

CDIO FINAL

3 Ugers Projekt Sommer 2015.

02324 Videregående programmering

Projektnavn: *CDIO final*

Gruppe nr: *14*

Afleveringsfrist: *fredag den 19 juni 2015, Kl. 12:00*

Denne rapport er afleveret via Campusnet og i printet kopi, underskrevet.

Denne rapport indeholder **36 sider** inkl. denne side.

S112011	Okholm,	Niko
S132957	Qasim,	Amal
S133983	Saidi,	Favad
S141356	Lindgaard Andersen,	Jacob
S144842	Drewes Rasmussen,	Kim
S144870	Medom Thomassen,	Suzie Kim

Timeregnskab

Dato	Deltager	Design	Impl.	Test	Dok.	Andet	I alt
02/06/2015	Amal	6					6
02/06/2015	Favad	6					6
02/06/2015	Kim	6					6
02/06/2015	Niko	6					6
02/06/2015	Jacob	6					6
02/06/2015	Suzie Kim	6					6
04/06/2015	Amal	5				1	6
04/06/2015	Favad	5				1	6
04/06/2015	Kim	5			1		6
04/06/2015	Niko	5				1	6
04/06/2015	Jacob	5			1		6
04/06/2015	Suzie Kim	5			1		6
05/06/2015	Amal		4		1		5
05/06/2015	Favad		4		1		5
05/06/2015	Kim		4	1			5
05/06/2015	Niko		4	1			5
05/06/2015	Jacob	2				3	5
05/06/2015	Suzie Kim		4		1		5
08/06/2015	Amal		6				6
08/06/2015	Favad		3			3	6
08/06/2015	Kim		6				6
08/06/2015	Niko		3			3	6
08/06/2015	Jacob		3		3		6

08/06/2015	Suzie Kim		6				6
09/06/2015	Amal		5	1			6
09/06/2015	Favad	2	4				6
09/06/2015	Kim		6				6
09/06/2015	Niko		3			3	6
09/06/2015	Jacob		6				6
09/06/2015	Suzie Kim		6				6
10/06/2015	Amal		5	1			6
10/06/2015	Favad		5	1			6
10/06/2015	Kim	1	5				6
10/06/2015	Niko		5	1			6
10/06/2015	Jacob		3		3		6
10/06/2015	Suzie Kim		6				6
11/06/2015	Amal		5	1			6
11/06/2015	Favad		6				6
11/06/2015	Kim		5			1	6
11/06/2015	Niko		4		1	1	6
11/06/2015	Jacob		4			2	6
11/06/2015	Suzie Kim		6				6
12/06/2015	Amal		5				5
12/06/2015	Favad		5				5
12/06/2015	Kim		5				5
12/06/2015	Niko		5				5
12/06/2015	Jacob		5				5

12/06/2015	Suzie Kim		5				5
15/06/2015	Amal		4	1		1	6
15/06/2015	Favad		5	1			6
15/06/2015	Kim		6				6
15/06/2015	Niko		5			1	6
15/06/2015	Jacob		5			1	6
15/06/2015	Suzie Kim		3			3	6
16/06/2015	Amal		7				7
16/06/2015	Favad		6			1	7
16/06/2015	Kim		7				7
16/06/2015	Niko		6			1	7
16/06/2015	Jacob		4		2		7
16/06/2015	Suzie Kim		5	2			7
17/06/2015	Amal		7	2			9
17/06/2015	Favad		9				9
17/06/2015	Kim		8	1			9
17/06/2015	Niko		7	1		1	9
17/06/2015	Jacob		9				9
17/06/2015	Suzie Kim		8			1	9
18/06/2015	Amal		8			1	9
18/06/2015	Favad		8			1	9
18/06/2015	Kim		8	1			9
18/06/2015	Niko		7	1		1	9
18/06/2015	Jacob		6		3		9

18/06/2015	Suzie Kim		8	1			9
19/06/2015	Amal			2		2	4
19/06/2015	Favad		4				4
19/06/2015	Kim		4				4
19/06/2015	Niko		3			1	4
19/06/2015	Jacob		1		2	1	4
19/06/2015	Suzie Kim		1		2	1	4
I alt		71	335	20	22	37	486

Brugsvejledning

Web GUI

Programmet startes, og brugeren vises et login view. Der logges ind med et bruger ID og en kode. Herefter kommer præsenteres man for main viewet; alt efter ens bruger-type vil man ses de kategorier som man har adgang til. Administratoren som har adgang til alt, vil se kategorierne; Produktbatch komponent, Råvarebatches, Produktbatches, Råvare/Ingredienser, Recepter og brugeradministration. Hver af disse har mulighed for at se en listevisning, og med undtagelse af Produktbatch komponent, kan administratoren oprette i alle disse.

Ønskes der at oprette, klikkes på opret, og data udfyldes. Ønskes der at vise eller redigere klikkes på disse. Redigering foregår således at der i listevisningen kan ses en redigér mulighed udfor alle datane.

Der kan ydermere også logges ud af systemet igen.

ASE

ASEN starter selv op når man starter projektet, ASEN er lavet efter afvejningsproceduren, så brugeren bør tage udgangspunkt i denne.

[Timeregnskab](#)
[Brugsvejledning](#)
[Web GUI](#)
[ASE](#)
[Indledning](#)
[Konfigurationsstyring](#)
[Udviklingsplatform](#)
[Analyse](#)
[Domænemodel](#)
[Afvejningsprocedure](#)
[Kravspecifikation](#)
[Usecases](#)
[Bruger administration \(UC1\)](#)
[Råvare administration \(UC2\)](#)
[Recept administration \(UC3\)](#)
[Råvarebatch administration \(UC4\)](#)
[Produktbatch administration \(UC5\)](#)
[Afvejning \(UC6\)](#)
[Autorisering\(UC8\)](#)
[System Sekvens Diagrammer](#)
[BCE-diagram](#)
[Design](#)
[Systemarkitektur](#)
[Service](#)
[DAL, DTO og DAO](#)
[Klassediagram](#)
[Om database](#)
[Views](#)
[Exceptions](#)
[3-lags-model](#)
[Implementation](#)
[Fejl og mangler](#)
[Konklusion](#)
[Bilag 1 : Sekvens diagram af start](#)
[Bilag 2 : Diagram af RåvareAdmin](#)
[Bilag 3 : Analyse klassediagram](#)
[Bilag 4 : Afvejningsprocedure](#)
[Bilag 5 : Tests](#)

Indledning

Denne rapports formål er at oplyse om projektets tilblivelse fra start til ende, så kunden kan erfare, at opgaven er løst på tilfredsstillende vis og lever op til kundens forventninger.

Projekt tager udgangspunkt i oplægget 'CDIO - Final' og rapporten fungerer som gennemgang af vores arbejdsproces, samt en gennemgang af dokumentation for projektet. Rapporten er opbygget kronologisk og skal læses lineært; indledningsvis fra analyse over i design til implementation, test og afrunding.

I dette afsluttende CDIO-projekt er opgaven at udvikle et softwaresystem for en medicinalvirksomhed til afvejninger over et webinterface. Afvejninger og den nødvendige lagerstyring i denne forbindelse ønskes dokumenteret med implementation af en database og forskellige bruger-niveauer, som tillader administration af forskellige ting i databasen igennem webinterfacet. En mere detaljeret kravspecifikation følger i analyse-afsnittet.

Projektet afslutter forløbet i kurset 02324:Videregående Programmering, og det ønskede produkt skal benytte en række værktøjer gennemgået i kurset, samt værktøjer fra andre kurset i dette semester - såsom database, TCP-forbindelse, server-klient kommunikation og Google Webkit Tool (GWT) til den visuelle kommunikation med brugeren i webinterfacet. Det er brugen og spillet mellem disse værktøjer, der ønskes dokumenteret og implementeret i projektet.

Konfigurationsstyring

Udviklingsplatform

Software Platforme anvendt i dette projekt omfatter: MagicDraw til illustration af analyse- og design diagrammer; Google-docs til rapportskrivning; Eclipse (v. Luna 4.4.0) til Java-kode med hvilken vi implementerer SQL og har forbindelse til databasen; JDBC-driver og MySQL til database platformen ; GWT (v. 2.6.1) til udvikling webinterfacet. Brugertestede operativsystemer omfatter Arch Linux, Windows 7 og 8 og OS X Maverick.

