

Hovedopgave - Datamatiker



Dato: 14. januar 2015

Datamatiker 5. Semester, EASJ campus Næstved, efterår 2014/2015

Vejledere: Annette Tjørnemark & Karsten Vandrup

Virksomhed:



Anders Bo Rasmussen

Brian Thorning

Nicklas Kolls

Maren S. Håkansson

Indholdsfortegnelse

1	Indledning	4
2	Problemformulering	4
3	Afgrænsning	4
4	Metode	4
5	Udviklingsproces	6
5.1	Generelle sprint goals	6
5.1.1	Alle sprints	6
5.1.2	Elaboration & construction	6
5.2	Release planning	7
5.2.1	Project management - opstart	8
5.2.2	Revise requirements	8
5.2.3	Project management	9
5.3	Inception	11
5.3.1	Revise requirements	11
5.3.2	Project management	12
5.4	Elaboration sprint 1, Users	14
5.4.1	Revise requirements	14
5.4.2	Project management	14
5.5	Construction sprint 1, Usergroups	15
5.5.1	Revise requirements	15
5.5.2	Project management	15
5.6	Construction sprint 2, Login	16
5.6.1	Revise requirements	16
5.6.2	Project management	16
5.7	Construction sprint 3, Projects & labels	18
5.7.1	Revise requirements	18
5.7.2	Project management	18
5.8	Construction sprint 4, Tasks & import	19
5.8.1	Revise requirements	19
5.8.2	Project management	19
5.9	Construction sprint 5, Time registration & Excel reports	21
5.9.1	Revise requirements	21
5.9.2	Project management	21
5.10	Construction sprint 6, System accounts	22
5.10.1	Revise requirements	22
5.10.2	Project management	22
5.11	Construction sprint 7, Clients & contacts	23
5.11.1	Revise requirements	23
5.11.2	Project management	23
5.12	Construction sprint 8, Google login & Change language	24
5.12.1	Revise requirements	24
5.12.2	Project management	24
6	Problemløsning	25
6.1	Release planning	25

6.1.1	Vision.....	25
6.1.2	Projektorganisation & ressourcer	25
6.2	Inception.....	27
6.2.1	Research af teknologier	27
6.2.2	Business analysis.....	31
6.2.3	Requirements	33
6.3	Elaboration	45
6.3.1	Iteration 1, Users	45
6.4	Construction	50
6.4.1	Iteration 1, Usergroups.....	50
6.4.2	Iteration 2, Login.....	53
6.4.3	Iteration 2, Projects & Labels.....	60
6.4.4	Iteration 4, Tasks & Import	66
6.4.5	Iteration 5, Time registration & Reports	71
6.4.6	Iteration 6, System accounts.....	77
6.4.7	Iteration 7, Clients & contacts.....	81
6.4.8	Iteration 8, Change language & Google login.....	85
7	Konklusion.....	90
7.1	Problemløsning	90
7.1.1	Projektets status.....	90
7.2	Process.....	90
7.2.1	Projektstyring og SCRUM	90
7.2.2	Udviklingsmetoder.....	90
7.2.3	Samarbejde.....	91
7.2.4	Samarbejde med kunde	91
7.2.5	Udviklingsværktøjer.....	91
8	Perspektivering	92
9	Litteraturliste	93
10	Bilag	95
10.1	Oplæg fra Supeo	95
10.2	Mødereferater fra kundemøder.....	97
10.2.1	10. nov. 2014	97
10.2.2	25. nov. 2014	98
10.2.3	9. dec. 2014	98
10.2.4	9. Jan 2015.....	98
10.3	Dokumentation fra Pivotal Tracker	99
10.3.1	Inception	99
10.3.2	Elaboration	99
10.3.3	Construction	99
10.4	Guides.....	100
10.4.1	Netbeans og Git	100
10.4.2	Netbeans, Xampp, Composer, Zend framework 2, PostgreSQL	102
10.4.3	Doctrine – Create entities	104
10.5	Design class diagrams	105

1 Indledning

Følgende rapport omhandler udviklingen af et tidsregistreringssystem til udviklingsvirksomheden Supeo ApS. Projektteamet består af 4 medlemmer fra 5. semester på Datamatikerstudiet på EASJ i Næstved. Systemet skal være et stand-alone system, men også kunne fungere som tillægsmodul til virksomhedens nuværende projektstyringssystem, Pivotal Tracker, samt til andre opgave-systemer. Systemet skal dokumentere nøjagtigt hvor lang tid hver medarbejder bruger på de enkelte arbejdsopgaver. Systemet skal primært aflaste den daglige ledelse i forhold til fakturering, ressourceforbrug og planlægning, samt gøre tidsregistrering meget mere simpel og bedre organiseret for medarbejdere. Denne rapport beskriver udviklingsproces, problemløsning og viser resultater heraf. Vores gruppe arbejder meget tæt sammen derfor deltager vi næsten altid alle sammen i alle opgaver (i mere eller mindre grad) og derfor beskriver vi ikke hvem der har skrevet eller udformet hvad i rapporten. Ligeledes har vi valgt at lægge vægt på at vise processen i rapporten (altså primært henvende os til vejledere og sensor) og derfor beskriver/viser vi, under hovedafsnittet "Problemløsning" artefakternes tilstand på det tidspunkt hvor de udvikles. Dvs. at f.eks. "detailed use cases" kan ændres/udvikle sig gennem projektet. Dette beskrives naturligvis, men det betyder at rapporten ikke er så anvendelig for Supeo som opslagsværk, da en gennemlæsning af "Problemløsning" kræves da fuld forståelse af systemet. Dette vil vi i transition-fasen (som ikke er beskrevet i rapporten), forsøge at afhjælpe ved at udforme artefakter til self-supportability.

2 Problemformulering

Hvordan udvikles et stand-alone system til medarbejder-registrering af tidsforbrug på arbejdsopgaver?
Systemet skal ligeledes:

- fungere som tillægsmodul til Pivotal Tracker
- lave udtræk af tidsregistreringer til fakturering
- lave udtræk af tidsregistreringsrapporter til ledelse

3 Afgrænsning

Det er lavt prioriteret at systemet er gjort klar til at importere data fra andre opgavesystemer (som Pivotal Tracker), men det skal forsøges designet på en sådan måde, at dette kan gøres muligt.

4 Metode

Under udviklingen af ovenstående system, stræbes der efter et godt samarbejde med kunden i form af møder eller workshops, test af systemet undervejs i udviklingen, samt evt. review af aktuelle artefakter^{1 2}.

Vi ønsker at benytte situationsbestemt udviklingsmetode med elementer fra bl.a. UP og OOAD til udviklingen, da vi er usikre på krav og problemsituation³. Fordi vi skal udvikle et større system, mener vi at det er vigtigt med en grundig foranalyse og det er vores erfaring af disse metoder komplimenterer hinanden godt i netop foranalysen hvor systemets formål skal identificeres og afgrænses (UP's Inception).

Fra UP benytter vi ligeledes fase- og disciplin opdelingen, da denne er agil og overskuelig. I Elaboration-fasen laver vi iterationer, indtil vi har "bevist" arkitekturen og at vi kan udvikle systemet efter de valg vi har

¹ Larman s. 47-51

² PS kap. 8

³ Avision s. 507-516

truffet. I construction-fasen udfører vi iterationer til de højst prioriterede dele af systemet er færdigudviklet. Disse iterationer vil evt. ikke gå lige så meget i dybden som elaboration-iterationerne⁴. I disciplinen Requirements, vil vi identificere krav til systemet og i Analysis & Design vil vi designe softwaren på baggrund af krav. Vi vælger udviklingsværktøjer fra metoderne fra iteration til iteration, på baggrund af opgavernes kompleksitet. Dog indeholder metoderne mange ens værktøjer til analyse og design og faktisk også implementation og testing – dog beskriver UP også Transition-fasen. Fælles for metoderne er det derfor også at de tilbyder utrolig mange værktøjer. Alligevel ønsker vi at undersøge om andre udviklingsmetoder med fokus på web-udvikling bør inddrages.

Vi ønsker at benytte par-programmering hvor vi finder det relevant, men ud over dette benytter vi ikke elementer fra XP og beskriver det ikke nærmere. Vi mener dog at par-programmering kan sikre kvalitet af kode, samt bevirkede at alle i teamet opnår sammenlignelige programmeringsfærdigheder i valgte sprog.

Vi ønsker at undersøge hvilke programmeringssprog og teknologier som kan og bør anvendes til udviklingen af systemet og først herefter foretage et valg i samarbejde med vores kunde. Vi ønsker ikke at udføre unitests i alle iterationer, men vil blot gøre det i relevante iterationer.

Til projektstyring ønsker vi at anvende elementer fra SCRUM. Denne metode er relevant for os at benytte, da der er tale om at større system som vi ønsker at udvikle iterativt og derfor kan SCRUM, som efter vores vurdering indeholder mange nyttige redskaber og principper, hjælpe os til at styre processen. Vores kunde (kontaktperson) vil fungere som project owner så godt som det kan lade sig gøre og vil deltage på møder som det er muligt. Rollen som SCRUM master skal gå på skift i gruppen fra sprint til sprint. Ligeledes vil rollerne som analyst, designer, implementer og tester⁵ gå på skift og alle i projektgruppen vil skulle påtage alle roller igennem projektet.

Fra SCRUM benyttes ligeledes prioriteret backlog⁶, daily SCRUM⁷, scrum board i form af Pivotal Tracker (se dokumentation afsnit 10.3), samt review⁸- og retrospect-møder, for at sikre kvalitet og evt. forbedring af proces. Ligeledes udarbejdes sprint goals⁹ og sprint plan før hvert sprint, indeholdende både kvantitative og kvalitative referencelinjer for hvert mål. På sprint backlog/plan¹⁰ skal der på hver opgave være en overordnet ansvarlig. Den ansvarlige i teamet er ikke nødvendigvis ene om at løse opgaven. Teamet kan hjælpe hinanden på kryds og tværs som nødvendigt både med opgaver og rapportskrivning. Dette har vi besluttet da vi er en gruppe som arbejder meget tæt sammen og derfor ved at vi laver næsten alle opgaver i fællesskab.

Teamet opsætter et fælles kontor hvor der arbejdes fra hver dag og for at etablere projektet, vil vi gennemføre fasen Release planning¹¹.

⁴ Larman s. 33

⁵ http://upedu.org/process/workers/ovu_works.htm

⁶ Scrum s. 5

⁷ Scrum s. 4

⁸ Scrum s. 14

⁹ Scrum s. 12

¹⁰ Scrum s. 6

¹¹ Scrum s. 9

5 Udviklingsproces

Dette afsnit beskriver hele udviklingsprocessen og indeholder nye/forandrede krav til systemet, risk lists, sprint-planer og sprint goals, samt dokumentation af review- og retrospect-møder. For at kunne planlægge hvert sprint bedst muligt og for at opdage nye krav så tidligt som muligt, identificeres nye/forandrede krav til systemet, samt risks¹² i hvert sprint, som UP foreskriver. For at sikre kvalitet af artefakter og produkt, laves sprint goals, som beskriver hver artefakts endelige tilstand. Under review og retrospect godkendes eller tilrettes alle artefakter ved at måle artefakternes tilstand op mod sprint goals. Ligeledes diskuteses forrige sprint og dets proces (samarbejde mv), for at afgøre om processen skal ændres eller forbedres. Tilslut planlægges et nyt sprint. Til dette anvendes risk list, sprint goals og informationer fra retrospect-møder.

5.1 Generelle sprint goals

Nedenstående sprint goals skal opfyldes for hvert eneste sprint.

5.1.1 Alle sprints

- Revise requirements: Alle nye eller forandrede krav til systemet skal identificeres og beskrives.
- Review & retrospect: Review af alle sprint goals skal dokumenteres og retrospect af processen skal beskrives (hvordan er sprintet forløbet og hvad kan gøres anderledes/optimeres).
- Revised risk list: Skal indeholde alle identificerede risks. Disse skal være risiko-estimeret og have en tilhørende mitigation strategy.
- Sprint plan: En sprint plan indeholdende alle opgaver for kommende sprint. Sprintplanen skal vise start, deadline, ansvarlig medarbejder og varighedsestimat for hver opgave. *Den/de ansvarlige medarbejder(e) er ikke nødvendigvis ene om at udføre opgaven, men er overordnet ansvarlig for løsning af opgaven samt for rapportskrivning som omhandlende dette. Vores gruppe arbejder meget tæt sammen derfor deltager vi næste altid alle sammen i alle opgaver i mere eller mindre grad.*
- Sprint goals: Mål skal beskrives så nøjagtigt som muligt, altså både kvantitet og kvalitet, så teamet ved review-mødet målbart kan afgøre om artefakterne har den rette og ønskede tidstand.

5.1.2 Elaboration & construction

Nedenstående sprint goals skal ydermere opfyldes for hvert sprint i elaboration- og construction-fasen:

- Domain model (requirements): Domæne-modellen skal indeholde alle konceptuelle klasser som er identificerede i sprintet, samt foregående sprints. Klasserne skal indeholde identificerede attributter, vise relationer og multiplicitet.
 - Vi vælger at lave domæne-modeller for at modellere/illustrere "den virkelig verden". Vi ønsker altså at forstå omgivelser og relationer og herudaf danne konceptuelle klasser^{13 14}.
- System sequence diagrams *eller* detailed use cases (requirements): *Kun for relevante use cases.* Diagrammerne/beskrevelserne skal sekvensvis illustrere/beskrive hvordan systemet og bruger udveksler data.
 - Disse diagrammer og beskrivelser ønsker vi at lave hvor det er relevant, da vi mener at vi herved får diskuteret og bliver enige om hvordan systemet og bruger skal udveksle data,

¹² Larman s. 27

¹³ Larman s. 131-161

¹⁴ OOAD s. 4

samt hvilke data der skal udveksles¹⁵. Artefakterne kan støtte eller helt afløse vores prototypes og give en forståelse for kompleksitet og nødvendige funktioner, som alternativ til f.eks. en funktionsliste. Diagrammerne/beskrivelserne kan også hjælpe os til at tænke over hvordan objekterne påvirker hinanden ved ændring i tilstand og ikke mindst give vores kunde en forståelse af use cases. *Diagrammerne/beskrivelserne skal ikke betragtes som en færdig-designet software, men som et værktøj til at forstå use cases (software designes i disciplinen Analysis & Design) og til at bestemme bruger-input.*

- Database diagram (Analysis & Design): Skal vise alle tabeller og attributter. Ligeledes skal der være anført primary- og foreign keys, relationer og multiplicitet og også beskrive unikke data og vise om attributter er nullable.
 - Vi ønsker at designe databasen ud fra domæne-modellerne for at sikre at tabellerne i databasen har de rette relationer og multiplicitet.
- Design class diagram (Analysis & Design): Dette diagram kan for overskuelighedens skyld tegnes i klynger. Diagrammet skal vise alle sprintets klasser (utilities mv. kan undlades og blot beskrives), klassernes attributter, metoder, relationer og multiplicitet¹⁶ ¹⁷.
 - Anvendes til design af software. Dette vil sidenhen lette programmeringsarbejdet og ligeledes vil vi gerne sikre at systemet er designet fleksibelt, så videreudvikling og vedligehold er muligt. Diagrammet vil sidenhen også kunne anvendes af Supeo's medarbejdere, som heraf kan opnå større forståelse af systemet.
- Implementering: Koden af god kvalitet. Direkte sporbarhed mellem diagrammer og kode. Koden skal være refaktureret og kommenteret. Nødvendige informationer om implementering, som ikke fremgår af design klasse diagrammet, skal beskrives i afsnittene "Implementation". Er anvendeligt/nødvendigt for vores kunde i forhold til forståelse, vedligehold og videreudvikling
 - Det er vigtigt med en kommenteret, refaktureret kode af god kvalitet, da dette muliggør vedligehold og videreudvikling. Ligeledes er sporbarhed vigtigt for forståelse af systemet.
- Test: Programmet skal funktionstestes af projektteamet. Funktionstests skal overordnet beskrives, men der skal ikke skrives test cases for hver enkelt case. Ved funktionstest skal følgende som minimum testes: Use cases, validering, udtræk, grænseværdier, oprettelse af ens objekter og fejlbeskeder. Også bruger-rettigheder når dette implementeres.
 - Vi vil udføre disse tests i hver iteration, da vi ønsker at identificere og udbedre fejl, så hurtigt som muligt. Her testes alt fra use cases, udtræk, grænseværdier, oprettelse af ens objekter, sletninger mv. Både for at sikre funktionerne, men også for at sikre at brugerne får de rigtige fejlmeldelser og kan "recover" fra fejl i systemet.

Vi ville som udgangspunkt gerne have at vores kunde kunne udføre userability test i hver iteration, men da vi ikke er sikre på at kunne mødes med vores kunde så ofte, vil vi bede vores kunde teste (eller på anden måde godkende) use cases for ca. 2 iterationer pr. møde. Dette skal gøres løbende, da vi ikke ønsker at skulle lave mange, og måske meget afgørende, ændringer sidst i projektet. Vi dokumenterer ikke tests ved at lave test cases, men laver et referat af hvert møde med Supeo (10.2).

5.2 Release planning

Vi udfører denne fase, for at etablere projektet bedst muligt.

¹⁵ Larman 61-87

¹⁶ Larman s. 251

¹⁷ OOAD s. 270-272

5.2.1 Project management - opstart

5.2.1.1 Sprint plan & sprint goals, Release planning

Scrum master: Brian

Projektet skal afleveres d. 14. januar 2015 – eksamen d. 29.-30. januar 2015

Uge	Dato	Mandetimer til rådighed	Opgave	Startdato	Varighedsestimat i mandetimer	Deadline	Hvem
43	20-okt	18	Gennemgang af oplæg fra Supeo Ressourcer: Team, tekniske, tid Risk list Sprint plan & sprint goals release planning Vision Research af teknologier Opsætning af sever Opsætning af valgte programmer/teknologier Opsætning af pivotal tracker & cloud station Review & retrospect Risk list Sprint plan & sprint goals Inception	20-okt	6	20-okt	Nicklas, Brian, Anders
	21-okt	18		20-okt	6	20-okt	Nicklas, Brian, Anders
	22-okt	18		20-okt	1,5	20-okt	Nicklas, Brian, Anders
	23-okt	18		20-okt	1,5	20-okt	Nicklas, Brian, Anders
				20-okt	3	20-okt	Nicklas, Brian, Anders
				20-okt	27	23-okt	Nicklas, Brian, Anders
				22-okt	8	23-okt	Nicklas, Brian, Anders
				22-okt	10	23-okt	Nicklas, Brian, Anders
				23-okt	3	23-okt	Nicklas, Brian, Anders
				23-okt	1,5	23-okt	Nicklas, Brian, Anders
				23-okt	1,5	23-okt	Nicklas, Brian, Anders
		72			3	23-okt	Nicklas, Brian, Anders

Sprint goals (udover generelle sprint goals):

- Ressourcer: Komplet liste over tekniske- (hardware og software), team- (medlemmer og kompetencer) og tidsressourcer.
 - Denne ressource-liste laves for at teamet kan afgøre om projektet er muligt og om alle nødvendige ressourcer er til rådighed¹⁸.
- Vision: En kort tekst som beskriver kundens syn på det system som ønskes udviklet. Visionen skal laves ud fra mini-oplægget fra Supeo og skal, med evt. modifikationer, afspejle dette.
 - Visionen laves for at give teamet et billede af det system kunden, som udgangspunkt, ønsker udviklet. Denne fungerer også, som en uformel aftale mellem team og kunde¹⁹. Aftaler vil dog dokumenteres undervejs i projektet i form af mødereferater.

5.2.1.2 Risk list, projektstart

Impact H (high), M (medium), L (low) Magnitude: 1-10 (10 højest)

Magnitude	Description	Impact	Indicator	Mitigation strategy
8	Nye teknologier & frameworks	H	Projektplan/tid	Der laves en meget grundig research på mulige frameworks & teknologier og valg heraf.
2	Manglende gruppemedlem i 2 uger (praktik)	L	Samarbejde & ressourcer	Projektteamet bruger 1 hel dag på at sætte et manglende medlem ind i projektet og programmer

5.2.2 Revise requirements

Systemet er overordnet beskrevet i den udformede vision. Det skal dog diskuteres nærmere om man skal kunne oprette projekter og opgaver i systemet, da dette allerede gøres i Pivotal Tracker og der er

¹⁸ Scrum s. 9

¹⁹ Larman s. 58

usikkerhed omkring hvorvidt systemet er et stand-alone system eller et tillægsmodul til Pivotal Tracker. Vi skal ligeledes diskutere med product owner hvordan systemet skal udveksle data med Economic, da projektteamet ikke har dette program eller adgang til det.

5.2.3 Project management

5.2.3.1 *Review & retrospect, Release planning*

Dato: 23. oktober 2014 - deltagere: Nicklas, Anders og Brian

- Visionen er udformet og afspejler Supeo's nuværende ønsker til systemet.
- Listen af ressourcer, ny risk list, sprint plan og spring goals er godkendt. Vi har valgt udviklingsmetoder til foranalyse/inception (se sprint plan 5.2.3.3). Vi har ikke programmet Economic og derfor skal dataudveksling med dette program diskuteres med product owner.

Retrospect: Teamet har arbejdet godt sammen om research af teknologier og Pivotal Tracker fungerer på nuværende tidspunkt godt som projektstyringsredskab.

5.2.3.2 *Revised risk list*

Impact H (high), M (medium), L (low) Magnitude: 1-10 (10 højest)

Magnitude	Description	Impact	Indicator	Mitigation strategy
7	Nye teknologier & frameworks	H	Projektplan/tid	Der laves en meget grundig research på mulige teknologier & frameworks og valg heraf.
6	Uklare krav og tilgængelighed til kunde	M	Projektplan/tid	Vi skal være meget forsigtige med ikke at tage for mange beslutninger omkring systemet, før vi har afholdt første møde med Supeo d. 10. nov (det tidligste møde vi kunne få).
5	Projektteamet har ikke programmet Economic	M	Manglende datakommunikation mellem systemer	Hvis programmet ikke kan anskaffes laves dataudtræk til Excel, som så kan indlæses i Economic. Dette skal dog afklares med vores kunde
2	Manglende gruppemedlem i 2 uger (praktik)	L	Samarbejde & ressourcer	Projektteamet bruger 1 hel dag på at sætte et manglende medlem ind i projektet og programmer

5.2.3.3 *Sprint plan & goals, Inception*

Scrum master: Nicklas

Uge	Dato	Mandetimer til rådighed	Opgave	Startdato	Varighedsestimat i mandetimer	Deadline	Hvem
44	27-okt	18	Research rapport	27-okt	36	28-okt	Nicklas, Brian, Anders
	28-okt	18	Virksomhedsanalyse	29-okt	9	30-okt	Nicklas, Brian, Anders
	29-okt	18	Supeos nuværende system & hændelser	29-okt	6	30-okt	Nicklas, Brian, Anders
	30-okt	18	Stakeholders	29-okt	3	30-okt	Nicklas, Brian, Anders
			Rige billeder	29-okt	18	30-okt	Nicklas, Brian, Anders
		72			72		
45 (& 46)	03-nov	21	Sætte Maren ind i projektet, samt installation af programmer	03-nov	21	03-nov	Maren, Brian, Anders
	04-nov	28	BATOFF	04-nov	6	07-nov	Maren, Brian, Anders
	05-nov	21	Systemdefinitioner	04-nov	9	07-nov	Maren, Brian, Anders
	06-nov	28	Problemformulering	04-nov	4	07-nov	Alle
	07-nov	28	Metode & afgrensnings	04-nov	4	07-nov	Alle
	10-nov	28	Find actors & use cases incl. use case diagram	04-nov	25	07-nov	Nicklas
			Backlog	05-nov	4	07-nov	Alle
			Logo design	05-nov	5	07-nov	Brian
			Prototyping	05-nov	20	07-nov	Nicklas & Maren
			Arkitektur	05-nov	28	07-nov	Brian & Anders, Maren
			Mode Supeo	10-nov	14	10-nov	Alle
			Revise requirements	10-nov	4	10-nov	Alle
			Review & retrospect	10-nov	4	10-nov	Alle
			Risk list	10-nov	2	10-nov	Alle
			Sprint plan & sprint goals, elaboration sprint 1	10-nov	4	10-nov	Alle
		154			154		

Inception-fasen er planlagt som et langt sprint på varighed af ca. 2 uger. Fremover vil vi lave sprints af ca. 1 uges varighed, men da vi vil lave en grundig foranalyse i dette sprint, synes vi at 2 uger er nødvendigt og mener ikke at de vil være meningsfuldt at sprintet bliver splittet op i flere mindre sprints.

Sprint goals (udover generelle sprint goals):

- Research-rapport & valgt af framework: En grundig research rapport, som sammenligner mulige applikationstyper og frameworks, udformes og valg begrundes. Ydermere ønskes research af følgende: PHP unittesting, Doctrine, repositories/versionsstyringsredskaber, server, Pivotal Tracker's API og mulighed for synkronisering med Pivotal Tracker.
 - Supeo har ikke fremsat krav om sprog i backend eller til brug af frameworks og andre teknologier. Dog ønsker vores kunde en redegørelse for valg af sprog. Derfor vil vi lave grundlæggende research for at kunne tage et kvalificeret valg.
 - Under vores forrige projekt havde vi problemer med serveren under Transition-fasen. Da vi gerne vil undgå dette problem i projektet, vil vi tidligt teste kommunikation mellem server og klient fra Supeo's server.
- Virksomhedsanalyse: En kort beskrivelse af virksomheden, dens kunder og produkter, samt en SWOT-analyse.
 - For at få en bedre forståelse af vores kundes virksomhed²⁰, vil vi lave en mindre virksomhedsbeskrivelse og en swot-analyse. Dette kan hjælpe med at afdække hvor systemet kan lette kunden, samt hvilken fleksibilitet systemet skal have i forhold til videreudvikling. Denne analyse er ikke direkte nødvendig, men kan afdække nye krav eller muligheder, så derfor mener vi den er værd at lave.
- Nuværende system, hændelser og rige billeder:²¹ En beskrivelse af nuværende arbejdsgange (tidsregistrering) og hændelser, samt rige billeder.

²⁰ Scrum s. 9

²¹ OOAD s. 24

- Ved at opnå en bredere forståelse af Supeo's nuværende system og de hændelser der sker, kan vi bedre forstå og identificere krav til systemet²²
- BATOFF, systemdefinitioner og rige billeder: Systemet beskrives vha. BATOFF-kriterierne og en systemdefinition udformes. Ligeledes udformes 1 eller flere alternative systemdefinitioner.
 - Ved hjælp af BATOFF-kriterierne²³ kan vi lave systemdefinitioner²⁴. Vi ønsker at lave alternative definitioner ud fra identificerede hændelse og krav, så vores kunde kan vælge den løsning som han mener er bedst.
- Problemformulering, metode og afgrænsning: Efter en systemdefinition er valgt af kunden, laves en endelig problemformulering. Ligeledes beskrives metoder til løsning af problemformuleringen, samt evt. afgrænsning.
- Use case diagrammer (laves i klynger for overskuelighed): Use case diagrammer indeholdende alle identificerede use cases og alle roller²⁵. Diagrammerne skal vise hvilke actors der benytter hvilke use cases og skal kunne benyttes når brugerrettigheder skal implementeres.
 - *Dette diagram kan ændre sig løbende under processen og vil derfor løbende blive opdateret.*
- Backlog: Use cases prioriteres og backlog udformes i Pivotal Tracker (se dokumentation 10.3), således at vores kunde kan (om)prioritere som ønsket.
 - Ved at product owner kan se og evt. omprioritere backlog, sikre vi at arbejdet udføres, som kunden finder det vigtigst. *Backlog kan ændre sig under processen og vil derfor løbende blive opdateret.*
- Prototypes: Der udformes prototypes i form af hurtigt skitser af skærmbilleder på papir²⁶. Disse testes med product owner som testperson.
 - Vi har undersøgt flere udviklingsmetoder som fokuserer på web development. Metoderne benytter alle prototyping af denne type. Ved at teste prototyperne på vores kunde, kan vi specificere krav til systemet, evt. tilføje eller fjerne use cases, samt sikre en velfungerende GUI-opbygning, således at vi undgår at laver store ændringer for sent i processen.
- Arkitektur: Vælges et framework skal arkitekturen og lagdelingen beskrives og illustreres detaljeret og sammenlignes med den MVC arkitektur vi før har anvendt i Java.

5.3 Inception

5.3.1 Revise requirements

Supeo har valgt systemdefinition (se referat fra møde, afsnit 10.2.1). Systemet skal både kunne fungere som stand alone applikation og som tillægs-modul til opgave-systemer som f.eks. Pivotal Tracker. Dette

²² OOAD s. 48

²³ OOAD s. 38

²⁴ OOAD s. 22

²⁵ Larman s. 90

²⁶ OOAD s. 31

medfører at brugerne kan være vidt forskellige med meget forskellig IT-erfaring, samt at terminologien ikke skal tale til udviklere, men til en meget bred gruppe af brugere.

Export af data til Economic: Kan eksporteres til excel til en start. Dog har dette ikke så høj prioritet og derfor er dette sat lavt i backlog.

Et yderligere krav er at der kan tilknyttes kontaktpersoner til kunder (contacts & clients). Systemet skal kunne anvendes som kundedatabase og medarbejderdatabase, men dette skal ikke være pålagt bruger-virksomheden at den anvendes således. Dvs. at ikke alle oplysninger om kunder og ansatte skal, men kan angives, når disse oprettes i systemet. Både kunder og medarbejdere skal have et system-id og et id man selv kan indtaste (kan f.eks. være et id fra et andet system eller et CVR).

Brugerne af systemet skal have permissions alt efter hvilken brugergruppe de tilhører. Brugergrupperne kan være Admin, Project manager eller User. Brugerne skal evt. kunne logge på med google account, men dette er ikke et krav.

Rapporter om tidsregistrering skal være excel-format og hvis der er tid til det, også pdf. Dette skyldes at der skal kunne arbejdes videre med disse data i excel.

Systemet skal synkroniseres med supporting system (Pivotal Tracker) så ofte som muligt. Det er ikke et krav, men hvis det kan lade sig gøre, skal systemet kunne starte og afslutte opgaver i Pivotal Tracker.

Systemet skal anvende Zend framework 2. Dette var ikke et krav tidligere, men Supeo lægger nu højere vægt på dette. Det var dog også det framework vi selv havde valgt under vores research – netop af den grund, at det ville lette videreudvikling og vedligehold for Supeo. Det er ligeledes et stort ønske fra Supeo at vi benytter Doctrine sammen med frameworket, men ikke et ultimativt krav.

5.3.2 Project management

5.3.2.1 Review & retrospect

Dato: 10. november 2014 - deltagere: Nicklas, Anders, Maren og Brian, Troels

Vores kunde har valgt og godkendt systemdefinition og testet prototypes - backlog og use case diagram er herefter opdateret.

Problemformulering er udarbejdet ud fra kundens valg af systemdefinition.

Valg af framework (ZF2) er godkendt (krav nu) og vi har begrundet valg og beskrevet arkitektur (både med og uden brug af Doctrine)

Alle øvrige artefakter er gennemgået og godkendt og Maren er blevet sat godt ind i projektet. Vi har manglet lidt struktur i Inception-fasen, men det har også været en *meget* omfangsrig fase og vi har fået undersøgt meget – både tekniske aspekter og krav til systemet. Det har været lidt svært at få afklaret krav, da vores møde med Supeo var planlagt sent (det tidligste vi kunne få). Derfor har vi allerede nu aftalt et nyt møde med vores kontaktperson fra Supeo (Troels).

5.3.2.2 Revised risk list

Impact H (high), M (medium), L (low) Magnitude: 1-10 (10 højest)

Magnitude	Description	Impact	Indicator	Mitigation strategy
7	Nye sprog og nyt framework, ingen	M	Projektplan/tid og kodekvalitet	Vi har valgt at arbejde med nogle meget nemme use cases i første sprint i elaboration-fasen. Dermed

	erfaring med Doctrine			håber vi at vi kan opnå mere erfaring med php, zend framework og doctrine, til vi skal i gang med mere avancerede use cases.
3	Valgte teknologier vs. programmeringseksemen	L	Resultat eksamen	Da vi har valgt at anvende doctrine og zend framework 2, undgår vi noget programmering og arkitekturen er på forhånd skabt af frameworket. Vi mener dog at opgaven alligevel er meget kompleks og derfor mener vi ikke et det er et problem i forhold til programmerings-eksamen. Alligevel skal vi sikre at vi forstår arkitekturen i frameworket rigtig godt, samt diskutere denne risk med vores vejledere.

Vi har fjernet risk ang. manglende gruppemedlem, da vi nu er fuldtallige i teamet og alle sat godt ind i projektet. Vi har ligeledes fjernet risk ang. uklare krav, da vi har lavet en grundig foranalyse og opdateret backlog. Til slut har vi fjernet risk som omhandler Economic, da vi i stedet for kommunikation mellem systemer, vil genererer excel-filer. Vi har tilføjet risk omhandlende brug fra framework og doctrine.

5.3.2.3 Sprint plan & goals, elaboration sprint 1 (Users)

Scrum master: Maren

Use cases: "Create user", "Edit user", "Delete user" og "Search user", samt "Edit profile"

Uge	Dato	Mandetimer til rådighed	Opgave	Startdato	Varighedsestimat i mandetimer	Deadline	Hvem
46	11-nov	28	Domain model	11-nov	2	11-nov	Nicklas & Brian
	12-nov	28	System sequence diagrams/detailed use cases	11-nov	4	11-nov	Nicklas, Maren & Brian
	13-nov	28	Database diagram	11-nov	4	11-nov	Nicklas & Brian
	14-nov	28	Design class diagram (incl. klassekommentering)	11-nov	2	11-nov	Maren
			Research doctrine og test	11-nov	14	11-nov	Brian & Anders
			Implement database entities & generate model	11-nov	2	11-nov	Anders
			Implement main design	11-nov	14	12-nov	Brian
			Implement use cases	11-nov	26	13-nov	Maren og Nicklas
			Styling	12-nov	12	13-nov	Brian og Anders
			Unittesting exception handling	12-nov	4	13-nov	Anders
			Funktionsitest	14-nov	8	14-nov	Alle
			Implement changes	14-nov	6	14-nov	Alle
			Revise requirements	14-nov	4	14-nov	Alle
			Revise risk list	14-nov	2	14-nov	Alle
			Review & retrospect	14-nov	4	14-nov	Alle
			Sprintplan Construction it 1 & Sprint goals	14-nov	4	14-nov	Alle
		112			112		
47			Håndtering af usergroups				
48			Håndtering af login			24. nov kl 12: Møde med vejleder 25. nov kl 10: Møde hos Supeo	
49			Håndtering af projects og labels				
50			Håndtering af tasks og import				
51			Håndtering af tidsregistrering & reports (export til excel)				
52 (2 dage)			Import fra andre systemer				
2			Håndtering af clinets og contacts				
3 (2 dage)			Håndtering af sprogskifte og evt. login m. google account			Aflevering d. 14. jan.	
14.-28. jan			Transition & evt. mindre tilføjelser			Eksamens 29.-30. jan.	

Sprint goals (udover generelle sprint goals):

- Unittest som tester at exceptions kastes ved fejl i databasen.

- Vi ønsker ikke at teste alle metoder vha. unittest, da vi mener at vores funktionstests er tilstrækkelige. Dog mener vi det er vigtigt at teste om exceptions bliver kastet ved fejl i databasen, så vi kan søge for at databasen er velfungerende og at brugerne kan "recover".

Vi har valgt ikke at lave sequence diagrams til at vise metodekald og lagdeling, da arkitekturen allerede er fastlagt ved brug af frameworkt. Frameworkets arkitektur er redegjort for i Inception (se afsnit 6.2.3.8).

5.4 Elaboration sprint 1, Users

5.4.1 Revise requirements

I denne fase har vi ikke identificeret nye krav til systemet, men der skal naturligvis påføres brugerrettigheder på de use cases som netop er udviklet, når sikkerhed og rettigheder implementeres.

5.4.2 Project management

5.4.2.1 Review & retrospect

Dato: 14. nov. 2014 -deltagere: Maren, Nicklas, Brian og Anders

Alle artefakter incl. kode er gennemgået og godkendt og der er udført unittesting.

Systemet er funktionstestet af projektteamet og små fejl/ændringer er udbedret.

Retrospect: Det har været godt at starte med disse "nemme" use cases og alle kan nu programmere i de nye sprog og forstår lagdelingen. Dog går programmeringen langsommere end vi er vant til pga. manglende erfaring, så derfor skal de næste sprints ikke planlægges for presset.

Det går ikke så godt med at anvende Git og branches, samt Pivotal Tracker. Vi er en gruppe som arbejder meget tæt sammen og mødes på vores "kontor" hver dag og derfor er disse værktøjer næsten mere til gene for os ind imellem. Alligevel ønsker vi dog at blive ved med at anvende dem, for at undersøge om vi blot mangler rutine med værktøjerne eller om vi skal skifte til et fysisk SCRUM board, som vi har rigtig gode erfaringer med. Dette vil kræve at "kommunikationsform" med Supeo ændres, da vi "kommunicerer" med product owner ved at opdaterer status på stories i PT. Det er ikke den bedste kommunikationsvej og endvidere en-vejs.

5.4.2.2 Revised risk list

Impact H (high), M (medium), L (low) Magnitude: 1-10 (10 højest)

Magnitude	Description	Impact	Indicator	Mitigation strategy
7	Nye teknologier	M	Projektplan/tid og kodekvalitet	Vi fortsætter med at sætte god tid af til implementering i hvert sprint
1	Valgte teknologier vs. programmeringseksemplar	L	Resultat eksamen	Efter at vi har gennemført elaboration-fasen kan vi se at projektet, selv med brug af framework og doctrine, er meget komplekst og derfor er denne risk mindsket. Dog vil vi stadig gerne have svar fra vores vejledere, før den fjernes fra listen.

5.4.2.3 Sprint plan & goals, construction sprint 1 (Usergroups)

Scrum master: Anders

Use cases: "Add user to usergroup" og "Remove user from usergroup"

Uge	Dato	Mandetimer til rådighed	Opgave	Startdato	Varighedsestimat i mandetimer	Deadline	Hvem
47	17-nov	28	Research ACL & security	17-nov	28	17-nov	Alle
	18-nov	28	Domain model	18-nov	4	18-nov	Alle
	19-nov	28	Database diagram	18-nov	2	18-nov	Nicklas & Maren
	20-nov	28	Design class diagram (incl. klassekommentering)	18-nov	1	18-nov	Anders
	21-nov	21	Implement database entities & generate model	18-nov	2	18-nov	Brian
			Implement use cases incl. styling	18-nov	54	20-nov	Nicklas og Maren
			Funktionstest	20-nov	14	20-nov	Brian og Anders
			Implement changes	21-nov	16	21-nov	Alle
			Revise requirements	21-nov	2	21-nov	Alle
			Revise risk list	21-nov	2	21-nov	Alle
			Review & retrospect	21-nov	4	21-nov	Alle
			Sprintplan Construction it 2 & Sprint goals	21-nov	4	21-nov	Alle
		133			133		

Sprint goals:

- Research ACL & security: ACL (håndtering af brugerrettigheder i ZF2) skal undersøges og beskrives således at research (bl.a.) kan danne grundlag for bestemmelse af konceptuelle klassers relationer. Ligeledes laves research vedr. sikkerhed som er relevant for dette sprint.

Vi har valgt at vente med unit testing til næste iteration, hvor Login mv. skal implementeres, da vi så bliver i stand til at teste sikkerheden bedre.

