Webapplikationsudvikling

Udvikling af et system til håndtering og planlægning af ansattes vagter i en virksomhed.

Udarbejdet af Shahroz Ali Azmat

Afsluttende hovedopgave
Semester og sted: 5. Semester - ErhvervsAkademi Sjælland Roskilde
Vejleder: Michael Claudius

Indholdsfortegnelse

1 Indledning

Denne opgave sætter fokus på udvikling af en webapplikation ved hjælp af en iterativ og objektorienteret softwareudviklingsproces. Projektet vil således belyse tiltag der kan udføres i forbindelse med udviklingsprocessen for at finde frem til brugerens krav og behov samt hvordan udviklingsmetodologien Unified Process (UP) og notationssproget Unified Modeling Language (UML) kan inddrages i forbindelse med analyse og design af en webapplikation. Endvidere vil opgaven med afsæt i udviklingsværktøjer som HTML5 (HTML5, CSS3 og JavaScript) samt JSP (Java Server Pages) belyse hvordan disse kan anvendes i forbindelse med udvikling og implementering af en webapplikation og dermed vurdere fordele og ulemper ved valg af netop JSP som teknologi i sådanne projekter. Det overordnet formål med opgaven er at udvikle et system til håndtering og planlægning af ansattes vagter i en virksomhed.

2 Projektetablering

Hensigten med denne projektetablering er at få afklaret og beskrevet en række formål og forpligtelser med hensyn til blandt andet estimering af ressourcebehov og tidsplaner til projektet.

Der vil i den følgende del af opgaven blive lagt fokus på planlægningen af projektets overordnet udførsel indenfor de givne rammer, herunder hvordan de forskellige faser skal tilrettelægges, hvilke aktiviteter der er relevante samt hvilke teknikker og beskrivelsesværktøjer der tænkes anvendt.

2.1 Introduktion til samarbejdspartneren og den tilhørende opgave

I forbindelse med udarbejdelsen af denne hovedopgave blev der blevet taget kontakt til virksomheden Sales-Point der beskæftiger sig med telemarketing og mødebooking.

Sales-Point blev grundlagt i starten af 2014 af to ildsjæle med det formål at styrke og udbygge virksomheders positioner på erhvervsmarkedet igennem menneskelig forståelse og forretningsmæssig kontakt.

Virksomheden som har hovedkontor i Søborg, arbejder primært med salg af mobiltelefoner, abonnementer, hosted telefoniløsning/omstilling samt IP- og fastnettelefoni målrettet erhvervsmarkedet.

I virksomhedens korte levetid har der været stor fremgang og den beskæftiger i dag over 12 ansatte. Grundet denne stigende udvikling står virksomheden overfor en række praktiske og tekniske udfordringer i den nærmere fremtid.

I forbindelse med dette blev der afholdt et møde med virksomheden hvor de forskellige problemstillinger blev fremlagt for undertegnede. Efter en kort gennemgang af de mulige projekter blev der relativt hurtigt enighed omkring et særligt projekt som både havde en særlig betydning for virksomhedens daglige drift og samtidig kunne være medvirkende til at udfordre mig på mine faglige kompetencer.

Virksomheden har ønsket at få udviklet en applikation som kan være medvirkende til at give et overblik over de ansattes vagter og den dertilhørende information på en let og praktisk måde.

2.2 Betingelser

Det forventes, at den første version af applikationen kører i drift ca. 2 måneder efter projektstart. Denne version af applikationen skal leve op til de overordnede funktionelle krav, der er beskrevet i projektgrundlaget[1] og derudover skal den kunne kører på virksomhedens eksterne server.

Målet for projektet er at arbejde igennem de forskellige faser og niveauer i et projektforløb der indebærer analyse, design og dokumentation samt udarbejdelse af et færdigt produkt. Dette vil således ske i henhold til systemudviklingsmetoden Unified Process og notationssproget UML som vil blive beskrevet senere i rapporten.

2.3 Ressourcer

De forskellige former for ressourcer og kompetencer der vil blive gjort brug af under dette forløb vil blive gennemgået i det følgende afsnit.

Ressourcer omfatter i dette tilfælde de personer eller de redskaber der anvendes til at fuldføre projektet.

2.3.1 Menneskelige ressourcer

Undertegnede

I forhold til mine kompetencer kan jeg med fordel deltage i udvikling af praksis inden for softwareudvikling, projektarbejde på kompetent vis, samt bidrage i et fagligt og tværfagligt samarbejde i forbindelse med udvikling, fejlsøgning og idriftsættelse af it-systemer.

I forbindelse med mit uddannelsesforløb har jeg tilegnet mig viden inden for forskellige sprog og metoder herunder Java, HTML5 og SQL, samt objektorienteret programmering og objektorienteret analyse og design.

[1] Der henvises her til Inception fasen som vil blive påbegynde i den næste del af rapporten i overensstemmelse med Unified Process og i forlængelse af denne projektetablering.

Virksomhedens ejere

Virksomhedens ejere kommer fra en uddannelsesmæssig baggrund som henholdsvis eksportingeniør og økonom. De har blandt andet beskæftiget sig med projektstyring af store udviklingsprojekter samt salg og markedsføring indenfor virksomheder såsom NNIT og Tryg.

Deres erfaringer og kompetencer indenfor blandt andet udviklingsprojekter, projektstyring og markedsføring kan give et betydeligt løft til hele arbejdsprocessen, herunder særligt i forbindelse med udarbejdelsen af kravspecifikationerne.

Vejleder

Under dette projektløb vil Michael Claudius agere som vejleder og der vil derfor kunne drages stor fordel af hans særlige erfaringer og evner indenfor blandt andet programmering og opgaveskrivning.

2.3.2 Teknologiske ressourcer

Blandt de teknologiske ressourcer indgår aktiviteter, der kan være medvirkende til forbedre og effektivisere det primære produkt. I forbindelse med dette kan nævnes nogle af de teknologiske ressourcer som virksomheden har stillet til rådighed heriblandt deres eksterne webserver til hosting af applikationen. Derudover vil der blive gjort brug af forskellige værktøjer og programmer til at effektivisere design- og udviklingsprocessen i forbindelse med projektet. Blandt disse værktøjer kan nævnes:

- Microsoft Visio (Illustrationer og diagrammer)
- MockFlow (Prototyper og Mockups)
- Adobe Photoshop CC (Design og grafik)
- Twitter Bootstrap (Layout og Frontend)
- Apache Tomcat (Webserver)

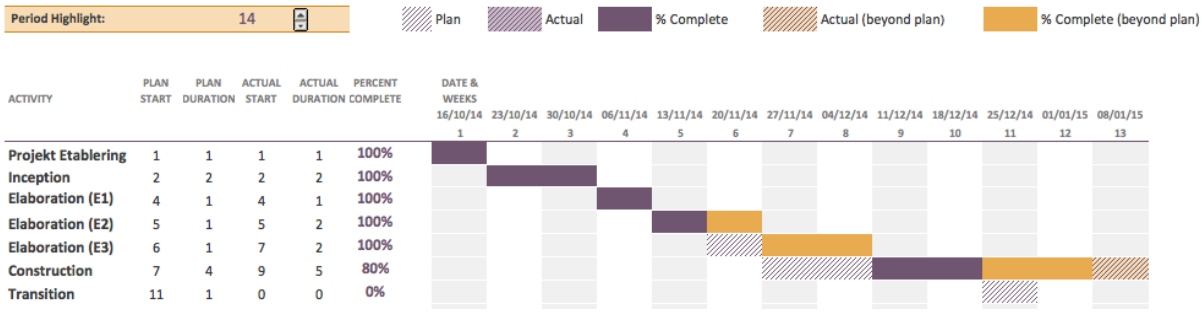
2.4 Produkter og ydelser

Projektet skal resultere i følgende produkter og ydelser:

- 1. version af applikationen, som beskrevet i projektgrundlaget.
- Vedligeholdelsesdokumenter.
- Brugeruddannelse.

2.5 Projektplan

Nedenstående er en oversigt over de forskellige aktiviteter og faser med tilhørende estimat af ressourcer i form af tid. Estimeringen er foretaget ud fra undersøgelser og tidligere erfaringer med lignende projekter.



Figur 2.1: Sidste udkast af det såkaldt Gantt-skema der giver et overblik over projektets tidsplan med hensyn til de forskellige faser og perioder.

2.5.1 Inception

Projektets grundlæggende etablering er færdig og der er blevet fastlagt en indledende fælles vision. Derudover er de første kravspecifikationer i form af de forskellige artefakter, herunder Use-Case modellen udarbejdet, og der er blevet dannet et overblik over de betydningsfulde "*high risk*" områder.

2.5.2 Elaboration (E1)

Første iteration i en lidt længere fase, hvor der dels er udviklet centrale dele af applikationen og dels fået en dybere forståelse for systemets krav og arkitektur. Derudover er udarbejdelsen af en række nye artefakter påbegyndt som i de efterfølgende iterationer vil blive tilpasset yderligere.

2.5.3 Elaboration (E2)

I den anden iteration af elaboration er problemdomænet analyseret; en grundlæggende arkitektur fastlagt og de største risici i projektet elimineret. Derudover vil de forskellige

sekvensdiagrammer blive bearbejdet og der vil blive foretaget en analyse af GUI-designet i henhold til applikationens udvikling.

2.5.4 Elaboration (E3)

I den tredje fase af elaboration vil de sidste use cases blive bearbejdet og der vil blive foretaget de afsluttende ændringer på de forskellige modeller i henhold til udviklingen. Derudover vil der blive gjort klar til construction-fasen hvori fokus primært vil være på at udvikling og dokumentring af systemet så det kan overdrages til brugeren.

2.5.5 Construction

Der vil denne del af projektet sættes fokus på at udvikle og teste de resterende dele af systemet og dennes funktioner. Endvidere vil der blive udvikles og implementeret en database og et kørende system hos virksomheden hvorved der kan findes eventuelle oversete krav.

2.5.6 Afslutning

Den sidste del af projektet vil blive færdiggjort og eventuelle rettelser i dokumentationen vil blive foretaget så applikationen kan komme i drift.

2.6 Aktivitetsoversigt

2.6.1 Inception

Artifakter der udarbejdes i denne fase:

- Vision og den tilhørende narrative beskrivelse.
- Use-Case Model
- Use-Case Diagram (UCD)
- Use-Case Tekst (UCT)
- System Sequence Diagram (SSD)
- Operation Contract (OC)
- Glossary

Discipliner der beskæftiges med i denne fase:

- Business Modeling
  - Visualisere betydningsfylde koncepter i programmets domæne.
- Requirements
  - For at indfange funktionelle såvel som ikke-funktionelle krav.
- Design
  - Designe software objekterne.

2.6.2 Elaboration (E1)

Artifakter der udarbejdes i denne fase:

- Domain Model
- Design Model
- Software Architecture Document
- Data Model
- Implementering
- UI Prototyper

Discipliner der beskæftiges med i denne fase:

- Der arbejdes hovedsagligt i requirement-disciplinen.

2.6.3 Elaboration (E2 og E3)

Artifakter der bearbejdes i denne fase:

- Domain Model
- Design Model

- Software Architecture Document
- Data Model
- Implementering
- UI Prototyper

Discipliner der beskæftiges med i denne fase:

- Der arbejdes hovedsagligt i requirement-disciplinen.

2.6.4 Construction

Artifakter der bearbejdes i denne fase:

- Alt efter hvilke ændringer der foretages i systemet.

Discipliner der beskæftiges med i denne fase:

- Der arbejdes hovedsagligt i implementerings-disciplinen.

Formål med de forskellige aktiviteter og deres udfald vil blive beskrevet yderligere i metode-afsnittet.

2.7 Kritiske forudsætninger og risici

I det følgende vil der blive beskrevet de planlægnings-, forretnings- og ressourcemæssige risici i forbindelse med dette projekt, samt forslag til hvordan man imødekommer disse således projektet kan fuldføres i henhold til den udarbejdet projektplan.

Der skal her bemærkes at den udarbejdet liste af risici skal ses i sammenhæng med den kommende Inception-fase hvori de mest betydningsfulde risici og særlige kritiske aspekter ved projektet vil blive analyseret yderligere.

For at se en oversigt over de forskellige risici henvises der til bilag 12.1.

2.8 Delkonklusion

Der er nu blevet udarbejdet en indledende projektetablering med fokus på især formål, tidsplan og organisation. Der er her blevet klarlagt et projektgrundlag og en række mål

som samlet præcisere projektets udgangspunkt og videre forløb i samarbejde med den valgte virksomhed. I forbindelse med dette er der blevet udarbejdet en projektplan med en beskrivelse af tilhørende aktiviteter samt en oversigt over forskellige risici og deres mulig påvirkning på forløbet.

Endvidere skal der her bemærkes at der endnu ikke har været betydelig fokus på de funktionelle og mere tekniske kravspecifikationer. Den manglende fokus skyldes at den udarbejdet projektetablering skal ses i sammenhæng med Inception-fasen som påbegyndes i henhold til Unified Process metodikken i det kommende afsnit. Her vil en mindre del af de samlet funktionelle og tekniske krav blive analyseret og de vil dermed danne grundlag for videre bearbejdelse.

3 Problemformulering

Formålet med dette forløb er at besvare følgende problemformulering:

Hvordan kan der udvikles et system til håndtering og planlægning af ansattes vagter i en virksomhed?

For at besvare denne problemstilling, vil nedenstående undersøgelsesspørgsmål danne grundlag for analysen samt den efterfølgende vurdering og diskussion:

- *Hvilke tiltag kan udføres i forbindelse med udviklingsprocessen for at finde frem til brugerens krav og behov?*
- *Hvordan kan udviklingsmetodologien Unified Process (UP) og notationssproget Unified Modeling Language (UML) inddrages i forbindelse med analyse og design af en webapplikation?*
- *Hvordan kan udviklingsværktøjer som HTML5 (HTML5, CSS3 og JavaScript) samt JSP (Java Server Pages) anvendes i forbindelse med udvikling og implementering af en webapplikation?*
- *Vurdering af fordele og ulemper ved valg af JSP som teknologi i forbindelse med udvikling af en webapplikation?*

4 Metodeafsnit

Formålet med dette afsnit er, at sikre denne opgaves gentagelighed ved at introducere de metoder, der er anvendt til at bearbejde den udarbejdet problemformulering. Indledningsvis redegøres for de forskellige systemudviklingsmetoder i forhold til problemstillingerne i denne opgave, herunder vil valget af metoder og data blive belyst. Endvidere vil der i forlængelse af dette sættes fokus på de forskellige processer og metodiske tilgange af det analytiske og diskuterende aspekt i opgaven. Afslutningsvis vil incitamentet til valg af cases og afgrænsning blive behandlet.

4.1 Metode og valg af data

I forbindelse med udviklingen af et system findes der en række metodologier og værktøjer som kan anvendes. Fælles for alle softwareudviklings processer og metodologier er at de definer en liste af aktiviteter til udarbejdelse; fastsætter rækkefølge for de forskellige aktiviteter; fortæller hvad input og hvad output skal være, og så fremdeles.

Blandt de mest udbredte typer af metodologier også kaldet udviklingsprocesser findes[2]:

- Vandfaldsmodellen
    - Sekventiel udvikling hvor et projekt går fra fase til faste igennem et streng planlagt forløb.
- V-modellen
    - Kan betragtes som en udvidet vandfalds-model hvor hver fase har en tilhørende testfase.
- Agil udvikling
    - En "letvægts" udviklingsproces der gør brug af en iterativ fremgangsmåde med fokus på fleksibilitet og bæredygtig udvikling.[3]
- Iterativ udvikling
    - Korte cyklusser og faser delt op i iterationer.

[2] Fowler, Martin: UML Distilled, A Brief Guide to the Standard Object Modeling Language. Side 26-26. 3. udg. Addison-Wesley Professional, 2003. (Bog)
[3] Agil udvikling er et begreb som dækker over mange processer som deler et fælles sæt af værdier og principper defineret af Manifestet for agil software udvikling. (http://agilemanifesto.org/iso/dk/)

Ovenstående er naturligvis en meget simplificeret beskrivelse og der findes også andre måder at opdele de forskellige metoder og processer. De fleste metoder gør dog brug af en eller flere af de ovenstående metodologier i mere eller mindre omfang. Det blev hurtigt klart at henholdsvis vandfaldsmodellen og V-modellen var udelukket grunden deres meget faste, ufleksible og utidssvarende tilgang til udvikling af et system.[4] Både undertegnet og den samarbejdende virksomhed mente at der burde anvendes en metode som bestod af en iterativ og inkrementel tilgang med regelmæssig prototyper af systemet som der kunne gives feedback og eventuel tilpasses i henhold til dette. Derudover blev det vurderet at den mest hensigtsmæssige fremgang baseret ud fra tidligere erfaringer med lignende projekter ville være at vælge en metode hvori det var muligt at tilføje nye krav og ønsker senere i udviklingsforløbet.

Blandt de metodikker som overholder ovenstående krav og som også er blevet arbejdet med tidligere kan nævnes den iterative udviklingsmetode ved navn Unified Process (UP) og de to agile udviklingsmetoder Scrum og Extreme Programming (XP).

Det endelige valg faldt på Unified Process som udviklingsmetode, hvilket skyldes dens særlige fokus på risici, analyse og design i de tidligere faser af udviklingen samt dens måde at håndtere dokumentationen af hele udviklingsprocessen. Derudover skyldes valget også en række praktiske årsager herunder de andre systemudviklingsmetoders krav om at arbejde i mindre hold med særlige fastsatte roller hvorimod Unified Process med fordel kan anvendes af en enkelt person.

