal	OK
5	"Slette" knappens funksjonalitet virker mot databasen	OK
6	"Endre" knappens funksjonalitet virker mot databasen	OK
7	Begge "Lagre" knappenes funksjonalitet virker mot databasen	OK
8	Begge "Angre" knappene tar deg riktig til bake til "vis" pårørende	OK

Funksjonalitets test av "Behandling-tabsheetet"

Test nr.	Test	Resultat
1	Siden initieres riktig når tabsheetet blir valgt	OK
2	Funksjonalitet for oppdatering av teksten når en annen behandling blir valgt	OK
3	Menyvalg "endre behandling" virker som den skal.	OK
4	Alle feilsjekker på inntastingsverdier virker som de skal	OK
5	"Slette" knappens funksjonalitet virker mot databasen	OK
6	"Endre" knappens funksjonalitet virker mot databasen	OK
7	Begge "Lagre" knappenes funksjonalitet virker mot databasen	OK
8	Begge "Angre" knappene tar deg riktig tilbake til "vis pårørende"	OK

5.3.3 Test av totalprosjektet

Når vi nå skal se nærmere på systemet som en helhet ønsker vi å teste de to applikasjonene sammen. Det som er ønskelig er at DBO oppretter tabellene og fyller inn korrekte data. Deretter skal den samme databasen benyttes av HJA.

Test nr.	Test	Resultat
1	Sjekke at den databasen som DBO lager er riktig (innholdet i et felt hadde byttet med innholdet i et annet) -> rettet	~~FEIL~~ Rettet
2	Sjekke at den databasen som HJA krever er riktig	OK
3	Sjekke at en kan opprette en database med DBO og at denne kan brukes av HJA (følgefeil fra test nr.1) -> rettet	~~FEIL~~ Rettet

Etter at de to feltene som hadde byttet plass ble rettet opp, var det ingen problemer med denne testen. Det er imidlertid en svakhet i systemet at det ikke er noen funksjonalitet som fanger opp lesing av uinitierte kort i HJA. Dersom en forsøker å hente data fra et kort som ikke er blitt initiert i DBO vil en få en SQL-feilmelding i HJA.

6 TRUSSELANALYSE

Dette er en veiledning for å sette opp en trusselanalyse [2]. Ved en trusselanalyse er det strengt tatt tre momenter man må ta hensyn til. Det er ”Protection Profiles (PP)”, ”Security Targets (ST)” og ”Target of Evaluation (TOE)”. Disse skal følge en standard som finnes i ISO/IEC 154048.

Det er laget en Mal for PP og ST. Disse ser slik ut.

PP:

1	Introduksjon Stikkord: Identifikasjon, Oversikt
2	TOE Beskrivelse
3	TOE Sikkerhets omgivelser/ miljø Stikkord: Antagelser, Trussel, Organisasjonens sikkerhetspolitikk
4	Objektivitets sikkerhet
5	Krav til IT sikkerhet
6	PP Applikasjonsnotater
7	Logiske forklaringer

ST: (helt lik PP, men har noen flere elementer)

1	ST Introduksjon
6a	TOE Sammendrags spesifikasjon Stikkord: TOE Sikkerhetsfunksjoner, sikkerhetsmålestokk
7	Oppkrav Stikkord: PP Referanser, PP raffinement, PP tilgang
8	Logiske forklaringer

Det skal lages en PP og en ST for hver TOE. Dette er en meget stor arbeidsoppgave. Vi skal kun ta med noen momenter fra PP som er viktig med tanke på en trusselanalyse. Ellers henviser vi til ISO/IEC 154048 for ytterligere detaljer.

6.1 Introduksjon

Hensikten med PP er å fastslå sikkerhetshull som strengt tatt omhandler en samling av systemer eller produkter, kjent som TOE. I dette tilfelle våre applikasjoner. Det skal spesifiseres krav som henvender seg mot problemene uten å ta hensyn til disse i implementasjonen. ST er forholdsvis lik PP, bortsett fra at denne inneholder ytterligere detaljert informasjon om hvordan sikkerhetskravene realiseres.
DBO og HJA er begge laget i Jbuilder 4 og har versjonsnummer 0.8.

6.2 TOE beskrivelse

DBO og HJA er applikasjoner som er tilrettelagt for omsorgssektoren. De skal lette arbeidet og lagre persondata på en sikrere måte. Data skal lagres i en database på smartkort. DBO vil opprette tabeller, lagre pasientdata og pårørende data i databasen. HJA vil kunne lese og lagre informasjon om pasienten.