5.5 Construction sprint 1, Usergroups

5.5.1 Revise requirements

Vi har ikke identificeret nye requirements i dette sprint, men har opdateret de use cases som blev implementeret i forrige sprint, således at en bruger tilføjes til en bruger-gruppe ved "Create user" og "Edit user".

5.5.2 Project management

5.5.2.1 Review & retrospect

Dato: 21. november 2014 - deltagere: Nicklas, Brian, Anders, Maren

Alle artefakter er gennemgået og godkendt.

Umiddelbart troede vi at vi havde planlagt for lidt aktiviteter i dette sprint, men som vi også oplevede i sidste sprint, tager aktiviteterne længere tid at udføre, når nye værktøjer tages i brug. Vi har kæmpet med Doctrine og "mange til mange"-relationer, da det var svært for os at finde dokumentation om dette (samtidig med ACL) og vi prøvede derfor forskellige metoder indtil vi fandt en brugbar løsning. Det kan altså konstateres at vi fortsat skal planlægge god tid til hver iteration.

5.5.2.2 Revised risk list

Impact H (high), M (medium), L (low) Magnitude: 1-10 (10 højest)

Magnitude	Description	Impact	Indicator	Mitigation strategy
7	Nye teknologier	H	Projektplan/tid og kodekvalitet	Vi fortsætter med at sætte god tid af til implementering i hvert sprint
5	Manglende dokumentation	M	Projektplan/tid og kodekvalitet	Vi fortsætter med at sætte god tid af til implementering i hvert sprint

1	Valgte teknologier vs. programmeringseksempler	L	Resultat eksamen	Efter at vi har gennemført elaboration-fasen kan vi se at projektet, selv med brug af framework og doctrine, er meget komplekst og derfor er denne risk mindsket. Dog vil vi stadig gerne have svar fra vores vejledere, før vi fjernes risken.
---	--	---	------------------	---

5.5.2.3 Sprint plan & goals, construction sprint 2 (Login)

Scrum master: Brian

Use cases: "Login", "Confirm user", "Logout", "Reset password", "Confirm reset password"

Uge	Dato	Mandetimer til rådighed	Opgave	Startdato	Varighedsestimat i mandetimer	Deadline	Hvem
48	24-nov	28	Research security	24-nov	10	25-nov	Alle
	25-nov	28	Domain model	24-nov	4	24-nov	Alle
			System sequence diagrams/detailed use cases	24-nov	8	24-nov	Alle
26-nov	28		Database diagram	24-nov	2	24-nov	Maren og Anders
27-nov	28		Design class diagram (incl. klassekommentering)	24-nov	4	24-nov	Nicklas
			Møde med Anette vejleder kl 12	24-nov	8	24-nov	Alle
28-nov	21		Implement database entities & generate model	24-nov	2	24-nov	Maren
			Møde med Supeo kl 10 incl. userbility test	25-nov	8	25-nov	Alle
			Implement use cases incl. styling	25-nov	25	27-nov	Brian
			Opdatering af tidligere use cases	25-nov	8	27-nov	Nicklas & Maren
			Unittesting	27-nov	16	27-nov	Anders
			Funktionstest	27-nov	12	27-nov	Nicklas & Maren
			Implement changes	28-nov	12	28-nov	Nicklas
			Revise requirements	28-nov	4	28-nov	Alle
			Revise risk list	28-nov	2	28-nov	Alle
			Review & retrospect	28-nov	4	28-nov	Alle
			Sprintplan Construction it 3 & Sprint goals	28-nov	4	28-nov	Alle
		133			133		

Sprint goals (udover generelle sprint goals):

- Research af sikkerhed: Yderligere research og beskrivelse af sikkerhed som er relevant for dette sprint.
- Unit testing: Test usergroups (permissions) & security vha. unit tests.
- Userbility testing: Under det planlagte møde med Supeo skal de use cases som er implementerede testes/godkendes og eventuelle ændringer skal derefter tilføjes i systemet.

5.6 Construction sprint 2, Login

5.6.1 Revise requirements

Der er ikke identificeret nye krav i denne iteration.

5.6.2 Project management

5.6.2.1 Review & retrospect

28. november 2014 – deltagere: Anders, Brian, Nicklas og Maren

Alle artefakter er gennemgået og godkendt. Sikkerhed er implementeret og testet ved både funktionstests og unit tests inkl. tidligere use cases, da vi gerne ville teste de forskellige brugergrupper.

Under vores møde med Supeo har product owner godkendt de use cases som indtil nu er implementeret. Dog lå mødet først i sprintet og derfor kunne vi desværre ikke vise product owner hvordan sikkerhed var implementeret, da dette ikke var færdigt implementeret endnu og derfor vises dette ved næste møde d. 9. dec.

Det fungerer ikke godt for vores gruppe at benytte Pivotal Tracker. Vi får ikke opdateret det hver dag og får ikke tilføjet vores commits til hver story. Bl.a. dette gør at vores kontakt med vores kunde ikke er optimal. Vi vil benytte PT indtil vi igen mødes med vores kunde og måske kan finde en anden løsning. Indtil da anvender vi også et fysisk scrum-board.

5.6.2.2 Revised risk list

Impact H (high), M (medium), L (low) Magnitude: 1-10 (10 højest)

Magnitude	Description	Impact	Indicator	Mitigation strategy
5	Pivotal Tracker er ikke et godt redskab for vores gruppe	M	Manglende kommunikation med kunde	Vi benytter Pivotal Tracker uden commits og benytter også fysisk SCRUM-board. Vi må på næste møde med Supeo d. 9. dec finde en løsning.
3	Nye teknologier	L	Projektplan/tid og kodekvalitet	Vi har fået mere erfaring med frameworket, men skal stadig bruge mere tid end normalt på implementering. Der laves altid review af kode.
2	Manglende dokumentation	L	Projektplan/tid og kodekvalitet	Vi fortsætter med at sætte god tid af til implementering i hvert sprint, samt review af kode.

Vi har afholdt møde med én af vores vejledere og fjerner derfor risk som omhandler brug af framework og Doctrine.

5.6.2.3 Sprint plan & goals, construction sprint 3 (Projects & labels)

Scrum master: Nicklas

Use cases: "Create project", "Edit project", "Delete project", "Search project", "Create label", "Edit label", "Delete label", "Add user to project", "Remove user from project"

Uge	Dato	Mandetimer til rådighed	Opgave	Startdato	Varighedsestimat i mandetimer	Deadline	Hvem
49	01-dec	28	Domain model	01-dec	4	01-dec	Alle
	02-dec	28	System sequence diagrams/detailed use cases	01-dec	8	01-dec	Alle
	03-dec	28	Database diagram	01-dec	2	01-dec	Anders
	04-dec	28	Design class diagram (incl. klassekommentering)	01-dec	4	01-dec	Brian & Nicklas
	05-dec	21	Implement database entities & generate model	01-dec	2	01-dec	Brian
			Implement use cases incl. styling	01-dec	57	04-dec	Maren
			Funktions-test	03-dec	20	04-dec	Nicklas & Brian
			Møde med Vejledere, Karsten og Anette kl 14	04-dec	12	04-dec	Alle
			Implement changes	05-dec	12	05-dec	Anders
			Revise requirements	05-dec	2	05-dec	Alle
			Revise risk list	05-dec	2	05-dec	Alle
			Review & retrospect	05-dec	4	05-dec	Alle
		133	Sprintplan Construction it 4 & Sprint goals	05-dec	4	05-dec	Alle
					133		

Sprint goals: Ingen ud over generelle sprint goals

5.7 Construction sprint 3, Projects & labels

5.7.1 Revise requirements

Vi har ikke identificeret nye requirements i denne iteration.

5.7.2 Project management

5.7.2.1 Review & retrospect

5. december 2014 – deltagere: Anders, Brian, Nicklas og Maren

Alle artefakter er gennemgået og godkendt.

Vi kan mærke at vi får mere erfaring med programmeringssprog og framework, samt Doctrine.

Implementeringen begynder derfor at gå hurtigere og risks kan nedjusteres. Dog synes vi at frameworkt og doctrine har nogle begrænsninger og generelt bliver vores kode ikke af lige så høj kvalitet, som hidtil når vi har arbejdet med Java. Vi savner at programmere i et type-strækt sprog og vi synes at phtml er meget rodet i forhold til hvad vi er vant til. Vi forsøger dog at lave refakturering mv. for at koden bliver nænere og mere forståelig.

Dette sprint var det første sprint, hvor vi skulle lave et nyt modul i frameworkt. Vi fandt ud af at dette ikke kunne lade sig gøre (for os), da vi anvender Doctrine. Dette skyldes at entiteter som har en relation til andre entiteter (foreign key i databasen) skal ligge i samme modul og derfor skal model-klasserne ligge i flere moduler, hvilket ikke er en mulighed. Derfor har vi besluttet at vi kun laver et modul. Vores kunde ønsker flere moduler, men vi har valgt at dette ikke vil blive prioriteret. Vi vil evt. finde en løsning senere, hvis vi har tid. Vi har snakket med vores vejledere om dette og de synes det er den rigtige beslutning. Derfor bliver dette ikke tilføjet som en risk.

Som nævnt fungere det ikke for os at anvende Pivotal Tracker. Vi har snakket meget om det i gruppen og anvender også et fysisk SCRUM-board. Den eneste grund til at vi stadig anvender PT er at vores kunde ønsker det og følger backlog på denne måde, men vi associerer ikke stories og commits. Vi anser os ikke som ansatte hos Supeo, men som et selvstændigt organ, som leverer et produkt. Produktet skal naturligvis leve op til kundens krav, men vi mener ikke det er nødvendigt for Supeo at følge processen så tæt som PT muliggør og vi mener heller ikke at det er op til Supeo hvordan vi i vores gruppe vælger at styre processen (netop SCRUM lægger jo op til at man arbejder i små selvorganiserende teams), så længe vi når milestones. Vi mener der kan være tvivl om kundens rolle (arbejdsgiver/kunde). Dette må klargøres på næste planlagte møde d. 9. dec. og der skal naturligvis findes en løsning, således at product owner stadig har adgang til- og kan prioritere backlog. Vi må i dialog med Supeo og finde ud af hvordan Pivotal Tracker skaber værdi for dem.

5.7.2.2 Revised risk list

Impact H (high), M (medium), L (low) Magnitude: 1-10 (10 højest)

Magnitude	Description	Impact	Indicator	Mitigation strategy
7	Manglende erfaring med Pivotal Tracker's API og brug af CRON-jobs og Rabbit MQ	H	Synkronisering af database, Pivotal Tracker og klienter	Vi vil spørge Supeo til råds ang. Rabbit MQ og CRON-jobs, samt lave mindre forsøg med import fra PT.
5	Pivotal Tracker er ikke et godt redskab for vores gruppe	M	Manglende kommunikation med kunde og evt. uenighed om rollefordeling	Vi benytter Pivotal Tracker uden commits og benytter også fysisk SCRUM-board. Vi må på næste møde med Supeo d. 9. dec finde en løsning.

3	Nye teknologier	L	Projektplan/tid og kodekvalitet	Vi fortsætter med at sætte god tid af til implementering i hvert sprint
2	Manglende dokumentation	L	Projektplan/tid og kodekvalitet	Vi fortsætter med at sætte god tid af til implementering i hvert sprint

5.7.2.3 Sprint plan & goals, construction sprint 4 (Tasks & import)

Scrum master: Maren

Use cases: "Create task", "Edit task", "Delete task", "Search task", "Add task owner", "Remove task owner", samt import from Pivotal tracker

Uge	Dato	Mandetimer til rådighed	Opgave	Startdato	Varighedsestimat i mandetimer	Deadline	Hvem
50	08-dec	28	Domain model	08-dec	4	08-dec	Alle
	09-dec	28	System sequence diagrams/detailed use cases	08-dec	8	08-dec	Alle
	10-dec	28	Database diagram	08-dec	2	08-dec	Maren
	11-dec	28	Design class diagram (incl. klassekommentering)	08-dec	6	08-dec	Brian & Nicklas, Anders
	12-dec	21	Implement database entities & generate model	08-dec	1	08-dec	Maren
			Implement use cases incl. styling	08-dec	30	10-nov	Maren og Brian
			Møde med Supeo kl 10 incl. userability tests	09-dec	12	09-nov	Alle
			Import from Pivotal Tracker	09-dec	30	10-nov	Nicklas og Anders
			Funktionstest	11-dec	16	11-dec	Alle
			Implement changes	12-dec	10	12-dec	Maren
			Revise requirements	12-dec	4	12-dec	Alle
			Revise risk list	12-dec	2	12-dec	Alle
			Review & retrospect	12-dec	4	12-dec	Alle
			Sprintplan Construction it 5 & Sprint goals	12-dec	4	12-dec	Alle
		133			133		

Sprint goals (udover gennelige sprint goals):

- Userability testing/godkendelse af de use cases som er lavet i Construction sprint 2 og 3 (testperson fra Supeo)

5.8 Construction sprint 4, Tasks & Import

5.8.1 Revise requirements

System accounts: Vi havde af en eller anden grund forestillet os at hver virksomhed som anvender systemet, ville have en særskilt database. Dette er ikke tilfældet og derfor et nye use cases og roller identificeret. Alle relevante tabeller i databasen skal have en foreign key til en system account og alt hvad der oprettes skal tilføjes en account – dvs. hvad der før var unikke værdier, er nu kun unikt for brugerens system account. Dette gælder dog ikke workEmail, som bruges ved login og altid er unik. Ligeledes skal alle udtræk kun hente objekter fra brugerens account og rollen "system owner" skal tilføjes i databasen.

5.8.2 Project management

5.8.2.1 Review & retrospect

12. dec 2015 – deltagere: Maren, Brian, Anders og Nicklas

Alle artefakter er gennemgået og godkendt. Vi ville gerne have haft product owner til at teste og godkende use cases som omhandler sikkerhed, samt projects og labels, men vi må nok indrømme at vi ikke var helt godt nok forberedt til mødet. Vi arbejder på forskellige branches (git) og vi havde ikke fået merged alle branches korrekt ind i master lige før mødet og derfor var der ting som ikke fungerede i systemet. Dette

gjorde det sværere for product owner at danne sig et ordentligt billede af systemet og dets nuværende tilstand og vi må tage ved lære af dette til næste møde.

Vi kan nu importere fra Pivotal tracker og opdatere importerede elementer. Dog fjerner vi ikke elementer som er slettet i Pivotal Tracker, udover projectusers. Projectusers er medlemmer af projekter og vi synes det var vigtigt at fjerne disse, hvis de er fjernet i PT. Dermed han fluxusers stadig kun se projekter som de har tilladelse til at se. Vi kunne sagtens have valgt at fokusere meget mere på import, men vi synes vi har bevist at vi kan anvende PT's API til import og derfor vil vi nu gå videre med udvikling af systemets øvrige dele og lade Supeo optimere import efter overdragelse af systemet, hvis dette ønskes.

Vi har aftalt med Supeo at vi fortsætter med at bruge Pivotal Tracker til projektstyring, da dette er et stort ønske. Vi fik snakket om det på vores møde og til trods for at vi nok havde lidt svært ved at forstå hinanden, lærte vi hvordan værktøjerne skaber værdi for Supeo. PT benyttes som en form for systemdokumentation, da der ikke anvendes UML i så høj grad. Det giver lidt dobbeltarbejde for os, da vi allerede dokumenterer systemet vha. UML diagrammer mv., men vi vil gerne prøve at gøre en indsats vedr. brug af Pivotal Tracker da dette forekommer meget vigtigt for vores kunde.

Vi har identificeret nye krav som kræver en mindre gennemgang af hele systemet og derfor er klargøring af import fra andre systemet nedprioriteret. Vi regner ikke med at når dette eller export af data til economic (lavest prioritet på backlog).

Det ville have været nemmest at implementere System accounts i den kommende iteration (jo senere det implementeres, jo flere ændringer gennem systemet), men da time registration og reports har højere prioritet, laves dette først.

5.8.2.2 Revised risk list

Impact H (high), M (medium), L (low) Magnitude: 1-10 (10 højest)

Magnitude	Description	Impact	Indicator	Mitigation strategy
6	Manglende erfaring med generering af csv-filer i ZF2	M	Generering af rapporter	Vi laver lidt research på dette område når sprintet startes.
5	Kommunikation med kunden	M	Systemet's match af kundens krav	Vi bruger meget forskellige udviklingsværktøjer og arbejdsgange og har derfor lidt svært ved at forstå hinanden. Derudover har Supeo travlt og det er derfor forståeligt at dette projekt ikke har top prioritet, men vi føler at vi bliver hjulper godt når vi henvender os.
5	Pivotal Tracker er ikke et godt redskab for vores gruppe	M	Samarbejde (også med kunde)	Vi forsøger at organisere projektet og stories bedre i PT.

Vi har fjernet risks ang. nye teknologier, da vi nu synes vi har fået en del erfaring med værktøjerne og programmering i ZF2. Vi har dog tilføjet 2 risks da vi synes vores kommunikation med Supeo godt kunne være bedre og ikke kun foregå gennem Pivotal Tracker. Vi har også tilføjet en risk ang. CSV-filer, da vi ikke har erfaring med dette.

5.8.2.3 Sprint plan & goals, construction sprint 5 (time registration og reports)

Scrum master: Anders

Use cases: "Start timeregistration", "Stop timeregistration", "Finish task", "Search timeregistration", "Add timeregistration", "Edit timeregistration", "Delete timeregistration", "Generate project report" og "Generate user report"

Uge	Dato	Mandetimer til rådighed	Opgave	Startdato	Varighedsestimat i mandetimer	Deadline	Hvem
51	15-dec	28	Research csv	15-dec	8	15-dec	Brian
	16-dec	28	Domain model	15-dec	8	15-dec	Alle
	17-dec	28	System sequence diagrams/detailed use cases	15-dec	12	15-dec	Alle
	18-dec	28	Database diagram	15-dec	2	15-dec	Anders
	19-dec	21	Design class diagram (incl. klassekommentering)	15-dec	4	16-dec	Maren
			Implement database entities & generate model	15-dec	1	16-dec	Maren
			Implement use cases incl. styling	15-dec	54	18-dec	Nicklas og Brian
			Funktionstest	18-dec	20	19-dec	Maren og Anders
			Implement changes	18-dec	10	19-dec	Maren og Anders
			Revise requirements	19-dec	4	19-dec	Alle
			Revise risk list	19-dec	2	19-dec	Alle
			Review & retrospect	19-dec	4	19-dec	Alle
			Sprintplan Construction it 6 & Sprint goals	19-dec	4	19-dec	Alle
		133			133		

Sprint goals: Ingen uover generelle sprint goals.

5.9 Construction sprint 5, Time registration & Excel reports

5.9.1 Revise requirements

Vi ville i dette sprint gerne benytte Pivotal Tracker's API og starte stories, når der første gang registreres tid på en task. Ligeledes ville vi gerne give status "finished" i PT, når en task får denne status i systemet. Derfor skal en FluxUser have en API token for at have tilladelse til at skrive til PT. Derfor måtte 2 yderligere use cases implementeres "Add Pivotal Tracker API token" og "Remove Pivotal Tracker API token".

5.9.2 Project management

5.9.2.1 Review & retrospect

Dato: 19. Dec 2014 – deltagere: Nicklas, Anders, Brian og Maren

Alle artefakter er gennemgået og godkendt. Use cases er funktionstestet og mindre rettelser er tilføjet i koden. Use cases "Add Pivotal Tracker API token" og "Remove Pivotal Tracker API token" er også implementeret.

Vi har fået organiseret stories bedre I Pivotal Tracker og det går bedre med at anvende værktøjet til styring af projektet.

5.9.2.2 Revised risk list

Impact H (high), M (medium), L (low) Magnitude: 1-10 (10 højest)

Magnitude	Description	Impact	Indicator	Mitigation strategy
5	Kommunikation med kunden	M	Systemet's match af kundens krav	Vi bruger meget forskellige udviklingsværktøjer og arbejdsgange og har derfor lidt svært ved at forstå hinanden. Derudover har Supeo travlt og det er derfor forståeligt at dette projekt ikke har top prioritet, men vi føler at vi bliver hjulper godt når vi henvender os.

2	Pivotal Tracker er ikke et godt redskab for vores gruppe	L	Samarbejde (også med kunde)	Vi forsøger at organisere projektet og stories bedre i PT.
---	--	---	-----------------------------	--

Vi har fjernet risks ang. CSV-filer, da reports er implementeret i systemet.

5.9.2.3 Sprint plan & goals, construction sprint 6 (System accounts)

Scrum master: Maren

Use cases: "Create account", "Edit account", "Delete account", "Search account", samt gennemgang og opdatering af systemet

Uge	Dato	Mandetimer til rådighed	Opgave	Startdato	Varighedsestimat i mandetimer	Deadline	Hvem
52	22-dec	28	Domain model	22-dec	4	22-dec	Alle
	23-dec	28	System sequence diagrams/detailed use cases	22-dec	4	22-dec	Alle
			Database diagram	22-dec	1	22-dec	Brian
			Design class diagram (incl. klassekommentering)	22-dec	2	22-dec	Anders
			Implement database entities & generate model	22-dec	1	22-dec	Brian
			Implement use cases incl. styling	22-dec	16	23-dec	Maren og Brian
			Opdatering af tidligere use cases	22-dec	10	23-dec	Nicklas og Anders
			Funktionstest	23-dec	8	23-dec	Alle
			Implement changes	23-dec	2	23-dec	Nicklas og Anders
			Revise requirements	23-dec	2	23-dec	Alle
			Revise risk list	23-dec	2	23-dec	Alle
			Review & retrospect	23-dec	2	23-dec	Alle
			Sprintplan Construction it 7 & Sprint goals	23-dec	2	23-dec	Alle
		56			56		

Denne iteration er meget kort (2 dage). Vi kan evt. ikke nå at blive helt færdige, men det vi ikke når, må laves over julen før den 5. jan 2015 hvor vi mødes igen og starter et nyt sprint.

5.10 Construction sprint 6, System accounts

5.10.1 Revise requirements

Vi fandt ud af at alle id'er i url skal krypteres, så brugere ikke kan editere objekter tilhørende en anden account. Dette, samt yderligere debugging forgår over julen.

5.10.2 Project management

5.10.2.1 Review & retrospect

Vi har gennemgået og godkendt alle artefakter. Vi havde meget kort tid til dette sprint og derfor vil vi gerne bruge mere tid på debugging og har også fundet ud af at url skal krypteres. Derfor laves disse opgaver over julen (deadline 4. jan 2015):

Kryptering af url: Maren

Test af alle use cases på forskellige konti: Nicklas, Brian og Anders

5.10.2.2 Revised risk list

Impact H (high), M (medium), L (low) Magnitude: 1-10 (10 højest)

Magnitude	Description	Impact	Indicator	Mitigation strategy
5	Kommunikation med kunden	M	Systemet's match af kundens krav	Vi bruger meget forskellige udviklingsværktøjer og arbejdsgange og har derfor lidt svært ved at forstå hinanden. Derudover har Supeo travlt og det er derfor forståeligt at dette projekt ikke har top prioritet,

				men vi føler at vi bliver hjulper godt når vi henvender os.
5	Sen userbilitytest	M	Systemet's match af kundens krav	Vi har ikke kunne afholde møde med Supeo før jul og derfor skal mange use cases godkendes i en omgang

Vi har fjernet risks ang. brug af Pivotal Tracker da vi synes at det går bedre med at anvende PT.

5.10.2.3 Sprint plan & goals, construction sprint 7 (Clients & contacts)

Scrum master: Brian

Use cases: "Create client", "Edit client", "Delete client", "Search client", "Add contact", "Edit contact", "Delete contact", "Add contact to project", "Remove contact from project"

Uge	Dato	Mandetimer til rådighed	Opgave	Startdato	Varighedsestimat i mandetimer	Deadline	Hvem
2	05-jan	28	Domain model	05-jan	8	05-jan	Alle
	06-jan	28	System sequence diagrams/detailed use cases	05-jan	8	05-jan	Alle
	07-jan	28	Database diagram	05-jan	2	05-jan	Nicklas
	08-jan	28	Design class diagram (incl. klassekommentering)	05-jan	6	05-jan	Brian
	09-jan	28	Implement database entities & generate model	05-jan	1	05-jan	Nicklas
			Implement use cases incl. styling	05-jan	40	08-jan	Maren og Brian
			Funktionstest	07-jan	15	08-jan	Alle
			Implement changes	07-jan	8	08-jan	Alle
			Revise requirements	08-jan	10	08-jan	Alle
			Revise risk list	08-jan	10	08-jan	Alle
			Review & retrospect	08-jan	10	08-jan	Alle
			Sprintplan Construction it 8 & Sprint goals	08-jan	10	08-jan	Alle
			Møde med Supeo kl 10 incl. userbility test	09-jan	12	09-jan	Alle
		140			140		

Sprint goals (udover generelle sprint goals):

- Userbility test (test person: product owner): Test/godkendelse af use cases vedr. Tasks og import, time registration og reports, system accounts, samt client og contacts.
 - Vi er egentlig ret kede af at vores product owner skal teste og godkende så mange dele af systemet på én gang. Vi kunne desværre ikke få et møde før jul og derfor skal vi først mødes hos Supeo d. 9. januar 2015. Dette kan betyde at vi skal lave mange ændringer sent i processen. Heldigvis har vi holdt os meget til vores prototypes som er testet på product owner.

5.11 Construction sprint 7, Clients & contacts

5.11.1 Revise requirements

Ingen nye requirements er identificeret.

5.11.2 Project management

5.11.2.1 Review & retrospect

Vi har gennemgået og godkendt alle artefakter, samt testet use cases vha. funktionstest. Product owner har godkendt alle use cases som er implementeret siden sidste møde.

Vi har afholdt møde med Supeo (se referat afsnit 10.2.4). Det var et rigtig godt møde og product owner virkede meget tilfreds med systemet og havde ikke nogen ændringer til systemet, men et par ønsker til bl.a. paging på tabeller og mulighed for at tilføje og fjerne Pivotal Tracker api token under Edit profile.

5.11.2.2 Revised risk list

Vi har fjernet alle risks fra risk listen. I næste iteration vil vi bl.a. lave "Google login". Dette har vi ingen erfaring med, men da det ikke er et krav fra Supeo, synes vi ikke dette er en risk for projektet.

5.11.2.3 Sprint plan & goals, construction sprint 8 (Google login & Change language)

Scrum master: Nicklas

Use cases: "Google login" og "Change language"

Uge	Dato	Mandetimer til rådighed	Opgave	Startdato	Varighedsestimat i mandetimer	Deadline	Hvem
3	12-jan	28	Database diagram	12-jan	1	12-jan	Anders
	13-jan	28	Design class diagram (incl. klassekommunikering)	12-jan	2	12-jan	Maren
			Implement database entities & generate model	12-jan	1	12-jan	Anders
			Implement use cases incl. styling	12-jan	34	13-jan	Anders og Nicklas
			Funktionstest	13-jan	8	13-jan	Alle
			Implement changes	13-jan	2	13-jan	Brian og Maren
			Revise requirements	13-jan	2	13-jan	Alle
			Revise risk list	13-jan	2	13-jan	Alle
			Review & retrospect	13-jan	2	13-jan	Alle
			Sprintplan Transition	13-jan	2	13-jan	Alle
		56			56		

Ingen sprint goals uover generelle sprint goals.

5.12 Construction sprint 8, Google login & Change language

5.12.1 Revise requirements

Ingen nye krav er identificeret i dette sprint.

5.12.2 Project management

5.12.2.1 Review & retrospect

Vi har implementeret "Google login" og "Change language". Alle artefakter er gennemgået og godkendt. Alle use cases er nu implementeret og vi er færdige med construction-fasen.

5.12.2.2 Revised risk list

Ingen risks på risk list.

5.12.2.3 Sprint plan & goals, Transition

Scrum master: Anders

Sprint goals:

- Små tilføjelser til programmet: Paging på tabeller og add/remove api token under profile
- Complete debugging
- Brugervejledning og andet self supportability

5.12.2.4 Sprint plan & goals, Transition

Uge 3 (og 4): Individuel eksamsforberedelse

Uge 4 og 5:

Implementering: Paging, add/remove api token under profile, complete debugging

Brugervejledning og andet self supportability

Training af superuser

Product release (systemet sættes i beta test men er nu er nu overdraget til Supeo)

Eksamens d. 29.-30. jan 2015

6 Problemløsning

6.1 Release planning

I denne fase etableres projektet og planlægges overordnet.

6.1.1 Vision

Projektteamet har modtaget et oplæg fra vores kunde (se afsnit 10.1), hvori kunden beskriver ønsker og krav til systemet. Ud fra denne præsentation har projektteamet skrevet denne vision. Visionen er kundens billede på det system som ønskes udviklet. Visionen hjælper projektteamet i udviklingen og fungerer ligeledes som en uformel aftale mellem projektteamet og kunden²⁷.

Supeo ønsker et stand-alone system til tidsregistrering af arbejdsopgaver. Systemet skal ligeledes kunne fungere som tillægsmodul til Pivotal Tracker eller andre opgavesystemer. Systemet skal kunne oprette eller importere projekter, epics og opgaver fra opgavesystemer og være synkroniseret hermed. Medarbejderne skal kunne registrere når de starter på en opgave. Efterhånden som dagen eller dagene går, skal medarbejderne kunne pause, starte og til sidst stoppe tidsregistreringen helt.

Som administrator skal man kunne trække diverse rapporter på de enkelte medarbejdere og se deres tidsregistrering. Rapporterne skal bygges op som pivot rapporter. Endvidere skal man kunne fortage diverse søgninger og visninger.

Systemet skal kunne kommunikere med Economic som er et faktureringssystem.

Systemet skal kun kunne benyttes af oprettede brugere og skal være beskyttet af login. Endvidere skal systemet skal laves som en web-applikation som benytter Apache web-server og PostgreSQL, samt HTML5, CSS3 og Javascript.

6.1.2 Projektorganisation & ressourcer

Nedenstående er en liste over ressourcer, som vi har brugt og vil bruge til planlægning²⁸.

6.1.2.1 Team & roller

Alle i projektteamet har stærke kompetencer inden for udviklingsmetode, herunder UP, OOA/D og RDD, samt projektstyringsmetoden SCRUM.

Nedenstående er en liste over programmeringssprog og udviklingsværktøjer, som teamet har erfaring med:

Nicklas Kolls Larsen: Java & netbeans, SQL (MySql), Subversion (versionsstyring), Visual paradigm, PHP, Javascript, CSS, HTML5, WordPress.org

Brian Thorning Jørgensen: Java & netbeans, SQL (MySql, PostgreSQL), Javascript, CCS, HTML5, Subversion & Pivotal Tracker, Git (versionsstyring), Visual paradigm (diagrammer), Jquery, Ajax, PHP, JSON

²⁷ Larman s. 58

²⁸ Scrum s. 9

Ander Bo Rasmussen: Java & netbeans, SQL (MySql, PostgreSQL), Javascript, CCS, HTML5, Subversion & Pivotal Tracker, Git (versionsstyring), Visual paradigm (diagrammer), Ajax, PHP, JSON, Ruby on Rails

Maren S. Håkansson: Java & netbeans, SQL (Sql server & MySql), C# (Visual studio), Asp.net (webforms & DevExpress), Javascript, CCS & less, HTML5, Trello & Subversion (versionsstyring), Visual paradigm, Visio og Blend (diagrammer)

Troels: Kontaktperson hos Supeo, Product owner

Vurdering: Vi har de fornødne ressourcer i projektteamet. Dog mangler vi erfaring med mulige frameworks og udviklingsredskaber og skal derfor lave research på dette område.

6.1.2.2 Tekniske ressourcer

Hardware: Laptops, Printer, Scanner, Kamera, Server

Software: Netbeans(ZF2), Composer, Xampp, Git/Github (fildeling og versionsstyring), MS Office, Visual Paradigm (diagrammer), Skype (kommunikation), PostgreSQL, Supeo bruger-profiler

Vi har ikke Economic til rådighed, som vores system skal kommunikerer med. Til dette kan vi evt. anvende en trial version eller ændre krav til systemet.

6.1.2.3 Tidsressourcer

1 dag = 28 timer (4 gruppemedlemmer á 7 timer)

Uge 43-44 = 105 timer (3 gruppemedlemmer) / pr uge

Uge 45 – 51: 133 timer / uge

Uge 52: 56 timer (2 dage, rest ferie)

Uge 1: Ferie

Uge 2: 133 timer/uge

Uge 3: 56 timer (2 dage)

Vurdering: Det vurderes at der er tid nok til projektet, men som altid skal backlog prioriteres, da vi kan komme til at mangle tid, såfremt kommunikation med andre systemet viser sig at være meget tidskrævende eller nye krav identificeres.

6.2 Inception

6.2.1 Research af teknologier

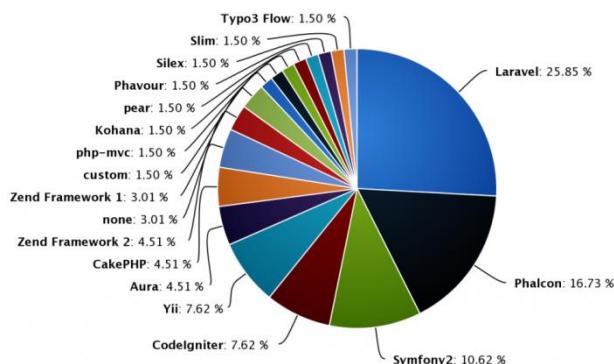
6.2.1.1 Applications-type

For at beslutte hvilken applications-type vil skal anvende, er vi nødt til at se på hvilke fordele der er ved at anvende et framework, et CMS-system og et custom CMS-system.

Et CMS-system er typisk et færdigt system med en GUI til at oprette og vedligeholde web-sider og indhold. Det er typisk meget hurtigere at lave et system vha. et CMS-system end et Framework, men frameworkt er meget mere egnet til at lave speciallavede applicationer. Et framework er at foretrække i forhold til skræddersyede løsninger, da de bygger på ordssproget "Don't reinvent the wheel". Frameworket er også bedre egnet til teamwork og unit-testing. Derfor bliver frameworks tit foretrukket til udviklingsarbejde. Typisk hjælper frameworket også med andre ting som asset management, sikkerhed og utilities, som man ellers selv ville implementere helt fra bunden.

Det rette valg må være, at til større og mere specialsyede løsninger, hvor man har brug for mere end bare lige det mest basale, så bruger man et framework. Det er velegnet til teamwork, hvor hver udvikler kan arbejde med hver del samtidigt.

Vi har fundet 3 framework-kandidater: Zend Framework 2, Laravel og Phalcon. Der findes andre kandidater, men vi har valgt disse, da vi mener at det er de mest aktuelle kandidater.



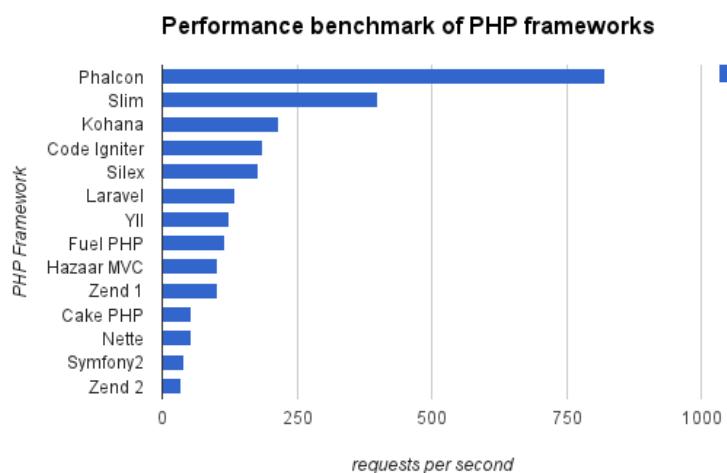
Figur 1²⁹

De fleste frameworks ligner hinanden på mange områder. De er alle cross-platform, har multiuser systemer, de er extendable (plugin-system), har flersprogs mulighed, kan sende request med unicode etc. Så hvor er så forskellen?

Supeo bruger i øjeblikket Zend Framework 1, så derfor har de selvfølgelig en interesse i at beholde systemet i et velkendt framework. Vi har dog valgt at undersøge flere frameworks, for at finde den bedste løsning.

Der er kommet to meget vigtige med spillere på markedet. Laravel og Phalcon. Larevel bliver rost overalt for deres simple API, som skulle være utrolig nem at forstå og deres framework skulle kunne rigtig mange ting. Phalcon skulle være det absolut hurtigste framework, hvis man mäter på opstart og på at sende flest request.

²⁹ <http://www.sitepoint.com/best-php-frameworks-2014/>



Figur 2³⁰

Licensmæssigt har Laravel en MIT og de to andre BSD hvilket ikke gør den store forskel, men kan gøre en forskel senere i forhold til patenteret. De er alle open source.

Vi skal bruge PostgreSQL til database og dette understøttes af alle 3 kandidater.

Laravel	ZF2	Phalcon
Object-oriented	Relational	Relational
	NoSql	NoSql
	XML database	Object-oriented
		Document-oriented

Da vi er interesserede i at benytte MVC er Zend Framework 2 rigtig godt da dens struktur er bygge op efter designmønstret.

Man taler ofte om hvor lang tid det tager at sætte sig ind i en API og hvor god dokumentationen er. Her vinder Laravel, da deres API er meget vel beskrevet og de har en yderst simpel API. Zend Framework er kendt for at have en meget stejl læringskurve og er ikke så dokumenteret

Vi har talt om at vi gerne vil have et framework der har Jquery, CSS3, HTML5 og kan bruge Bootstrap 3 (.css), font awesome og tablesorter. Vi skal i øvrigt kunne sende emails. Alt dette kan man i alle 3 frameworks.

Oversigt over forskelle³¹:

	Laravel	ZF2	Phalcon
Email protocols	SMTP	SMTP	SMTP
	IMAP	IMAP	IMAP
	POP3	POP3	POP3
	Mailgun		

³⁰ <http://systemsarchitect.net/performance-benchmark-of-popular-php-frameworks/>

³¹ <http://vschart.com/compare/laravel/vs/zend-framework>

	Mandrill		
Web flows	Yes	No	Yes
Creation date	2011	September 2012	January 2012
Data encryption	Yes	Yes	Yes
Cloud platform support	Google App Engine	Windows Azure	Amazon EC2
	Amazon EC2	Amazon EC2	Windows Azure
	Heroku	Heroku	Rackspace Cloud
	Fortrabbit		Fortrabbit
	Pagoda box		Digital ocean
	Digital ocean		Linode
	Linode		
	Open Shift		
Complier	Yes	No	Yes
Library file size	17 MB	8 MB	3 MB
API comprehensibility	★★★★★	★★★★☆	★★★★★
Realtime	Yes	Yes	Yes

6.2.1.2 Valg

Ved valget af framework ligner Larevel umiddelbart et godt valg fordi det har en let læringskurve, er godt dokumenteret og det skulle være rigtig nemt at bruge og forstå API'en. De forskellige frameworks er meget ens på de punkter som er vigtige for vores projekt, men da vi ikke mener at hastighed er så vigtigt for dette system, vil vi ikke vælge Phalcon^{32 33}.

Vi vil foreslå at man bruger Zend Framework 2. Dette vil være sværere for os at lære, men Supeo har stor erfaring med brugen af dette, så det vil ikke tage lang tid for medarbejderne hvis de skal udvikle videre på produktet. Der er lige så godt som Laravel til det vi skal anvende det til, men langsommere en Phalcon. Vi har dog læst at Zend er hurtigere end beskrevet, når det kører på en produktion environment server.

Vi har lavet nogle guides til installation af udviklingsværktøjer (se afsnit 10.4). Disse kan anvendes af vores kunde til videreudvikling.