4.1.1 Valg af data

Det analyserende samt vurderende niveau i opgaven vil være baseret på en deduktiv tilgang, da der tages udgangspunkt i allerede eksisterende teorier og metoder, herunder Craig Larmans analyse- og designmetoder[5] samt Dr. E.F. Codd's tilgang til relationelle databaser.[6] De primære og sekundære kilder til udarbejdelsen af opgaven vil således bestå

[4] Agile and Iterative Development: A Manager's Guide af Larman indeholder en række praktiske tips i forhold til den udbredte holdning omkring vandfaldsmodellen som mislykket og fordele ved iterative udvikling,

[5] Craig Larman: Applying UML and Patterns, 3. udg. Pearson Education, 2005 (Bog)

[6] Begg, Carolyn : Database Solutions, A Step-by-Step Approach to Building Databases. 2. udg. Pearson Education, 2003. (Bog)

af teoretiske bøger og videnskabelige artikler udarbejdet af forskere og forfattere med viden og indsigt i de specifikke områder.

I forhold til analyse-delen vil der blive gjort brug af objektorienterede analyse (OOA) hvori de forskellige objekter og koncepter vil findes og videre blive beskrevet inden for domænet.

I den efterfølgende designproces vil der blive gjort brug af objektorienterede design (OOD), som baseres ud fra de krav som bliver identificeret i analysen. I denne del vil der udarbejdes, på baggrund af kriterierne fra analysen, en række modeller som skal medvirke til at beskrive og illustrere systemets opbygning og funktionalitet.

Herefter vil ovenstående implementeres igennem objektorientende programmering (OOP) hvor design objekterne implementeres igennem Java, HTML5 (HTML5, CSS3 og JavaScript), og SQL.

Dette vil ske igennem en iterative udviklingsmetode Unified Process (UP) og endvidere vil de forskellige modeller igennem analyse- og designprocessen være baseret på UML standarden, hvor de dermed vil fungere som en løbende beskrivelse af systemets funktionalitet og de forskellige aktøres sammenspil.

4.2 Beskrivelse af Unified Process og UML

Unified Process er en iterativ softwareudviklingsproces til udvikling af objektorienteret systemer. Iterativ udvikling er gået hen og blevet en fremstående praksis indenfor moderne systemudvikling og går i sin enkelthed ud på at planlægge udviklingen til at bestå af en række miniprojekter eller faser kaldet iterationer. Disse iterationer indeholder hver især særlige aktiviteter indenfor analyse, design, implementering, og test som skal munde ud i et delvist eksekverbar system der kan præsenteres for slutbrugeren så eventuel feedback kan bearbejdes og systemet tilpasses deraf.[7]

Unified Process består af fire faser i form af Inception, Elaboration, Construction og Transition hvis formål, indhold og aktiviteter vil blive beskrevet i det følgende afsnit:

[7] Fowler, Martin: UML Distilled, A Brief Guide to the Standard Object Modeling Language. Side 26-27. 3. udg. Addison-Wesley Professional, 2003. (Bog)

4.2.1 Inception

Formålet med denne del af opgaven er at indsamle, analysere og fastlægge en vision for den valgte virksomheds applikation. I denne del af analysen fokuseres der primært på brugerens grundlæggende behov og ønsker med hensyn til hvordan disse kan blive opfyldt af den udarbejdet applikation, samt defineringen af en mindre del af de krav som applikationen skal opfylde.

Blandt de artefakter der udarbejdes i denne fase:

- Vision og den tilhørende narrative beskrivelse.
  - Visionen og den efterfølgende narrative beskrivelses formål er at give en ide for applikationen samt definere de indledende og overordnet visioner for projektet til fremvisning for de interesserede parter. Det skal vise de vigtigste behov og funktioner samt indeholde en oversigt over de mest centrale krav der skal ligge til grund for den videre bearbejdelse og udvikling.
- Use-Case Model
  - Use-Case Modellen er en model, som indeholder en række elementære artefakter. Modellen er et sæt typiske scenarier for anvendelsen af et system i forhold til de primære funktionelle krav og typiske scenarier.

    Formålet med en Use-Case Model er at give et eksternt syn på systemet. En Use Case Model giver et godt billede af systemet og dens grænseflade, samt hvordan den bliver brugt. Derudover bliver modellen benyttet til at vise systemets og dens aktøres ageren i forhold til systemet.
- Use-Case Diagram (UCD)
  - Use Case Diagrammet vil illustrere de forskellige aktøres forhold og interaktion til de forskellige use cases. Således illustrere et Use Case Diagram de forskellige typer af brugere i et system og de forskellige måder

hvorledes de interagere med systemet på, samt hvordan brugergrænsefladen er opbygget.

- Use-Case Tekst (UCT)

  - Use Case teksten har til formål at belyse krav til et nyt, eller ændring af et eksisterende system. Hver enkelt Use Case indeholder en eller flere scenarier, der viser hvordan systemet skal kunne interagere med en bruger eller et andet system for at løse en specifik opgave. Derudover skrives Use Case teksten i et let forståeligt sprog, så man derved kan inddrage systemets slutbrugere i processen for derved at kunne optimere og præcisere de krav der skal kunne opfylde kundens behov.[8]

- System Sequence Diagram (SSD)

  - Supplementary specifikation (ikke-funktionelle krav) er medvirkende til at give et overblik over de forskellige krav og specifikationer som ikke nødvendigvis dækkes af use casene og de tilhørende modeller.

- Operation Contract (OC)

  - Udover Use Cases bliver Operation Contracts (OC) også brugt til at beskrive et systems adfærd. Operation Contract illustrer igennem pre-condition og post-condition gennemgående ændringer i objekter i domæne modellen i forbindelse med en handling i systemet.

    En Operation Contract er derfor et fremragende værktøj til kravs specificering og analyse, der beskriver i detaljer de ændringer som kræves af en handling i et system (med hensyn til domænet modelobjekter) uden dog at beskrive, hvordan de skal opnås.

[8] Craig Larman: Applying UML and Patterns (side 63-88) 3. udg. Pearson Education, 2005 (Bog)

Med andre ord kan selve designingen glemmes for en stund, og vi kan fokusere på selve analysen af, hvad der skal ske, snarere end hvordan den skal udføres.[9]

- Glossary
  - En liste med forklaring over særlige domain terminologier og begreber

4.2.2 Elaboration

Denne del af projektets fase er en del længere end den forgående, hvilket til dels skyldes at man udvikler centrale dele af systemet for derved at kunne få en dybere forståelse for systemets krav og arkitektur. Derudover analyseres problemdomænet; en grundlæggende arkitektur fastsættes og de største risici i projektet elimineres. Til slut arbejdes der imod en implementering af systemet, hvori de praktiske test foretages. [10]

Blandt de artefakter der udarbejdes i denne fase:

- Domain Model
  - Domain modellen (domæne modellen) er et af de mest betydningsfulde modeller indenfor OOA (objektorienterede analyse). Formålet med domain modellen er at illustrere fysiske objekter og abstrakte koncepter i form af konceptuelle klasser relateret til systemets domæne. Derved kan domain modellen med stor fordel bruges til at belyse betydningsfulde koncepter i et domæne og endvidere danne grundlag for designingen af de efterfølgende software objekter.
- Design Model
  - Design modellen har til formål at illustrere og beskrive det logiske design i form af blandt andet software klassediagrammer, sekvens diagrammer og pakke diagrammer.
- Software Architecture Document

[9] Craig Larman: Applying UML and Patterns () 3. udg. Pearson Education, 2005 (Bog)
[10] Craig Larman: Applying UML and Patterns (side 126-131) 3. udg. Pearson Education, 2005 (Bog)

- Illustrere signifikante aspekter af applikationens struktur og de forskellige lags indbyrdes forhold.

- Data Model

  - Beskriver implementeringen af en relationel database til persistent lagring af data.

4.2.3 Construction

Dette er en af de længste faser og det er her størstedelen af implementeringen foregår. Denne del af fasen forsætter ind til det endelige system er færdigudviklet og klar til at blive frigivet.

4.2.4 Transition

Transition består af afsluttende aktiviteter som ikke kan udføres iterativt, hvilket kan være dokumentering, brugeruddannelse og lignende.

5 Afgrænsning

På grund af opgavens omfangsmæssige begrænsninger er det klart, at væsentlige emner, som også har relevans for opgaven, ikke har været muligt at inddrage.

5.1 Sikkerhed

Principper og retningslinjer indenfor sikkerhed er et essentielt men også utrolig bredt emne indenfor moderne webapplikationsudvikling. Noget som kunne være yderst relevant at inddrage i denne sammenhæng er en analyse af systemets arkitektur og design med henblik på at identificere potentielle sårbare områder som kan tillade individer at kompromittere systemets sikkerhed. Her tænkes der særligt på anvendelsen af sikkerhed på de tre primære lag: netværkslaget, hostlaget og applikationslaget. Som nævnt er dette et bredt emne og derfor vil de forskellige sikkerhedsproblematikker kun blive behandlet marginalt og kun enkelte udvalgte sikkerhedsaspekter vil blive behandlet i opgaven.

5.2 Frameworks

Der findes en række forskellige frameworks indenfor webapplikationsudvikling som hver især tilbyder særlige værktøjer i forbindelse med udviklingen af et system. Nogle af de mest populære på nuværende tidspunkt kan nævnes Ruby on Rails, Django, CakePHP , AngularJS og Node.js.

Derudover bliver der igennem hele opgaven anvendt en række teorier og modeller fra studiet og der kunne i den forbindelse være yderst relevant at belyse samtlige fordele, ulemper og ekstern kritik af disse samt hvad anvendelse af andre modeller eller teorier ville have betydet for det samlede udfald af denne opgave. Dette har dog i lyset af opgavens omfangsmæssige og tidsmæssige begrænsninger ikke været muligt.

6 Inception

I den følgende del af opgaven vil der blive lagt særligt vægt på projektets grundlæggende etablering (*inception*), hvoraf de første kravspecifikationer vil blive bestemt, samt udarbejdelsen af en Use-Case Model som skal bidrage med at indkredse de forholdsvis centrale krav, emner og særlige såkaldte "*high risk*" områder. Dette skal ses i lyset af at *Inception* er en del af *Requirement-disciplinen* (kravs-disciplinen).[11]

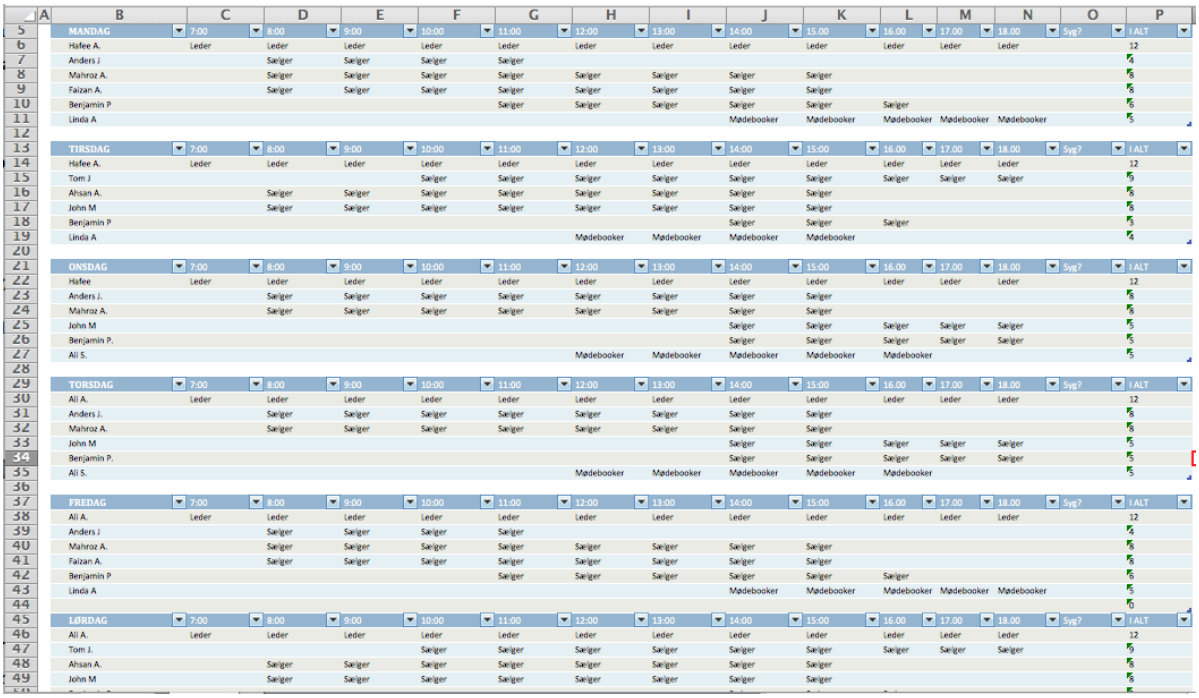
6.1 Vision

Der vil i denne vision tages afsæt i den projektetablering der blev udarbejdet tidligere og samtaler med virksomheden.

Virksomheden ønsker at få udviklet en moderne, brugervenlig og praksis applikation der kan være medvirkende til at håndtere og planlægge de ansattes vagter.

På nuværende tidspunkt planlægges de ansattes vagter ved hjælp af Microsoft Excel og Dropbox[12]. I slutningen af hver måned bliver der afholdt et såkaldt vagtmøde hvor den kommendes måneds vagter bliver fordelt blandt virksomhedens ansatte. Dette sker ved at en Leder opretter en skabelon ved hjælp af Excel som efterfølgende lægges ud på en Dropbox mappe der kan tilgås af de ansatte. Således kan de ansatte i virksomheden på ethvert tidspunkt gå ind og tjekke månedens vagter og eventuelt bytte kommende vagter.

[11] Craig Larman: Applying UML and Patterns (side 47-48) 3. udg. Pearson Education, 2005 (Bog)
[12] Gratis lagerapplikation der giver mulighed for at dele filer og mapper online mellem computere.



	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
5	MANDAG	7:00	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15.00	16.00	17.00	18.00	Syg?	I ALT
6	Hafee A.	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder		12
7	Anders J		Sælger	Sælger	Sælger	Sælger									4
8	Mahroz A.		Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger					6
9	Faizan A.		Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger					6
10	Benjamin P					Sælger	Sælger	Sælger	Sælger	Sælger	Sælger				5
11	Linda A								Mødebooker	Mødebooker	Mødebooker	Mødebooker	Mødebooker		5
12															
13	TIRSDAG	7:00	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00	16.00	17.00	18.00	Syg?	I ALT
14	Hafee A.	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder		12
15	Tom J				Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger		6
16	Ahsan A.		Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger					6
17	John M		Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger					6
18	Benjamin P								Sælger	Sælger	Sælger				3
19	Linda A						Mødebooker	Mødebooker	Mødebooker	Mødebooker					4
20															
21	ONSDAG	7:00	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00	16.00	17.00	18.00	Syg?	I ALT
22	Hafee	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder		12
23	Anders J.		Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger					6
24	Mahroz A.		Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger					6
25	John M								Sælger	Sælger	Sælger	Sælger	Sælger		5
26	Benjamin P.								Sælger	Sælger	Sælger	Sælger	Sælger		5
27	Ali S.						Mødebooker	Mødebooker	Mødebooker	Mødebooker	Mødebooker				5
28															
29	TORSDAG	7:00	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00	16.00	17.00	18.00	Syg?	I ALT
30	Ali A.	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder		12
31	Anders J.		Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger					6
32	Mahroz A.		Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger					6
33	John M								Sælger	Sælger	Sælger	Sælger	Sælger		5
34	Benjamin P.								Sælger	Sælger	Sælger	Sælger	Sælger		5
35	Ali S.						Mødebooker	Mødebooker	Mødebooker	Mødebooker	Mødebooker				5
36															
37	FREDAG	7:00	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00	16.00	17.00	18.00	Syg?	I ALT
38	Ali A.	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder		12
39	Anders J		Sælger	Sælger	Sælger	Sælger									4
40	Mahroz A.		Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger					6
41	Faizan A.		Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger					6
42	Benjamin P					Sælger	Sælger	Sælger	Sælger	Sælger	Sælger				5
43	Linda A								Mødebooker	Mødebooker	Mødebooker	Mødebooker	Mødebooker		5
44															
45	LØRDAG	7:00	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00	16.00	17.00	18.00	Syg?	I ALT
46	Ali A.	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder	Leder		12
47	Tom J.				Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger		6
48	Ahsan A.		Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger					6
49	John M		Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger	Sælger					6

Figur 6.1: Følgende er en illustration over hvordan et typisk vagtskema ser ud efter fordelingen af vagter blandt de ansatte.

Udover månedens vagtplan har de ansatte i virksomheden også adgang til forskellige filer med kontaktoplysninger, ferieplaner og en række andre dokumenter.

Ovenstående er en meget uprofessionel, uhensigtsmæssig og upraktisk løsning for virksomheden på grund af en række årsager:

- I visse tilfælde opdateres vagtplanen ikke da filerne kan være åben flere steder på samme tid, hvilket medfører såkaldte ”conflicted copies”.
- Kræver en masse ressourcer i form af tid og papir.
- Lav datasikkerhed.
- Inkonsistent information.
- Uholdbar løsning i længden.

Derfor ønsker virksomheden at få udviklet et nyt system der kan være medvirkende til at løse de ovenstående problemstillinger. Virksomheden ønsker et system som kan bruges af alle de ansatte og som kan oprette, slette og bytte vagter samt oprette ferieplaner og hente nødvendige kontaktinformationer. Systemet skal udvikles således at det ikke er afhængig

af platform eller sted så de ansatte derved kan tilgå det fra der hvor de nu engang befinder sig. Endvidere skal det være muligt for virksomhedens ledere at søge efter medarbejdere, godkende ferie- og vagtplaner samt oprette og slette medarbejdere.

6.1.1 Narrativ beskrivelse

Vi befinder os i slutningen af måneden og virksomhedens leder skal til at oprette en ny vagtplan til medarbejderne så arbejdsfordelingen for den kommende måned kan begynde.