Produktionsplatform

Produktionsplatformen inkluderer selvsamme operativsystemer, som der er udviklet i. Java-projektet er komprimeret som ZIP-fil. Det kan importeres som et færdigt projekt i Eclipse ved at udpakke ZIP-filen, og importerer det som "Existing Projects into Workspace". Derpå kan forbindelse til databasen oprettes, projektet køres og interfacet åbnes i en webbrowser.

Versionsstyring

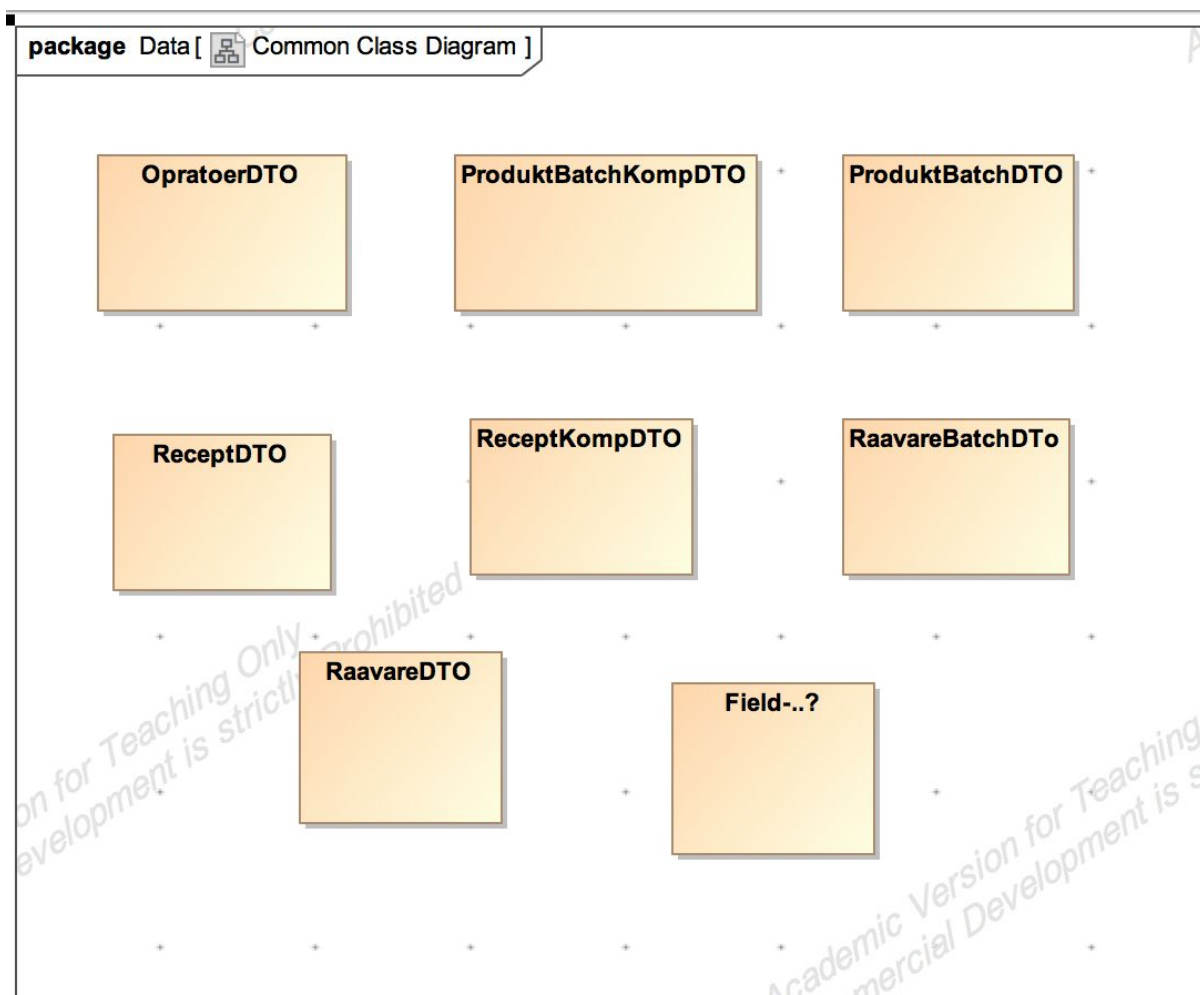
For udviklings- og arkitekturmæssige hensyn, er et GIT-repo blevet brugt. GIT-repository dokumenterer commits til koden, og giver et indblik i, hvordan projektets klasser har udviklet og ændret sig siden start oprettelse

Link til GIT-repo'et findes her: https://github.com/nikookholm/14_cdio_final.git

Analyse

Dette afsnit beskriver analysefasen i projektet: Hvordan vi har indledt vores projekt, analyseret opgaveoplægget, lavet domænemodel, usecases og defineret kravene til programmet. På de følgende sider fokuseres på at modellere, hvad programmet skal kunne i en typisk bruger/system-interaktion, så vi følgende har kunnet arbejde videre med design og implementation.

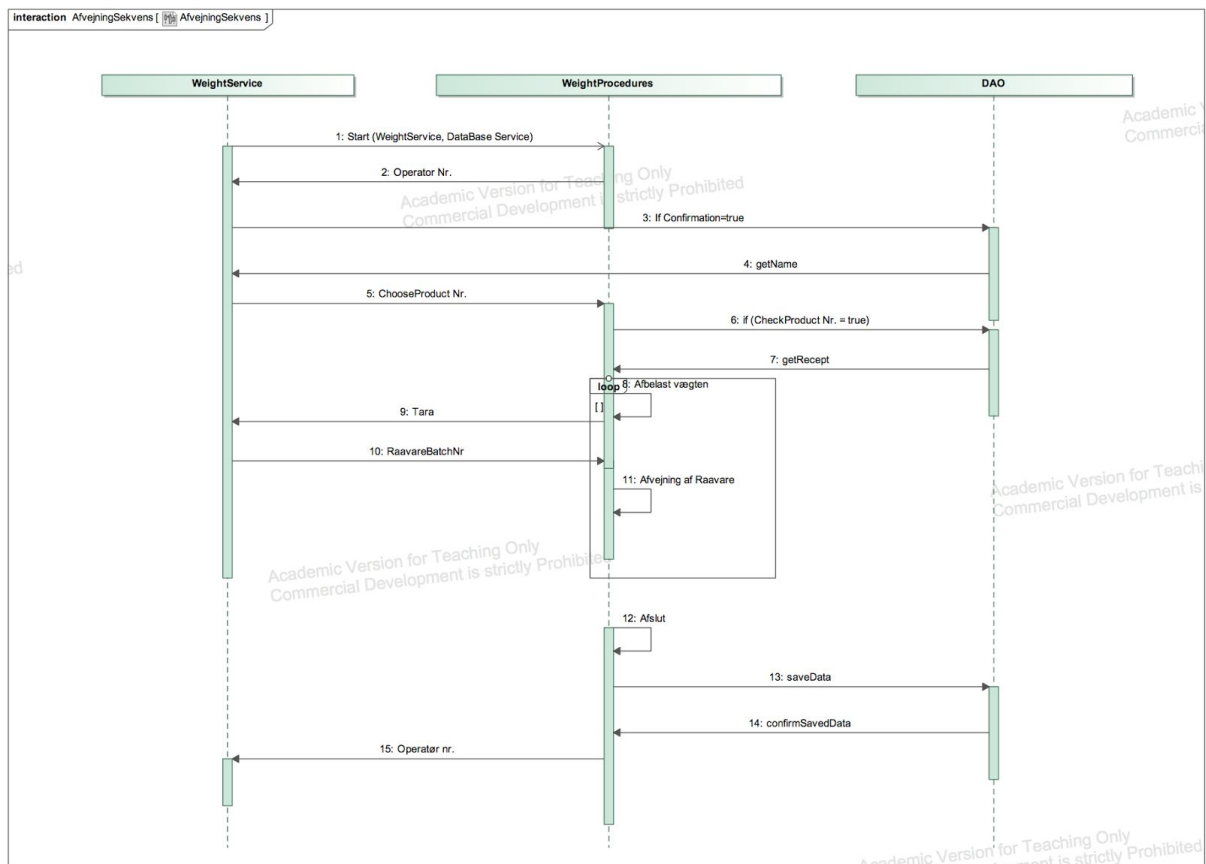
Domænemodel



Ovenstående domænemodel illustrerer DAL (Data Access Layer) med DTO-klasserne (Data Transfer Object) i vores system. Information om objekterne tilgås i disse klasser, der repræsenterer den fysiske forståelse af objekterne

Afvejningsprocedure

Afsnittet omhandler afvejningsproceduren og vores analyse af denne, samt de overvejelser vi har gjort os om den. Proceduren illustreres i et analyse-sekvens-diagram, som viser kald- og svar-sekvenserne i den standardiserede afvejningsprocedure.