6.2.1.3 PHP-UNIT Testing

PHPUnit er en del af det populære xUnit testing framework. xUnit bruges, som navnet også antyder, til unit tests. xUnit startede ud med Sunit (for programmeringssproget SmallTalk), men blev først rigtig kendt med jUnit (for programmeringssproget Java)³⁴.

Vi kunne have valgt nogle andre test-frameworks til at teste vores kode med. F.eks. PHPSpec, som er et Behaviour-driven-development framework³⁵, som er en nyfortolkning af unit test og acceptance test. Det handler om at man navngiver alle sine test med fulde sætninger, og at de alle sammen bør begynde med 'should'. Acceptance test bør skrives med det standard framework for en user story: "As a [role] I want [feature] so that [benefit]". Acceptance kriterierne bør beskrives som scenarier og implementeres som klasser: "Given [initial context], when [event occurs], then [ensure some outcomes]"³⁶.

³² <http://codegeekz.com/20-best-php-frameworks-developers-august-2014/>

³³ <http://systemsarchitect.net/performance-benchmark-of-popular-php-frameworks/>

³⁴ <http://en.wikipedia.org/wiki/XUnit>

³⁵ http://en.wikipedia.org/wiki/Behavior-driven_development

³⁶ <http://en.wikipedia.org/wiki/PHPUnit>

Grunden til at vi valgte PHPUnit som vores test framework, er at denne understøttes af Netbeans, vores IDE (Integrated Developer Environment), og at det er det samme test framework som der bliver brugt i Zend Framework 2's guide på deres hjemmeside, og at det, som nævnt tidligere, minder om Junit, som vi har erfaring med³⁷.

6.2.1.4 Pivotal API

Pivotal Tracker (PT) er et agilt projektstyringsværktøj, der letter planlægning, ledelse og styring af projekter. Man kan bruge API'en til at oprette, hente, opdaterer og slette næsten alt i PT. Vi skal dog kun bruge en lille del af api'ens store udvalg. For at kunne kommunikerer med Pivotal Trackers API bruges HTTP requests. Hver bruger har et API token (id), som man kan logge på f.eks. et projekt og hente de ønskede informationer. Man kan vælge 2 forskellige løsninger på kommunikationen mellem PT og webapplicationen enten CURL eller CORS - xtracker som begge giver et JSON-response.

CORS er indbygget i Javascript og kan bruges uden at man installerer andet. Hvor CURL derimod man skal installeres på serveren. Vi har valgt at bruge CURL da den er fuldt implementeret i APIn og CORS ikke kan benytte med CRON-job.

6.2.1.5 Synkronisering med Pivotal Tracker

Vores kunde har et ønske om at hver klient, og ikke kun databasen, skal synkroniseres med Pivotal Tracker løbende. RabbitMQ er en kommunikationserver, der har et centralt placeret

```
function executeTrackerApiFetch() {
    // get parameters
    var token = $('#token').val();
    projectId = $('#project_id').val();

    // compose request URL
    var url = 'https://www.pivotaltracker.com/services/v5';
    url += '/projects/' + projectId;
    url += '/stories?filter=state:delivered,finished,rejected,start';
    url += ',unstarted,unscheduled';
    url += '&limit=20';

    // do API request to get story names
    $.ajax({
        url: url,
        beforeSend: function(xhr) {
            xhr.setRequestHeader('X-TrackerToken', token);
        }
    }).done(displayTrackerApiResponse);
}
```

kø kørende. Serveren sørger for at holde styr på kø'er der sendes mellem klienter(FluxTime) og serveren. Man sender kommunikation igennem HTTP Request (små pakker). På serveren kan man køre Cron-jobs (Scheduled tasks) for at synkronisere med Pivotal Tracker. Disse cron-jobs kan f.eks. gøre sådan at man kan få opdateret programmet hver minut. Dette kan ikke gøre oftere på denne måde, men det er sådan at vi vil få programmerne synkroniseret. Man ville også kunne en while-løkke i stedet til at opdatere hver klient, men dette vil tage for mange systemressourcer. Alternativt man vi undersøge om vi evt. kan anvende en timer eller lignende som alternativt til Rabbit MQ for at opdatere klientens GUI med databasen.

6.2.1.1 Versionstyring

Vi har brug for et repository at hoste vores application. Man kan vælge mange forskellige. Bitbucket, Subversion, GIT (github). Først valgte vi Bitbucket da man kunne være flere end 3 om et gratis private repository. Men vi skrev til Github og spurte om vi ikke kunne være med i deres student program. Vi fik lov til at bruge 5 private repositories i 2 år. Dette ville hellere da vi kender github i forvejen. Det er nemmere at finde ud af, da det har et mere brugervenligt flow. Zend Framework 2 bliver også hostet hos github. Derudover har vi fået repositories af Supeo.

³⁷ <https://phpunit.de/>

6.2.1.2 Server

Til at teste vores web application der kører Zend Framework 2 med PostgreSQL, har vi installeret en server på en computer og sørget for at man tilgår via FTP. Vi havde dog nogle problemer med dette, så Supeo har sørget for at vi har fået serverplads, database plads, github repositories og vil også oprette Supeo-mailkonti & brugerprofiler til os. Brugerprofilerne kan ligeledes benyttes til udviklingen og til test af systemet.

6.2.1.3 Doctrine

Doctrine er en object-relational mapper (ORM) som kan anvendes med PHP. Det er inspireret af Hibernate (Java) og gør det muligt at auto-generere objekter ud fra database-modellen og omvendt. Yderligere håndterer Doctrine kommunikationen mellem systemet og persistens-laget. Dvs. at storedprocedures og database-handlers ikke skal oprettes, men at Doctrine stiller database-kald til rådighed via mappers, som kan anvendes i PHP-control-laget.

6.2.2 Business analysis

For at få en bedre forståelse af vores kunde har vi lavet en mindre virksomhedsbeskrivelse og en swot-analyse. Dette kan hjælpe med at afdække hvor systemet kan lette kunden, samt hvilken fleksibilitet systemet skal have i forhold til videre-udvikling³⁸.

6.2.2.1 Virksomhedsbeskrivelse

Supeo er en mindre udviklingsvirksomhed i Næstved med 6-12 udviklere inkl. praktikanter. Supeo blev stiftet i 2007 og har udviklet og leveret systemer til jernbane industrien i alle 7 år. Deres system til dette er Sitra, som er et Enterprise Asset Management system (EAM). Sitra bliver leveret til firmaer som Regionstog, Midtjyske Jernbaner og Bravida.

6.2.2.2 SWOT

Styrker: Supeo har få, men meget store og veletablerede kunder, som det leverer store systemer til, samt vedligeholder. Dette er ikke kun en fast indtægt, men blåstempler også virksomheden i forhold til nye kunder.

Supeo er en af de eneste udviklingsvirksomheder i Næstved og har derfor nogle af de dygtigste nyuddannede udviklere fra nærområdet ansat. Endvidere tager Supeo mange praktikanter og får derved billig arbejdskraft.

Svagheder: Supeo har næsten udelukkende ansat nyuddannede medarbejdere fra samme uddannelsessted og derfor får virksomheden ikke bragt så meget ny erfaring ind i virksomheden. Ligeledes er flertallet af medarbejdere praktikanter, hvilket kan være en ulempe da erfarringsniveauet aldrig når at blive højt, fordi disse er ansat i en tidsbegrænset periode.

Trusler: En trussel for Supeo er at det kan være risikabelt for en virksomhed at have så få store kunder. En meget stor del af deres omkostning vil forsvinde, hvis de mister bare en af kunderne.

En anden trussel kan være konkurrenter. Enten i nærområdet, som kan tage kvalificeret arbejdskraft, eller i form af større virksomheder som kan overtage de store kunder, fordi de er mere stabile og veletablerede og dermed konkurrencedygtigt.

Supeo er en lille, men voksende virksomhed og derfor er virksomheden begyndt at få forbedret administrative værktøjer.

³⁸ SCRUM s. 9

Muligheder: Det ville være en mulighed for Supeo at ansætte medarbejdere fra andre virksomheder, for at få en bredere palet af kompetencer og erfaringer.

Ligeledes ville det være en mulighed for Supeo at udvide sin kundekreds, for at sikre en stabil indtægt fremover.

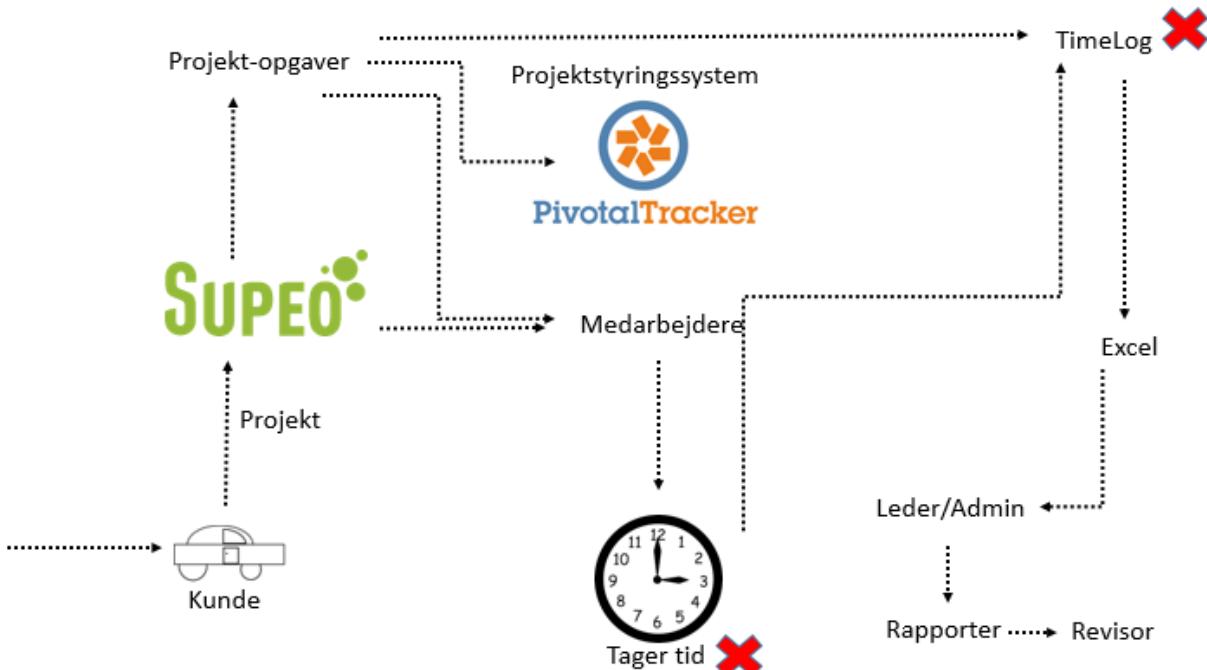
Det ville være tidsbesparende for Supeo's medarbejdere og administration at styrke deres administrative værktøjer.

6.2.2.3 Supeo's nuværende system

Supeo's medarbejdere anvender TimeLog til registrering af tidsforbrug. TimeLog er en stand alone applikation og derfor skal medarbejdernes opgaver, uover at oprettes i Pivotal Tracker, også angives i TimeLog. TimeLog præsenterer opgaverne på uoverskuelig vis og derudover skal medarbejderne selv holde øje med hvor meget tid de bruger på hver opgave og så skrive det ind i TimeLog. Dette er besværligt for hver medarbejder som skal regne tidsforbrug sammen løbende og også tidskrævende for ledelsen/administrationen som skal bruge unødig tid på fakturering og rapportering pga. den manglende struktur i TimeLog.

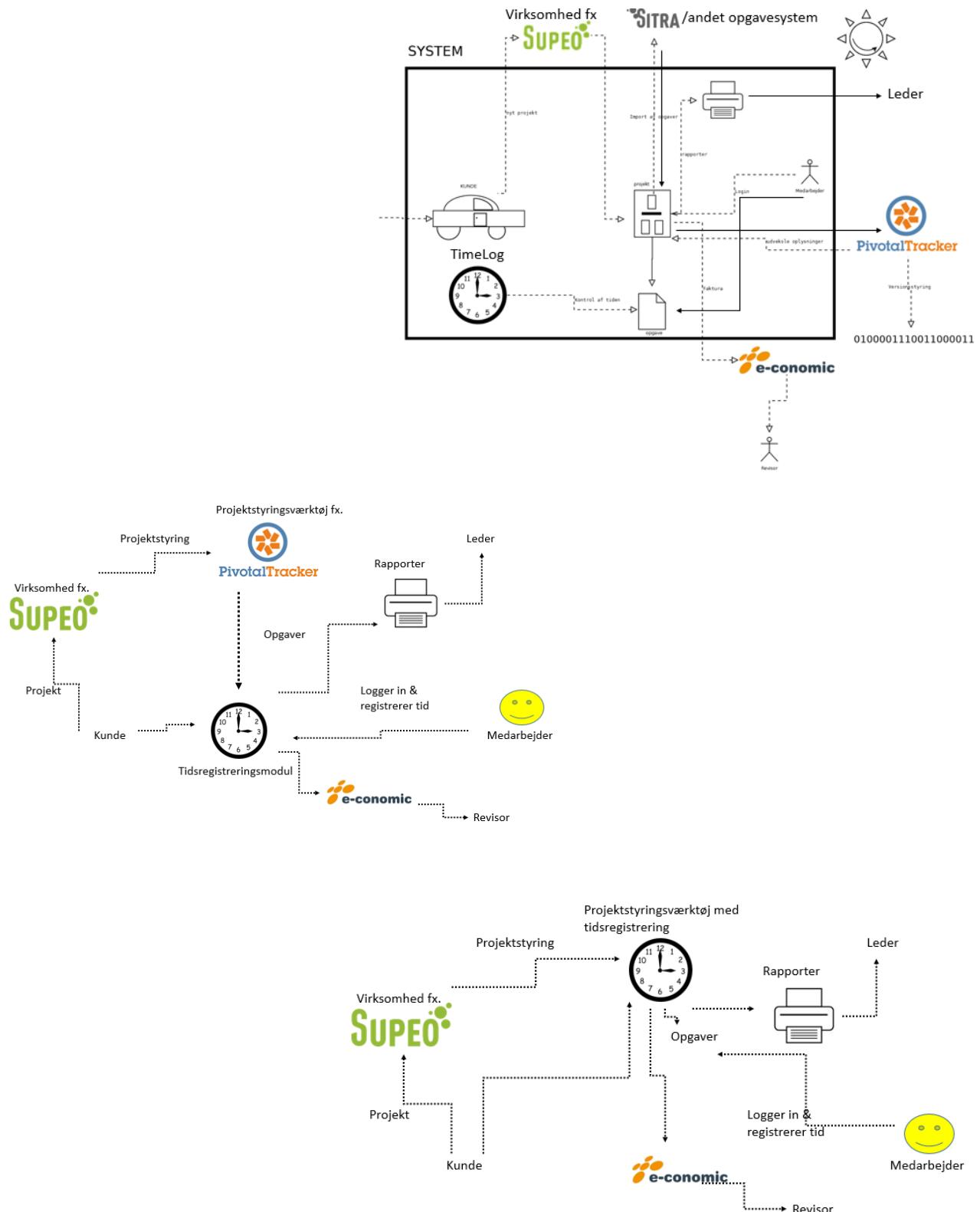
Sitra, som er Supeo's største produkt, håndterer også arbejdsopgaver. Disse opgaver tidsregistreres ikke på nuværende tidspunkt, men det er på sigt meningen at dette system også skal registrerer tidsforbrug på opgaver.

6.2.2.4 Hændelser (rigt billede)



6.2.3 Requirements

6.2.3.1 Rige billedeer (forslag til systemløsninger)



6.2.3.2 BATOFF

Betingelser: Relativt rutinerede brugere, som har erfaring med opgave- eller projektstyringssystemer.

Anvendelsesområdet: Aktørerne kan være alle brugere af opgavesystemer. Ledere eller administrative kan benytte en fakturerings- og rapporteringsfunktion.

Teknologi: Programmeret i et PHP-framework (ZF2) der inkluderer Javascript, CSS3, JSON, jQuery, HTML5 og evt. RabbitMQ server til kommunikation via API. Systemet kan kommunikere med Pivotal Tracker eller andet opgave-system og evt. E-economic. Systemet understøtter brug af flere platforme og kan anvendes cross browser. For at registrere og redigere data bruges PostgreSQL server.

Objekter: Medarbejder, Kunde, Rapport, Projekt, Epic(kategori), Opgave, Faktureringsrapport, Tidsregistreringer

Funktioner: Create, read, update, delete Medarbejder, Projekt, Epic (kategori), Opgave og Kunde, login, importer og synkroniser projekter, epics og opgaver (evt. fra Pivotal Tracker), eksporter fakturering, diverse søgninger, samt udskrift af rapporter.

Filosofi: Et tidsregistreringssystem som kan lette tidsregistrering for medarbejdere og skal afhjælpe ledelse med fakturering og resourcefordeling.³⁹

6.2.3.3 Systemdefinitioner

6.2.3.3.1 Systemdefinition (valgt af Supeo)

En web-applikation til medarbejder-tidsregistrering og styring af forbrug af medarbejdertimer til konkrete projekter og opgaver. Applikationen fungerer som administrativt værktøj og skal både kunne anvendes som stand-alone-applikation og som et tillægsmodul til opgavesystemer som f.eks. Pivotal Tracker. Man skal kunne oprette og importere projekter, epics og opgaver i systemet fra andre opgave-systemer. Det skal være i stand til at kommunikerer med E-economic, for at lette fakturering. Systemet skal være beskyttet med login og skal kunne håndtere bruger-informationer og kunde-informationer.

6.2.3.3.2 Alternativ systemdefinition 1

En web-applikation til medarbejder-tidsregistrering og styring af forbrug af medarbejdertimer til konkrete projekter og opgaver. Applikationen fungerer som administrativt værktøj og er et tillægsmodul til Pivotal Tracker. Man skal kunne importere projekter, epics og opgaver i systemet fra Pivotal Tracker. Det skal være i stand til at kommunikerer med E-economic, for at lette fakturering. Systemet skal evt. via en API, kunne kommunikerer med andre eksterne programmer som Pivotal Tracker, hvor man skal kunne importerer projekter og opgaver. Systemet skal være beskyttet med login og skal kunne håndtere bruger-informationer og kunde-informationer.

6.2.3.3.3 Alternativ systemdefinition 2

En stand-alone web-applikation som skal erstatte Pivotal Tracker: Versionsstyrings-client der benytter Git og tilfører tidsregistrering til opgaver og derved styre ressource-forbrug til projekter og opgaver. Ligeledes kan systemet generere rapporter og kommunikere med E-economic. Dette er både et projektstyrings- og versionsstyringssystem og et administrative værktøj.

³⁹ OOAD s. 38

6.2.3.4 Actors and use cases

6.2.3.4.1 Actors

Primary actors: User, Admin, Project manager

Off-stage actors: Accountant, Client/customer

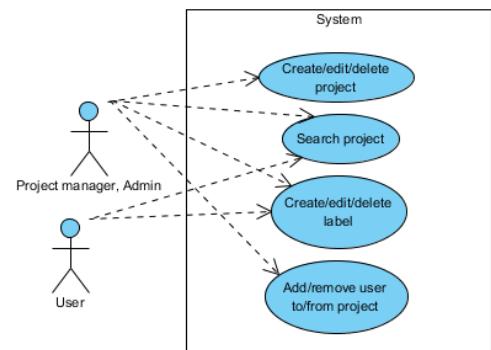
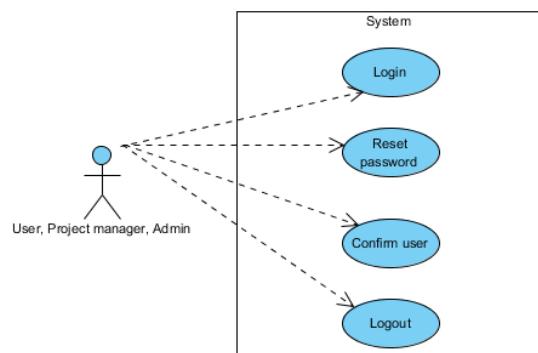
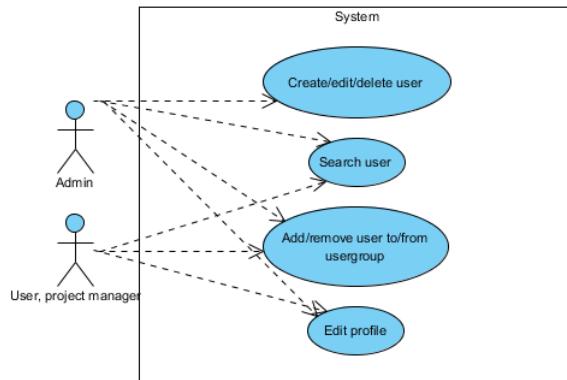
Supporting systems: Economic og Pivotal tracker (senere også andre opgave-systemer)

6.2.3.4.2 Backlog & Use case diagrammer

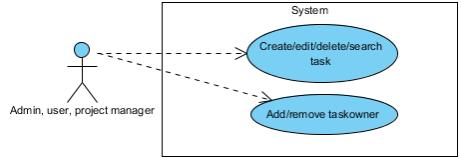
Backloggen⁴⁰ vil løbende udvikle sig under projektet.

For at illustrerer hvilker actors, som anvender hvilke use cases, har vi lavet disse use case diagrammer. De er delt op i flere diagrammer for overblikkets skyld.

- User
 - Create/edit/delete user: Admin opretter/redigerer/sletter user
 - Edit profile: User, Admin eller Project manager redigerer profil
 - Search user: Admin, User eller Project manager søger efter user
 - Usergroups (permissions)
 - Add/remove user to usergroup: Admin tilføjer/fjerner user til/fra usergroup
 - Login
 - Login: Admin, Project manager eller User logger ind
 - Confirm user: Admin, Project manager eller User skifter midlertidigt password
 - Reset password: Admin, Project manager eller User ændrer password
 - Logout: Admin, Project manager eller User logger ud
 - Project
 - Create/edit/delete project: Admin eller project manager opretter/redigerer/sletter projekt
 - Search project: Admin, User eller project manager søger efter projekt
 - Add/remove user to/from project: Admin eller project tilføjer/fjerner user til/fra projekt

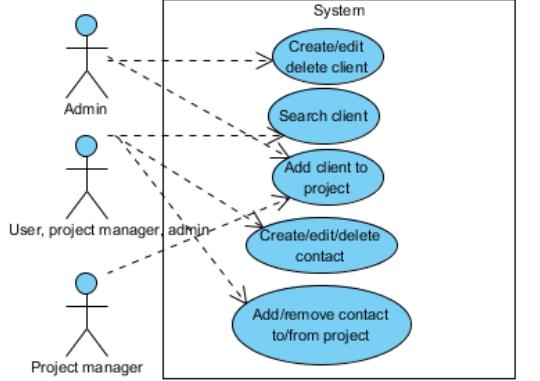


⁴⁰ Scrum s. 19

- Label
 - Create/edit/delete label: Admin, User eller project manager opretter/redigerer/sletter label
 - Task
 - Create/edit/delete/search task: Admin, Project manager eller User opretter/redigerer/sletter/søger efter task
 - Add/remove taskowner: Admin, Project manager eller User tilføjer/fjerner user til/fra task
 - Systemet/actor importerer projects, tasks og labels from PT/other system
 - Timeregistration
 - Start/stop timeregistration: Admin, Project manager eller User starter/stopper opgave-tidsregistrering
 - Finish task: Admin, Project manager eller User afslutter task
 - Add/edit/delete timeregistration: Admin opretter/redigerer/sletter timeregistration
 - Search timeregistration: Admin, Project manager eller User søger efter opgave-tidsregistrering
 - Report
 - Make user report: Admin genererer user-rapport (timeregistrations)
 - Make project report: Admin eller project manager genererer project-rapport (timeregistrations)
 - Client
 - Create/edit/delete client: Admin opretter/redigerer/sletter client
 - Make client report: Admin genererer client-rapport (timeregistrations)
 - Add client to project: Admin eller project manager tilføjer client til projekt
 - Search client: Admin, Project manager eller User søger efter client
 - Client contact
 - Create/edit/delete contact: Admin, project manager eller user opretter/redigerer/sletter contact
 - Add/remove contact to/from project: Admin, project manager eller user tilføjer/fjerner contact til projekt
- 
- ```

useCaseDiagram
 actor AdminUserPM as "Admin, user, project manager"
 actor Admin as "Admin"
 actor ProjectManager as "Project manager"
 system System

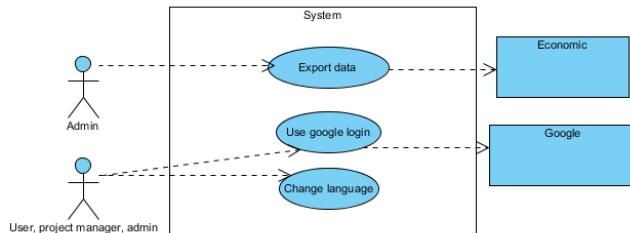
 AdminUserPM --> CreateEditDeleteSearchTask
 AdminUserPM --> AddRemoveTaskOwner
 Admin --> StartStopSearchTimereg
 Admin --> FinishTask
 Admin --> AddEditDeleteTimereg
 Admin --> MakeTimeregReport
 Admin --> ImportProjectsLabelsTasks
 ProjectManager --> StartStopSearchTimereg
 ProjectManager --> FinishTask
 ProjectManager --> AddEditDeleteTimereg
 ProjectManager --> MakeTimeregReport
 ProjectManager --> ImportProjectsLabelsTasks

```
- 
- ```

useCaseDiagram
    actor AdminUserPM as "Admin, user, project manager"
    actor Admin as "Admin"
    actor ProjectManager as "Project manager"
    system System

    Admin --> CreateEditDeleteClient
    Admin --> SearchClient
    Admin --> AddClientToProject
    Admin --> CreateEditDeleteContact
    Admin --> AddRemoveContactToFromProject
    User --> CreateEditDeleteClient
    User --> SearchClient
    User --> AddClientToProject
    User --> CreateEditDeleteContact
    User --> AddRemoveContactToFromProject
    ProjectManager --> CreateEditDeleteClient
    ProjectManager --> SearchClient
    ProjectManager --> AddClientToProject
    ProjectManager --> CreateEditDeleteContact
    ProjectManager --> AddRemoveContactToFromProject
    
```

- Export
 - Export report – Admin exports data to Economic
- Other
 - Change language: User, project manager eller Admin skifter sprog (dansk og engelsk)
 - Login med google account: User, project manager eller Admin logger ind med google account



6.2.3.5 Design

Supeo har umiddelbart ingen ønsker til design eller navn. Vi har valgt navnet FLUXtime til systemet. Både flux og time afspejler tid. Teamet har designet et logo i et enkelt og moderne design. Dette vil vi præsenterer for product owner.



6.2.3.6 Prototyping & testing

Vi har valgt at lave prototypes⁴¹ som hurtige skitser på papir, som vi så vil teste på vores kontaktperson hos Supeo. Vi har lavet grænsefladetesten ud fra den formodning om at vores kunde vælger alternativ systemdefinition 1. Prototyperne er ikke endelige skitser, men et udgangspunkt, som skal definere krav, brugere og grænseflade yderligere. Før testen er udført ved vi endnu ikke hvordan vi skal understøtte user familiarity og tage højde for user diversity, da vi ikke er sikre på disse, før en system definition er valgt. Grænsefladerne vil dog være opbygget på sådan en måde at brugere med mindre IT erfaring vil have nemt ved at finde rundt i og anvende systemet, for at opnå den bedste user experience.

Vi vil understøtte user guidance ved at lave tip-tool-text hvor det er nødvendigt. For consistency og minimal surprise vil vi forsøge at anvende samme terminologi og udseende gennem hele systemet, samt gøre at knapper mv. nøjagtigt beskriver hvad de gør i systemet. Vi vil lave et fast sted på skærmen hvor fejlbeskeder bliver vist.

Vi vil forsøge at understøtte recoverability i form af en altid synlig hovedmenu, samt bede brugeren bekære afgørende valg i systemet.

Systemet vil komme med standard Engelsk som sprog, men der vil være mulighed for at skifte til Dansk. Grunden til at det ville komme med Engelsk som standard er at det så har potentielle til at blive solgt udenlandske kunder.

Vi har valgt at lave en fast menu i toppen af siden. Denne menu vil se forskellig ud, alt efter hvilken permission-gruppe brugeren er med i. Dvs. at brugeren ikke vil blive præsenteret for valg, som er irrelevante/mangler rettigheder til.

⁴¹ OOAD s. 31

Login: Oprindeligt ville vi have lavet login som en pop-up, som vi ville have haft gjort hvis det havde været en Java applikation. Men efter mere overvejelse har vi valgt at login skal have en side for sig, da det er en webapplikation. Der er ingen grund til at man skulle kunne se andet end login siden, hvis man ikke kan logge ind. Man indtaster sit brugernavn og password og trykker login. Det er ligeledes muligt herfra at resette password hvis brugeren har glemt det. Dette havde Supeo ingen ændringer til. Vi snakkede dog om, om man kunne logge ind med google account, hvilket vi vil undersøge nærmere senere.

A hand-drawn sketch of a login form titled "POP-UP". The form includes fields for "User" and "Password", a "Forgot password?" link, and a "Login" button.

Home / My Projects: Når man er logget ind, vises en side hvor man kan se de projekter og opgaver man er tilknyttet til igennem Pivotal Tracker. Her kan ligeledes søges efter opgaver. Man kan starte og slutte tidsregistrering på hver opgave og færdiggøre en opgave.

A hand-drawn sketch of a dashboard titled "My projects". It shows a sidebar with "Project 1", "Project 2", "Project 3" (expanded to show "Epic 1", "Epic 2", "Epic 3"), "Project 4", and "Project 5". A "main panel" is shown with a "Story" card for "Opgave 1" (Story: Opgave 1, ID: 187159, Type: Feature, Points: 3, Status: Started, Requester: AMB, Owner: [redacted]). The main panel also has sections for "Opgave 2" and "Opgave 3". Annotations include: "HOME" with arrows pointing to the sidebar and main panel; "Search..." above the main panel; "Logout" in the top right; "Gammes huis ikke relevant" and "Hvilke grupper?"; and "Projects: der loggedes ud man viser alle projekter".

Men efter mødet med Supeo er det blevet besluttet at siden skal hedde My Current Tasks og opgaverne skal ikke være delt op efter epics/kategorier og projekter, som i Pivotal Tracker, men skal blot ligge som en række af igangværende/startede opgaver.

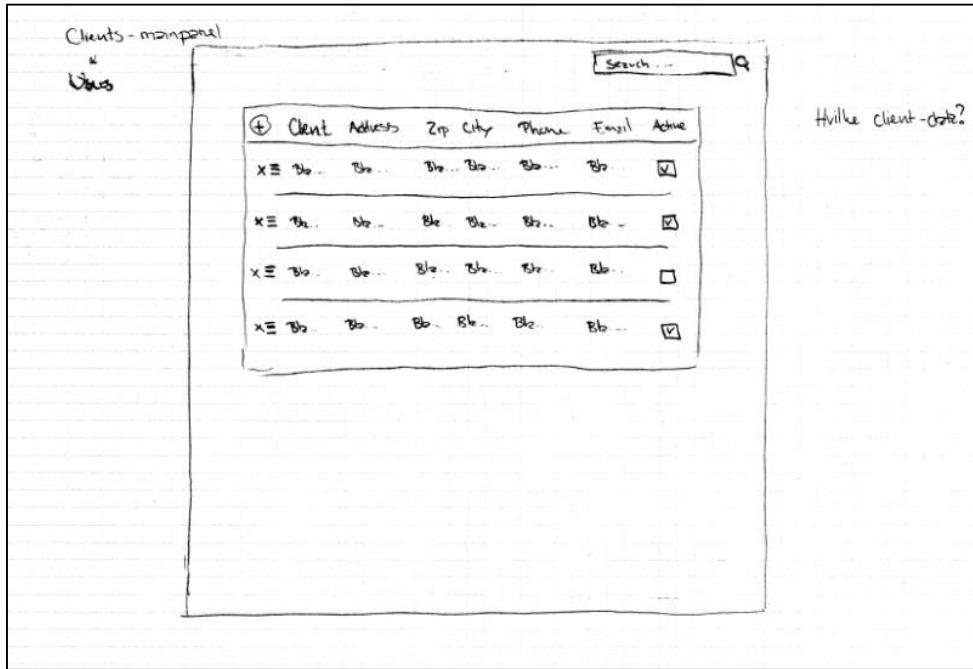
My Profile: Denne side er en lille pop-up hvor man kan ændre enkelte bruger-oplysninger og password til systemet. Grunden til brugeren ikke kan ændre mere, er at nogle af de andre ting er faste variable der bruges andre steder i systemet.

Supeo's ændringer: Systemet må gerne kunne anvendes som medarbejder-database og derfor skal user have flere

A hand-drawn sketch of a "My Profile" pop-up. It contains fields for "User" (AMK), "Name" (Anne-Mette), "Surname" (Knudsen), "ID" (1875720), "Email" (anne@supeo.dk), "Password" (AMK1234), and an "Update" button. An annotation points to the "User" field with "flere user info?"

attributter. Dette skal dog ikke være mandatory fields i systemet. User skal have et id som man kan taste ind, samt et system-id.

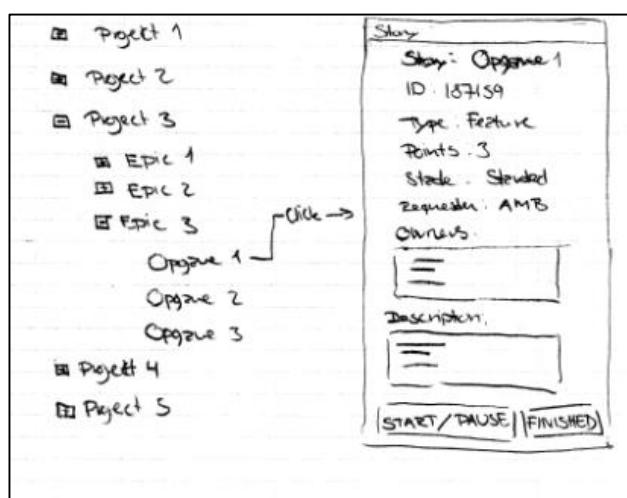
Clients & users: Clients (users-siden vil ligne denne side) kan håndteres- og er listet på denne side.



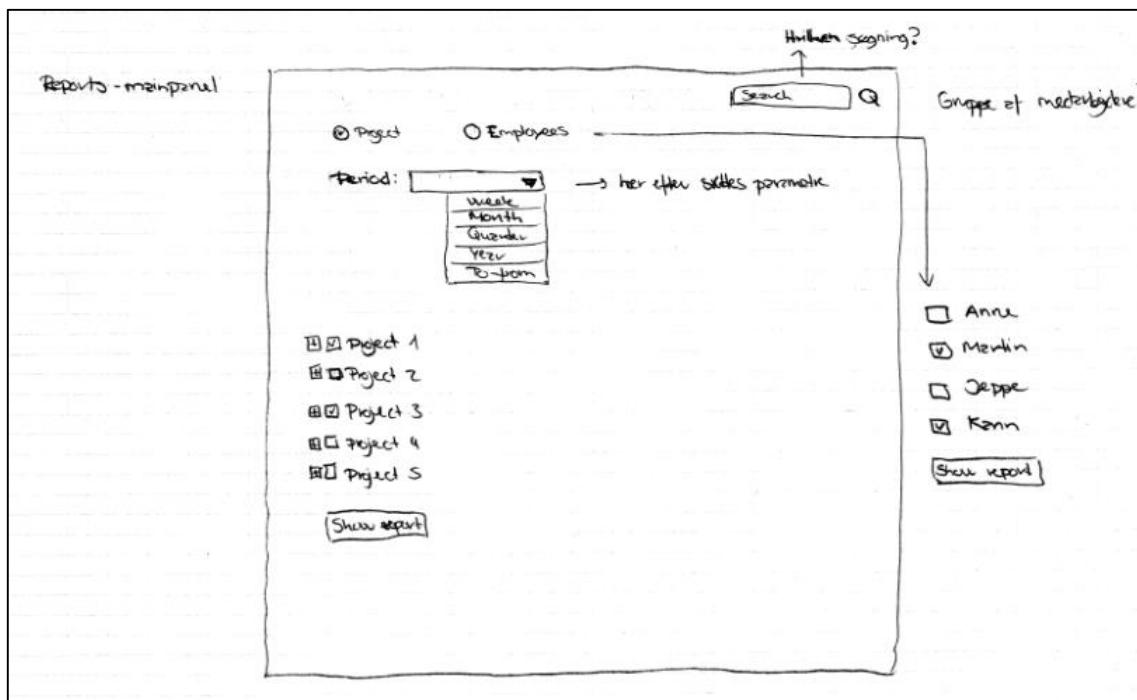
Supeo's ændringer: Systemet må gerne anvendes som kunde-database og derfor skal Client have flere attributter. Dette skal dog ikke være mandatory fields i systemet. Client skal have et id som man kan taste ind, samt et system-id. Ligeledes skal det være muligt at tilføje kontaktpersoner til kunder.

Users skal være tilknyttet permissions-grupper. Dette kan være Admin, project manager eller user.

Projects: Skal vise alle projekter og tilhørende opgaver. Her skal man ligeledes kunne starte opgaver, tidsregistrerer og afslutte. Hertil havde Supeo ingen ændringer.

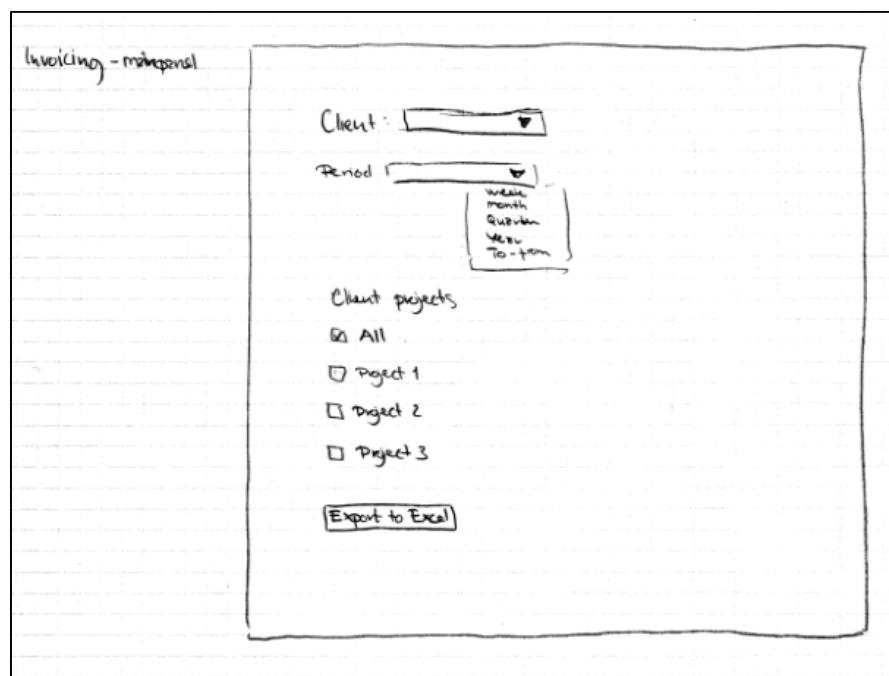


Reports: Rapporter udtrækkes enten ud fra en medarbejder eller et projekt, og så ud fra en tidsperiode. Systemet åbner en pdf-rapport.



Supeo's ændringer: Rapporten skal vises i Excel, da der skal kunne arbejdes videre med data. Det skal også være muligt at lave en Client-rapport.

