

Syddansk Universitet



Det Tekniske Fakultet / MMMI

Civilingeniøruddannelsen i Software Engineering

Diplomingeniøruddannelsen i Softwareteknologi

SI2-PRO Organisationsorienteret Softwareudvikling

Sensum Udred

SI2-PRO: Semesterprojekt F18

Semesterkoordinator: Lone Borgersen - lobo@mmmi.sdu.dk

Vejleder: Jan Emil Larsen - jel@lit.dk

Projektgruppe 2

Navn	Email
Sigurd Eugen Espersen	siesp17@student.sdu.dk
Josed Ngoc Can Pham	jopha15@student.sdu.dk
Loc Hoang Thanh Nguyen	longu17@student.sdu.dk
Anders Bensen Ottsen	anott17@student.sdu.dk
Peter Stærdahl Andersen	pande15@student.sdu.dk
Erik Bjørnholt Nielsen	ernie17@student.sdu.dk

Projektforløb: 01/02/2018 til 31/05/2018

Afleveringsfrist: Torsdag d. 31. Maj 2018

Projekt: [Sensum Udred]

[Projektcase](#), [Elaborationsoplæg](#), [Checkliste for projektaflevering](#)

Abstract

This paper examines the design, implementation and the overall thoughts behind the project: Sensum Udred. The authors and developers of this project were second semester students, studying Software Technology and Software Engineering at University of Southern Denmark. These students were given a case about a possible system called Sensum Udred made by EG Team Online. The system would focus on disabled and vulnerable adults and how their cases would be managed digitally. In this project the students set the system boundaries to primarily focus on making new cases in their system.

The paper documents, the work and effort of group 2, was put into the project which focused on organizational software development. The project followed the development method Unified Process, while the planning process was supplemented with Scrum, thus starting with an Inception Phase. In the inception phase, information was gathered about the corporation and the business requirements. A business case was made about the company EG Team Online, examining whether developing the system Sensum Udred would be profitable. The elaboration phase consisted mainly of modelling the different use cases, realizations by interaction diagrams, class diagrams and implementing these through the use of Java.

During the elaboration phase the students worked in two different iterations, where the first one focused mainly on creating a solid architecture, and developing the domain layer of the system. Another requirement in the first iteration was to create a single class in the presentation layer and it was decided to design a text based user interface in addition to a persistence layer to store data in textfiles. The second iteration consisted of making a graphical user interface and implementing a SQL database as well as adding additional functionality to the overall system.

Forord

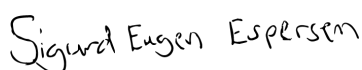
Dette projekt på 2. semester indeholder følgende fagligheder Organisation og Ledelse (OL), Grundlæggende Software Engineering (GSE), Videregående Objektorienteret Programmering (VOP), Databasesdesign og -programmering (DB), der tilsammen bidrager til semesterprojektet.

Temaet for semesterprojektet er Objektorienteret Softwareudvikling og tilsigter at projektgruppen både undersøger samt fremstiller en kørende udgave af et softwaresystem til en virksomhed. Systemet skal opfylde virksomhedens stillede krav samt overholde de grundlæggende principper for godt design. Systemet gennemføres i en systematisk udvikling og i en anvendelsesorienteret, iterativ og objektorienteret proces.

Denne rapport er udarbejdet af alle 6 gruppemedlemmer fra projektgruppe 2, og dokumenterer hele gruppens projektforsløb over de 5 hovedfaser: Idefasen, Inceptionsfasen, Elaborationsfasen, Færdiggørelsesfasen samt Evaluerings- og Refleksionsfasen. Rapporten beskriver projektgruppens arbejdsproces i de forskellige faser, hvor der arbejdes med letvægtsudgaven af Unified Process kombineret med Scrum.

Projektet er udført i perioden fra d. 01.02.2018 til d. 31.05.2018 på Syddansk Universitet i Odense under Mærsk Mc-Kinney Møller Institutet på Det Tekniske Fakultet under vejledning af Jan Emil Larsen og semesterkoordinator Lone Borgersen.

Projektdeltagerne anerkender deres aktive deltagelse i projektforsløbet gensidigt ved forfatternes underskrifter nedenfor.



Sigurd Espersen



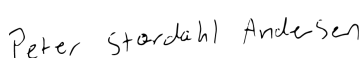
Josed Pham



Loc Nguyen



Anders Ottsen



Peter Andersen



Erik Nielsen

Indholdsfortegnelse

Abstract	2
Forord	3
Læsevejledning	6
Systemguide	6
Ordliste	7
Redaktionelt	7
1 Indledning	8
1.1 Problemstilling	8
1.2 Formål og mål	9
1.3 Motivation	10
1.4 Målgruppe	10
2 Metoder, planer og forløb	11
2.1 Metoder	11
2.2 Planlægning	12
2.3 Projektforløb	13
2.4 Første Iteration vs. Anden Iteration	14
3 Elaboration	15
3.1 Overordnet kravspecifikation	15
3.2 Detaljeret kravspecifikation	18
3.3 Analyse	20
3.3.1 Den statiske side af analysemodellen	20
3.3.2 Den dynamiske side af analysemodellen	21
3.4 Design	24
3.4.1 Softwarearkitektur	24
3.4.2 Design af persistenslaget	25
3.4.3 Design af domænelaget	29
3.4.4 Design af præsenteringslaget	30
3.5 Implementering	34
3.5.1 Implementeringen af centrale dele i persistenslaget	34
3.5.2 Implementeringen af centrale dele i domænelaget	38
3.5.3 Implementeringen af centrale dele i præsenteringslaget	40
3.6 Test	44
3.7 Konklusion på elaboration	44
4 Evaluering	45
5 Konklusion	47

6 Litteraturliste	48
B: Kildekode	50
C: Projektforslag	51
D: Inceptionsdokument	57
E: Projektlog	76
F Interne Bilag	77
Bilag F1: Scrum Release Backlog	77
Bilag F2: Scrum Product Backlog	78
Bilag F3: Scrum Sprint Backlog	79
Bilag F4: Scrum Burndown Chart	81
Bilag F5: Domænemodel	82
Bilag F6: Analyseklassediagram	83
Bilag F7: Sekvensdiagrammer	84
Bilag F8: Designklassediagram	88
Bilag F9: Arkitektur Diagram	89
Bilag F10: 3NF E/R Model	90
G Eksterne Bilag	91

Læsevejledning

Hovedrapporten er opbygget med en prolog, en krop og en epilog, der tilsammen udgør afsnit A. Herefter findes afsnit B-G med de specifikt angivne afsnit, i den udleverede checkliste til projektaflevering.

Rapportens enkelte sektioner og afsnit er udarbejdet, så det er let for læseren at læse disse uafhængigt af hinanden, eksempelvis med en lille indledning ved hvert afsnit. Dog er afsnittene opstillet kronologisk, og der opnås derved bedst mulig forståelse ved at følge rapporten fra start til slut.

Det bør nævnes at det meste af arbejdet med de første trin i projektforløbet såsom projektetablering og inceptionsfasen er vedlagt i rapporten i afsnit C og D, og der vil undervejs i rapporten blive henvist til disse afsnit. Ligeledes er det en fordel for læseren, at have kendskab til de tidligere faser i projektforløbet, for derved at opnå bedre forståelse i rapporten.

Afsnittet omkring *Elaboration* skiller sig ud, da afsnittet går i dybden med eventuelle implementeringer. Disse kodeafsnit samt beskrivelser vil indblande engelske termer, da det er valgt at programmere kode på engelsk. Hertil har gruppen valgt at medtage et afsnit med en [Ordliste](#)(s. 7), der burde fremme sammenhængen mellem de termer oversat til engelsk.

Systemguide

Dette afsnit præsenterer kort hvorledes projektet importeres og køres:

- Importer projektgruppens .zip mappe til NetBeans IDE
- Til at starte med skal man 'Clean and Build' projektet. Dette kan gøres ved enten at højreklikke på projektet 'Sem02_SemesterProjekt_SensumUdred' og trykke 'Clean and Build' eller ved hjælp af genvejen **SHIFT + F11**
- Herefter skal man lokalisere klassen Main.java i Starter-pakken inde under Source Packages
- Main klassen startes ved genvejen **SHIFT + F6** eller ved at højre-klikke inde i selve koden og trykke 'Run File'
- Herefter opstartes programmet, hvor systemets login-side vises
- Mulige login (Brugernavn/Password): admin/admin, socialworker/socialworker, secretary/secretary
- Disse forskellige login, giver forskellige rettigheder i systemet

Gruppen har brugt github til at holde styr på koden, da flere personer har skullet arbejde på samme tid. Gruppens git til projektet kan findes på nedenstående link:

<https://github.com/AndersBensen/2-semesterprojekt-sensum-udred>

Ordliste

Dette afsnit er tildelt specifikke termer og ord brugt i rapporten og/eller de første faser i projektudviklingen som senere er blevet oversat til engelsk i elaborationsfasen hvor der implementeres relevante modeller til kode.

Dansk	Engelsk	Beskrivelse
Borger	Person (citizen)	En borger er en person som indgår i CPR registeret.
Sag	Case	En sag er helheden af oplysninger som knytter sig til en henvendelse, som starter et forløb.
Henvendelse	Case request	En henvendelse er en forespørgsel fra en borger om at få løst et problem.
Ansæt	Employee	En ansat har adgang til systemet med forskellige rettigheder.
Ansæt (arbejder med henvendelser)	Case employee	En ansat med rettigheder til at arbejde med henvendelser (fx sekretær)
Sagsbehandler	Social worker	En ansat med rettigheder til at oprette henvendelser og sager.
Sekretær	Secretary	Sekretæren er en ansat som arbejder med henvendelser.

Redaktionelt

Dette afsnit beskriver skriveprocessen og ansvarsområder i produkt og projekt.

Alle projektdeltagere deler den fælles holdning, at hver enkelt deltager bidrager ligeligt til produktet og står til ansvar for hele projektet. Derfor har det været et internt krav fra start, at alle medlemmer tager del i hele skriveprocessen samt produktudviklingen. Alle gruppens medlemmer har derfor været indover alle arbejdsområder inden for projektet og dermed deltaget ligeligt. Årsagen til denne tilgang er, at gruppen mener, der vil opnås højere fagligt niveau, hvis alle deltagere individuelt har forståelse for alle projektets elementer.

1 Indledning

Dette projekt omhandler udviklingen af et nyt sagsbehandlingssystem kaldet Sensum Udred, som er tiltænkt handicappede og udsatte voksne. Projektet afvikles med udgangspunkt i en case stillet af virksomheden EG Team Online, hvor det forventede resultat bliver et produkt i form af et funktionelt kørende system, samt en tilhørende komplet dokumentation over både projektudvikling og det færdige produkt. Sagsbehandlingssystemet skal udvikles som et DHUV-system¹, hvilket står for Digitalisering af Handicap og Udsatte Voksne.

Udfordringen i den stillede case er at danne sig et overblik og få organiseret information og data omkring sagsbehandling på en nem og overskuelig måde. Sensum Udred skal agere som et sagsbehandlingssystem, der netop skaber dette overblik over selve sagsbehandlingen og facilitere kommunens arbejdsprocesser. Systemet skal understøtte kommunikationen mellem den enkelte borger og kommunen - hvor det potentielt udvikles til en sag og videre til en udredning af borgeren i form af en behandling. For DHUV er denne proces beskrevet i sagsmodellen for voksenudredningsmetoden (VUM)².

Voksenudredningsmetoden består af følgende aktiviteter i sagsforløbet: sagsåbning, sagsoplysning, sagsvurdering, afgørelse og valg af social indsats og sagsopfølgning.

Projektets afgrænsning er bestemt tilbage i inceptionsfasen - hvor dette projekt specifikt er afgrænset til at omhandle de første trin af VUM. Projektet vil derfor konkret omhandle udviklingen af et system over sagsåbning.

1.1 Problemstilling

Den udleverede projektcase præsenterer problemstillingen, at der er sket et monopolbrud i den offentlige sektor omkring IT-løsninger. Af denne årsag bevæger virksomheden EG Team Online sig ud på et nyt marked, hvor de bl.a. vil forbedre kommunikation og eksisterende arbejdsprocesser indenfor det socialpædagogiske område. Ligeledes skal der tages stilling til kommunernes høje krav omkring datasikkerhed, især fordi der for nyligt er blevet vedtaget en ny lov om databeskyttelsesforordning i EU. Dette medfører at den nye persondataforordning har høj prioritet i det nye system, og derfor fokuseres der på hvem, hvad og hvornår eventuelle følsomme data tilgås i systemet.

Tilbage i projektetablering blev det præsenteret, at projektet ville undersøge, hvorvidt det kunne betale sig for EG Team Online at lave et sagsbehandlingssystem.

¹ Digitalisering af handicap og udsatte voksne:

<https://socialstyrelsen.dk/tvaergaende-omrader/sagsbehandling/voksenudredningsmetoden>

² Voksenudredningsmetoden, afsnit 1.4, side 6:

<https://socialstyrelsen.dk/filer/tvaergaende/sagsbehandling-og-organisering/link-metodehandbog-vum-1.pdf>

Problemformulering

Med udgangspunkt i problemformuleringen fra projektforslaget³ vil der blive set på, hvordan systemet kan blive bygget op med fokus på sagsåbning, sikkerhed og brugervenlighed, ved at se på følgende spørgsmål:

- Kan systemet bruges til at oprette og administrere henvendelser og sager på en hensigtsmæssig måde?
- Hvad skal der gøres for, at brugeren får et naturligt flow i sin arbejdsgang i det digitaliserede system?
- Er det muligt at opretholde sikkerheden, så de ansatte kun har adgang til deres eget arbejdsområde, samt overholde kravene fra persondataforordningen?

1.2 Formål og mål

Formålet med dette projekt er at anvende de forskellige færdigheder fra semesterets faglige kurser og kombinere disse til at udarbejde et kørende softwaresystem. Formålet er ligeledes at projektet skal give større indsigt i videregående programmering, databasesystemer og softwareudvikling i en organisatorisk kontekst. Dette opnås bl.a. ved systematisk udvikling i en anvendelsesorienteret, iterativ og objektorienteret proces. Endvidere er formålet at udvikle produktet i samarbejde med virksomheden EG Team Online, hvor et af målene er, at produktet lever op til de stillede specifikationer og krav fra EG Team Online. Til slut er målet at danne et solidt beslutningsgrundlag for, hvorvidt det kan betale sig for EG Team Online at enten igangsætte og/eller fortsætte det videre arbejde med udviklingen af sagsbehandlingssystemet Sensum Udred. De overordnede kompetencemål⁴ for semesteret er opstillet nedenfor:

- Analysere og evaluere en organisations strukturelle og sociale elementer.
- Tilrettelægge og gennemføre et projekt som er brugsmønsterstyret, iterativt og objektorienteret baseret på letvægtsudgaven af Unified Process.
- Programmere et softwareprodukt på grundlag af en designmodel og implementeringsmodel.
- Designe og programmere en relationel database og benytte den i organisatorisk kontekst.
- Planlægge og organisere projekter og reflekterer over deres faktiske gennemførelse.

³ Problemformulering fra Projektforslaget er blevet indført, [Bilag C](#) s. 51

⁴ Kompetencemål beskrevet i Projekthåndbog

1.3 Motivation

Projektgruppen består af både studerende på Software Engineering og Softwareteknologi - og deler derfor den fælles interesse omkring software. Der forekommer derfor naturligt en motivation hos deltagerne, der indebærer en trang til læring og udvikling. Endvidere deler gruppen en interesse for at arbejde med et projekt, som afspejler en virkelighedsnær problemstilling.

Først og fremmest er visionen hos projektgruppen at levere et semesterprojekt, der har et højt fagligt niveau og repræsenterer et professionelt projekt. Desuden vil gruppen lære at arbejde struktureret og inkrementelt med Unified Process og Scrum - der resultere i en kørende udgave af systemet med tilhørende database.

Projektgruppen ønsker at opnå større forståelse og bedre indsigt i videregående programmering og databasesystemer samt softwares betydning for virksomheder og den gensidige vekselvirkning mellem software og organisation.

1.4 Målgruppe

Projektets målgruppe er virksomheden EG Team Online, mens produktet, sagsbehandlingssystemet, er tiltænkt kommunerne. Efter projektets slutning, vil det være op til EG Team Online at vurdere og tage stilling til om projektet har overholdt og opfyldt de stillede kriterier og behov hos virksomheden. Projektet er derfor tiltænkt de personer i ledelsen hos EG Team Online, samt relevante udviklere og arkitekter - der skal sidde med beslutningen om, hvorvidt der er nok beslutningsgrundlag for at projektet skal igangsættes og videreudvikles.

2 Metoder, planer og forløb

Det følgende afsnit omhandler hvilke beslutninger, der er blevet taget omkring udviklingsprocesser og arbejdsmetoder. I afsnittet præsenteres de metoder, der er blevet arbejdet med i de sidste faser og begrundelser for valg af disse processer. Der vil også være en specifikation over planlægningen af hele forløbet, samt en refleksion over hvilket udbytte, projektet har fået ved disse valg.

2.1 Metoder

I dette projekt er der blevet anvendt en letvægtsudgave af den objektorienteret udviklingsmetode, Unified Process(UP)⁵. Letvægtsudgaven af UP for dette projekt, bestod i at arbejde med inceptions- og elaborationsfasen. UP er en ramme for de væsentligste aspekter inden for udvikling af software systemer. Processen berører mange aspekter inden for udvikling af software og er derfor meget omfattende. Ud over det, er den faseopdelt hvori hver fase består af forskellige fokusområder. Disse faser definerer iterationer for hhv. krav, analyse, design, implementering og test. Dog vil der i dette projekt kun være fokus på inception og elaboration, som bliver beskrevet i UP.

UP bliver typisk brugt til projekter, der har med store offentlige systemer at gøre, som også er forretningskritiske. Analyse og dokumentation fylder desuden en stor del af UP. Eftersom tanken bag dette projekt bygger på et stort offentligt system, som samtidig er forretningskritisk, er UP en god udviklingsmetode. Artefakterne, som bliver udarbejdet ved brug af metoden, er med til at dokumenterer analyse og krav, der prioriteres højt for offentlige og andre risiko-styrede systemer.

UP er meget brugercentreret, størstedelen af artefakterne indeholder derfor udarbejdede kravspecifikationer, som afspejler de vigtigste brugsmønstre for systemet. En tilføjelse til UP, som projektgruppen har valgt at benytte sig af, er *Pair Programming* fra Extreme Programming⁶. Dette har projektgruppen valgt at gøre på baggrund af god erfaring med par programmering, når et team består af individer med forskellig erfaringsniveauer.

I forlængelse med udviklingsmetoden UP, er der blevet anvendt Scrum til at strukturere selve arbejdsprocessen⁷. Scrum er en arbejdstilgang, som fokuserer på en agil udviklingsprocess, og dette er gjort ved at arbejde med en iterativ udvikling af produktet, i form af trinvis iterationer kaldet sprints. Udbyttet efter hvert sprint skulle gerne svare til en kørende udgave eller delprodukt ud fra de elementer som er blevet beskrevet i backlog.

⁵ (Larman, 2005, kap. 2): [Iterative, Evolutionary, and Agile](#)

⁶ Sommerville, 2016, s.77

⁷ Scrum guide: <http://www.scrumguides.org>

I dette projekt er der anvendt Scrum-buts, da alle metoderne, som hører til under Scrum, ikke benyttes. F.eks. mødes gruppen ikke hver dag, og det er svært at have en fuldgældig Product Owner, eftersom ingen af gruppens medlemmer står som ejer af produktet eller har direkte kontakt til virksomheden. Dog blev der valgt at anvende en Scrum Master, der har sørget for at hjælpe teamet og samtidig har ført scrum ceremonier såsom sprints, daglige scrum møder og sprint reviews. Gruppens Scrum Master har dog ikke ageret under de originale regler, da rollen som Scrum Master har skiftet på ugebasis mellem gruppen, udover at personen også har været med til at udvikle. I forhold til udviklingsprocessen med den estimerede tid, har der været for meget belastning i produkt backloggen. I det traditionelle scrum, fjerner man elementer fra produkt backloggen for at kunne afvikle sprint målet. I projektet er der dog valgt ikke at følge denne version af scrum, og i stedet valgt at sætte arbejdshastigheden op for at nå sprint målene.

Herunder anvendes Moscows prioriteringsmetode⁸ (Must have, should have, could have, won't have), som definerer en velformuleret produkt backlog på baggrund af de kundemæssige værdier. Målet med scrum er, at der efter hver enkelt iteration står et fungerende delprodukt af det samlede system, som kan fremvises til kunden.

Både Scrum og UP er med til at styrke arbejdet i projektet grundet omfattende dokumentation og sammenhold mellem opgaver. UP har givet værktøjer og retningslinjer til at designe og tegne diagrammer og modeller, som kunne vises til resten af gruppens medlemmer mens Scrum har dikteret bl.a. daglige møder og sprint reviews, hvor det er muligt at samle op på forskellige opgaver og hjælpe hinanden på tværs af de mindre parprogrammeringsgrupper.

Dog kan det nævnes at UP har meget dokumentation og noget af det, kan virke overflødig i dette projekt, da det kun er en del af et større system. Artefakter fra UP såsom domænemodellen og analyseklassediagrammet, som bliver brugt til kommunikation med interessenter, kunden og arkitekter er blevet nedprioriteret, efter gruppen er gået videre til næste fase, da der ikke skulle holdes kontakt med tidligere nævnte personer. Det virkede også omstændigt at have en produktejer, som Scrum foreskriver, at der bør være, eftersom ingen af gruppens medlemmer kender ekstra meget til systemet. I stedet er der blevet brugt mere tid på analyse af systemet, så flest mulige udfordringer og muligheder kunne blive afdækket.

2.2 Planlægning

I forbindelse med Unified Process, er der blevet planlagt at arbejde i to iterationer i elaborationsfasen. Disse to iterationer anses som projektgruppens sprints, fra Scrum, og disse forløber hhv. med start d. 6/4 og d. 4/5 og har begge en varighed på ca. fire til fem

⁸ MoSCoW prioriteringsmetode: https://en.wikipedia.org/wiki/MoSCoW_method

uger. Hver uge har gruppen valgt at udnævne en ny Scrum Master, hvor denne samtidig har den fælles rolle med resten af gruppen som udvikler.

I hvert af de to sprints har det været essentielt at sætte tid af til at planlægge opstarten af selve sprintet - bl.a. planlægning og beskrivelse af indholdet i gruppens Product Backlog samt en tidsestimering af hver enkelt arbejdsopgave. Denne planlægning udføres af hele projektgruppen, hvorefter selve arbejdet påbegyndes, og gruppen arbejder med par programmering. Ligeledes har det også været vigtigt at sætte tid af til at afslutte sprintet og udføre review og retrospektiv. Her samles projektgruppen og der skabes et fælles overblik over systemet samt opsamling af sprint forløbet med væsentlige hændelser i form af udarbejdelse af produktet.

Herudover har der sideløbende med projektet været ugentlige møder med gruppens projektvejleder. På disse møder er udfordringer og forslag blevet fremlagt og vurderet, og med feedback fra vejlederen har gruppen fortsat sit arbejde.

2.3 Projektforløb

Selve forløbene i de to iterationer begynder med en planlægning af sprintet. Her vil der først blive valgt en scrum master blandt gruppemedlemmerne, og denne rolle bliver tildelt et nyt medlem hver uge. Scrum master har ansvar for både at starte og afslutte sprintet samt at holde alle deltagere ajour i selve scrum processen. Herefter arbejder projektgruppen med de elementer, som er beskrevet i den nuværende release backlog⁹. Ligeledes mødes gruppen næsten dagligt og udfører dermed den daglige scrum. Til sprintets afslutning samles gruppen, og der bliver lavet fælles review og retrospektiv.

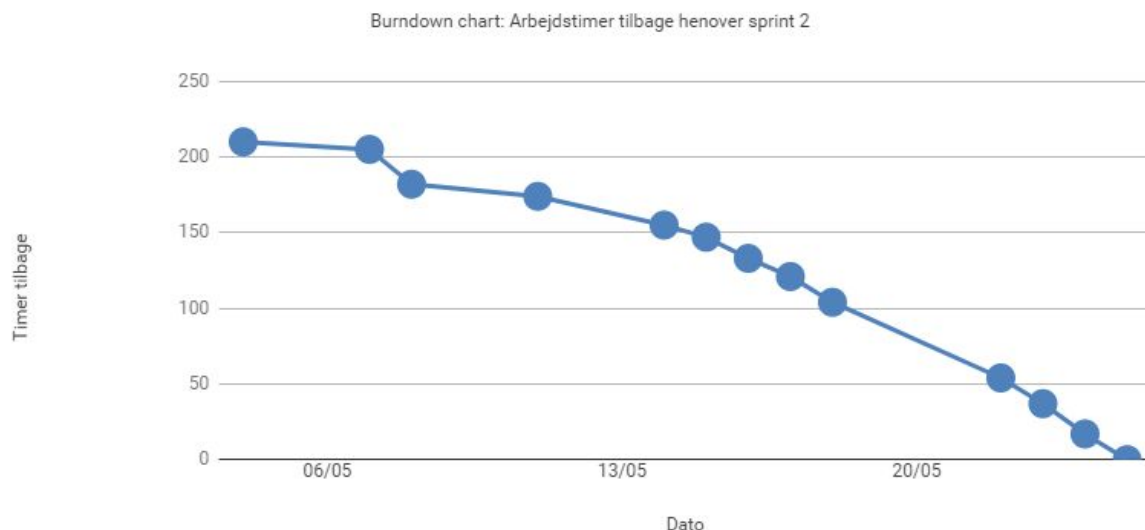
Mere specifikt vil hele arbejdsprocessen starte i gruppens Produkt backlog¹⁰, hvor alle væsentlige arbejdsopgaver er beskrevet med deres kategori, status og prioritering. Derefter føres disse opgaver over i Release Backlog, hvor gruppen forsøger at estimere den tid, det ville tage, at udføre opgaverne. Efter tidsestimeringen af opgaverne, bliver de overført til et sprint backlog¹¹ og sprintet påbegyndes af scrum master. Projektdeltagerne vil løbende i sprintet skære ned på arbejdstiden på de specifikke opgaver - herved bliver der udarbejdet et burndown chart¹² over sprintet (se figur 2.3.1 for chart over andet sprint) - og samtidigt opdateres opgavernes status. Statussen rettes således i selve sprint backlog, release backlog og til slut i selve product backlog - eksempelvis fra en igangværende status til færdig.

⁹ [Bilag F1: Scrum Release Backlog](#), side 77

¹⁰ [Bilag F2: Scrum Product Backlog](#), side 78

¹¹ [Bilag F3: Scrum Sprint Backlog](#), side 79-80

¹² [Bilag F4: Scrum Burndown Chart](#), side 81



Figur 2.3.1: Denne figur præsenterer et burndown chart over andet sprint.