Diagrammet viser vores analyse af afvejning procedures opbygning

Weightprocedures

Vores overvejelser på at udvikle et program som kan kommunikere med en vægt og en database, har været at vi opretter en klasse som tildeles både en vægt og en database, måden hvorpå programmet gerne skulle kunne fungere på, er at man logger ind som en operatør, henter en receipt i databasen, og afvejer en råvare på vægten som så angiver vægten på råvaren hvor den derefter gemmer værdien inde i databasen.

Når afvejningsproceduren for en produktbatch er færdig, sættes variabelen status til: 2/ "afsluttet", når afvejningsproceduren er færdig, kan laves en printProduktbatch metode, som kunne printe produktbatchen for operatøren. Dette er dog ikke et specifikt krav, og efter dialog med kunden, har vi har valgt ikke at gå videre med dette aspekt af programmet, da det ikke er signifikant for den overordnede funktionalitet.

Kravs specifikation

Der findes en række umiddelbare krav til projektet med henblik på projektoplægget. Disse krav er opstillet i punktform herunder som non- og funktionelle krav.

Funktionelle krav

1. Der skal laves et **webinterface**, som tillader en **afvejnings-procedure** ligesåvel som **operatør-, råvare-, recept- og produkt-administration** afhængig af **bruger-niveauet**.
 - 1.1. Der skal i den forbindelse oprettes flere Data Access Layers (DALs) til administration af databasen, og adskillelse af databasen fra de andre komponenter i system-arkitekturen.

- 1.1.1. Optræder der fejl i DALen, skal der kastes en exception, som beskriver fejlen til de øvre lag
- 1.2. Der skal gives mulighed for en super-administrator kan oprette, rette, inaktivere(slette), samt vise operatørerne i systemet. CRUD-funktioner.
- 1.3. Der skal være forskellige bruger-niveauer, som tillader forskellig administration.
- 1.4. Der skal benyttes GWT (Google Webkit Tool) til udvikling af webinterfacet.
 - 1.4.1. Webinterfacet skal ydermere benytte sig af login/logud til autorisering af aktøren.
2. Der skal være interfaces, DAOs som deler datalaget, som skal være transient, og gøre brug af passende containerklasser, og have enMySQL-database.
 - 2.1. Systemet skal kunne tilgå databasen i en typisk bruger-interaktion og opdatere de informationer, der er indtastet/afvejet, således at systemet er up-to-date
3. Der skal være input-validering for de DTO-views, der kræver, at man kan oprette/redigere.
4. Alle aktør-typer skal kunne tilgå *afvejningsproceduren*¹, som en konsol baseret vægt simulator, som implementerer de nødvendige funktioner for interaktion og kommando/svar.
 - 4.1. Afvejningsproceduren skal kunne anvende vægtfunktionerne:
 - 4.2. Afvejning skal være inden for definerede tolerancegrad
 - 4.3. Skal ændre status fra "oprettet" til "under produktion" under indtastning af råvare, og ændre status til "afsluttet" når alt er afvejet
 - 4.4. Skal være brugervenlig og intuitiv.
5. Der skal laves givne data-transfer-object(DTO'er), som holder styr på oplysninger. I disse er get og set metoder og en række krav til tjek af oplysningerne i disse objekter.
 - 5.1. For en **operatør**, uafhængig af bruger-niveau skal:
 - 5.1.1. Operatørens id være entydigt og mellem 1 og 99999999
 - 5.1.2. Password være 5 eller 8 tegn langt og bestå af store og små bogstaver, tal og følgende specialtegn; {'.', '-', '_', '+', '!', '?', '='}
 - 5.1.3. Operatørens navn være mellem 2 og 20 karakterer langt
 - 5.1.4. Operatørens initialer være 2 eller 4 karakterer
 - 5.1.5. Operatørens cpr-nr være 10 tal
 - 5.1.5.1. Tilføjes en operatør-rolle til at afgøre, på hvilket niveau operatøren har tilladelse til interagerere med systemet.
 - 5.1.5.2. Tilføjes en boolean *active* variabel, som holder styr på om brugeren er aktiv eller ej, da brugere ikke må slettes.
 - 5.2. For en **råvare** skal:
 - 5.2.1. Råvarens id være entydigt og et brugervalgt tal mellem 1 og 99999999
 - 5.2.2. Råvarens navn være mellem 2 og 20 karakterer langt
 - 5.2.3. Råvarens leverandør være mellem 2 og 20 karakterer langt
 - 5.3. For en **råvarebatch** skal:
 - 5.3.1. Råvarebatchens id være entydigt og et brugervalgt tal mellem 1 og 99999999
 - 5.3.2. Råvarens id passe med råvaren og mellem 1 og 99999999
 - 5.3.3. Råvarebatchens mængde holde styr på lager mængden og være et tal af typen *double*

¹ Bilag 1 - Afvejningsprocedure

- 5.4. For en **recept** skal:
 - 5.4.1. Receptens id være entydigt og mellem 1 og 99999999
 - 5.4.2. Receptens navn være mellem 2 og 20 karakterer langt
- 5.5. For en **receptkomponent** skal:
 - 5.5.1. Recept id og råvare id passe med recept og råvare
 - 5.5.2. Receptkomponentens vægt angives med *nom_netto* og være et tal af typen *double* mellem 0.5 kg-20.0 kg.
 - 5.5.3. Receptkomponentens tolerance være et tal af typen *double* mellem 0.1-10.0 %
 - 5.5.4. En receptkomponent gemmes som en produktbatchkomponent i takt med produktet fremstilles.
- 5.6. For en **produktbatch** skal:
 - 5.6.1. Produktbatchens id være et tal mellem 1-99999999
 - 5.6.2. Produktbatchen recept id stemme overens med recepten den tilhører
 - 5.6.3. Produktbatchen status være **0**, **1** eller **2**, som angiver:
 - 5.6.3.1. 0: ikke påbegyndt
 - 5.6.3.2. 1: under produktion
 - 5.6.3.3. 2: afsluttet
 - 5.6.4. der angives en fjerde dato variable, som nedfældes, når en produktion startes/afsluttes
 - 5.6.5. En oprettet produktbatch, gemmes og udskrives, og sendes til den udvalgte operatør
- 5.7. For en **produktbatchkomponent** skal:
 - 5.7.1. produktbatch id og råvarebatch id stemme overens
 - 5.7.2. produktbatchkomponentens tara være et tal af typen *double* og angive tara i kg
 - 5.7.3. produktbatchkomponentens netto være et tal af typen *double* og angive den afvejede nettomængde i kg.
 - 5.7.4. operatørens id stemme overens og betegne hvilken operatør, der har foretaget afvejningen.
- 6. Aktører skal kunne autoriseres med et login og være i stand til at afslutte en session med logud.