6.3 TOE Sikkerhets omgivelser/ miljø

Databasen på smartkortet har et eget operativsystem, GemDB. Kommunikasjonen foregår gjennom ODBC-grensesnitt og SQL-spørringer. For å kunne hente eller lagre data i databasen trengs det en smartkort leser. Andre momenter, se begrensninger side 22 og 29.

6.3.1 Antagelser

Her nevnes bare noen antagelser som kan påvirke sikkerheten.

1. Uforsiktige brukere.
2. Tap av kort eller applikasjon.
3. Fysisk lagring av kort eller applikasjon.
4. Sikkerhetsrutiner.

6.3.2 Trusler

For en database vil det vær nødvendig å beskytte tabellene, og spesielt dataene som er lagret i disse. Trusselfaktorer vil være autoriserte og ikke autoriserte brukere av databasen. Dette kan være:

T1. En angriper får aksess til databasen ved å utgi seg for å være en autorisert bruker.
T2. En autorisert databasebruker får tilgang til informasjon som ikke er tiltenkt denne brukeren.
T3. Administrasjonsbrukere misbruker sine rettigheter som de er tildelt.
T4. Ved tap av smartkort kan en angriper får tilgang til databasen.

6.3.3 Organisasjonens sikkerhetspolitikk

Dette vil være veldig individuelt for hver enkelt organisasjon. Når det gjelder pasientdata som skal lagres og oppbevares i en database, skal man følge de regler som datatilsynet har.

6.4 Objektivitets sikkerhet

For å sikre databasen må TOE(HJA eller DBO) beskyttes.

O1. TOE systemet bør innholde virkemidler som identifiserer brukere ved pålogging.

O2. TOE systemet bør kontrollere brukere for antall aksesser og rettigheter som er satt til hvert enkelt individ. Organisasjonens sikkerhetspolitikk vil være veiledende på dette punktet.

6.5 Krav til IT sikkerhet

For å opprettholde applikasjonen:

1. For å opprettholde sikkerheten i O1, kreves det en innloggingsboks som sjekker at riktig brukernavn og passord er skrevet inn. Denne bestemmer om HJA skal åpnes eller ikke, ut i fra inntastet data.
2. For å opprettholde sikkerheten i O2, kreves det at bruker bare kan prøve å logge seg på x-antall ganger før denne brukeren blir utestengt. Er påloggingen korrekt, får denne brukeren visse rettigheter.
3. Applikasjonen vil inneholde passord til databasen på kortet, disse bør krypteres.

For databasen på kortet:

4. For å opprettholde sikkerheten i O1, krever det en innloggingsboks som sjekker at det er tastet inn riktig brukernavn og passord.
5. For å opprettholde sikkerheten i O2, kreves det at bruker bare kan prøve å logge seg på x-antall ganger før denne brukeren blir utestengt.

6.6 PP Applikasjonsnotater

HJA har en innloggingsboks som krever brukernavn og passord. Det er ikke et begrenset antall forsøk slik at en angriper kan prøve å logge seg inn så mange ganger han vil. Dette er den største trusselen i vårt system. En annen ting er at applikasjonen er pakket som en .jar fil. Da kan man åpne alle .class filene og finne brukernavn og passord lagret som en streng. For å klare dette, må man ha en del datakunnskap, men det er ofte det en angriper har. Dette må det gjøres noe med, for eksempel at brukernavn og passord blir kryptert.

Dette er kun noen elementer som viser at trusselanalyse er meget omfattende. Med tanke på at det er pasient data som skal lagres på kortet, bør det lages en fullverdig analyse til alle TOE hvis disse applikasjonene skal brukes til kommersielt bruk. Det vil være viktig å ta med det essensielle fra organisasjonspolitikken slik at alle momenter dekkes.

7 DISKUSJON AV RESULTATER

7.1 Problemer under prosjektet

Skal vi sammenlikne løsningen opp mot kravspesifikasjonen finner vi avvik. Dette skyldes tekniske problemer som har oppstått underveis og som vi ikke rår over. Disse problemene skal vi gå mer detaljert inn på.