2.4 Første Iteration vs. Anden Iteration

I første iteration var formålet at danne et solidt fundament til et løsningsforslag, der kunne anvendes i den videre udvikling i elaborationsfasen, som er den anden iteration. Første iteration havde stort fokus på domænelaget, hvor mest mulig logik og funktionalitet blev udarbejdet. Dette skulle danne et grundlag for den videre udvikling i projektet, da man derved fik opnået en stabil struktur. Derudover blev der valgt at arbejde med 3-lags-arkitekturen, hvor persistenslaget i første iteration blev bygget op som et system, der kunne gemme objekterne i tekstfiler, og interaktion med brugeren blev opnået med en tekstbaseret brugergrænseflade. Den væsentligste forskel mellem iterationerne lå i præsentation- og persistenslaget.

Den anden iteration byggede ovenpå alt arbejdet fra forrige iteration, og der blev fokuseret på at opnå persistens i form af en relationel database, samt interaktion med brugeren via en grafisk brugergrænseflade. Domænelaget havde meget funktionalitet klar til anden iteration, men det skulle modificeres, så den nye funktionalitet fra anden iteration kunne integreres med præsentation- og persistenslaget. Begge iterationer nåede i mål, som ses illustreret i gruppens burndown chart, se [Bilag F4: Scrum Burndown Chart](#) (side 81) for burndown chart for første iteration og burndown chart for anden iteration foroven (figur 2.3.1).

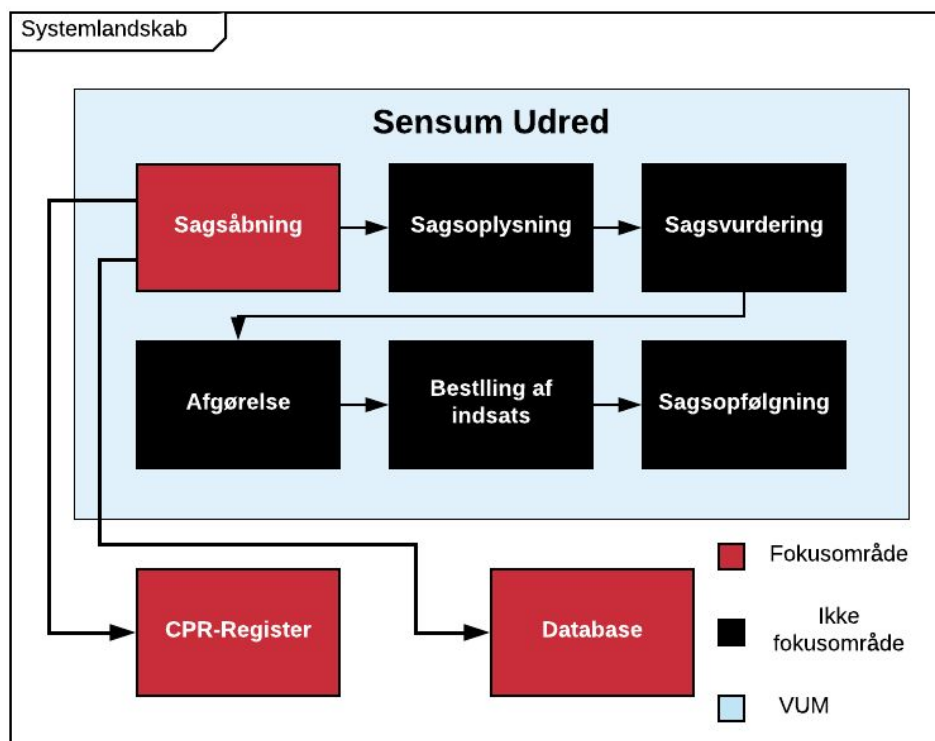
3 Elaboration

Dette afsnit præsenterer elaborationsfasen, hvori der sker en iterativ udvikling gennemført i løbet to iterationer. Arbejdet i denne fase vil få en nærmere beskrevet gennemgang af følgende trin: krav, analyse, design, implementering og test.

3.1 Overordnet kravspecifikation

Den overordnede kravspecifikation omfatter en beskrivelse af systemet og den foretagne afgrænsning (fra afsnit [D: Inceptionsdokument](#), side 62) illustreret i et opdateret uddrag af systemafgrænsningen. Derudover vil kravene til systemet blive nærmere beskrevet og hvorledes disse opfyldes i den overordnede brugsmønstermodel.

Nedenfor præsenteres det endelige systemlandskab (figur 3.1.1). Den blå boks illustrerer hele systemet som helhed med Voksenudredningsmetoden (VUM). De røde felter herunder er det afgrænsede problemdomæne, som projektet befinder sig i. De sorte felter beskriver det miljø, som ville være til stede for hele systemet, men som er fravalgt i projektet. Det afgrænsede problemdomæne omhandler derfor de første trin i VUM, som er sagsåbningen.

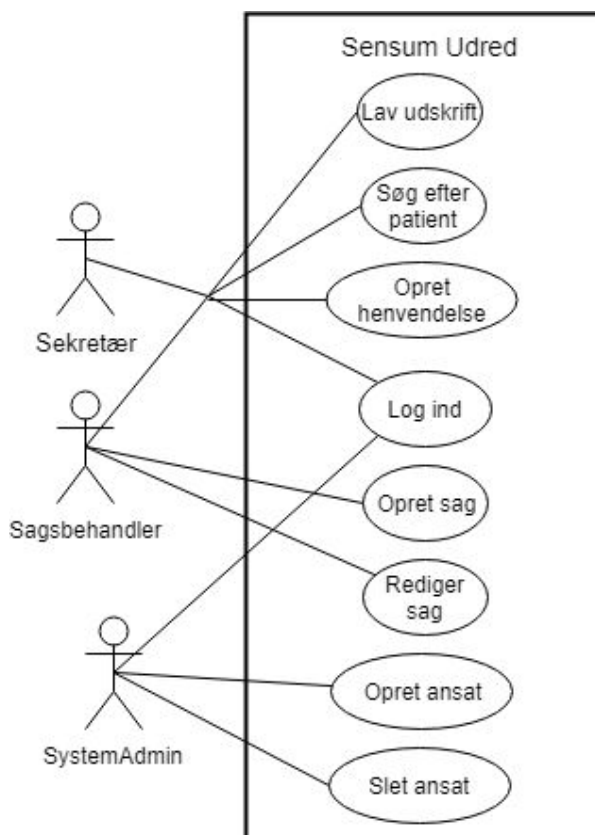


Figur 3.1.1: Denne figur præsenterer gruppens systemlandskab.

Derudover vil der også være en kravspecifikation af systemet udarbejdet i form af en overordnet brugsmønstermodel, som ses på næste side (figur 3.1.2). Brugsmønstermodellen erstatter den traditionelle funktionelle specifikation af krav - så der også vises hvad systemet skal gøre for hver enkelte bruger. Der ses i modellen tre primære aktører, som alle kommer til at interagere med systemet. Måden der interageres på beskrives i brugsmønstrene. De tre

aktører har forskellige rettigheder, eksempelvis er det kun administratoren, som kan oprette og slette ansatte. Det ses også at sagsbehandleren har de samme rettigheder som sekretæren, men med udvidet funktionalitet, således at denne rolle også kan oprette og redigere i sager.

Aktørerne vil blive nærmere beskrevet i tabellen med aktørbeskrivelserne (tabel 3.1.1), og det samme vil de valgte brugsmønstre, der bliver beskrevet i tabellen med brugsmønsterbeskrivelserne (tabel 3.1.2).



Figur 3.1.2: Denne figur præsenterer den overordnede brugsmøstermodel.

Nedenfor beskrives de valgte aktører og deres eventuelle funktion.

Navn	Beskrivelse	Mål og tjenester
Sekretær	I større kommuner vil der være sekretærer ansat, som er involveret i de første simple trin af sagsbehandlingen. Sekretæren kan tage mod en henvendelse, og starte en sag. Dog kan sekretæren aldrig gå hele vejen med sagen, men blot lave forarbejdet til en sagsbehandler.	Hjælpe med første trin af sagsbehandling - evt. henvendelse

Sagsbehandler	En sagsbehandler er den person som vil tage sig af en persons forløb i en sag. Det vil som regel være en sagsbehandler der starter en sag (udover i større kommuner, hvor en sekretær vil starte) og så følge den hele vejen til dørs. Det er primært sagsbehandleren som vil sidde med systemet til dagligt, og holde styr på de forskellige sager.	At hjælpe en patient gennem en sagsbehandling
SystemAdmin	Admin er den person der er administrator for systemet. Administratoren vil have flere rettigheder i systemet end de andre ansatte. Administratoren vil være den første person der er oprettet i systemet, og vil stå for at oprette andre brugere i systemet og eventuelt slette disse brugere.	Oprette/slette/redigere ansatte i systemet

Tabel 3.1.1: Denne tabel præsenterer den overordnede beskrivelse af de forskellige aktører.

Nedenfor beskrives de valgte brugsmønstre og deres prioritering.

Nr.	Brugsmønster	Formål	Prioritering
BM1	Lav udskrift	En patient kan få en udskrift af sin sag, så patienten kan se alle de informationer der er i sin sag.	Could have / Won't have
BM2	Søg efter patient	En ansat kan søge efter de patienter der er i systemet, så den ansatte hurtigt kan tilgå og få et overblik over patienter og se de sager der er knyttet til de tilsvarende patienter.	Should have
BM3	Opret henvendelse	En ansat kan oprette en henvendelse, hvilket er første trin før der oprettes en sag. Her bedømmes om de givne informationer er nok til at gå videre til en sag.	Must have
BM4	Opret sag	En ansat kan oprette en sag, så der elektronisk bliver holdt styr på de informationer der vedrører en sag.	Must have
BM5	Rediger sag	Sager kan blive redigeret så informationer i sagen kan ændres til noget nyt.	Must have
BM6	Log ind	Logger ind i systemet med unikt brugernavn og kodeord. Dette giver adgang til systemets funktioner og data.	Must have
BM7	Opret ansat	En ansat bliver oprettet af en systemadministrator. Hertil bliver informationer oprettet til den ansatte, såsom	Must have

		kontaktinformationer.	
BM8	Slet ansat	En ansat bliver fjernet fra systemet. Dette sikrer at data på en tidligere ansat bliver fjernet fra systemet, således overflødig information bliver fjernet, samt at den tidligere ansatte ikke længere har tilgang til systemet.	Could have

Tabel 3.1.2: Denne tabel præsenterer den overordnede brugsmønsterbeskrivelse.

3.2 Detaljeret kravspecifikation

Til den detaljerede kravspecifikation er der ikke valgt at lave en detaljering af brugsmønstermodellen. Der er valgt at udarbejde detaljerede brugsmønsterbeskrivelser over brugsmønstrene "Opret sag" og "Opret henvendelse" - da disse to spiller en helt central rolle i sagsåbningen. Der er hertil anvendt modellen på side 78, figur 4.8 (Arlow, 2005) til at detaljere brugsmønstrene nedenfor (Tabel 3.2.1 & 3.2.2).

Brugsmønster: Opret henvendelse
ID: BM3
Primære aktører: Sekretær, Sagsbehandler
Sekundære aktører: Ingen
Kort beskrivelse: En ansat i form af enten en sekretær eller sagsbehandler bliver kontaktet af en borger, og hvis der er grundlag for dette, vil der oprettes en henvendelse i systemet.
Prækonditioner: Der er logget en sagsbehandler eller sekretær ind.
Hovedhændelsesforløb: <ul style="list-style-type: none"> - Brugsmønstret starter ved at en borger kontakter kommunen. - Enten en sekretær eller en sagsbehandler, vil i dialog med borgeren, indtaste de nødvendige informationer i en elektronisk henvendelse i systemet. - Hvis der er nok grundlag for dette - vil en henvendelse blive oprettet i systemet og systemet vil logge denne handling. - Henvendelsen får tildelt et ID i system, og den specifikke borger bliver tilføjet henvendelsen.
Postkonditioner: En henvendelse med korrekt ID og tilsvarende knyttet borger er oprettet i systemet.
Alternative hændelsesforløb: Hvis der ikke er grundlag nok, eksempelvis manglende informationer, vil der ikke blive oprettet en henvendelse i systemet.

Tabel 3.2.1: Brugsmønster BM3 Opret Henvendelse

Brugsmønster: Opret sag
ID: BM4
Primære aktører: Sagsbehandler
Sekundære aktører: Ingen
Kort beskrivelse: En ansat i form af en sagsbehandler behandler en elektronisk henvendelse i systemet, hvorpå der vurderes - og hvis der er grundlag for dette - vil en sag blive oprettet i systemet.
Prækonditioner: Der er oprettet mindst en konkret henvendelse. Brugeren er logget ind som sagsbehandler.
Hovedhændelsesforløb: <ul style="list-style-type: none">- Brugsmønstret starter ved at en sagsbehandler vælger at behandle en henvendelse i systemet.- Hvis der er grundlag for dette - vil sagsbehandleren oprette en ny sag, hvortil den tilsvarende henvendelse bliver knyttet hertil.- Sagsbehandleren vil indtaste de nødvendige oplysninger i sagen, og til slut oprette denne i systemet med korrekt ID. Denne handling bliver ligeledes logget i systemet.
Postkonditioner: En sag med korrekt ID og tilknyttet borger er oprettet i systemet.
Alternative hændelsesforløb: Hvis borgerens tilstand eller forhold ændres, og der ikke længere er grundlag for det, vil en sag ikke blive oprettet i systemet.

Tabel 3.2.2: Brugsmønster BM4 Opret Sag

Den detaljerede kravspecifikation vil bl.a. også omfatte en beskrivelse af de supplerende krav og disse bliver organiseret efter FURPS+ (Functionality, usability, reliability, performance, supportability & constraints)¹³. Eftersom dette afsnit allerede er blevet beskrevet i inceptionsfasen - henvises der til [Afsnit D: Inceptionsdokument](#), side 66.

Endvidere blev det muligt at udarbejde en domænemodel (Se [Bilag F5: Domænemodel](#), side 82) ved afslutning af kravspecifikationen. Denne model var et vigtigt redskab til kommunikationen med kunden, da man kunne se hvordan systemet ville blive modelleret ud fra problemdomænet. Modellen blev brugt til den videre analyse og udarbejdelse af analyseklassediagrammet.

¹³ FURPS+: <http://agileinaflash.blogspot.dk/2009/04/furps.html>

3.3 Analyse

Analyseafsnittet omhandler den første proces der skal behandles, når man skal implementere software. Dette afsnit beskriver, hvordan man kommer fra de udvalgte krav i systemet til et analyseklassediagram med sekvensdiagrammer. Disse artefakter skal bruges i næste process, som omhandler at lave et fyldestgørende design til implementering.

3.3.1 Den statiske side af analysemodellen

Den statiske side af analysemodellen beskriver hvilke informationer, der er nødvendige for at brugsmønstrene kan realiseres.¹⁴ For at finde ud af hvilke informationer der skal anvendes, foretages der en navneord- og udsagnsordsanalyse. Vurderingen af hvilke oplysninger der skal medtages, gøres ud fra hvorvidt oplysningerne befinder sig i problemdomænet. Disse oplysninger vil afspejle klasser, attributter og metoder, som analyseklasserne kan indeholde. Det udarbejdede analyseklassediagram kan ses under [Bilag F6: Analyseklassediagram](#), side 83. Her ses en udvikling fra domænemodellen, idet analysemodellen indeholder flere tekniske informationer heriblandt datatyper og metoder.

I den foregående inceptionsfase, blev der udarbejdet otte brugsmønstre, som afspejlede det afgrænsede system (Se [Afsnit D: Inceptionsdokument](#), side 62). Herefter analyseres de identificerede brugsmønstre.

Nedenstående tabel viser en analyse af brugsmønstret *opret sag* (BM4 afsnit 3.2).

Navneord Analyse	Type	Udsagnsord Analyse	Type
Sagsbehandler	klasse	Logge sag	metode
Borger	klasse	Oprette sag	metode
Henvendelse	klasse	Knytte henvendelse på sag	
Sag	klasse	Indtaste oplysninger	metode
ID	attribut		

Tabel 3.3.1: Resultat af analyse for BM4

¹⁴ [Borgersen, 2011, kap. 8](#)

3.3.2 Den dynamiske side af analysemodellen

Den dynamiske side af analysemodellen handler om at bearbejde de informationer, der er fundet under arbejde med den statiske analyse. Det handler mere specifikt om, hvordan analyseklasserne og informationerne skal kommunikere med hinanden for at realisere et givent brugsmønster.¹⁵ Denne process udarbejdes af to artefakter. Det første handler om at oprette kontrakter, som beskriver hvilket ansvar et systems operation har ift. hvilket brugsmønster, der er tale om. Kontrakterne beskriver hvilken tilstand, systemet gennemgår, når en specifik operation eksekveres. En kontrakt er vist i en tabel med følgende informationer (operation, krydsreference, ansvar, output, præ- og postkondition).

Det sidste element i den dynamiske analyse omhandler sekvensdiagrammer, som visualisere kommunikationen mellem analyseklasserne. Denne kommunikation vil afspejle, hvordan et givent brugsmønster realiseres ved at modellere flowet i systemet, fra en operation kaldes til operationens eksekveringen er færdig. Realiseringen af brugsmønstermodellen resultere i en række sekvensdiagrammer, som kan ses under [Bilag F7: Sekvensdiagrammer](#), side 84-87. Der er valgt at have fokus på operationerne, hvor der åbnes en henvendelse og en sag, da disse spiller den mest centrale rolle i systemet.

Der er ved den dynamiske analyse valgt, at gå fra dansk til engelsk, da brugsmønsterrealiseringen er et af de første trin i implementeringsflowet. Det blev besluttet i fællesskab i gruppen, at kodning blev på engelsk da det forekom naturligt og programmering blev lært på denne måde.

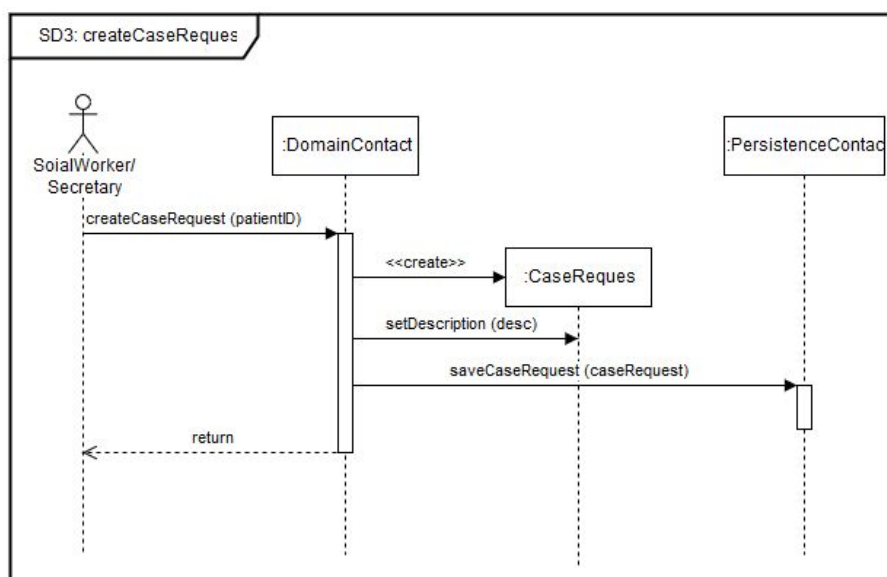
Nedenstående kontrakt (Tabel 3.3.2) er baseret på metoden *createCaseRequest* over brugsmønstret BM3 *Opret Henvendelse*.

Kontrakt	
Operation	createCaseRequest(caseRequestID, patientID)
Krydsreference	BM3 Opret Henvendelse
Ansvar	At oprette en henvendelse og indsætte information fra borgeren. At gemme henvendelsen i databasen.
Output	Brugeren får besked om at oprettelsen af henvendelsen
Prækondition(er)	Bruger skal være logget ind.
Postkondition(er)	En henvendelse er oprettet. Henvendelsen er gemt i databasen.

Tabel 3.3.2: Denne tabel viser kontrakten med udgangspunkt i BM3 Opret Henvendelse.

¹⁵ [Borgersen, 2011, kap.13](#)

Det nedenstående sekvensdiagram (figur 3.3.1) hører sammen med ovenstående kontrakt og viser interaktionen mellem den specifikke aktør - sagsbehandler eller sekretæren - og systemet. Horisontalt vises livslinjerne, der kan repræsentere objekter, deltagere og aktører. Den vertikale akse viser forløbene i tid. Dvs. at der startes oppefra venstre hjørne, hvorefter interaktionen forløber mod højre og nedad.



Figur 3.3.1: Denne figur viser sekvensdiagrammet ud fra BM3 Opret Henvendelse.

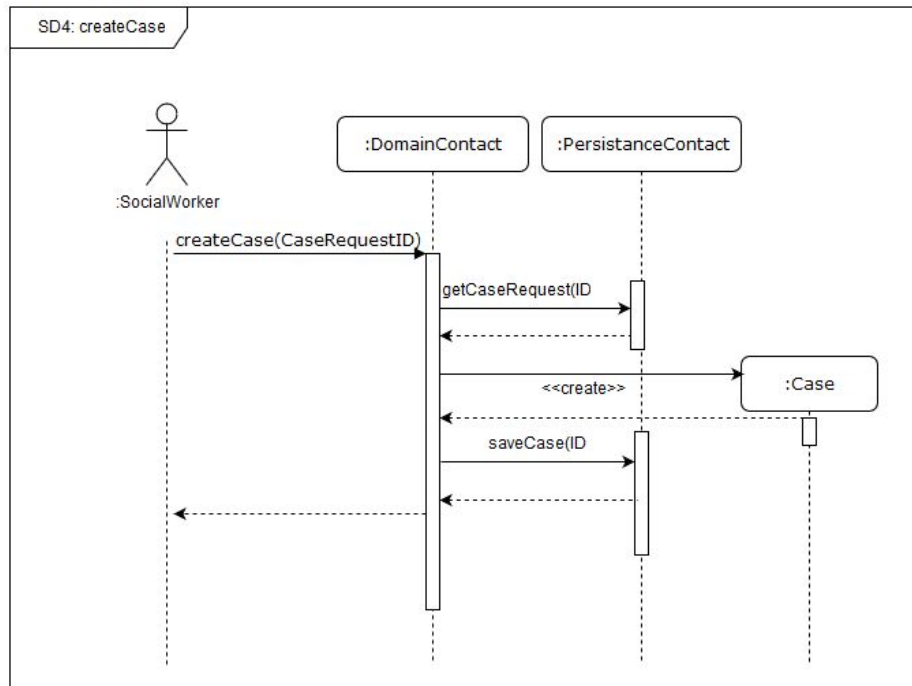
Nedenstående kontrakt (Tabel 3.3.2) er baseret på metoden *createCase* (Opret Sag).

Kontrakt	
Operation	createCase(caseRequestID)
Krydsreference	BM4: Opret Sag
Ansvar	At oprette en sag og indsætte information fra henvendelsen. At gemme sagen i databasen.
Output	En digital sag er oprettet. Sagsbehandleren får en bekræftelse på at sagen er oprettet.
Prækondition(er)	Sagsbehandleren skal være logget ind, og henvendelsen skal være konkret.
Postkondition(er)	En sag er oprettet. Sagen er gemt i databasen.

Tabel 3.3.3: Denne tabel viser kontrakten med udgangspunkt i BM4 Opret Sag.

Kontrakten forklarer, hvad der ideelt skal ske, når metoden *createCase* bliver kaldt. Både med hensyn til outputtet men også med de konditioner, det kræver.

Det nedenstående sekvensdiagram (figur 3.3.2) hører sammen med ovenstående kontrakt og viser interaktionen mellem den specifikke aktør og systemet.



Figur 3.3.2: Denne figur viser sekvensdiagrammet for brugsmønsteret BM4 Opret Sag.

3.4 Design

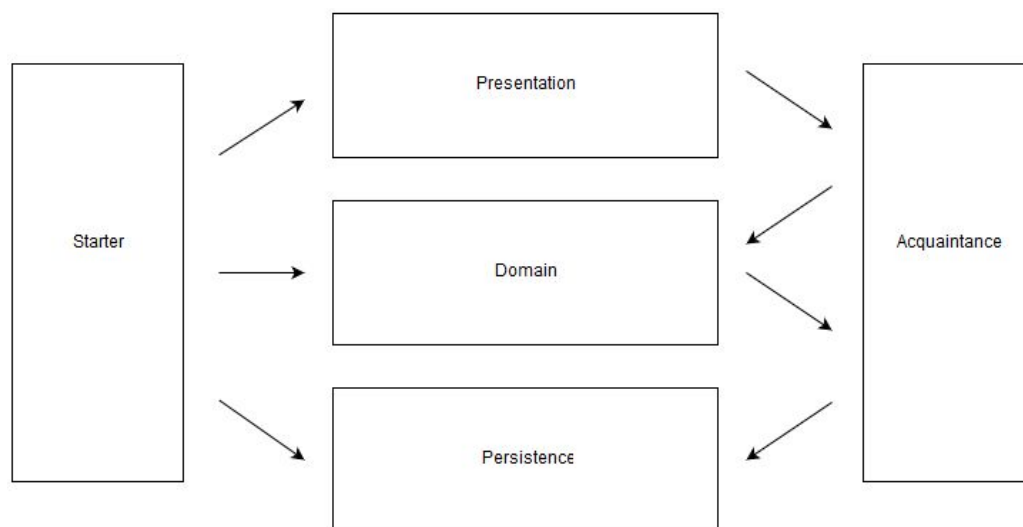
Dette afsnit omhandler design - der i særdeleshed bygger videre på analyseafsnittet. Der er i designafsnittet fokus på, hvordan man går fra analyse til design og videre til den første implementering i kode. Et af designvalgene har fx været omkring arkitektur, og hvorledes denne skulle implementeres. Sideløbende med design og implementering er der blevet lavet et designklassediagram som kan findes i [Bilag F8: Designklassediagram](#), side 88.

3.4.1 Softwarearkitektur

Arkitekturen som er blevet brugt i projektet, er en trelags arkitektur med følgende lag:

- Præsentation (Presentation layer)
- Domæne (Domain layer) og
- Persistens (Persistence layer)

Udover de tre lag, er der også 2 yderligere pakker ved navn Starter og Acquaintance, som blev tilføjet i iteration 2. I ([Bilag F9: Arkitektur Diagram](#) side 89, kan man se arkitektur diagrammet fra iteration 1). Kigger man på nedenstående figur (3.4.1) kan man se en afbildning af arkitekturen.



Figur 3.4.1: Denne figur præsenterer tre-lags-arkitekturen.

En af fordelene ved at bruge en tre-lags arkitektur er, at det forøger sikkerheden, ift. hvordan data bliver tilgået. Ved at anvende denne arkitektur vil man f.eks. ikke kunne kommunikere direkte med persistenslaget, da dette ville være et sikkerhedsbrud. I stedet bliver der oppe fra præsentationslaget sendt en forespørgsel til domænelaget igennem acquaintance pakken. Domænelaget vil så sende en forespørgsel til persistenslaget, også igennem acquaintance pakken. Acquaintance pakken indeholder systemets kontrakter, i form af interfaces, som gør at der ikke burde være nogen direkte kald mellem de forskellige lag¹⁶. Starter pakken sørger for at binde de forskellige "lag" sammen når systemet skal startes op.