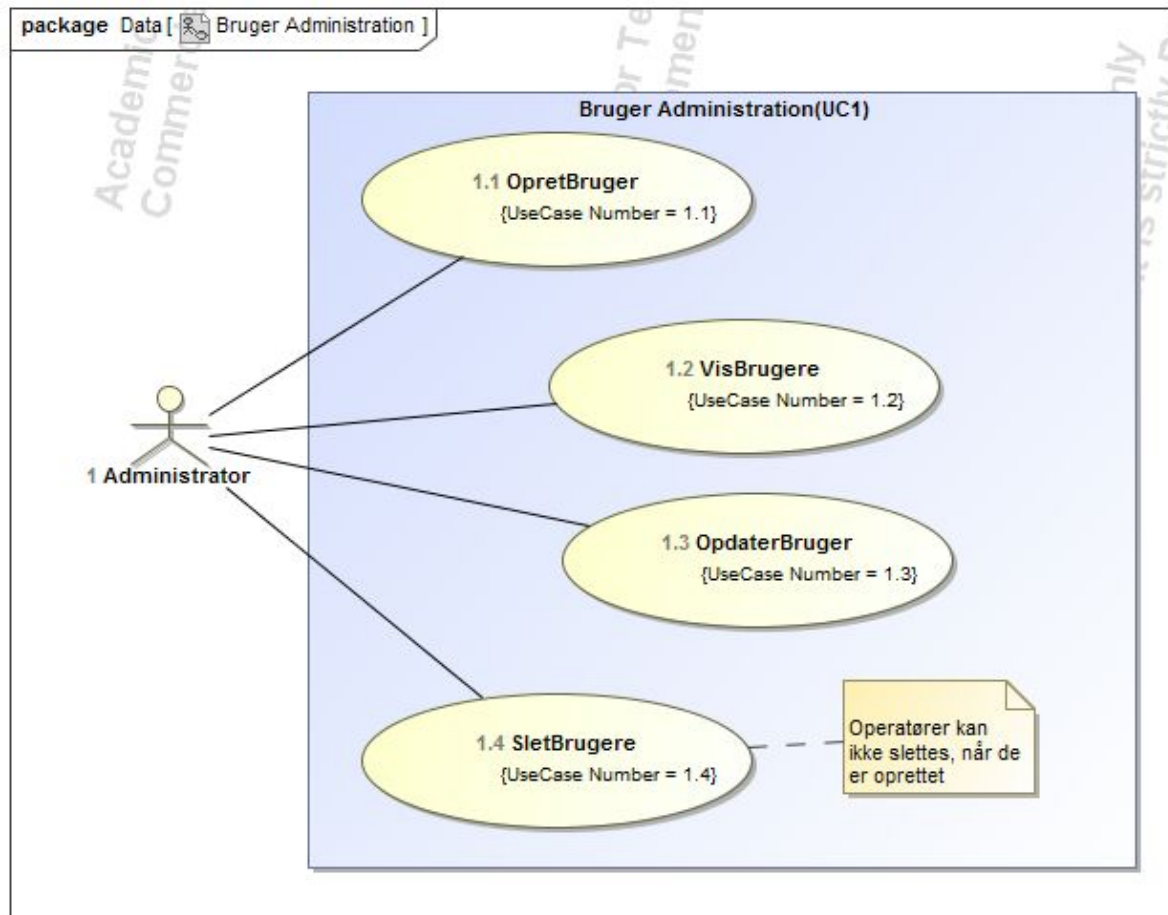
Non-funktionelle krav

- 1. Programmet skal køre på en computer, der har adgang til internettet
- 2. Programmet skal interagere med brugeren og være brugervenligt
- 3. Programmet skal have adgang til databasen.
- 4. Programmet skal være tilsluttet vægten.

Usecases

Ud fra ovenstående kravspecifikation og de angivne usecases i opgaveoplægget, har vi på de følgende mange sider opstillet usecases for aktørens tænkelige interaktioner med systemet.

Bruger administration (UC1)



Use Case Navn	opretBruger
Use Case ID	1.1
Beskrivelse	Opretter brugere(Operatør, Værkfører, Farmaceut)
Primary	<u>Administrator</u>
Secondary	None
Pre conditions	Aktøren er logget ind
Main flow	<ol style="list-style-type: none"> 1. Use-casen starter, når aktøren vælger Opret Bruger fra Brugeradministrations-menuen. 2. Systemet viser de påkrævede felter, der skal indtastes. 3. Aktøren udfylder skemaer. 4. Systemet frigiver "Opret" - knappen 5. Aktøren trykker på "Opret" -knappen 6. Systemet sender oplysningerne til databasen, hvor brugeren bliver oprettet

Post conditions	Brugeren bliver oprettet
Alternative flows	1.1.1 Forkert Indtastning af felterne 1.1.2 CPR-nummer findes i forvejen 1.1.3 bruger ID findes i forvejen

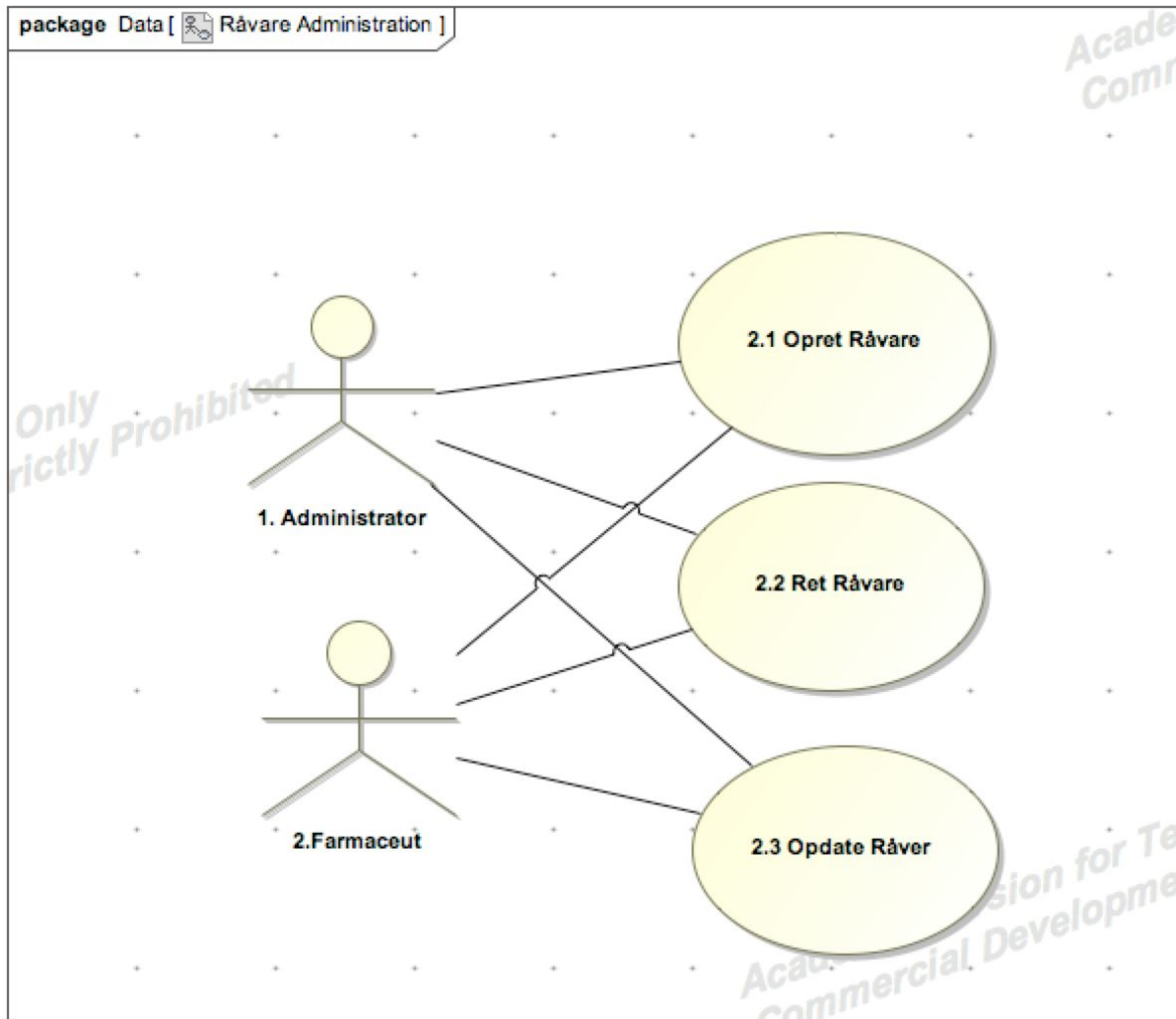
Use Case Navn	Forkert Indtastning af felterne
Use Case ID	1.1.1
Beskrivelse	Felterne bliver farvet rødt så længe der er fejl i udfyldning
Pre conditions	Aktøren er logget ind og står i opretBruger-view
Alternative flows	loop (Så længe at felterne er forkert udfyldt) 3.1 Systemet farver feltet, der er udfyldt forkert, rødt.
Post conditions	Felterne er farvet rødt