Lederen logger ind på den nye webapplikation med sine brugeroplysninger og påbegynder oprettelsen af den nye vagtplan. I forbindelse med dette indtaster lederen en række oplysninger såsom periode og dage. Herefter bekræfter lederen den nye vagtplan og den lægges dermed ud i systemet til virksomhedens ansatte.

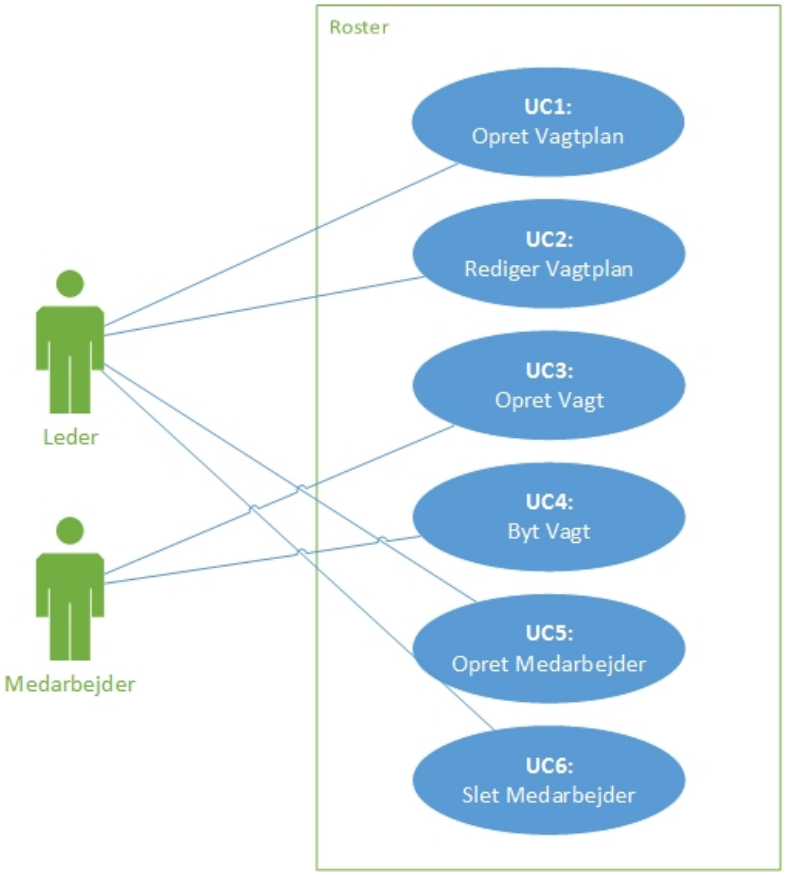
En af virksomhedens ansatte sidder i toget på vej til et ærinde. Via sin smartphone kan vedkommende se at der er oprettet en ny vagtplan for den kommende måned. Den ansatte ved allerede nu hvilke dage vedkommende ønsker at arbejde. Derfor påbegynder den ansatte oprettelsen af nye vagter via applikationen. Først vælger han den nye vagtplan og derefter vælger han de ledige dage samt arbejdstidspunkter.

For yderligere informationer omkring applikationens andre krav og funktioner henvises der til Supplementary Specification, hvor en lang række af de ikke-funktionelle krav vil blive beskrevet detaljeret.

6.2 Use-Case Model

Use-Case Modellen indeholder som beskrevet tidligere en række elementære artefakter, i form af fire modeller som vil blive beskrevet i det følgende.

6.2.1 Use Case Diagram



Figur 6.2: Det indledende Use Case Diagram der illustrere de forskellige aktøres forhold og interaktion til de forske use cases. *(Version 1.0, 24-10-2014, Inception)*

Use-Case tekst

UC1: Opret Vagtplan

Scope: System

Level: Brugerniveau

Primary Actor: Leder

Stakeholders og Interests:

- Cheferne: Ønsker et overblik over virksomhedens daglige ledelse og dennes drift, økonomi og arbejdsmæssige ressourcer.
- Skat: Er interesseret i virksomhedens og de ansattes økonomi.
- Leder: Ønsker at der bliver oprettet en vagtplan i henhold til de forventninger virksomheden har til de ansatte, så de kan forholde sig til den kommende måneds arbejde.
- Ansatte: Ønsker oprettelse af en vagtplan så de kan planlægge den kommende måneds arbejde.

Precondition: Lederen er identificeret og autentificeret af systemet. Lederen befinder sig på systemets startskærm.
Postcondition (Succes Guarantee): En vagtplan blev oprettet og gemt i systemet.

Main Success Scenario:

1. Lederen starter oprettelsen af ny vagtplan.
2. Systemet beder om indtastning af informationer omkring vagtplanen.
3. Lederen indtaster de nødvendige informationer såsom periode, antal timer, antal dage og antal medarbejdere på vagtplanen.
4. Lederen bekræfter oprettelsen af den nye vagtplan.
5. Vagtplanen oprettes og gemmes i systemet.

Extensions

2a. Invalid eller manglende information:

1. Systemet advarer om fejl i indtastningen eller mangel på nødvendig information.

5a. Vagtplanen bliver ikke gemt i systemet.

1. Systemet fejler i at oprette forbindelse til databasen.

UC2: Rediger Vagtplan

Main Success Scenario: Lederen logger ind i systemet og anmoder om en tidligere vagtplan. Lederen bekræfter at denne vil redigere i vagtplanen og de ønsket redigeringer foretages. Herefter bekræfter lederen de indtastet redigeringer og systemet gemmer den opdateret vagtplan.

Alternate Scenarios: Hvis systemet ikke formår at gemme de indtastet oplysninger meddeler systemet lederen om en opstået fejl.

Hvis der er fejl i indtastningen eller mangler nødvendig information beder systemet om at rette fejlen før vagtplanen kan opdateres.

UC3: Opret Vagt

Main Success Scenario: Medarbejderen logger ind i systemet og anmoder om den kommende måneds vagtplan. Medarbejderen vælger de ønsket dage og antal timer blandt de ledige vagter i vagtplanen. Herefter bekræfter medarbejderen de valgte vagter og de indtastet informationer oprettes og gemmes i systemet.

Alternate Scenarios: Hvis medarbejderen anmoder om en vagtplan som endnu ikke er tilgængelig meddeler systemet om manglende vagtplan.

Hvis medarbejderen vælger en eller flere vagter som ikke er ledige meddeler systemet om at vælge vagter som er ledig.

Hvis systemet mister forbindelse til det eksterne system eller på anden vis ikke formår at gemme de indtastet oplysninger meddeler systemet om fejlen.

UC4: Byt Vagt

Medarbejderen logger ind i systemet og vælger den vagt vedkommende ønsker byttet. Herefter vælger medarbejderen den ansat i virksomheden vedkommende ønsker vagten byttet med hvorefter der bliver sendt en anmodning om vagtbytte af sted til den ansatte. Systemet bekræfter afsendelsen og medarbejderen skal nu afvente en bekræftelse fra den ansatte.

UC5: Opret Medarbejder

Lederen logger ind i systemet og påbegynder oprettelse af medarbejder. Herefter bliver de nødvendige informationer såsom navn, adresse, fødselsdato samt telefon- og mobilnummer indtastet. Herefter bekræfter lederen oprettelsen af medarbejderen hvorefter systemet opretter og gemmer de nye informationer i databasen.

UC6: Slet Medarbejder

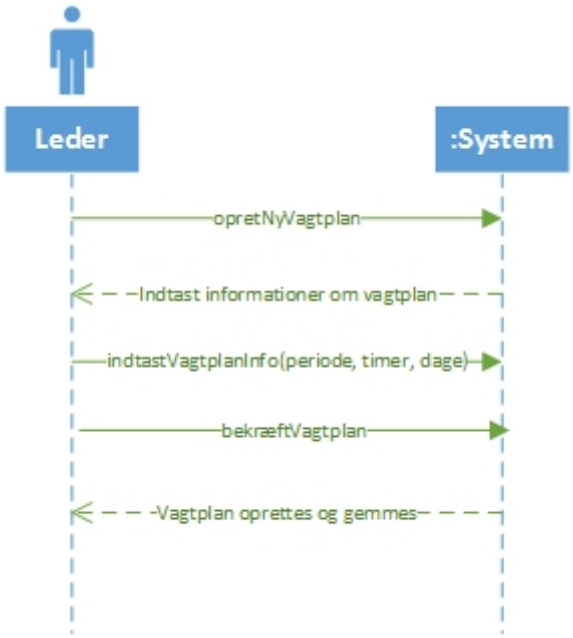
Lederen logger ind i systemet og påbegynder sletningen af medarbejder. Den ønsket medarbejder bliver fundet og der bekræftes at man ønsker at slette vedkommende fra systemet. Systemet sletter herefter medarbejderen fra systemet og bekræfter hermed den foretaget ændring.

System Sequence Diagram

Et System Sequence Diagram (SSD) anvendes til at vise et bestemt forløb indenfor en Use Case, samt de eksterne aktører der interagere direkte med systemet, og dets begivenheder. Man kan derfor sige at SSD'en er en visualisering af de interaktioner der bliver beskrevet i Use Case teksten.

Derved er det en utrolig effektiv og praktisk artefakt, der grafisk illustrere input- og output-handlinger relaterede til systemet hvilket gøre det til et meget fundamental diagram indenfor objekt orienterede design (OOD) når man vil analysere et systems adfærd.

UC1: Opret Vagtplan



Operation Contract

Contract C01: opretNyVagtplan

Operation: opretNyVagtplan()

Cross References: Use Cases: Opret Vagtplan

Preconditions: System har identificeret og autentificeret brugeren.

Postconditions:

- En instans vp af Vagtplan blev skabt (instans skabelse).
- Attributter til vp blev initialiseret.
- v blev associeret til en List (association dannet).

Contract C02: redigerVagtplan

Operation: redigerVagtplan(vagtplanID: VagtplanID)

Cross References: Use Cases: Rediger Vagtplan

Preconditions: System har identificeret og autentificeret lederen til at foretage de nødvendige ændringer. Lederen har fundet og valgt den ønsket vagtplan der skal redigeres.

Postconditions:

- Den ønskede instans vp af Vagtplan blev fundet.
- Attributter til vp blev ændret (attribut modifikation).
- De ønskede ændringer af vp blev gemt i systemet.

Contract C03: opretVagt

Operation: opretVagt(vagtplanID: VagtplanID)

Cross References: Use Cases: Opret Vagt

Preconditions: System har identificeret og autentificeret brugeren.

Postconditions:

- En instans v af Vagt blev skabt (instans skabelse).
- Attributter til v blev initialiseret.
- v blev associeret til en Vagtplan der matcher vagtplanID (association dannet).

6.2.2 Supplementary Specifikation (Ikke-funktionelle krav)

Nedenstående beskrives, de ikke-funktionelle krav, i henhold til FURPS+ modellen af Robert Grady der bygger og kategorisere sin krav i henhold til functionality (funktionalitet), usability (brugervenlighed), reliability (Systempålidelighed), performance (ydeevne), supportability (vejledning) og en række andre krav.[13]

Functionality

I forhold til applikationens funktionalitet er der en række områder som bør fremhæves på baggrund af systemets indhold og karakter.

[13] Craig Larman: Applying UML and Patterns (side 101-107) 3. udg. Pearson Education, 2005 (Bog)

Sikkerheden

Først og fremmest skal systemet indeholde en række sikkerhedsformer og procedurer for at sikre mod forskellige former for risici.

I forhold til de interne skal der være en række sikkerhedsformer på systemet så det ikke er sårbar over for angreb udefra i form af virus og hacking. Endvidere skal det ikke være muligt for en tredjepart at kunne få adgang til fortrolige personoplysninger fra systemet. Her tænkes der særligt på de mest typiske metoder der bliver brugt til at infiltrere en webbaseret applikation, nemlig cross-site scripting, SQL injection og Denial-of-service angreb. Desuden skal der sørges for backup af databasen på en sikker og forsvarlig måde, muligvis i samarbejde med andre (ekstern backup).

Logning og behandling af fejl (error)

Alle system- og fejlmeddelelser skal logges og lagres.

Usability

I forbindelse med udviklingen af applikationen skal der lægges vægt på en rækker områder indenfor brugervenlighed der tilsammen skal være medvirkende til at skabe et enkelt, intuitiv og responsive brugergrænseflade der kan tilpasses og bruges effektivt på forskellige enheder.

Simplicitet

Et vigtigt aspekt i forbindelse med udviklingen af en brugervenlig webapplikation er at holde tingene så simpel som muligt. Dette kan gøres ved at følge et sæt af regler og holde sig indenfor disse regler igennem hele design- og udviklingsprocessen.

Her kan der blandt andet nævnes valget af særlige farver til forskellige elementer og korrekt valg af funktioner til særlige situationer, samt opbygning og strukturen af selve applikationen. Applikationen bør i høj grad struktureres på en måde der er logisk og giver mening for slutbrugeren. Det kan for eksempel være særdeles forvirrende for brugeren hvis vedkommende skal foretage en masse valg eller hvis der bruges atypiske måder at håndtere standard situationer.

Holde brugeren informeret

Det er uundgåeligt at brugeren under visse omstændigheder enten er nødsaget til at vente på applikationen eller at der opstår andre uventet situationer. Den bedste måde at håndtere dette på er holde brugeren informeret omkring situationen og eventuelt fortælle hvornår brugeren kan fortsætte på sin brug af applikationen.

Reliability

Systemets oppetid

Der skal arbejdes på at holde en høj "oppetid" på systemet da det vil være utrolig skadeligt hvis der opstår nedbrud og systemet derved er utilgængeligt i længere tid.

Lav reparations tid

Systemet skal kunne være relativ nem at reparere så man derved ved eventuelle system nedbrud kan få det til at køre igen hurtigt, og derved formindske den tid systemet er utilgængeligt så meget som muligt.

Håndtering af nedbrud og lignende

Der skal udarbejdes en plan og vejledning til brugeren omkring hvordan og hvorledes denne skal agere i tilfælde af nedbrud i systemet. Desuden skal der i tilfælde af nedbrud være mulighed for at gennemfører handlinger på en række andre måder, for eksempel i form af såkaldte "offline" løsninger. På denne måde vil der altid være en mulighed for at forsætte arbejdet på trods af nedbrud og fejl, hvilket vil mindske gene for både kunde og bruger.

Performance

Kapaciteten

I forhold til kapaciteten af systemet er det nødvendigt at vide hvor stort pres systemet skal kunne klare og derfor er det nødvendigt at vide hvor mange brugere der vil benytte sig af systemet og hvor mange samtidig brugere systemet kan klarer. Derfor skal systemet være sat op til at kunne klarer de mange besøgende og dermed undgå ventetider og nedbrud. Endvidere skal der er også være muligheder for eventuelle udvidelser af kapaciteten.

Lav transaktionstid

For at maksimere brugerens oplevelse, skal det være relativt hurtigt og nemt at gennemfører en transaktion. Svartiden på systemet skal derfor være så lav som muligt.

Supportability

Kompabilitet

Applikationen skal være platformsuafhængig og skal kunne bruges på både stationære og mobile enheder på tværs af skærmstørrelser. Der skal ikke kræves særlig software eller installation for at bruge applikationen.

Vedligeholdelse og udvidelsesvenlighed

I forhold til vedligeholdelsen og udvidelsesvenligheden af systemet er også vigtige faktorer da det således vil være nemmere hvis der skulle blive brug for yderligere udvidelser af systemet. Desuden vil en grundig vedligeholdelse betyde at systemet vil blive opdateret med de nyeste informationer omkring virksomhedens arbejdsforhold og så videre.

(+) Andre vigtige ikke-funktionelle faktorer

Udover de ovenstående ikke-funktionelle krav er der også en række andre vigtige faktorer der skal tages hensyn til og som skal tages med i overvejelserne ved udvikling af systemet.

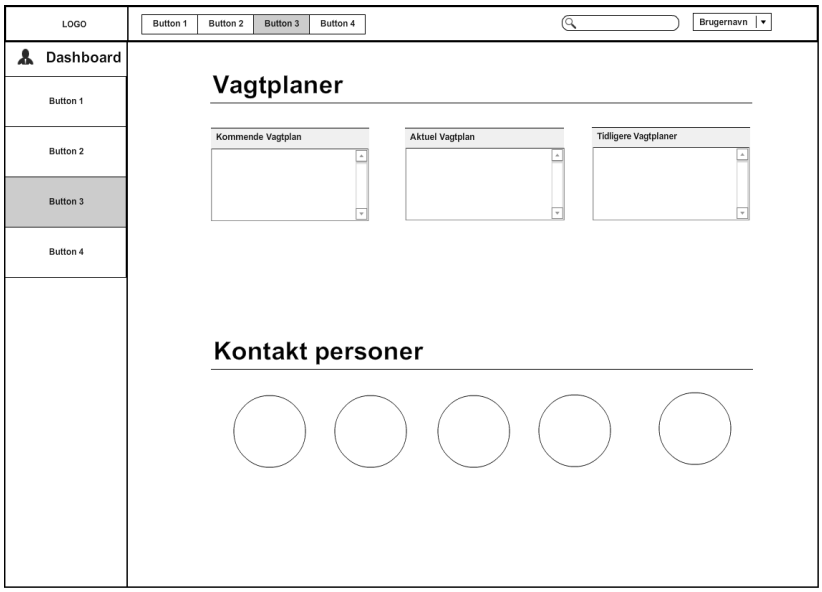
Rent juridisk er der en række lovmæssige krav som der skal tages højde for med hensyn til persondataloven og en række begrænsninger dette medfører.