¹⁶ Acquaintance: <https://drive.google.com/file/d/1gktHhYQmmpI-Jmv0vImclrAuWZE6wrl4/view>

3.4.2 Design af persistenslaget

Persistenslagets design er blevet skiftet mellem iterationerne. I første iteration var der klasserne WriteTXT og ReadTXT, som skrev til tekstfiler. Disse tekstfiler fungerede som en database.

Man kan se tekstfilerne i nedenstående tabel (tabel 3.4.1):

Tekstfiler	Oplysninger
CPR-register	Indeholder oplysninger omkring personerne
Cases	Indeholder informationen vedrørende sagerne
Case requests	Indeholder informationen vedrørende henvendelserne
Employee	Indeholder på de ansattes information
ID fil	Indeholder informationen som PersistenceContact bruger til at give nye ID'er
Log	Indeholder informationer om logning af de ting der er blevet registreret i systemet.

Tabel 3.4.1: Tabellen viser oversigt over filer samt deres oplysninger.

I anden iteration blev persistenslaget ændret drastisk. Der blev lavet klasser til at manipulere dataen og sende den ned i en SQL database. Klasserne er følgende:

- AbstractDB
- WriteDB
- ReadDB

AbstractDB

Denne klasse er en abstraktion, som beskriver noget fælles, som de andre klasser i persistenslaget skal kunne, nemlig at oprette en forbindelse (connection) til databasen. Derfor er AbstractDB en abstrakt klasse (kan ikke instantieres) og fungerer som en superklasse for både ReadDB og WriteDB, da begge klasser skal bruge forbindelsen.

ReadDB

Denne klasse skal kunne hente alt den nødvendige information fra databasen og sender informationen videre til domænelaget i form af String Arrays.

WriteDB

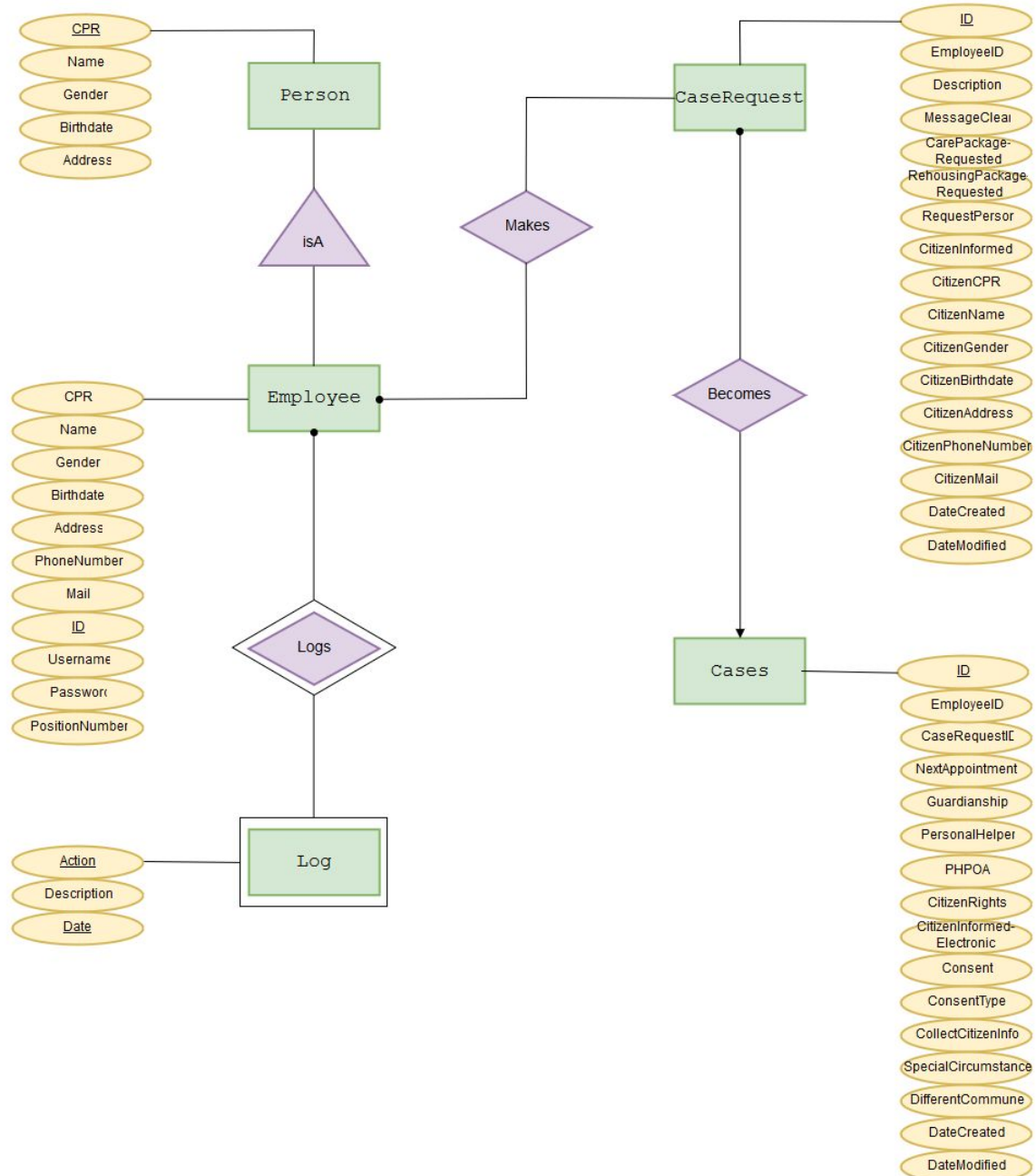
Denne klasse skal kunne skrive sager, henvendelser, ansatte og logge hver gang der sker noget i systemet, til databasen.

Facadeklasse: PersistenceContact

For at kunne kommunikere med persistenslaget bliver der anvendt en facadeklasse kaldet *PersistenceContact*. Den er placeret i domænelaget og kan kommunikere med persistenslaget via interfaces, der bliver linket til den med 'inject' metoder. Med disse metoder kan der sendes en reference af en reader og en writer, og systemet kan derefter sende informationen videre til persistenslaget ved brug af referencerne. *PersistenceContact* laver også *ReadDB's* String Arrays om til objekter ved de kald, der handler om at hente information fra tekstfilerne. Alt dette gør at klassen både fungerer som en facadeklasse (som ligger i det forkerte lag) og en oversætter. Denne funktionalitet kunne med fordel splittes op i to klasser, som ville ligge i henholdsvis domæne- og persistenslag.

Database

Kigger man på nedenstående figur (figur 3.4.2) kan man se en E/R model over databasen. Her kan der ses de entities, som er i databasen og deres forhold (relationships). De grønne firkanter repræsenterer en entity, mens de gule cirkler præsenterer attributterne. Forholdene er illustreret ved de lilla diamanter, hvor teksten giver en passende beskrivelse mellem entiteterne. Den lille trekant, isA, illustrerer nedarvningen mellem person og ansat.



Figur 3.4.2: Denne figur viser E/R modellen over databasen.

For at danne et overblik er der lavet en relationel model med tabellerne:

Person(CPR, Name, Gender, Birthdate, Address)

Employee(CPR, Name, Gender, Birthdate, Address, PhoneNumber, Mail, ID, Username, Password, PositionNumber)

CaseRequest(ID, EmployeeID, Description, MessageClear, CarePackageRequested, RehousingPackageRequested, RequestPerson, CitizenInformed, CitizenCPR, CitizenName, CitizenGender, CitizenBirthdate, CitizenAddress, CitizenPhoneNumber, CitizenMail, DateCreated, DateModified)

Cases(ID, CaseRequestID, NextAppointment, Guardianship, PersonalHelper, PHPOA, CitizenRights, CitizenInformed, Consent, ConsentType, CollectCitizenInfo, SpecialCircumstances, DifferenCommune, State, DateCreated, DateModified)

Log(EmployeeID, Action, Description, Date)

Her kan man se de nøgler (de ord der er understreget) og attributter, som er lavet til hver tabel. Ser man på E/R modellen (figur 3.4.2) kan man se at forholdet mellem Person tabellen og Employee tabellen er modelleret ved et isA forhold. Grunden til det forhold er, at Employee bliver lavet ud fra Person og er en mere specifik version af Person (derfor minder det lidt om et arve hierarki). Dette forhold er grunden til at meget af dataen går igen, og den redundante data er altså bevidst.

Mellem Employee og Log tabellerne er der et forhold ved navn Logs, som er et én til mange forhold. Da en Employee laver flere Log(s), men en Log hører til præcis én Employee. Dette er modelleret som et weak entity set, da Log settets attributter ikke alene er nok til at definere en primær nøgle. Når det er modelleret som et weak entity set, kræver det at Log har en fremmednøgle (foreign key) til Employee. Denne fremmednøgle består af EmployeeID som hænger sammen med ID på Employee tabellen.

Ved CaseRequest tabellen kan man se, at den også har et forhold til Employee, foruden at den er modelleret som et mange til én forhold. Grunden til dette er, at en CaseRequest er lavet af præcis én Employee, men én Employee kan lave mange CaseRequest. Da CaseRequest indeholder alt den data som en Person består af, er der egentlig redundant data. Dette gøres, fordi Person tabellen egentlig agerer som et CPR register, hvor dataen i tabellen skal opdateres konstant, og dataen om en Person (citizenCPR til og med citizenAddress) i CaseRequest ikke skal opdateres konstant. Den data, som er om en borger i en CaseRequest, skal være den data, som blev nedskrevet på daværende tidspunkt, da CaseRequest(en) blev lavet. Hvis der var et forhold mellem Person og CaseRequest, og man fjernede Person informationen caseRequest, så ville Person oplysningerne i CaseRequest hele tiden blive opdateret.

Fra CaseRequest tabellen til Cases tabellen er der et forhold ved navn Becomes (bliver til), som er modelleret som et én til én forhold. Forskellen er dog at Cases hører til præcis én

CaseRequest, hvor CaseRequest hører til 0 eller 1 Cases. Grunden til dette er, at en sag (Case instans af Cases) altid vil komme fra én henvendelse (instans af CaseRequest), men en henvendelse ikke altid vil blive til en sag.

Ingen af de mange forhold der er i systemet, bliver gemt i databasen. Der er altså ikke noget data at hente på f.eks. forholdet Becomes. Grunden til dette er, at ingen af forholdene er modelleret som mange til mange, hvilket er kravet for at der skal være en særskilt tabel til forholdet.

Det var planen at databasen skulle være på tredje normalform (3NF) og ville være modelleret som E/R modellen i [Bilag F10: 3NF E/R Model](#), side 90. Dette har vist sig ikke at være et realistisk billede, da der blandt andet ønskes redundant data nogle steder i databasen, og det at dele tabellerne op i flere tabeller ikke er passende til den funktionalitet, som er tiltænkt i systemet. Tredje normalform havde altså givet mening i det ideelle scenarie, men har vist sig ikke at være anvendeligt her.

3.4.3 Design af domænelaget

Domænelaget indeholder flere forskellige klasser (både abstrakte og konkrete), som bruges til at håndtere den forretningsmæssige logik i systemet. Det er i dette lag, at der bliver oprettet sager, ansatte mm., som bliver sendt ned til persistenslaget, og det er også her, at alt logningen af handlinger, som bliver udført i systemet, finder sted.

I klassediagrammet kan domænelaget ses i midten af diagrammet. Domænelaget er bygget op omkring flere arvehierakier, heriblandt er klassen *Person* værd at nævne, som mange andre klasser arver fra. *Person* har nogle basale informationer omkring en person, og disse informationer bliver også givet videre til klassen *Employee*, som arver fra den. Fra *Employee* bliver klasserne yderligere specialiserede, og der kommer en *CaseEmployee* klasse som både *Secretary* og *SocialWorker* klasserne arver fra med deres specielle egenskaber. Der er også klassen *Admin*, som arver direkte fra *Employee*, da administratoren ikke skal kunne oprette og ændre sager og henvendelser.

Disse "person" klasser munder ud i, at hver ansat har forskellige muligheder i systemet ud fra deres position. En *Secretary* kan lave henvendelser i systemet, hvor en *SocialWorker* kan lave reelle sager (som bliver skabt på baggrund af henvendelser). Derudover kan en *SocialWorker* også redigere sager, samt oprette henvendelser ligesom en sekretær kan. En *Admin* kan oprette nye "personer" i systemet og slette dem. Valget, at lade de forskellige ansatte arve fra forskellige klasser, blev taget, da det umiddelbart virkede som en god måde at adskille de forskellige ansattes adfærd.

Der er både en *LogAction* og en *Log* klasse, som tilsammen er med til at logge de ting der sker i systemet. Klassen *LogAction* består af nogle faste værdier, som svarer til de

handlinger, der kan blive udført i systemet, og *Log* tager en instans af *LogAction*, for dermed at fortælle, hvad der sket i den pågældende *Log*. Når der sker noget i systemet, som på nogen måde berører personer (om det er ansatte eller borgere), bliver det gemt i en instans af *Log* klassen og sendt ned til *PersistenceContact*, hvor det derfra sendes ned i persistenslaget.

Der er mange interfaces i domænelaget, og mange af de forskellige klasser implementerer et eller flere af de interfaces. Med denne løsning er det muligt at sende information ned til persistenslaget uden at klasserne dernede kender til den konkrete implementering, og dermed holdes de forskellige lag adskilt.

Facadeklasse: DomainContact

I domænelaget ligger der også en facadeklasse ved navn *DomainContact*, som er designet som en Singleton (klassen kan kun instantieres en gang). Det er denne klasse, som bliver brugt til kommunikation mellem præsenterationslaget og domænelaget. *DomainContact* er en klasse med mange af de metoder, som skal bruges i brugergrænsefladen, hvor disse metoder kalder andre metoder i de reelle klasser i domænelaget. Det er også *DomainContact* der holder styr på den nuværende bruger, som er i systemet. Valget af facadeklasser hjælper med at holde forskellige lag, så adskilt som muligt og giver mulighed for at holde kommunikationen enstrenget. Med *DomainContact* klassen kan præsenterationslag udskiftes uden det store besvær, eftersom den adskiller domæne- og præsenterationslaget.

Auto logout: SystemTimer

I 2. iteration blev der udviklet en auto logout mekanisme i systemet. Dvs, at en bruger bliver logget ud af systemet, hvis han/hun har været inaktiv i en længere periode. Timeren bliver implementeret som en tråd, der kører sideløbende med selve systemet og kalder på præsenterationslaget, for at fortælle at en bruger nu eller snart skal logges af.

Metoder i præsenterationslaget tager hånd om at sende informationen videre og fortælle det til brugeren. Eftersom klasser i domænelaget ikke direkte må kalde på klasser i præsenterationslaget, bliver der sendt en reference af brugerfladen, som overholder kontrakten *IVisualController*. Med denne kontrakt kan SystemTimer nu fortælle brugerfladen, både hvornår brugeren snart bliver logget ud og når brugeren rent faktisk bliver logget ud.

3.4.4 Design af præsenterationslaget

Kravet til første iteration er, at der skal være en brugerflade (UI), som brugeren skal bruge til at kunne kommunikere med systemet. Denne UI blev designet som en tekstbaseret brugerflade, hvor brugeren først skulle skrive den handling som var ønsket udført og efterfølgende tilføje de detaljer som handlingen påkrævede. Handlingerne kan ses i nedenstående tabel (tabel 3.4.2).

Brugsmønster	Kommando	Handling
Opret henvendelse	CaseRequest	Denne kommando opretter en henvendelse ud fra inputtet
Opret sag	Case	Denne kommando opretter en sag ud fra inputtet
Rediger sag	EditCase	Denne kommando henter en sag og gør det muligt at tilføje ny information i sagen
Tilføj ansat	AddEmployee	Denne kommando tilføjer en ansat ud fra inputtet
Fjern ansat	DeleteEmployee	Denne kommando fjerner den ansatte som svarer til det input der er givet

Tabel 3.4.2: Tabellen præsenterer brugsmønstre, kommando-navn samt handling.

Præsentationslaget er delt op i 2 dele. Et objekt som tager imod input fra brugeren og et objekt som omsætter inputtet til det signal som sendes videre. Det ses i form af de to klasser *TextInputer* og *CommandConverter*.

TextInputer

Denne klasse står for at tage imod input fra brugeren jvf. tabel 3.4.1. Brugeren skriver først navnet på den kommando/handling, som ønskes udført. Efterfølgende bliver brugeren stillet en række spørgsmål, som svarer til de informationer, der skal bruges i forbindelse med handlingen. For eksempel hvis kommandoen 'CaseRequest' bliver aktiveret, vil brugeren blive spurgt til information om den pågældende borger og henvendelse. Når al informationen er indsamlet bliver den sendt videre til *CommandConverter*.

CommandConverter

Denne klasse tager imod input og viderefører det til domænelaget. Når et input er blevet opsamlet, bliver det omsat til de korrekte datatyper og sendes derefter ned til domænelaget gennem facadeklassen, *DomainContact*'s metoder.

Med denne opsætning er det muligt, at skifte den tekstbaserede brugerflade ud med en grafisk brugerflade, som kun skal kunne kommunikere med klassen *CommandConverter*.

I andet sprint blev der arbejdet med at få skiftet den tekstbaserede brugergrænseflade ud, med et dynamisk og intuitivt brugergrænseflade. Grundet den lagdelte arkitektur, kunne der arbejdes videre i præsentationslaget, uden konflikter med de andre lag.

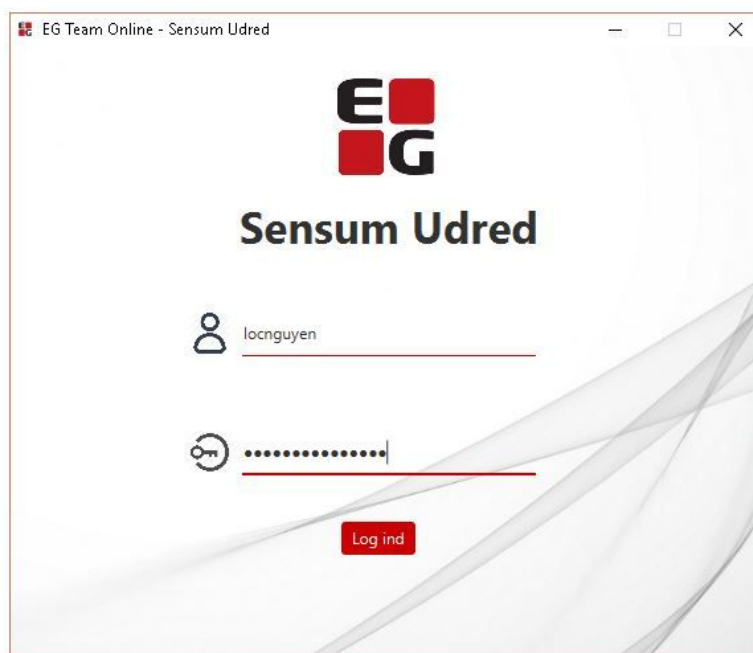
Gruppen indså ved dette sprint, at det var fejlagtigt at begynde at implementere på engelsk, da der var tale om et dansk system. Derfor forekommer danske termer i implementeringen, dog kun i præsentationslaget, da det gav bedst mening når man skulle få fat på relevante elementer m.m. i brugergrænsefladen.

I første sprint fungerede brugergrænsefladen ved, at man skulle indtaste de engelske kommandoer, som man ville bruge. Dette blev ændret, så man i stedet for at skulle indtaste

kommandoerne ordret hver gang, blev der tilføjet javaFX. Disse kommandoer er i andet sprint blevet ændret til knapper, således man ikke skal skrive hele navnet på den ønskede kommando, men blot klikke eller trykke. Dette håndteres ved at der er kommet event handlers, som lytter til når der bliver interageret med brugergrænsefladens knapper. Informationen skiftede fra at blive sendt fra TextInputpter til *CommandConverter*, til i stedet at blive sendt fra GUI til *CommandConverter*. Disse informationer bliver sendt vha. de forskellige handle metoder i Controller klasserne, *FXMLDocumentController* og *LoginScreenController*. Begge af disse klasser har hver en reference til *CommandConverter*, således at kommandoerne i *CommandConverter* kan blive kaldt med informationer fra brugergrænsefladen. *CommandConverteren* sørger selv for at sende informationerne videre til domænelaget ved hjælp af referencen til facaklassen som er givet gennem metoden: *injectDomainContact*. Derfra vil domænelaget snakke sammen med *PersistansContact* som på samme måde snakker sammen med databasen.

Når systemet bliver startet, vil det første, som brugeren ser, være en login skærm. Denne login skærm er blevet lavet, for at sørge for at brugeren kun kan logge ind, såfremt han/hun er oprettet i systemet. Det bliver også brugt til at adskille funktionalitet, som brugerne hver især har (Se tabel 3.4.2- brugsmønstre, side 30-31). F.eks. skal en sagsbehandler ikke kunne oprette ansatte, ligesom en systemadministrator kan. Brugergrænsefladen er designet således, at der bliver tjekket for hvilken type ansat som er logget ind. Alt efter hvilken ansat der er logget ind, vil kun de knapper blive vist, som brugeren er autoriseret til at benytte.

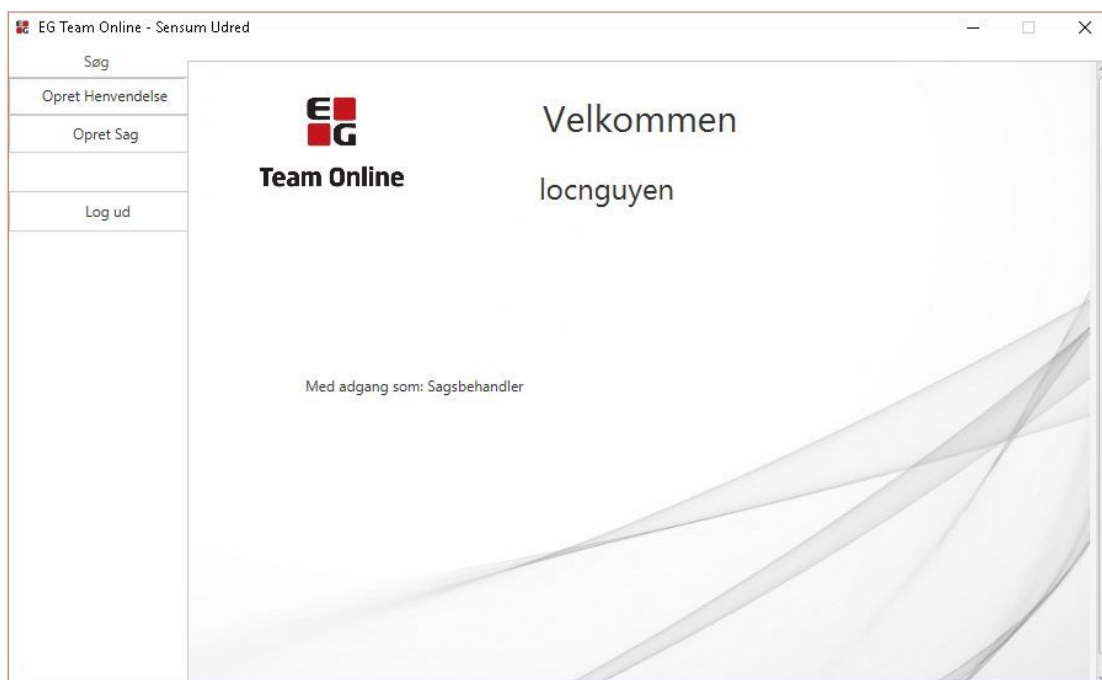
Nedenfor vises et eksempel hvor man logger ind som en sagsbehandler:



Figur 3.4.3: Figuren præsenterer projektets login skærm.

Ved log ind, vil knappen som vedrører oprettelse eller opsigelse af en ansat blive fjernet, da en sagsbehandler ikke har autoritet til at benytte denne funktion. Dog har en sagsbehandler autoritet til de resterende handlinger, som systemet har.

Det kan ses ude i siden af systemet, at der faktisk kun vises 3 ud af 4 knapper når man er logget ind som en sagsbehandler.



Figur 3.4.4: Figuren præsenterer projektets forside efter login som sagsbehandler.

Søge-feltet bliver brugt til at indtaste CPR-numre, for at søge efter henvendelser eller sager i systemet, som er tilknyttet den specifikke borger (man kan se både sager og henvendelser som sagsbehandler, men kun henvendelser som sekretær). Hvis der findes henvendelser eller sager på borgeren, vil man enten kunne oprette en sag for de henvendelser der eksisterer, eller redigere i en sag, hvis der er en aktuel sag.

Opret henvendelse-knappen er her hvor brugeren føres til en digital og dynamisk formular, hvor brugeren kan tilføje de oplysninger, som bliver modtaget ved henvendelsen.

Opret sag-knappen sender brugeren hen til en sagsformular, hvori brugeren kan udfylde de nødvendige informationer om den sag, som skal igangsættes.

Administrer ansat-knappen (det felt som ikke er vist på billedet), kan kun ses af systemadministratoren. Knappen sender brugeren videre til en side hvor der håndteres oprettelse eller opsigelse af ansatte.

Log ud-knappen logger brugeren ud af systemet, og man vil blive sendt tilbage til log ind scenen.

3.5 Implementering

I dette afsnit vil der blive gennemgået dele af den fuldendte implementering i form af kode. Dog vil alting ikke være relevant i klasserne, og derfor vil visse ting ikke være med i koden. Er der interesse for at inspicere den fuldendte kode, refereres der til source koden. Når der står ** i kode eksemplerne, er det fordi noget af koden er udeladt i eksemplet.

3.5.1 Implementeringen af centrale dele i persistenslaget

Eftersom designet af persistenslaget har gennemgået et stort skift igennem de to forskellige iterationer, ses dette ligeledes også i implementeringen. Da systemet har været lagdelt, og der har været tilhørende interfaces til persistenslaget, har det været relativt nemt at skifte de forskellige lag ud.

Persistenslaget - Første iteration

I første iteration bestod persistenslaget af to klasser, nemlig WriteTXT og ReadTXT. Disse klasser havde til opgave at holde styr på det forskellige data i form af en tekstfil. WriteTXT stod for at skrive og slette til og fra tekst filerne, vha. af forskellige input/output klasser. I den pågældende metode (f.eks. writeCase) fik man en reference med til et givent objekt igennem det tilhørende interface (f.eks. ICase hvis det var en instans af Case). Alt dataen blev hentet ud i form af Strings fra referencen, og så skrevet ned i den tilhørende tekstfil. ReadTXT stod for at læse tekst filerne og hive den information ud som skulle bruges, hvor der ofte blev søgt efter et specifikt ID eller lignende. Der blev brugt Scanner klassen til at læse filerne, og når den fandt den rigtige linje, gemte den alt informationen i et String Array og returnerede det.