Use Case Navn	visBrugere
Use Case ID	1.2
Beskrivelse	Viser samtlige brugere der er oprettet i databasen
Primary	<u>Administrator</u>
Secondary	None
Pre conditions	Aktøren er logget ind og står i brugeradministrations-menuen
Main flow	1. Use casen starter når aktøren klikker på "Vis Brugere" knappen. 2. Systemet henter listen med brugere fra databasen, med en "Opdater" hos alle brugere, og en "Slet" knap hos alle undtagen operatører
Post conditions	Listen over brugere bliver fremvist til aktøren
Alternative flows	None

Use Case Navn	opdaterBruger
Use Case ID	1.3
Beskrivelse	kan redigere data om eksisterende bruger i systemet

Primary	<u>Administrator</u>
Secondary	None
Pre conditions	Aktøren er logget ind og er i "Vis Brugere" menuen
Main flow	<ol style="list-style-type: none"> 1. Use casen starter, når aktøren klikker på "Opdater Bruger" i "Vis Brugere" menuen, hos den bruger der ønskes opdateres 2. Systemet viser de felter, der skal kunne opdateres, hvilket er: navn, initialer og password. 3. Aktøren udfylder felterne 4. Systemet frigiver "Opdater" knappen 5. Aktøren klikker "Opdater" knappen 6. Systemet forbinder til databasen, og opdaterer den ønskede brugere
Post conditions	Brugeren er opdateret, og databasen efterlades konsistent
Alternative flows	1.1.1 Forkert Indtastning af felterne

Råvare administration (UC2)



Use Case Navn	opretRaavare
Use Case ID	2.1
Beskrivelse	Aktøren kan oprette en ny råvare.
Primary	<u>Farmaceut</u> , Administrator
Secondary	None
Pre conditions	Aktøren er logget ind

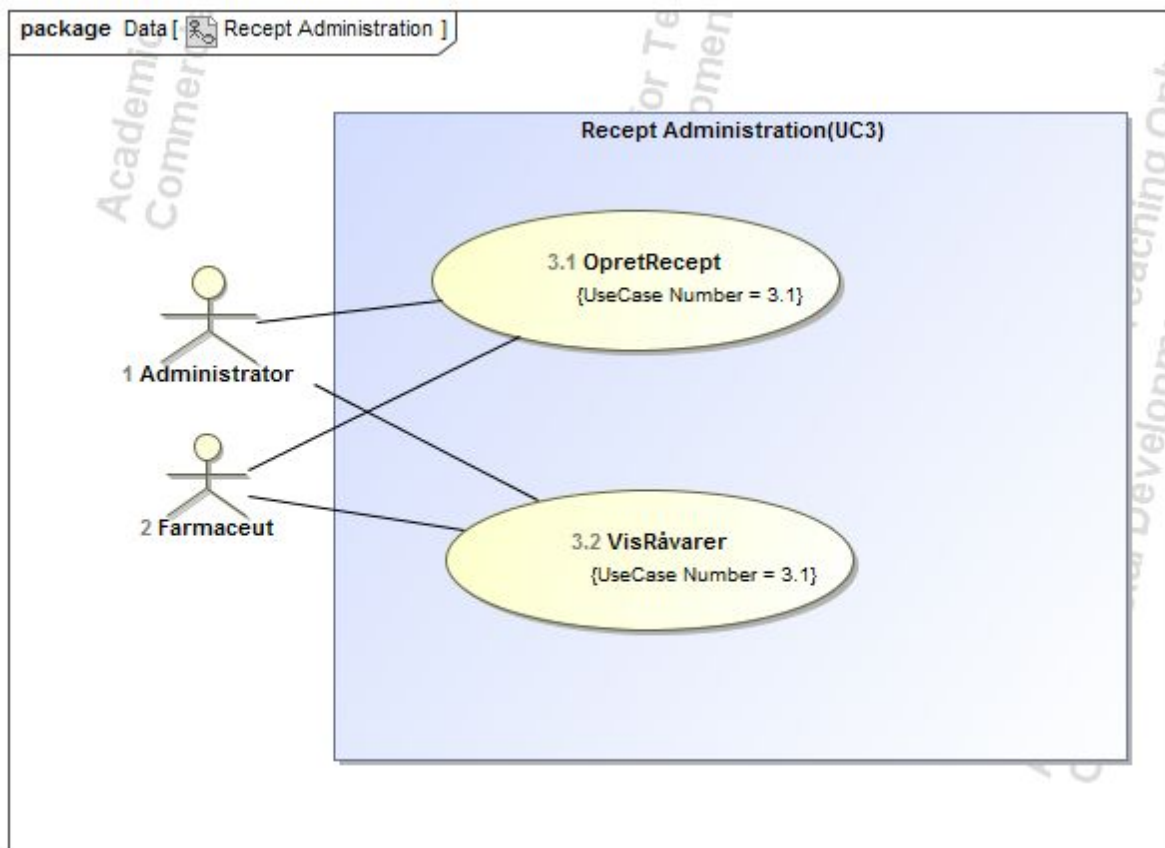
Main flow	<ol style="list-style-type: none"> 1. Use Casen starter, når aktøren vælger "Opret Råvare" fra råvare-administrations-menuen 2. Systemet viser de påkrævede felter, der skal indtastes som 3. input 4. Aktøren udfylder de påkrævede felter 5. Systemet frigiver "Opret" knappen 6. Aktøren klikker "Opret" knappen 7. Systemet opretter råvaren i databasen, og giver aktøren besked om at råvaren er oprettet.
Post conditions	Råvaren er oprettet i databasen og aktøren befinder sig tilbage i menuen, som var udgangspunktet.
Alternative flows	1.1.1 Forkert Indtastning af felterne

Use Case Navn	visRaavare
Use Case ID	2.2
Beskrivelse	Aktøren kan få vist en liste over råvarer i databasen
Primary	<u>Farmaceut</u> , Administrator
Secondary	None
Pre conditions	Aktøren er logget ind
Main flow	<ol style="list-style-type: none"> 1. Use Casen starter, når aktøren vælger "Vis Råvarer" fra råvare-administrations-menuen 2. Systemet viser en liste over råvarer i databasen, ud for hver råvare vises der en "Opdater" knap
Post conditions	Listen over råvarer er vist
Alternative flows	None

Use Case Navn	opdaterRaavare
Use Case ID	2.3
Beskrivelse	Aktøren kan redigere informationer om en råvare i databasen
Primary	<u>Farmaceut</u> , Administrator
Secondary	None
Pre conditions	Aktøren er logget ind

Main flow	<ol style="list-style-type: none"> 1. Use casen starter, når aktøren klikker på “Opdater Råvare” i “Vis Råvarer” menuen, på den råvare, der ønskes opdateres 2. Systemet viser de felter, der skal kunne opdateres, hvilket er: råvare ID, råvareNavn og leverandør 3. Aktøren udfylder felterne 4. Systemet frigiver “Opdater” knappen 5. Aktøren klikker “Opdater” knappen 6. Systemet forbinder til databasen, og opdaterer den ønskede råvare
Post conditions	Råvaren er opdateret i databasen og aktøren befinder sig tilbage i menuen, som var udgangspunktet.
Alternative flows	1.1.1 Forkert Indtastning af felterne

Recept administration (UC3)