Invoicing: Også data til fakturering vil blive udtrukket i Excel, da vi ikke har adgang til Economic.



Supeo's ændringer: Det er ikke muligt at indlæse fra Excel til Economic og derfor skal Economic's API anvendes. Dog skal dette kun laves hvis der er tid til det.

Yderligere ændringer:

Det skal være muligt at oprette opgaver, labels/kategorier (epics) og tasks i systemet. Der skal altså kunne

fungere som både stand alone applikation og som tillægsmodul til andre opgave-systemer som Pivotal tracker.

Dette gør at user familiarity ikke rigtig kan understøttes, da systemets brugere kan være fra mange forskellige brancher, stillinger og virksomheder. Der er altså en stor user diversity, hvilket giver ekstra udfordringer, når vi skal vælge terminologi mv.

Systemet skal anvende Zend Framework 2. Dette er nu et krav. Det skal lige ledes synkronisere med Pivotal Tracker hvert minut. Yderligere vil Supeo meget gerne have at vi anvender doctrine sammen med Zend Framework 2.

6.2.3.7 Zend Framework 2

ZF er et open source framework til at udvikle web applicationer og services ved hjælp PHP 5.3. ZF2 bruger 100% object-orienteret kode. En stor fordel ved at bruge et framework er dets stærke datastruktur, der gør at man nemmere kan implementere og vedligeholde systemer. ZF2 bruger autoloadere og configurationsfiler til at hente assets ind med. ZF2 kan deles op i moduler, hvilket passer perfekt til brugen af en webapp da url'en kan deles sådan op. Det har indbygget PHP-Unit-Test.

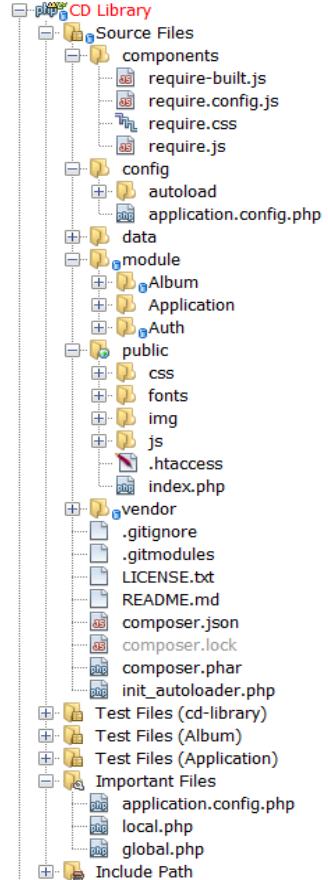
6.2.3.7.1 Datastruktur

Hvis man skal udvikle større systemer er det vigtigt med en god datastruktur. Dette har ZF2. Faktisk kan man selv sætte sin struktur op, men vi har valgt at følge den som man, fra Zend, anbefaler.

Selve applicationen kan indstilles under config mappen. Her indstiller man hvilke autoloaded filer man vil bruge hver gang siden loader plus hvordan applicationen er konfigureret.

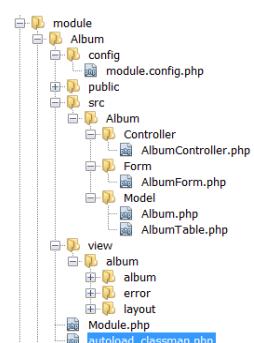
Public mappen indholder CSS, fonte, billeder og javascript som siden skal kunne tilgå. Det ligger også en index-fil, det er denne man starter når programmet skal køre. Vi har sat det op sådan at man i CSS mappen kan ligge en modulmappe og lægge de tilhørende stylesheets derind. Det samme gælder for billeder og javascript.

Vendor mappen er ZF's egen mappe hvor der ligger de eksternt hentede modules som man kan udvide med. F.eks. RabbitMQ-modulet.



6.2.3.7.2 Modul

I modul mappen ligger alle moduler som der skal kunne angives i url'en. Det enkelte modul er delt op efter en specifik struktur. Man kan indstille modulet i config/module.config.php og i /Module.php. I src-mappen skal der altid være en mappe med modullets navn. I den er der controllerne, forme til views og model-klasserne. I view-mappen ligger der både sider til Error handling f.eks. error 404. Der ligger også et generelt layout for siden. Her kan man lave de mere statiske elementer på siden. I /modulet navn mappen ligger de respektive views som kan skiftes ud alt efter hvordan URL'en ser ud.



6.2.3.7.3 URL opdeling

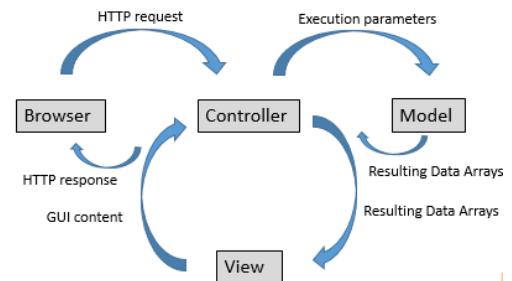
`http:// systemname.countrydomain / modulename / action / params`

Eksempel: `http:// fluxtime.dk / fluxuser / edit / 213`

URL'en er helt specielt opdelt. Først applicationens navn, derefter modulets navn og så en action (metode) og så til sidst parameter. Actions bliver lavet i controllerne.

6.2.3.7.4 MVC

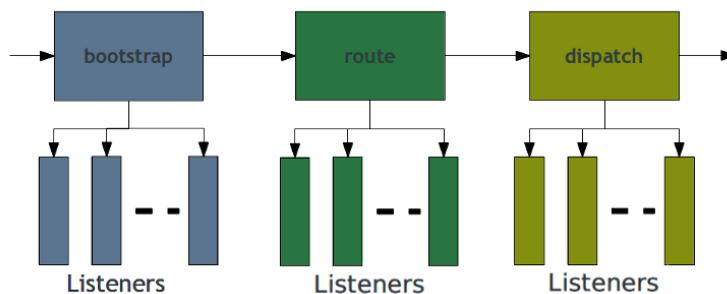
Som figuren viser, anvender ZF2 også en MVC-arkitektur. Dvs. at der findes et view-, model- og control-lag. Dog ligger alle control-klasser (det samme gælder model og view klasser) ikke i samme package. De kan deles op i moduler, som beskrevet. Et modul kan være en samling af use cases som benyttes til ét object, f.eks. CRUD-metoder for et User-objekt. Under hvert modul er så lavet en MVC-lagdeling. På figuren kan ses hvordan et kald browseren laves som en http request til controller. Herefter eksekveres på modellen, som returnerer arrays. Arrays sendes til view-laget hvor et nyt view oprettes. Dette sendes tilbage til controlleren, som så sender et http response tilbage til browseren.



6.2.3.7.5 Routing

Routing kan i korte træk beskrives som at komme fra punkt A til B. En request fra browseren skal altså sendes til controllerne. Routing mapper et http request til en controller (routes er defineret i modulernes config-filer). En routing algoritme sammenligner url (request) med routes og returnerer nødvendige parametre som controllers navn mv. Vi vil anvende segment routing, som sammenligner segment(er) af url og literal routing til login.

Figuren nedenfor viser hvordan en applikations-bootstrap loader alle nødvendige system-filer når systemet opstartes. Herefter routes og der sendes et request-objekt til en dispatcher. Dette objekt indeholder controller-navn, action og andre nødvendige parametre. Dispatcher instantierer controller og kalder den relevante action.



6.2.3.8 Research af arkitektur

6.2.3.8.1 Eksempel på MVC arkitektur med DAO pattern (f.eks. Java)

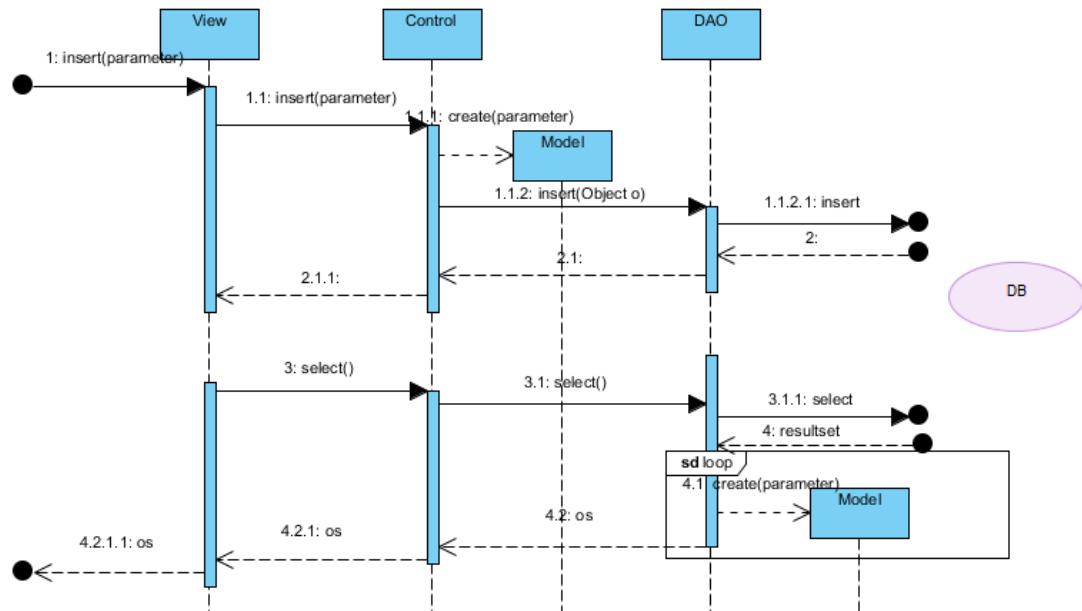
-View-lag: GUI-klasser

-Control-lag: Control-klasser som modtager kald fra view og sender kald videre, samt kan skabe objekter

(controller & creator pattern⁴²)

-Model-lag: Model-klasser

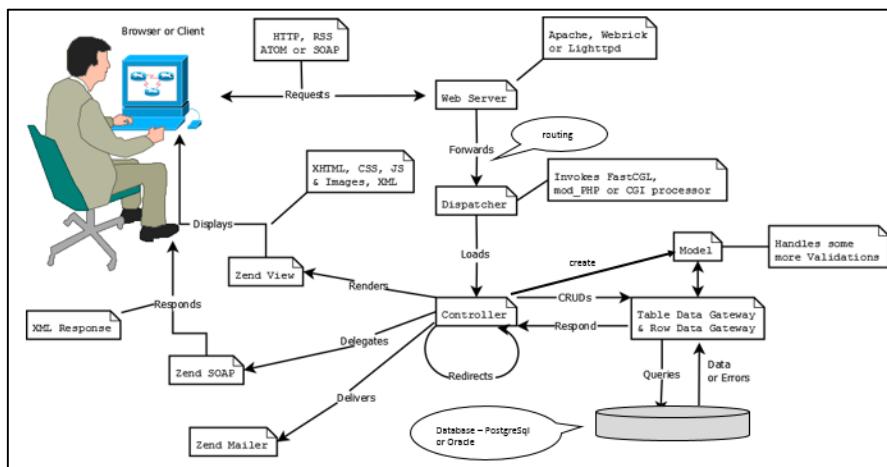
-DAO-lag (data access objects): Kalder databasen, samt skaber objekter af result set fra databasen (creator pattern & information expert)



Figuren viser hvordan view sender et insert-kald til control-lag, som skaber et objekt og sender det til DAO-laget, som indsætter det i databasen. I view kaldes en metode som henter alle objekter fra databasen, gennem control, til DAO, til database. I DAO skabes objekter fra resultset som sendes tilbage gennem systemet og ud til view. *Udviklerne skaber alle lag og metoder, samt stored procedures i databasen og skal håndtere synkronisering af metoder, transactions og locks.*

6.2.3.8.2 Zend Framework 2 arkitektur

Nedenstående figur viser hvordan datakommunikation er i Zend Framework 2. Vi har lavet dette billede for at kunne forstå Zend's arkitektur og dermed omsætte figuren til et sekvensdiagram.

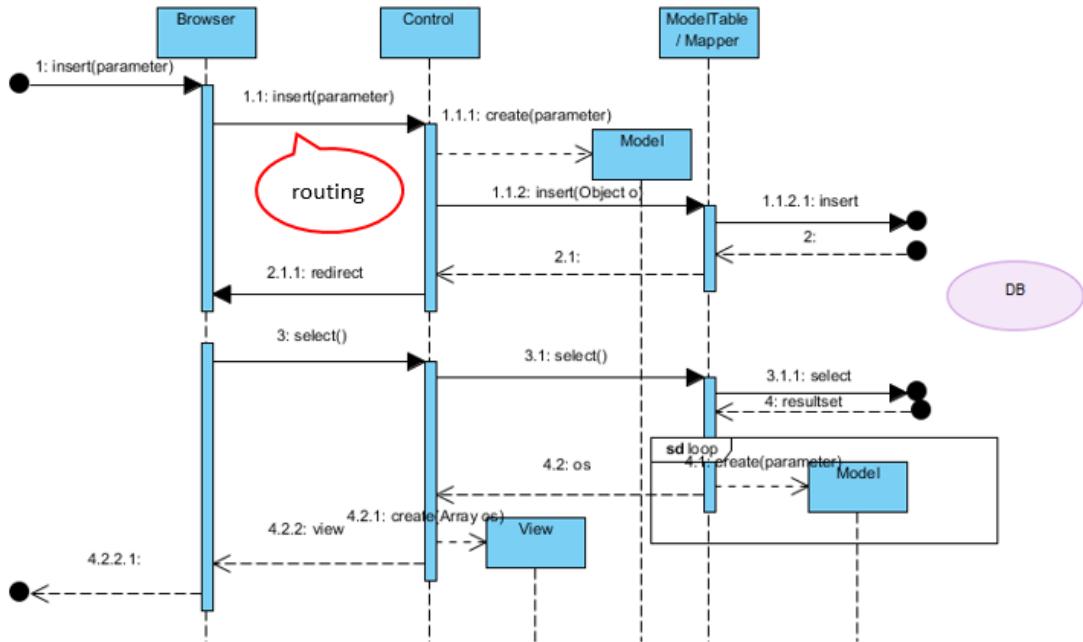


⁴² Larman s. 271

-View-lag: Forms (GUI)

-Control-lag: Control-klasser som modtager kald fra browser og sender kald videre til Mapper, samt kan skabe objekter (controller & creator pattern)

-Model-lag: Model-klasser og Mapper-klasser (eller ModelTable-klasser): *Mapper-klasser er DAO-klasser og arver fra klasserne TableDataGateway og RowDataGateway (klasser i frameworket). Disse klasser kan anvendes med og uden stored procedures og klasserne skal implementeres af udviklerne.*



Figuren viser hvordan et objekt indsættes. Fra en browser kaldes controller vha. routing med en url som parameter, der så opretter et objekt. Objektet sendes til Mapper som indsætter i databasen. Controller sender et redirect til browseren (for at vise ny side med alle objekter). Alle objekter hentes gennem controller, mapper og database, som så sender alle objekterne tilbage til controller. Controller opretter et view og sender dette tilbage til browser. *Frameworket synkroniserer selv metoder og udviklerne skal derfor ikke gøre dette og skal ikke anvende transactions og locks.*

6.2.3.8.3 Zend Framework 2 med Doctrine arkitektur

Arkitekturen ser ud som vist ovenfor. Forskellen er at Doctrine opretter både model-klasser (kaldet entity-klasser) og mapper-klasser, samt også håndterer transactions og locks. Ved brug af Zend Framework 2 og Doctrine, programmerer vi derfor kun view- og control-lag og evt. mere komplekse metoder til database-kald i mapper-laget. *Stored procedures i databasen er altså unødvendige. Dog kan udviklerne selv lave mere custom og komplekse metoder til kald af databasen i disse mapper-klasser, såfremt det er nødvendigt.*

6.3 Elaboration

6.3.1 Iteration 1, Users

Use cases: "Create user", "Edit user", "Delete user", "Search user", "Edit profile"

6.3.1.1 Requirements

6.3.1.1.1 Domain model

FluxUser er en bruger af systemet. Vi har valgt at give FluxUser alle attributterne på figuren, da det er prioriteret fra Supeos side at systemet kan anvendes som medarbejder-database. Employeeld kan være et ID fra et andet system, f.eks. lønnummer.

6.3.1.1.2 Detailed use cases

Vi har lavet detailed use cases, men ikke system sequence diagrams, da vi ikke fandt dette nødvendigt. Vi har ikke lavet en detailed use case for "Search user", men blot besluttet at det skal være muligt at angive et søgeord som matches med firstname, lastname og username.

Create user:

Actor: Admin

Main success scenario:

1. Actor starter Create user
2. Actor angiver user-oplysninger(employeeld, firstname, lastname, phone, privateEmail, workEmail, username, password, street, houseNumber, zipCode, city, country, phonePrivate)
3. Systemet gemmer user-oplysninger
4. Systemet præsenterer users

Edit user:

Actor: Admin

Pre-condition: Systemet præsenterer users

Main success scenario:

1. Actor starter Edit user og angiver user(FluxUser)
2. Systemet præsenterer user-oplysninger
3. Actor redigerer user-oplysninger(employeeld, firstname, lastname, phone, privateEmail, street, houseNumber, zipCode, city, country, phonePrivate)
4. Systemet gemmer user-oplysninger
5. Systemet præsenterer users

Vi har valgt at username og workEmail ikke kan redigeres. workEmail vil blive anvendt til login og username er et fast "short name" som brugeren kaldes i systemet.

Delete user:

Actor: Admin

Pre-condition: Systemet præsenterer users

Main success scenario:

1. Actor starter Delete user og angiver user(FluxUser)
2. Actor bekræfter sletning af user

FluxUser
-employeeld
-firstname
-lastname
-phone
-privateEmail
-workEmail
-username
-password
-street
-houseNumber
-city
-zipCode
-country
-phonePrivate

3. Systemet sletter user
4. Systemet præsenterer users

Edit profile:

Actor: Admin, Project manager eller User

Main success scenario:

1. Actor starter Edit profile
2. Systemet præsentere profile-oplysninger
2. Actor redigerer profile-oplysninger(password, newPassword, phonePrivate, privateEmail)
3. Systemet gemmer profile-oplysninger
4. Systemet præsenterer tasks

Vi har valgt at en bruger gerne selv må opdatere sit private telefon nummer og private email og selvfølgelig også ændre password. Vi har anvendt vores prototypes til flow og use casen afsluttes ved at My current tasks vises. Dette kan selvfølgelig ikke implementeres endnu og der vises en tom side.

6.3.1.2 Analysis & design

6.3.1.2.1 Database diagram

Vi har designet tabellen neden for vha. domæne modellen. Vi har besluttet at en FluxUser aldrig må slettes fra systemet og derfor har vi tilføjet "state" i tabellen i databasen. "state" sættes til 0 ved sletning.

Vi har valgt ikke at normalisere by og postnummer. Vi havde ellers overvejet om det skulle gøres, sådan at brugeren kunne anvende auto completion (systemet viser automatisk bynavn, når postnummer indtastes). Dette gik vi dog bort fra da det kun er admin, som må oprette fluxusers og derfor synes vi ikke det var relevant at lave denne funktion til så få brugere af systemet. En anden grund er at systemet skal kunne benyttes i andre lande og derfor kan vi ikke oprette alle postnumre og byer i databasen.

flux_user		
id	integer(10)	
employee_id	varchar(30)	
firstname	varchar(40)	
lastname	varchar(40)	
phone	varchar(20)	
private_email	varchar(50)	
username	varchar(20)	
work_email	varchar(50)	
password	varchar(72)	
street	varchar(50)	
house_number	varchar(10)	
city	varchar(50)	
zip_code	varchar(10)	
country	varchar(50)	
phone_private	varchar(20)	
state	integer(10)	

Vi har valgt at "work_email" erunik og skal derfor senere skal benyttes ved login. Vi har valgt ikke at benytte username (dette vil blot være et short name i systemet), da Supeo har ønsket at kunne logge ind med google konto, hvor systemet ligeledes vil skulle bruge "work_email".

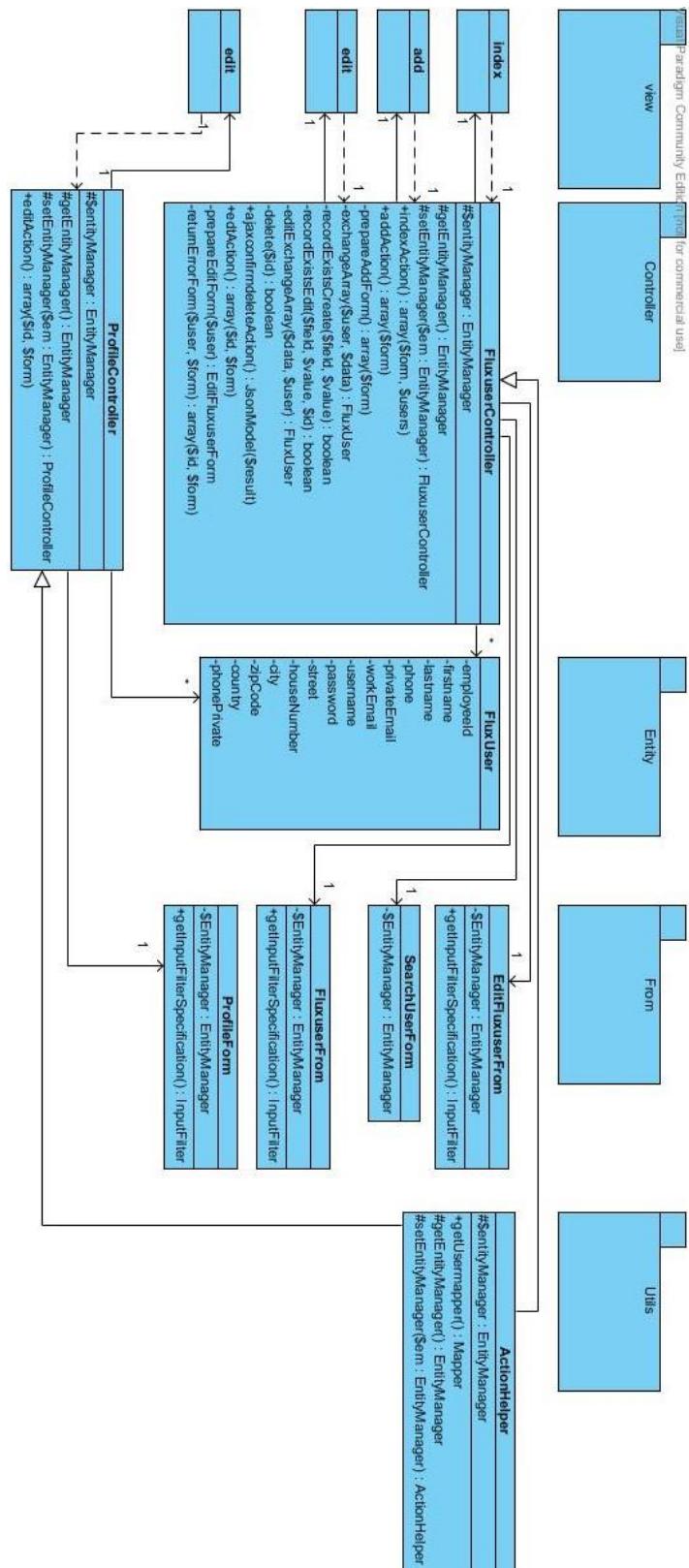
"employee_id" skal i princippet også være unikt, men da dette gerne må være null, må dette sikres i programmet.

6.3.1.2.2 Design class diagram

Vi har designet klasser og metoder og illustreret dette vha. et design class diagram, som ligeledes viser attributter og klassernes relationer. Hver action i controllerne har en tilhørende view-fil af samme navn som controller-klasserne kan oprette model- og form-objekter og det er ligeledes fra controller-klasserne at vi anvender doctrine-mappers til database-kald.

Hver action i controllerne tager ikke parametre (input) fra brugeren som vi kender det fra Java, men modtager en post (http request) og derfor ser diagrammet også forenklet ud. Vi har allerede bestemt hvilke

data der bliver udvekslet mellem bruger og system vha. detailed use cases og model-objekterne autogenereres ud fra databasen vha. Doctrine, så datatyper bestemmes af databasen.



Vi har bestemt at hver controller skal extende ActionHelper. ActionHelper skal extende AbstractActionController. De metoder som kaldes i flere controllers, kan så med fordel placeres i ActionHelper.

"Delete user" har vi besluttet skal implementeres som ajax-kald (ajaxconfirmdeleteAction), så index-siden ikke skal genindlæses efter sletning. Metoden kalder delete-metoden.

Metoderne exchangeArray og editExchangeArray kan benyttes når en post modtages (editAction og addAction), til at sætte attributter på et objekt, før det gemmes i databasen.

Hver form skal extende Form og skal implementeres vha. detailed use cases og database diagrammet.

Vi har, som nævnt, besluttet at give FluxUser en "state". Dvs. at der i indexAction skal hentes alle fluxusers hvor state ikke er 0 (også ved søgning). Ligeledes sættes "state" = 0 i metoden delete.

6.3.1.3 Implementation

6.3.1.3.1 Implemented components

Klasserne i diagrammet ovenfor er implementeret i systemet i modulet Fluxuser bl.a. vha. Doctrine (se guide 10.4.3). For at undgå fejl i databasen valideres input både vha. javascript på klient-siden og vha. form-validering på server-siden i controllerne. Længden af alle input-data valideres, så det passer med max-værdier i databasen og felter som ikke er nullable i databasen er gjort required. Derudover valideres workEmail vha. et regular expression. Ligeledes gør username og password. Kun bogstaver og tal er tilladt i password og username trimmes for whitespaces.

For at validere workEmail, username og employeeld, som skal være unikke (employeeld kun hvis den ikke er null), havde vi først lavet validering vha. et ajax-kald. Dette var desværre ikke hurtigt nok og brugeren kunne når at submitte formen før data var valideret. Derfor besluttede vi at lave 2 recordExists-metoder i FluxuserController, som anvendes til validering. Endvidere er metoden returnErrorForm tilføjet. Denne metode anvendes i editAction til at returnere formen incl. error message hvis employeeld allerede eksistere i databasen.

Ved ajax-kald, hvor modaler er anvendt (Delete user) bruges javascript/jQuery. js-filer ligger i public-package.

En brugers password skal krypteres i databasen, men dette tilføjes når "Login" implementeres.

Vi har valgt at systemets brugere ikke kan benytte systemet hvis javascript er slået fra. Der vil så komme en fejlbesked. Dette har vi valgt da meget validering og alle ajax-kald laves vha. javascript. Alligevel validerer vi også input på server-siden, for vha. formenes inputfiltre, da vi vil sikre at database-fejl ikke ikke forekommer. Dette er gjort i layout.

For at overskueliggøre vores kode, har vi lavet metoderne prepareAddForm og prepareEditForm. Alle tilføjelser og ændringer er tilføjet til vores design class diagram ovenfor.

Nedenstående klasser er også implementeret og beskrives her kort:

module.config (i Fluxuser module): Routes og andre indstillinger for modulet defineres her

Module (i Fluxuser module): Indstillinger for modulet defineres her incl. metoder

global (i autoload): Globale indstillinger

local (i autoload): Nøgler

error-package (view): 404 og index – views til error

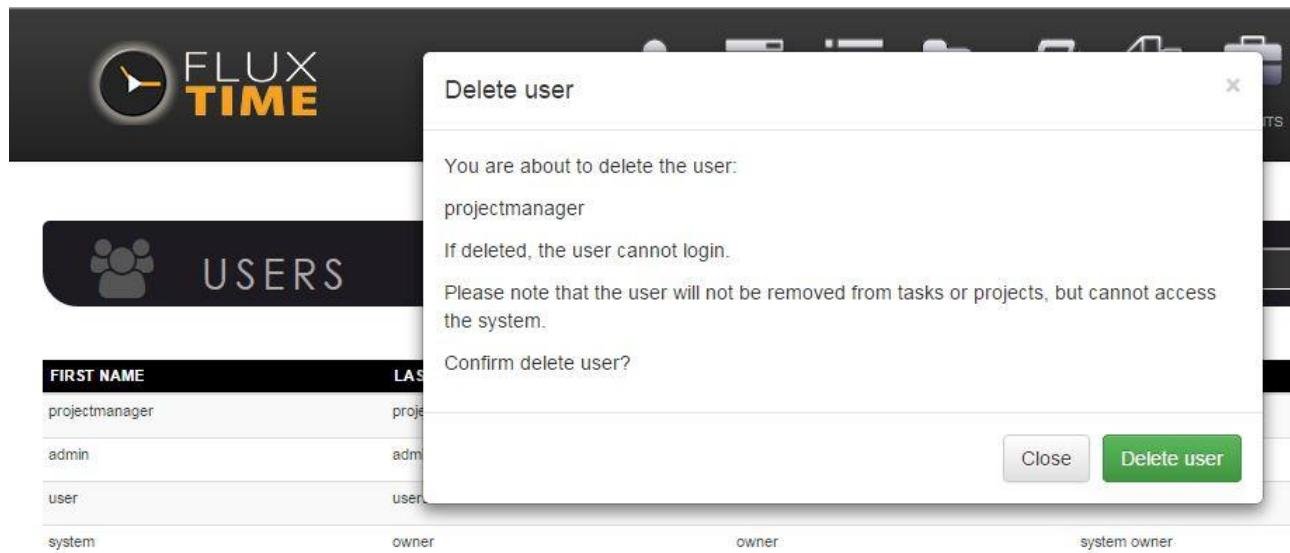
layout-package (view): layout (custom.css indlæses, samt javascript) og menu (hovedmenu)

public-package: I denne package ligger bl.a. alt CSS, billeder og alt javascript. Vi har anvendt fontawesome til ikoner og bootstrap til styling. I package "css" har vi implementeret "custom.css", som anvendes på alle sider (styling). I package "js" er javascript-filer. Under "module" i "public" laves packages til javascript som benyttes på de enkelte sider. Vi har implementeret 2 javascript-filer (index og add_edit_fluxuser), som ligger i package "fluxuser". Disse anvendes til validering af input og til ajax-kald.

vendor-package: Eksterne plugins

composer.json: Util som holder styr på eksterne plugin's versioner. Kan update plugins efter hvad der er angivet i filen.

6.3.1.3.2 Screen prints

A screenshot of the FluxTime application interface showing a 'REDIGER PROFIL' (Edit Profile) form. The form includes fields for 'Password*', 'Ny password' (New password), 'Brugernavn' (Username), 'Fornavn' (First name), and 'Efternavn' (Last name). Below these are fields for 'Privat telefon' (Private phone) and 'Privat email' (Private email). At the bottom is a green 'Gem profil' (Save profile) button.

6.3.1.3.3 Unit testing

I dette sprint vil vi gerne sikre at man får SQL-exceptions når ikke-validate data indsættes. Dette har vi testet vha. unit tests. Vi har testet følgende:

Show fluxuser index page - ingen exception: Passed

Create new valid user – ingen exception: Passed

Create new invalid user – exception: Passed

Delete existing user – ingen exception: Passed

Delete not existing user – exception: Passed

Edit user with valid input – ingen exception: Passed

Vores tests er alle "passed" og vores use cases og database fungerer.

6.3.1.4 *Testing*

6.3.1.4.1 *Funktionstest*

Vi har testet alle use cases og at både vores javascript-validering og form-validering i controlleren giver brugeren de rigtige fejlmeldelser og at der ikke kan opstå fejl i databasen. Endvidere har vi testet recordExists-metoderne (unikke data i databasen), regular expressions og database udtræk (brugeren må ikke kunne se fluxusers med "state" = 0).

6.4 Construction

6.4.1 *Iteration 1, Usergroups*

Use cases: "Add user to usergroup" og "Remove user from usergroup"

6.4.1.1 *Research ACL (Access control list)*

En smart ting ved Zend Framework 2 er deres page-permissions. Man kan definere en liste af page-permissions, som Zend benytter til at bestemme om man har rettighed til den side man vil se. Man bruger et modul der hedder ACL. Man kan statisk lave en liste for permissions for hver brugergruppe, men man kan også benytte en mere dynamisk tilgang, og gemme permissions i en database, dette ønsker vi at gøre. Man kan så oprette en tabel med ressource-navne, som er alle URLs der bliver brugt på sited. F.eks. "fluxuser/index ". Ressource-navne kan så knyttes til en brugergruppe via en associationstabell (ressourcepermissions) og hver enkelt bruger kan så tilknyttes en gruppe. Disse ressourcepermissions loades hver gang en side indlæses.

Når websiden loades vil Zend starte med at køre en metode der hedder onBootstrap(MvcEvent \$e). Den kan man overloade og definere selv i module.php filen. Man kan så oprette et tilhørende plugin, hvori man tjekker den ønskede url⁴³. Her kan også angives hvilken fejlside der skal vises hvis brugeren ikke har tilladelse eller hvis den action ikke findes i systemet.

Hvis en bruger ikke er logget ind, vil man automatisk give brugeren rollen "guest" og redirekte til en login-side. Kun rollen "guest" skal have permission til login.

Man kan ligeledes, i views og controllers, tjekke om brugeren har adgang til bestemte use cases på en side. Der kan man benytte samme side til flere roller, da man kan gemme buttons mv. for brugeren ved at undersøge brugerens rolle.

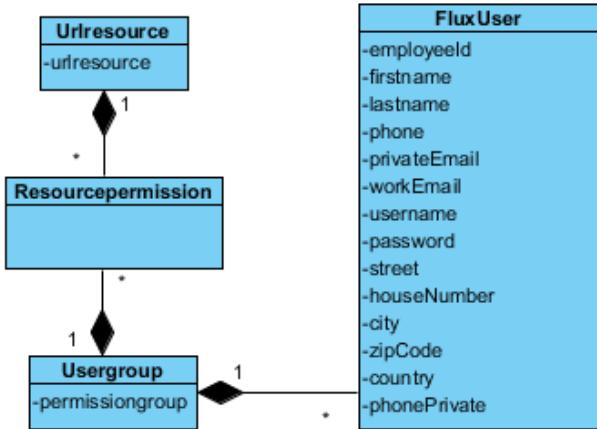
6.4.1.2 *Requirements*

6.4.1.2.1 *Domain model*

Vi har tilføjet 3 konceptuelle klasser til vores domæne model. Urlressource er en permission (rettighed) og en Usergroup er en brugergruppe der angiver hvilke rettigheder brugeren har i systemet (Resourcepermission er en associationsklasse). Oprindeligt havde vi forestillet os at vi ville lave en mange-til-mange relation mellem Usergroup og Fluxuser, således at brugeren kunne tilføjes flere usergroups og

⁴³ <http://ivangospodinow.com/zend-framework-2-acl-setup-in-5-minutes-tutorial/>

dermed gøre systemet mere fleksibelt. Efter vores research af ACL besluttede vi dog at lave en en-til-mange relation i stedet, da vi så lettere kan tjekke hvilke permissions brugeren har. Alternativt skulle en liste gennemløbes og dette ønsker vi ikke. Vi har valgt en composition mellem Usergroup og FluxUser, da en FluxUser skal være associeret til en Usergroup.



6.4.1.2.2 Detailed use cases

Da vi oprindeligt ville have lavet en mange-til-mange relation mellem Usergroup og Fluxuser, havde vi tænkt at "Add user to usergroup" og "Remove user from usergroup" var selvstændige use cases, men da FluxUser nu kun er associeret til én Usergroup, vil vi blot tilføje eller ændre brugerens Usergroup når brugeren oprettes eller redigeres. Derfor har vi opdateret nedenstående detailed use cases for forrige sprint. Vi vil ligeført opdatere "Search user" så søgeord også kan matche Usergroup.

Create user:

Actor: Admin

Main success scenario:

1. Actor starter Create user
2. Actor angiver user-oplysninger(`employeed`, `firstname`, `lastname`, `phone`, `privateEmail`, `workEmail`, `username`, `password`, `street`, `houseNumber`, `zipCode`, `city`, `country`, `phonePrivate`, `Usergroup`)
3. Systemet gemmer user-oplysninger
4. Systemet præsenterer users

Edit user:

Actor: Admin

Pre-condition: Systemet præsenterer users

Main success scenario:

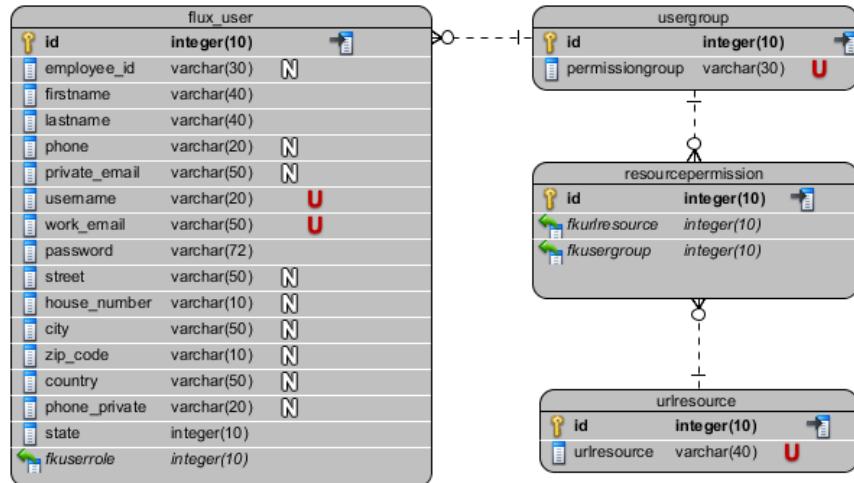
1. Actor starter Edit user og angiver user(`user`)
2. Systemet præsenterer user-oplysninger
3. Actor redigerer user-oplysninger(`employeed`, `firstname`, `lastname`, `phone`, `privateEmail`, `street`, `houseNumber`, `zipCode`, `city`, `country`, `phonePrivate`, `Usergroup`)
4. Systemet gemmer user-oplysninger
5. Systemet præsenterer users

6.4.1.3 Analysis & design

6.4.1.3.1 Database diagram

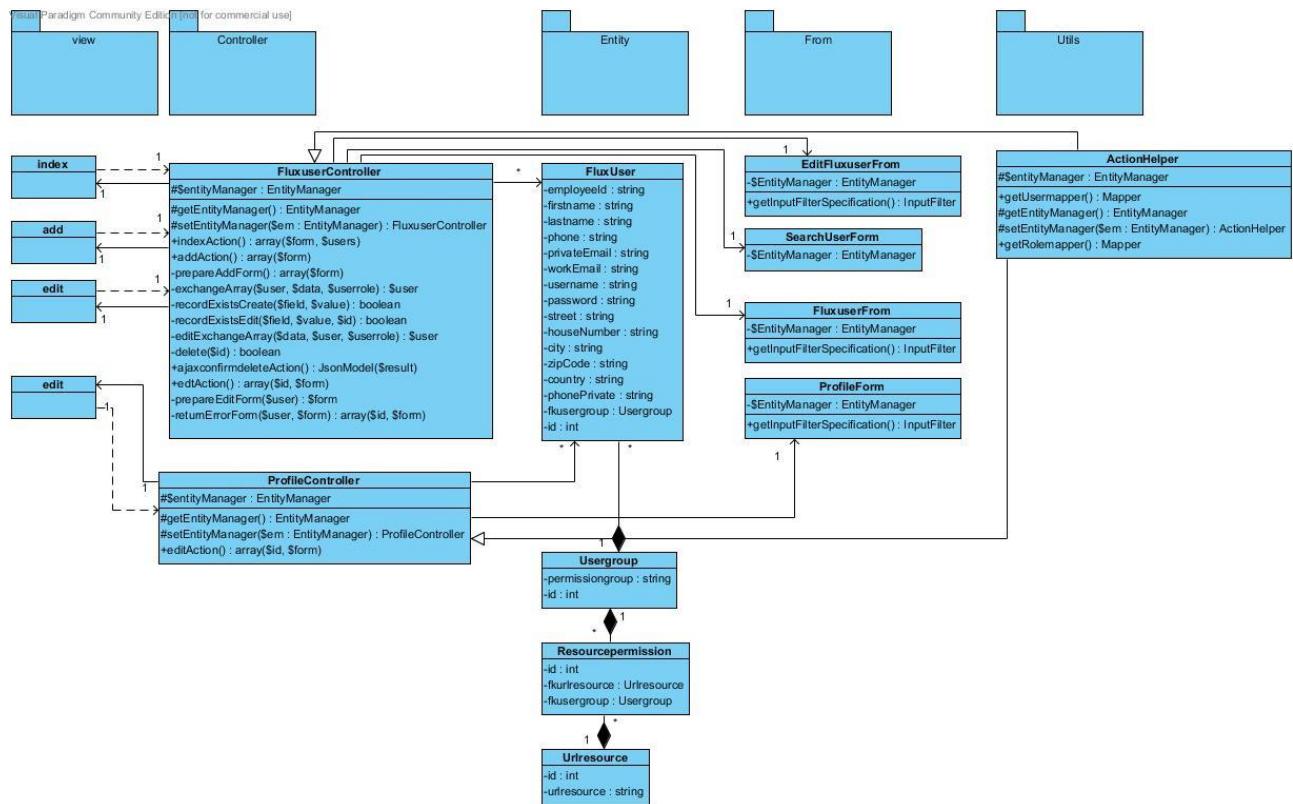
Diagrammet er design vha. vores domæne model. Disse roller skal lægges fast i databasen: "admin", "user", "guest" og "project manager"

"resourcepermission" skal have en unique constraint(fkurlresource, fkusergroup)



6.4.1.3.2 Design class diagram

Vi har tilføjet de 3 nye model-klasse på diagrammet nedenfor. Ligeledes er der i ActionHelper tilføjet metoden getRoleMapper() og i FluxuserController hvor de 2 exchangeArray-metoder findes gives Usergroup med som parameter.