Ovenstående forskellige faktorer taget i betragtning vil der i forbindelse med dette projekt især lægges vægt på kvalitetsfaktorer såsom ”Correctness”, ”Reliability” ”Usability” og ”Interoperability” da det endelig produkt skal fremstilles til en virksomhed som skal kunne gøre brug af det på daglig basis i forbindelse med den daglige planlægning af arbejdet, uden at det skal kunne kræve særlig kompetencer indenfor IT.

6.2.3 User Interface (UI) Prototyper

Der i det følgende blevet udarbejdet nogle prototyper af et muligt User Interface (Brugergrænseflade) for applikationen. Prototyperne er lavet i samarbejde med virksomheden og i henhold til de aftalte kriterier og særlige ikke-funktionelle krav.

Der er en række formål med at skabe disse såkaldte UI prototyper, herunder at udarbejde en artefakt der gør det muligt at undersøge problemområdet og finde mulige løsninger med samarbejdspartneren; for at få en forståelse for hvordan brugergrænsefladen for applikationen bør se ud; for at skabe et muligt fundament for den videre udvikling uden at bruge ressourcer på at skrive kode.



Figur 6.3: En prototype over applikationens startside (dashboard).

Ovenstående figur er en prototype over den startside (*dashboard*) som bliver mødt brugeren af applikationen efter denne er logget ind. Startsiden skal give et overblik over de vigtigste funktioner bl.a. vagtplaner, kontaktpersoner samt de forskellige menuer og valgmuligheder.

Figur 6.4: Figur over er en User Interface (UI) prototype af applikationens login-side.

Figur 6.3 er en User Interface (UI) prototype af applikationens login-side som enhver bruger møder som det første.

6.2.4 Definitioner

Term	Definition og Information	Format	Validerings Regler	Aliasser
Leder	En leder i virksomheden er den overordnet for de ansatte.			Chef, Ejer.
Medarbejder	En der er ansat i virksomheden til at udføre et arbejde i form af salg eller mødebooking.			Ansat, Sælger, Mødebooker.
Responsive	En responsive webapplikation er udviklet til at kunne tilpasses og køres på enheder med forskellige skærmstørrelse uden gene for brugeren.			Responsive webdesign (RWD), Mobile First.

Tabel 6.1: Ovenstående tabel giver et overblik og forklaring over terminologier og særlige betegnelser som bliver brugt i forskellige sammenhænge. *(Version 1.0, 01-10-2014, Inception)*

6.3 Delkonklusion

Der kan hermed konkluderes at de forskellige aktiviteter for inception-fasen er udarbejdet og fuldendt. Der er blevet indsamlet, analyseret og fastlagt en indledende vision for den valgte virksomheds system baseret ud fra de tidligere analyser der er foretaget for casen. Brugerens behov og ønsker med hensyn til hvordan disse kan blive opfyldt af det udarbejdet system, samt defineringen af en række krav som systemet skal opfylde er desuden blevet gennemarbejdet og fundet.

Derudover er de første kravspecifikationer blevet bestemt, samt udarbejdet en Use Case Model som har bidraget med at indkredse de centrale krav, emner og særlige såkaldte ”high risk” områder.

Endvidere har projektet efter fasens fuldendelse givet et betydeligt bedre indsigt i det videre forløb i forhold til den iterative udvikling.

Ovenstående viser i praksis hvilke tiltag der kan foretages i forbindelse med en udviklingsproces for at finde frem til brugerens krav og behov.

7 Elaboration – 1. Iteration (E1)

I det følgende vil den første fase (*iteration*) af *elaboration* blive udarbejdet i henhold til Unified Process og den iterative udvikling samt den udarbejdede problemstilling.

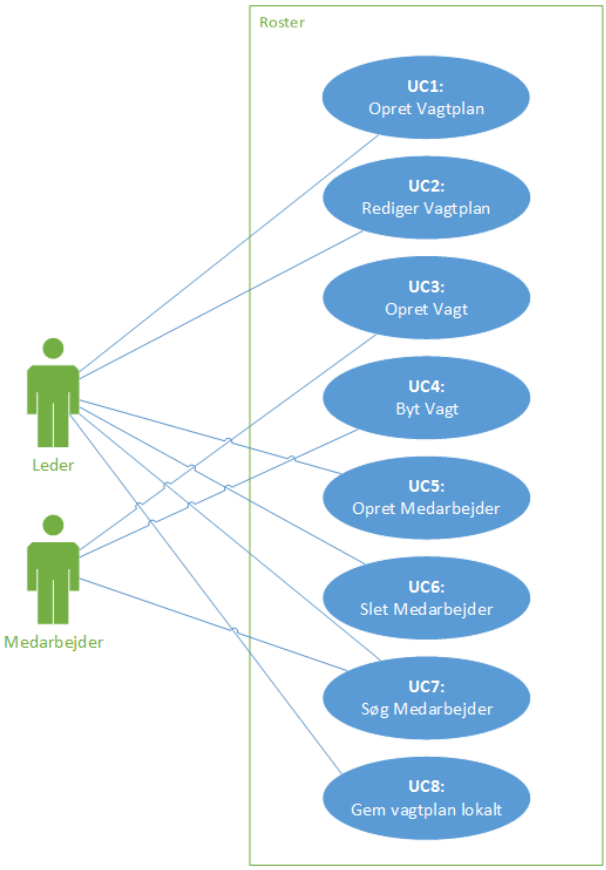
Formålet med den denne første *iteration* i *elaboration* er først og fremmest at forbedre og klarificere kravene fra *inception*. Dernæst vil *kerne-arkitekturen* blive analyseret, samt udarbejdet yderligere artefakter i form af de forskellige modeller, såsom domain- og designmodellen.

7.1 Revideret Use-Case Model

I henhold til den iterative udvikling og med fokus på de undersøgelser og samtaler der er foretaget med virksomheden er der blevet foretaget en revidering af Use Case modellen hvor to nye use cases er blevet udarbejdet. Derudover arbejdes der hen imod fuldførelse og implementering af yderligere use cases også i henhold til Unified Process.[14]

[14] Craig larman uml (Side 96)

7.1.1 Use-Case Diagram



7.1.2 Use-Case Tekst

UC2: Rediger Vagtplan

Scope: System

Level: Brugerniveau

Primary Actor: Leder

Stakeholders og Interests:

- Cheferne: Ønsker et overblik over virksomhedens daglige ledelse og dennes drift, økonomi og arbejdsmæssige ressourcer.
- Leder: Ønsker at den offentliggjorte vagtplan er opdateret i henhold til de forventninger virksomheden har til de ansatte, så de kan forholde sig til den kommende måneds arbejde.

- Ansatte: Ønsker en vagtplan der afspejler virksomhedens reelle ønsker så de kan planlægge den kommende måneds arbejde.

Precondition: Lederen er identificeret og autentificeret af systemet. Lederen befinder sig på oversigten over vagtplaner.
Postcondition (Succes Guarantee): Den ønsket vagtplan blev ændret og gemt i systemet.

Main Success Scenario:

1. Lederen anmoder om en tidligere oprettet vagtplan.
2. Lederen bekræfter at denne ønsker at redigere i en tidligere oprettet.
3. De ønskede ændringer i vagtplanen foretages.
4. Lederen bekræfter ændringer i vagtplanen.
5. Vagtplanen redigeres og gemmes i systemet.

Extensions

3a. Invalid eller manglende information:

1. Systemet advarer om fejl i indtastningen eller mangel på nødvendig information.

5a. Vagtplanen bliver ikke gemt i systemet.

1. Systemet fejler i at oprette forbindelse til databasen.

UC7: Søg Medarbejder

Main Success Scenario: Lederen logger ind i systemet og søger efter en medarbejders for- eller efternavn. Systemet søger efter medarbejderen blandt oprettede medarbejdere i databasen. Den ønsket medarbejder findes og returneres til lederen.

Alternate Scenarios: Hvis systemet ikke formår at finde den ønsket medarbejder informeres brugerne om at medarbejderen ikke findes.

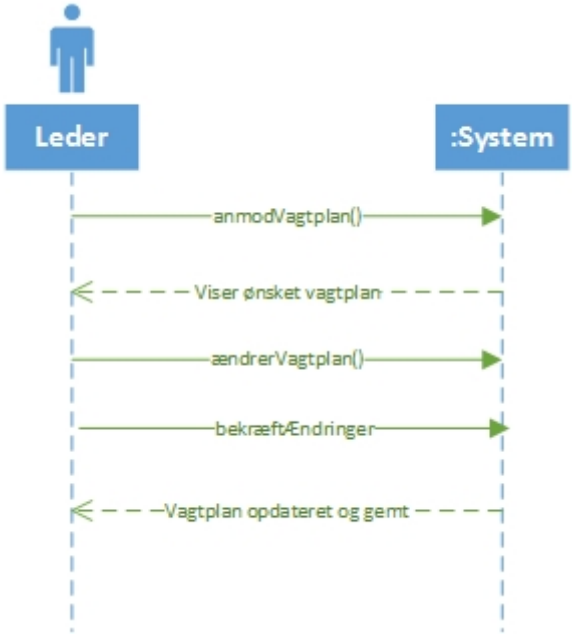
UC8: Gem vagtplan lokalt

Main Success Scenario: Lederen logger ind i systemet og anmoder om en tidligere oprettet vagtplan. Lederen beder systemet om at gemme den valgte vagtplan lokalt. Herefter vælger lederen den ønskede sti vagtplanen skal gemmes på det lokale system. Systemet gemmer vagtplanen lokalt.

Alternate Scenarios: Hvis systemet ikke formår at gemme vagtplanen lokalt meddeler systemet lederen om en opstået fejl.

7.1.3 System Sequence Diagram (SSD)

UC2: Rediger Vagtplan

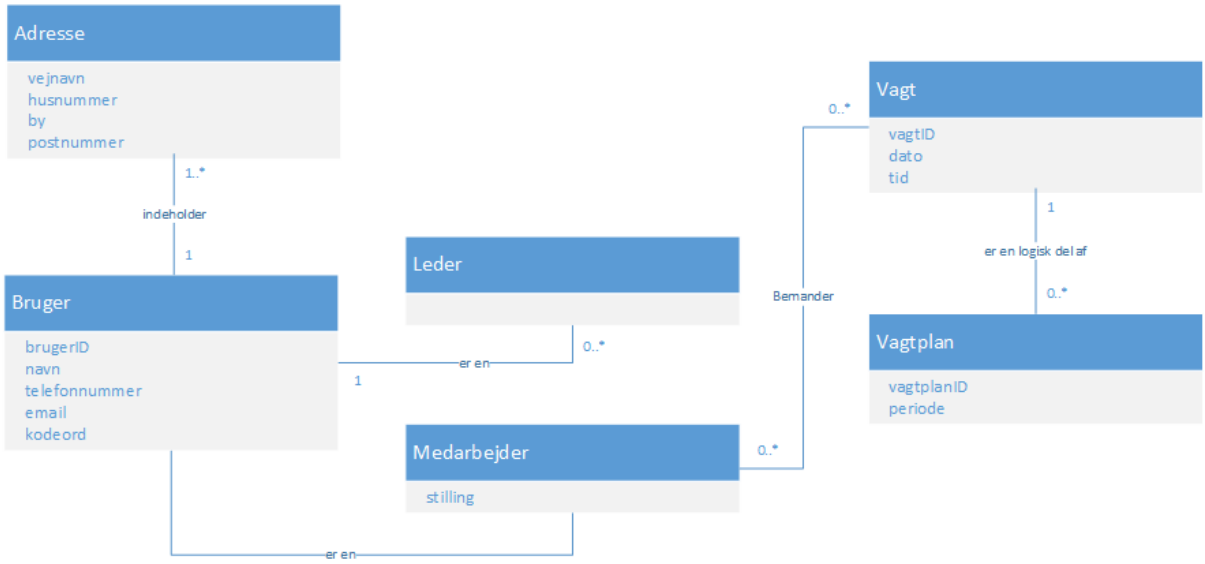


7.2 Domain Modellen

Domain modellen er lavet ud fra de tanker og analyser der er gjort i forbindelse med udarbejdelsen af de indledende kravspecifikationer herunder de forskellige use cases samt medfølgende diagrammer og artefakter.

Der er gjort en række overvejelser og refleksioner forhold til at skabe en bæredygtig arkitektur i forbindelse med fremstillingen af ovenstående domain model. Derudover har der i denne iteration været særligt fokus på "brugeren".

Først og fremmest er der blevet gjort brug af en såkaldt *description class*[15] (Bruger), hvilket er en klasse som indeholder nogle nødvendige informationer der beskriver noget andet, i dette tilfælde informationer såsom navn, telefonnummer og anden nødvendig



information til klasserne "Leder" og "Medarbejder". Således bruges et *pattern*[16] ved navn *Item-descriptor* for at undgå redundans og gentagelse af informationer.

Herudover er der blevet lavet en separat klasse for informationer vedrørende brugerens adresse i form af klassen "Adresse" i stedet for det måske mere naturlig valg; som en attribut i "Bruger". Dette er sket i henhold til både David C. Hay[17] Craig Larmans

[15] Craig Larman: Applying UML and Patterns (side 147-149) 3. udg. Pearson Education, 2005 (Bog)
[16] Betegnes indenfor software udvikling som en løsning på en almindelig forekommende problemstilling indenfor en given kontekst i software design.
[17] C. Hay, David: Data Model Patterns. (side 147-149) 1. udg. Dorset House, 2011. (Bog)

vejledning[18] til såkaldte *data type classes* hvori der bliver rådet til at skabe særskilte klasser for informationer såsom adresser da de indeholder separate sektioner og da det er et komplekst domain koncept.

I forbindelse med ovenstående valg opstod der en yderligere problemstilling som krævede en beslutning. For hvorvidt burde der laves en særskilt klasse for attributter såsom "brugerID", "navn" og "telefonnummer", da attributterne jo ligeledes opfylder kriterierne for en *non-primitive* klasse. Her vælger Larman dog at give plads til flere valgmuligheder og den endelig beslutning afhænger af en række faktorer. For simplicitetens skyld er der derfor ikke blevet lavet separate såkaldte *non-primitive* klasser for andre af disse former for attributter.

7.3 Design model

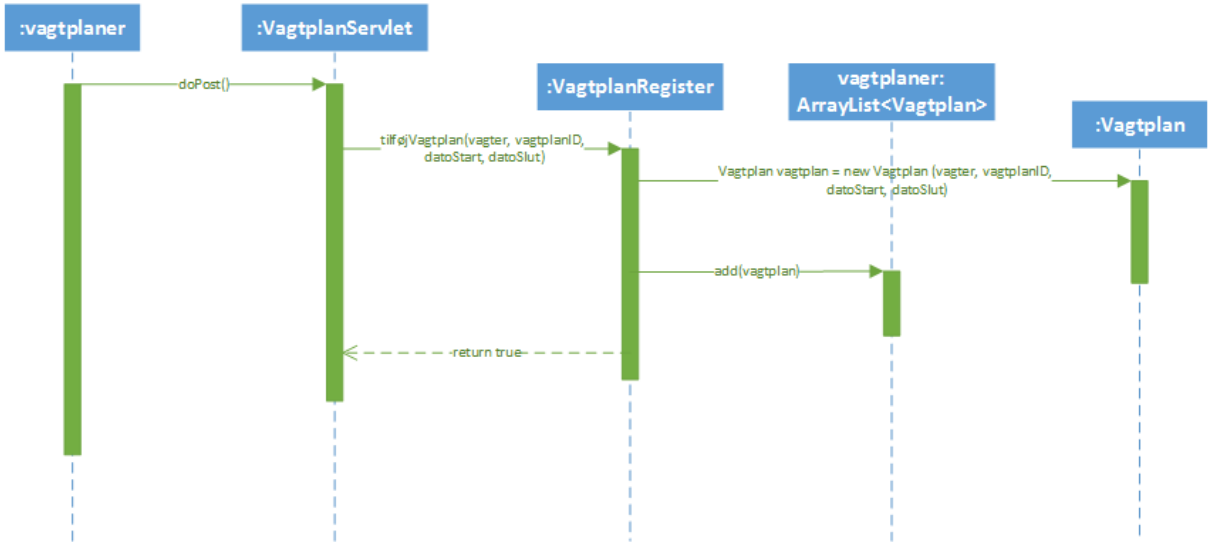
7.3.1 Sequence Diagram

Der vil i det følgende udarbejdes et *sequence diagram* (sekvens diagram) der skal illustrere hvordan de forskellige objekter interagere og arbejder med hinanden, ved hjælp af beskeder i form af metodekald med parametre og returværdier.

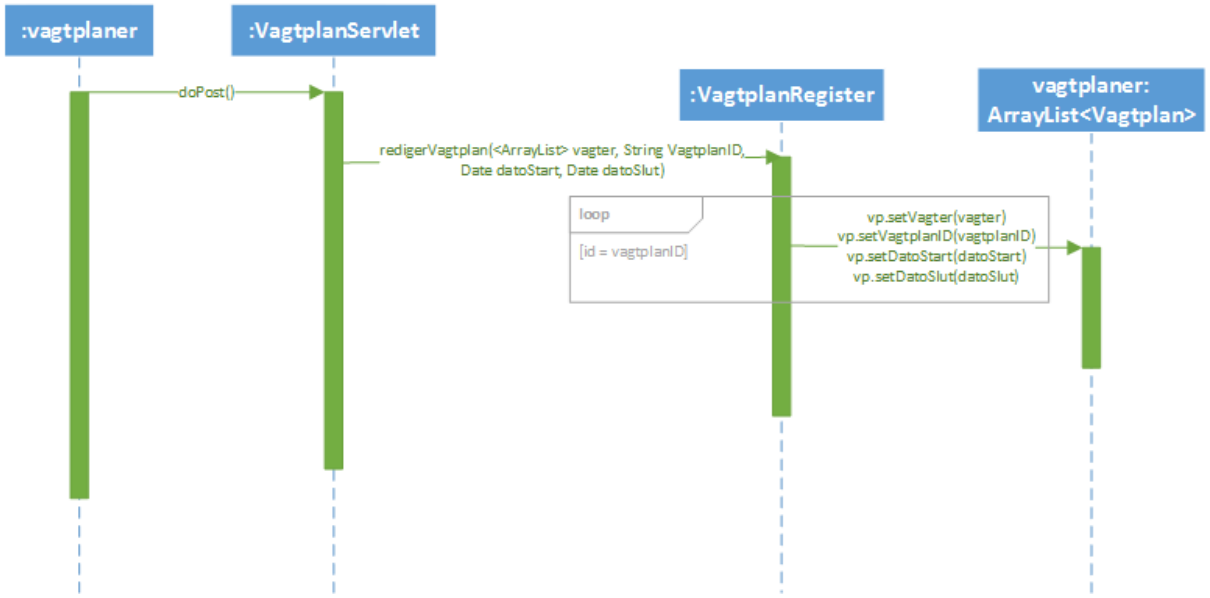
I sekvens diagrammet følger vi de forskellige processer de enkelte metoder udfører. Vi får endvidere indsigt i hvordan program strukturen er opbygget og hvordan de enkelte klasser arbejder sammen og hvordan objekter skabes.