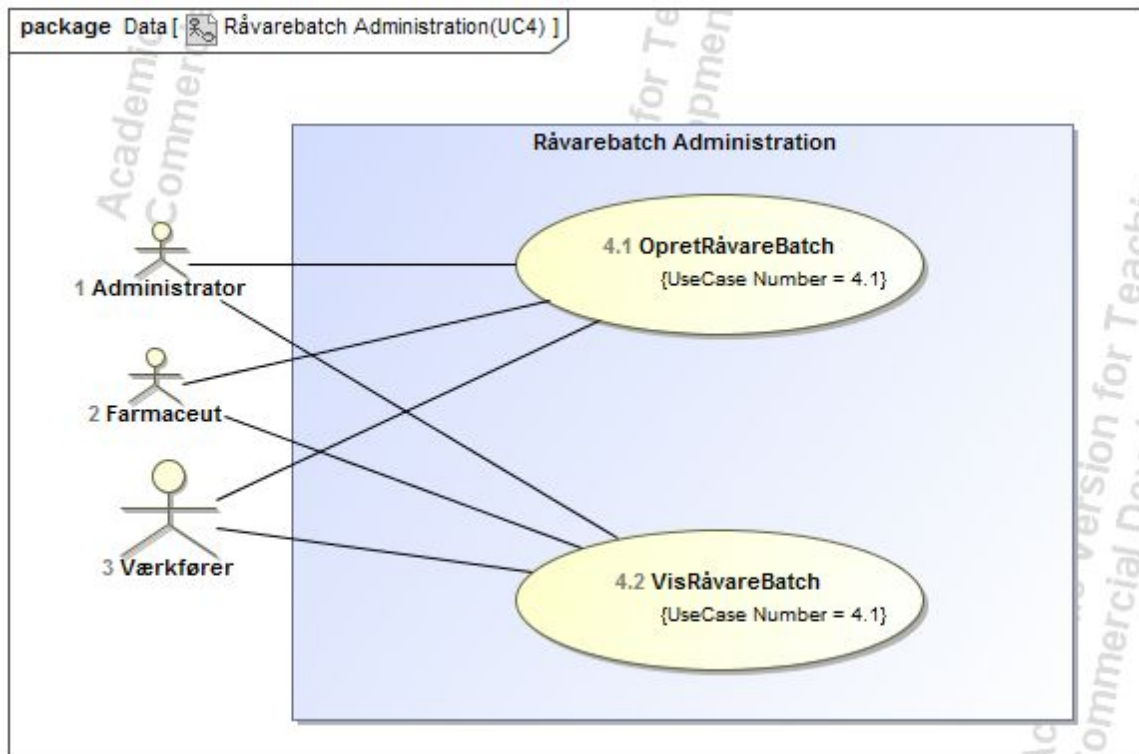


Use Case Navn	opretRecept
Use Case ID	3.1
Beskrivelse	Aktøren kan oprette en ny recept.

Primary	<u>Farmaceut</u> , Administrator
Pre conditions	Aktøren er logget ind
Main flow	<ol style="list-style-type: none"> 1. Use Casen starter, når aktøren vælger "Opret Recept" fra recept-administrations-menuen 2. Systemet viser de påkrævede felter, der skal indtastes som input. 3. Aktøren udfylder de påkrævede felter 4. Systemet frigiver "Opret" - knappen 5. Aktøren trykker på "Opret" -knappen 6. Systemet sender oplysningerne til databasen, hvor recepten bliver oprettet
Post conditions	Recepten er oprettet i databasen, og aktøren befinder sig tilbage i menuen, som var udgangspunktet.
Alternative flows	1.1.1 Forkert Indtastning af felterne

Use Case Navn	visRecept
Use Case ID	3.2
Beskrivelse	Aktøren kan få vist en liste over recepter i databasen.
Primary	<u>Farmaceut</u> , Administrator
Secondary	None
Pre conditions	Aktøren er logget ind
Main flow	<ol style="list-style-type: none"> 1. Use Casen starter, når aktøren vælger "Vis Recept" fra recept-administrations-menuen 2. Systemet viser en liste over recepter i databasen
Post conditions	Systemet har vist en liste over recepter.
Alternative flows	None

Råvarebatch administration (UC4)

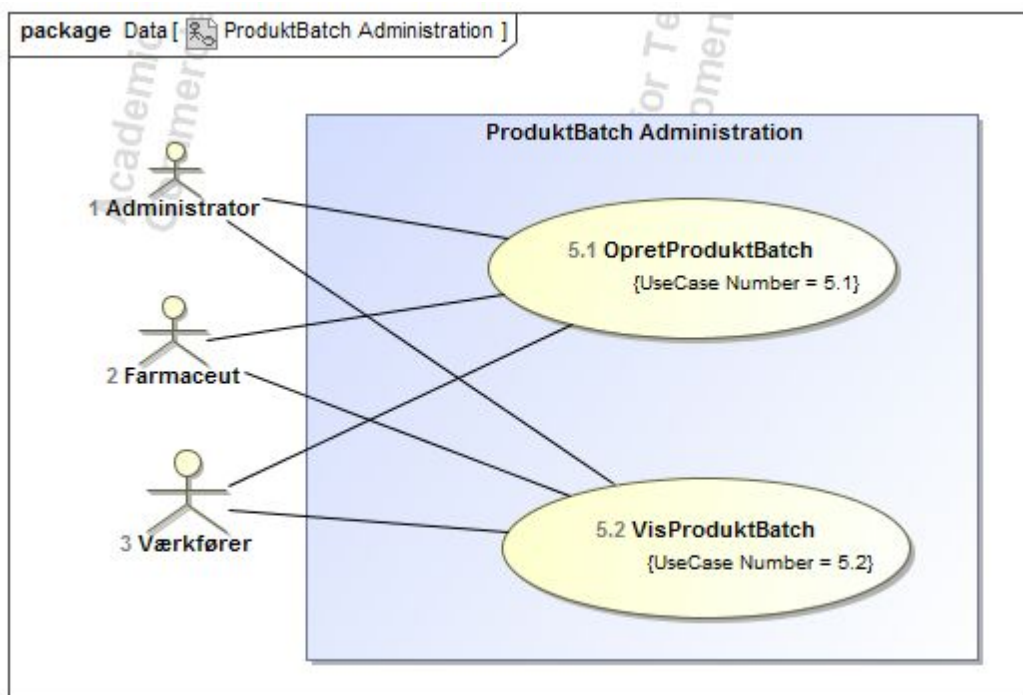


Use Case Navn	opretRåvareBatch
Use Case ID	4.1
Beskrivelse	Opretter råvarer
Primary	<u>Værkfører</u> , Administrator, Farmaceut
Secondary	None
Pre conditions	Aktøren befinder sig i råvareBatch-menuen
Main flow	<ol style="list-style-type: none"> 1. Use Casen starter når aktøren vælger "Opret Råvare" i råvareBatch-menuen. 2. Systemet viser de påkrævede felter, som der skal udfyldes² 3. Aktøren udfylder felterne. 4. Systemet frigiver "Opret" knappen 5. Aktøren klikker "Opret" 6. Systemet sender oplysningerne til databasen, hvor råvare-batchen bliver oprettet
Post conditions	Råvaren er oprettet i databasen.
Alternative flows	1.1.1 Forkert Indtastning af felterne

² RåvarebatchNr. skal være et brugervalgt id.

Use Case Navn	visRaavarebatch
Use Case ID	4.2
Beskrivelse	Viser en liste over råvare batches der findes i databasen
Primary	<u>Værkfører</u> , Administrator, Farmaceut
Secondary	None
Pre conditions	Brugeren er logget ind, og befinder sig i Råvareadministrations-menuen
Main flow	<ol style="list-style-type: none"> 1. Use casen starter når aktøren vælger "Vis Råvarebatches" i råvareadministrations-menuen. 2. Systemet forbinder til databasen, og fremviser en liste over eksisterende råvarebatches for aktøren
Post conditions	En liste over råvarebatches er fremvist
Alternative flows	None.