6.4.1.4 Implementation

6.4.1.4.1 Implemented components

Disse roller er tilføjet i databasen i tabellen "usergroup": admin (id: 1), user (id: 2), guest (id: 3), project manager (id: 4). Vi har tilføjet "guest" da vi pga. vores research af ACL allerede nu ved at vi skal anvende denne rolle ved login.

Disse permissions er tilføjet i tabellen "urlressource" og permissions er tilføjet til hver brugergruppe:

UrlResource	admin	user	guest	project manager
fluxuser/add	X			
fluxuser/ajaxconfirmdelete	X			
fluxuser/edit	X			
fluxuser/index	X	X	X	X
profile/edit	X	X	X	X

Vi har valgt ikke at skjule buttons i view endnu, da vi alligevel først kan teste dette, når "Login" er implementeret. Dette gør vi derfor i næste sprint.

6.4.1.5 Testing

6.4.1.5.1 Funktionstest

Vi har testet "Edit user" og "Add user". Det eneste vi på nuværende tidspunkt kan teste er at usergroup "guest" ikke hentes fra databasen og dermed ikke kan tilføjes til en bruger, samt at en Usergroup kan tilføjes (og ændres) til en FluxUser, samt at en bruger ikke kan eksistere uden en Usergroup.

6.4.2 Iteration 2, Login

Use cases: "Login", "Logout", "Reset password" og "Confirm user"

6.4.2.1 Research

Da ingen i projekt-teamet har erfaring med it-sikkerhed, startes denne iteration med research af dette område.

Når man skal beskytte password kan man både bruge encryption, hashing og encoding. Alle tre oversætter data til et nyt format. Ved encryption og encoding kan originalteksten genskabes. Ved decryption anvendes en nøgle til oversættelse af en krypteret tekst. Hashing er når man uigenkaldeligt samler tekst i et kort fingeraftryk, som ikke kan decrypteres. Dette vil vi anvende til brugerpassword.

En af fordelene ved at bruge et Framework, er at man har flere metoder tilrådighed – også metoder til at håndtere sikkerhed. Zend's dokumentation er dog til tider ukomplet og en anelse misvisende. Dette gør at vi vil bruge en mere PHP-inspireret tilgang til kryptering, hvor vi vil laver vores egne metoder til at de- og encrypte med. Men når det er sagt så byder Zend på virkelig mange og fantastiske tidsbesparende metoder, bl.a. authentication som vi vil anvende.

Vores system har brug for et authentication system til login. Vi har bestemt at alle brugere skal logge ind før de kan se noget funktionalitet, foruden Reset password. Når man skal logge ind skal man bruge et password og en email som brugernavn. Hvis man gemmer et password ukrypteret i databasen, så vil man være mere følsom over for angreb på databasen. Vi har selvfølgelig valgt at vi vil encrypte vores bruger-passwords. Man kunne selvfølgelig sagtens have brugt en capcha eller en honeypot for at beskytte mod "maskiner" der

forsøger at komme ind på siden, men dette behøver vi ikke da det kun er vores admin rolle der kan oprette nye brugere.

Der findes et utal af måder man kan sikre sit password. For et årti siden brugte man hashing-metoderne MD5 og SHA1 til at beskytte sine vigtige data med, men hackere begyndte at oprette store databaser med millioner af oversatte passwords kaldet Rainbow Tables og det blev derved meget nemmere at finde og oversætte passwords og man skulle derved ikke at bruge brute-force metode til alle, men derimod kun et enkelt kald til en database. I PHPs nyere version 5, har de valgt at lave en større refactorering af deres sikkerhedssystem. Bland andet er der indført password_hash(). Normalt vil en standardiseret funktion ikke nødvendigvis være tilstrækkelig, men i dette tilfælde ser det ud som det er sikkert. Vi vil derfor benytte denne password_hash() metode til at beskytter med vores password med. password_hash er en metode som kan hashe en tekst ved at den bruger den meget sikre BCrypt algoritme. BCrypt er en adaptiv funktion, dvs. den fungerer via iterationer og den tid det tager at decrypte øges ved at angive en cost-værdi. Dette skal gøre metoden langsommere, så den forbliver modstandsdygtig over for Brute force-angreb. Også selvom vores beregningsstyrke hele tiden øges på vores computere.

BCrypt algoritmen er en one-way-hashing algoritme og bruger blowfish algoritmen til at hashe password. Hvis en hacker prøver at decrypte, vil én dekryptering kunne iterere flere milioner krypterede elementer i minuttet, men med Bcrypt nedsætte dette betydeligt. Dette gør at det vil tage mange år for at løbe alle de nødvendige password igennem for at finde det rigtige password.

Blowfish algoritmen er en algoritme fra 1993 der tager udgangspunkt i et 32-448 bit password. Mange algoritmer der fandtes før er udviklet af virksomheder eller stat, og er derfor ikke tilgængelige eller var copyrighted. Blowfish er designet med det formål at alle kan bruge den, og den er, så vidt vides, ikke brutt endnu.

BCrypts hash er opdelt således: \$[algoritmeidentifier]\$[Cost]\$[Salt]\$[PasswordHash]

algoritmeidentifier er hvilken version af algoritmen der skal bruges. 1 = MD5, 2 = bcrypt 2y=seneste algoritme.

Cost er en iterationsfaktor som man bruger når man beregner antal iterationer ved beregningen af hashet. Default værdi er 10, som betyder at man $2^{10} = 1024$ iterationer. Det er derfor at den bliver forsinket⁴⁴.



⁴⁴ <http://stackoverflow.com/questions/5393803/can-someone-explain-how-bcrypt-verifies-a-hash>

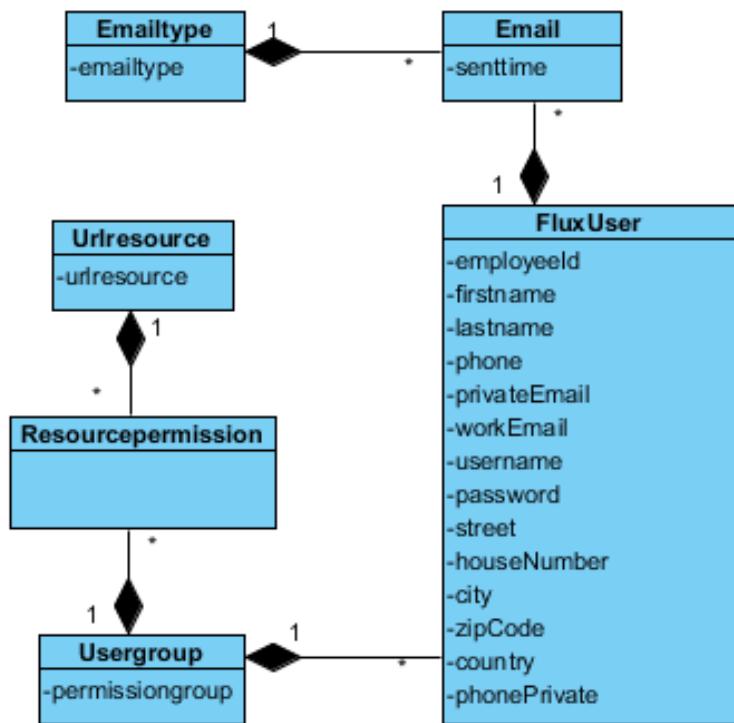
For at styrke login processen vil vi gerne anvende en velkomst-email, som skal sendes når brugeren oprettes i systemet. I emailen skal der være brugernavn og midlertidige kodeord, samt et link til systemet. Via linket, der henviser til siden hvor man kan bekræfte hvem man er, indtastes nyt password. I url'en skal der være et encrypted id til den email som er tilsend til brugeren, sådan så vi kan finde den i databasen og checke om vedkommende er hvem de siger de er. Vi har som vist i de tidligere sprints valgt at admin skal angive brugerens første password, men man kunne evt. have lavet det sådan at det var autogenereret. Ved "Reset password" skal ligeledes sendes en mail til brugeren med link og krypteret email-id.

Til mail encryptionen vil vi bruge blowfish algoritmen og en secret key. Vi har valgt at bruge encrypt funktionen i PHP (bcrypt).

6.4.2.2 Requirements

6.4.2.2.1 Domain model

Vi har tilføjet 2 konceptuelle klasser til domæne modellen, Emailtype og Email. En Email er en mail, som skal sendes til en FluxUser, når denne oprettes i systemet eller ved reset password. Email er associeret til EmailType, da en email naturligvis ikke vil være ens ved oprettelse af FluxUser og ved "Reset password".



6.4.2.2.2 Detailed use cases

Vi har valgt ikke at lave en detailed use case for "Logout", da vi ikke finder dette nødvendigt. "Confirm user" har vi valgt at vise i sammenhæng med "Create user" og "Login".

Create user, Confirm user og Login:

Actor: Admin, User eller Project manager

Main success scenario:

1. Admin starter Create user
2. Admin angiver user-oplysninger(employeeld, firstname, lastname, phone, privateEmail, workEmail, username, password, street, houseNumber, zipCode, city, country, phonePrivate, Usergroup)

3. Systemet gemmer user-oplysninger
4. Systemet sender velkomst-email og link til ny user
5. Systemet præsenterer users

Reset password:

Pre-condition: Actor har modtaget email fra systemet

1. Actor starter Confirm user(url)
2. Actor angiver bruger-oplysninger(tempPassword, password, confirmPassword)
3. Systemet gemmer user(User)
4. Actor starter login(workEmail, password)
5. Systemet præsentere tasks

Vi har anvendt vores prototypes til flow og use casen afsluttes ved at My current tasks vises. Dette kan selvfølgelig ikke implementeres endnu og der vises en tom side (hovedmenuen er synlig og der kan fra menuen navigeres).

Reset password og Login:

Actor: Admin, User, Project manager

Pre-condition: Precondition: FluxUser eksisterer

Mail success scenario:

1. Actor starter resetPassword
2. Actor angiver email(workEmail)
3. Systemet sender email og link til actor
4. Actor starter confirm reset password(url)
5. Actor angiver bruger-oplysninger(oldPassword, password, confirmPassword)
6. Actor starter login
7. Actor angiver login-oplysninger(workEmail, password)

6.4.2.3 Analysis & design

6.4.2.3.1 Database diagram

Der skal tilføjes 2 værdier i tabellen "email": "lost" og "welcome". "lost" skal anvendes ved Reset password og "welcome" ved Create user.



6.4.2.3.2 Design class diagram

Vi har bestemt at der skal implementeres en LoginController (extender ActionHelper), hvor metoderne loginAction() og logoutAction() skal placeres. Metoden checkBootURL skal tjekke om en bruger er logget ind og hvis ikke, skal rollen "guest" tildeles. En view-fil login er også illustreret på diagrammet.

I klassen ProfileController skal metoderne confirmuserAction(), confirmresetpasswordAction() og resetpasswordAction() placeres. Alle actions har ligeledes en tilhørende view-fil af samme navn.

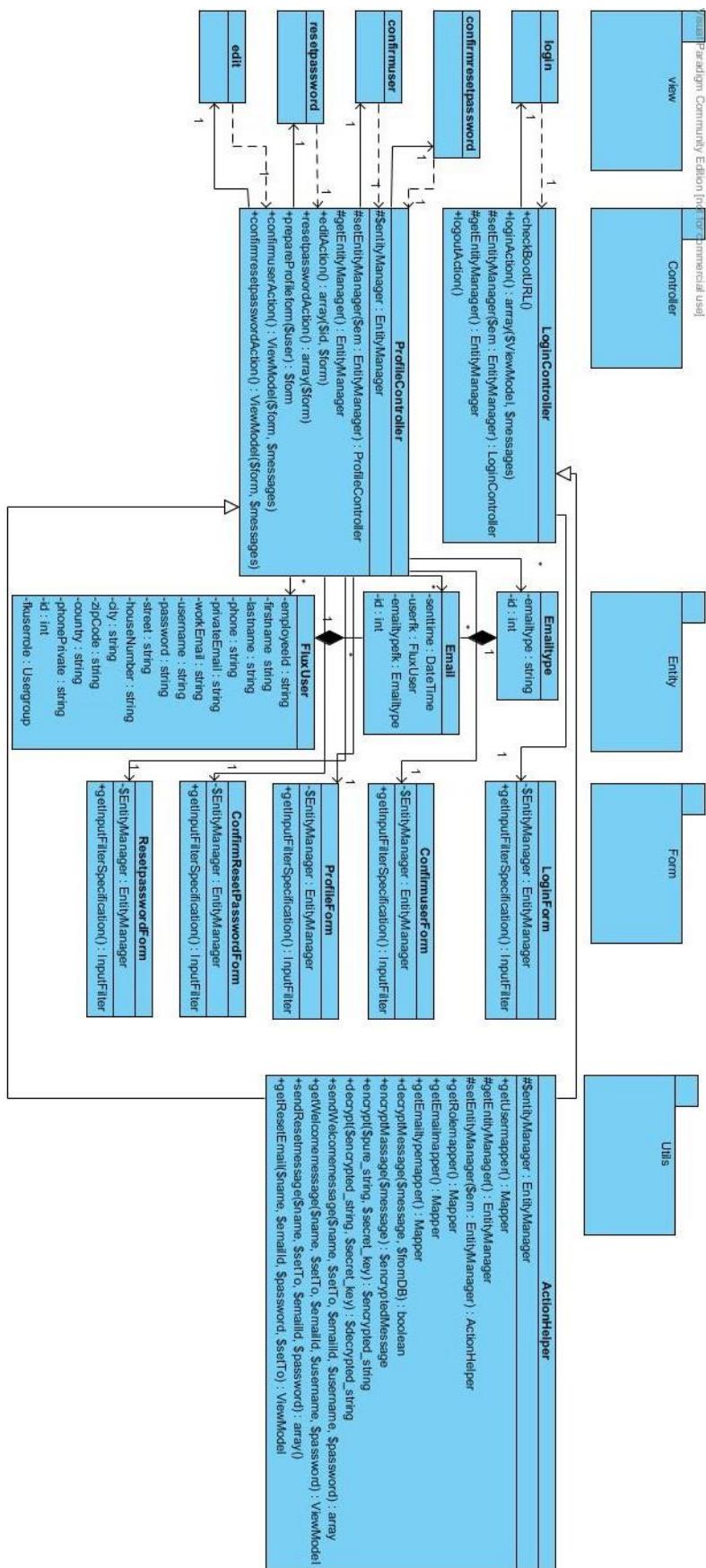
Derudover har vi tilføjet 4 nye forme til diagrammet: LoginForm, ResetpasswordForm, ConfirmuserForm og ConfirmResetPasswordForm.

2 nye model-klasser, som auto-genereres vha. doctrine, er tilføjet. Ved Create user og Reset password skal genereres en email, som gemmes i databasen, samt sendes til brugeren.

Tilføjelser til klassen ActionHelper: Her er placeret metoder til at anvende doctrine-mapperne, til encryption og decryption, til hashing af password og til matching af input og password, samt metoder til generering af emails. Vi vil placere metoderne i ActionHelper, da vi mener at vi kan få brug for at anvende disse metoder i andre controllers senere i projektet.

I ActionHelper skal defines en secret key som skal anvendes til kryptering og dekryptering.

For overskuelighedens skyld er diagrammet lavet i klynge.



6.4.2.4 Implementation

6.4.2.4.1 Implemented components

Klasserne og metoderne i diagrammet ovenfor er implementeret i systemet i modulet Fluxuser. For at undgå fejl i databasen valideres input både vha. javascript på klient-siden og vha. form-validering på serversiden i controllerne. Længden af alle input-data valideres, så det passer med max-værdier i databasen og felter som ikke er nullable i databasen er gjort required.

Passwords valideres vha. regular expression, så kun bogstaver og tal er tilladt. Vi overvejede under implementeringen om styrken af passwordet skulle tjekkes, men dette har vi valgt ikke at gøre. Endvidere kunne vi have lavet en længere secret key, for at gøre krypteringen mere sikker, men dette kan nemt tilføjes, hvis det findes nødvendigt.

Vi har også overvejet om der ved login skulle være en "Husk password-option" eller om vi skulle spærre brugere som forsøger at logge ind mange gange på kort tid, begge dele har vi fravalgt.

Vi har valgt at en user skal ændre password via link max. 24 timer efter at systemet har afsendt emailen. Dette har vi også valgt for at styrke sikkerheden i systemet. Hvis en bruger ikke ændre passwordet indenfor 24 timer, kan brugeren igen requeste en email via Reset password.

Følgende er opdateret i systemet:

addAction() i FluxUserController og editAction() i ProfileController: Før et password gemmes i databasen krypteres det ved brug af hashing. Når en bruger oprettes, sættes "state" til 2. Denne state angiver at brugeren har tilstanden "not confirmed". Brugeren kan ikke logge ind hvis state er 2. Endvidere sendes en email til brugeren med et link, samt login-informationer. Linket indeholder emailens id, som er krypteret. Via linket kan brugeren ændre sit midlertidige password (Confirm user) og logge ind.

Fluxuser-index-siden: Delete-, create- og edit-icons er gemt for brugere som ikke har disse permissions. Fordi Login nu er implementeret kan user-permissions indlæses i systemet.

Forsøger en bruger at skrive en andens bruger-id i URL ved Edit profile, redirectes systemet. Derfor er det kun den bruger som er logget ind som kan ændre sin egen profile.

Disse permissions er tilføjet i databasen:

UrlResource	admin	user	guest	project manager
home/login			X	
login/login			X	
logout/logout	X	X		X
profile/confirmuser			X	
profile/resetpassword			X	
profile/confirmresetpassword			X	

6.4.2.4.2 Screen prints

The screenshot shows a user registration form with the following fields:

- Email*
- Midlertidig password*
- Nyt password*
- Bekræft nyt password*

A green 'Bekræft' button is located at the bottom left.

6.4.2.4.3 Unit testing

I dette sprint har vi udført unit tests for følgende:

Show login-page: Passed

Login with valid login: Passed

Login with invalid login and get error message: Passed

User is being redirected to login page if not logged in: Passed

Logged in user can logout and is being redirectet to login page:

Passed

Only the usergroups with permission to edit an user can access edit user page: Passed

Only the usergroups with permission to add an user can access add user page: Passed

Alle tests er "passed" og authentication og permissions fungerer.

Welcome to FLUXtime

Steen have been created as a user in FLUXtime

Brugernavn: steen@flux.dk

Password:password

Please varify account within 24 hours.

Follow this link to varify.

[Sign up](#)

6.4.2.5 Testing

6.4.2.5.1 Funktionstest & security test

Vi har testet alle use cases som er implementeret i denne iteration, samt at javascript-validering og validering på serveren virker og giver de rigtige fejlmeldelser til brugeren. Vi har testet alle permissions for alle bruger-roller, samt at buttons er gemt for brugere uden rettigheder hertil. Ligeledes har vi testet at en bruger kun kan redigere sin egen profile.

6.4.2.5.2 Usability test

Under mødet med Supeo kunne product owner teste og godkende "Create user", "Edit user", "Delete user" og "Edit profile". Se mødereferat afsnit 10.2.2.

6.4.3 Iteration 2, Projects & Labels

Use cases: "Create project", "Edit project", "Delete project", "Search project", "Create label", "Edit label", "Delete label", "Add user to project", "Remove user from project"

6.4.3.1 Requirements

6.4.3.1.1 Domain model

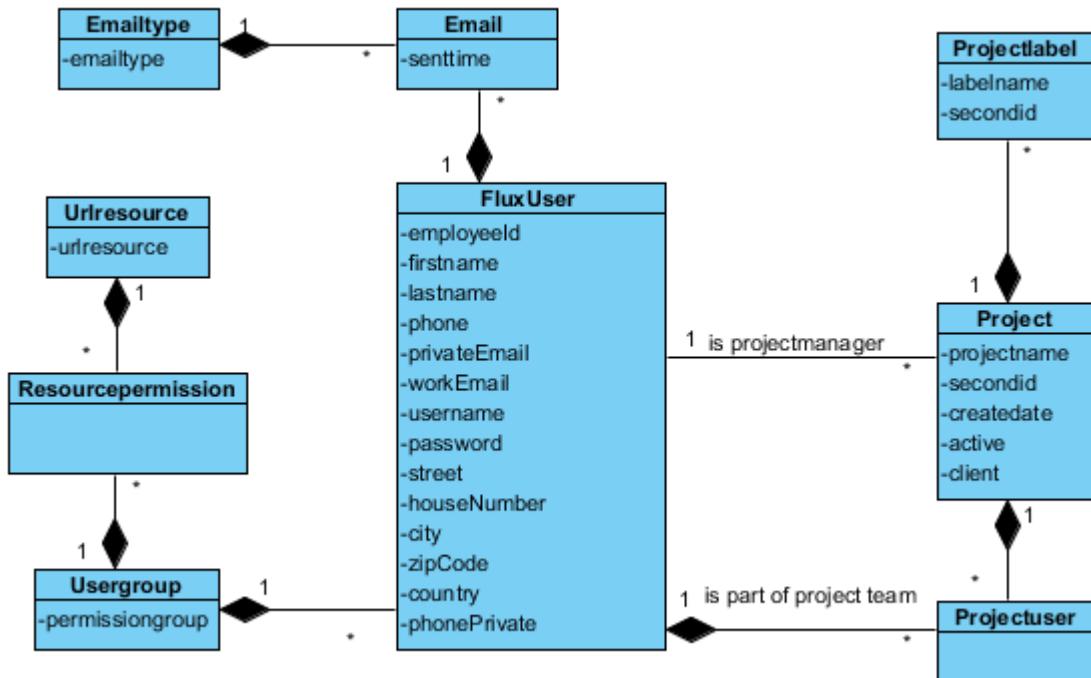
Vi har tilføjet 3 klasser til domæne modellen: Project, Projectlabel og Projectuser. For at kunne finde konceptuelle klasser og attributer, har vi måtte finde inspiration i projektstyringsprogrammet Pivotal

Tracker. Det er Supeo's ønske at systemet skal kunne importere "stories", "projekter" og "epics" fra Pivotal Tracker. Vi benytter Pivotal Tracker til projektstyring (se 10.3) i dette projekt og kender derfor redskabet. Pivotal Tracker benyttes til udviklingsprojekter. Vores system skal kunne benyttes til forskellige typer af projekter og opgaver og derfor er det vigtigt at vi navngiver vores konceptuelle klasser, så de afspejler dette.

Project har en association til FluxUser. Et Projekt har altså ikke nødvendigvis en FluxUser tilknyttet som project manager, men kan have. Vi har givet Project attributten "secondid", da vi allerede nu ved at projekter som importeres fra PT eller andre systemet, skal have et id med fra det eksterne system. Project har ydermere attributten "client". Client er den kunde som projektet udføres for. Client kunne også være en klasse som var associeret med Project, men dette har vi valgt ikke at lave, da use cases til håndtering af clients og contacts er lavt prioriteret på backlog.

Mellem FluxUser og Project har vi sat en associationsklasse Projectuser. En projectuser er et projektteam-medlem.

I PT kan man tilføje "epics" til et projekt. I PT anvendes "epics" som faser eller kategorier til "stories" (opgaver). I PT kan en kategori laves som et objekt (kaldet epic) eller som string (kaldet label). Vi har valgt at kalde vores konceptuelle kategori-klasse Projectlabel og vi har, som ved Project, også tilføjet attributten "secondid", da vi ved at en projectlabel kan være importeret fra et eksternt system.



6.4.3.1.2 Detailed use cases

Vi har valgt ikke at lave detailed use cases for "Delete project", "Delete label" og "Remove user from project", da vi ikke har fundet dette nødvendigt. Vi har heller ikke udformet detailed use case for Edit project da input fra brugeren er de samme som ved Create project. Det samme gælder Edit label.

Create project:

Main success scenario:

1. Admin starter Create project

2. Systemet præsenterer projectmanager-options
3. Admin angiver project-oplysninger(projectname, active, client, Fluxuser)
4. Systemet gemmer project
5. Systemet præsenterer projects

Alternative scenario:

1. Project manager starter Create project
2. Systemet returnere project manager
3. Project manager angiver project-oplysninger(projectname, active, client, Fluxuser)
4. Systemet gemmer project
5. Systemet præsenterer projects

Som use casen ovenfor beskriver, kan en project manager kun tilføje sig selv som project manager når et projekt oprettes.

Create label:

Actor: Project manager eller Admin

Pre-condition: Project eksisterer, systemet præsenterer projects

Main success scenario:

1. Actor starter Edit project(Project)
2. Systemet præsenterer project-oplysninger, projectmanager-options, projectusers og labels
3. Actor starter Create label
4. Actor angiver label-oplysninger(labelname)
5. Systemet gemmer label
6. Systemet præsenterer labels

Add user to project:

Actor: Project manager eller Admin

Pre-condition: Project eksisterer, systemet præsenterer projects

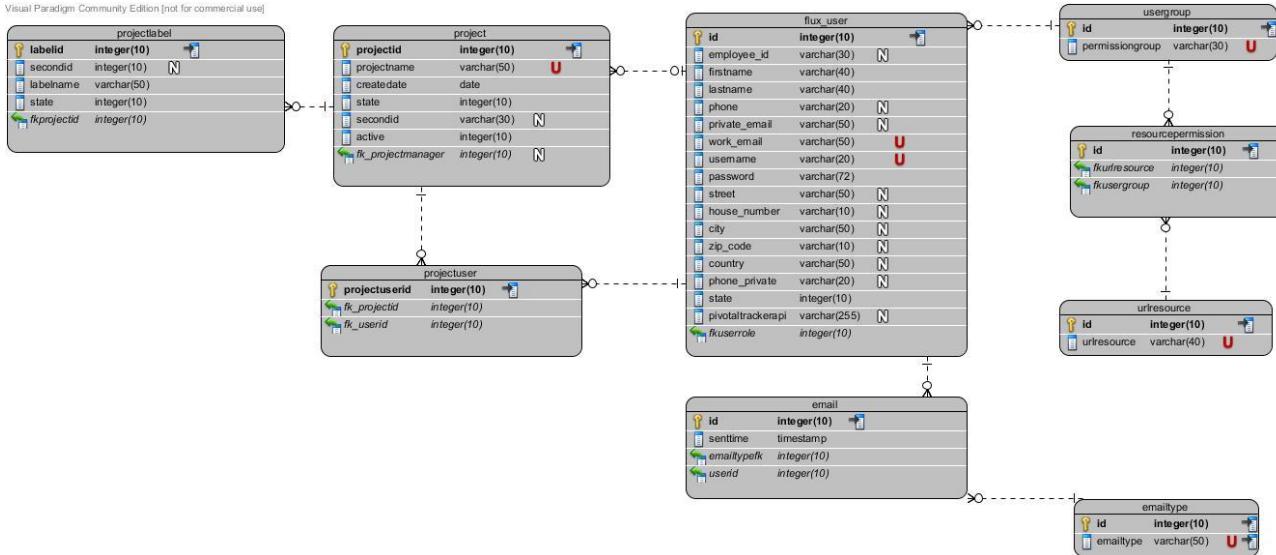
Main success scenario:

1. Actor starter Edit project(Project)
2. Systemet præsenterer project-oplysninger, projectmanager-options, projectusers og labels
3. Actor starter Add user to project
4. Actor angiver projectuser(FluxUser)
5. Systemet gemmer projectuser
6. Systemet præsenterer projectusers

6.4.3.2 Analysis & design

6.4.3.2.1 Database diagram

Tabellerne "project", "projectlabel" og "projectuser" er tilføjet til diagrammet som er lavet vha. domæne modellen. I tabellen "projectuser" skal være en unique constraint(fk_projectid, fk_userid). Ligeledes i "projectlabel" (fkprojectid, labelname).



6.4.3.2.2 Design class diagram

Vi har bestemt at der skal implementeres en ProjectController (extender ActionHelper), hvor metoderne addAction() og editAction() skal placeres. Samt metoderne ajaxconfirmdeleteAction(), ajaxaddmemberAction, ajaxremovememberAction. View-filerne add, edit og index, er også illustreret på diagrammet.

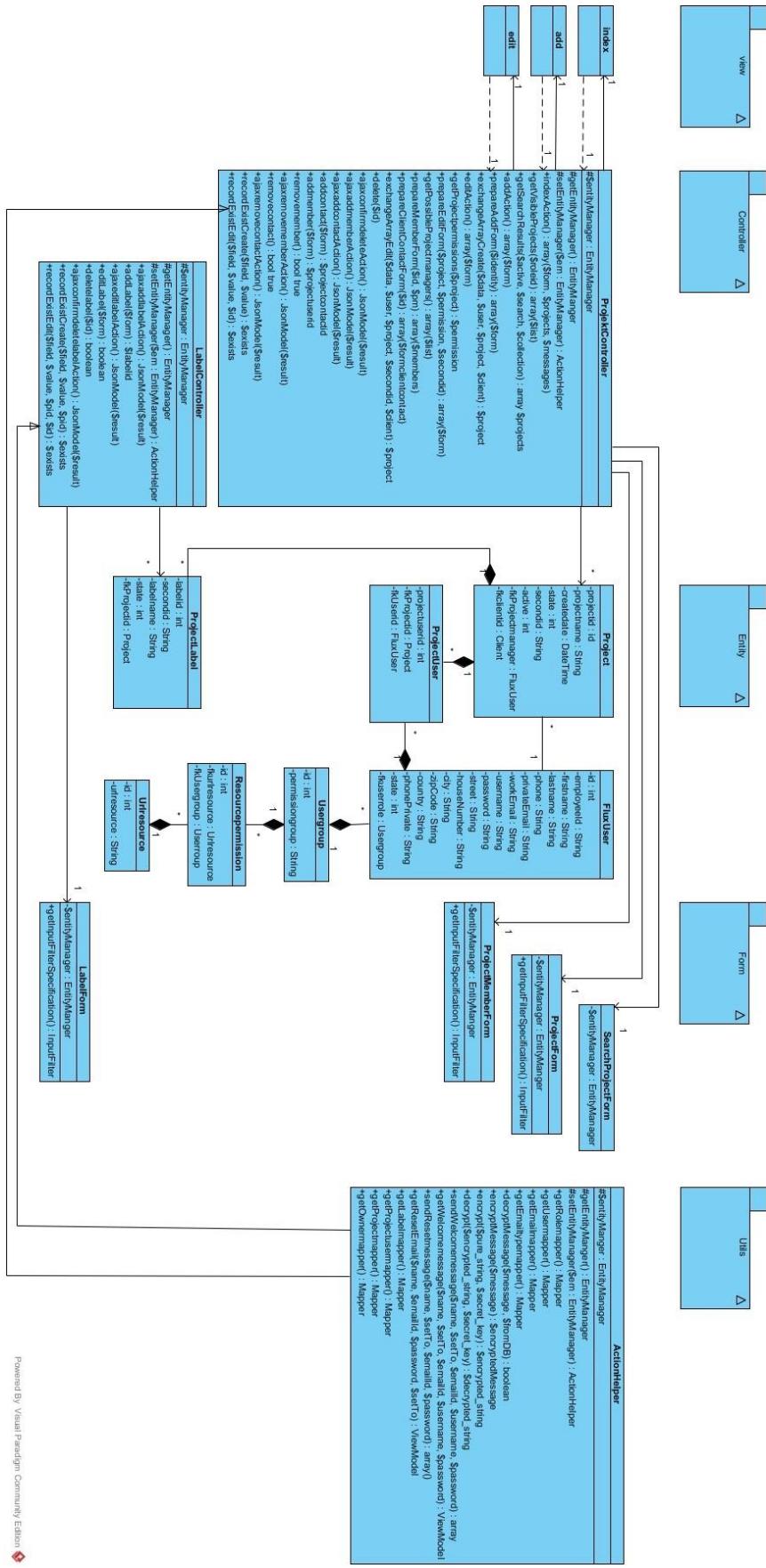
I klassen LabelController skal metoderne ajaxaddlabelAction(), ajaxeditlabelAction() og ajaxconfirmdeleteAction() placeres. Alle actions har ligeledes en form-fil af samme navn.

Derudover har vi tilføjet 4 nye forme til diagrammet: SearchProjectForm, ProjectForm, ProjectMemberForm og LabelForm.

3 nye model-klasser, som auto-genereres vha. doctrine, er tilføjet. Ved edit project kan man oprette project labels og tilføje project members.

Tilføjelser til klassen ActionHelper: Her er placeret metoder til at anvende doctrine-mapperne. Vi vil placere metoderne i ActionHelper, da vi mener at vi kan få brug for at anvende disse metoder i andre controllers senere i projektet.

For overskuelighedens skyld er diagrammet lavet i klynge.



6.4.3.3 Implementation

6.4.3.3.1 Implemented components

Vi har implementeret klasserne som vist ovenfor i diagrammet. Vi har besluttet at en FluxUser med rollen "project manager" kun kan angive sig selv som project manager. Admin kan angive alle fluxusers med rolle "admin" eller "project manager" som project manager.

På index-siden kan admin se alle projekter, men project manager kun kan se de projekter han er project manager for, samt dem hvor han er projectuser. User-roller kan se de projekter hvor de er tilføjet som projectusers.

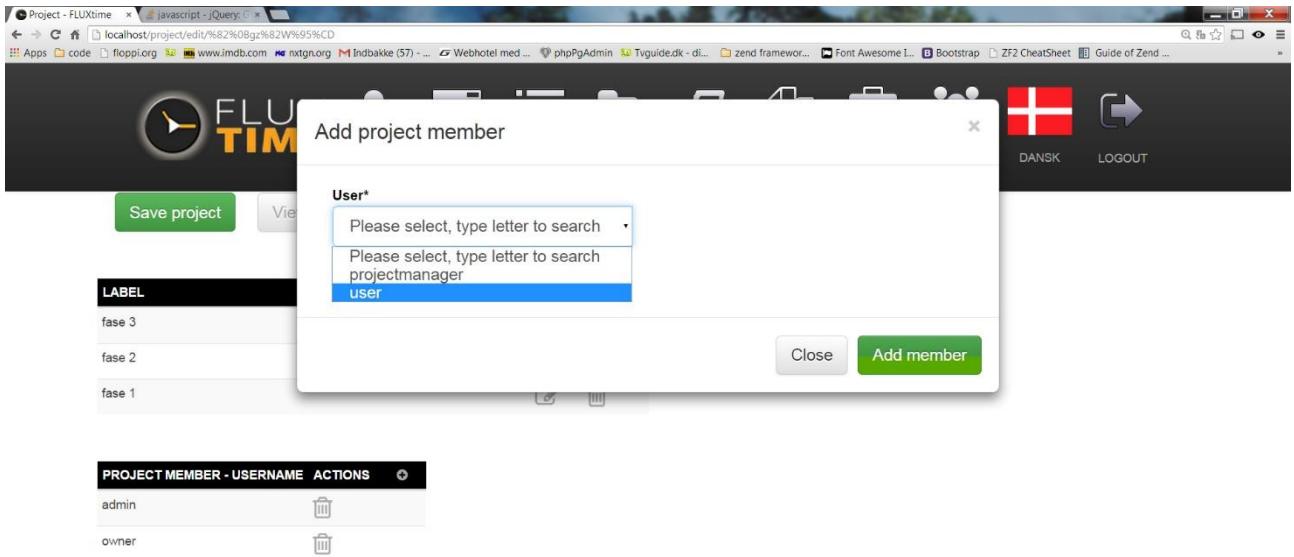
Alle label-actions er implementeret ved at benytte ajax-kald. Ligeledes er addMember() og removeMember().

Disse permissions er tilføjet i databasen:

UrlResource	admin	user	guest	project manager
project/index	X	X		X
project/add	X			X
project/edit	X			X
project/ajaxconfirmdelete	X			
project/ajaxaddmember	X			X
project/ajaxremovemember	X			X
label/ajaxaddlabel	X	X		X
label/ajaxeditlabel	X	X		X
label/ajaxaconfirmdeletelabel	X	X		X

6.4.3.3.2 Screen prints

The screenshot shows the 'PROJECT' application interface. At the top, there's a header bar with a folder icon and the word 'PROJECT'. Below it, the main form for creating a new project has fields for 'PROJECT NAME*' (Næstved Banegård projekt), 'CLIENT' (Bravida Syd), 'ACTIVE' (checked), and 'PROJECT MANAGER' (owner). There are two buttons: 'Save project' (green) and 'View tasks' (grey). Below the form is a table for managing project labels, with rows for 'fase 3', 'fase 2', and 'fase 1', each with edit and delete icons. At the bottom is a table for 'PROJECT MEMBER - USERNAME' with rows for 'projectmanager', 'owner', and 'admin', each with a delete icon.



6.4.3.4 Testing

6.4.3.4.1 Funktionstest

Vi har testet alle use cases som er implementeret i denne iteration, samt at javascript-validering og validering på serveren virker og giver de rigtige fejlmeldelser til brugeren. Vi har testet alle permissions for alle bruger-roller, samt at at buttons er gemt for brugere uden rettigheder hertil.

Vi har testet at user kan se alle de projekter han er tilføjet til (også ved søgning), project manager kan se projekter han er tilføjet til, samt de projekter hvor han er tilføjet som projectmanager (også ved søgning) og admin kan se alle projekter.

Ved oprettelse af et project har vi testet at en project manager kun kan tilføje sig selv om project manager til et projekt. Ligeledes ved Edit project. Admin kan tilføje alle med rollen admin eller project manager. Er en project manager allerede tilføjet (ved Edit project), bliver denne altid hentet med ud, også selv om denne fluxuser er har "state 0". Dette var vi gjort for at bevare historiske data. Slettes en fluxuser, fjernes denne bruger altså ikke fra projekter hvor han er tilføjet som project manager (ligeledes hvor han er tilføjet som project member). Dette gives brugeren en besked om ved sletning.