Persistenslaget - Anden iteration

I anden iteration blev der oprettet tre nye klasser, disse klasser skulle erstatte de foregående WriteTXT og ReadTXT, ved at skabe en kobling til projektets database. Klasserne er: AbstractDB, WriteDB og ReadDB.

AbstractDB

Kigger man på nedenstående figur (figur 3.5.1) ser man AbstractDB klassen.

AbstractDB.java (getCase)

```
public abstract class AbstractDB {  
    protected Connection db;  
  
    public AbstractDB() {  
        db = getDBConnection();  
    }  
  
    private Connection getDBConnection() {  
        db = null;  
        try {  
            Class.forName("org.postgresql.Driver");  
        } catch (java.lang.ClassNotFoundException e) {
```

```
        System.out.println(e);
    }

    String url = "jdbc:postgresql://horton.elephantsql.com:5432/vojxahth";
    String username = "vojxahth";
    String password = "2AaRD453-Lm37Z5wgl4SBdXANa7jycpC";
    try {
        db = DriverManager.getConnection(url, username, password);
    } catch (SQLException e) {
        System.out.println("SQL error in getDBConnection()");
    }
    return db;
}
```

Figur 3.5.1: AbstractDB klassen.

Denne klasse står for at lave en forbindelse til databasen, dette gør den vha. metoden `getDBConnection`. Først i klassen bliver der lavet en protected variabel af klassen `Connection`. Grunden til dette er, at når den er protected, så kan alle klasserne der arver fra klassen få direkte adgang til variabelen. I metoden `getDBConnection` er der tre Strings, hvilket er de informationer der skal bruges til at tilgå databasen. Disse Strings bliver givet med når forbindelsen skal laves.

WriteDB

Helt overordnet sørger `WriteDB` klassen for at skrive den information, som brugeren indtaster i den grafiske brugergrænseflade, til databasen, hvor informationen her bliver gemt efter de efterspurgte datatyper fra databasen. Herunder i figur 3.5.2 fremvises et eksempel fra `writeCase` metoden.

```
WriteDB.java (writeCase) ** Se koden for fuld implementering

public int writeCase(ICase cases) {
    try {
        **
        String query1 = "DELETE FROM Cases " +
            "WHERE ID = " + cases.getID();
        PreparedStatement ps = db.prepareStatement(query1);
        ps.execute();
        String query2 = "INSERT INTO Cases "
            + "VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
        ps = db.prepareStatement(query2);
        ps.setInt(1, cases.getID());
        **
        ps.setString(4, cases.getNextAppointment());
        **
        ps.setBoolean(9, cases.isCitizenInformedElectronic());
        **
        ps.execute();
        ps.close();
        return cases.getID();
    } catch (SQLException ex) {
        System.out.println("WriteDB: SQL error in writeCase()");
    }
    return -1;
}
```

Figur 3.5.2: WriteDB klassen med writeCase metoden.

WriteCase metoden sørger både for at kunne oprette og redigere en sag i/til databasen. Når en ny sag skal oprettes, bliver der i metoden oprettet en ny query, som klargøre 17 elementer til at blive skrevet ned i "Cases" tabellen, vha. et "INSERT INTO" statement. Herefter bliver der hentet hver enkel information som skal bruges, fra den case der bliver givet med som parameter i metoden. Disse informationer bliver lagret i et "PreparedStatement" der giver dataene videre i et SQL statement, og slutteligt kalder "execute" metoden, hvor sagen bliver gemt i databasen.

Hvis en sag derimod skal redigeres, startes der med at slette det ID som passer med den specifikke sag man vil redigere i, herefter indsættes informationerne på ny, på præcis samme procedure, som når der skal oprettes en ny sag, herved overskriver man blot den gamle sag med den nye.

ReadDB

ReadDB klassen har mange metoder, der hver især står for at læse noget fra databasen, og sende det tilbage i form af et String array. Et eksempel på dette er getCase metoden, som man kan se i nedenstående figur (Figur 3.5.3):

ReadDB.java (getCase) ** Se koden for fuld implementering

```
public String[] getCase(int id) {
    String[] caseInfo = new String[17];
    try {
        Statement st = db.createStatement();
        String query = "SELECT * FROM Cases WHERE id = " + Integer.toString(id) + "";
        ResultSet rs = st.executeQuery(query);
        while (rs.next()) {
            caseInfo[0] = rs.getString(1);
            **
            caseInfo[16] = rs.getString(17);
        }
        rs.close();
        st.close();
    } catch (SQLException e) {
        System.out.println("SQL error in getCase()");
    }
    return caseInfo;
}
```

Figur 3.5.3: ReadDB klassen med getCase metoden.

Kigger man på metoden, bliver der begyndt med at lave et String array med plads til 17 elementer, da der er 17 elementer, som skal læses og gemmes. Der bliver brugt klassen Statement, hvor der bliver kaldet createStatement metoden på variabelen db (den protected variabel der kommer fra AbstractDB). Herefter bliver den SQL query der skal kaldes, gemt i en String, som bliver brugt i et ResultSet. Queryen henter alt information fra tabellen Cases hvor ID'et er lig med det ID som bliver givet med som argument i metoden. Informationerne bliver hentet en af gangen i et while loop, og bliver gemt på hver sin plads i String arrayet. Til sidst bliver alle de forskellige ressourcer lukket, og arrayet bliver returneret.

PersistenceContact

PersistenceContact fungerer som ensretter af information til persistenslaget. Den er oprettet som en singleton, hvilket sikrer, at der kun er én instans af klassen, og at den kan tilgås fra hele domænelaget. *PersistenceContacts* attributter og singleton implementering ses i nedenstående kode (3.5.4):

PersistenceContact.java ** Se koden for fuld implementering

```
private static PersistenceContact instance = null;
private IWriter writer;
private IReader reader;
private CaseRequest caseRequest;
private int currentCaseID;
private int currentCaseRequestID;
private int currentEmployeeID;

public static PersistenceContact getInstance()
{
    if (instance == null)
        instance = new PersistenceContact();
    return instance;
}
```

Figur 3.5.4: PersistenceContact klassen.

Når en reader bliver injected i *PersistenceContact*, bliver den private metode 'readCurrentIDs()' kaldt, hvilket læser attributterne ind som kan ses i nedenstående kode (figur 3.5.5).

PersistenceContact.java (readCurrentIDs)

```
private void readCurrentIDs(){
    int[] ids = reader.getCurrentIDs();
    this.currentCaseID = ids[0];
    this.currentCaseRequestID = ids[1];
    this.currentEmployeeID = ids[2];
}
```

Figur 3.5.5: PersistenceContact klassen med readCurrentIDs metoden.

Dette sørger for at når der bliver lavet nye cases, caserequests og employees så vil de altid have et ID der er én større end det nuværende største.

3.5.2 Implementeringen af centrale dele i domænelaget

En hel central del af domænelaget består i at logge de handlinger som ansatte laver i systemet. Dette bliver gjort vha. Log og *LogAction* klassen. Kigger man på nedenstående figur (figur 3.5.6) ser man *LogAction* klassen.

LogAction.java** Se koden for fuld implementering

```
public enum LogAction {  
  
    SAVE_CASE_REQUEST("saveCaseRequest"), SAVE_CASE("saveCase"),  
    SAVE_EMPLOYEE("saveEmployee"), DELETE_EMPLOYEE("deleteEmployee"),  
    GET_CASE_REQUEST("getCaseRequest"), GET_CASE("getCase"), GET_EMPLOYEE("getEmployee"),  
    LOG_IN("logIn"), LOG_OUT("logOut");  
    private String commandString;  
  
    public LogAction(String commandString) {  
        this.commandString = commandString;  
    }  
}
```

Figur 3.5.6: LogAction klassen

LogAction klassen er blevet lavet som et enum, hvor der er nogen faste muligheder for hvad der kan blive logget. Denne enum bliver brugt i Log klassen som man kan se på nedenstående figur (figur 3.5.7):

Log.java** Se koden for fuld implementering

```
public class Log implements ILog{  
    private int employeeID;  
    private LogAction action;  
    private String desc;  
    private Date date;  
  
    public Log(int employeeID, LogAction action, String desc) {  
        this.employeeID = employeeID;  
        this.action = action;  
        this.desc = desc;  
        date = new Date();  
    }  
}
```

Figur 3.5.7: Log klassen.

Log klassen består af fire attributter, som er relevante, når der skulle logges en handling. Et ID for den ansatte, der har udført handlingen, den Action som er blevet udført (f.eks. deleteEmployee), en String med en beskrivelse af, hvad der er blevet gjort, og det tidspunkt handlingen er foregået. Tidspunktet bliver automatisk genereret når handlingen bliver udført.

På nedenstående figur (figur 3.5.8), kan der ses et eksempel på, hvor der bliver logget:

SocialWorker.java (createCase)** Se koden for fuld implementering

```
public int createCase(int caseRequestID, **) {  
    DomainContact dc = DomainContact.getInstance();  
    PersistenceContact pc = PersistenceContact.getInstance();  
    **
```

```
pc.logAction(dc.getCurrentUser().getId(), LogAction.SAVE_CASE, "Created a new case");  
**  
}
```

Figur 3.5.8: SocialWorker klassen med createCase metoden.

I SocialWorker klassen er der en metode til at lave en sag (case), når denne metode bliver kaldt bliver metoden logAction fra PersistenceContact klassen kaldt. I metoden bliver der gemt den aktuelle brugers ID, hvilken action det er og en beskrivelse. Disse informationer bliver herefter sendt ned til persistenslaget.

Domænelaget har implementering af det meste af auto logout funktionaliteten. Først kalder præsentationslaget metoden injectVisualController i *DomainContact*, hvorefter denne opretter en ny tråd og timer til tråden og starter den (figur 3.5.9).

DomainContact.java (injectVisualController)** Se koden for fuld implementering

```
public void injectVisualController(IVisualController IVC) {  
    timer = new SystemTimer();  
    timer.injectVisualController(IVC);  
    **  
    timerThread = new Thread(timer);  
    timerThread.setDaemon(true);  
    timer.injectTimerThread(timerThread);  
    timerThread.start();  
}
```

Figur 3.5.9: DomainContact klassen med injectVisualcontroller metoden.

SystemTimer har en run metode, som bliver kørt at tråden timerThread. Denne metode sørger for at tælle timeren ned og se, om der skal gives en alarm eller brugeren skal logges ud (figur 3.5.10). Hvis dette er tilfældet kaldes en metode på referencen til brugerfladen IVC.

SystemTimer.java (run)** Se koden for fuld implementering

```
public void run() {  
    Thread thisThread = Thread.currentThread();  
    while (timerThread == thisThread) {  
        while (currentTimer > 0) {  
            try {  
                Thread.sleep(1000);  
            } catch (InterruptedException e) {  
                System.out.println(e);  
            }  
            currentTimer--;  
            if (currentTimer == ALERT_TIMER)  
                IVC.alert();  
        }  
    }  
    **  
}
```

Figur 3.5.10: SystemTimer klassen med run metoden.

3.5.3 Implementeringen af centrale dele i præsentationslaget

I første sprint var den centrale klasse, som sørgede for et grafisk afbildning af systemet, *TextInputter* klassen. Denne klasse bestod af prompt strenge, som blev vist til brugeren, som kunne vælge mellem nogle kommandoer. De inputs som brugeren gav, blev herefter sendt videre til *CommandConverter* klassen, som sørgede for at viderelevere informationen til det næste lag for tilsidst at blive gemt i en tekstfil.

I andet sprint, blev ovenstående udskiftet med et JavaFX grafisk brugergrænseflade, og de to væsentligste klasser i præsentationslaget blev controllerne *FXMLDocumentController* og *LogInScreenController*. Disse to klasser styrer de to forskellige scener som systemet bygger på, og det er også her, at informationen fra brugeren bliver taget imod. *TextInputter* klassen fra første sprint, bliver ikke brugt længere i det nye præsentationslag. Informationerne som bliver overført fra det nye præsentationslag, bliver dog stadigvæk sendt til *CommandConverter* klassen.

Kommunikationen fra præsentationslaget ned til domænelaget, sker gennem *CommandConverter*, som har en instansvariable og en inject metode til en reference af domænelaget. Denne kommunikation vil blive startet når event handlerne fra Controller klasserne modtager et event, når systemet kører.

Figur 3.5.11 viser kommunikationen præsentationslaget:

FXMLDocumentController.java (handleOpretHenvendelse)** Se koden for fuld implementering
<pre>@FXML private void HandleOpretHenvendelse(ActionEvent event) { String msc = "false"; if (klarHenvendelseJa.isSelected()) { msc = "true"; } String hbdso = ""; if (støtteTilPraktiskOpgave.isSelected()) { hbdso += "Støtte til praktiske opgaver i hjemmet#"; } ** ** CC.performCommand("caserequest", CPR.getText(), name.getText(), gender.getText(), birthday.getText(), address.getText(), phone.getText(), email.getText(), videreForløbTarea.getText(), msc, hbdso, angivTilbud.getText(), henvendelsesPerson.getText(), indforstået); }</pre>

Figur 3.5.11: FXMLDocumentController med handleOpretHenvendelse metoden.

Metoden *handleOpretHenvendelse* refererer til knappen 'Opret Henvendelse' i brugergrænsefladen og sørger for at sende de indtastede informationer videre til *CommandConverter* (CC).

Udsnit af opret henvendelses formularen:

Figur 3.5.12: Udsnit af henvendelsesformular i systemet.

Metoden fra Figur 3.5.11 tager et `ActionEvent` som argument, når metoden registrerer et event fra brugergrænsefladen, og vil dermed eksekvere koden inden for klammerne. Der er erklæret to lokale variabler i metoden, *msc*(Klar Henvendelse) og *hbdso*(Hvad bliver der søgt om), disse to strenge gemmer information på hvilke checkboxes som er valgt i brugergrænsefladen.

Metoden løber igennem alle checkboxes og input felter ved hjælp af if-sætninger, for at indsamle den information, som er indtastet i formularen. De resterende informationer hentes direkte fra tekstfelterne, da de kan indsamle informationerne med `getText()` metoden, som et `javaFX TextField` kan kalde.

Til sidst i metoden bliver metoden `performCommand` kaldt, ved brug af referencen til `CC.performCommand` ("caserequest, CPR.getText()....). Som argument indsættes alle de informationer som er blevet trukket ud af brugergrænsefladen. Ved brugsmønsteret opret sag, hentes informationerne fra brugergrænsefladen på samme måde.

Figur 3.5.13 viser opret henvendelse delen metoden `performCommand` i *CommanConverter*.

CommandConverter.java (performCommand)** Se koden for fuld implementering
<pre> public void performCommand(String command, String... args) { switch (command) { case "caserequest": try { long citizenCPR = Long.parseLong(args[0]); String citizenName = args[1]; char citizenGender = args[2].charAt(0); ** domainContact.createCaseRequest(citizenCPR, desc, messageClear, carePackageRequested, rehousingPackageRequested, requestPerson, citizenInformed, citizenName, citizenGender, citizenBirthdate, citizenAddress, citizenPhoneNr, citizenMail); } catch (NumberFormatException e) { e.printStackTrace(); } break; ** } } </pre>

Figur 3.5.13: CommandConverter klassen med metoden performCommand.

Ved hjælp af en switch og den tilsendte kommando-streng udvælges den specifikke funktionalitet, som skal bruges. Helt specifikt modtager `performCommand` den information som brugeren indtastede og som ligger i det String Array, der sendes fra brugerfladen. Herefter bliver hvert index i String Array konverteret til den rette datatype, for så at blive sendt videre til *DomainContact* for at blive oprettet som fx en henvendelse. Exception Handling bliver foretaget for at sikre, at datatypen matcher det den skal.

Udover oprettelse af henvendelser og sager fra brugergrænsefladen er der også blevet implementeret en søge funktion. Denne funktion er placeret øverst på venstre side, som vist under design afsnittet. Funktionen fungerer ved, at man indtaster et CPR nummer, som er registreret i CPR-registeret. Herefter vil der blive vist en liste over de henvendelser og sager, som findes på den borger, som har det angivne CPR nummer.

FXMLDocumentController.java (`handleSøg`)** Se koden for fuld implementering

```
@FXML
private void handleSøg(KeyEvent event) {
    dropDown.getItems().removeAll(dropDown.getItems());
    if (event.getCode().equals(KeyCode.ENTER)) {
        if (!ScrollTest.getContent().equals(vboxSøg)) {
            ScrollTest.setContent(vboxSøg);
        }
        this.icb = DomainContact.getInstance().getCaseObject(søgSide.getText()); // returnerer cases og
        caserequests hvis sagsbehandler, caserequests hvis sekretær
        System.out.println("Størrelsen på ICB: " + icb.size());
        for (ICaseObject c : icb) {

            dropDown.getItems().add(c.getType() + ": " + c.getDateCreated() + ": " + c.getDesc());
            dropDown.setValue(c.getType() + " " + c.getDateCreated() + " " + c.getDesc());
        }
        oprettetAf.setText(DomainContact.getInstance().getEmployee(c1.getEmployeeID()));
        søgSide.clear();
    }
}
```

Figur 3.5.14: CommandConverter klassen med injectDomainContact metoden.

Event handleren til at håndtere søge funktionen, starter når der bliver registreret et tastetryk, hvor tasten som er trykket er lig med "Enter" knappen. Metoden vil nu krydstjekke, om man allerede er i "vboxSøg" scenen, ellers skal scenen skiftes til denne scene. Controller klassen indeholder en liste (`icb`), som indeholder *ICaseObjects*. Det er denne liste som vil blive vist i en dropdown menu i søg scenen. Informationerne bliver hentet ind i listen ved linje 6, hvor metoden `getCaseObjects` kaldes. CPR nummeret kan findes i tekstfeltet `søgSide` og bliver hentet med metoden `getText()` og brugt som argument.

Til sidst i metoden, tilføjes informationerne til listen `dropDown` i brugergrænsefladen hvortil beskrivelse- og oprettelses informationer bliver sat. Det skal nævnes at nogle af felterne i kodeeksemplet er navngivet med dansk, pga. navngivning i javaFX dokumentet. Det er gjort bevidst, for at kunne holde styr på navnene på formularen som man udfylder.

Figur 3.5.14 viser injectDomainContact metoden, som skaber forbindelsen til domænelaget. Herved kan der kaldes metoder fra DomainContact klassen, uden at skulle hente en reference til objektet hver gang information skal sendes.

CommandConverter.java (injectDomainContact)** Se koden for fuld implementering
<pre>public void injectDomainContact(IDomainContact domainContact) { this.domainContact = domainContact; }</pre>

Figur 3.5.14: CommandConverter klassen med injectDomainContact metoden.

3.6 Test

Da der er arbejdet med udgangspunkt i brugsmønstrene, er testene definerede af interaktionsdiagrammerne. Det vil sige, at der forventes, at når man eksekverer kode svarende til et brugsmønster og derefter kan følge processen gennem det tilsvarende interaktionsdiagram.

Eftersom der i 1. iteration er arbejdet uden grafisk brugerflade, vil alle tests være tekstbaserede. Tests er blevet foretaget ved at printe til konsollen i udviklingsmiljøet. Hver gang en fejl er opstået bliver der indsat linjer med print-kommandoer i koden for at se, hvor langt programmet når, inden fejlen opstår. En anden metode, som også er blevet brugt, foregår ved at indsætte break points i koden, hvor man kan se de forskellige værdier, mens man fejlfinder koden. Ved eksekvering af hver kommando bliver der skrevet til en tekstfil. Derfor består den endelige test i at se om den information, som er blevet indtastet, også står i tekstfilerne i persistenslaget.

I 2. iteration blev der implementeret en grafisk brugerflade og en database, hvilket også har haft en betydning for den måde, der er blevet testet på. Mange af de tests der er blevet udført, er foregået på samme måde som i 1. iteration. Nu kunne der dog også opstå fejl i brugergrænsefladen, hvilket ikke nødvendigvis var forårsaget af de nedre lag. Mange af de test, der er blevet lavet i anden iteration, handlede om at sikre sig at den information der kom ind i brugergrænsefladen, var det samme, som der blev sendt ned gennem lagene. Dette blev gjort med de samme metoder som i 1. iteration og ved at dobbelttjekke, at der blev hentet data fra de rigtige FXML variabler.

3.7 Konklusion på elaboration

I elaborationsfasen er der blevet udarbejdet flere forskellige modeller til at beskrive systemet, foruden implementeringen af funktionel kode til systemet. Modellerne har hjulpet med at give overblik over systemet, samt hvilke klasser, der skal kunne kommunikere med hinanden for at opfylde de krav der er blevet arbejdet med. Ydermere er arbejdsmetoden Scrum blevet anvendt, som har hjulpet med at strukturere arbejdet iterativt og løse problemstillingerne for fasen løbende.

Der er først blevet udarbejdet et analyseklassediagram på baggrund af brugsmønstrene og domænemodellen, som blev konstrueret under sidste fase. Brugsmønsterrealiseringen gjorde det muligt at lave første udkast af designklassediagrammet og dermed påbegynde kodning af det konkrete system.

Forskellen mellem 1. og 2. iteration ligger i detaljering af analyse- og designmodeller, samt implementeringen af persistens- og præsentationslaget. I første iteration var modellerne meget overordnede, da modellerne skulle fungere som et grundlag, for at kunne detaljere dem yderligere under design workflowet.

Ved skiftet mellem de to iterationer har trelagsarkitekturen gjort det muligt at arbejde på persistens og præsentationslaget uafhængigt af domænelaget. Dette medførte at der opstod mindre konflikter og en mere flydende arbejdsform, hvilket har reduceret arbejdstiden for skiftet mellem tekstbaseret- og grafisk brugergrænseflade.

4 Evaluering

I dette semesterprojekt er der blevet arbejdet med Unified Process (UP) kombineret med Scrum. Projektet er blevet udarbejdet i fastlagte rammer, idet der forinden påbegyndelse af projektet i forvejen var planlagt specifikke faser og forløb for projektet. Disse faser var følgende: idefasen (projektetablering), inception, elaboration, færdiggørelse samt evaluering og refleksion. Herunder var forløbene inception- og elaborationsfase beskrevet i UP, da der i projektet kun blev valgt at arbejde med en letvægtsudgave af UP. Dette har ført til mange fordele men også visse ulemper i projektarbejdet.

Første fase under projektetableringen handlede primært om at danne en ny projektgruppe og om at forstå den udleverede projektcase. Processen med at forme den nye gruppe gik ubesværet hen, men foregik også udenom den angivne procedure, da gruppen fandt det omstændigt. Fra begyndelsen har gruppearbejdet fungeret godt, og der har været en god balance mellem arbejde på projektet og socialt samvær.

Der har været mange positive sider ved arbejdet med semesterprojektet, og det kan her nævnes at projektet havde fastlagte rammer, hvilket gjorde at projektet blev overskueligt og gav gruppen et godt overblik. Her kan fremhæves den kronologiske fremgang, som var let at følge og dermed også at planlægge. Dette har resulteret i at tidsplanlægningen via Scrum er blevet mere præcis, selv med en underestimering af arbejdstimer for specielt GUI.

Ydermere er gruppens personlige mening om at arbejde med både UP og Scrum hovedsageligt positivt. Særligt efter flere virksomhedsbesøg, hvor det blev tydeligt at særligt Scrum er en arbejdsproces, som mange virksomheder anvender. Desuden har opdelingen af projektet i faser fra UP været med til at sikre, at fokusområdet er blevet placeret på det rette sted, således at fx selve kodningen af systemet ikke blev påbegyndt, før der var udarbejdet et tilstrækkeligt antal modeller, såsom design- og analysediagrammer.

Udover Scrum og UP er der også blevet arbejdet med Extreme Programming (XP), hvor arbejdsmetoden "Pair Programming" er blevet anvendt. Netop denne arbejdsmetode er super god, eftersom man ved at programmere i par højner læringsniveauet. Derudover sikrer det også bedre kode, da antallet af fejl bliver reduceret når man er to om at løse en problemstilling.

Der er dog også negative sider ved arbejde med semesterprojektet. Her kan de fastlagte rammer i projektet igen nævnes, idet visse faser har krævet megen tid og ressourcer. I de første faser af projektet har det fyldt meget at følge projektrammeplanen, og dette har til tider holdt projektarbejdet tilbage, hvis projektgruppen var kommet lidt forud. Dette har særligt været et problem, da der var afsat en bestemt mængde tid til hver fase, hvilket betød, at selvom gruppen havde færdiggjort en fase, kunne der ikke fortsættes til næste fase,

eftersom tiden ikke var gået endnu. Alt dette har medført, at der i starten blev brugt markant længere tid på nogle faser i forhold til, hvad gruppen anså nødvendigt.

I forhold til Scrum er den fulde version af denne arbejdsform ikke blevet anvendt, da dette ikke har været muligt. Det kan blandt andet nævnes, at der ikke har været nogen i gruppen, som har kunnet agere "product owner", fordi ingen af gruppemedlemmerne har ejet produktet eller haft særlig kontakt til virksomheden. Dette er dog blevet afhjulpet med 2 virksomhedsmøder med EG Team Online, som står for produktet, selvom det fordelagtigt godt kunne have været et mere tæt forhold og samarbejde med kunden.

Overordnet set har det dog været positivt at arbejde med både Scrum og UP, da det har givet en god fornemmelse af udviklingen af et virkelighedsnært system og projekt.

Under det afgørende forløb Elaboration, hvor implementeringen til et kørende system blev udarbejdet, stødte gruppen dog på et problem. Tidligt i forløbet kom en fælles beslutning omkring at kode på engelsk, da dét var hvad alle følte naturligt, og det der blev lært til programmering. Dog indså gruppen mellem første og anden iteration af elaboration, at det ville være bedst at kode på dansk i dele af præsentationslaget, da projektet omhandler et dansk sagsbehandlingssystem. Så fra første iteration, hvor den tekstbaserede brugergrænseflade var på engelsk, gik man over til i anden iteration en grafisk brugergrænseflad, hvor delene omhandlende referencer til brugergrænsefladen blev implementeret på dansk.

Gruppen har under projektudviklingen haft mange ideer til, hvordan systemet ville kunne videreudvikles og vedligeholdes. Havde gruppen haft længere tid til at udvikle, ville der blive lavet et nyt vindue, når man arbejder med en sag, således at man kan se den påliggende henvendelse sideløbende med sagen. Dette vil gøre sagsarbejdet mere overskueligt, da brugeren lettere ville kunne overskue sagens omfang. På grund af tidsmangel, nåede gruppen heller ikke at rette søge funktionaliteten, hvor der kunne vises hvilken borger, som henvendelsen eller sagen handlede om. Det kunne også have været rart for brugeren, hvis han/hun fik visuelt feedback, som fortalte, om handlingen var vellykket eller der gik noget galt.

Derudover ville gruppen gerne have lavet forskellige ændringer i databasen. For eksempel ville det være mere optimalt hvis Employee tabellen i databasen havde brugernavn som nøgle. Hermed ville brugere ikke kunne have samme brugernavn.

5 Konklusion

Ved projektets begyndelse blev der udleveret en case omhandlende systemet, Sensum Udred, fra EG Team Online. Denne case skulle danne grundlag for et løsningsforslag, hvor gruppen undersøgte, hvorvidt det kunne betale sig for virksomheden at udvikle sagsbehandlingssystemet for handicappede og udsatte voksne.