Produktbatch administration (UC5)

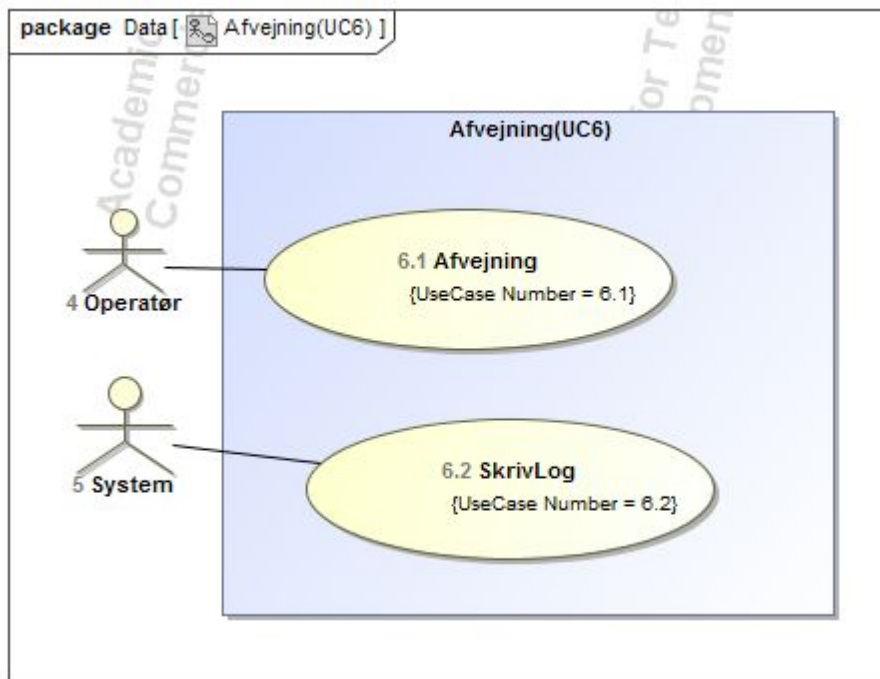


Use Case Navn	OpretProduktBatch
---------------	-------------------

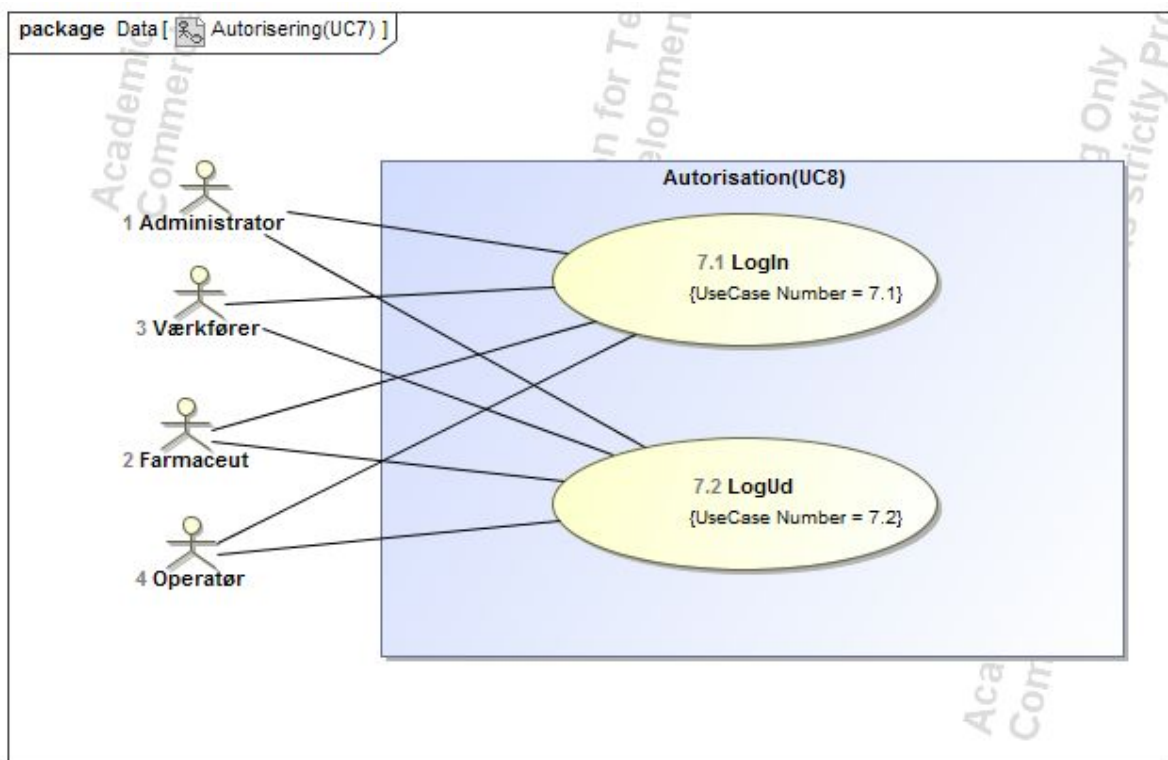
Use Case ID	5.1
Beskrivelse	Værkføren skal oprette en ny produktbatch
Primary:	<u>Værkfører</u> , Administrator, Farmaceut
Secondary:	None.
Pre conditions	Aktøren er logget ind i systemet, og befinder sig i produktbatch-administrations-menuen
Main flow	<ol style="list-style-type: none"> 1. Use casen starter når aktøren klikker på "Opret ProduktBatch" i produktbatch-administrations-menuen 2. Systemet fremviser de påkrævede felter. 3. Aktøren udfylder felterne 4. systemet frigiver "Opret" knappen 5. Aktøren klikker "Opret" Knappen 6. Systemet forbinder til databasen, og opretter en ny produktbatch. Systemet printer produktbatchen ud.
Post conditions	Der er oprettet en ny produktbatch, med status 0 ("Oprettet")
Alternative flows	1.1.1 Forkert Indtastning af felterne

Use Case Navn	VisProduktBatches
Use Case ID	5.2
Beskrivelse	Værkføren kan se hvilke produktbatches, der er lavet
Primary:	<u>Værkfører</u> , Administrator, Farmaceut
Secondary:	None.
Pre conditions	Aktøren er logget ind i systemet, og befinder sig i produktbatch-administrations-menuen
Main flow	<ol style="list-style-type: none"> 1. Use casen starter, når aktøren klikker på "Vis Produktbatches" 2. Systemet forbinder til databasen, og viser de eksisterende produktbatches.
Post conditions	En oversigt af produktbatches er fremvist
Alternative flows	None.

Afvejning (UC6)



Autorisering(UC8)