Vi har testet at et projekt ikke kan slettes hvis der er tilføjet projectusers (members) eller labels.

Vi har testet at members og labels med samme labelname kun kan tilføjes en gang til samme projekt.

6.4.3.4.2 Usability test

Product owner har testet/godkendt use cases. Desværre, som beskrevet i process-afsnittet, var vores branched ikke ordentligt merged og derfor var der flere ting som ikke fungerede i programmet (se møderef. 10.2.3).

6.4.4 Iteration 4, Tasks & Import

Use cases: Create task, Edit task, Delete task, Search task, Add taskowner, Remove taskowner, Import from Pivotal tracker

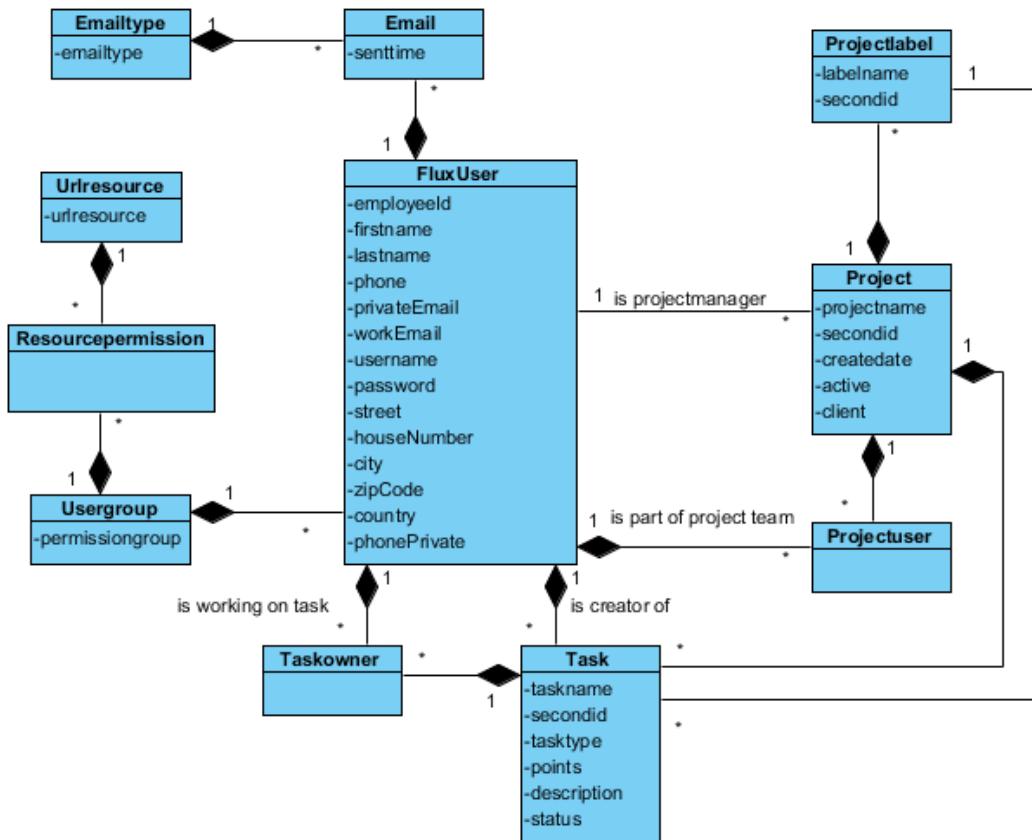
6.4.4.1 Requirements

6.4.4.1.1 Domain model

Vi har tilføjet 2 klasser til domæne modellen: Task og Taskowner. For at kunne finde konceptuelle klasser og attributer, har vi igen måtte finde inspiration i projektstyringsprogrammet Pivotal Tracker. Vi har bl.a. givet den konceptuelle klasse Task attributterne tasktype, status og points, som stories i PT også har. Points er en bedømmelse af varighed man kan tilføje til opgaven. Status kan fx være "started" eller "finished" mv.

Task kan ikke eksistere uden et projekt, men er ikke nødvendigvis knyttet til en label. Taskowners er de Fluxusers som skal udføre en task. Vi har givet Task attributten "secondid", da vi allerede nu ved at projekter som importeres fra PT eller andre systemet, skal have et id med fra det eksterne system.

En Task kan ikke eksistere uden en FluxUser, som er "creator" af Task.



6.4.4.1.2 Detailed use cases

Vi har kun lavet en detailed use case for Create task:

Actor: Project manager, Admin eller User

Pre-condition: FluxUser og Project eksisterer, Actor angiver project, systemet præsentere tasks

Main success scenario:

1. Actor starter Create task()
2. Actor angiver task-oplysninger(taskname, tasktype, points, description)

3. Systemet gemmer task

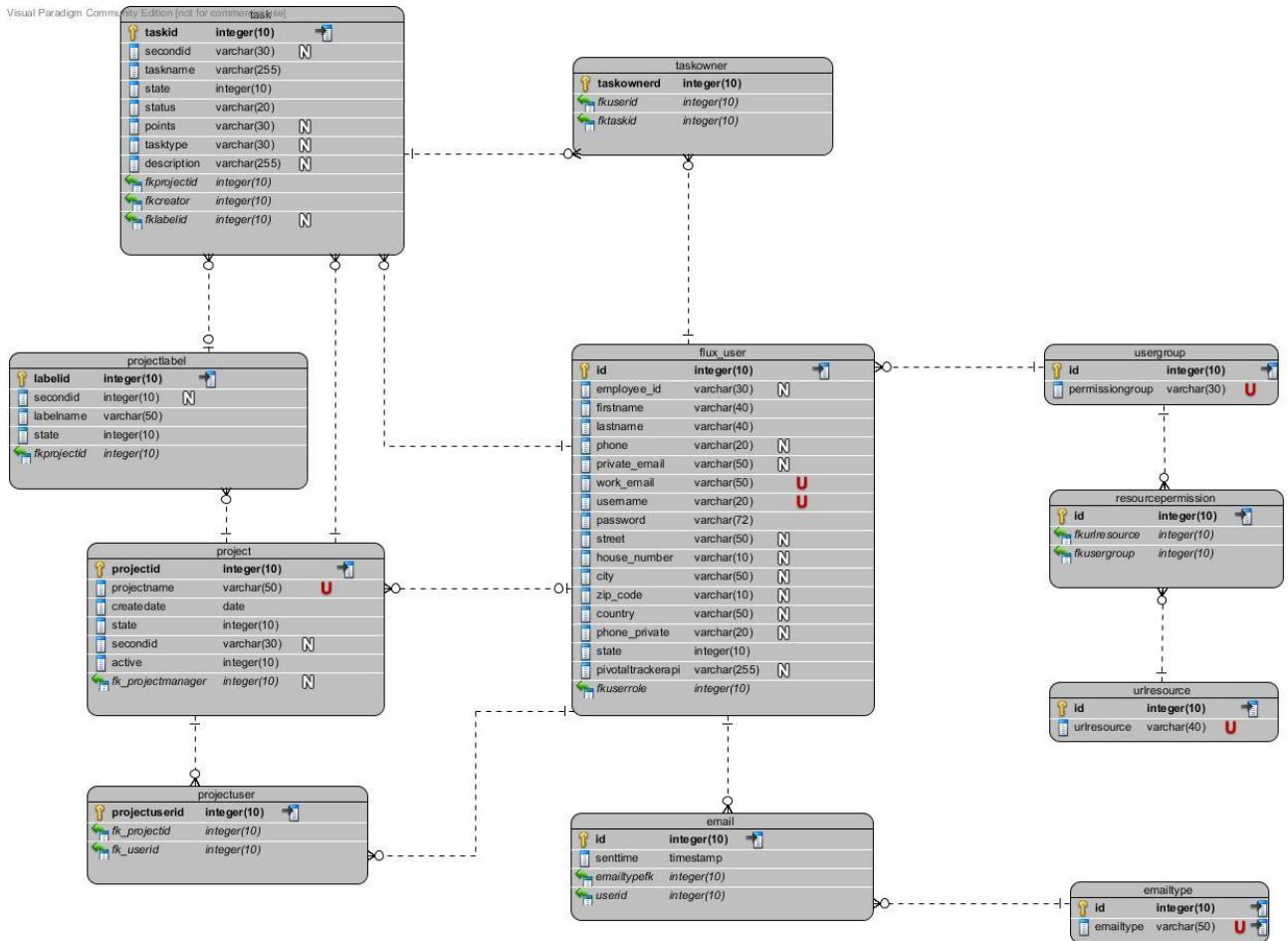
4. Systemet præsenterer task og taskowner-options

Ved Edit task kan actor også angive status.

6.4.4.2 Analysis & design

6.4.4.2.1 Database diagram

Vi har besluttet at der gerne må oprettes flere ens tasks – dvs. tasks som er tilknyttet samme projekt og har samme taskname. Hvis vi relaterer til dette projekt kunne vi oprette 2 labels f.eks "Construction sprint 1" og "Construction sprint 2". Til hver label kan tilknyttes en Task med taskname "Unit testing". Da label er nullable, er den eneste løsning at flere "ens" tasks må oprettes. For at importere fra PT, skal vi anvende en API token for hver bruger. Derfor skal denne tilføjes i databasen.



6.4.4.2.2 Design class diagram

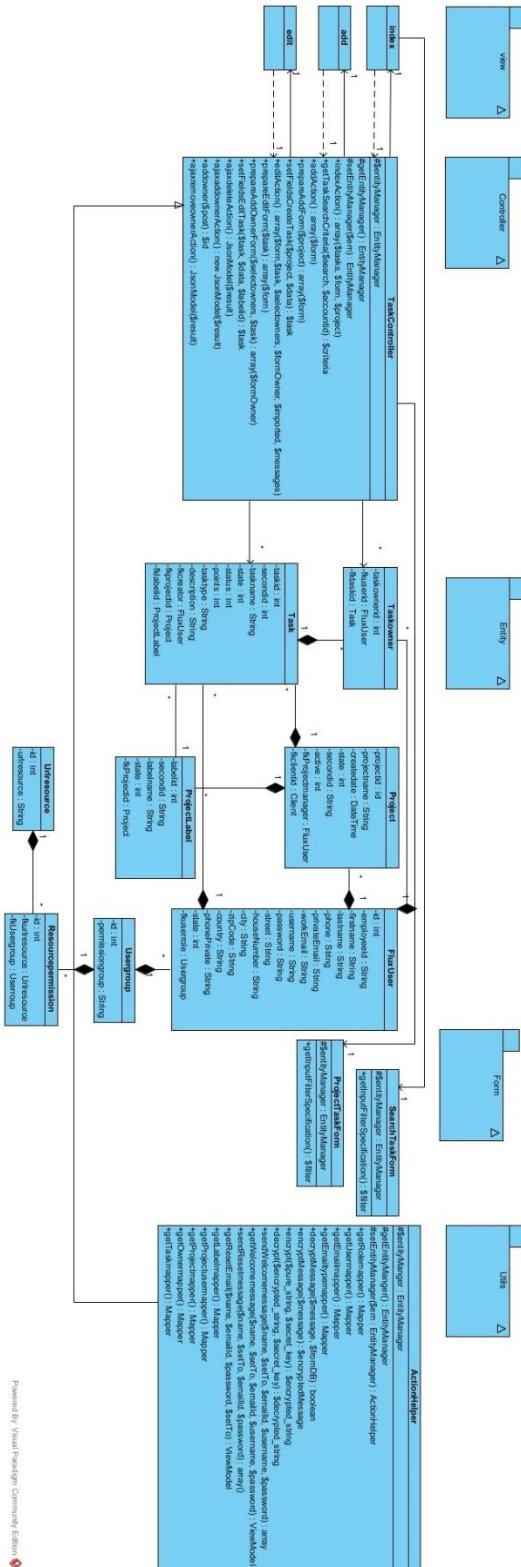
Vi har bestemt at der skal implementeres en TaskController (extender ActionHelper), hvor metoderne addAction() og editAction() skal placeres. Samt metoderne ajaxdeleteAction(), ajaxaddownerAction, ajaxremoveownerAction(). View-filerne add, edit og index, er også illustreret på diagrammet.

Derudover har vi tilføjet 2 nye forme til diagrammet: SearchTaskForm og ProjectTaskForm.

2 nye model-klasser, som auto-genereres vha. doctrine, er tilføjet.

Tilføjelser til klassen ActionHelper: Her er placeret metoder til at anvende doctrine-mapperne. Vi vil placere metoderne i ActionHelper, da vi mener at vi kan få brug for at anvende disse metoder i andre controllers senere i projektet.

For overskuelighedens skyld er diagrammet lavet i klynge.



Ud over disse classer skal vi implementere 2 klasser i Utils: 1 som importere fra PT og en som konvertere til FLUXtime objekter. Vi vil beskrive klasserne nærmere under "Implemented components".

6.4.4.3 *Implementation*

6.4.4.3.1 Implemented components

Vi har implementeret klasserne som vist i diagrammen ovenfor. Derudover har vi implementeret disse 2 klasser:

PivotalTracker:

Denne klasse bruger til at importerer data via pivotaltrackers api til vores system. Man kan hente projekter, stories, labels/epics og users ind i FLUXtime. Vi bruger curl til at kommunikerer med Pivotal Trackers api.

PivotalToProject:

Denne klasse bruges til at konverterer de importerede projecter, stories, labels/epics og users fra Pivotal Tracker til FLUXtimes database. F.eks. createProject som konverterer et Pivotal Tracker projekt. Hvis det findes i vores system, vil det blive opdateret hvis der er nogle nye emner og hvis det ikke findes så oprette det på ny. Der hentes både projectusers, labels og tasks(stories).

Vi har anvendt cUrl eller libcurl som er et library lavet af Daniel Stenberg, som bliver brugt til at forbinde og kommunikere med mange forskellige typer servere med mange forskellige protokoller.⁴⁵ cUrl er også det der bliver brugt i Pivotal Trackers API eksempler.

For at kunne importere fra Pivotal Tracker er man nødt til at sende en token med i sit request. Denne skal tilføjes under Edit eller Create user, men dette har vi ikke implementeret endnu. Vi har testet ved at skrive token direkte i databasen.

Ved Zend Framework 2 er der 2 måder man kan lave Cronjobs på. Den ene indbefatter at man laver en URL-route kun til cronjobs og den anden metode kræver at man laver en console-route. Den første metode er den nemmeste, da den benytter den samme fremgangs måde som når man laver nye sider i Zend Framework 2. Dette gør desværre også at alle har adgang til ens cronjobs, selvfølgelig kan man blokke dette med ACL, men så kræver det jo at man logger ind med en bruger først, inden man kan køre et cronjob. Den anden metode er meget smartere, da det kun er folk med konsol adgang til din applikation der kan køre dine cronjobs. Dette vil typisk være serveren og udviklerne, og hvis andre folk har konsol adgang til din applikation, så har de alligevel adgang til serveren og så har de adgang til alt hvad man har der ligger på den.

Disse permissions er tilføjet i databasen:

UrlResource	admin	user	guest	project manager
task/ajaxaddowner	X	X		X
task/ajaxremoveowner	X	X		X
task/index	X	X		X
task/add	X	X		X
task/edit	X	X		X
task/ajaxdelete	X	X		X
cronjob/crontasks			X	

⁴⁵ <http://php.net/manual/en/intro.curl.php>

6.4.4.3.2 Screen prints

Næstved Banegård projekt

This task is not visible in any external systems (created in FLUXtime)

Task*	Label	Points	Description	Task type
Renovering af spor 2	Please select...	3		Feature

Creator
owner

Status* Creator

started owner

Save task **Return to tasks** **View project**

USERNAME	ACTIONS
admin	
user	
owner	

6.4.4.4 Testing

6.4.4.4.1 Funktionstest

Vi har testet validering, samt at en taskowner ikke kan tilføjes til en task flere gange. Da alle usergroups har alle rettigheder ang. tasks, var der ikke så meget vi skulle teste i denne iteration.

Vi har testet det cron-job som importere elementer fra pivotal tracker. Metoden er meget tung og burde rent performance-mæssigt optimeres. Vi har testet at alle projekter, labels, tasks, projectusers og taskowners hentes fra PT og opdateres.

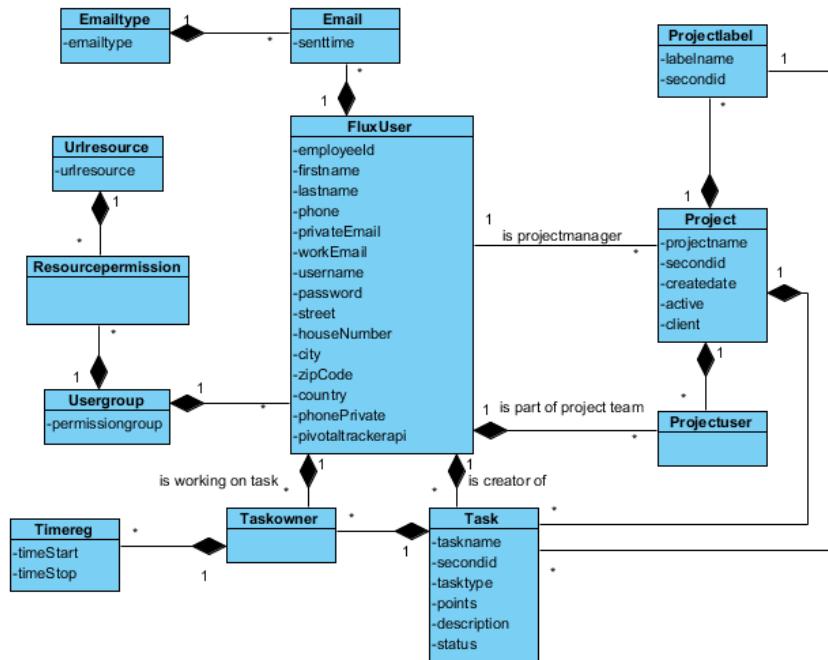
6.4.5 Iteration 5, Time registration & Reports

Use cases: "Start timeregistration", "Stop timeregistration", "Finish task", "Search timeregistration", "Add timeregistration", "Edit timeregistration", "Delete timeregistration", "Generate project report" og "Generate user report"

6.4.5.1 Requirements

6.4.5.1.1 Domain model

Klassen Timereg er tilføjet til domæne modellen. Timereg klassen skal bruges til tidsregistrering. Der bliver oprettet en ny tidsregistrering for gang man har startet og stoppet med at arbejde på en Task. Timereg kan ikke eksistere uden en Taskowner. En Taskowner henviser til Task og FluxUser.



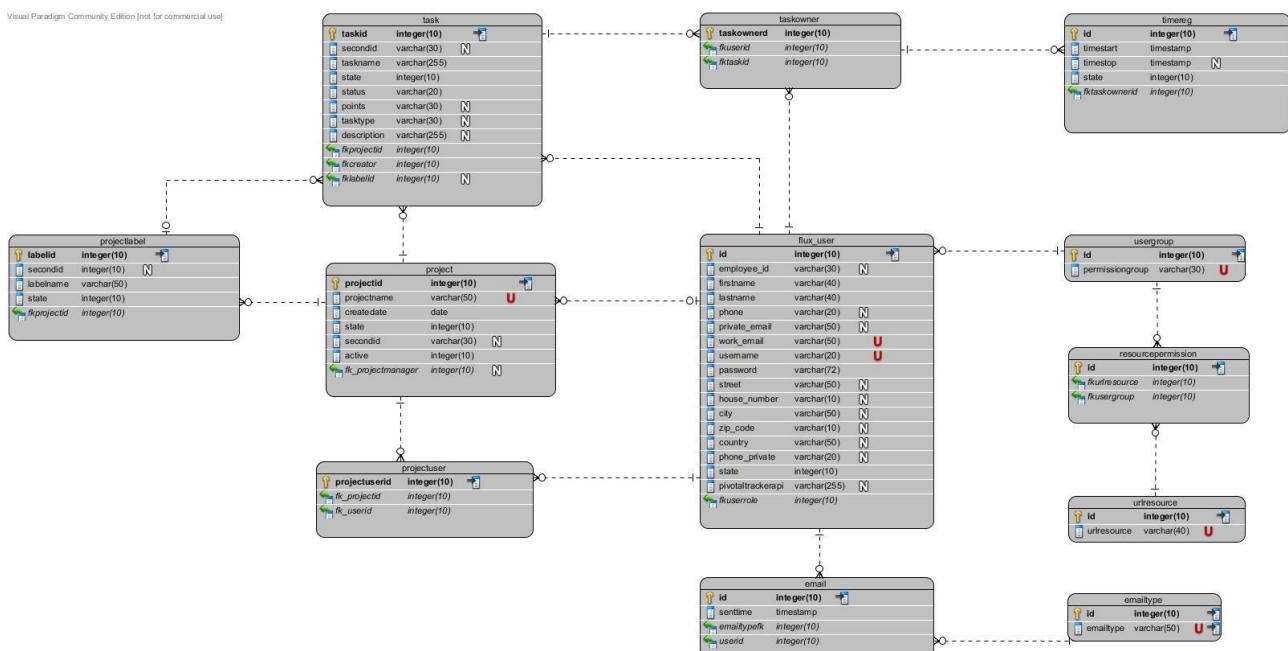
6.4.5.1.2 Detailed use cases

Vi har beslutte ikke at skrive detaliet use cases i denne iteration, da vi ikke finder det nødvendigt. Der sker ikke så meget data-udveksling i lange sekvenser mellem system og bruger og derfor synes vi ikke at disse skulle udformes.

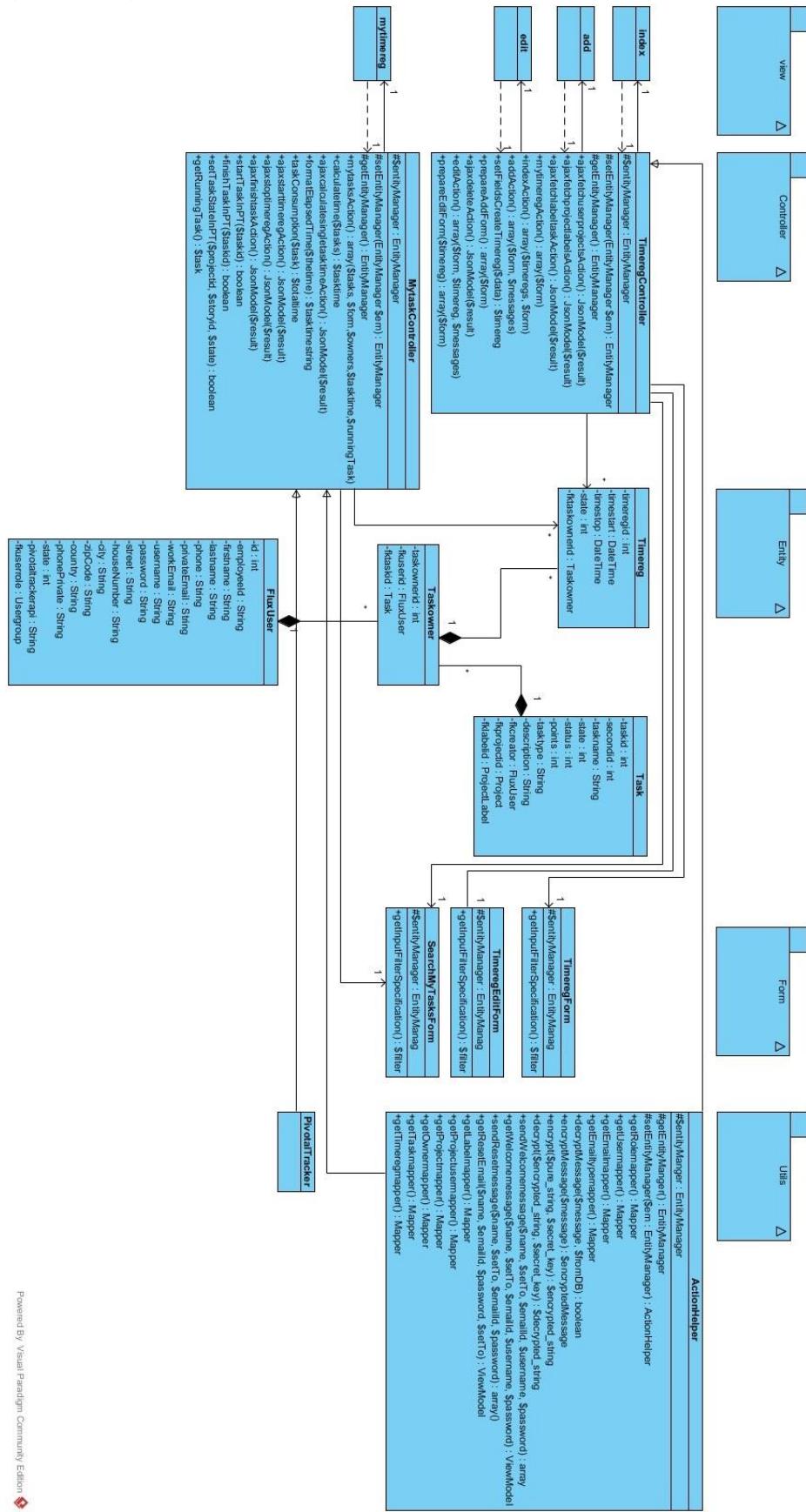
Add timereg, Edit timereg og Delete timereg er lavet for at en admin-bruger kan rette en timereg hvis en bruger har registreret tid på en opgave og f.eks. har glemt at stoppe opgaven el. andet.

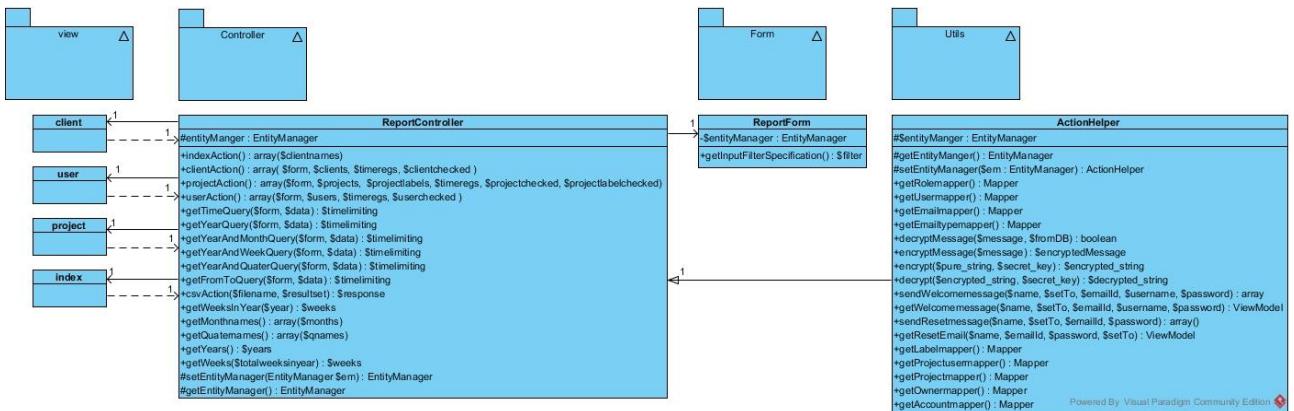
6.4.5.2 Analysis & design

6.4.5.2.1 Database diagram



6.4.5.2.2 Design class diagram





6.4.5.3 Implementation

6.4.5.3.1 Implemented components

Vi har implementeret klasserne som vist ovenfor i diagrammerne.

En af hovedemnerne i bagloggen er rapporter. Det skal være muligt at udfærdige forskellige rapporter hvorefter man skal kunne få gemt rapporten i enten en pdf eller en CSV-fil. Supeo foretrak at det skulle være en CSV-fil.

Efter noget research viste det sig at der var to måder at håndterer dette på. Enten kunne man bruge et tillægsmodul, som PHPExcel. Det er et fuldstændigt færdigt sæt af klasser parat til at udfærdige excel-filer med. Her kan man håndterer de fleste ting man kan lave i excel så det er derfor rigtig stærkt, men tilgengæld fylder det rigtig meget. Den anden løsninger var at man kunne lave sin egen fil med en specielheader så Excel kan importerer den. Fordelen ved dette er det er hurtigt at lave, simpelt og rækker til det vi skal bruge det til. Man kan ikke ændre i rækkestørrelser og andet sjov for at designe dokumentet. Vi har dog alligevel foretrukket den sidste del, på grund af enkeltheden og at vi ikke skal trækkes med den kæmpe samling af klasser. Alt funktionalitet der vedrører rapporter er implementeret i **ReportController**.

Vi har implementeret en side "My log". På denne side kan brugeren se sine egne timeregistreringer, men ikke administrerer dem.

På siden log har admin adgang til alle timeregistrations og han kan her oprette, editere og slette.

Startsiden (efter login) i systemet er siden "My tasks". Her vises de opgaver hvor man er tilknyttet som owner. Det er kun opgaver som er planlagte eller i gang som vises (ikke færdige opgaver). Det er her tidsregistrering foregår. Brugeren har mulighed for at starte og stoppe tidsregistrering, samt færdiggøre (finish) en task.

Vi har også implementeret at admin kan tilføje en PT API token til en fluxuser. Dette gør at brugeren kan starte en opgave i PT gennem FLUXtime, samt "finish" en task i PT. For at kunne snakke med Pivotal Tracker er man nemlig nødt til at sende en token med i sit request. Man kan hente denne token via Pivotal Tracker's API. Dette gøres ved at sende brugerens brugernavn og password i et request og så hente brugerens token via det respons man får. Man kan evt. også bruge brugeren brugernavn og password i hvert request man sender, men da det er nemmest at skifte en token end et brugernavn og password, så er det god skik at bruge en token i sit request.

Vi har opdateret logoutAction(). Hvis en tidsregistrering ikke er stoppet, stoppes den og brugeren modtager en email herom. Ligeledes har vi implementeret et cron-job som kører hver dag kl 24.00 og stopper alle tidsregistreringer, samt sender mails til brugerne herom.

Disse permissions er tilføjet i databasen:

UrlResource	admin	user	guest	project manager
report/index	X			X
report/project	X			X
report/user	X			
mytask/mytasks	X	X		X
mytask/ajaxfinishtask	X	X		X
mytask/ajaxcalculatesingletasktime	X	X		X
mytask/startTaskInPT	X	X		X
mytask/finishTaskInPT	X	X		X
mytask/ajaxstoptimereg	X	X		X
mytask/ajaxstarttimereg	X	X		X
timereg/mytimereg	X	X		X
timereg/index	X			
timereg/add	X			
timereg/ajaxdelete	X			
cronjob/midnightmail			X	
timereg/ajaxedit	X			
timereg/ajaxfetchlabeltask	X			
fluxuser/deleteApiToken	X			
timereg/ajaxfetchprojectlabels	X			
fluxuser/fetchApiToken	X			
timereg/ajaxfetchuserprojects	X			

6.4.5.3.2 Screen prints

The screenshot shows a web application titled "USER REPORT". At the top, there's a header with a back button. Below it, a section titled "Please choose users" contains filters for "Limit choices*", "Year*", "Quarter*", "Month*", "Week*", and date ranges "From date" and "To date". A dropdown menu for "Limit choices*" is set to "week". The "Week*" dropdown is set to "1". The "From date" is "13-01-2015" and the "To date" is "13-01-2015". There's also a checkbox labeled "ALL" which is checked. Below this, a table lists users: "owner", "user", "admin", and "projectmanager", each with a checked checkbox next to it. At the bottom, a table displays task history with columns: TASKID, TASKNAME, CLIENT, PROJECT, OWNER, LABEL, PIVOTALID, START, STOP, DURATION, and OWNER. One row is shown: Task ID 1, Task Name "Udskiftning spor 1", Client "Bravida Syd", Project "Næstved Banegård projekt", Owner "admin", Label "fase 1", Start "13-01-2015 23:28 (57)", Stop "13-01-2015 23:51 (34)", Duration "0:22:37", and Owner "admin". At the very bottom, there are buttons for "Create report", "Create csv-report", and "Return to reports".

FLUX TIME

FETCH PIVOTAL TRACKER API TOKEN

Insert Pivotal Tracker Username and Password.

Username:

Password:

Close **Fetch**

CREATE NEW TIME REGISTRATION

Employee ID:

Email*:

Zip code:

Pivotal Tracker API Token:

Save user

Insert Pivotal Tracker API Token

MY LOG

2015-01-13 00:15 | 2015-01-14 00:15 |

TASK NAME	START	STOP	PROJECT	LABEL	TIME
Udskiftning spor 1	13-01-2015 23:28 (57)	13-01-2015 23:51 (34)	Næstved Banegård projekt	fase 1	0:22:37
Udskiftning spor 1	14-01-2015 00:08 (10)	14-01-2015 00:08 (12)	Næstved Banegård projekt	fase 1	0:0:2
Udskiftning spor 1	14-01-2015 00:09 (43)	14-01-2015 00:10 (14)	Næstved Banegård projekt	fase 1	0:0:31
Udskiftning spor 1	14-01-2015 00:10 (36)	14-01-2015 00:11 (40)	Næstved Banegård projekt	fase 1	0:1:14
Udskiftning spor 1	14-01-2015 00:11 (02)	14-01-2015 00:11 (46)	Næstved Banegård projekt	fase 1	0:0:44
Renovering af spor 2	14-01-2015 00:08 (01)	14-01-2015 00:09 (12)	Næstved Banegård projekt		0:1:11
Renovering af spor 2	14-01-2015 00:09 (49)	14-01-2015 00:09 (51)	Næstved Banegård projekt		0:0:2
Renovering af spor 2	14-01-2015 00:10 (46)	14-01-2015 00:10 (58)	Næstved Banegård projekt		0:0:12
Renovering af facade	14-01-2015 00:07 (55)	14-01-2015 00:09 (20)	Næstved Banegård projekt		0:1:25
Renovering af facade	14-01-2015 00:08 (07)	14-01-2015 00:09 (25)	Næstved Banegård projekt		0:1:18
Renovering af facade	14-01-2015 00:09 (46)	14-01-2015 00:10 (10)	Næstved Banegård projekt		0:0:24
Renovering af facade	14-01-2015 00:10 (41)	14-01-2015 00:11 (07)	Næstved Banegård projekt		0:0:26
Renovering af facade	14-01-2015 00:11 (01)	14-01-2015 00:11 (25)	Næstved Banegård projekt		0:0:24
Renovering af facade	14-01-2015 00:11 (31)	14-01-2015 00:11 (36)	Næstved Banegård projekt		0:0:5

CREATE NEW TIME REGISTRATION

User*: projectmanager

Active project*: Næstved Banegård projekt

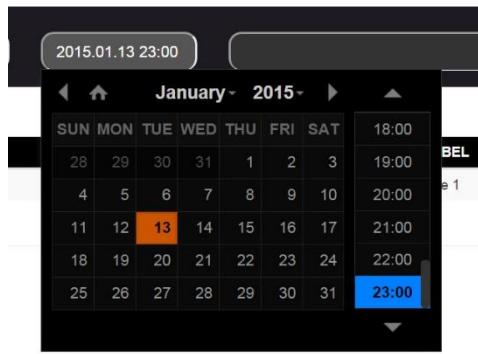
Label*: fase 1

Task*: Udskiftning spor 1

From*: 2015-01-04 23:45

To*: 2015-01-13 23:00

Save time registration



6.4.5.4 Testing

6.4.5.4.1 Funktionstest

Vi har testet use cases og validering, samt udtræk og permissions, da vi ikke ønsker at andre end admin-rollen kan oprette, redigere og slette timeregistrations.

Vi har også testet vores kommunikation med pivotal tracker, samt vores cron-job som stopper alle tidsregistreringer og sender mails til brugerne.

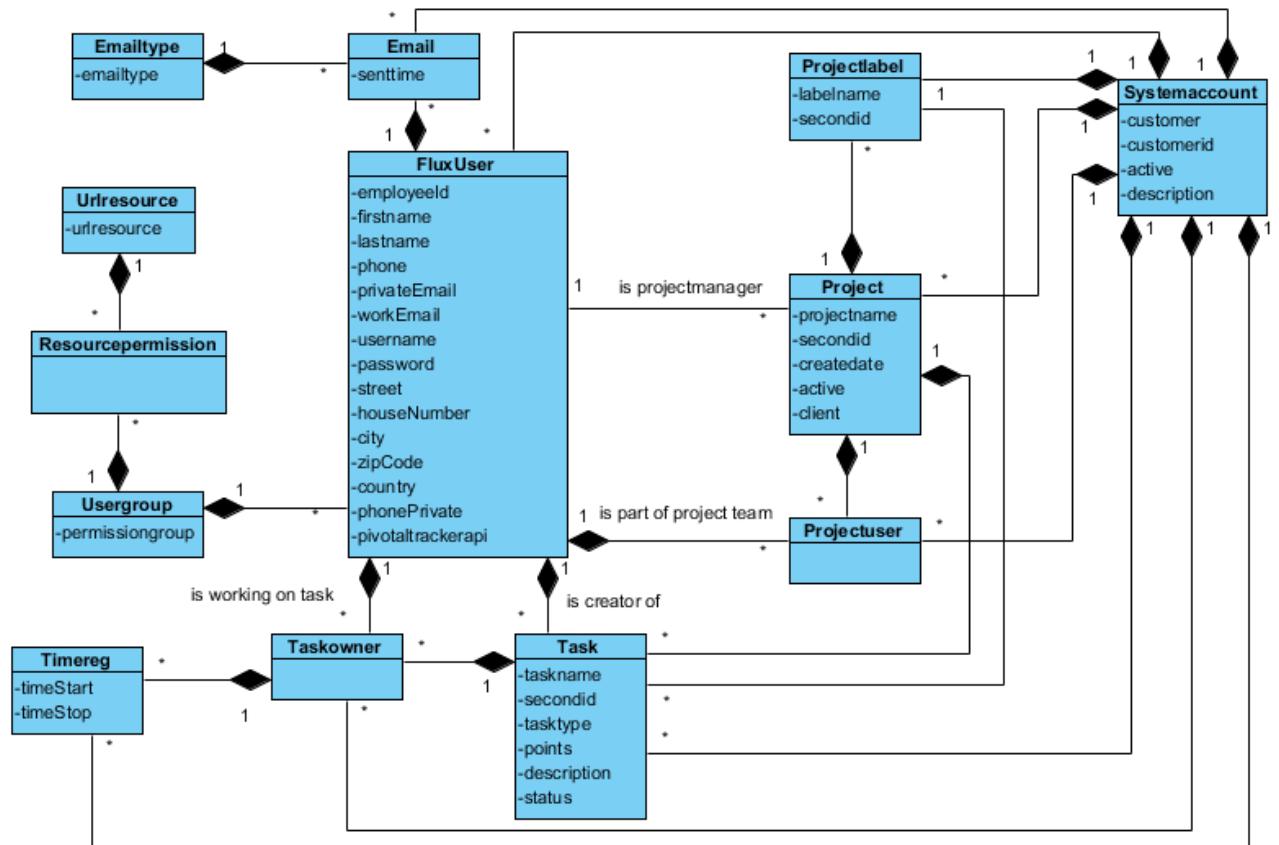
6.4.6 Iteration 6, System accounts

Use cases: Create account, Edit account, Delete account og Search account

6.4.6.1 Requirements

6.4.6.1.1 Domain model

Vi blev opmærksomme på at flere virksomheder kunne anvende systemet og samme database. Derfor må hver virksomhed have en System account og alle virksomhedens data er tilknyttet. Klassen har fået attributten active, da en account evt. først skal være aktiv fra en vis dato. Vi har givet klassen attributten description, så det er muligt at tilføje en kommentar om f.eks licens-dato mv.