UC1: Opret Vagtplan

[18] Craig Larman: Applying UML and Patterns (side 164-166) 3. udg. Pearson Education, 2005 (Bog)



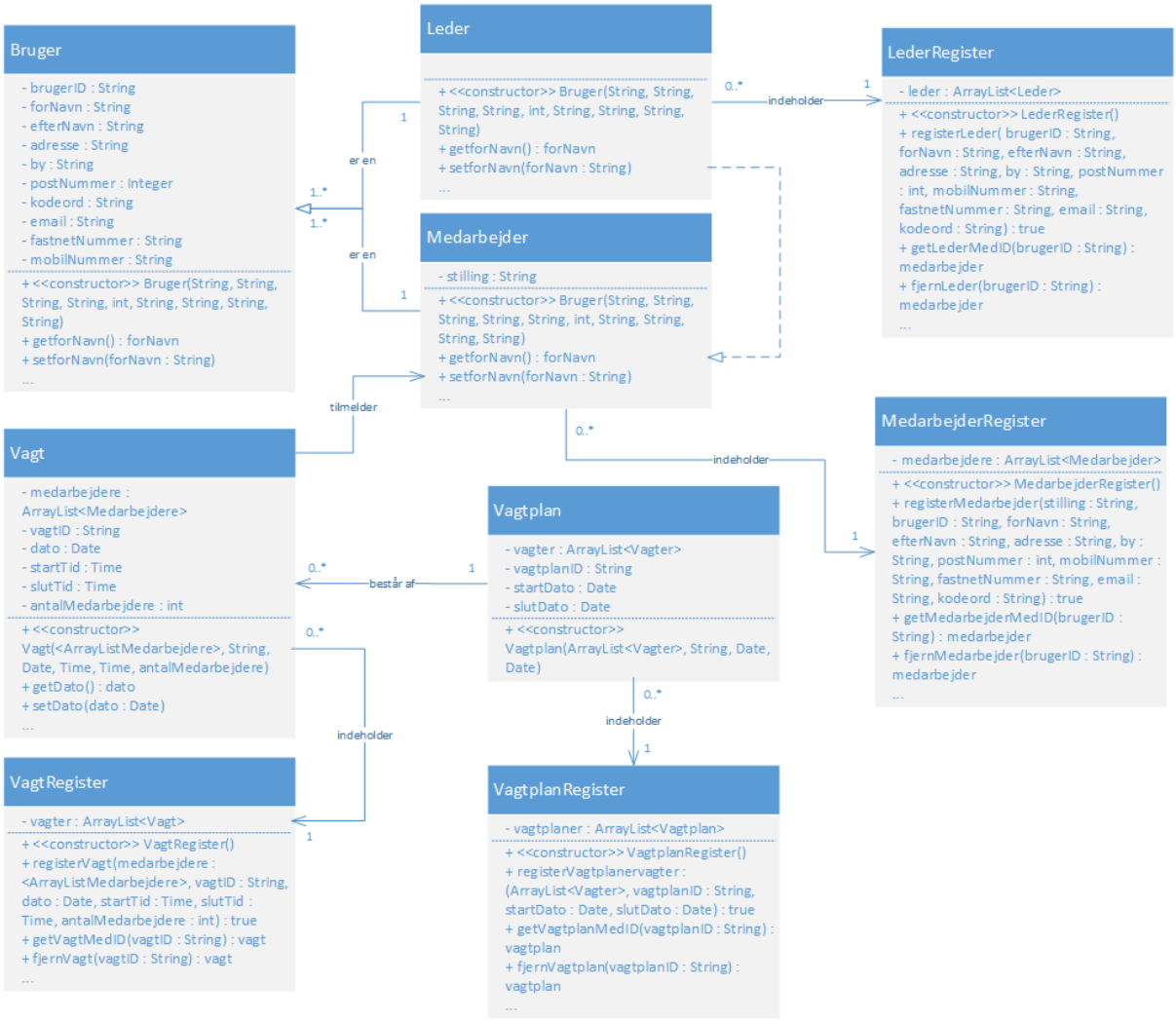
UC2: Rediger Vagtplan



7.3.2 Design Class Diagram

Et Design Class Diagram beskriver statiske relationer imellem forskellige former for software klasser og objekter. Derudover giver klasse diagrammet et indblik i de forskellige klassers attributter og metoder samt de begrænsninger som de bliver påført gennem måde objekterne er sammensat på. [19]

[19] Fowler, Martin: UML Distilled, A Brief Guide to the Standard Object Modeling Language. Side 36. 3. udg. Addison-Wesley Professional, 2003. (Bog)



Klassediagrammerne er blevet udarbejdet i lyset af de tidligere fremstillede Sequence Diagrammer hvor en række klasser udsprang med tilhørende metoder. Der er i ovenstående klassediagram først og fremmest blevet lagt fokus på at få udviklet de centrale dele af systemet herunder forretningslogikken (business logic), som derved kan danne grundlag og fundament for den videre udvikling i den iterative proces.

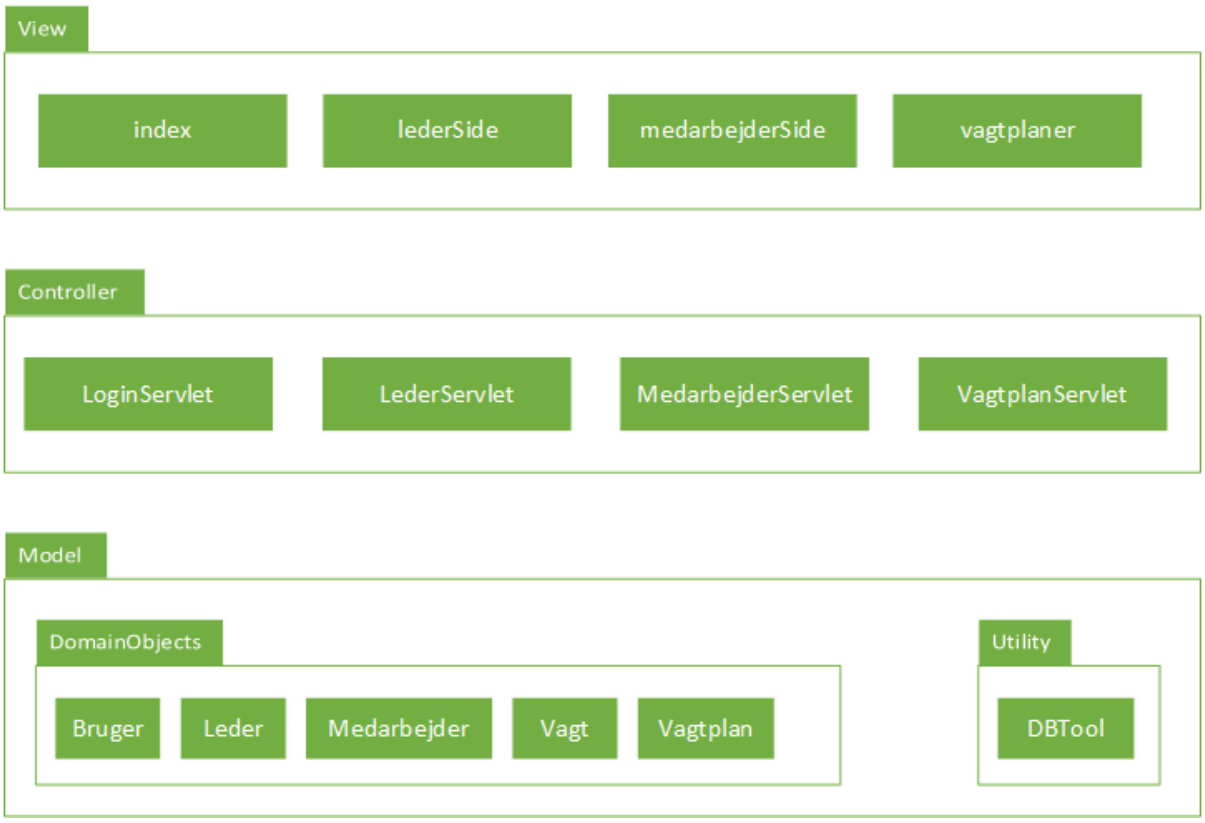
Der er blevet udvikles basale modelklasser i form af ”Bruger” og dens tilhørende subklasser ”Leder” og ”Medarbejder”. Disse klasser indeholder informationer og brugeroplysninger i form af navn, adresse, kontaktinformationer, og personlig kodeord. Derudover er der ligeledes udviklet modelklasser ”Vagt”, ”Vagtplan” som indeholder informationer omkring vagter og deres tilhørende vagtplaner.

Endvidere er der blevet udviklet såkaldte Holder-klasser (Register klasserne) som indeholder datastrukturer til lagring af modelobjekterne.

Der var en lang række overvejelser under klassediagrammets udviklingsproces herunder hensyntagen til de forskellige patterns så der kunne designes en bæredygtig arkitektur. Disse overvejelser vil blive uddybet i forbindelse med 2. Iteration af Elaboration (E2) hvor klassediagrammet vil blive bearbejdet yderligere.

7.4 Software Architecture Document

Der er i det følgende blevet udarbejdet et UML Package Diagram som ofte bruges til at få er overordnet overblik over systemets logiske arkitektur og de tilhørende lag. Det følgende diagram skal ses som en vejledende model til den videre udvikling.



En del af de ovenstående pakkers klasser er endnu ikke implementeret, men vil i de kommende iterationer blive inddraget i de forskellige modeller og dernæst dokumenteret i implementeringen.

Som diagrammet illustrere så er applikationens arkitektur bygget op omkring Model-View-Controller (MVC). Grunden opgavens omfang vil der ikke blive gået i dybden med beskrivelsen af MVC som arkitektur pattern og mulige alternativer. Derimod vil der i stedet fokuseres på hvad valget af denne form for arkitektur i praksis betyder for en webapplikations struktur samt hvilke årsager der ligger bag dette valg.

I praksis betyder MVC at applikationen opdeles i tre lag; hver med ansvar for forskellige opgaver. Model-laget indeholder blandt andet grundlæggende koncepter fra den virkelige verden i form af klasser som "Bruger", "Vagt" og "Vagtplan". Det er i dette lag koncepter som "Bruger" bliver defineret til at bestå af attributter såsom "Fornavn", "Efternavn" og "Adresse".

Controller-laget indeholder de forskellige servlet klasser som står for at styre input fra klienten (browseren) i form af HTTP GET og POST-anmodninger. Dets vigtigste funktion består i at kalde og koordinere forskellige ressourcer og objekter nødvendigt for at udfører en brugerhandling.

View-laget indeholder JSP-filerne for de forskellige sider i applikationen. Dette skyldes at dette lag står for at præsentere data i form af HTML5(HTML5, CSS3 og JavaScript) modtaget fra model-laget (og deraf den tilhørende database ved hjælp af DBTool) igennem controller-laget.

Dette forløb illustreres på bedste vis i de udarbejdet Sequence Diagrammer som viser hvordan de forskellige handlinger (Use Cases) bliver håndteret af de forskellige lag.

MVC er et arkitektur pattern som kan tilføre et system bæredygtighed og effektivitet på en række planer hvis den da bruges korrekt. Desværre ses det tit at denne tilgang til systemarkitektur ikke bliver fulgt efter påskrifterne hvilket resultere i dårlig kode som bliver omkostningsfuld og tidskrævende at vedligeholde. Derfor er det vigtigt at påpege at der i en veldesignet webapplikation som gør brug af MVC ikke findes Java-kode i JSP-filerne og derudover indeholder servlet-klasserne ikke JDBC-kode. Denne opdeling er

essentiel når man snakker om MVC i en webapplikation og noget som havde særlig fokus i forbindelse med udviklingen af dette vagtplansystem.[20]

Som med alt andet findes der både fordele og ulemper ved denne form for systemarkitektur. Som nævnt tidligere så er et af de store fordele ved MVC dens adskillelse af de forskellige lag herunder *view* fra *business logic* hvilket er utrolig fordelagtigt især ved komplekse og "store" applikationer. Derved er det betydelig lettere at vedligeholde eller helt at ændrer en del af et system uden at berøre resten af systemet. Dette er især nyttigt når der udvikles webapplikationer som denne hvor brugergrænsefladen kan være under konstant ændring både før, under og efter udviklingen. Endvidere gør denne adskillelse det muligt for personer med forskellige kompetencer at arbejde på et projekt sideløbende - for eksempel kan en designer arbejde med layoutet og brugergrænsefladen mens en programmør kan udvikle den bagvedlæggende kode uden at de påvirker hinandens arbejde.

Nogle af ulemperne ved denne type af arkitektur er at det i visse tilfælde kan være relativ svært at bestemme hvor et specifik stykke af applikationen skal befinde sig. For denne applikations vedkommende skulle der blandt andet foretages et valg om hvorvidt de såkaldte Data Access Objects (DAO), som vil blive udviklet i den kommende iteration, skulle befinde sig i controller-laget eller model-laget. Det er netop ved denne slags valg en masse udviklere fejler og gør derved brug af MVC på en forkert måde som i sidste ende skader mere end det gavner.

Ovenstående faktorer taget i betragtning blev det vurderet at fordelene ved denne form for arkitektur vægtede højere end de få ulemper der var. Det var især vigtigt for samarbejdspartneren at applikationens kode var struktureret og derved lettere at vedligeholde, teste og genbruge.

7.5 Implementation Model

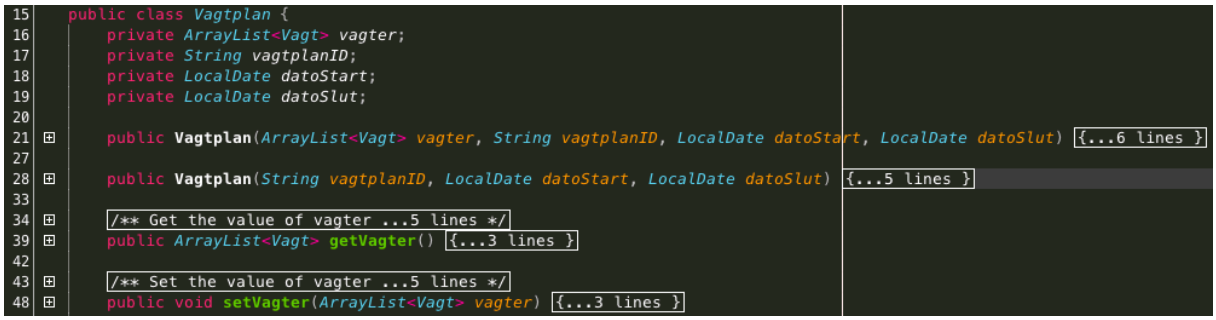
På baggrund af de undersøgelser og analyser der er foretaget samt de udarbejdet design og arkitektur modeller herunder System Sequence Diagram og Design Class Diagram, er

[20] Fowler, Martin: Patterns of Enterprise Application Architecture . Side 330-332. 1. udg. Addison-Wesley Professional, 2002. (Bog)

der nu indsamlet tilstrækkelig med information og viden omkring applikationens grundlæggende struktur til at foretage en implementering af forskellige dele af systemet.

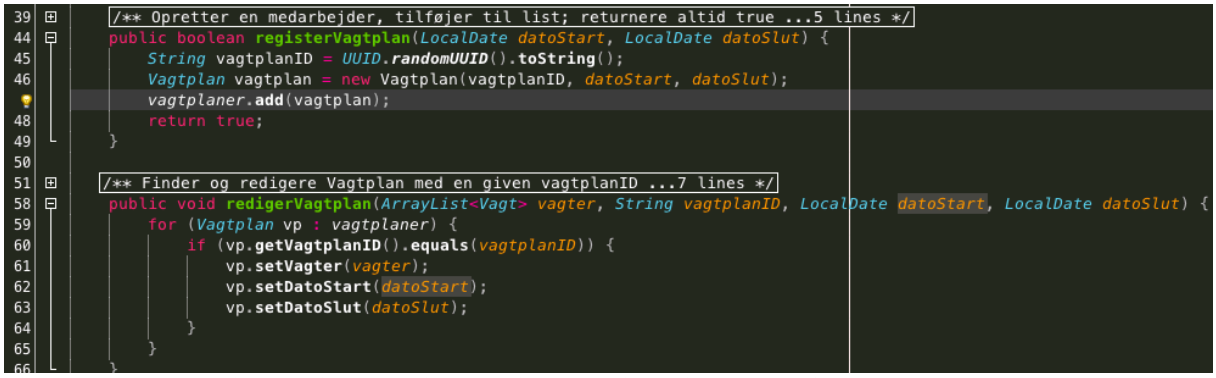
For simplicitetens skyld og for at holde opgaven overskuelig vil der kun fremlægges eksempler på de grundlæggende dele af systemets implementering således at disse kan være medvirkende til at give en bedre forståelse for overgangen mellem design og implementering samt hvordan de forskellige modeller har bidraget til denne udvikling.