Jbuilder

Ingen av gruppemedlemmene hadde brukt Jbuilder tidligere. Oppbyggingen av programmet er ganske likt Borland Delphi, som vi har brukt i faget "Grafisk Brukergrensesnitt". Dette var en stor fordel, da vi følte oss "hjemme" nesten med en gang. Samtidig er det store forskjeller i måten verktøyene brukes på. Siden mye var nytt og ukjent måtte vi sette av tid til opplæring. Det viste seg fort at dette er et verktøy som krever mange arbeidstimer for å kunne beherskes brukbart. Der det oppsto små problemer, som vi ikke klarte å løse, har det vært god hjelp å få hos ErgoSolutions. De har vært imøtekommende og blide.

Smartkort

GemDB er et nytt smartkort som ErgoSolutions ikke har prøvd ut før. Det er et operativsystemet og en database på kortet. Det har ikke vært utført tester opp mot java med dette kortet tidligere. Det som viste seg å bli et problem var kommunikasjonen med databasen på smartkortet. Vi får ikke kontakt med databasen på kortet gjennom java. Vi har fått hjelp fra ErgoSolutions, men de fant ikke ut av problemet. Vi sendte en e-post til GemPlus og konfronterte dem med våre erfaringer. Det viser seg at operativsystemet på kortet ikke er støttet i java. Dette medfører at vi ikke kan legge inn eller hente data fra databasen på kortet ved hjelp av java.

PDA

Det gikk lang tid før ErgoSolutions fikk tak i en PDA. Da den kom, fant vi fort ut at det er store begrensninger i å utvikle programvare for en PDA. Den Java VM som er tilpasset vår PDA har begrenset muligheter fordi den kun inneholder følgende pakker:

- java.awt
- java.io
- java.math
- java.rmi
- java.sql
- java.util.zip

Applikasjon vår er i stor grad utviklet rundt swing-pakka. Dette betyr at den ikke kan kjøres på PDA. Vi har vært i kontakt med Sun hvor vi etterlyser muligheten for å importere eksterne pakker. De hadde ikke mulighet til å svare på slike spørsmål. Vi går ut ifra at denne muligheten kommer etterhvert.

Vi har heller ikke klart å kjøre .jar filer på PDA'en. Dette er ikke så viktig når det er mulig å kjøre .class filer.

Det er heller ikke utviklet ODBC-drivere til GemDB for PDA. Hadde vi fått applikasjonen til å kjøre, hadde vi uansett ikke fått kommunikasjon med smartkortet på grunn av dette.

Utviklingsmiljø.

ErgoSolutions hadde dette året tre hovedprosjekter gående i sine lokaler, to andre prosjekter som også rettet seg mot bruk av smartkort. Gruppene fikk tildelt to rom i deres lokaler. Et rom var utstyrt med to PC'er og det andre var mest egnet til møtevirksomhet. I tillegg hadde gruppa vår fått tildelt et grupperom på skolen. Dette rommet delte vi med en annen hovedprosjektsgruppe.

Det viste seg imidlertid at dette ikke var noen særlig gunstig løsning for oss. Den ene av gruppene var avhengig av å benytte maskinene som stod i grupperommet på grunn av opphavsrettighetene på softwaren. Når vi i tillegg hadde en "annenhver dag" ordning med grupperommet på skolen ble arbeidet veldig oppstykket og bar lite preg av kontinuitet. Vi løste problemet med å flytte hele prosjektarbeidet hjem til en av gruppedeltakerne.

7.2 Det endelige resultatet kontra kravspesifikasjon

Kravspesifikasjonen vi har laget er ment for et system der database på smartkort og PDA er satt i fokus. Den endelige resultatet ble jo ikke som forventet. Vi hadde alle gledet oss til å få applikasjonen over på PDA, men slik skulle det jo ikke gå. Vi ble raskt enige om å designe løsningen på en slik måte at den enkelt kan tilpasses en PDA senere. Selve applikasjonen er laget på størrelse med skjermen på den PDA'en vi har hatt til rådighet. Vi har god tro på at det ikke er så lenge til det finnes pakker og drivere som gjør det mulig å kjøre noe slikt på PDA. Slik vi ser det er det en forholdsvis enkel oppgave å overføre applikasjonen til en PDA og kjøre den der. Da vi fant ut at vi ikke fikk kontakt med kortet, tenkte vi på mulige løsninger på problemet. Vi kom opp med noen forslag.

1. Simulere kommunikasjon med kortet/bruke SQL-logger til å kommuniseres med.
2. Lage applikasjon uten smartkortsjekker. Lagre data i en annen database.
3. Utvikle applikasjonen i et annet programmeringsspråk der kommunikasjon med kortet er støttet. Vi kunne for eksempel brukt Delphi, som genererer pascal-kode.
4. Ikke lage noen applikasjoner.