I inceptionsfasen blev der afleveret både en udførlig forretningsmodel og en kravspecifikation til det afgrænsede system. Disse blev udarbejdet med information fra virksomhedsmøder med EG Team Online. Ud fra analyse af disse modeller og kravspecifikationer blev det konkluderet at det ville være profitabelt for virksomheden at arbejde videre med Sensum Udred.

Elaborationsfasen er blevet afviklet gennem to iterationer. I den første iteration blev fundamentet for systemet dannet, hvori de arkitektoniske lag blev separeret ved lagdeling af systemet. Det meste af domænelaget blev også udviklet her, hvor præsentations- og persistenslaget var en forsimplet udgave af det endelige system, med en tekstbaseret brugerflade og lagring i tekstfiler. Anden iteration havde mere fokus på de to andre lag. Her blev der udviklet en relationel database samt en grafisk brugergrænseflade.

Begge iterationer har været styret af Scrum, hvor der blev taget inspiration fra Extreme Programming til at udvikle i par. Gennem elaborationen har der også været stort fokus på planlægning og tidsestimering af arbejdsopgaver, hvilket ledte til et mere flydende sprint, hvor gruppen udviklede iterativt og objektorienteret på projektet.

Igennem projektforsløbet blev der anvendt en letvægtsudgave af UP, som har givet gruppen indblik i, hvordan der arbejdes med organisatorisk softwareudvikling. Inden for det socialpædagogiske område, er der blevet arbejdet med voksenudredningsmetoden, som understøttede udarbejdelsen af systemets krav og design. Udviklingen har været brugercentreret, hvilket har medført, at systemet afspejler det tidligere anvendte papir system. Dette har gjort at systemet er genkendeligt, hvilket gør det mere intuitivt for de ansatte at benytte.

På baggrund af resultaterne for den udviklede prototype, kan det konkluderes, at systemet lever op til digitaliseringen af voksenudredningsmetoden, og fungerer i henhold til at gøre kommunernes arbejdsgang nemmere. Derudover lever den også op til de sikkerhedsmæssige krav, blandt andet persondataforordningen, da der tages hånd om alt persondata. Det er også påkrævet at brugerne logger ind i systemet og dermed kun får adgang til den funktionalitet, som de har tilladelse til at bruge. Der tages højde for inaktive brugere, som af sikkerhedsmæssige årsager vil blive logger ud.

Der kan ud fra ovenstående, konkluderes at videreudvikling af det resterende system er muligt, og en god investering for EG Team Online.

6 Litteraturliste

00 Takeoff: Forventningsafstemning og gruppedannelse. Udgivet af Lone Borgersen. Internetadresse: https://docs.google.com/document/d/1BqzIXCvTK5k4B2YAd1_gvcyffNuerXxwHdZTpze5ZDI/edit - Besøgt d. 28.05.2018 (Internet)

01 Projektetablering. Udgivet af Lone Borgersen. Internetadresse: <https://docs.google.com/document/d/1AJYdiit9IsCetQyacnTDQ1JqtrzXfh-oPbsRsP1L8Bw/edit> - Besøgt d. 28.05.2018 (Internet)

02 Inceptionsfasen. Udgivet af Lone Borgersen. Internetadresse: <https://docs.google.com/document/d/15dTfnH5YqGPnBI8EhAm0raiRtLdFuQMohd7auYWS344/edit> - Besøgt d. 28.05.2018 (Internet)

03 Elaborationsfasen. Udgivet af Lone Borgersen. Internetadresse: <https://docs.google.com/document/d/19bw3iOMcfhskI6jL-qoqRvDD7o3ELXO9PvZeo6p3QcY/edit> - Besøgt d. 28.05.2018 (Internet)

Acquaintance. Udgivet af Maciaszek + Liong. Internetadresse: <https://drive.google.com/file/d/1gktHhYQmmpl-Jmv0vImclrAuwZE6wrl4/view> - Besøgt d. 28.05.2018 (Internet)

Agile in a Flash. Udgivet af Tim Ottinger & Jeff Langr. Internetadresse: <http://agileinaflash.blogspot.dk/2009/04/furps.html> - Besøgt d. 28.05.2018 (Internet)

Business Model Canvas og Alexander Osterwalder. Udgivet af Jan Bendtsen. Internetadresse: <http://forretningsmodellen.dk/2012/10/business-model-canvas-og-alexander-osterwalde/> - Besøgt d. 28.05.2018 (Internet)

Digitalisering af handicap og udsatte voksne. Udgivet af Socialstyrelsen. Internetadresse: <https://socialstyrelsen.dk/tvaergaende-omrader/sagsbehandling/voksenudredningsmetoden> - Besøgt d. 28.05.2018 (Internet)

Eg Team Online logo. Udgivet af EG Team Online. Set: <https://bit.ly/2kgUk58> - d. 28.05.2018 (Billede)

FURPS+. Udgivet af Tim Ottinger & Jeff Langr. Internetadresse: <http://agileinaflash.blogspot.dk/2009/04/furps.html> - Besøgt d. 28.05.2018 (Internet)

Inceptionsdokument, checkliste. Udgivet af Lone Borgersen. Internetadresse: <https://docs.google.com/document/d/1ktLSuOYURHfrKI2HGcxCblzcci4zFDvLD30BDxY41vk/edit> - Besøgt d. 28.05.2018 (Internet)

Inceptionsdokument, eksempel. Udgivet af Lone Borgersen. Internetadresse: https://docs.google.com/document/d/1eg9Y-hHakmerF15Q8-l_J0NGPRPIKhZnq5iYmSX9fyl/edit - Besøgt d. 28.05.2018 (Internet)

Inceptionsfasen. Udgivet af Lone Borgersen. Internetadresse: <https://docs.google.com/document/d/15dTfnH5YqGPnBI8EhAm0raiRtLdFuQMohd7auYWS344/edit> - Besøgt d. 28.05.2018 (Internet)

Iterative, Evolutionary, and Agile. Larman, Craig: I: Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. 3. udg. Prentice Hall, 2005. Internetadresse: http://www.craiglarman.com/wiki/downloads/applying_uml/larman-ch2-applying-agile-iterative-evolutionary.pdf - Besøgt d. 28.05.2018

MoSCoW prioriteringsmetode. Fundet d. 28.05.2018 på:
https://en.wikipedia.org/wiki/MoSCoW_method

Projektaflevering, checkliste. Udgivet af Lone Borgersen. Internetadresse:
<https://docs.google.com/document/d/1rsWI1OkLQZf8IHPlONqWwPdMzb2rkKo4xVnVwGaAUvU/edit> -
Besøgt d. 28.05.2018 (Internet)

Projektcase. Udgivet af Lone Borgersen. Internetadresse:
https://docs.google.com/document/d/1dClqpUKcdwUbBVU1hBGsJkT5CHpCzSpzbK_tteK4vfY/edit -
Besøgt d. 28.05.2018 (Internet)

Projektforslag, checkliste. Udgivet af Lone Borgersen. Internetadresse:
https://docs.google.com/document/d/1JdDi1XLOEnpg5LluOHJRc0hGj7wtDG3oDZT_Y3bXNJl/edit -
Besøgt d. 28.05.2018 (Internet)

Projekthåndbog. Udgivet af Lone Borgersen. Internetadresse:
https://docs.google.com/document/d/1G8rAjazbWZDzpAEsm2lb30yandwzvOZxJoFrc_F9PgY/edit -
Besøgt d. 28.05.2018 (Internet)

Projektrammeplan. Udgivet af Lone Borgersen. Internetadresse:
https://docs.google.com/document/d/1ckwwShOKKYt4x4Gjb79D4-sQ_OgBtVosRksFBAnGTw/edit -
Besøgt d. 28.05.2018 (Internet)

Voksenudredningsmetode. Udgivet af Socialstyrelsen. Internetadresse:
<https://socialstyrelsen.dk/filer/tvaergaende/sagsbehandling-og-organisering/link-metodehandbog-vum-1.pdf> - Besøgt d. 28.05.2018 (Internet)

Scrum guide. Udgivet af Ken Schwaber and Jeff Sutherland. Internetadresse:
<http://www.scrumguides.org> - Besøgt d. 28.05.2018 (Internet)

Software Engineering, 9. edition. Udgivet af Ian Sommerville. 2016. (Bog)

UML 2 And the Unified Process. Udgivet af Lone Borgersen. 2011. Internetadresse:
<https://drive.google.com/file/d/0B3ME4awmGS9ac2tReIJCZ0Z0LWM/view> - Besøgt d. 28.05.2018
(Internet)

B: Kildekode

Se vedlagt zip mappe for kode. Denne kode ligger i mappen 'src'.

C: Projektforslag

Titel

Sensum Udred - Et sagsbehandlingssystem til handicappede og udsatte voksne.

Deltagere

Gruppe 02

Navn	E-mail
Erik Bjørnholt Nielsen	ernie17@student.sdu.dk
Loc Hoang Thanh Nguyen	longu17@student.sdu.dk
Sigurd Eugen Espersen	siesp17@student.sdu.dk
Peter Stærdahl Andersen	pande15@student.sdu.dk
Anders Bensen Ottsen	anott17@student.sdu.dk
Josef Ngoc Can Pham	jopha15@student.sdu.dk

Tabel 1.1: Medlemmer af gruppe 02.

Vejleder

Navn	E-mail
Jan Emil Larsen	jel@lit.dk

Tabel 1.2: Vejleder for projektet

Introduktion

KMD blev solgt tilbage i 2008, hvilket gav mulighed for at private firmaer kunne bevæge sig ind på den offentlige sektors marked. Dette resulterede i en frigørelse for kommunerne fra KMD's monopol på it-løsninger, på områder som kontanthjælp, sygedagpenge og sagsoverblik. Virksomheden EG Team Online påtænker sig i den forbindelse at lave et sagsbehandlingssystem (Sensum Udred) til handicappede og udsatte voksne. De har tidligere udviklet et system, Sensum Bosted, der fokuserede på patientbehandling. Med den udleverede case fra virksomheden vil de vide, hvorvidt det kan betale sig at lave systemet, og hvilke krav der bliver stillet til systemet. Der vil blandt andet blive kigget på, hvordan dataen bliver beskyttet på grund af den nye databeskyttelsesforordning, som blev vedtaget d. 27. april 2016.¹⁷

Problemstilling

I dag bliver der brugt mange ressourcer på kommunikation på det socialpædagogiske område. Det sker bl.a., fordi der skal rapporteres frem og tilbage igennem systemet fra

¹⁷ Projektcase af EG Team Online:

https://docs.google.com/document/d/1dClqpUKcdwUbBVU1hBGsJkT5CHpCzSpzbK_tteK4vfY/edit

kommuner til de specifikke afdelinger. Kan kommunikation mellem sagsbehandlere og patient-behandlere måske optimeres vha. et elektronisk system?

Idéen med et sådant system er, at det skal skabe et overblik over sagsbehandlingen. Det er vigtigt, at systemet understøtter det arbejdsflow, der er i et sagsforløb, da et sagsforløb består af flere forskellige aktiviteter. Disse aktiviteter er bl.a.: sagsåbning, sagsoplysning, sagsvurdering, afgørelse og sagsopfølgning. Systemet skal overholde kravene i den nye persondataforordning og det vil derfor være essentielt at have fokus på sikkerhed omkring persondata - omkring hvem, hvad og hvordan disse følsomme data bliver tilgået.

Dette projekt vil undersøge om det kan betale sig for EG Team Online, at lave et sagsbehandlingssystem for handicappede og udsatte voksne. Der vil blive udarbejdet en forundersøgelse hvor de overordnede krav vil blive udspecificeret. Derudover vil der også blive lavet en afgrænsning af systemet hvori de vigtigste krav vil blive analyseret, designet og implementeret.

Problemformulering

Kan det betale sig for EG Team Online at lave et sagsbehandlingssystem til at holde styr på sager for handicappede og udsatte voksne? Dette vil blive undersøgt ved at lave en prototype på systemet, hvor der vil være fokus på sagsåbning.

Formål og mål

Formålet med projektet er at anvende de forskellige færdigheder vi har lært fra de faglige kurser og bruge disse fagligheder til at udarbejde et gennemtænkt system. Derudover vil formålet også være at lave et system som overholder alle specifikationer og krav, som EG Team Online har defineret. Dette vil gøres med fokus på modulet omhandlende sagsåbning, som er en del af voksenudredningsmetoden.¹⁸ Med dette afgrænsede system vil de få et bud på hvordan en eventuel løsning kan tage form.

Kompetencemålene for projektet er at kunne anvende de forskellige fagligheder. Der vil blive anvendt organisatoriske værktøjer til at udarbejde en business case og skabe et beslutningsgrundlag. Ydermere vil der blive lavet en kravspecifikation med udgangspunkt i en brugsmønster model, hvor der vil blive arbejdet iterativt og objektorienteret med udgangspunkt i Unified Process. Relational database udvikling og objektorienteret programmering skal hjælpe os til at udarbejde et system, som løser vores problemstilling.

¹⁸ Voksenudredningsmetode:

<https://socialstyrelsen.dk/filer/tvaergaende/sagsbehandling-og-organisering/link-metodehandbog-vum-1.pdf>

Motivation

Vi vil i projektet gerne lære at arbejde struktureret og inkrementelt med UP. Samtidig vil vi arbejde med at omdanne kravspecifikationen til et færdigudarbejdet system med tilhørende database. Fagligt vil vi gerne udvikle os og opnå en bedre forståelse for software som helhed samt en organisations opbygning og betydning for et projekt.

Det vil være spændende at arbejde med et system, som afspejler en virkelighedsnær problemstilling. Det motiverer os især, at arbejde med et system, som skal hjælpe folk på det socialpædagogiske område.

Forventede resultater

Vi forventer at arbejde i en iterativ og objektorienteret proces - for derved at lave et afgrænset system på det rette beslutningsgrundlag samt en tilhørende, detaljeret dokumentation over det færdige produkt og udviklingsprocessen. Vi forventer at udarbejde et produkt, indeholdende en rapport samt et funktionelt system omkring sagsåbning der afspejler voksenuddannelsesmetoden. Dette skulle gerne resultere i et virkelighedsnært projekt som vil give os en bedre forståelse for, hvorledes en softwareingeniør arbejder.

Foreløbig tidsplan¹⁹

Uge 9: 26/2	Aflevering af projektforslag til godkendelse
Uge 10: 5/3	Spørgsmål til- og tilhørende svar fra EG Team Online
Uge 12: 19/3	Aflevering af inceptionsdokument til review
Uge 13: 28/3	Aflevering af inceptionsdokument til bedømmelse
Uge 13-18:	Elaborationfasen, projektarbejde + virksomhedsbesøg
Uge 18: 4/5	Aflevering af udkast til projektaflevering
Uge 22: 31/5	Aflevering af færdigt projektrapport

¹⁹ Projektrammeplan:

https://docs.google.com/document/d/1ckwwShOKKYt4x4Gjb79D4-sQ_OgBtVosRksFBAnGTw/edit

Bilag 1: Samarbejdsaftale

Vores forventninger til hinanden.

- Vi forventer at vi hjælper hinanden, og er åbne omkring vores tvivl. Vi stiller spørgsmål, hvis der er noget vi ikke har forstået og deltager aktivt
- Man melder ud i god tid, hvis man ikke kan deltage. 24 timer før som minimum.
- Der er en dagsorden til hver gang vi mødes, således vi undgår at spilde tiden, når vi er samlet.
- Der skal være plads til at have det sjovt og skabe en god stemning.
- Der mødes også for at styrke det sociale.

Hvordan og hvad vil vi samarbejde om?

- Primært arbejde på semesterprojektet.
- Generel brug af studiegruppen i diverse fag.

Hvor, hvornår og hvor tit vil vi mødes?

- Vi mødes som udgangspunkt tirsdag klokken 10.00 til 13.00, onsdag klokken 10.00 til 14.00 og fredag 09.15 til 14.00 i projektrummet.
- Vi er forberedt på at mødes udenfor dette tidsrum også.

Hvad skal vores mål være for semesterprojektet?

- Vi vil opnå en bedre forståelse for programmering, databaser, software engineering og organisation og ledelse.
- Vi vil lave et godt projekt, vi alle kan være stolte af og sige, vi hver især har taget del i.
- Der skal være rapportskrivning på et godt niveau.

Hvordan vil vi opnå dem?

- Arbejde struktureret og være aktiv i processen.
- Være en hjælp for vores medstuderende.

Hvad er vores styrker og svagheder (ift. gruppearbejde)?

- Anders
 - **Styrker:** Meget struktureret og punktlig
 - **Svagheder:** Har svært ved at komme videre, hvis man sidder fast
- Loc
 - **Styrker:** God til at arbejde i par, kontra individuelt
 - **Svagheder:** Det kan være en udfordring at arbejde med ting, som er nye
- Peter
 - **Styrker:** God til at arbejde i mindre grupper
 - **Svagheder:** Dårlig til at arbejde med uoverskuelige opgaver
- Erik
 - **Styrker:** Kan godt lide at arbejde individuelt, men gruppearbejde er også fint
 - **Svagheder:** Dårlig til at arbejde med opgaver der ikke er meget veldefinerede og bliver meget let distraheret af uorden
- Josef
 - **Styrker:** Fleksibel - kan arbejde alle timer i døgnet
 - **Svagheder:** Bliver let distraheret ved støj
- Sigurd
 - **Styrker:** Arbejder indtil stoffet er gennemlært. God til at arbejde i par

- **Svagheder:** Bliver let distraheret. Tager et stykke tid før stoffet sidder helt fast

Hvad gør vi hvis et gruppemedlem ikke lever op til forventningerne/overholder kontrakten (f.eks. ikke møder op eller deltager)?

- Undersøge grundlaget for mangel - ikke se passivt til, hvis der er et problem.
- Vi rækker ud for at hjælpe hinanden.
- En omgang drikkevare til gruppen, hvis man ikke overholder gruppens regler.
- Notere problemerne, så vi kan tage det løbende.

Bilag 2: Vejlederaftale

Vejledningen vil primært komme til at foregå på vejledningsmøder, men også gennem fælles arrangementer som fx reviews og seminarer. Som hovedregel er der et fast vejledningsmøde hver uge, bortset fra de uger, hvor der er andre aktiviteter.

Vejledningens form

Der vil blive vejledt både mht til projektet konkret og gruppearbejdet. Der vil løbene blive set på mål, plan og status for projektet. Vejledningen vil støtte gruppen i

- at sætte mål for projektarbejdet
- at planlægge hvordan målene realiseres
- at bruge planen til at arbejde hen mod målene

Dagsorden og referater for vejledningsmøder

Dagsordenen for hvert enkelt møde med vejlederen vil tage udgangspunkt i følgende punkter.

1. Hvem er ordstyrer og referent for dette møde?
2. Godkendelse af dagsorden
3. Endelig godkendelse af referat
4. Emne, kapitel eller anden indikation af hvor gruppen er i fagkurserne. Ganske kort
5. Hvad har hver enkelt arbejdet på/sammen med siden sidst? Helt kort
6. Særlige (konkrete) emner, fx aftalt ved forrige møde
7. Er gruppens arbejde så lang som det burde være i følge planen. Er der noget vejleder skal hjælpe med
8. Opfølgning på konkrete aktiviteter ift. Mål og plan – og evt. uden for
9. Kommende aktiviteter ift. Mål og plan – og evt. uden for
10. Eventuelt (ingen beslutninger)

Det forventes at hele gruppen deltager og de sørger for en ordstyrer og referent.

Referatet skal tydeliggøre hvad der blev besluttet, og hvad der er særligt udfordrende, og som der skal arbejdes med og nok på dagsorden til næste møde.

Ekstra kommentarer

Møderne holdes fortrinsvist fredage og vejleder modtager referat fra seneste møde og dagsorden til næste møde senest tirsdagen før.

Vejleder og gruppe kommunikerer via aftalte platform, fx dokumentdeling på et Google drev, og via mail, altid med cc: hele gruppen, og kun til adviseringer ikke til dokumentudveksling, sagsbehandling o.l.

Vejleder reviewer "frivillige" så vel som obligatoriske artefakter (afleveringer) og yder generelt coachende vejledning. Vejleder vil også interessere sig for gruppedynamikken

Vejleder har en responstid på max 24 timer ("jeg har modtaget ..."), men behandlingstiden vil variere efter omstændigheder og kompleksitet. Udeståender tages op på vejledningsmøde.

Afviselser fra aftaler, herunder manglende fremmøde til vejledningsmøde, meddeles tidligst muligt – og begrundet. Effekt varetages om muligt af gruppen.

D: Inceptionsdokument

Sensum Udred

Inceptionsdokument

Deltagere i gruppe 2

Navn	Email
Sigurd Espersen	siesp17@student.sdu.dk
Josed Pham	jopha15@student.sdu.dk
Loc Nguyen	longu17@student.sdu.dk
Anders Ottsen	anott17@student.sdu.dk
Peter Andersen	pande15@student.sdu.dk
Erik Nielsen	ernie17@student.sdu.dk

Vejleder: Jan Emil Larsen - jel@lit.dk

Projektforløb: 01/02/2018 til 31/05/2018

Afleveringsfrist til review: Mandag 19. marts kl. 16

Afleveringsfrist til bedømmelse: fredag 6. april kl. 16

1 Indledning

Dette projekt omhandler udviklingen af et sagsbehandlingssystem tiltænkt handicappede og udsatte voksne. Projektet er afgrænset til at omhandle de første trin i VUM (Voksenudredningsmetoden) - dvs. et system over sagsåbning. Systemet udvikles i et virksomhedssamarbejde hvor interessenten er virksomheden EG Team Online, samt deres kundekreds som er kommunen.

Inceptionsfasen er en præudviklingsfase - hvilket skal forstås som, at der i denne fase vil blive etableret den nødvendige viden og oplysninger for at komme videre i udviklingen af et system og deraf videre til næste udviklingsfase i projektet. Der vil i denne fase blive afviklet en beskrivelse af virksomheden, samt de forretningsmæssige behov og fordele som virksomheden får ud af ved brug af systemet og systemets indvirkning på virksomheden.

2 Formål og målgruppe

Formålet med denne inceptionsfase er at få udarbejdet et korrekt inceptionsdokument baseret på den nødvendige viden fra bl.a. udvikling af systemet og samarbejde med virksomheden. Med dette inceptionsdokument dannes der et beslutningsgrundlag for, hvorvidt det kan betale sig at fortsætte det videre arbejde med udvikling af systemet.

Målene i denne fase er at:

- udarbejde en forretningsmodel over virksomheden EG Team Online.
- forklare hvilken indvirkning dette system har på virksomheden
- udarbejde en overordnet kravspecifikation over systemet.
- danne et overblik over prioriteringen af kravene
- beslutte arbejdsmetoder for elaborationsfasen
- udføre dagbog/logs i projektforløbet.

Målgruppen i dette projekt er virksomheden EG Team Online. I sidste ende skal ledelsen i EG Team Online samt deres udviklere, vurdere og tage stilling til om projektet er fyldestgørende til deres kriterier og behov - og om det kan betale sig for dem, at eventuelt implementere og viderefører projektet.

3 Fremgangsmåde

Fremgangsmåden for inceptionsfasen vil først og fremmest tage udgangspunkt i en systemafgrænsning. Her skal det overordnede system med alle dets komponenter begrænses til et mere håndgribeligt system, som er mere overskueligt i forhold til den begrænsede tidsperiode, der er tilgængelig for projektet. Dette har vi gjort ved at lave et brugsmønster model, som afspejler det afgrænsede system.

Et vigtigt udgangspunkt for fremgangsmåden er, at der vil blive brugt en letvægtsudgave af UP(Unified Process), som danner fundamentet for projektet og dets rammer og faser - inception, elaboration, færdiggørelse og evaluering/refleksion.

Der vil blive holdt et diskussionsmøde med virksomheden, hvor spørgsmål til systemet og projektet vil blive besvaret af EG Team Online. Virksomhedsmødet vil danne grundlag for den forretningsmodel, hvor der anvendes Osterwalders forretningsmodel²⁰.

Forretningsmodellen vil have en beskrivelse af virksomheden - en beskrivelse af de forretningsmæssige behov og fordele, og systemets indvirkning på virksomheden.

Systemet vil blive beskrevet i sit miljø med en systemlandskabsmodel. Hertil vil der også blive beskrevet alternative løsninger fra konkurrenter på markedet.

Med udgangspunkt i Voksenudredningsmetoden vil der blive udformet en overordnet kravspecifikation. Det store overblik vil blive dannet med en overordnet brugsmønstermodel, som danner grundlag for de brugsmønstre, der efterfølgende vil blive uddybet. Dette leder til en mere dybdegående analyse af kravene, hvor metoden FURPS+ også vil blive taget i brug. Kravspecifikationen vil blive rundet af med en domænemodel, som helt overordnet skal beskrive de forskellige klasser og objekter, som skal være i systemet.

Som afrunding på dokumentet vil der gøres brug af MoSCoW til at prioriterer de tidligere nævnte krav. Efterfølgende vil metoderne for elaborationsfasen samt ressourcerne blive beskrevet.

4 Business case

Dette afsnit beskriver den forretningsmodel som er blevet udarbejdet på baggrund af mødet med virksomheden EG Team Online og den generelle information som er blevet udleveret fra projektcasen.

4.1 Beskrivelse af EG Team Online

Team Online blev tilbage i 2014 opkøbt, og blev da en del af koncernen EG A/S, der er en af Skandinaviens førende it-virksomheder. Virksomheden EG har omtrent 2000 medarbejder og operere i 26 lokationer i Danmark, Sverige og Norge.

EG Team Online laver softwaresystemer og services for både private virksomheder og det offentlige med speciale i løsninger til social- og sundhedssektoren. Virksomheden skaber værdi for kunderne, ved at deres systemer er af høj kvalitet og er meget brugercentrerede.

Selvom at EG Team Online er et mindre selskab med under 100 ansatte, har de tidligere lavet lignende systemer som har været en succes.

4.2 Forretningsmæssige behov og fordele ved systemet

EG Team Onlines behov er at optimere og effektivisere sagsbehandlingsprocessen i kommunerne, dette kan sikre et mere loyal kundegrundlag, som i sidste ende resultere i større indtjening for EG Team Online. De ønsker at forbedre kommunikationen mellem sagsbehandlere og deres patienter. Dette giver også en fordel for alle parter, da det reducere fejl og erstatter tidligere tidskrævende procedurer. Kunden opnår også nogle miljømæssige fordele, da de kan reducere mængden af papir som skal bruges til at dokumentere sagsbehandlingen.

²⁰ Osterwalders forretningsmodel:

<http://forretningsmodellen.dk/2012/10/business-model-canvas-og-alexander-osterwalde/>

Endvidere er det et behov for kunden, at persondataforordningen bliver overholdt samt at virksomheden skal implementere de nye regler i deres systemer. Dette giver også de fordele, at systemerne bliver mere sikre i håndteringen af persondata.