Nedenstående illustration viser et eksempel på hvordan attributter og metoder overføres fra Klassediagrammet og implementeres i henhold til Javas definition på, i dette tilfælde, en *Vagtplan*.



Figur 7.2: Delvis illustration af klassen Vagtplan i Java.

Endvidere bliver de forskellige kald illustreret i Sequence Diagrammerne implementeret i form af en række metoder herunder tilføjVagtplan i UC1 og redigerVagtplan i UC2 som illustreres i den nedenstående figur.



Figur 7.1: Viser metoderne registerVagtplan og redigerVagtplan i klassen VagtplanRegister.

7.6 Delkonklusion

Der kan hermed konkluderes at de forskellige aktiviteter for *elaboration-fasen* er under udvikling i henhold til den iterative udvikling. Først og fremmest er kravene fra *inception-fasen* blevet klarificeret og forbedret. Dernæst er *core-arkitekturen* blevet analyseret og der er udarbejdet yderligere artefakter i form af de forskellige modeller såsom domain- og designmodellen.

Endvidere er processen for udvikling af de centrale dele af systemet påbegyndt hvilket vil fortsætte i den næste *iteration* af fasen. Som afslutning er der arbejdet hen imod en implementering af systemet, hvori de praktiske test er foretaget.

Dette stadie af analyse- og design fasen er i store træk begyndt at vise hvordan der kan udvikles et system til håndtering og planlægning af ansattes vagter i en virksomhed.

8 Elaboration – 2. Iteration (E2)

I det følgende vil den anden fase (*iteration*) af *elaboration* blive udarbejdet i henhold til den iterative udvikling, Unified Process og den fremlagte problemstilling.

Formålet med denne *iteration* af elaboration er at beskrive og illustrere forlængelser til use cases, bearbejdelse af sekvens diagrammerne, analyse af UI-designet i henhold til systemets udvikling, samt en brobygning til databasens forbindelse. Derudover vil de forskellige modeller udarbejdet i første *iteration* tilpasses de forskellige ændringer der er opstået undervejs i udviklingen.

Med andre ord vil der fremhæves det essentielle objekt design, brug af *patterns* til at udarbejde et bæredygtigt design samt brug af UML til at visualisere de forskellige modeller.

8.1 Revideret Use Case Model

8.1.1 Use Case text

UC3: Opret Vagt

Scope: System
Level: Brugerniveau
Primary Actor: Leder

Stakeholders og Interests:

- Cheferne: Ønsker et overblik over virksomhedens daglige ledelse og dennes drift, økonomi og arbejdsmæssige ressourcer.
- Skat: Er interesseret i virksomhedens beskæftigelsesforhold.
- Leder: Ønsker at der bliver oprettet en vagt i henhold til de forventninger og ressourcer virksomheden ønsker på en given dag .
- Ansatte: Ønsker at få et overblik over de forskellige ledige vagte der kan bemandes.

Precondition: Lederen er identificeret og autentificeret af systemet.
Postcondition (Succes Guarantee): En vagt blev oprettet og gemt i systemet.

Main Success Scenario:

1. Lederen påbegynder oprettelse af en ny vagt.
2. Systemet beder om indtastning af informationer omkring vagten.
3. Lederen indtaster de nødvendige informationer såsom tid, antal medarbejdere og type af arbejde.
4. Lederen bekræfter oprettelsen af den nye vagt.
5. Vagten oprettes og gemmes i systemet.

Extensions

2a. Invalid eller manglende information:

1. Systemet advarer om fejl i indtastningen eller mangel på nødvendig information.

5a. Vagten bliver ikke gemt i systemet.

1. Systemet fejler i at oprette forbindelse til databasen.

UC4: Byt Vagt

Scope: System
Level: Brugerniveau
Primary Actor: Medarbejder

Stakeholders og Interests:

- Medarbejder: Ønsker at bytte sin vagt med en kollegas vagt.

Precondition: Medarbejderen er identificeret og autentificeret af systemet.
Postcondition (Succes Guarantee): En medarbejder har byttet sin vagt med en kollega

Main Success Scenario:

1. Medarbejderen påbegynder bytning af en vagt.
2. Den ønsker vagt med den specifikke kollega bliver fundet i systemet og der bliver spurgt om bekræftelse på bytning.
3. Medarbejderen bekræfter bytning af den valgte vagt.
4. Vagten byttes nu med den ønsket kollegas vagt.

Extensions

4a. Vagten kan ikke byttes.

1. Vagten er af anden type.

UC5: Opret Medarbejder

Scope: System
Level: Brugerniveau
Primary Actor: Leder

Stakeholders og Interests:

- Cheferne: Ønsker et overblik over virksomhedens daglige ledelse og dennes drift, økonomi og arbejdsmæssige ressourcer.
- Skat: Er interesseret i virksomhedens beskæftigelsesforhold.
- Leder: Ønsker at skabe et overblik over virksomhedens ansatte og deres kontaktoplysninger i forbindelse med den daglige ledelse.
- Ansatte: Ønsker at få adgang til virksomhedens vagtplanssystem så de kan planlægge den kommende måneds arbejde.

Precondition: Lederen er identificeret og autentificeret af systemet.
Postcondition (Succes Guarantee): En medarbejder blev oprettet og gemt i systemet.

Main Success Scenario:

1. Lederen påbegynder oprettelse af en ny medarbejder.
2. Systemet beder om indtastning af informationer omkring medarbejderen.
3. Lederen indtaster de nødvendige informationer såsom navn, adresse, fødselsdag samt telefon- og mobilnummer.
4. Lederen bekræfter oprettelsen af den nye medarbejder.
5. Medarbejderen oprettes og gemmes i systemet.

Extensions

2a. Invalid eller manglende information:

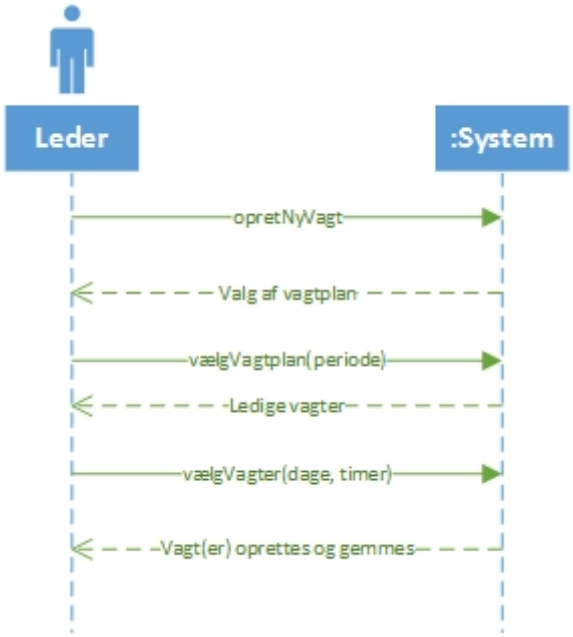
1. Systemet advarer om fejl i indtastningen eller mangel på nødvendig information.

5a. Medarbejderen bliver ikke gemt i systemet.

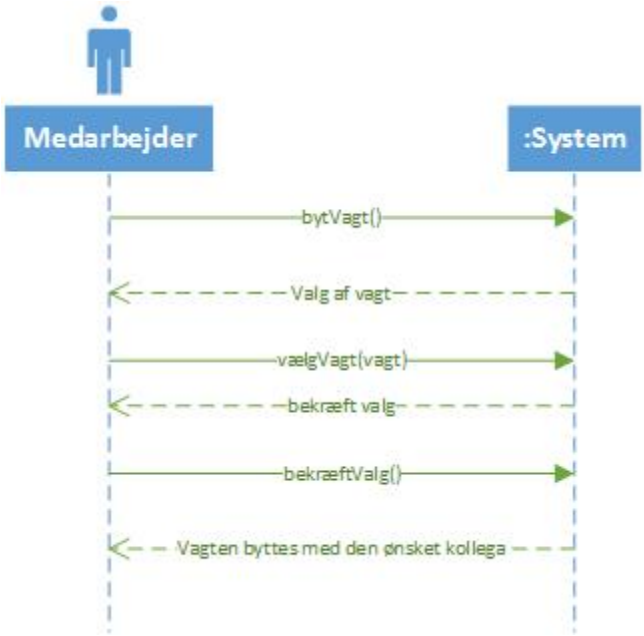
1. Systemet fejler i at oprette forbindelse til databasen.
2. Medarbejderen findes allerede i systemet.