6.4.6.1.2 Detailed use cases

Vi har valgt at lave en detailed use case for Create system account:

Actor: Ny usergroup er identificeret - System owner

Main success scenario:

1. Actor starter Create system account(url)
2. Actor angiver adminuser og account-oplysninger(employeed, firstname, lastname, phone, privateEmail, workEmail, username, password, street, houseNumber, zipCode, city, country, phonePrivate, Usergroup, customer, customerid, active, description)
3. Systemet genererer account-oplysninger
4. Systemet gemmer adminuser-oplysninger
5. Systemet sender velkomst-email og link til ny admin-user
6. Systemet præsenterer account

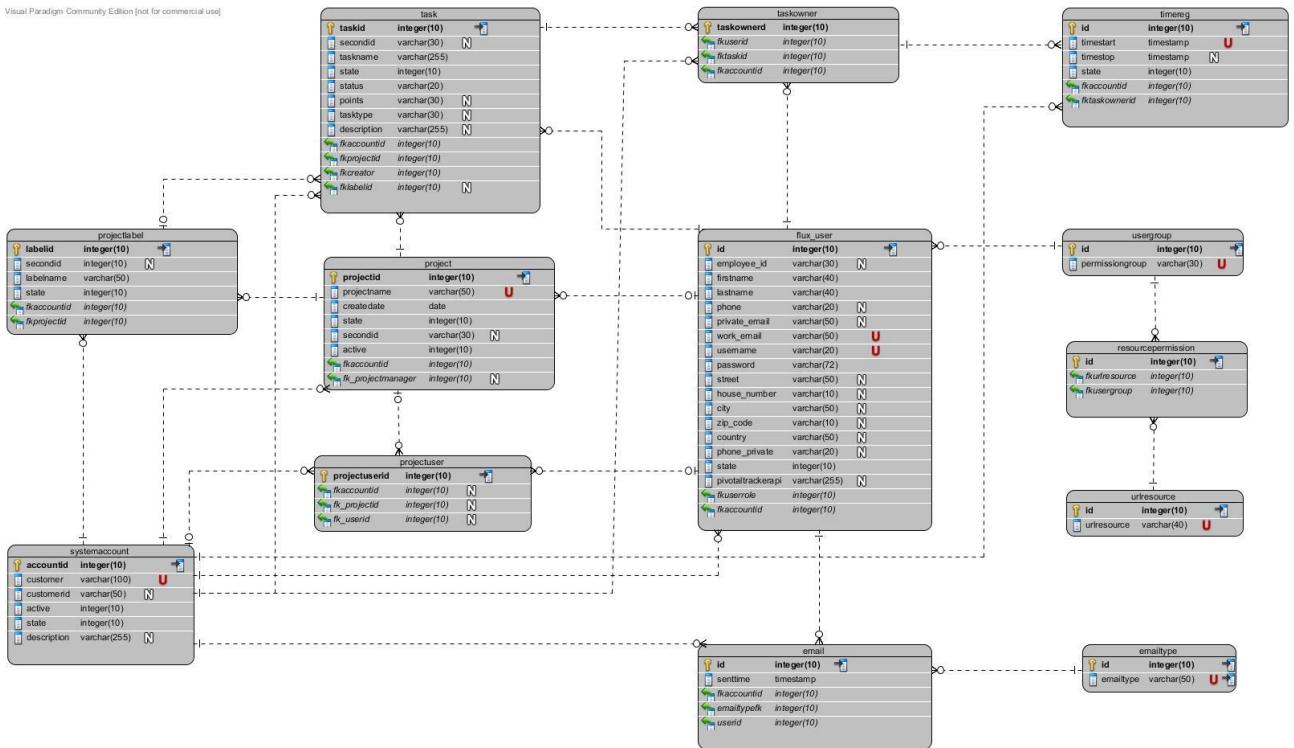
Ved edit account kan gives disse input fra brugeren(customer, customerid, active, description).

6.4.6.2 Analysis & design

6.4.6.2.1 Database diagram

Som diagrammet viser har næsten alle tabeller fået en foreign key til System account. Nogle tabeller behøver ikke denne key, da de i forvejen er knyttet til Systemaccount gennem andre tabeller, men da man evt. kan komme ud for at en account skal fjernes fuldstændigt fra databasen, er det meget nyttigt at alle relevante tabeller indeholder denne key.

Usergroup "system owner" skal gemmes fast i databasen. Ligeledes skal systemOwnerAccount i systemaccount.



6.4.6.2.2 Design class diagram

Vi har bestemt at der skal implementeres en AccountController (extender ActionHelper), hvor metoderne addAction() skal placeres. Samt metoderne ajaxdeleteAction(), og ajaxeditAction(). View-filerne add og index, er også illustreret på diagrammet.

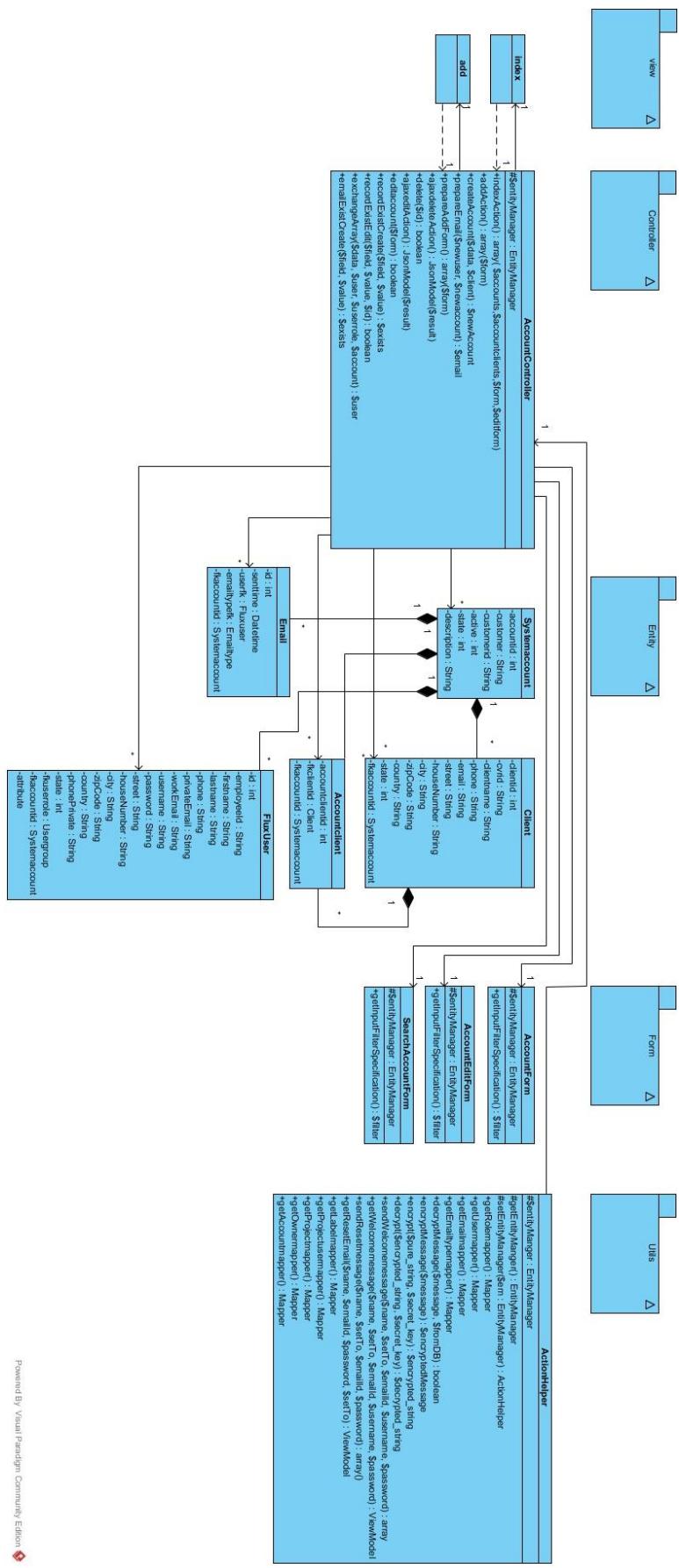
Derudover har vi tilføjet 3 nye forme til diagrammet: SearchAccountForm, AccountForm og AccountEditForm.

1 ny model-klasse, som auto-genereres vha. doctrine, er tilføjet.

Tilføjelser til klassen ActionHelper: Her er placeret metoder til at anvende doctrine-mapperne. Vi vil placere metoderne i ActionHelper, da vi mener at vi kan få brug for at anvende disse metoder i andre controllers senere i projektet.

For overskuelighedens skyld er diagrammet lavet i klynge.

Alle design klasse diagrammer er opdateret, se 10.5.



Disse permissions er tilføjet i databasen:

UrlResource	admin	user	guest	System owner	project manager
account/index				X	
account/add				X	
account/ajaxdelete				X	
account/ajaxedit				X	

6.4.6.3 *Implementation*

6.4.6.3.1 Implemented components

I denne iteration har vi foretaget mange ændringer igennem hele systemet:

Usergroup "system owner" er tilføjet i databasen. System owner er den eneste som har rettigheder i use cases vedr. Systemaccounts. Vi har ikke implementeret nogle account-button i menuen, men i stedet skriver admin direkte i url (account).

Alt hvad der oprettes i database bliver tilføjet en foreing key til en systemaccount.

Brugeren som er logget ind kan kun se objekter som har en foreign key til brugerens systemaccount.

Usergroup "system owner" hentes, som "guest", ikke ud ved "Create user" og "Edit user" og kan derfor ikke tildeles.

Fluxusers med usergroup "Admin" og "system owner" kan ikke slettes (buttons er gemt i view). Dette har vi valgt da en account ikke må slette sin admin-bruger. Når en account oprettes, oprettes samtidig denne account's admin-bruger.

System owner har sammerettigheder som Admin, samt ovenstående.

Ved login tjekkes om brugerens account er active (1) eller har state 0 (slettet).

Endvidere opdagede vi at alle id'er i url skal krypteres, så de forskellige systemaccounts ikke kan se hinandens data ved at skrive direkte i url'en. Hertil har vi anvendt url-encoding.

6.4.6.4 *Testing*

6.4.6.4.1 Funktionstest

Vi denne iteration var der rigtig meget at teste, da vi skulle være sikre på at de forskellige systemaccounts ikke kan se hinandens data. Ligeledes ville vi sikre at ingen ud over system owner kan benytte account-actions.

6.4.6.4.2 Screen prints

SYSTEM ACCOUNT ID	ACCOUNT OWNER	2ND ACCOUNT ID	STATUS	CLIENT	ACTIONS
2	Bravida	12-1364-12	Active		
1	systemOwnerAccount		Active		

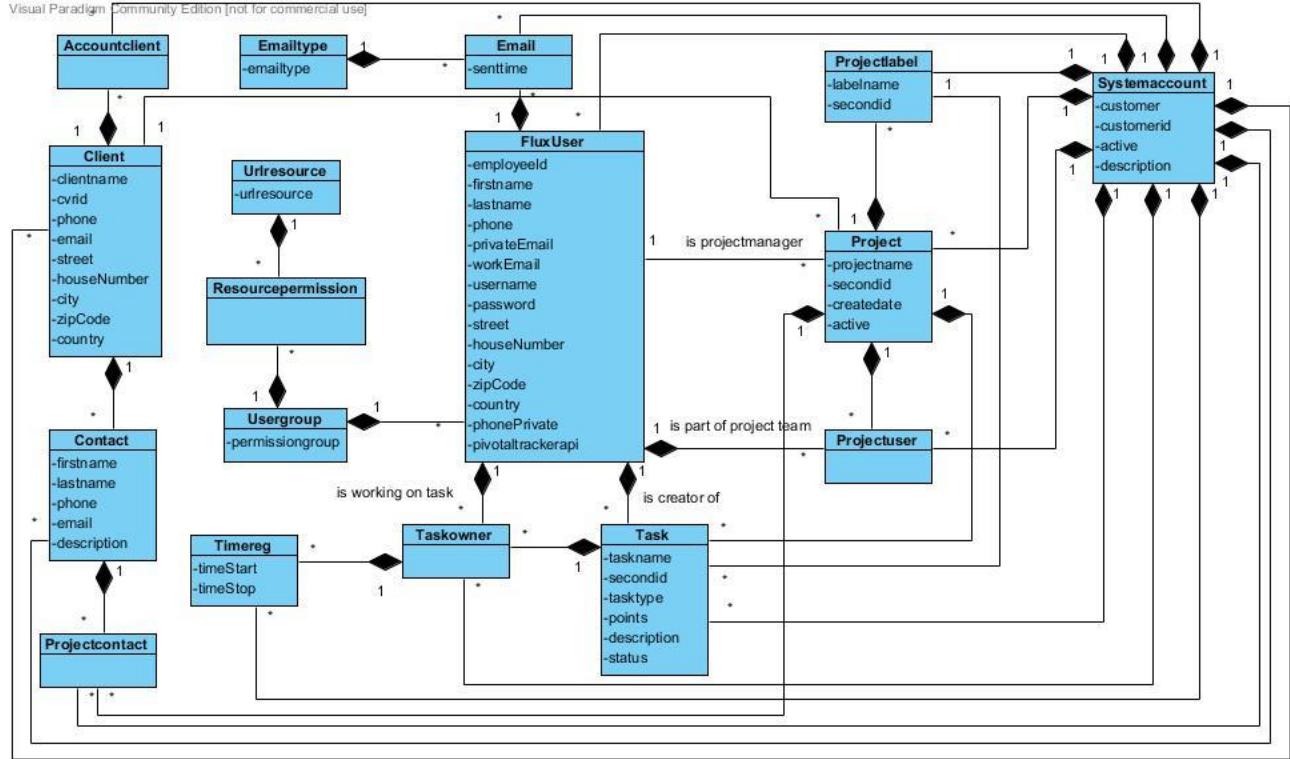
6.4.7 Iteration 7, Clients & contacts

Use cases: "Create client", "Edit client", "Delete client", "Search client", "Add contact", "Edit contact", "Delete contact", "Add contact to project", "Remove contact from project"

6.4.7.1 Requirements

6.4.7.1.1 Domain model

Vi har tilføjet 3 konceptuelle klasse til domæne modellen: Client, contact og Project contact. En Client er en kunde og en Contact er en kontaktperson hos kunden. En kunde kan associeres med et Projekt og til en Systemaccount. Contacts kan også associeres til projekter. Account client er clients som er købere af FLUXtime hos Supeo.



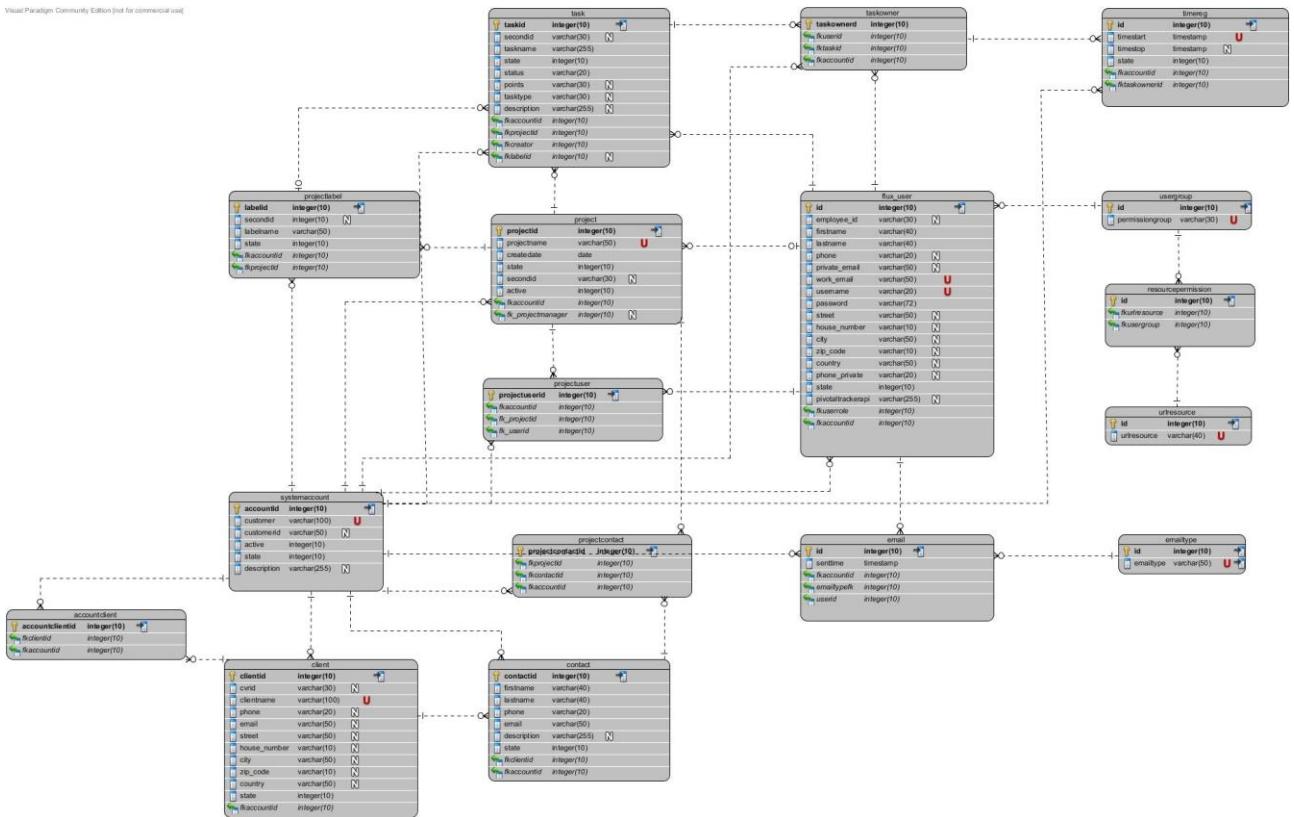
6.4.7.1.2 System sequence diagrams

Vi har valgt ikke at udforme detailed use cases, da sekvenserne minder om tidligere use cases.

6.4.7.2 Analysis & design

6.4.7.2.1 Database diagram

De 3 nye klasser i domæne-modellen er omsat til tabeller i databasen:



6.4.7.2.2 Design class diagram

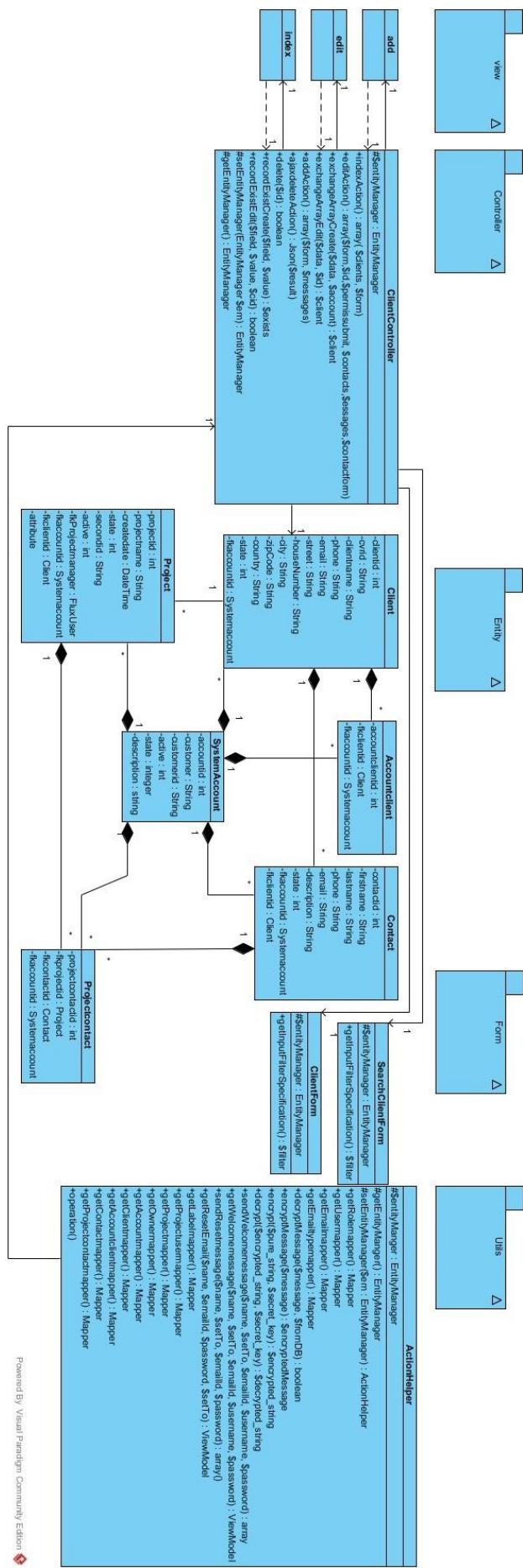
Vi har bestemt at der skal implementeres en ClientController (extender ActionHelper), hvor metoderne indexAction(), addAction() og editAction() skal placeres. Samt metoden ajaxdeleteAction(). View-filerne add, edit og index, er også illustreret på diagrammet.

Derudover har vi tilføjet 3 nye forme til diagrammet: SearchClientForm, ContactForm og ClientForm.

3 nye model-klasse, som auto-genereres vha. doctrine, er tilføjet.

Tilføjelser til klassen ActionHelper: Her er placeret metoder til at anvende doctrine-mapperne. Vi vil placere metoderne i ActionHelper, da vi mener at vi kan få brug for at anvende disse metoder i andre controllers senere i projektet.

For overskuelighedens skyld er diagrammet lavet i klynge.



6.4.7.3 Implementation

6.4.7.3.1 Implemented components

Vi har implementeret klasserne som vist i diagrammet ovenfor. Vi har besluttet at en client ikke kan fjernes fra et projekt, men den først er tilføjet. Vi kan ikke se at denne situation skulle opstå midt i et projekt og derved undgår vi også at fjerne contacts som kan være knyttet til projektet.

Da contacts oprettes under Edit client, har alle usergroups (ikke "guest") denne permission. Submit-button er dog gemt for project manager og user.

Disse permissions er tilføjet i databasen:

UrlResource	Admin & system owner	user	guest	project manager
client/index	X	X		X
client/add	X			
client/ajaxdelete	X			
contact/ajaxdelete	X	X		X
contact/ajaxadd	X	X		X
contact/ajaxedit	X	X		X
contact/ajaxaddtoproject	X	X		X
contact/ajaxremovefromproject	X	X		X
client/edit	X	X		X
Report/client	x			

6.4.7.3.2 Screen prints

EDIT CLIENT

Client*	Client id	Street	House number	Zip code
Novo Nordisk	1486443	nordiskvej	12	4760
City	Country	Phone	Email	
Vordingborg	Denmark			

Save client

Contacts

FIRST NAME	LAST NAME	PHONE	EMAIL	ACTIONS
Erik	Dresten	23 40 40 25	bt@supeo.com	

6.4.7.4 Testing

6.4.7.4.1 Funktionstest

Vi har testet alle use cases og alt validering. Vi har også testet at user og project manager ikke kan redigere et project, men går ind på edit-siden.

6.4.8 Iteration 8, Change language & Google login

Use case: google login og Change language

6.4.8.1 Requirements

6.4.8.1.1 Domain model

Ingen tilføjelser til domain model. En FluxUser's workEmail skal matche en google-account.

6.4.8.2 Analysis & design

6.4.8.2.1 Design class diagram

I ProfilController er metoden changeLanguageAction() tilføjet og i LoginController er metoderne getGoogleAuth() og googleLogin() implementeret.

6.4.8.3 Implementation

6.4.8.3.1 Implemented components

Supeo ønskede at man kunne bruge sit Google login til at logge ind i vores system. Google's login service bruger OAuth 2.0, som er bygget på OAuth 1.0. Den store forskel på v1 og v2, er at v2⁴⁶:

- Flere Oauth flows til bedre understøttelse af ikke-browser baserede applikation
 - Det største problem ved v1 var at klient applikationer var nødt til at dirrigere brugeren til at åbne sin browser, gå ind på en specific service, authentikere med den service, og kopiere sin authentication-token fra browseren tilbage til applikationen. Med v2 er der nye måder hvorpå en applikation kan authentikere en bruger.
- V2 kræver ikke at klient programmer har kryptografi længere.
 - Det stemmer helt tilbage fra den gamle Twitter Auth API, hvor der ikke kræves at applikationens hash tokens og request strenge blev krypteret med HMAC⁴⁷.
- V2 signaturere er nu mindre kompliceret
 - Man slipper for at parse, sortere og eller enkode.
- V2 access tokens er kort-livet
 - Typisk kunne v1 access token blive opbevaret i et år eller længere (Twitter lod aldrig sine udløbe). V2 bruger både access tokens og refresh tokens. Ideen er at access tokens er kortlivet, altså session basserede, mens dine refresh tokens kan holde nærmest for evigt. Du bruger en refresh token til at fornye din access token, fremfor at bede brugeren om at "re-authorize" din applikation igen.

I OAuth er der 3 roller når det kommer til authentikation⁴⁸:

Tredje-parts applikationen "klienten": Klienten er den applikation som prøver at få adgang til brugerens konto. Klienten har brug for at få tilladelse fra brugeren før den får adgang.

API'en "Resource Serveren": Resource Serveren er den API server som bliver brugt til at få adgang til brugerens information.

Brugeren "Resource Ejeren": Resource Ejeren er den person som giver adgang til en del af deres konto.

⁴⁶ <http://hueniverse.com/2010/05/15/introducing-oauth-2-0/>

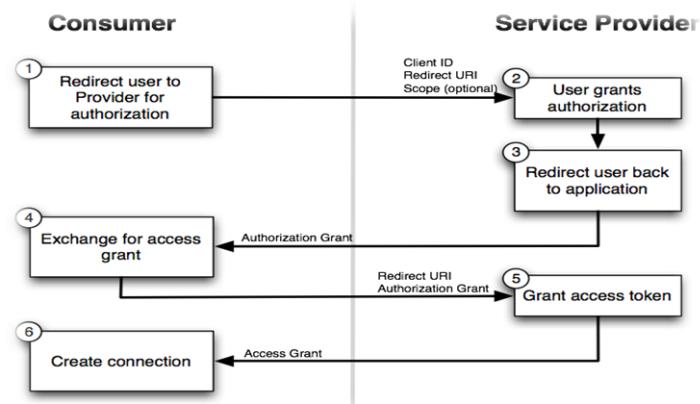
⁴⁷ http://en.wikipedia.org/wiki/Hash-based_message_authentication_code

⁴⁸ <https://aaronparecki.com/articles/2012/07/29/1/oauth2-simplified#roles>

Indtil videre har vi snakket om forskellen mellem OAuth 1.0 og OAuth 2.0, men hvad er det overordnede flow egentlig i OAuth 2.0 som vi bruger? Det er også kendt som 'the OAuth dance'⁴⁹

Først bliver brugeren henvist til Google (tjenesteyderen), som beder brugeren om at godkende applikationen. Derefter bliver brugeren henvist til den applikation som vil have adgang til brugerens konto. I denne henvisning har applikationen fået en 'authorization grant' som bruges til at udveksle en access token med tjenesteyderen. Se billedet nedenunder.

For at kunne bruge Google som en tjenesteyder, var vi først nødt til at oprette et projekt hos deres developer console (<https://console.developers.google.com/project>). Dette indbefatter at man godkender en masse terms og conditions. For at oprette et project skal man give det et navn og et id. Id'et består af 2 ord og et 3-cifret tal, dette kan Google hjælpe en med at vælge, da de giver en et random generated et. Derefter åbner man projektet og har nu adgang til en masse muligheder. De eneste ting vi var interesseret i lå under menu-punktet *APIs & auth*, under første punkt, *APIs*, skulle man vælge hvilken API man ville have adgang til, den eneste vi var interesseret i var deres Google+ API. Under næste punkt, *Credentials*, skulle man angive hvilket *redirect URI*'s som Google kunne sende brugeren hen til. Derefter skulle man angive hvilket *Origins* som Google skulle tillade request med vores oplysninger fra. Det næste punkt, *Consent screen*, gjorde at man kunne tilpasse den side som brugeren bliver henvist til når brugeren skal give tilladelse til applikationen. Man kan tilpasse logo, email-addresse, produkt navn, hjemmeside, privacy policy og terms of service samt linke til en Google+ profil.



Zend Framework 2 (ZF2) kommer med indbygget oversættelse plugin der hedder ZendI18n. Pluginet understøtter alle større oversættelses metoder: PHP Arrays og Gettext

Opsætningen af ZendI18n foregår i ZF2's module configurations fil:

⁴⁹ http://docs.spring.io/spring-social/docs/1.1.0.RELEASE/reference/htmlsingle/#section_establishingConnections

```

4  /* Translator */
5  'translator' => array(
6      'locale' => 'da_DK',
7      'translation_file_patterns' => array(
8          array(
9              'type' => 'gettext',
10             'base_dir' => __DIR__ . '/../language',
11             'pattern' => '%s.mo',
12         ),
13     ),
14 )

```

I opsætningen kan man se at sproget vi gerne vil oversætte til er Dansk. Vi vil gerne bruge Gettext metoden, den skal finde oversættelses filen i language mappen i roden af vores ZF2 module. Da det er Gettext vi bruger skal vi kigge efter .mo filen med navn da_DK.

PHP Arrays

Måden man bruger et PHP Array til oversættelse af ZF2 foregår ved man angiver et statisk Array med alle oversættelser. Man laver først en ny .php fil med oversættelserne den kalder vi da.php:

```

1 return array(
2     'Work' => 'Arbejde',
3     'Packaging' => 'Indpakning,
4 );
5
6 Og så hører man skal bruge en oversættelse.
7
8 echo translate('Work');
9
10 Output: Arbejde

```

Men vi fandt ud af at denne metoder ikke var særlig fleksibel i forhold til nogen vores oversættelser, forstået på den måde at nogle af de ting vi skulle oversætte ikke var i ren tekst.

Vi har valgt at bruge Gettext i vores system, der er gratis software og allerede er implementeret i ZF2.

Gettext

Gettext fungere ved brug af .po filer der er en liste af oversættelser. Hver oversættelse har et id og en tilhørende string. Id'et er den originale sætning og stringen er oversættelsen. Den første linje er en reference, så man kan se hvor oversættelsen hører til.

```

35 #: src/Fluxuser/Form/EditFluxuserForm.php:76
36 msgid "Last name"
37 msgstr "Efternavn"

```

Før Gettext kan tage .po filen i brug skal den først kompileres om til en .mo fil, der er .po filen bare i binær tekst.

Men for at bruge det indbyggede Gettext pluggin ude i vores view af programmet, skal vi kalde en metode der i ZF2 er *translate()* eller Gettext's egen *get()*:

```

5 echo translate("Last Name");
6     eller
7 echo get("Last Name");
8
9 Output: Efternavn

```

Disse permissions er tilføjet i databasen:

UrlResource	admin	user	guest	project manager
Profile/changeLanguage	X	X	X	X

6.4.8.3.2 Screen prints

The image shows a login form with the following fields and options:

- Email***: admin@flux.dk
- Password***: (redacted)
- Action buttons: **Login** (green), **g+** (red), **Sign in with Google** (red), and **Reset password**.

6.4.8.4 Testing

6.4.8.4.1 Funktionstest

Vi har testet at en fluxuser med en google account kan logge ind med google og at brugeren kan skifte sprog fra dansk til engelsk og omvendt.

7 Konklusion

7.1 Problemløsning

7.1.1 Projektets status

Gennem rapportens afsnit har vi dokumenteret hvordan vi har udviklet et system til registrering af tidsforbrug på opgaver. System kan anvendes som stand-alone system og kan også importere projekter, epics/labels/kategorier og opgaver fra et eksternt projektstyringssystem, Pivotal Tracker.

Via systemet kan vores kunde generere excel-rapporter med tidsregistreringer og lave statistik mv. Vha. disse rapporter kan faktureringsarbejdet lettes og filerne vil kunne benyttes til at eksportere data til faktureringssystemet som f.eks. Economic.

Vi er blevet meget tilfredse med systemet og det lever op til kundens krav.

Indtil eksamen vil vi tilføje enkelte ønsker fra vores kunde og forberede materiale til overdragelse af systemet. Ligeledes vil vi uddanne en superuser. Vi har lagt meget vægt på at vise processen gennem rapporten (hvordan udvikles systemet og artefakter gennem iterationerne), da dette jo er en skole-opgave, og derfor er vores rapport ikke et "opslagsværk". Derfor vil vi gerne lave en "quick-guide" til Supeo med meget forklarende system sekvens diagrammer for (næsten) alle use cases (eller lign.), samt brugervejledning.

7.2 Process

7.2.1 Projektstyring og SCRUM

Vi har afholdt korte SCRUM-møder hver morgen. I starten af projektet var møderne mere formelle, men efter kort tid blev det mere en vane og det har fungeret rigtig godt.

I hvert sprint har vi afholdt review- og retrospect møder og vi har faktisk fået snakket rigtig meget om process. Vha. reviewmøderne har vi sikret at mål var nået, incl. kvalitetsmål. Vi har planlagt sprints vha. en risk list og dermed har vi hele tiden været få forkant med risks og har kunne finde løsninger på forhindringer.

Brugen af Git til projektstyring, er gået rigtig godt, men det krævede en del tilvending, og opsætningen var heller ikke problemfri, men efterfølgende har vi haft stor glæde af git. Til at starte med var vi ikke særligt gode til at arbejde på branches. De branches vi så endelig lavede blev enten for store, så der kom en masse bøvl med merging til master, eller også fik vi ikke commitet det ordentligt. Men efter en periode, blev vi vant til at bruge det og vi fik branches til at fungerer godt.

Under projektet har vi anvendt scrum-værktøjet Pivotal Tracker, der også krævede en del tilvending. Faktisk var vi lige ved at gå over til et fysisk scrum-board, men da product owner næsten insisterede på at vi brugte det, samt vi skulle komme PT's story id ind i vores commits, gav vi det et forsøg igen. Det har gået okay med Pivotal Tracker, men da vores gruppe der arbejder så tæt sammen, ville et fysisk scrumboard være bedre.

7.2.2 Udviklingsmetoder

Vi har anvendt situationsbestemt udviklingsmetode og har brugt værktøjer fra både OOA/D og UP. Vi har lavet de artefakter vi mente var nødvendigt for at identificere krav og designe softwaren. Vi valgte at lave en meget grundig foranalyse og dette tror vi har hjulpet os meget til at forstå systemet – især når det til tider kunne virke meget komplekst.

7.2.3 Samarbejde

Vi samarbejder rigtig godt sammen i gruppen og vores forskellige kompetencer komplimenterer rigtig godt hinanden. Vi har stort set alle været med til at løse alle opgaver, i større eller mindre grad, og vi har derfor diskuteret meget undervejs og i fællesskab nået frem til nogle gode løsninger. Det har været en stor fordel at vi fra projektets start, oprettede et kontor, hvor vi mødtes hver dag.

7.2.4 Samarbejde med kunde

Kunde-samarbejdet har ikke været optimal i dette forløb. Vi tror dette skyldes at vi er vant til at kommunikere på forskellige måder og via forskellige værktøjer. Vi har lært at vi en anden gang skal huske at sætte nogle rammer for kommunikation i projektets opstartsfas, så forventninger afstemmes. Supeo har været rigtig hjælpsomme med at sørge for at de tekniske rammer.

7.2.5 Udviklingsværktøjer

Vi havde del bekymringer om hvor vidt vores valg af framework var det rigtige. Der stod rigtig mange steder at Zend havde en stejl læringskurve. Vi må bare siges at det er fuldstændig rigtigt. Faktisk er det meget besværligt at finde god dokumentation, tutorials eller anden hjælp. Zend har ellers en rimelig god hjemmeside. Men vi mener at API'en er lidt dårligt beskrevet. Der findes hjælp på nettet og der ligger en del tutorials, men de fleste er kæmpe store, dårligt lavet eller er Youtube-videoer, med mange timers indhold. Et andet større problem er configurationen, som er for svær at gennemskue, man kan simpelthen lave tingene på for mange måder.

Vi valgte at bruge Doctrine fordi vores kunde havde et ønske om dette. Ideen bag er rigtig god og vi er blevet mere og mere glade for måden at bruge forbindelsen til databasen på. Men Doctrine til Zend Framework 2 kan igen gøres på flere forskellige måder. Hver tutorial vælger at bruge hver deres måde og vi havde nær opgivet at styre udtræk fra mange joinede tabeller, men til sidst lykkedes det at finde en god metode. Vi har måtte erfarer at man ikke kan bruge alt den sql som man kan i PostgreSQL. Doctrine har lavet deres egen version og det kan forvirre en del. Når man først har fundet fidusen, bliver det bedre og bedre for hver gang man bruger det.

8 Perspektivering

Der er flere muligheder for videreudvikling af systemet FLUXtime:

- Import
 - Import fra Pivotal Tracker kan optimeres, således at elementer som er slettet i PT også slettes i FLUXtime. Endvidere er import-metoden meget tung og kan performance-mæssigt optimeres. Ligeledes kan man eventuelt anvende RabbitMQ (kø-server) for at hver klient er synkroniseret med Pivotal Tracker.
 - Import fra andre opgave-systemer kan implementeres.
- Export
 - Vi nåede ikke at ekspotere data til Economic og havde heller ikke en licens til programmet. Dette kunne også implementeres.
- Contact-FluxUsers
 - Client contacts kan tilføjes til projekter i FLUXtime, men er blot en kontaktperson. Nogle gange er ansatte hos kunden af projektet også med i projektteamet og ville have glæde af at kunne følge med i status på opgaver mv. Derfor kunne man give mulighed for at oprette contacts som fluxusers og lave en usergroup hertil for at definere permissions.

9 Litteraturliste

Websites:

Unit testing:

<https://phpunit.de/>

<http://en.wikipedia.org/wiki/PHPUnit>

<http://en.wikipedia.org/wiki/XUnit>

http://en.wikipedia.org/wiki/Behavior-driven_development

Framework research:

<http://systemsarchitect.net/performance-benchmark-of-popular-php-frameworks>

<http://www.sitepoint.com/best-php-frameworks-2014>

<http://vschart.com/compare/laravel/vs/zend-framework>

<http://codegeekz.com/20-best-php-frameworks-developers-august-2014/>

<http://systemsarchitect.net/performance-benchmark-of-popular-php-frameworks/>

Web development methods:

<http://www.ifiptc8.org/asp/aspecis/20000053.pdf>

UP:

www.upedu.org

SCRUM:

www.scrum.org (taget fra fronter)

ACL:

<http://ivangospodinow.com/zend-framework-2-acl-setup-in-5-minutes-tutorial/.>

Sikkerhed:

<http://stackoverflow.com/questions/5393803/can-someone-explain-how-bcrypt-verifies-a-hash>

CURL:

<http://php.net/manual/en/intro.curl.php>

Google-login:

<http://hueniverse.com/2010/05/15/introducing-oauth-2-0/>

http://en.wikipedia.org/wiki/Hash-based_message_authentication_code

<https://aaronparecki.com/articles/2012/07/29/1/oauth2-simplified#roles>

http://docs.spring.io/springsocial/docs/1.1.0.RELEASE/reference/htmlsingle/#section_establishingConnections

Bøger:

Avision:

David Avison og Guy Fitzgerald. *Information systems development, methodologies, techniques & tools*, 4. Udgave, McGraw-Hill Education, 2006

OOAD:

Lars Mathiassen og Andreas Munk-Madsen m.fl. *Objekt orienteret analyse og design*, 3. Udgave, Marko ApS, 2001

Larman:

Craig Larman. *APPLYING UML AND PATTERNS*, 3. Udgave, Pearson Education, 2011

Professionel systemudvikling (PS):

Niels Erik Andersen, Finn Kensing m.fl. *Professionel Systemudvikling, Erfaring, muligheder og handling*, Ingeniøren A/S, 2002

10 Bilag

10.1 Oplæg fra Supeo

Tidsregistrering

Kort opsummering af opgaven

Supeo er et mindre Software hus med 6-12 udviklere, der søger at gøre vores tidsregistrering mere simpel, og mere opgave orienteret. Dette skal opnås ved at koble kunde ønsker, bugs og øvrige opgaver sammen med den tidsregistrering vi foretager.