Mulighet nummer 4 ble utelukket med en gang. Vi måtte ta tak i problemene som lå foran oss og finne en løsning som både vi og ErgoSolution kan være fornøyd med. Mulighet 3 ble også glemt med en gang for oppdragsgiver krevde at vi skulle bruke java. De andre to alternativene tenkte vi en del på. Simulere kommunikasjon med kortet, nummer 1, vil virke. Vi ville fått en løsning, men den vil være utrolig tungvinn og langt fra det en hjemmesykepleier kan bruke. Vi ville måttet lage en type loggfiler over SQL-operasjonene som lages og kjøre disse i et program som kan kommunisere med kortet. Vi kom til slutt fram til at å lage applikasjonen med en access database vil

være det beste ut ifra vår situasjon. Smartkortet vi skulle brukt kommuniserer også med SQL, så forandringene vil ikke bli så veldig store. Det vi mangler før løsningen kan brukes med smartkort, er kontroller for at smartkort faktisk befinner seg i leseren. Den tiden vi sparte på å ikke lage slike tester, har vi brukt opp mange ganger før vi skjønte at det faktisk er umulig å lage systemet slik det opprinnelig var tenkt.

7.3 Mangler og svakheter i systemet.

DBO har følgende svakheter/ mangler:

- Ikke kommunikasjon med smartkortet.
- Lager/oppretter ikke databasen, den må eksistere fra før.
- Ikke laget koden for å få kontakt med smartkortet.
- Det er ikke bruk av hurtigtaster.

HJA har følgende svakheter/ mangler:

- Ikke kommunikasjon med smartkortet.
- Ikke laget koden for å få kontakt med smartkortet.
- Det er kun et passord som åpner.
- Der er ikke laget en brukerveiledning.
- Når det kommer en feilmelding plasseres ikke markøren seg i feltet som inneholder feil.

7.4 Forslag til forbedringer og videreutvikling.

Vi skulle utviklet et system som går spesifikt mot smartkortet, og som ivaretar den journalen som i dag ligger hjemme hos hver pasient. Dersom hjemmesykepleien skal slippe dobbeltarbeidet må det også utvikles et større system som ligger bak og ivaretar journalarkivet inne på hovedkontoret. Vi ser for oss at dette systemet bør virke omtrent slik:

- Pleieren kommer til en pasient, utfører sine arbeidsoppgaver og registrer det som skal inn i journalen via PDA'en. Dette lagres i pasientens smartkort og samtidig i PDA'en.
- Pleieren reiser deretter videre til de andre pasientene på dagens rute.
- Når pleierne kommer tilbake til hovedkontoret kobler hun PDA'en opp mot journalarkivet og laster alle dagens journaler over i den sentrale løsningen.

Det kan kanskje være aktuelt at pleieren kobler seg opp mot journalarkivet før hun legger ut på besøksrunden. Det kan ha skjedd endringer i journalen som må oppdateres på kortet. Disse dataene blir da liggende i PDA'en til pleieren kommer til den aktuelle pasienten. PDA'en kjenner igjen pasientens smartkort og laster de nye dataene over i smartkortet.

HJA

Det bør være en bedre identifiseringssjekk på hvem som åpner applikasjonen. HJA skal kun brukes av administrasjonen i omsorgssektoren, leger, sykepleiere og hjelpepleiere. Dette kan løses ved at de som skal ha tilgang til pasientjournalen får sitt eget smartkort, et ansattkort, med en personlig pin-kode. Når HJA åpnes, må brukeren putte sitt smartkort i leseren, taste sin kode og HJA åpnes. Applikasjonen bør da sjekke at det er tastet inn riktig kode og hente brukeres profil fra ei passord-fil. Pasientkortet settes i kortleseren og data kan behandles.
På denne måten kan man legge inn sperrer i HJA, slik at leger har alle rettigheter, men for eksempel syke- og hjelpepleier har andre rettigheter.

DBO

Slik systemet er i dag må det ligge en tom database på smartkortet før DBO kan opprette tabeller på kortet. Dette bør endres slik at DBO også oppretter selve databasen på kortet. Ellers vil det være naturlig at det er nettopp denne applikasjonen som blir utvidet til å omfatte administrasjon av journalarkivet som nevnt lengre opp. Det trengs en del ekstra funksjonalitet dersom systemet skal bli slik vi har foreslått over. Det er behov for funksjonalitet som håndterer brukere på systemet. Mulighet for å kunne laste dagens rapporter over i journalarkivet, samt mulighet for å sitte ved en terminal inne på hovedkontoret for å lese i pasientens journaler.