8.1.2 System Sequence Diagram

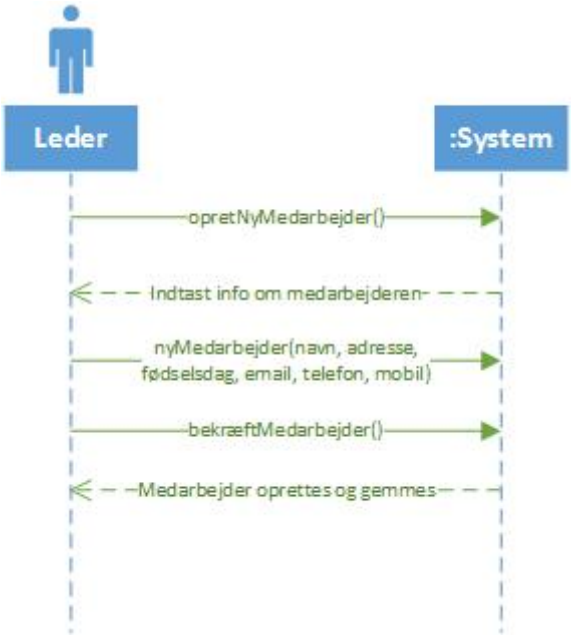
UC3: Opret Vagt



UC4: Byt Vagt



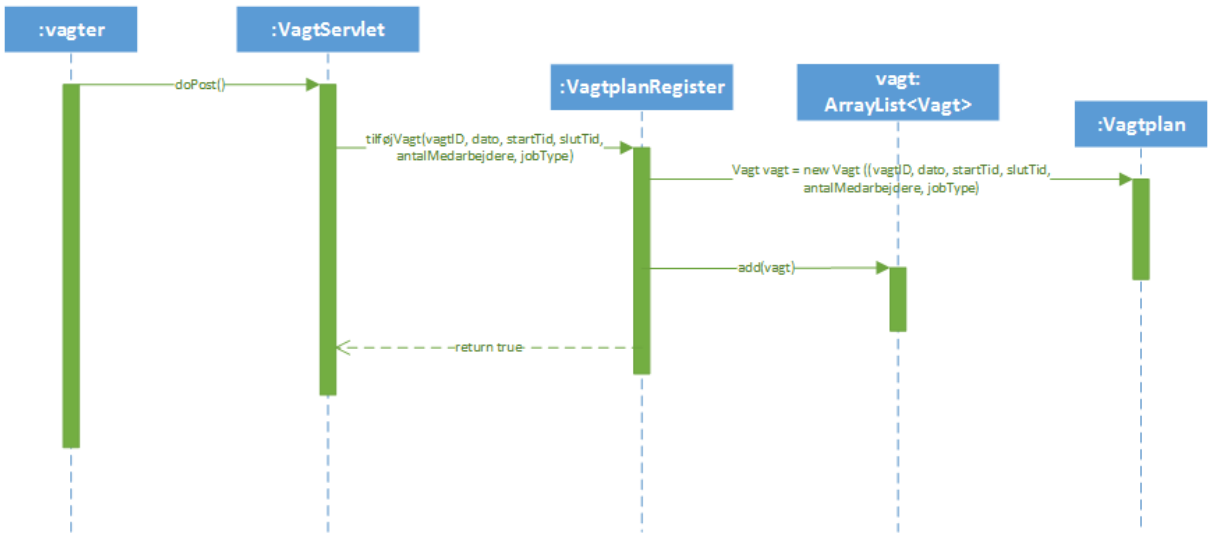
UC5: Opret Medarbejder



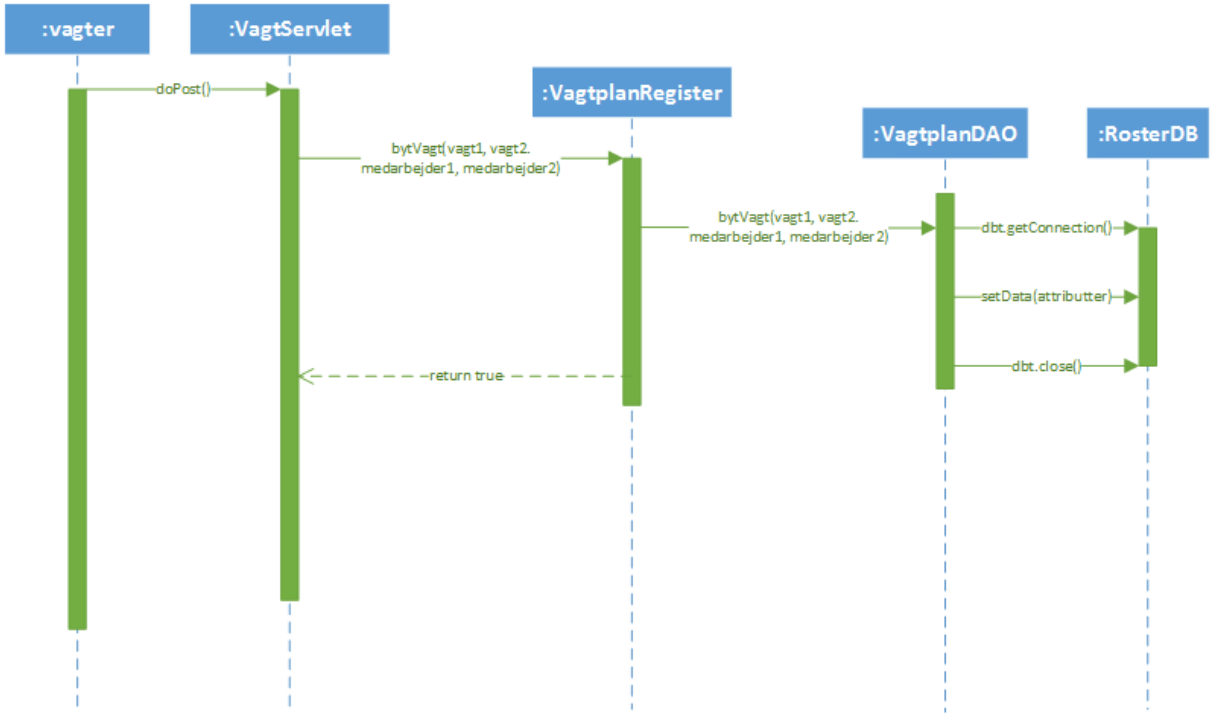
8.2 Revideret Design Model

8.2.1 Sequence Diagram

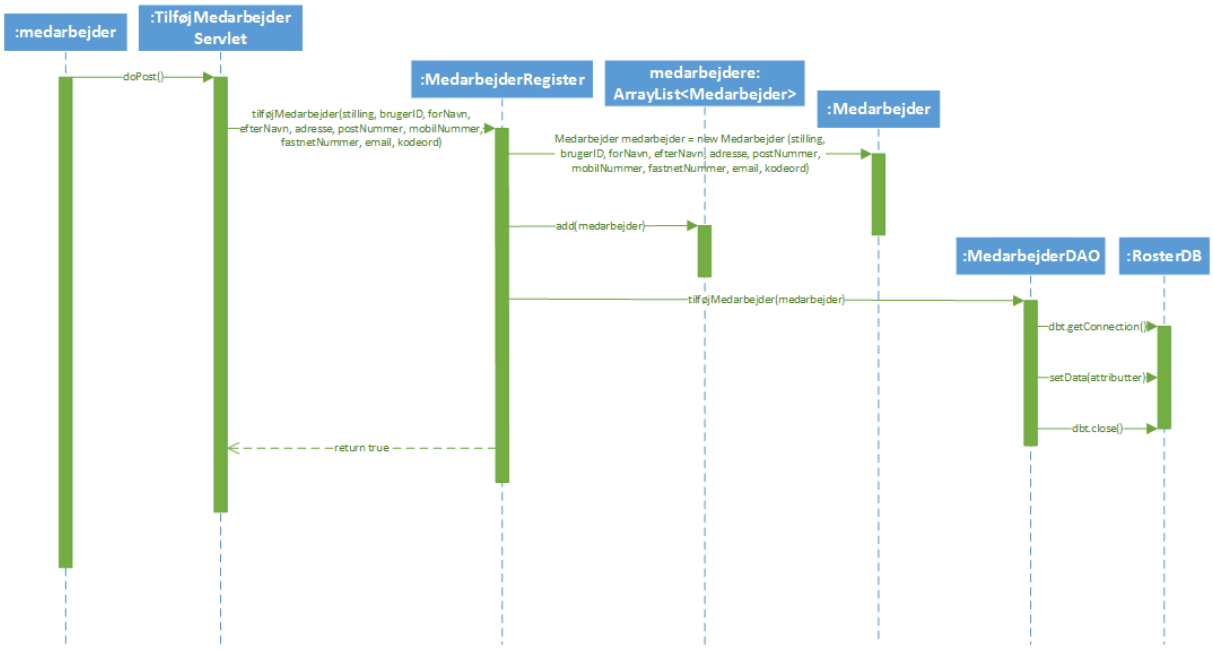
UC3: Opret Vagt



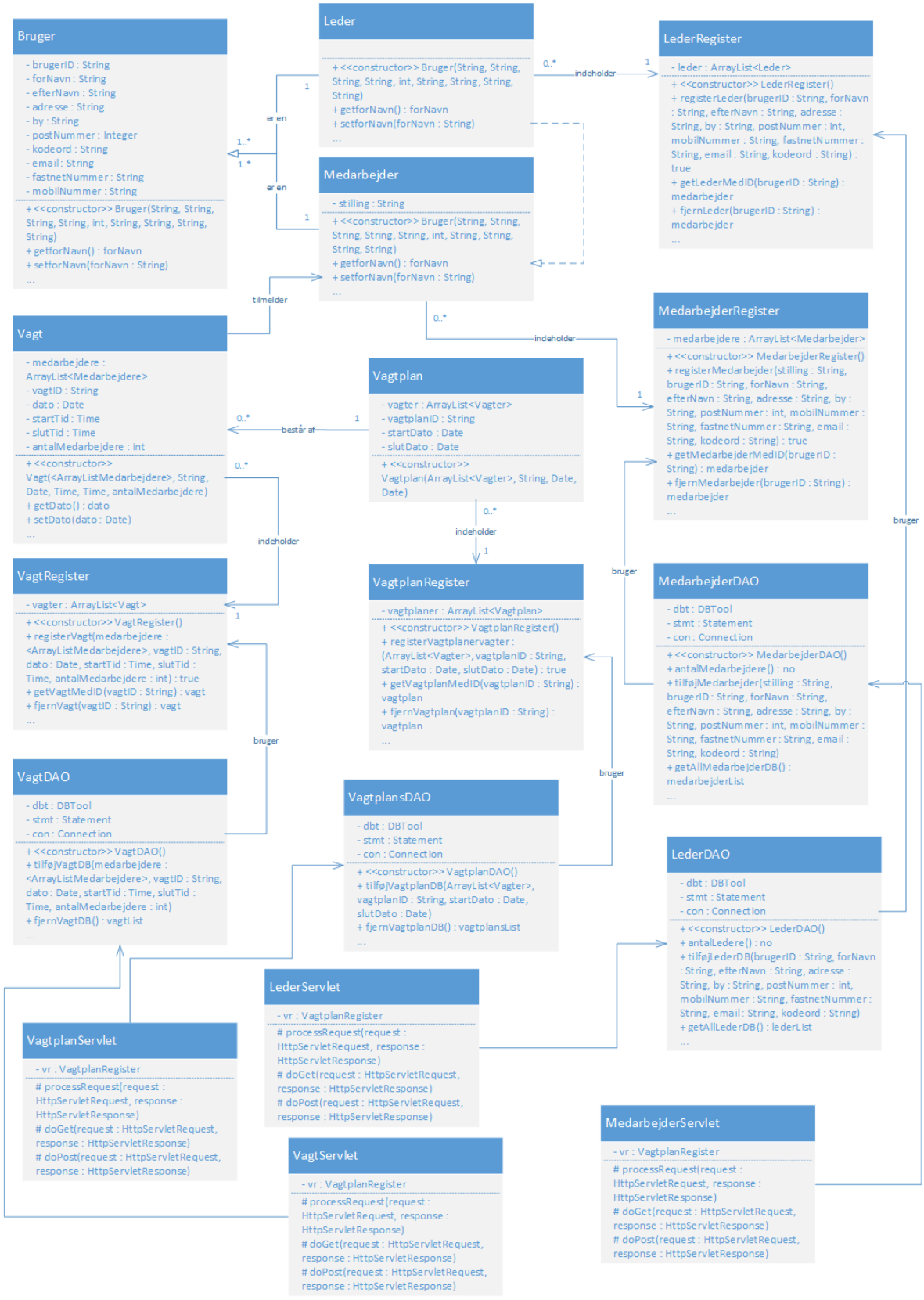
UC4: Byt Vagt



UC5: Opret Medarbejder


:medarbejder
:TilføjMedarbejder Servlet
:MedarbejderRegister
medarbejdere: ArrayList<Medarbejder>
:Medarbejder
:MedarbejderDAO
:RosterDB
doPost()
tilføjMedarbejder(stilling, brugerID, forNavn, efterNavn, adresse, postNummer, mobilNummer, fastnetNummer, email, kodeord)
Medarbejder medarbejder = new Medarbejder (stilling, brugerID, forNavn, efterNavn, adresse, postNummer, mobilNummer, fastnetNummer, email, kodeord)
add(medarbejder)
tilføjMedarbejder(medarbejder)
dbt.getConnection()
setData(attributter)
dbt.close()
return true

8.2.2 Design Class Diagram



I henhold til den iterative udvikling er der tilført klassediagrammet en række nye klasser i denne fase af udviklingen, og fokus har primært været på controller-laget samt at forbedre og forfine arkitekturen ved hjælp af forskellige design patterns.

I forbindelse med udarbejdelsen af ovenstående model, har der også være stor fokus på at skabe en struktur der tager hensyn til GRASP patterns og særlige principper til at udvikle et bæredygtigt system. Herunder GRASP's Low Coupling (Lav Kopling) pattern som sætter fokus på hvordan man reducere påvirkningen af ændringer i et system, og samtidig mindsker klassernes afhængighed af hinanden. Denne problemstilling er løst ved at man har ladet klasserne have så få referencer til andre klasser som muligt. Samtidig er de enkelte klasser tildelt et let forståeligt og ensartet ansvarsområde hvilket har resulteret i en høj Cohesion (kohæsion).[21]

Dette bringer os til et andet punkt med hensyn til GRASP's såkaldte *patterns*. For en af måderne man opnår lav kobling i forbindelse med designingen af ens system er ved hjælp af *Expert-princippet* (Ekspert-princippet) som går ud på at man tildeler den klasse, med den nødvendige information til at udføre handlingen, ansvaret for handlingen. I denne sammenhæng har klassen *Bruger* for eksempel personoplysninger omkring brugeren og derfor har den også ansvaret for handlinger der vedrøre informationer omkring en specifik bruger.[22]

Dog er ovenstående princip i nogle tilfælde ikke den mest hensigtsmæssige tilgang. Selvom objektorienterede design er karakteriseret af koncepter fra den virkelige verden som bliver implementeret til domain layer software klasser som f.eks. *Bruger-*, *Vagt-* og *Vagtplan-klasserne* så er der situationer hvor denne tilgang blandt andet fører til dårlig kohæsion og kobling. Dette var en problemstilling som opstod i forbindelse med tilkoblingen af en relationel database for persistent lagring af data. Hvis ovenstående Expert-princip skulle følges kan der argumenteres for at en model klasse som for eksempel *Vagt* som indeholder de nødvendige informationer omkring en instans af *Vagt* bør stå for overførslen til den relationelle database. Dette medfører dog en række komplikationer:

[21] Craig Larman: Applying UML and Patterns (side 294-299) 3. udg. Pearson Education, 2005 (Bog)
[22] Craig Larman: Applying UML and Patterns (side 291-294) 3. udg. Pearson Education, 2005 (Bog)

- Opgaven kræver et relativt stort antal af database-orienteret operationer hvor ingen har relationer til for eksempel konceptet Vagt, hvilket betyder at klassen bliver inkonsistent og usammenhængende.

- Vagt klassen skal som sagt kobles til en relationel database interface (JDBC i dette tilfælde) hvilket resultere i at koblingen forhøjes.

- At lagre objekter i en relationel database er en meget generel opgave som mange klasser har behov for at tilgå og derfor vil dette medfører en masse redundans på længere sigt.

I sådanne tilfælde er der brug for ”kunstige” eller såkaldte ”convenience classes” som skal medvirke til at bibeholde den høje kohæsion, lave kobling og genbrug. Derved løses ovenstående komplikationer ved hjælp af *Pure Fabrication* princippet hvorved de forskellige Data Access Object (DAO) klasser bliver udviklet. Disse klasser står udelukkende for at hente og lagre objekter i en relationel database.

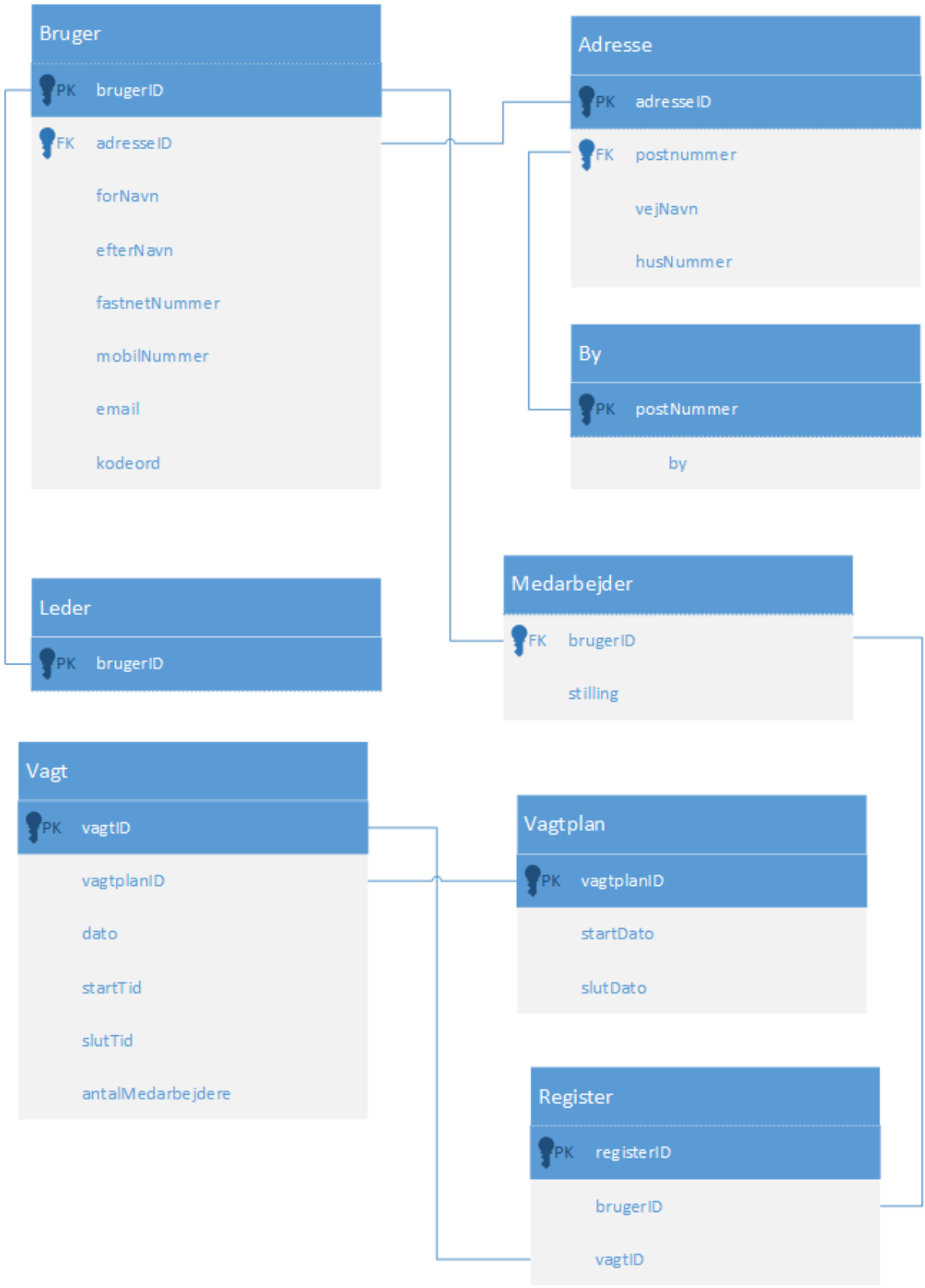
Derudover er der gjort brug af Creator pattern som indebærer at man tildeler en særlig klasse ansvaret for at skabe en instans af en anden klasse. Denne pattern var med i overvejelserne da de forskellige Register klasser blev udviklet, og derfor står de for at skabe deres respektive model objekt klasser.

8.3 Data Model

Formålet med denne aktivitet er at implementere selve designet i en Relationel Database (RDB). Før igangsættelsen af denne del af projektet kan der argumenteres for forskellige valg af tilgange, hvor en af disse er at skabe en direkte forbindelse mellem design klassediagrammet til relations tabellerne grunden den gennemtænkte og detaljeret design proces.

Der vil i det følgende blive udarbejdet og beskrevet aktiviteter i form af valg af relationelle database (RDB), gennemgang af design modellen og hvis nødvendigt foretage nye ændringer. Derudover vil de forskellige klasser kategoriseres og omdannes til relationer, heriblandt associationer og strukturer. Der vil desuden defineres og beskrives forskellige relationer, attributter, nøgler og domæner. Til slut vil der foretages en normalisering af tabellerne.

8.3.1 Database Diagram



8.4 Implementation Model

I den følgende del af projektet blev der tilføjet en række nye klasser til de forskellige diagrammer. Disse blev implementeret således:

```
93      $("button[name='bekræft']").click(function () {
94          var startDato = $('#inputDateFrom').val();
95          var slutDato = $('#inputDateTo').val();
96          var vp = new vagtplan2(startDato, slutDato);
97
98          $.ajax({
99              url: "Tilf_jVagtplanServlet",
100             type: 'POST',
101             dataType: 'json',
102             data: JSON.stringify(vp),
103             contentType: "application/json",
104             success: function (data) {
105                 var plan = jQuery.parseJSON(data);
106                 idVagtplan = plan.vagtplanID;
107                 console.log(idVagtplan);
108             }
109         });
```

Ovenstående figur viser det JSON (*JavaScript Object Notation*) objekt der bliver skabt ud fra de informationer som bliver tastet ind i de forskellige *forms* ved hjælp af JavaScript, og som så bliver sendt videre til den til den tilhørende Servlet via AJAX (*Asynchronous JavaScript And XML*).

```
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
74          throws ServletException, IOException {
75      request.setCharacterEncoding("UTF-8");
76
77      BufferedReader br = new BufferedReader(new InputStreamReader(request.getInputStream()));
78      String json = "";
79      if (br != null) {
80          json = br.readLine();
81      }
82
83      JSONParser parser = new JSONParser();
84      try {
85          Object obj = parser.parse(json);
86          JSONObject jsonObject = (JSONObject) obj;
87
88          String startDato = (String) jsonObject.get("startDato");
89          LocalDate date2 = LocalDate.parse(startDato);
90
91          String slutDato = (String) jsonObject.get("slutDato");
92          LocalDate date1 = LocalDate.parse(slutDato);
93
94          ArrayList<Vagt> v = new ArrayList<Vagt>();
95
96          vr = new VagtplanRegister();
97
98          vr.registerVagtplan(date1, date2);
99
100         System.out.println(vr.toString());
101         System.out.println("Success");
102     } catch (org.json.simple.parser.ParseException ex) {
103         Logger.getLogger(VagtplanServlet2.class.getName()).log(Level.SEVERE, null, ex);
104     }
105 }
```

Ovenstående figur viser hvordan Servlet klassen håndtere et POST-kald i form af dette såkaldte JSON-objekt som bliver konverteret i henhold til de forskellige attributter og bearbejdet af controller klassen *RegisterVagtplan.*

Derudover illustrere nedenstående figur en såkaldt superklasse, hvilket betyder at underklasserne *Medarbejder* og *Leder* arver på en række punkter hvor de ligner hinanden. Dette gør at man kan tildele aktører forskellige roller med tilhørende funktioner.

```
public class Bruger {
    private String brugerID;
    private String forNavn;
    private String efterNavn;
    private String adresse;
    private String by;
    private int postNummer;
    private String mobilNummer;
    private String fastnetNummer;
    private String email;
    private String kodeord;

    public Bruger(String brugerID, String forNavn, String efterNavn,
            String adresse, String by, int postNummer, String mobilNummer,
            String fastnetNummer, String email, String kodeord) {...12 lines }

    /** Get the value of kodeord ...5 lines */
    public String getKodeord() {...3 lines }

    /** Set the value of kodeord ...5 lines */
    public void setKodeord(String kodeord) {...3 lines }
```

8.5 Delkonklusion

Der kan hermed konkluderes at de forskellige aktiviteter for den 2. *iteration* af *Elaboration* er udarbejdet og fuldført. De forskellige problemstillinger der opstod undervejs i processen med hensyn til design og arkitektur er løst, og der er derved foretaget en række ændringer med henblik mod påbegyndelsen af den næste fase.

Dog er der desværre opstået den situation at projektet ikke holder den estimeret plan og derved er der begyndt at opstå forsinkelser som kræver at der foretages en række valg med hensyn til prioriteringen af applikationens udvikling. Disse valg vil blive belyst i den kommende iteration.

Der er endvidere foretaget bearbejdelse af sekvens diagrammerne samt analyse af GUI-designet i henhold til systemets udvikling. Samtidig er de forskellige modeller tilpasset de forskellige ændringer.

Herudover er der sat fokus på det essentielle objekt design i forhold til brug af *patterns* til udarbejdelse af et bæredygtigt design samt brug af UML til at visualisere de forskellige modeller. Ovenstående er sket med henblik på at skabe en forbindelse til den næste iteration af elaboration.

9 Elaboration – 3. Iteration (E3)

I det følgende vil den tredje fase (*iteration*) af *elaboration* blive udarbejdet hvor der vil blive taget fat om en række problemstillinger der er opstået i forbindelse med projektet.

Først og fremmest er den udarbejdet projektplan desværre ikke blevet holdt på grund af en række årsager, og projektet vil derfor tage længere tid at fuldføre end først antaget. Dette medfører at der må prioriteres i forhold til den indeværende iteration og der vil derfor være fokus på programmeringen af systemets grundlæggende funktion og struktur (back-end). Derfor vil udviklingen af en række funktioner og *view*-delen (front-end) først færdiggøres efter afslutningen af rapporten.