4.3 Tilsigtede indvirkning på EG Team Online

På baggrund af tidligere analyser af EG Team Online vha. PESTEL, Porters Five Forces og SWOT(Se [Bilag B: Omverdensanalyse](#)) vil vi lave en forretningsmodel med udgangspunkt i Osterwalders forretningsmodel.

De forskellige elementer i modellen bliver beskrevet i en mere sammenhængende form.

4.3.1 Organisation

EG Team Online er en del af det lukkede økosystem som koncernen EG A/S er. Derfor er deres partnere andre dele af selve koncernen. Udover disse er EG også partnere med bl.a. HP, Microsoft, IBM og Infor.

Virksomheden udvikler og leverer brugerecentreret IT-systemer og services med fokus på social- og sundhedssektoren. Kerneaktiviteten for virksomheden er derfor udvikling og vedligeholdelse af software.

EG A/S er en stor koncern - der groft kan deles i to divisioner: software til det offentlige og den sociale sektor - og konsulenttydelser. Under disse to divisioner findes mange projektgrupper, der hver specialiserer sig i en bestemt målgruppe. Der kan forekomme samarbejde mellem flere projektgrupper om fælles mål.

4.3.2 Produkter

Virksomheden sælger brugercentreret IT-løsninger, og skaber værdi ved at have meget erfaring og branchespecifik viden. Endnu en værdifaktor for kunden er, at virksomheden tilretter sig efter persondataforordningen i deres systemer. Ydermere fokuserer virksomheden meget på deres eksisterende kunder, hvilket skaber tillid og loyalitet, som tiltrækker nye kunder, hvilket adskiller EG Team Online fra andre.

4.3.3 Kunderne

EG Team Online har meget fokus på eksisterende kunder - og med deres høje kvalitet af produkter og ydelser, skaber dette en vis tillid og loyalitet hos kunderne. I og med at der bliver taget vare på eksisterende kunder, er forholdet mellem virksomheden og kunden langvarigt, dette åbner også op for en udvikling af kunderelationen over tid. Derudover er der tale om systemer som gerne skulle holde i længere tid, hvilket bidrager til at forholdet mellem kunde og virksomhed vil være langt.

Virksomheden har både kunder i den offentlige og private sektor, hvor de offentlige primære kunder er institutioner, der har at gøre med udsatte voksne og handicappede.

Da EG Team Online tilbyder plug & play kan deres produkt nemt tilpasses kundens behov. Præciseringen af de kunder, som vil få mest værdi ud af dette produkt, vil være sagsbehandlerne som til dagligt arbejder med voksenudredningsmetoden.

EG Team Online finder selv sine kunder fx ved at byde på systemer som er sendt ud til licitation. Det kan også ske at de har konsulenter ude, som snakker med en kunde og derved laver aftaler omkring nye tiltag eller udvidelser af systemer. De har en salgsafdeling, som også tager kontakt til potentielle kunder og forsøger at sælge deres produkter.

4.3.4 Økonomi

EG Team Online er en mindre virksomhed med omtrent 100 ansatte - dermed vil en af omkostningerne være lønninger til de ansatte.

Der er omkostninger i forbindelse med persondatahåndtering, men EG Team Online benytter EG Hosting til at håndtere servere. De arbejder med at automatisere logning af hvem der håndterer persondata. Dette medfører at det vil blive til en "engangsbetaling" frem for løbende omkostninger. Da der er tale om en IT-virksomhed, vil diverse software og værktøjer også være en omkostning fx IDE'er.

De fleste af EG Team Onlines indtægter kommer fra ordrer, der ligger under udbudsgrænsen. Dette kunne være, salg af systemer som har en fastlagt pris eller moduler til allerede eksisterende systemer. Efterfølgende betaler kunderne licenspenge pr bruger som anvender systemet, her vil der blive anvendt kontrakter - ofte varende 1 til 2 år. Ved tilkøb af bestemte moduler til systemet stiger prisen pr bruger.

4.4 Konklusion

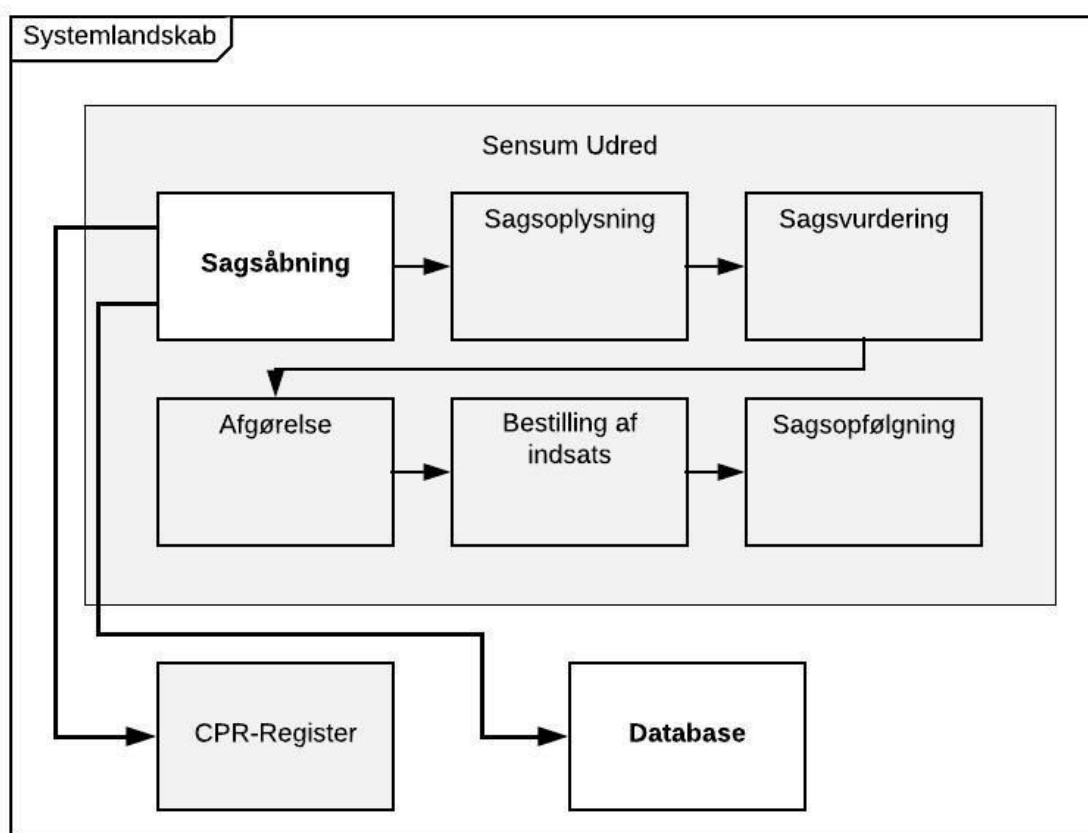
Gennem virksomhedsmøde med EG Team Online og ved brug af Porters Five Forces, PESTEL- og SWOT-analyse(Bilag B: Omverdensanalyse), blev det muligt at udarbejde en gennemført forretningsmodel (Osterwalders forretningsmodel) over virksomheden. Det er blevet vurderet at videreudviklingen af systemet vil være til stor gavn for virksomheden. Udover at virksomheden formentligt vil tjene på systemet vil det også give dem et større markeds omdømme.

5 Forretningsområdet og eksisterende løsninger

Den metode der i dag bruges til at arbejde med VUM (Voksenudredningsmetoden) foregår på papir, den digitale understøttelse af dette hedder DHUV (Digitalisering af handicappede, udsatte og voksne området). Der findes forskellige løsninger på udfordringen med det store papirforbrug samt det vil fremme kommunikationen mellem patienter, pårørende og sagsbehandlere.

Da virksomheden befinder sig på IT-markedet, findes der mange konkurrenter såsom KMD, Systematic og NetCompany. De har alle lignende løsninger med systemer som kan tage sig af DHUV, konkret kan der nævnes KMD Nexus.

5.1 Systemlandskab



Figur 5.1: Denne figur præsenterer vores systemlandskab.

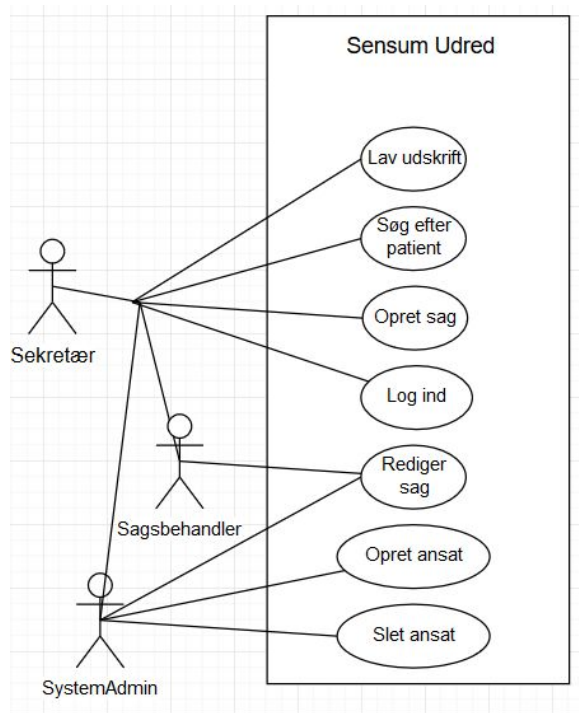
Der vil blive fokuseret på de hvide bokse. De grå bokse beskriver omgivelserne, men vil ikke blive bearbejdet.

6 Overordnet kravspecifikation

Dette afsnit omhandler den overordnede kravspecifikation - hvor der bliver udarbejdet en overordnet brugsmønstermodel, en domænemodel over problemområdet, en beskrivelse af krav samt deres formål.

6.1 Overordnet brugsmønstermodel

Nedenfor ses brugsmønstermodellen, som afspejler det afgrænsede system. Der vil også være tilhørende beskrivelser til de fundne aktører samt brugsmønstrene og deres formål.



Figur 6.1: Denne figur præsenterer vores overordnet brugsmønstermodel.

6.1.1 Beskrivelse af fundne aktører

Navn	Beskrivelse	Mål og tjenester
Sekretær	I større kommuner vil der være sekretærer ansat, som er involveret i de første simple trin af sagsbehandlingen. Sekretæren kan tage mod en henvendelse, og starte en sag. Dog kan sekretæren aldrig gå hele vejen med sagen, men blot lave forarbejdet til en sagsbehandler.	Hjælpe med første trin af sagsbehandling - evt. henvendelse
Sagsbehandler	En sagsbehandler er den person som vil tage sig af en persons forløb i en sag. Det vil som regel være en sagsbehandler der starter en sag (udover i større kommuner, hvor en sekretær vil starte) og så følge den hele vejen til dørs. Det er primært sagsbehandleren som vil sidde med systemet til dagligt, og holde styr på de forskellige sager der er.	At hjælpe en patient gennem en sagsbehandling
SystemAdmin	SystemAdmin er den person der er administrator for systemet. Administratoren vil have flere rettigheder i systemet end de andre ansatte. Administratoren vil være den første person der er oprettet i systemet, og vil stå for at oprette andre brugere i systemet og eventuelt slette disse brugere.	Oprette/slette/redigere ansatte i systemet

Tabel 6.1: Denne tabel præsenterer en beskrivelse over fundne aktører.

6.1.2 Overordnet beskrivelse af brugsmønstre

Nr.	Brugsmønster	Formål
BM1	Lav udskrift	En patient kan få en udskrift af sin sag, så patienten kan se alle de informationer der er i sin sag.
BM2	Søg efter patient	En ansat kan søge efter de patienter der er i systemet, så den ansatte hurtigt kan tilgå og få et overblik over patienter og se de sager der er knyttet til de tilsvarende patienter.
BM3	Opret sag	En ansat kan oprette en sag, så der bliver holdt elektronisk styr på de informationer der vedrører en sag.
BM4	Log ind	Logger ind i systemet med unikt brugernavn og kodeord. Dette giver adgang til systemets

		funktioner og data.
BM5	Rediger sag	Sager kan blive redigeret så informationer i sagen kan ændres til noget nyt.
BM6	Opret ansat	En ansat bliver oprettet af en systemadministrator. Hertil bliver informationer oprettet til den ansatte, såsom kontaktinformationer.
BM7	Slet ansat	En ansat bliver fjernet fra systemet. Dette sikrer at data på en tidligere ansat bliver fjernet fra systemet, således overflødig information bliver fjernet, samt at den tidligere ansatte ikke længere har tilgang til systemet.

Tabel 6.2: Denne tabel præsenterer en beskrivelse af brugsmønstre.

6.1.3 Yderligere krav til løsningen

Udover de krav som vores afgrænsede system har fokus på, er der også de andre krav til systemet i sin helhed. Disse supplerende krav er baserede på voksenudredningsmetoden²¹. Der er i høj grad sat fokus på i vores system er specielt selve sagsåbningen og det at finde sagen igen med oplysningerne. Udover disse egenskaber, er der mange andre krav. Andre krav inkluderer at vurdere en sag, at afgøre en sag, bestille social indsats og at følge op på en sag.

(SK1) Kravet med at vurdere en sag handler om at specificere de indsatser der kan blive gjort og registrerer et eventuelt "tilbud" til den udsatte. Der vil blive sat nogle indsatsmål som så senere skal vurderes.

(SK2) Når en sag skal afgøres registrerer man det endelige "tilbud", udarbejder en begrundelse for afgørelsen og formidler afgørelsen til borgeren.

(SK3) Når den sociale indsats skal bestilles skal sagsbehandleren videreformidle de oplysninger, som er registreret på borgeren, til udføreren (kunne være et bosted eller lignende) som så skal tage sig af borgerens "problem".

(SK4) Når sagsbehandlere skal følge op på en sag får han/hun en opfølgning fra udføreren og skal så vurdere det. Der vil være nogen indsatsmål som tidligere er defineret, hvor både borgeren og udføreren har kommenteret på det. Sagsbehandleren skal så tage stilling til hvordan det går med de indsatsmål.

Alle disse andre forskellige funktioner definerer nogle krav som det fulde system skal kunne, men vi ikke vil fokusere på.

²¹ Voksenudredningsmetoden:

<https://socialstyrelsen.dk/filer/tvaergaende/sagsbehandling-og-organisering/link-metodehandbog-vum-1.pdf>

6.2 Overordnet beskrivelse af supplerende krav

Der er mange supplerende krav, en god måde at kategorisere disse på er ved hjælp af FURPS+ (Functionality, usability, reliability, performance, supportability og constraints)²².

6.2.1 Functionality

Der er krav til sikkerheden i system. Alt data skal være forsvarligt sikret og der skal være en form for autorisation af brugere af systemet f.eks. gennem log ind. En del af sikkerheden vil også bestå i, at alle aktiviteter skal logges. Det vil sige at et sted i systemet skal der opbevares filer, hvor det fx er muligt at se, hvilke ansatte der har været inde og lave ændringer i en specifik sag.

Derudover skal systemet kunne tilgås fra en hjemmeside. Der skal tages højde for kryptering, hvilket gør dataen sværere at tilgå for uvedkommende.

6.2.2 Usability

Det er også vigtigt for systemet at det er brugervenligt, da dette har en stor betydning for systemets succes hos ansatte. Det vil derfor være en vigtig rolle at brugergrænsefladen skal være overskuelig og nem.

6.2.3 Reliability

Systemet skal være pålideligt, da borgere kan henvende sig døgnet rundt. Derudover er det vigtigt at databasen aldrig går ned, så man ikke mister vigtig data. Man kan dog gå ud fra et EG Hosting tager hånd om dette.

6.2.4 Performance

Systemet skal have en hurtig responstid, så brugeren ikke skal bruge lang tid på at vente. Derudover skal systemet gerne være hurtigere end den tidligere papirbaserede løsning.

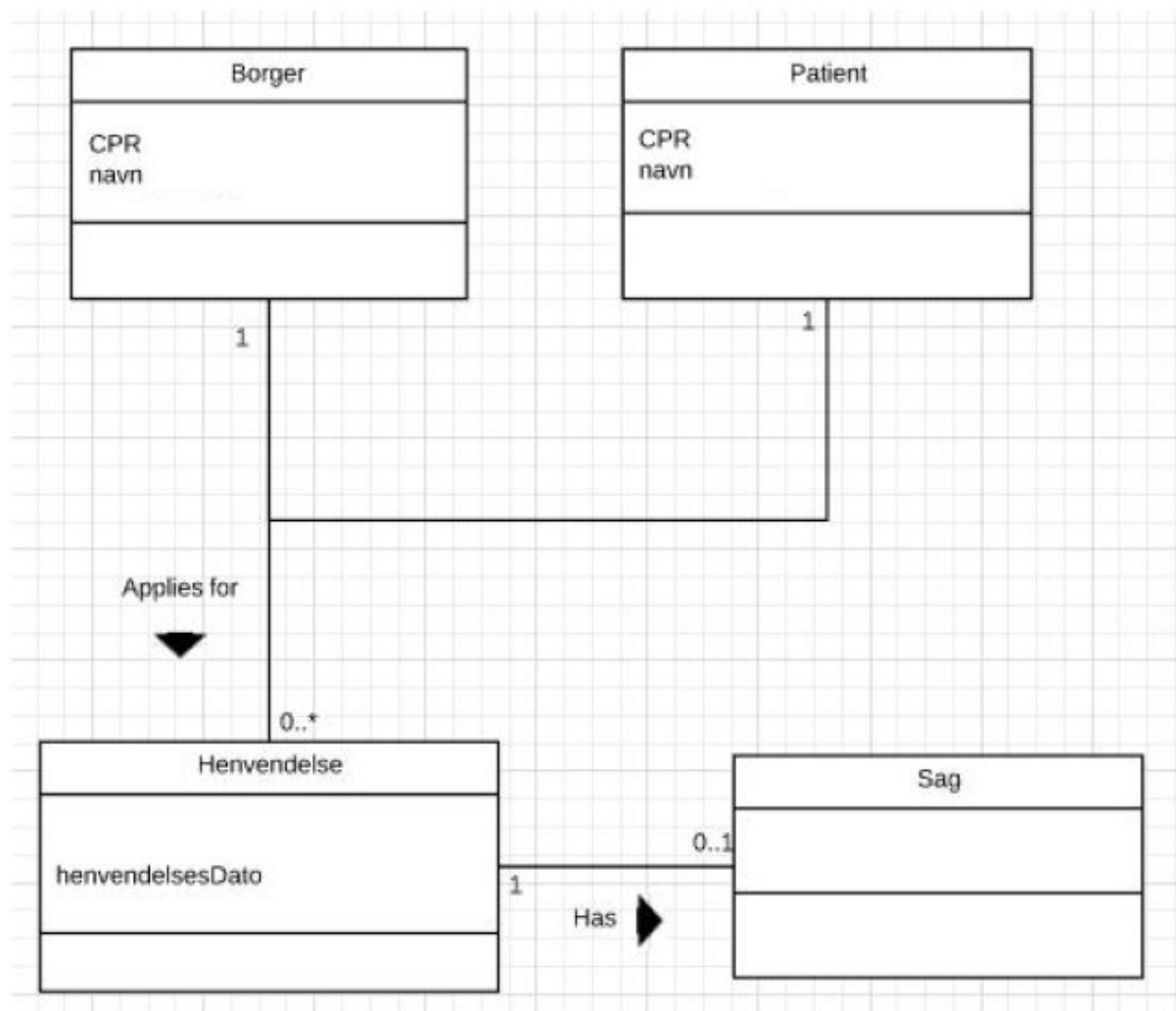
6.2.5 Supportability

Det kan forventes at systemet vil blive brugt i indtil flere år. Dette gør at systemet gerne skal være let at vedligeholde. Derudover skal der anvendes 'plug-&-play', hvilket gør at udvidelser til systemet vil forekomme, og at det derfor skal være sat op til dette.

²² FURPS+: <http://agileinaflash.blogspot.dk/2009/04/furps.html>

6.3 Domænemodel

Denne domænemodel er blevet udarbejdet ud fra det afgrænsede system, der omhandler specifikt omkring sagsåbningen. De klasser som er medtaget i modellen, beskriver derfor selve problemområdet og disse involvere den almene borger og en patient. En borger eller en patient kan lave en til mange henvendelser - hvorimod en henvendelse enten bliver til ingen eller en sag.



Figur 6.2: Denne figur præsenterer vores domænemodel.

De få attributter, som er medtaget i modellen blev udvalgt pga. de var meget beskrivende for deres klasser og gav derfor mening. Attributterne kunne også ses som klassens unikke nøgle - dvs. følgende borgerens eller patientens CPR-nummer.

7 Prioritering

Der vil i dette afsnit blive anvendt MoSCoW-modellen til prioriteringen af krav. Modellen består af følgende fire punkter: Must have, should have, could have, won't have.

MoSCoW prioriteringer af krav	Semantik
Must have	Obligatoriske krav, som er fundamentalt for systemet <ul style="list-style-type: none">• En medarbejder kan tildeles en eller flere adgangsroller som kan konfigureres på et system basis.• Logning skal forekomme ved håndtering af persondata. Lav udskrift, Søg efter patient, Opret sag, Log ind, Slet ansat, Rediger sag, Opret ansat. (Se evt. tabel 6.2 for beskrivelser.)
Should have	Vigtige krav, som kan kunne være udeladt <ul style="list-style-type: none">• En borger kan søge om ExtraNet adgang som giver begrænset læseadgang til sagens detaljer.• Systemet skal gerne være hurtigt, dvs. hurtigere end at skrive på papir.
Could have	Krav som udelukkende er valgfrie At kunne tilgå systemet fra en hjemmeside.
Won't have	Krav som kan vente, til senere implementationer til systemet Kryptere data, som bliver lagret i databasen.

Tabel 7.1: Denne tabel præsenterer vores prioritering ved brug af MoSCoW.

8 Metode i elaborationsfasen

I elaborationsfasen vil der blive gået i dybden med udvælgelsen af krav til projektet. Den overordnede model og arkitektur fra inceptionsfasen vil blive uddybet og dermed mere fastlagt. Problemdomænet vil blive analyseret og den del som skal laves i konstruktionsfasen vil også blive fastlagt.

Der vil selvsagt blive arbejdet med UP (Unified Process) eftersom det er denne arbejdsform som danner rammen om de forskellige faser som gennemgås i projektet: inception, elaboration, konstruktion og transition. Arbejdet med UP vil hjælpe med at strukturere de forskellige dele, som skal laves, når der udvikles et softwaresystem.

Det vil også være oplagt at anvende Scrum i projektarbejdet, da Scrum er en agil metode til softwareudvikling. Begge metoder passer godt sammen, da der vil blive arbejdet iterativt igennem projektet. Iterationerne i projektarbejdet bliver overholdt bl.a. ved brug af sprints fra scrum-modellen. I disse sprints vil delelementer blive udviklet for derefter at blive testet og vurderet og til sidst accepteret eller modificeret. De mest væsentlige roller i Scrum vil blive brugt. Gruppen vil udvælge en Scrum Master og en Product Owner ved hvert sprint. Rollen som Scrum Master skal sørge for at resten af teamet overholder Scrum principperne og hhv. både holde styr og igangsætte scrum workflowet. Rollen som Product Owner har ansvaret for Product Backlog hvor alle aftalte elementer i projektet indsættes og bliver prioriteret.

Ud over disse to arbejdsmetoder vil Pair Programming (som stammer fra Extreme Programming) også kunne nævnes som en anvendt metode. Både når der skal programmeres, men også allerede i inceptionsfasen og elaborationsfasen fx når der arbejdes med kravspecificering.

9 Ressourcer

Semesterprojektet som helhed udgør 10 ECTS point, og derfor forventes der fra de studerende at der i mere eller mindre grad bliver fordelt 250-300 timer over hele semestret på dette projekt. Mere konkret, skal der i denne fase lægges en del tid og kræfter i at udfærdige et inceptionsdokument. Gruppen har valgt at have to-tre ugentlige møder hhv. tirsdag, onsdag og fredag - hvor i sidstnævnte også indebærer vejledermøder. På disse ugentlige møder er der en fleksibel tilgang - gruppen aftaler på forhånd dagsorden og løser de udvalgte opgaver og arbejder ca. 4-6 timer til hvert møde. Til møderne bliver gruppen til vi oshhv. opgaverne er løst og ellers bliver resterende indhold aftalt til hjemmearbejde. Dette dokumentet skal reviewes den 23. marts og derefter rettes til og afleveres den 6. april.

10 Konklusion

Systemet skal faciliterer kommunikationen mellem sagsbehandlerne og patienterne i DHUV (Digitalisering af handicap og udsatte voksne-område). Ved at lave en ny løsning til det kommunale område, indenfor social- og sundhedssektoren gør det arbejdet lettere på tværs af kommunerne, og samtidigt giver det et mere sammenhængende forløb, som skaber værdi hos borgerne.

Ud fra en analyse af virksomheden EG Team Online er der blevet udviklet en forretningsmodel. Denne forretningsmodel har overvejet virksomhedens organisation, kunder, produkter og økonomi. På baggrund af denne analyse, er der kommet frem til at det vil være godt for virksomheden at fortsætte udviklingen af systemet.

Gruppen udførte en navneordsanalyse i fællesskab over den udleverede projektcase og med udgangspunkt i det afgrænsede system. Dette dannede grundlag for at kunne afvikle den overordnede kravspecifikation - hvori gruppen fik udarbejdet en brugsmønstermodel og en domænemodel.

Gennem en analyse af kravspecifikationen på socialstyrelsens hjemmeside har vi fundet frem til kravene, som er væsentlige inden for vores problemdomæne. Ud fra voksenudredningsmetoden, har vi med udgangspunkt i sagsåbningsfasen udarbejdet en domæne- og brugsmønstermodel som reflekterer de krav, der blev fundet frem til.

Ud fra dette arbejde kan vi konkludere, at vi har samlet nok viden til videreførelse af projektet. Med domænemodellen og de forskellige brugsmønstre vil det, i elaborationsfasen, blive muligt at designe systemet og dermed lave en implementering af kravene.