7.5 Tidsbruk og utviklingsmodell

Vi har vel ikke noe konkret tall på hvor mange timer som er lagt ned i prosjektet. Det er snakk om et betydelig antall timer. Å komme med et tall her vil være ren tipping, og det er like greit å la det være usagt. I forhold til ganttskjemaet føler vi at vi har kommet sånn nogenlunde i mål. En del oppgaver har tatt lenger tid en planlagt og en del har vært enklere. Fra midten av mars til og med første uka i april var vi i stor grad opptatt med obligatoriske innleveringsoppgaver og eksamenslesing. Vi tok en noe forkortet påskeferie, vel vitende om at vi var litt på etterskudd. Etter påske har vi hatt lite annet skolearbeid. Arbeidet med prosjektet har kommet høyere opp på prioriteringslista og det tok ikke mange dagene før vi følte oss à jour igjen.

Oppdragsgiver ønsket at vi skulle ta utgangspunkt i en inkrementell utviklingsmodell. Dette har passet oss bra siden modellen er godt planleggbar, noe som har hjulpet oss med å nå det overordnede målet om å bli ferdig til 23. mai. Vi føler at valg av utviklingsmodell har vært riktig for oss. Arbeidet vårt har helt klart bært preg av å være gjort etter denne modellen. Spesielt gjelder dette programmeringen, men også utviklingen av kravspek og design. Først skrev vi del 1 og 2 i kravspesifikasjonen. I første inkrement skrev vi detaljert kravspesifikasjon og design for DBO. Etterpå implementerte vi løsningen. Andre inkrement har bestått av detaljert kravspesifikasjon og design for HJA. Vi implemterte pasient og dagsrapport ”tabsheetene”. Tredje og siste inkrement ble benyttet til å implementere ferdig HJA og legge inn feilsjekking.

7.6 Gruppemedlemmenes egne vurderinger av prosjektet.

Kai Hallingstad

Jeg gledet meg til å begynne på hovedprosjektet. Endelig skulle vi få komme oss bort fra auditoriene og heller gjøre noe i praksis. At vi fikk den oppgava vi ønsket oss var ikke akkurat demotiverende. Vi skulle få jobbe med helt ny teknologi, teknologi som nå virkelig er på vei inn i samfunnet.

Som gruppe har samarbeidet fungert utmerket. Vi har også jobbet litt sammen med andre oppgaver. Noen dager har selvfølgelig vært enklere enn andre, men det er ingenting å si på samarbeidet.

I løpet av prosjektet har jeg virkelig skjønt hva vi skal med mange av fagene vi har hatt tidligere. Fag som kan ha virket kjedelig, men som i den store sammenhengen faktisk er morsomme å jobbe med.

Selve arbeidsprosessen har gått greit. Ikke alt har vært like motiverende, men vi har kommet gjennom alt sammen. Dagene vi fant ut av vi ikke kunne bruke verken PDA eller smartkortet, var ikke så veldig morsomme. Når jeg ser på resultatet har jeg ingen problemer med å si meg fornøyd med det vi har prestert. Selvfølgelig vil det være enkelte ting som kan forbedres, men et sted må vi sette sluttstrek.

Til slutt vil jeg benytte anledningen til å takke for godt samarbeid. Takk skal dere ha.

Frode Andersen

Dette prosjektet har vært meget lærerikt. Faglig sett har jeg lært meg masse nytt under hele prosjektperioden. Jeg har alltid måttet sette meg inn i noe nytt. Ny teknologi er fint det, bare det virker. Sosialt sett har jeg hatt det som plommen i egget. Ikke noe å utsette på mine gruppemedlemmer. Dette kan høres litt flott ut, men min oppfatning er nå slik. Vi har alle kommet godt overens, og kommer helt sikkert til å holde kontakten etter utdannelsen.

Når det gjelder prosjektoppgaven, har det vært oppturer og nedturer. Nedturen kom da vi fant ut at oppgaven vår ble redusert. Jeg ble skuffet da vi fant ut at vi ikke kunne bruke smartkort eller PDA. Det var jo mest derfor jeg hadde lyst å ta denne oppgaven. Oppturen kom etter dette, da så jeg at alle på gruppa prøvde å gjøre det beste ut av situasjonen. Og det klarte vi bra. Ser jeg på resultatet vårt, kan jeg fint si at vi har kommet i mål.