I dag anvendes Timelog.dk og deres Timelog Projekt sammen med Pivotaltracker. I Timelog styres tidsregistrering på baggrund af projekter, faser og underfaser. Dette giver en meget lang liste af faser man kan skrive sin tid på. Det vi ønsker at ændre er at man skal have en sag i Pivotaltracker når man har lavet denne, kan man via tidsregistrerings systemet, starte tidsregistreringen på opgaven og starte sit arbejde. Efterhånden som dagen går kan man pause, starte og til sidst stoppe tidsregistreringen helt.

Som administrator skal man kunne trække rapporter på de enkelte medarbejdere og se deres tidsregistrering. Der er flere krav til hvordan rapporten skal fungere og den skal bygges op som en pivot rapport.

Krav til løsningen

Der er følgende indledende krav til løsningen. Kravende skal fastsættes på møder med projekt gruppen derfor er dette ikke nødvendigvis en udtømmende kravliste.

Rating MK = Minimumskrav, AK = Almindeligt krav og Ø = Ønske.

ID	Beskrivelse	Rating	Bemærkninger
1	Platform		
1.1	Systemet skal laves som en webapp.	MK	
1.2	Der skal anvendes HTML5, CSS3 og Javascript (Jquery/Closure)	MK	
1.3	Der er ikke noget krav til valg af sprog der anvendes i backend, men der skal forelægges en mindre journal med research der forklare hvorfor man har foretaget dette valg.	MK	Der er en mulighed for at bruge sitra skelettet uden eksisterende moduler i en ny webapp. Man kunne også lave en ny Zend Framework 2 applikation.
1.4	Backend skal køre på Apache2 webserver	Ø	
1.5	Databasen skal være PostGRESQL	Ø	
2	Brugerstyring		
2.1	Man skal kunne oprette en ny bruger	MK	
2.2	Man skal kunne redigere en bruger	MK	
2.3	Man skal kunne nulstille brugerens password	MK	Uden at være logget ind
2.4	Man skal kunne aktivere/deaktivere en bruger	MK	
2.5	Brugere skal have Initialer (Brugernavn), Fornavn, Efternavn, Medarbejder ID, e-mail, password ...	MK	
2.6	Der skal være en tabel hvori man kan se alle brugere og deres basis info samt foretage ovenstående krav 2.1-2.4.	MK	
2.7	Der skal være en login side hvor man kan logge ind i systemet og nulstille glemt password.	MK	

3	Tidsregistrering		Denne del af system beskrivelsen skal der arbejdes på. Workshop?
3.1	Der skal være et brugerinterface hvor man kan registrere sin tid på opgaver som man nem skal kunne søge igennem dette interface.	MK	
3.2	Man skal kunne se sin tidsregistrering for den seneste uge.	MK	
3.4	Når man har fundet en opgave skal man kunne starte tidsregistrering på den, såfremt den er åben for dette.	MK	
3.5	Man skal kunne pause tidsregistreringen.	MK	
3.6	Man skal kunne stoppe tidsregistreringen.	MK	
3.7	Man skal kunne afslutte tidsregistrering på en opgave.	MK	
3.8	Det skal være muligt at vælge kun at se åbne sager og ikke afsluttede opgaver man ikke skal registrere mere på.	MK	
3.9	Man skal kunne se projekter, keywords og epics fra pivotal tracker således, at opgaver inddeltes efter dette. (Se krav 5..)	MK	
4	Oprettelse af Projekter / opgaver		
4.1	Det skal være muligt manuelt at oprette projekter og opgaver hertil.	MK	
4.2	Man skal kunne gøre alt med disse opgaver som man kan med opgaver fra andre systemer.	MK	
4.3			
5	Integration Pivotaltracker		
5.1	Systemet skal integreres således at det er koblet op på Pivotaltracker og igennem deres API kan hente projekter, opgaver ETC.	MK	
5.2	Synkroniseringen skal gerne være så realtids nær som muligt.	Ø	
6	Integration Sitra		
6.1	Der skal forberedes så man kan få opgaver fra andre systemer som fx Sitra via et API.	MK	
6.2	Systemet skal kunne hente opgaver igennem Sitras Case modul via et simpelt JSON API som Supeo udvikler til lejligheden.	AK	
6.3	Det kan implementeres ved hjælp af RabbitMQ køer men det er ikke et ultimativt krav.	Ø	
7	Integration / Eksport til E-Conomic		
7.1	Det skal være nemt at eksportere tidsregistrering på et projekt / kunde til E-Conomic således at det kan danne grundlag for en fakturering af tiden.	MK	
8	Rapport		
8.1	Man skal kunne trække rapporter på tidsregistrering.	MK	
8.2	Rapporten skal kunne vælges med fra dato og en til dato (midnat til midnat).	MK	

8.3	Man skal kunne vælge en gruppe af medarbejdere eller en enkelt medarbejder.	MK	
8.4	Man skal kunne vælge et eller flere projekter.	MK	
8.5	Man skal kunne vælge en eller flere epics i et projekt.	MK	
8.6	Man skal kunne vælge en periode (Seneste Uge, Måned, Kvartal, Halvår eller år)	MK	
8.7	Man skal kunne søge på en enkelt sag/opgave og vælge den.	MK	
8.8	...		

Projektet og dets vilkår

Ophavsret

Supeo har rettigheder til alt materiale, kode, diagrammer og øvrige tilblivelser i forbindelse med projektet.

Alt kode, dokumentation etc. Skal forsynes med en copyright notifikation som Supeo udarbejder.

Ret til brug af materiale ved eksamen

Projekt gruppen har ret til at anvende det nødvendige materiale ved eksamen. Efter eksamen skal materialet leveres tilbage til Supeo. Og projektet, kode etc. må ikke anvendes af skolen efterfølgende.

Tavshedspligt

Projekt gruppen har tavshedspligt i forbindelse med alt hvad de måtte erfare om Supeo igennem deres arbejde med projektet.

Lokaler, værktøjer m.m.

Supeo stiller lokaler, værktøjer og nødvendigt personale til rådighed for projektgruppen i projekt perioden. Derudover forventes det at projektgruppen deltager i møder og lign. Som Supeo måtte indkalde projektgruppen til.

10.2 Mødreferater fra kundemøder

10.2.1 10. nov. 2014

Deltagere: Maren, Brian, Nicklas, Anders, Troels & Jeppe (Supeo)

- Gennemgang af nuværende system
 - Visning af timelog.dk
 - e-economic
 - Pivotal Tracker
 - Sitra
- Valg af systemdefinition: Stand alone eller plugin (skal være meget fleksibelt) – begge!
- Spørgsmål til minioplæg /krav:
 - Mange forskellige slags brugere
 - Medarbejder og kunde 2 id'er. Et system og et til indtastning, f.eks. lønnummer
- Brainstorming

- Permissions grupper: Admin, project manager og user
- Pivotal tracker: Systemet må meget gerne kunne starte og afslutte opgaver i PT. Evt. også oprette opgaver og epics.
- Kontaktpersoner til clients
- Skal kunne anvendes som kunde- og/eller medarbejderdatabase, men mange data skal være nullable
- Kan der logges på med google account?
- Teknik
 - Export af data til Economic: Kan eksporteres til excel som start. Dog har dette ikke så høj prioritet og derfor er dette sat lavt i backlog.
 - Vi skal anvende Zend framework 2 - krav
 - Stort ønske at vi anvender doctrine
 - Nedprioritering af realtids-opdatering, for at opnå bedre performance – brug evt. RabbitMQ og opdater hvert minut
 - Rapporter skal eksporteres til excel

Aftaler: Super opretter bruger-profiler til teamet til bl.a. test af systemet, projekter i pivotal tracker til kode, fildeling og test. Ligeledes opsættes en server.

[10.2.2 25. nov. 2014](#)

Deltagere: Maren, Brian, Nicklas, Anders, Troels & Jeppe (Supeo)

- Præsentation af system og userbility testing (use cases Elaboration it. 1 og Construction it. 1)
 - Product owner er tilfreds med use cases og design

[10.2.3 9. dec. 2014](#)

Deltagere: Maren, Brian, Nicklas, Anders, Troels & Jeppe (Supeo)

- Præsentation af system og userbility testing (use cases Construction it. 2 og 3)
 - Product owner er tilfreds med use cases og design
- Pivotal tracker/kommunikation
 - Projektteamet ønsker ikke at anvende Pivotal Tracker, men et fysisk scrum-board. Supeo ønsker at teamet anvender PT for at de kan følge processen og se commits (kode).
 - Gennemgang af hvordan Supeo benytter PT og hvilken værdi det tilfører (anvendes som en form for systemdokumentation)
 - Aftale: Det var ikke muligt at finde et umiddelbart alternativ, så derfor forsøger projektteamet at anvende PT, sådan som Supeo ønsker
- Nye/forandrede krav:
 - Hvis systemet skal sælges til flere kunder som anvender samme server, skal hver bruger være tilknyttet en konto. Dette betyder at alle tabeller i databasen og udtræk skal have et konto-id som foreign key.

[10.2.4 9. Jan 2015](#)

Deltagere: Maren, Brian, Nicklas, Troels (Supeo)

- Præsentation af næsten færdigt system
 - Kunden virkede meget tilfreds med systemet

- Små ændringer og ønsker:
 - Vise taskname, id og secondid, samt tasktype på excel-rapporter
 - Evt. gøre change language-ikon mindre
 - Også Add/Remove PT API token under Edit profile
 - Tilføje pagers under MyLog, Log og Task

10.3 Dokumentation fra Pivotal Tracker

10.3.1 Inception

The screenshot shows the Inception board with two main sections:

- Actors, usecases, backlog (NL):**
 - Inception
 - Draw use case model (NL)
 - systemdefinitioner (NL)
 - Business Analysis (NL)
 - Architecture of the ZF2 (TH)
- Research:**
 - research installation of xamp, zend framework2 and postgres database både intern og extern (ABR, NL)
 - hvordan man installerer javascript korrekt i zend 2 (TH)
 - hvordan installerer bootstrap i zend 2 (TH)
 - How to translate ZF2 (NL)
 - Research remote depository (ABR, NL)
 - how to php unit test (ABR)
 - how to connect zend to pivotaltracker (NL)
 - how zend 2 use crud (TH)
 - hvordan man sender en mail i zend

Backlog stories and Icebox stories are listed at the bottom.

10.3.2 Elaboration

10.3.2.1 Elaboration sprint 1, Users

The screenshot shows the Elaboration iteration 1 board for the 'elaboration iteration 1' epic. It contains a single column of tasks:

- elaboration iteration 1 Made Super, revise requirements & risk list, review & retrospect, sprint plan & goals elaboration 1 (M-H)
- elaboration iteration 1 System sequence diagrams - user (M-H)
- elaboration iteration 1 Database diagram - user
- elaboration iteration 1 Sequence diagrams - user CRUD (M-H)
- elaboration iteration 1 Design class diagram - user (husk klasse-kommentering) (M-H)
- elaboration iteration 1 Implement db entities - user & login (ABR)
- elaboration iteration 1 Implement main design
- elaboration iteration 1 Implement stored procedures - user (ABR)
- elaboration iteration 1 Implement insert user (incl styling) (ABR)
- elaboration iteration 1 Implement select user/users (incl styling) (M-H)
- elaboration iteration 1 Implement search user (ABR)
- elaboration iteration 1 Implement edit my profile (incl styling)
- elaboration iteration 1 Implement delete user (incl styling) (M-H)
- elaboration iteration 1 Implement update user (incl styling) (ABR)
- elaboration iteration 1 Unittesting user (ABR)

Backlog stories and Icebox stories are listed at the bottom.

10.3.3 Construction

10.3.3.1 Construction sprint 1, Usergroups

The screenshot shows the 'My Work' board in the FLUXtime application. It lists several tasks under the 'elaboration it 2 - permissions' epic:

- Opret 3 permission groups (implement) (N)
- DB script (NKO)
- Implement select permission groups (NKC)
- Implement add user to permission group (
- Implement remove user from permission g

Backlog stories and Icebox stories are listed at the bottom.

10.3.3.2 Construction sprint 2, Login

The screenshot shows the 'Construction 2 - Login' board with tasks grouped by epic:

- Domain model? (BTJ) elaboration it 3 - login
- DB script? (BTJ) elaboration it 3 - login
- Implement login first time (BTJ) elaboration it 3 - login
- Implement login (BTJ) elaboration it 3 - login
- Implement logout (BTJ) elaboration it 3 - login
- Implement reset password (NKO) elaboration it 3 - login

10.3.3.3 Construction sprint 3, Projects & Labels

The screenshot shows the 'Epics' board with completed epics:

- Construction 3 - Projects & Labels
- Construction 2 - login
- Construction 1 - permissions
- Elaboration 1 - users
- Construction 4 - Tasks

Progress bars indicate completion for each epic.

10.3.3.4 Construction sprint 4, Tasks & Import

The screenshot shows the 'Construction 4 - Tasks' board with tasks for the 'construction - tasks' epic:

- Search tasks (BTJ) construction - tasks
- Add task construction - tasks
- Edit tasks construction - tasks
- Delete task construction - tasks
- Add task owner construction - tasks
- Remove task owner construction - tasks
- Search my tasks construction - tasks
- Finish task construction - tasks

Backlog stories and Icebox stories are listed at the bottom.

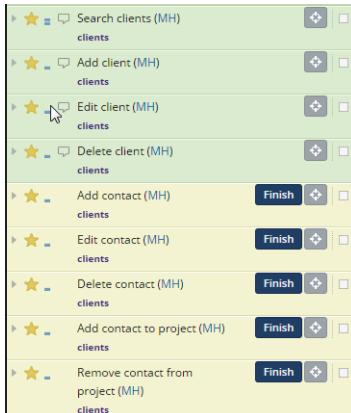
10.3.3.5 Construction sprint 5, Time registration & Excel reports



10.3.3.6 Construction sprint 6, System accounts



Construction sprint 7, Clients & Contacts



10.4 Guides

10.4.1 Netbeans og Git

```
#####
# How to use git with Netbeans #
#####
```

NOTE: most of this information will be
Autofilled in by Netbeans.

We will first clone a repository using the
Git client in Netbeans.

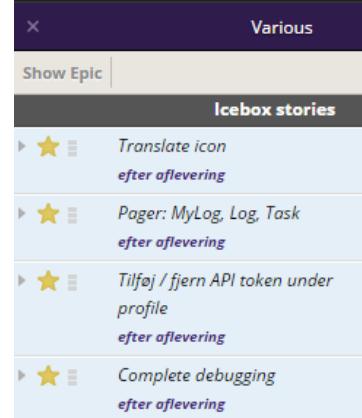
First go to:

Team -> git -> clone repository

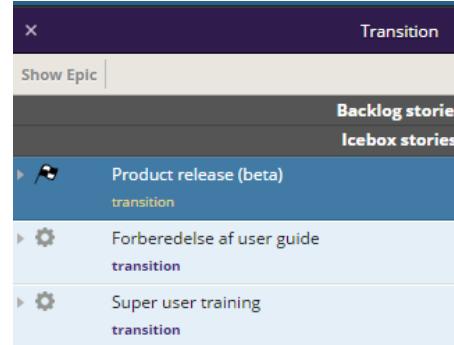
10.3.3.7 Construction sprint 8, Google login & Change language



10.3.3.8 Construction sprint 9, Various



10.3.3.9 Transition



```
# We will now fill out the following forms:  
Repository URL: https://github.com/Anras573/FluxDeveloper.git  
User: your-username-here  
Password: your-secret-password-here  
# NOTE: remember to press 'Save Password'!  
  
# Now we will specify a folder where we would  
# Like to clone our repository into.  
Clone into: C:\xampp\htdocs  
  
# Press 'Next'  
  
# Now we will choose a branch (or branches)  
# Most likely the only branch we're interested in  
# Will be the master branch  
  
# Press 'Next'  
  
# Now we will specify the Parent Directory  
# and this should already be the same as the  
# One we chose before:  
Parent Directory: C:\xampp\htdocs  
Clone Name: tidsregistrengs-projekt-supeo  
Checkout Branch: master*  
Remote Name: origin  
  
# Remember to press 'Scan for Netbeans'  
# Projects after Clone'  
  
# Press 'Finish'  
  
# Netbeans will now clone the repository  
# And ask if you want to open the Netbeans  
# Project now.  
  
# Press 'Open'  
#####  
  
# Now we will actually begin using git in  
# Netbeans for development.  
  
# First step is to pull master branch and  
# Create your own branch.  
Right click the project in Netbeans and choose  
Git -> Remote -> Pull...  
  
# After the pull, we'll create our new branch  
Right click the project in Netbeans and choose  
Git -> Branch/Tags -> Create Branch
```

```
# NOTE: I suggest we use our initials as a means
# To tell others who is responsible for the
# Branch.
# Example: abr/create_draft_user
Branch Name: initialsHere/what_this_branch_is_about
Revision: master

Press 'Checkout Created Branch'
Press 'Create'

# You are now working on your own branch instead of master!

#####
# The three most used features of git in Netbeans is
# Pull, Push and commit.

# When you pull, you pull the data from the repository onto
# Your computer.
Rigth click the project in Netbeans and choose
Git -> Remote -> Pull...

# Now you can choose which branch to pull from (usually master)

# When you commit, you take your changes and 'merge' them into a
# Commit with a message describing what you did in this commit.
# Example: created user login screen
Rigth click the project in Netbeans and choose
Git -> Commit...
Commit Message: what-is-this-commit-about?
Press 'Commit'

# When you push, you push the data (your commits) from your computer
# Onto the remote repository.
Rigth click the project in Netbeans and choose
Git -> Remote -> Push...

# Now you can choose which branch to push (usually the one you're
# Currently working on)

# NOTE: You will normally pull the master branch down and create a
# New branch from master.
# NOTE: You always commit
```

10.4.2 Netbeans, Xampp, Composer, Zend framework 2, PostgreSql

```
#####
# Installation guide for Netbeans 8.0,      #
# xampp, composer, zend framework & PostgreSQL #
```

```
#####
# First we install Netbeans
https://netbeans.org/downloads/

# Then we open up netbeans and go to
# Tools -> Options -> PHP -> Framework & Tools
# Choose 'Activate this feature'
# Download Zend Skeleton Application
https://github.com/zendframework/ZendSkeletonApplication/archive/master.zip

# Go back to Netbeans and browse for the zip
# File. Choose Apply.

# Next step is to download and install xampp
# NOTE: I suggest you install it on C://xampp/
# NOTE: you can't run Skype and Apache Webserver at
# The same time! (You can however change the ports
# In a config file)
http://downloads.sourceforge.net/project/xampp/XAMPP%20Windows/1.8.3/xampp-win32-1.8.3-5-VC11-installer.exe

# Next step is to download and install the Composer
# NOTE: php.exe is located at 'C://xampp/php/php.exe'
https://getcomposer.org/Composer-Setup.exe

# Run the installer, follow the instructions
# And restart Netbeans.

# Last step is to download and install PostgreSQL
# And connect it to xampp.

# First step is to download the correct version:
# NOTE: Do NOT install it yet!
32-bit: http://www.enterprisedb.com/postgresql-9351-installers-win32?ls=Crossover&type=Crossover
64-bit: http://www.enterprisedb.com/postgresql-9351-installers-win64?ls=Crossover&type=Crossover

# Last step is to follow this guide
# In this guide we will install the PostgreSQL database.
# NOTE: When asked where to install PostgreSQL change
# C://program files/PostgreSQL/9.3/
# To:
# C://xampp/PostgreSQL/9.3/
# NOTE: When asked if you want to launch Stack Builder, press
# Yes. Choose 'PostgreSQL 9.3 (x64/x86) port 5432'.
# Open the bottom arrow and choose both options (If you choose
# The bottom option, the installer will choose both options per
# Default).
# NOTE: the installer will freeze, but this is completely normal.
# NOTE: under headline "Getting PostgreSQL to talk with PHP"
```

```

# Step 3, this made Apache Webserver crash everytime, and under
# The installation, this line was added to another config file
# Instead.
# NOTE: When the guide ask you to write stuff like:
# 'C:\\xampp\\pgsql\\9.1\\pg_dump.exe';
# You should actually write:
# 'C:\\xampp\\PostgreSQL\\9.3\\pg_dump.exe';
http://w3guy.com/integrate-postgresql-database-xampp-windows/

# NOTE: If you want to be able to connect to your server from another machine,
# You should comment out the following line in 'httpd-xampp.conf':
# Require local
# The line is at the bottom of the file.

# NOTE: in order to tell the server which project to run, you must
# Specify the DocumentRoot in the following file:
# C:\\xampp\\apache\\conf\\httpd.conf
# Example:
# Change the following lines:
# DocumentRoot "C:/xampp/htdocs"
# <Directory "C:/xampp/htdocs">
# To:
# DocumentRoot "C:/xampp/htdocs/PhpProject1/public"
# <Directory "C:/xampp/htdocs/PhpProject1/public">
# If your project is called PhpProject1 and the source code
# Resides in the public folder.

```

10.4.3 Doctrine – Create entities

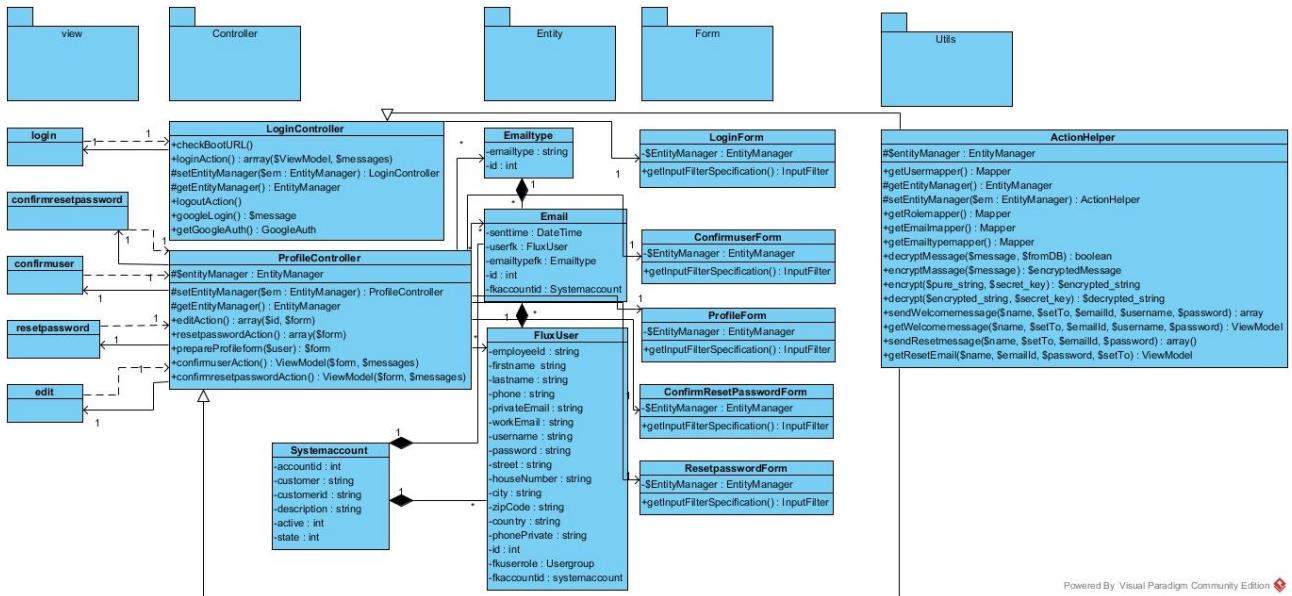
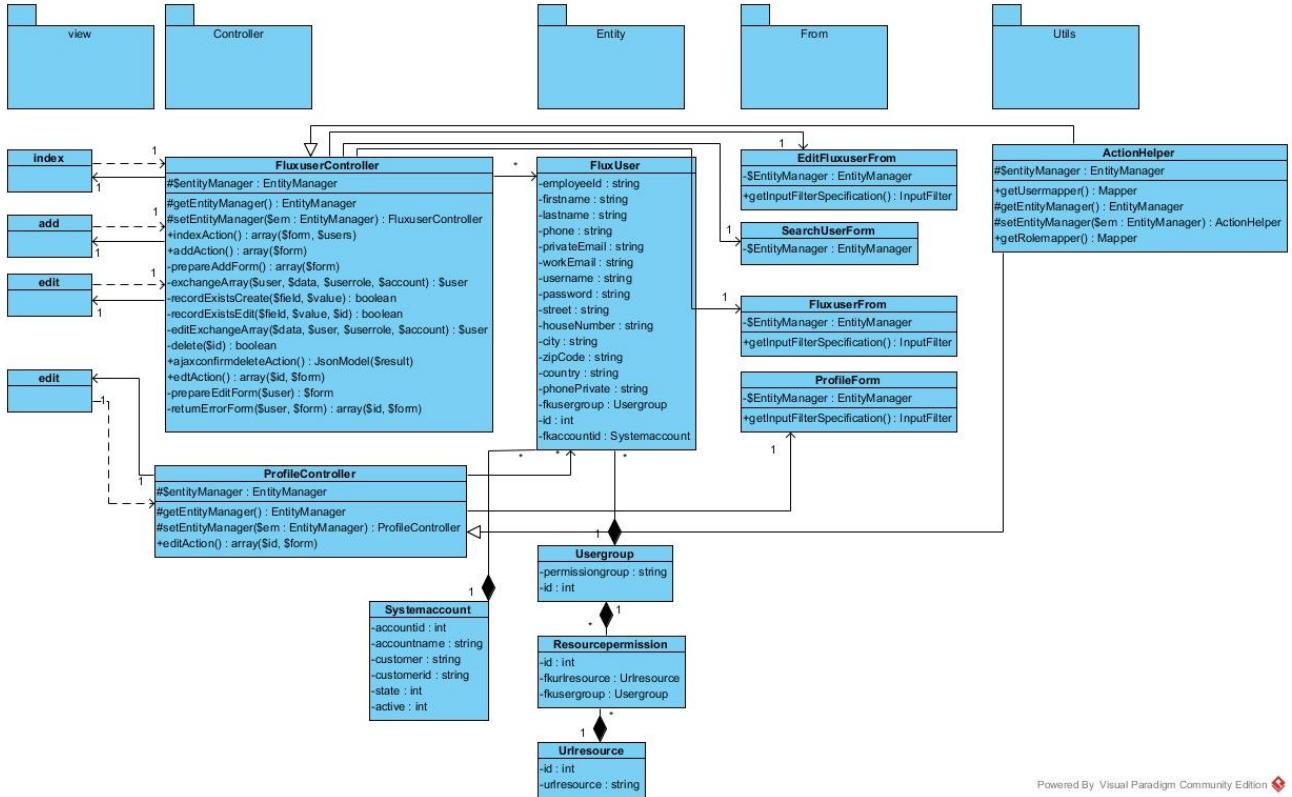
```

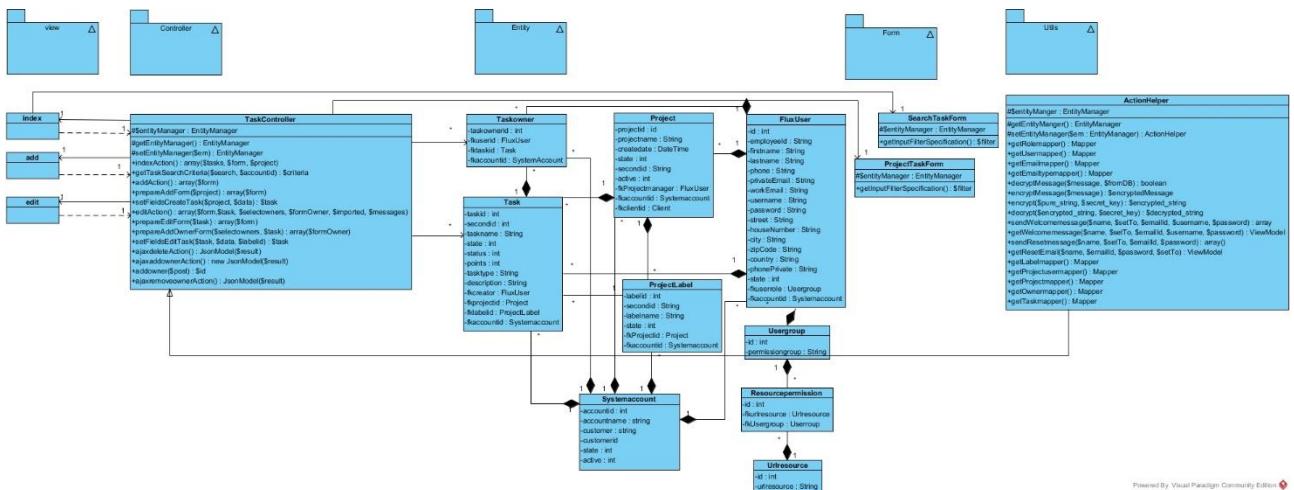
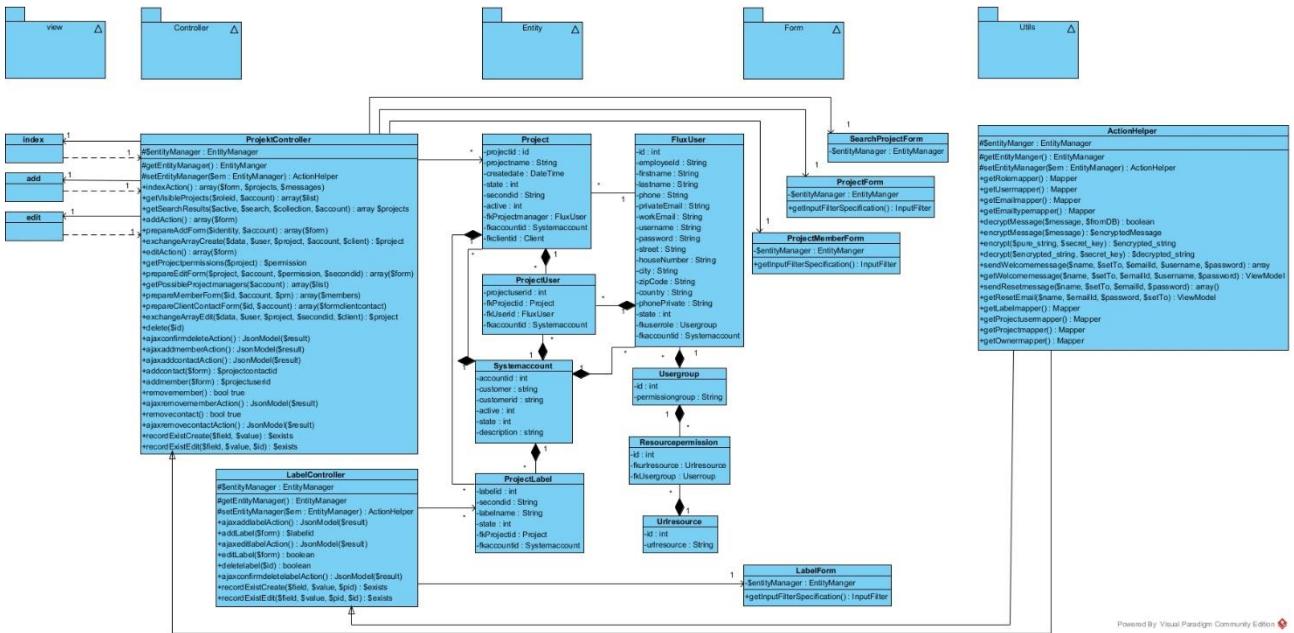
cmd (command prompt)

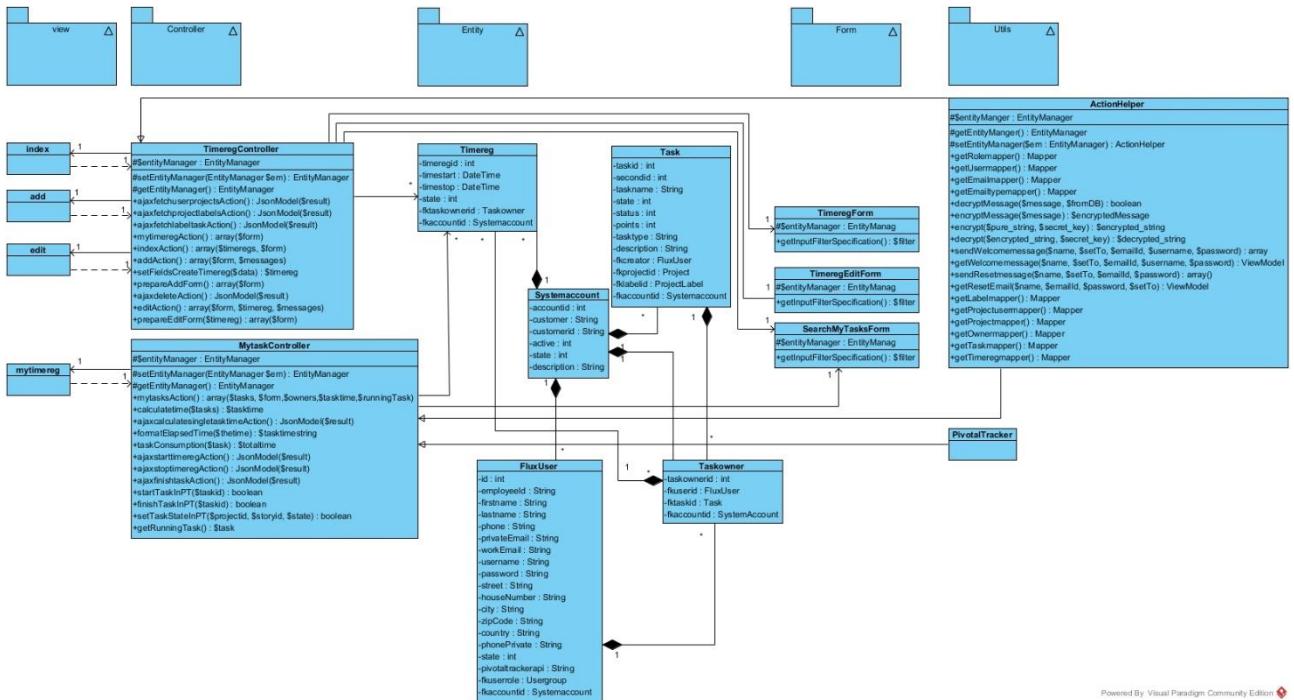
cd \
// Find project (FLUXtime\vendor\bin):
cd xampp\htdocs\FLUXtime\vendor\bin
// For at kopierer alle tabellen til Entity package:
doctrine-module orm:convert-mapping --namespace="Fluxuser\\Entity\\" --force --from-database
annotation ./module/Fluxuser/src/
//Dette skal tilføjes hvis man vil lave en entity (fx. FluxUser):
--filter="FluxUser"
//VIGTIGT: Når alle entities er oprettet skal man rette '\\\' i alle entity-klasser til '\\\' (ved namespace og 2
steder ved hver foreign key)
// For at oprette setter- og getters:
doctrine-module orm:generate-entities ./module/Fluxuser/src/ --generate-annotations=true

```

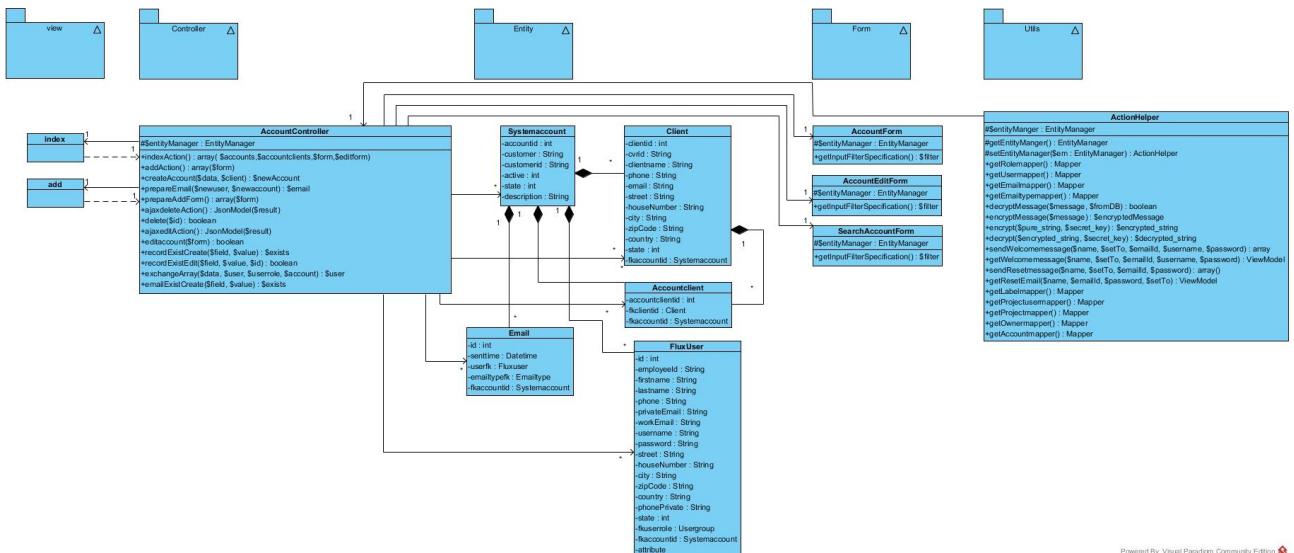
10.5 Design class diagrams



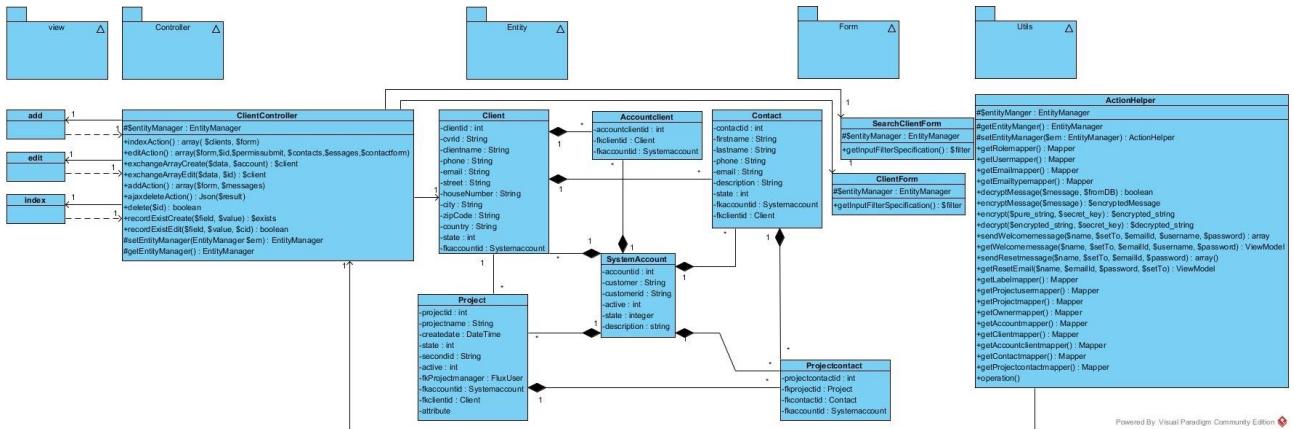




Powered By: Visual Paradigm Community Edition



Powered By: Visual Paradigm Community Edition



Powered By: Visual Paradigm Community Edition