9.1 Use case tekst

UC6: Slet Medarbejder

Scope: System
Level: Brugerniveau
Primary Actor: Leder

Stakeholders og Interests:

- Cheferne: Ønsker et overblik over virksomhedens daglige ledelse og dennes drift, økonomi og arbejdsmæssige ressourcer.
- Skat: Er interesseret i virksomhedens beskæftigelsesforhold.
- Leder: Ønsker at skabe et overblik over virksomhedens ansatte og deres kontaktoplysninger i forbindelse med den daglige ledelse.

Precondition: Lederen er identificeret og autentificeret af systemet.
Postcondition (Succes Guarantee): En medarbejder er blevet slettet fra systemet.

Main Success Scenario:

1. Lederen påbegynder sletningen af en medarbejder.

2. Den ønsker medarbejder bliver fundet i systemet og der bliver spurgt om bekræftelse på sletning.
3. Lederen bekræfter sletning af den valgte medarbejder.
4. Medarbejderen slettes fra systemet.

Extensions

4a. Medarbejderen bliver ikke fjernet fra systemet.

1. Systemet fejler i at oprette forbindelse til databasen.

UC7: Søg Medarbejder

Scope: System
Level: Brugerniveau
Primary Actor: Leder

Stakeholders og Interests:

- Leder: Ønsker at kunne finde en given medarbejder baseret ud fra vedkommendes navn.
-

Precondition: Lederen er identificeret og autentificeret af systemet.
Postcondition (Succes Guarantee): Den ønsket medarbejder bliver fundet i systemet.

Main Success Scenario:

1. Lederen påbegynder søgning af medarbejder.
2. Afhængig af søgekriterier bliver for- og/eller efternavn indtastet.
3. Systemet søger efter den givne medarbejder.
4. Den ønsket medarbejder findes og returneres.

Extensions

4a. Medarbejderen bliver ikke fjernet fra systemet.

1. Systemet fejler i at oprette forbindelse til databasen.

UC7: Søg Medarbejder

Scope: System
Level: Brugerniveau
Primary Actor: Leder

Stakeholders og Interests:

- Leder: Ønsker at kunne finde en given medarbejder baseret ud fra vedkommendes navn.

Precondition: Lederen er identificeret og autentificeret af systemet.
Postcondition (Succes Guarantee): Den ønsket medarbejder bliver fundet i systemet.

Main Success Scenario:

1. Lederen påbegynder søgning af medarbejder.
2. Afhængig af søgekriterier bliver for- og/eller efternavn indtastet.
3. Systemet søger efter den givne medarbejder.
4. Den ønsket medarbejder findes og returneres.

Extensions

4a. Medarbejderen bliver ikke fjernet fra systemet.

2. Systemet fejler i at oprette forbindelse til databasen.

UC8: Gem Vagtplan Lokalt

Scope: System
Level: Brugerniveau
Primary Actor: Leder

Stakeholders og Interests:

- Cheferne: Ønsker et overblik over virksomhedens daglige ledelse og dennes drift, økonomi og arbejdsmæssige ressourcer.
- Skat: Er interesseret i dokumenteringen af virksomheden og de ansattes arbejdsforhold.
- Leder: Ønsker at det er muligt at kunne gemme de forskellige vagtplaner lokalt så de eventuelt kan bruges i andre virksomhedsmæssig sammenhænge.

Precondition: Lederen er identificeret og autentificeret af systemet.

Postcondition (Succes Guarantee): Den ønsket vagtplan bliver gemt på det lokale system.

Main Success Scenario:

1. Lederen anmoder om en tidligere oprettet vagtplan.
2. Lederen beder systemet om at gemme den valgte vagtplan lokalt.
3. Systemet beder lederen om den ønskede sti på det lokale system vagtplanen skal gemmes.
4. Systemet gemmer vagtplanen lokalt.

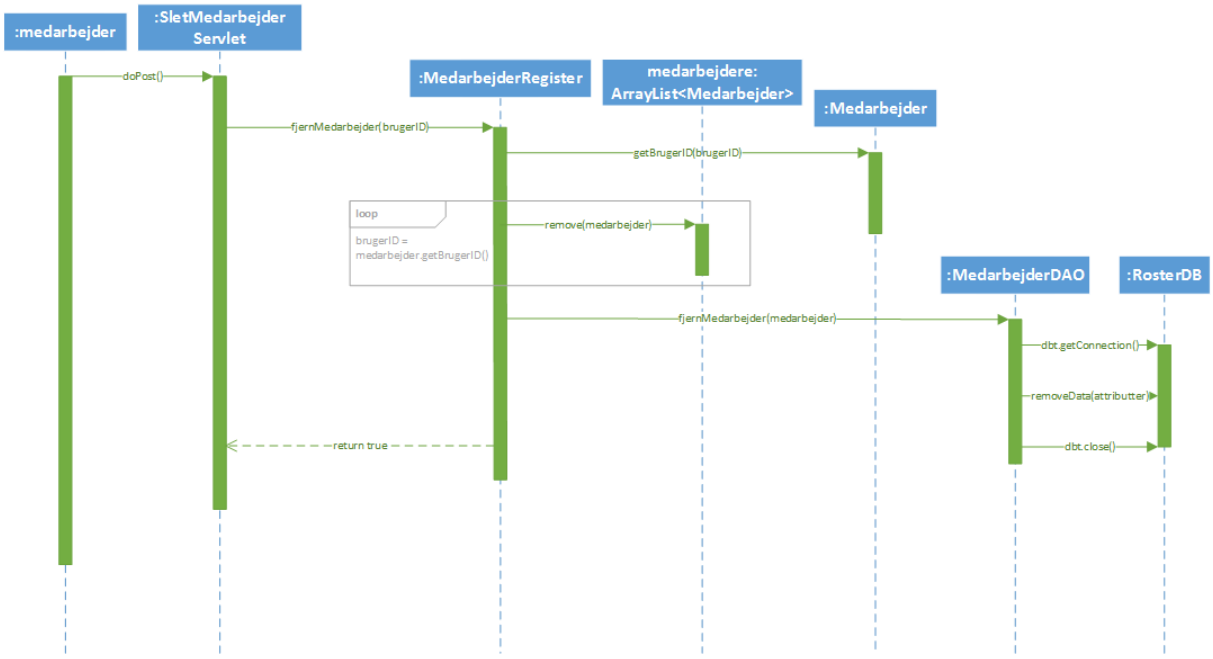
Extensions

3a. Det er ikke muligt at gemme på den ønskede sti.

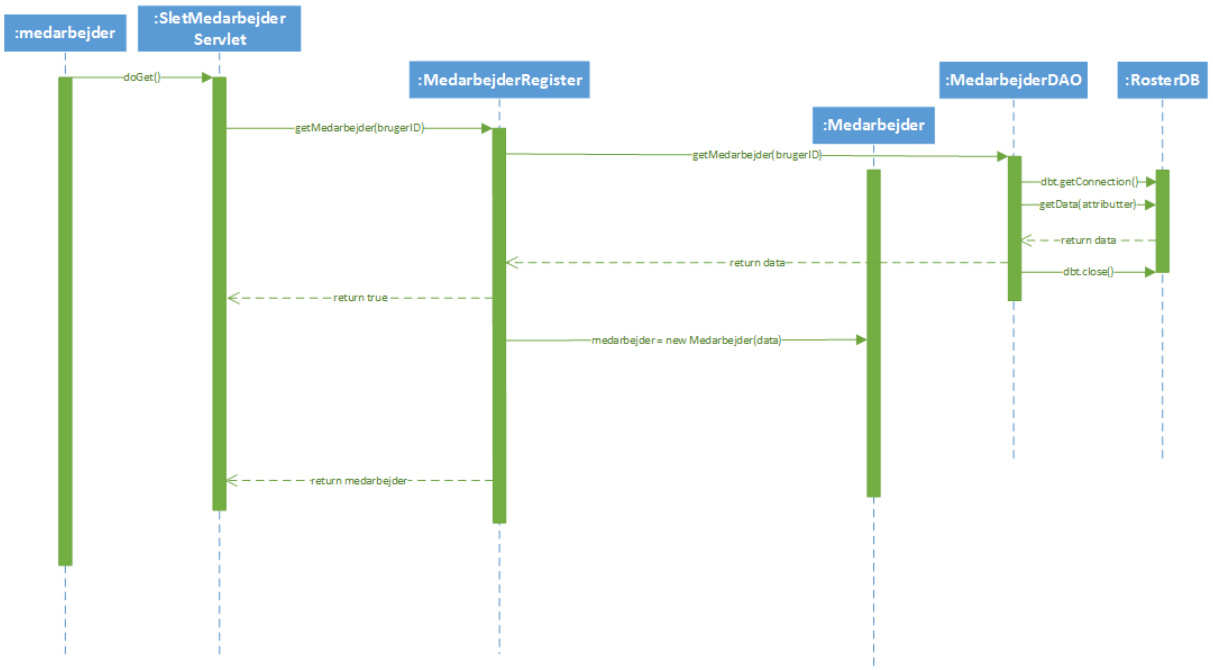
1. Stien findes ikke på det lokale system.
2. Systemet har ikke adgang til den valgte sti.

9.2 Sequence Diagram

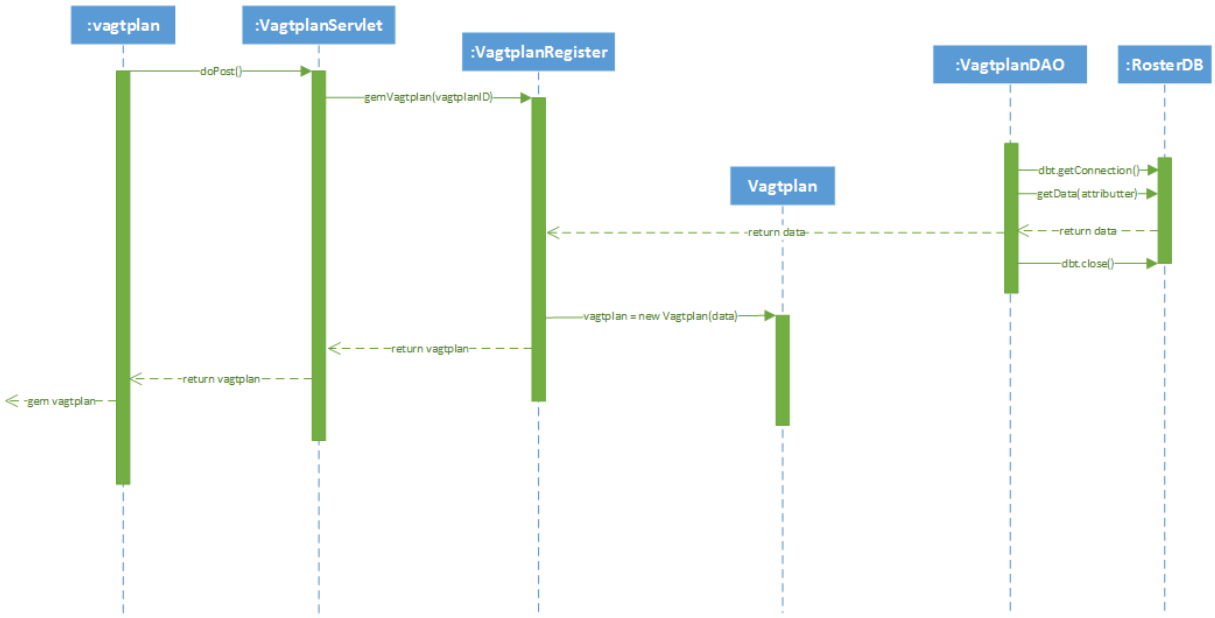
UC6: Slet Medarbejder



UC7: Søg Medarbejder



UC8: Gem Vagtplan


:vagtplan
:VagtplanServlet
:VagtplanRegister
:VagtplanDAO
:RosterDB
Vagtplan
doPost()
gemVagtplan(vagtplanID)
dbt.getConnection()
getData(attributter)
return data
return data
dbt.close()
vagtplan = new Vagtplan(data)
return vagtplan
return vagtplan
gem vagtplan

10 Samlet Konklusion

Der vil i det følgende afsnit konkluderes for den udarbejdet problemstilling og de tilhørende undersøgelsesspørgsmål.

Med afsæt i den analyse og sammenligning der er foretaget, kan der udledes man først og fremmest skal finde frem til kundens behov og krav, for derved at kunne fremstille et produkt der kan bruges i henhold til kundens daglige arbejde. Denne centrale problemstilling løser *Unified Process* ved hjælp af sine analyse-metoder og principper som de er taget i brug i forbindelse med udarbejdelsen af kravspecifikationerne og ikke mindst i form af den indledende *inception-fase*. Her er brugerens behov og ønsker, med hensyn til hvordan disse kan blive opfyldt af det udarbejdet system, blevet udledt. Herudover er defineringen af en række krav som systemet skal opfylde desuden blevet gennemarbejdet og fundet.

Herefter støder man på en anden problemstilling, i form af hvordan man formår at fremstille et produkt i lyset af disse særlige krav og behov. Her formår *Unified Process* at skabe en bro til selve implementeringen og de tilhørende teknologier som Java og HTML5, hvilket er sket igennem en række særlige principper og metoder i form af ikke mindst *GRASP-patterns* der er blevet brugt i forbindelse med programmeringen til at skabe et bæredygtigt produkt der kan bearbejdes og udvikles i fremtiden. Ved hjælp af blandt andet disse metoder og principper har det været muligt at fremstille og udvikle et system til håndtering og planlægning af ansattes vagter for den samarbejdende virksomhed.

Endvidere har dette forløb vist hvilke fordele og ulemper der ved de forskellige teknologier herunder JSP/Java. En af fordelen ved JSP har helt klart været dens understøttelse af *frameworks* og dens store kendskab og fællesskab. Dog er Java utrolig svær at hoste da meget få hosting firmaer understøtter sproget i dag da det kræver en masse ressourcer. Derfor er sproget særlig velegnet til komplekse eller forholdsvis store applikationer med en masse samtidig trafik.

11 Litteraturliste

C. Hay, David: Data Model Patterns . 1. udg. Dorset House, 2011. (Bog)

Fowler, Martin: Analysis Patterns: Reusable Object Models. 1. udg. Addison-Wesley Professional, 1996. (Bog)

Loudon, Kyle: Developing Large Web Applications. 1. udg. O'Reilly Media, Inc, 2010. (Bog)

Tidwell, Jenifer: Designing Interfaces. 2. udg. O'Reilly Media, Inc, 2011. (Bog)

Fowler, Martin: Patterns of Enterprise Application Architecture. 1. udg. Addison-Wesley Professional, 2003. (Bog)

S. Williams, Nicholas: Professional Java for Web Applications. 1. udg. John Wiley & Sons, Inc., 2014. (Bog)

ZAKI WARFEL, TODD: PROTOTYPING. 1. udg. Rosenfeld Media, 2009. (Bog)

Larman, Craig: Applying UML and Patterns. 3. udg. Pearson Education, 2005. (Bog)

Andersen, Niels Erik: Professionel systemudvikling. 1. udg. Ingeniøren Bøger, 2003. (Bog)

Begg, Carolyn : Database Solutions, A Step-by-Step Approach to Building Databases. 2. udg. Pearson Education, 2003. (Bog)

12 Bilag

12.1 Liste over risici forbundet med projektet

Risk	Effect	Probability	Impact	Exposure	Trigger Event	Mitigation Strategy	Actions
Urealistisk estimering af tidsfrister.	Projektet risikere at overskride de aftale tidsfrister og dermed blive forsinket.	4	4	16	Når den udarbejdet projektplan ikke stemmer overens med virkeligheden.	Der vil blive gjort brug af erfaringer fra tidligere projekter, herunder de projekter som ligner dette i mest i forskellige aspekter.	Afhængigt af graden af forsinkelse kan der foretages en reduktion forskellige steder i applikationen så den ønsket plan kan overholdes.
Produktet formår ikke at skabe værdi hos slutbrugeren.	Produktet kan ikke opfylde virksomhedens behov på tilfredsstillende vis.	3	5	15	Slutbrugeren inddrages ikke i en tilstrækkelig grad under design- og udviklingsprocessen.	Inddrage og kommunikere med slutbrugeren under hele projektforløbet.	Finde frem til igennem et møde med slutbrugeren hvilken faktorer der er skyld i, at produktet ikke skaber værdi hos slutbrugeren.
Manglende kompetencer ift. komplekse situationer	Kan påvirke projektet negativt på det tidsmæssige og tekniske niveau.	3	3	9	Hvis der opstår situationer hvori kompleksiteten i selve implementeringen af en specifik løsning overgår undertegnedes tekniske eller faglige evner.	Være forebyggende ift. Implementeringer af forskellige løsninger og mulig kompleksiteter. Derudover bør der søges råd blandt folk med erfaringer indenfor denne slags opgaver.	Finde løsninger hvori man undgår at støde ind i situationer der kræver specielle kompetencer indenfor et snævert fagområde.
Utilstrækkelig mængde eller lav kvalitet af produktets dokumentation.	Virksomheden kan ikke bearbejde produktet yderligere i fremtiden. Derudover bliver det også svære for virksomheden at løse eventuelle problemstillinger i den kommende fremtid.	2	3	6	Udviklingsmetodikken bliver ikke fulgt, og der bliver ikke dokumenteret løbende.	Sørge for at overholde retningslinjerne for den valgte udviklingsmetodik og dokumentere løbende.	Sørge for at udarbejde en dokumentation i samarbejde med kunden.
Kontaktpersoner i virksomheden kan ikke træffes.	Der opstår tvivl om forskellige dele af produktet og virksomheden risikere et produkt som ikke lever op til de ønskede krav.	1	5	5	Der går lang tid fra en anmodning bliver afsendt til der kommer svar fra virksomheden.	Lave nogen klare aftaler mht. møder og den generelle kontakt.	Der vil i stedet fokuseres på den indeværende information og arbejdes ud fra denne.