Prosjektoppgaver er noe som brukes ute i den store verden. Så dette har vært en bra erfaring å ta med seg videre inn i arbeidslivet.

Vil benytte anledningen til å takke alle som har hjulpet oss. Spesielt Anders og Kai som har gitt meg et artig og lærerikt semester. Takk.

Anders Helling

Å få være en del av dette prosjektet har vært utrolig utfordrende og meget lærerikt. Gruppen har fungert utmerket sammen og samarbeidet oss i mellom kan ikke sies annet enn å ha vært upåklagelig.

Det har vært mye jobbing, men det aller meste har jeg oppfattet som lystbetont. Alikevel var det en demotiverende nedtur da vi først oppdaget at vi ikke ville klare å få vår applikasjon over på PDA'n. Dette ble ikke bedre da vi oppdaget at smartkortet heller ikke var kompatibelt med Java. Dette er opplysninger jeg mener at ErgoSolutions burde hatt kunnskaper om fra før. For det første siden de har gitt denne oppgaven som en hovedprosjektsoppgave til HIG. Og siden det er de som til daglig jobber med smartkortteknologi, mener jeg at de burde sitte på nok spisskompetansen til at disse feilene ble oppdaget på et tidligere stadium.

Det har imidlertid vært veldig inspirerende å sette seg inn i et nytt programmeringsverktøy. Det var ingen på gruppa som tidligere hadde vært noe særlig bort i Jbuilder. Alle har fulgt fag hvor java kode har vært viktig, samt et fag om GUI og Delphi så disse to fagene har til sammen vært et godt grunnlag for å sette seg inn i Jbuilder.

Min totale vurdering av hele hovedprosjektet er en positiv opplevelse som jeg vet jeg vil komme til å dra nytte av når jeg nå skal ut i yrkeslivet. Jeg tror dette hovedprosjektet har gitt meg en myk overgang til fremtidig arbeid, samt verdifull kunnskap om nettopp det å jobbe i en prosjektgruppe.

8 KONKLUSJON

Vi skulle utvikle et system som skal lette arbeidet til omsorgssektoren. Denne sektoren er preget av mye skriftlig og muntlig informasjon. Vi så mulighetene for å gjøre arbeidsdagen deres enklere med omsorgskort.

Vi har laget en applikasjon som oppretter tabeller og fyller inn data i en database. Videre har vi også laget en applikasjon som brukes for å hente og oppdatere data i denne databasen. Løsningen vår kan fint brukes som en demonstrator ovenfor hjemmesykepleien. Det er i applikasjonen lagt vekt på at brukergrensesnittet skal være tilpasset en PDA, og brukervennligheten er prioritert.

På grunn av tekniske problemer som vi ikke rår over, har vi måttet utelate både PDA og smartkort i løsningen vår. Vi har tro på at det ikke er lenge før det blir utviklet teknologi som gjør det mulig å kjøre applikasjonen på en PDA. Det smartkortet vi skulle ha brukt har ikke støtte for java, og må byttes ut med en annen type dersom vår løsning skal benyttes. Disse problemene gjør at prosjektet ikke vil oppfylle kravspesifikasjonen fullstendig.

Hvis det skal lages et omsorgskort, kan vår løsning brukes som et hjelpemiddel i denne utviklingen.

Prosjektet har vært meget lærerikt. Vi har lært masse om ny teknologi, men også at ny teknologi ikke alltid er like enkel å bruke.

9 LITTERATURLISTE

[1] Borland. http://www.borland.com/techpubs/jbuilder/jbuilder5/
sist besøkt: 16/5/01

[2] Donaldson, Murray G(2001). *"Guide for the Production of PPs and STs, version 0.91"*

[3] Eckel, Bruce (1998). *Thinking in java.* Prentice Hall, New Jersey

[4] Gemplus(1999). *GemDB Administrative tool user guide. version 1.0*

[5] Gemplus(1999). *GemDB Overview. version 2.0*

[6] Gemplus(1999). *GemDB Reference manual*

[7] Jesnes, Brenden og Kucukoglu (2000). *Varekataloghandel med smartkort.*HIG

[8] Larman, Craig (1998). *Applying UML and patterns.* Prentice Hall

[9] Sosial og helsedepartementent (2001). *Elektronisk samhandling i helsesektoren "Si @!".*

[10] Sosial og helsedepartementet (1997). *Elektroniske pasientkort i norsk helsetjeneste.*

10 VEDLEGG