11 Litteraturliste

Agile in a Flash. Udgivet af Tim Ottinger & Jeff Langr. Internetadresse:
<http://agileinaflash.blogspot.dk/2009/04/furps.html> - Besøgt d. 23.03.2018 (Internet)

Business Model Canvas og Alexander Osterwalder. Udgivet af Jan Bendtsen.
Internetadresse:
<http://forretningsmodellen.dk/2012/10/business-model-canvas-og-alexander-osterwalde/> - Besøgt d. 14.03.2018 (Internet)

Inceptionsdokument, checkliste. Udgivet af Lone Borgersen. Internetadresse:
<https://docs.google.com/document/d/1ktLSuOYURHfrKI2HGcxCbIzcki4zFDvLD30BDxY41vk/edit> - Besøgt d. 14.03.2018 (Internet)

Inceptionsdokument, eksempel. Udgivet af SDU. Internetadresse:
https://docs.google.com/document/d/1eg9Y-hHAKmerF15Q8-I_J0NGRPiKhZnq5iYmSX9fyI/edit - Besøgt d. 14.03.2018 (Internet)

Inceptionsfasen. Udgivet af Lone Borgersen. Internetadresse:
<https://docs.google.com/document/d/15dTfnH5YqGPnBI8EhAm0raiRtLdFuQMohd7auYWS344/edit> - Besøgt d. 14.03.2018 (Internet)

Voksenudredningsmetode. Udgivet af Socialstyrelsen. Internetadresse:
<https://socialstyrelsen.dk/filer/tvaergaende/sagsbehandling-og-organisering/link-metode/handbog-vum-1.pdf> - Besøgt d. 14.03.2018 (Internet)

Bilag

Bilag A: Logbog

Igennem projektforløbet har gruppen ført en digital logbog på ugebasis - som kan findes på gruppens GitHub. Der udvælges en ny referent pr. uge, som står for både logbog og mødereferater med vejleder. Logbogen er indeholdende følgende elementer: lokation, forberedelse, dagsorden og resumé for pågældende dage. Gruppens logbog på GitHub kan findes på følgende link:

<https://github.com/AndersBensen/2-semesterprojekt-sensum-udred/wiki>

Bilag B: Omverdensanalyse

Pestel-analyse (Generelt miljø)

Politiske kræfter

Eftersom at Danmark er en velfærdsstat bliver der sat mange penge af til uddannelse. Disse penge er med til at sikre et højt niveau på uddannelsen. Herved bliver der uddannet mange højtuddannede, som kan hjælpe EG Team Online med deres udvikling.

Eftersom at KMD ikke har monopol på IT-løsninger til det offentlige, har EG Team Online en ny målgruppe at henvende sig til. Dette bevirker at deres omsætning mm. kan forøges.

Økonomiske kræfter

Eftersom at IT systemer er blevet så stor en integreret del af folks hverdag, så vil EG Team Online næsten altid have et grundlag til salg af deres produkter. Selv under en ny finanskriser vil Sensum Udred stadig være en nødvendighed.

Kulturelle kræfter

Danmark har en arbejdskultur hvor man ikke ser negativt på at tage på arbejde, tværtimod. I de danske firmaer behøver man ikke at bevise sit værd over for sin arbejdsgiver hele tiden, det er et mere afslappet miljø. Dette gør det let for EG Team Online at finde kvalificeret arbejdskraft, så længe de har udfordrende og spændende projekter.

Teknologiske kræfter

Eftersom at EG Team Online udvikler IT-løsninger, så er de i en stærk teknologisk branche. Software er i rivende udvikling så det er nødvendigt for dem som firma hele tiden at holde sig opdateret, så de ikke bliver forældet. Dette er både positivt og negativt.

Miljømæssige kræfter

Ved brug af EG Team Onlines løsning vil det blive muligt at formindske brugen af papir, hvilket har en positiv miljømæssig påvirkning for dem.

Lovgivning

Der er blevet vedtaget en ny lov omkring datasikkerhed, kaldet persondataforordningen. Det er vigtigt for EG Team Online at udvikle produkter med denne forordning, da deres systemer indeholder personfølsomme data. Dette er både en fordel og ulempe for EG Team Online. Det kræver mere arbejde at sikre persondataens sikkerhed, hvilket er tid- og ressourcekrævende. Dog vil dette også sikre større sikkerhed og fortrolighed for EG Team Onlines kunder.

Porters Five Forces (Specifikke miljø)

Leverandørens forhandlingsstyrke

Leverandører på dette marked har en meget lav forhandlingsstyrke, da der er tilstrækkelig med specialuddannet arbejdskraft inden for branchen. Der kan derfor argumenteres for at der i denne branche er et større omfang af alternative leverandørmuligheder til rådighed. Disse substitutionsmuligheder (virksomheder), som kan levere et lignende produkt er også med til at afspejle leverandørens forhandlingsstyrke i denne branche.

Kundernes forhandlingsstyrke

Det offentlige har en rimelig forhandlingsstyrke, da der er flere virksomheder, som ønsker at byde på de projekter, som de sender ud i licitering. Det er trods alt det offentlige der til sidst vælger, hvem der skal lave systemet. Det offentlige er dog meget afhængige af de pågældende virksomheder, hvilket svækker deres forhandlingsstyrke. Når det kommer til den private sektor har kunderne en større forhandlingsstyrke, fordi der er vil være flere udbydere af skræddersyede softwaresystemer.

Truslen fra potentielle indtrængere

Truslen fra potentielle indtrængere for EG Team Online er ret stor, da der er mange firmaer som vil kunne lave samme produkt som de kan. Selvom at de potentielle indtrængere ikke er direkte konkurrenter nu, så kan de nemt blive det i fremtiden og EG Team Online skal derfor sørge for at holde sig attraktive i miljøet.

Et firma som NetCompany laver ikke pt. social- og sundhedssystemer for det offentlige nu, men de har erfaring med udvikling af offentlige systemer. De har bl.a. lavet Borger.dk og skal i gang med at lave Skats nye system. Derfor har de en del erfaring og vil være en seriøs trussel som potentiel indtrængere.

Truslen fra substituerende produkter

Minimal trussel, da der altid skal rapporteres om korrespondancer mellem sagsbehandlere og patienter. Leverandører i dette marked, skal alle levere et lignende produkt, derfor kan man argumentere for der ikke er mange substituerende produkter.

Konkurrencesituationen i branchen

Arbejdet med udsatte voksne og handicappede er en nichebranche, hvilket medfører at efterspørgslen for disse systemer ikke er ret høj. Dette medvirker umiddelbart til en større konkurrence, men siden der kun er enkelte store virksomheder i branchen, vil konkurrencen ikke være særlig stor. Samtidigt er der heller ikke virksomheder af samme kaliber, som specialisere sig i dette område, som netop EG Team Online gør.

EG Team Online har også erhvervet sig alle danske kommuner som kunder og dermed også en slags "monopol" på branchen, hvilket gør at konkurrencesituationen automatisk vil være mindre.²³

²³ <https://www.fyens.dk/erhverv/Stifter-saelger-fynsk-it-succes-/artikel/2555454>

SWOT-analyse

Styrker

EG Team Online har stor erfaring med at udvikle systemer til kommuner inden for social- og sundhedssektoren. De er en mellemstor virksomhed som opererer i hele Norden. Dette giver dem en stor ressourcebank som de kan trække på, når de får større opgaver så som Sensum Udred.

IT systemer ser pt. ud til at være vedvarende, hvilket er en styrke, da det giver EG Team Online mulighed for fortsat at udvikle systemer. Der vil også fortsat være brug for vedligeholdelse af eksisterende systemer.

Svagheder

Hvis ikke virksomheden investerer nok i Research & Development, vil de komme bagud teknologisk, hvilket vil stille dårligt i forhold til konkurrenterne, og det vil være en svaghed.

Muligheder

Det er lettere at distribuere IT systemer end fysiske produkter og det giver god mulighed for EG Team Online til at komme ud til nye firmaer med deres produkter. Hvis de følger med i den teknologiske udvikling vil det give dem gode muligheder for at komme foran andre i udvikling og salg af nye produkter.

Trusler

Det er let at komme ind på IT markedet, og der er derfor en stor trussel i at konkurrenter og nye virksomheder vil etablere sig og forsøge at tage markedsandele.

Konklusion

EG Team Online er rigtigt godt stillet på markedet, hvor de leverer løsninger til kommuner m.m. De har god erfaring, men også anbefaling fra kommunerne på baggrund af produkter de tidligere har udviklet. Da de blandt andet har udviklet Sensum Bosted, som blev en stor success indenfor det socialpædagogiske område, har de gode muligheder for at lave lignende produkter.

Det er nemt at komme ind på IT markedet, da det ikke kræver stor startkapital eller ressourcer og det skal EG Team Online være meget opmærksomme på. De har dog en stor fordel med deres store erfaring i udvikling af IT produkter og er på den måde mere attraktive end konkurrenterne.

De skal dog være opmærksomme på den teknologiske udvikling, som både kan være en svaghed og en styrke for dem. Hvis de vælger at udvikle teknologien og lære af andre nye teknologier kan de bruge det, som en styrke for deres egne produkter, hvorimod hvis de ikke følger med kan deres konkurrenter overhale dem hurtigt udviklings fronten.

Miljøet er relativt komplekst og samtidigt meget ustabil, hvilket giver et meget usikkert miljø. Det vil derfor kræve flere ressourcer for EG Team Online at bevæge sig og udvikle produkter i miljøet.

E: Projektlog

Igennem projektforløbet har gruppen ført en digital logbog på ugebasis - denne kan findes på gruppens GitHub. Der udvælges en ny referent pr. uge, som står for både logbog og mødereferater med vejleder. Gruppens logbog på GitHub kan findes på følgende link:

<https://github.com/AndersBensen/2-semesterprojekt-sensum-udred/wiki>

Som eksempel på projektgruppens logbog - ses billedet på næste side. Logbogen er indeholdende følgende elementer: lokation, forberedelse, dagsorden og resumé for pågældende dage.

Disposition
1. Sidehovede [Uge Nr.]: Forfatter, Dato
2. Lokation Gruppens møde lokation og evt. lokale
3. Forberedelse Her aftales al forberedelse inden gruppen mødes
4. Dagsorden Her beskrives den gældende dagsorden for gruppens fremmøde
5. Resumé Her opsummeres hvad gruppen har opnået den gældende dag

Week 06

AndersBensen edited this page on 13 Mar · 1 revision

Edit New Page

Fredag

Lokation:

- Mødtes ved MMMI

Forberedelse:

- Læs Sensum casen

Dagsorden:

- Gruppekонтракт, møde med vejleder, forventninger
- Husk at book lokaler til fredag næste uge

Resume:

- Dagsorden fuldført
- Første møde med Jan-Emil
- Dokumenter oprettet til resume, planlægning osv.

▼ Pages 15

Home
Template for weeks
Week 06
Week 07
Week 08
Week 09
Week 10
Week 11
Week 12
Week 13
Week 14
Week 15
Week 16
Week 17
Week 18

Figur E1: Denne figur præsenterer gruppens logbog i uge 6.

F Interne Bilag

Bilag F1: Scrum Release Backlog

Release Backlog			
Krav der er valgt til prototypen			
Krav	Status	Prioritet	Est tid
Opret patient	Færdig	Must have	10
Opret sag	Færdig	Must have	20
Rediger sag	Færdig	Must have	12
Opret henvendelse	Færdig	Must have	15
Opret ansat	Færdig	Must have	10
Slet ansat	Færdig	Should have	10
Klassediagram	Færdig	Must have	12
Forberedelse til virksomhedsbesøg	Færdig	Must have	4
3-lags arkitektur	Færdig	Must have	30
Rapportelementer (1. udgave)	Færdig	Should have	50
...			
Tid i alt			173 timer

Figur F1.1: Denne figur præsenterer release backlog for første sprint.

Release Backlog			
Krav der er valgt til prototypen			
Krav	Status	Prioritet	Est tid
Log ind	Igang	Must have	15
Log af	Færdig	Should have	5
Søg Borger	Ikke startet	Should have	5
Timed log af	Igang	Could have	10
GUI design	Igang	Must have	15
GUI implementering	Igang	Must have	20
Database opsætning	Igang	Must have	20
Database implementering	Igang	Must have	10
Rapport (Færdig)	Igang	Must have	98
Arkitektur (acq)	Færdig	Should have	2
Tid i alt			200 timer

Figur F1.2: Denne figur præsenterer release backlog for andet sprint.

Bilag F2: Scrum Product Backlog

Product Backlog			
<i>Kommende krav til produktet</i>			
Krav	Kategori	Status	Prioritet
Lav udskrift	Bruksmønster	Ikke startet	Should have
Søg efter patient	Design	Igang	Should have
Opret patient	Bruksmønster	Afsluttet	-
Log ind	- -	Igang	Must have
Log af	- -	Igang	Should have
Opret sag	- -	Færdig	Must have
Rediger sag	- -	Færdig	Must have
Opret ansat	- -	Færdig	Must have
Slet ansat	- -	Færdig	Should have
Opret henvendelse	- -	Færdig	Must have
Kryptering	Sikkerhed	Ikke startet	Should have
Log actions	Sikkerhed	Færdig	Must have
Timed Log out	Sikkerhed	Igang	Could have
Klassediagram	Analyse	Færdig	Must have
Forberedelse til virksomhedsmøde	Analyse/Design	Færdig	Must have
Bruksmønsterrealisering	Analyse	Igang	Must have
3-lags arkitektur	Design	Færdig	Must have
GUI	Design	Igang	Must have
Database opsætning	Design	Igang	Must have
Database implementering	Design	Igang	Must have
Rapportelementer (1. udgave)	Dokumentation	Færdig	Must have
Rapportelementer (2. udgave)	Dokumentation	Igang	Must have

Figur F2.1: Denne figur præsenterer gruppens product backlog.

Bilag F3: Scrum Sprint Backlog

Sprint Backlog 1

6/4 - 4/5

Første sprint - Funktionelt sagsåbnings-system med lagdelstruktur

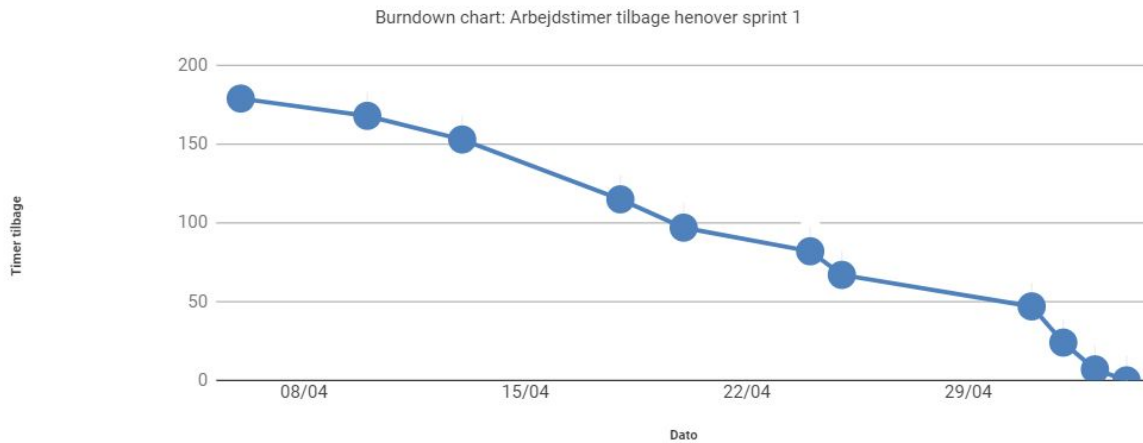
				Start											
				Dato: 06/0410/0413/0418/0420/0424/0425/0401/0502/0503/0504/05											
Krav	Kategori	Status	MoSCoW	Tid tilbage: 179	168	153	115	97	82	67	47	24	7	0	
Understøttende workflows				Hastighed:	12										
Planlægge sprintet	Planlægning	Færdig	M		4	0	0	0	0	0	0	0	0	0	
Etablering af udviklingsmiljø	Udviklingsmiljø	Færdig	M		4	4	4	3	3	0	0	0	0	0	
Afslutning af sprint	Planlægning	Færdig	M		2	2	2	2	2	2	2	2	1	0	
Rapportskrivning (1. udkast)	Dokumentation	Færdig	M		50	49	49	48	48	48	30	20	5	0	
Kerne-workflows															
Detaljer af brugsmønstre															
Brugsmønstre (Opret henvendelse)	Krav	Færdig	M		1	1	0	0	0	0	0	0	0	0	
Brugsmønstre (Opret sag)	Krav	Færdig	M		1	1	0	0	0	0	0	0	0	0	
Brugsmønstre (Rediger sag)	Krav	Færdig	M		1	1	0	0	0	0	0	0	0	0	
Brugsmønstre (Opret patient)	Krav	Færdig	M		1	1	1	0	0	0	0	0	0	0	
Brugsmønstre (Opret ansat)	Krav	Færdig	M		1	1	0	0	0	0	0	0	0	0	
Brugsmønstre (Slet ansat)	Krav	Færdig	S		1	1	0	0	0	0	0	0	0	0	
Analyse															
Analysediagram	Analyse	Færdig	M		12	6	6	5	5	5	3	2	1	0	
Brugsmønstreanalyse (Opret henvendelse)	Analyse	Færdig	M		1	1	0	0	0	0	0	0	0	0	
Brugsmønstreanalyse (Opret sag)	Analyse	Færdig	M		1	1	0	0	0	0	0	0	0	0	
Brugsmønstreanalyse (Rediger sag)	Analyse	Færdig	M		1	1	0	0	0	0	0	0	0	0	
Brugsmønstreanalyse (Opret patient)	Analyse	Færdig	M		1	1	1	0	0	0	0	0	0	0	
Brugsmønstreanalyse (Slet ansat)	Analyse	Færdig	M		1	1	0	0	0	0	0	0	0	0	
Brugsmønstreanalyse (Opret ansat)	Analyse	Færdig	S		1	1	0	0	0	0	0	0	0	0	
Design															
Arkitektur	Design	Færdig	M		4	4	4	3	2	2	0	0	0	0	
Brugsmønstreanalyse (Opret henvendelse)	Analyse	Færdig	M		1	1	0	0	0	0	0	0	0	0	
Brugsmønstreanalyse (Opret sag)	Analyse	Færdig	M		1	1	0	0	0	0	0	0	0	0	
Brugsmønstreanalyse (Rediger sag)	Analyse	Færdig	M		1	1	0	0	0	0	0	0	0	0	
Brugsmønstreanalyse (Opret patient)	Analyse	Færdig	M		1	1	1	0	0	0	0	0	0	0	
Brugsmønstreanalyse (Opret ansat)	Analyse	Færdig	M		1	1	0	0	0	0	0	0	0	0	
Brugsmønstreanalyse (Slet ansat)	Analyse	Færdig	S		1	1	0	0	0	0	0	0	0	0	
Brugergrænsefladelag	Design	Færdig	M		5	5	5	4	2	2	2	0	0	0	
Forretningslogik	Design	Færdig	M		16	16	16	13	10	8	4	4	0	0	
Persistenslag	Design	Færdig	M		5	5	5	4	4	2	1	1	0	0	
Implementering															
Opret henvendelse	Implementering	Færdig	M		11	11	11	6	2	2	0	0	0	0	
Opret sag	Implementering	Færdig	M		16	16	16	6	4	2	1	1	0	0	
Rediger sag	Implementering	Færdig	M		8	8	8	4	4	2	2	2	0	0	
Opret patient	Implementering	Færdig	M		6	6	6	0	0	0	0	0	0	0	
Opret ansat	Implementering	Færdig	M		6	6	6	5	3	1	0	0	0	0	
Slet ansat	Implementering	Færdig	S		6	6	6	6	3	1	0	0	0	0	
Test															
Opret henvendelse	Test	Færdig	M		1	1	1	1	1	1	0	0	0	0	
Opret sag	Test	Færdig	M		1	1	1	1	1	1	1	1	0	0	
Rediger sag	Test	Færdig	M		1	1	1	1	1	1	1	1	0	0	
Opret patient	Test	Færdig	M		1	1	1	1	1	1	0	0	0	0	
Opret ansat	Test	Færdig	M		1	1	1	1	0	0	0	0	0	0	

Figur F3.1: Denne figur præsenterer sprint backlog for første sprint.

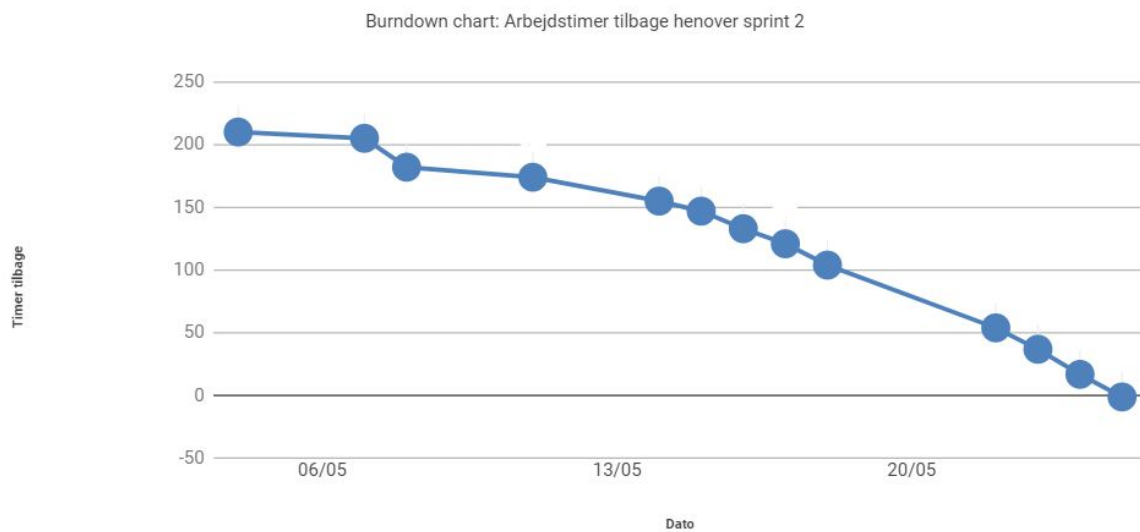
Sprint Backlog 2																	
7/5 - 31/5																	
Andet sprint - Funktionelt kørende system med fyldsestgørende dokumentation																	
Produkt: Intuitiv grafisk brugergrænseflade med relationel database																	
				Start													
				Dato: 04/05 07/05 08/05 11/05 14/05 15/05 16/05 17/05 18/05 22/05 23/05 24/05 25/05													
Krav	Kategori	Status	MoSCoW	Tid tilbage:	210	205	182	174	155	147	133	121	104	54	37	17	-1
Understøttende workflows				Hastighed:	18												
Planlægge sprintet	Planlægning	Færdig	M		5	0	0	0	0	0	0	0	0	0	0	0	0
Afslutning af sprint	Planlægning	Igang	M		5	5	5	5	5	5	5	5	5	5	5	5	5
Rapportskrivning	Dokumentation	Færdig	M		98	98	98	98	0	0	0	0	0	0	0	0	0
Rapport: Indledning	Dokumentation	Igang	M						10	10	10	10	5	1	1	1	1
Rapport: Metoder, Planer, Forløb	Dokumentation	Igang	M						20	10	10	10	5	1	1	1	1
Rapport: Elaboration	Dokumentation	Igang	M						30	30	30	30	25	10	5	3	3
Rapport: Evaluering	Dokumentation	Igang	M						20	20	20	20	20	20	10	3	3
Rapport: Konklusion	Dokumentation	Igang	M						10	10	10	10	10	6	4	2	2
Rapport: Prolog	Dokumentation	Igang	M						5	5	5	5	5	0	0	0	0
Rapport: Bilag	Dokumentation	Igang	M						3	3	3	3	3	3	3	3	2
Kerne-workflows																	
Detaljer af brugsmønstre																	
Brugsmønstre (Log ind)	Krav	Færdig	M		1	1	0	0	0	0	0	0	0	0	0	0	0
Brugsmønstre (Log af)	Krav	Færdig	S		1	1	0	0	0	0	0	0	0	0	0	0	0
Analyse																	
Brugsmønstrerealisering (Log ind)	Analyse	Færdig	M		1	1	0	0	0	0	0	0	0	0	0	0	0
Brugsmønstrerealisering (Log af)	Analyse	Færdig	S		1	1	0	0	0	0	0	0	0	0	0	0	0
Design																	
Arkitektur (acq)	Design	Færdig	S		2	2	0	0	0	0	0	0	0	0	0	0	0
Brugsmønstrerealisering (Log ind)	Design	Færdig	M		1	1	0	0	0	0	0	0	0	0	0	0	0
Brugsmønstrerealisering (Log af)	Design	Færdig	S		1	1	0	0	0	0	0	0	0	0	0	0	0
Klassediagram									5	4	3	2	2	1	1	0	0
Implementering																	
Log ind	Implementering	Igang	M		11	11	2	2	2	2	1	0	0	0	0	0	0
Log af	Implementering	Færdig	S		1	1	1	0	0	0	0	0	0	0	0	0	0
Seg Borger	Implementering	Færdig	S		4	4	4	4	4	4	1	0	0	0	0	0	0
Timed log af	Implementering	Igang	C		9	9	9	9	2	2	2	2	2	0	0	0	0
GUI design	Implementering	Igang	M		14	14	14	10	5	10	5	1	1	0	0	0	0
GUI implementering	Implementering	Ikke startet	M		19	19	19	19	19	19	15	10	10	1	1	0	0
Database opsætning	Implementering	Igang	M		19	19	10	8	2	2	2	2	1	0	0	0	0
Database implementering	Implementering	Igang	M		9	9	9	9	4	2	2	2	1	0	0	0	0
Test																	
Log ind	Test	Færdig	M		1	1	0	0	0	0	0	0	0	0	0	0	0
Log af	Test	Færdig	S		1	1	1	0	0	0	0	0	0	0	0	0	0
Seg Borger	Test	Ikke startet	S		1	1	1	1	1	1	1	1	1	0	0	0	0
Timed log af	Test	Igang	C		1	1	1	1	1	1	1	1	1	0	0	0	0
GUI design	Test	Igang	M		1	1	1	1	1	1	1	1	1	0	0	0	0
GUI implementering	Test	Igang	M		1	1	5	5	5	5	5	5	5	6	6	0	0
Database opsætning	Test	Færdig	M		1	1	1	1	0	0	0	0	0	0	0	0	0
Database implementering	Test	Ioano	M		1	1	1	1	1	1	1	1	1	0	0	0	0

Figur F3.2: Denne figur præsenterer sprint backlog for andet sprint.

Bilag F4: Scrum Burndown Chart

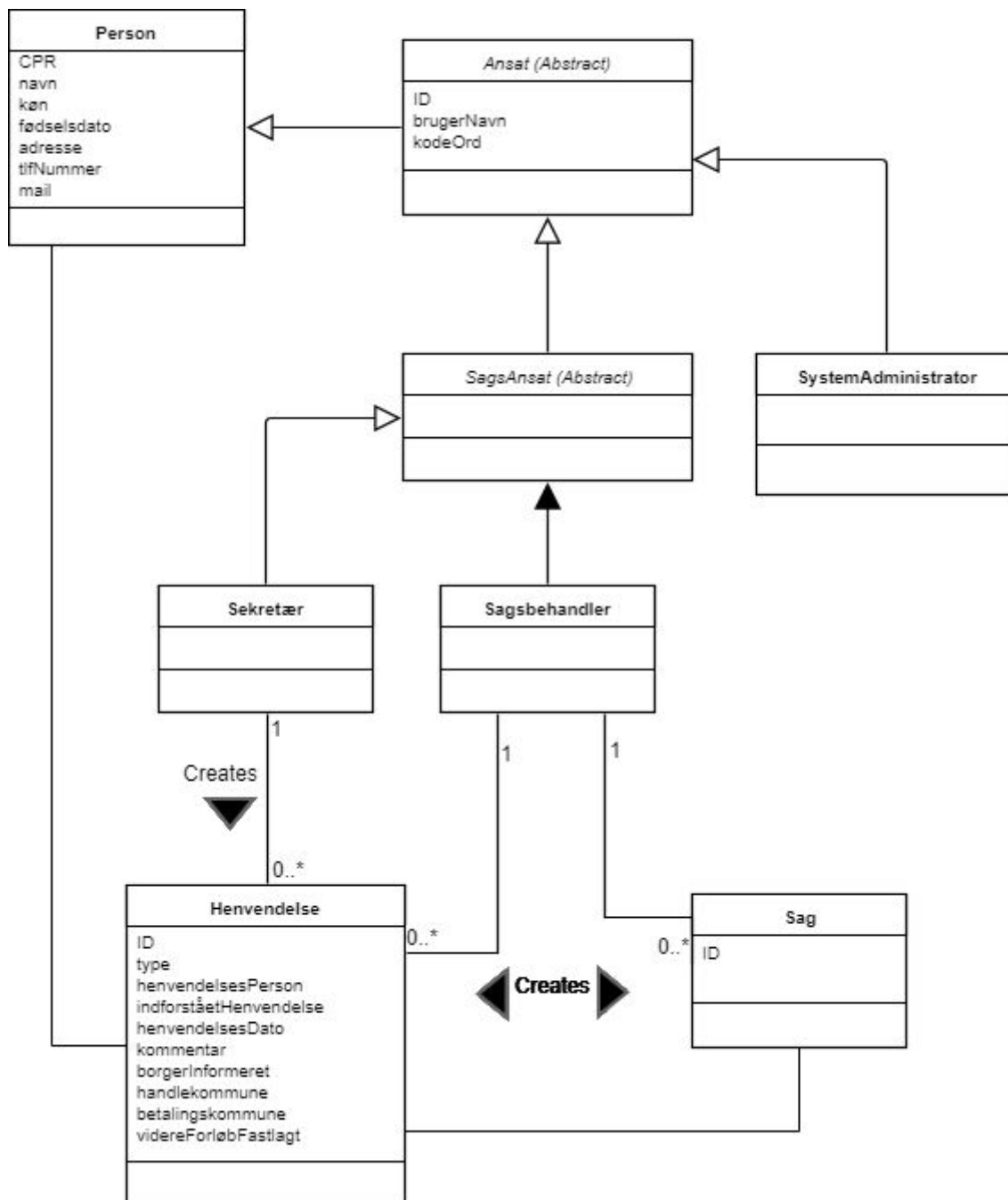


Figur F4.1: Denne figur præsenterer burndown chart for første sprint.

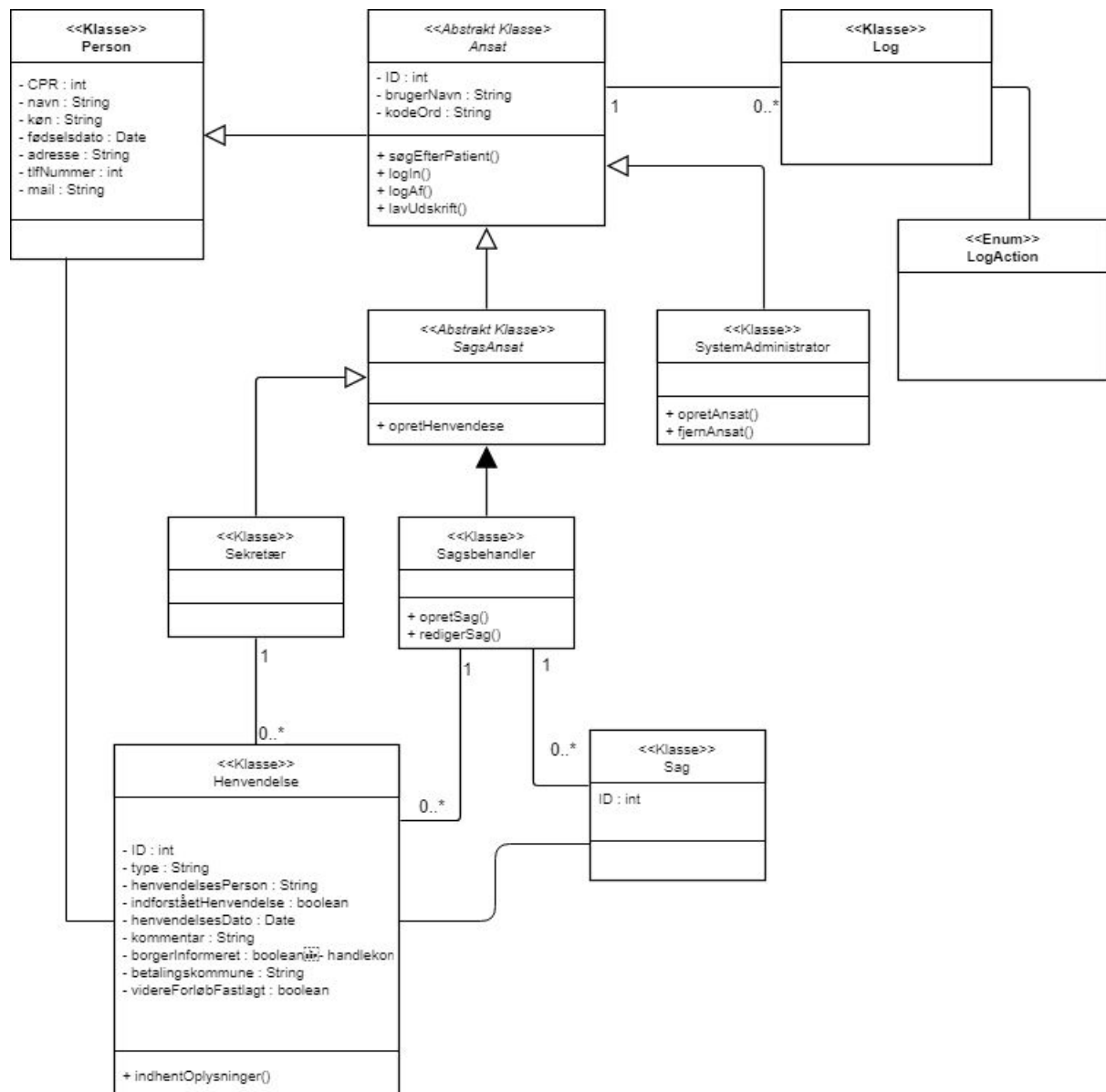


Figur F4.2: Denne figur præsenterer burndown chart for andet sprint.

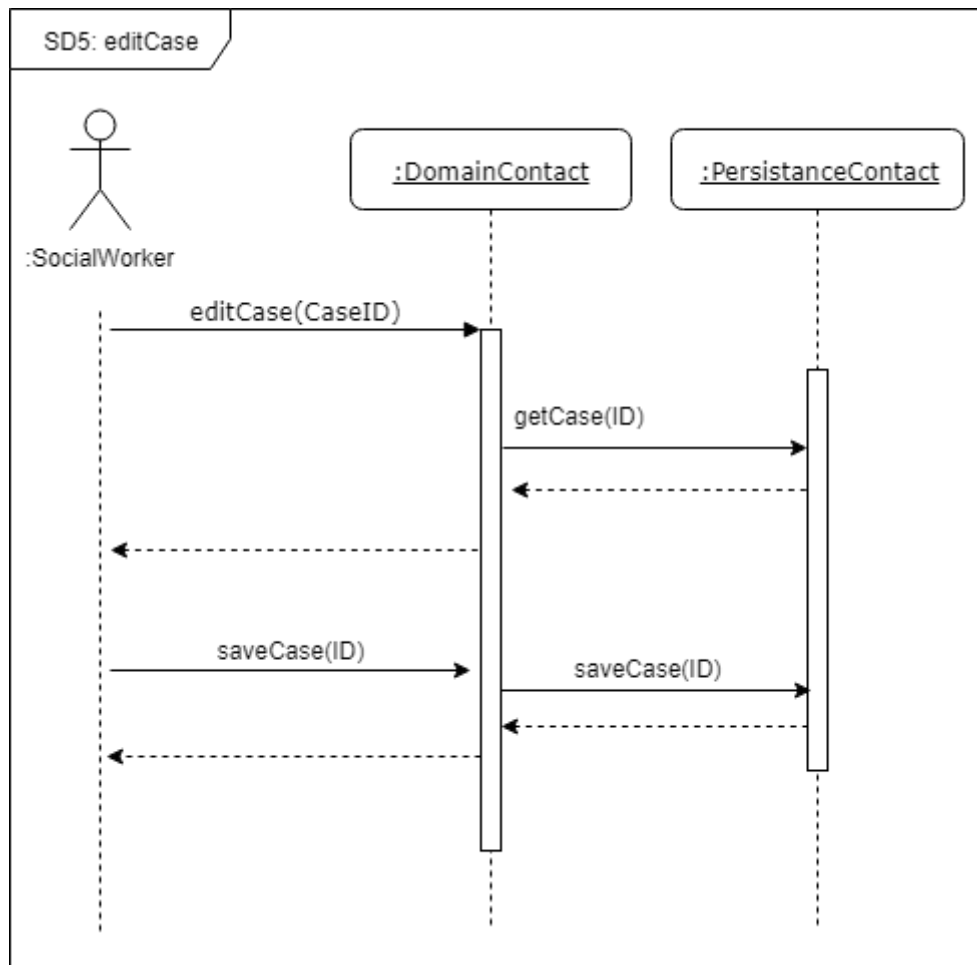
Bilag F5: Domænemodel



Bilag F6: Analyseklassediagram



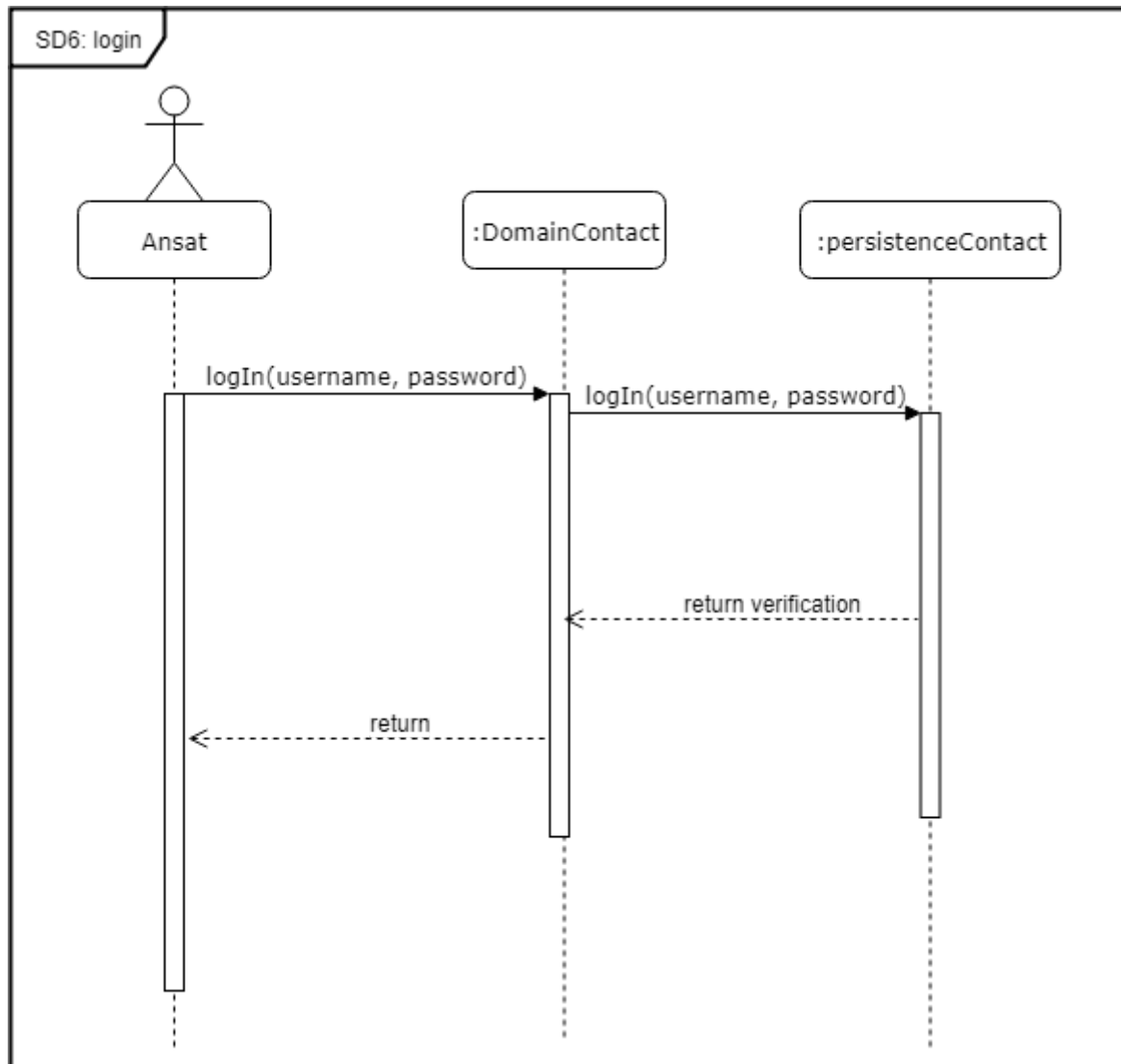
Bilag F7: Sekvensdiagrammer



Figur F7.1: Denne figur præsenterer sekvensdiagram rediger sag

Kontrakt	
Operation	editCase(caseID)
Krydsreference	BM05 rediger sag
Ansvar	At redigere en sag. At gemme den redigerede sag i databasen.
Output	Sagen er blevet redigeret. Sagsbehandleren får en bekræftelse på at sagen er redigeret.
Prækondition(er)	Sagsbehandleren skal være logget ind, og henvendelsen skal være konkret.
Postkondition(er)	Ændringen er gemt i databasen.

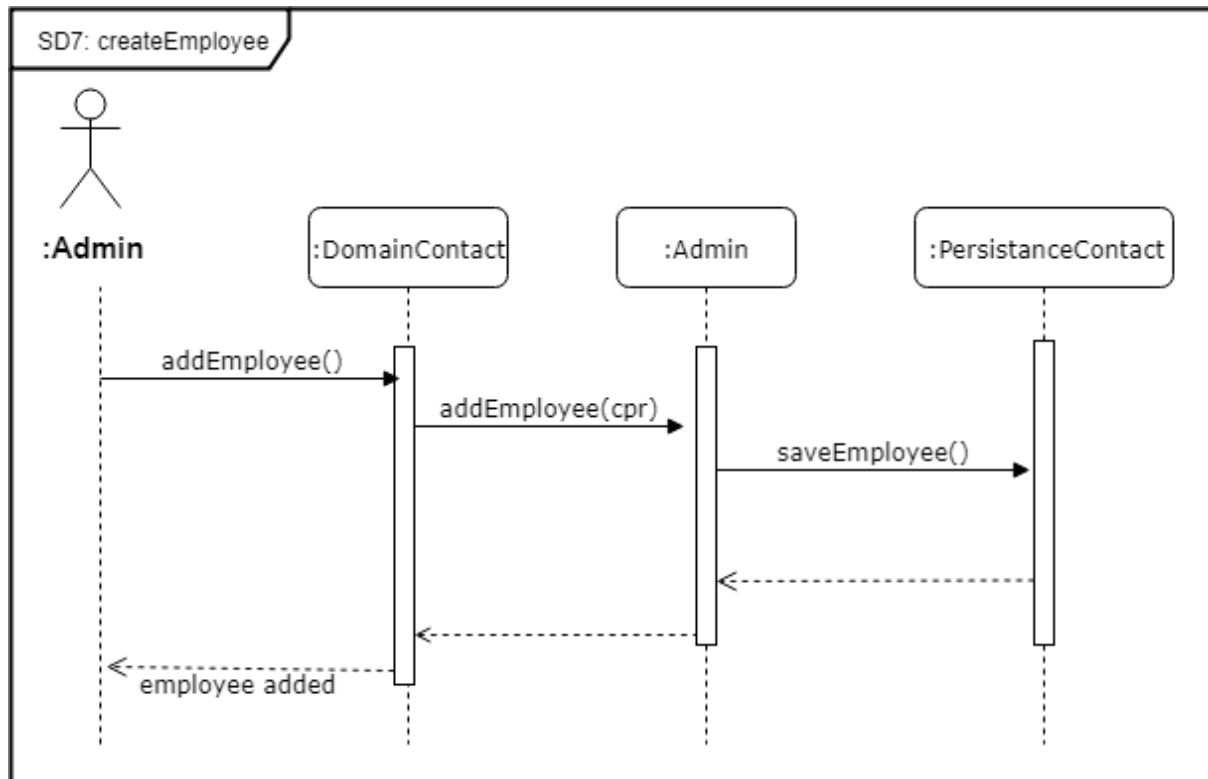
Tabel F7.1: Denne tabel præsenterer kontrakt til rediger sag



Figur F7.2: Denne figur præsenterer sekvensdiagram login

Kontrakt	
Operation	login(username, password)
Krydsreference	BM06 Log ind
Ansvar	At logge ind i systemet som bruger. At gemme hvem, som logger ind, i databasen.
Output	Brugeren får bekræftet at denne er logget ind.
Prækondition(er)	En ansat er ikke logget ind. Ansats skal have et brugernavn og adgangskode.
Postkondition(er)	En bruger er logget ind i systemet. Handlingen er gemt i databasen.

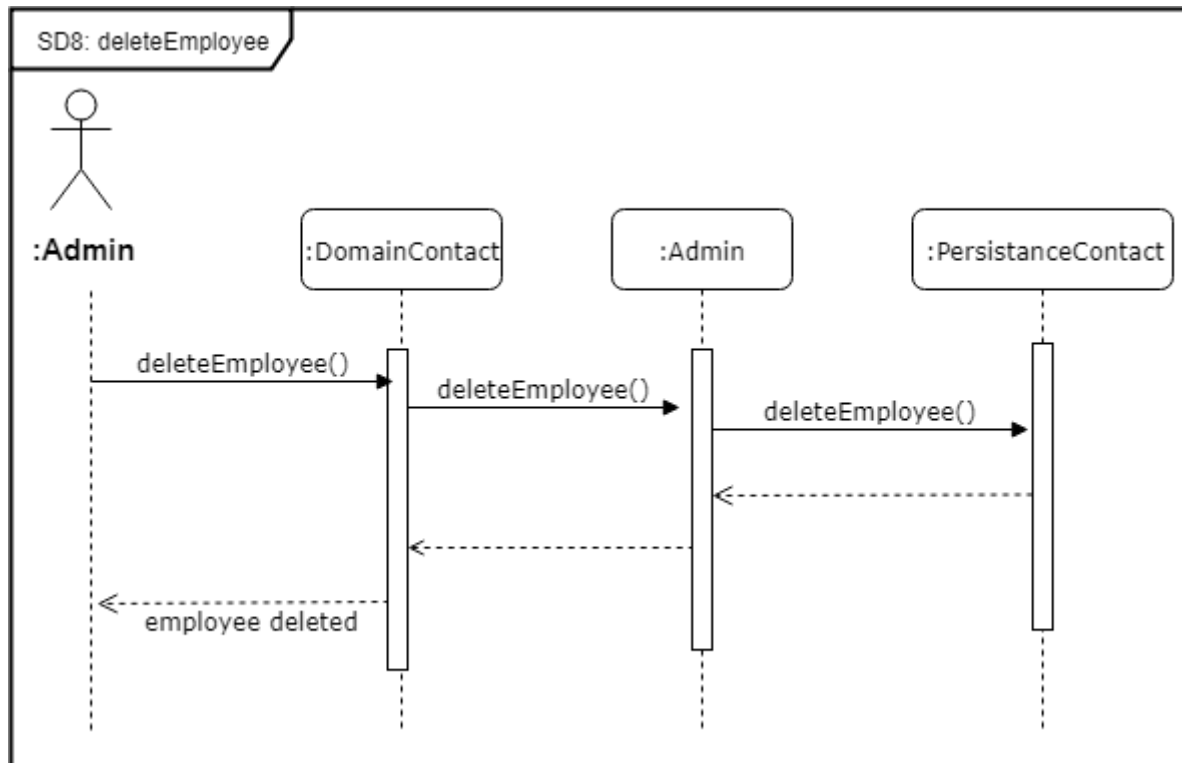
Tabel F7.2: Denne figur præsenterer kontrakt til login



Figur F7.3: Denne figur præsenterer sekvensdiagram opret ansat

Kontrakt	
Operation	addEmployee(ID, userName, password, securityLevel)
Krydsreference	Krydshenvisninger: BM07 Opret Ansat
Ansvar	At oprette en ansat i systemet med ID, brugernavn, kodeord og sikkerhedsniveau At admin får bekræftet at en ansat nu er oprettet med al given information
Output	Bekræftelsesbesked af ny ansat
Prækondition(er)	Bruger skal være logget ind som admin
Postkondition(er)	Ny ansat er oprettet i system Ansats har nu eget brugernavn og kodeord Ansats er tildelt et sikkerhedsniveau ud fra stilling Alle operationer er logget i databasen

Tabel F7.3: Denne figur præsenterer kontrakt til opret ansat



Figur F7.4: Denne figur præsenterer sekvensdiagram slet ansat

Kontrakt	
Operation	deleteEmployee(ID)
Krydsreference	Krydshenvisninger: BM08 Slet Ansat
Ansvar	At slette en ansat i systemet med unikt ID At admin får bekræftet at en ansat nu er slettet med det givne ID
Output	Bekræftelsesbesked af slettet ansat
Prækondition(er)	Bruger skal være logget ind som admin
Postkondition(er)	Ansats er slettet i system Ansats har ikke eget brugernavn og kodeord mere Alle operationer er logget i databasen

Tabel F7.4: Denne figur præsenterer kontrakt til slet ansat

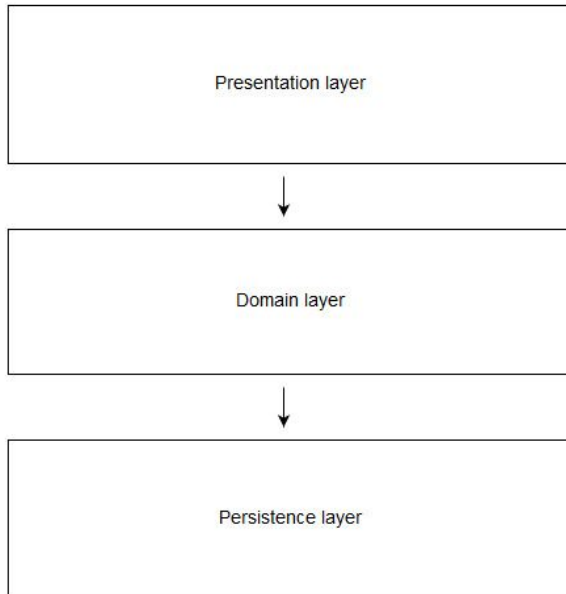
Bilag F8: Designklassediagram

Klassediagrammet er udarbejdet vha. draw.io og findes gennem det vedlagte link nedenfor.

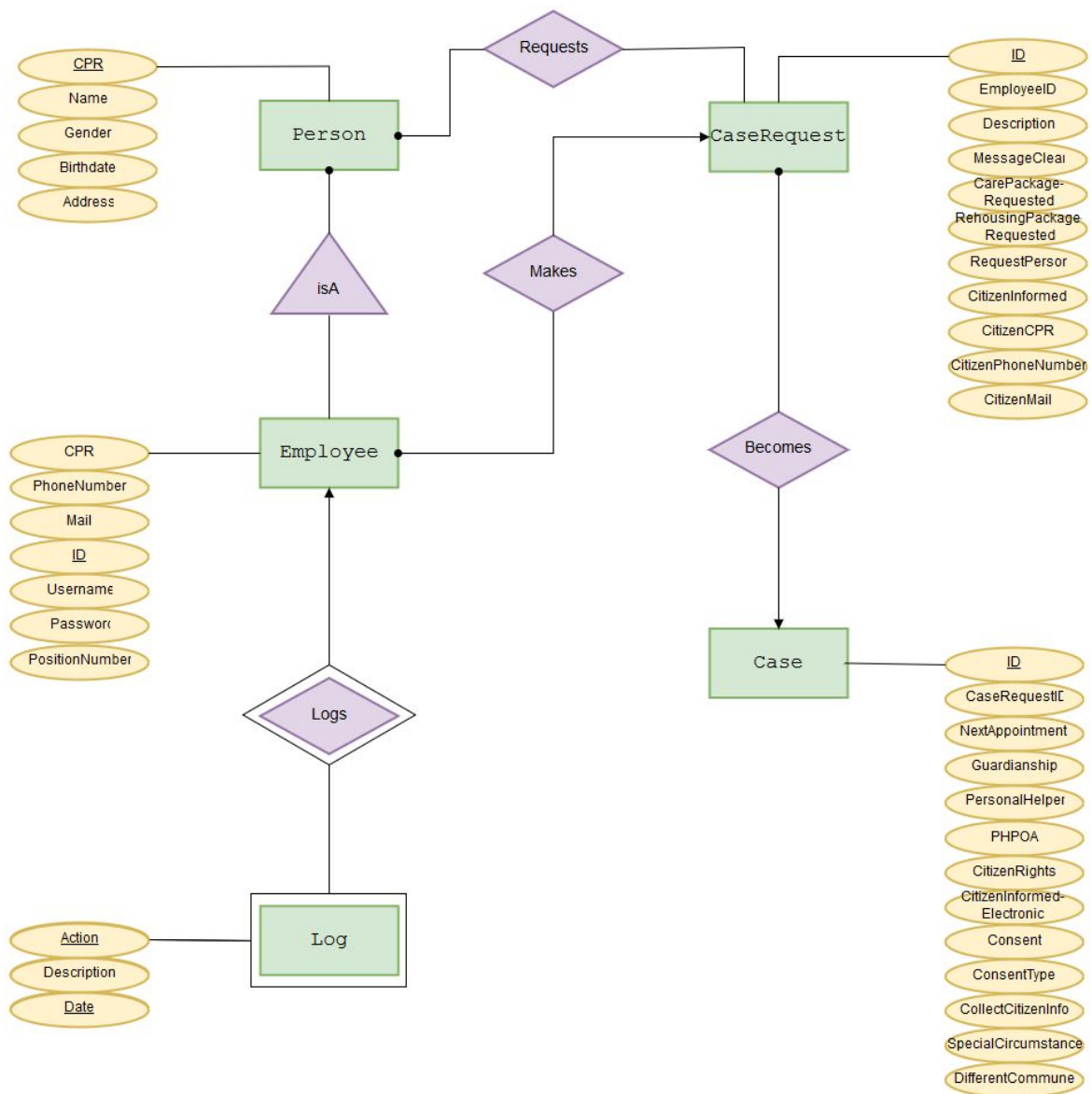
Link:

https://drive.google.com/file/d/1lg_56saJKJtVPYMtL7K4_kxsBHKtma9N/view?usp=sharing

Bilag F9: Arkitektur Diagram



Bilag F10: 3NF E/R Model



G Eksterne Bilag

Se vedlagt zip mappe for eksterne bilag. Disse ligger i mappen 'bilag'. Mappen er navngivet efter følgende opsætning: "SI2-PRO Grp " + gruppenr + "Bilag".