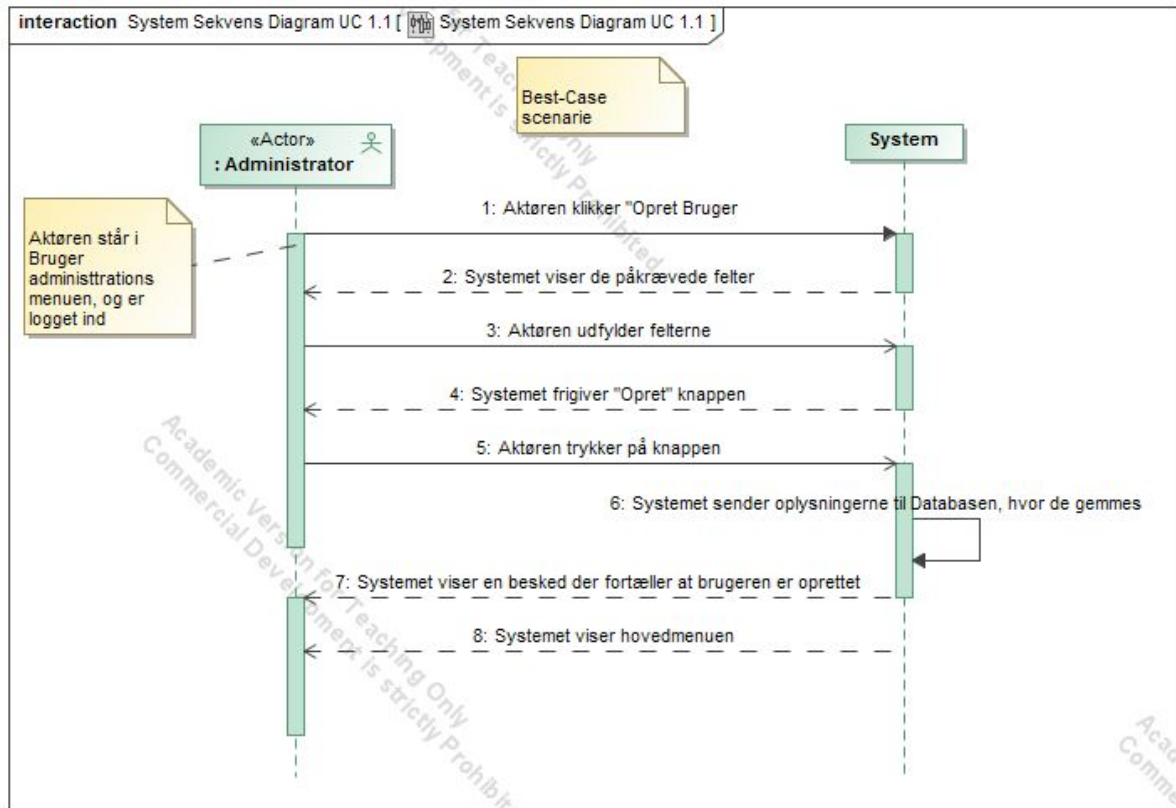


Aktør: Administrator, Farmaceut, Værkfører & Operatør

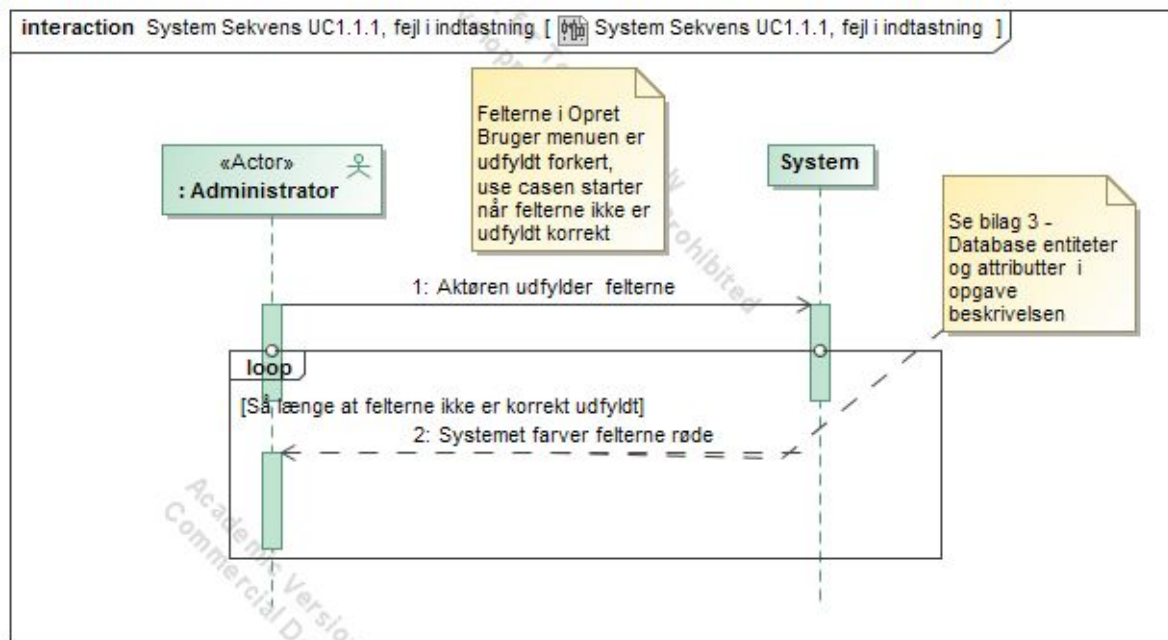
System Sekvens Diagrammer

Følgende afsnit omhandler vores analysearbejde af de udarbejdede use-cases. Diagrammerne er tiltænkt, til at kunne få overblik over, hvordan brugeren interagerer med vores system.

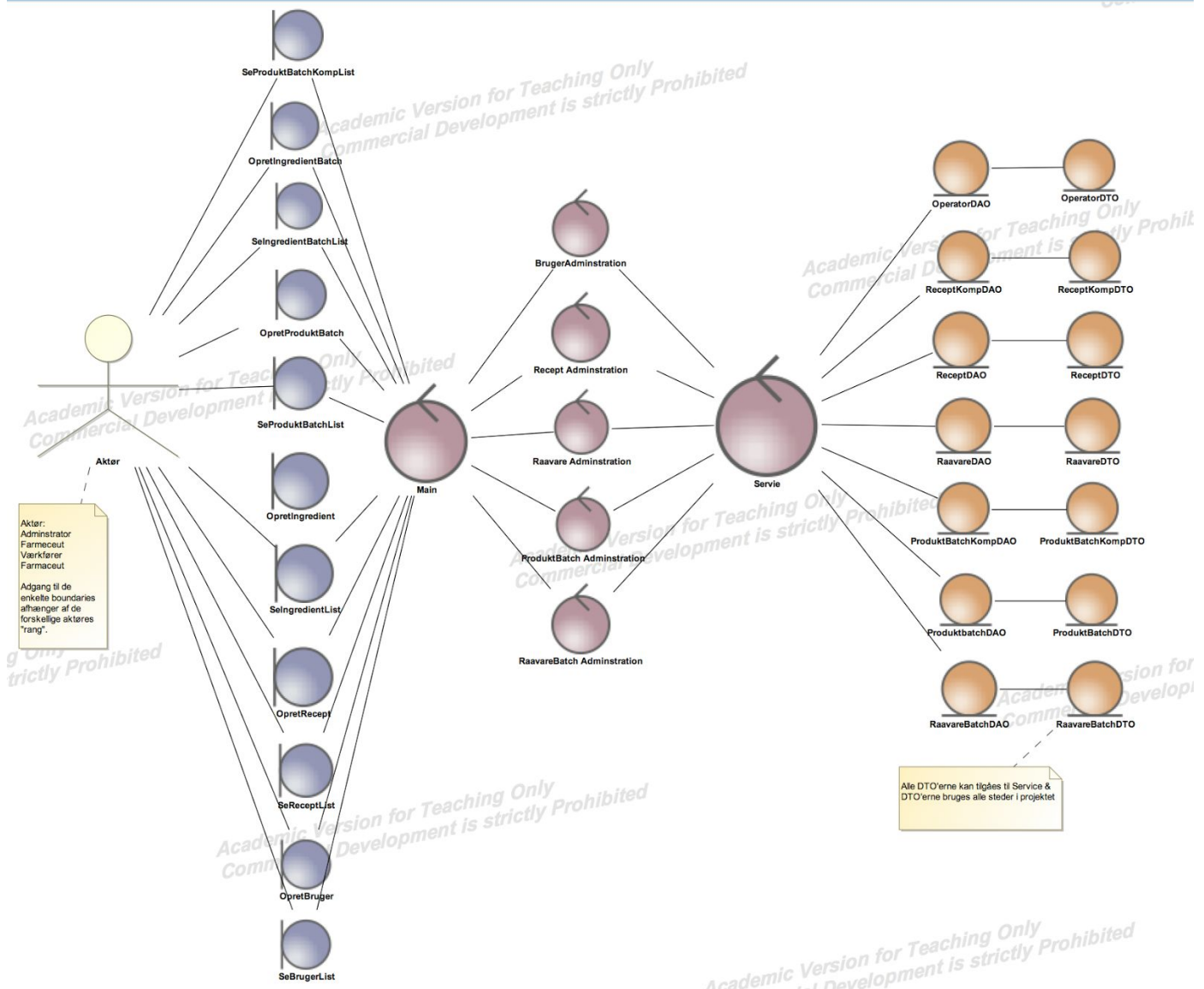
Use-case 1.1



Use-case 1.1.1 - Alternative Flow - Forkert Indtastning af brugeroplysninger



BCE-diagram



Ovenstående Boundary-Controller-Entity-diagram illustrerer opbygningen af vores program. Den er konstrueret tidligt i projektet, så kunden må tage forbehold for ændringer i klassenavne. Fx hedder administrations-Controller-klasserne, controller. Grænsefladen og Views har klasser for list/edit slået sammen.

Ikke så tilfældigt, illustrerer diagrammet også meget godt 3-lags-model konceptet i vores program. Entity-klasserne dækker over alle objekter, som skal transporteres videre i systemet. Informationerne sendes videre gennem Service-controlleren, alt imens Controllerne tager sig af funktioner og metoder til deres DTO og sender det videre op til MainController, som kommunikerer med grænsefladen og vores Views og præsenterer informationen for brugeren.

Design

Dette afsnit omhandler designet af vores program. På de følgende sider gives en redegørelse over de forskellige komponenter, der har været nødvendige for at tilfredsstillende produkt

Systemarkitektur

Systemet består af følgende komponenter, som vi redegør for på de følgende sider:

- En database med oplysninger om *operatører, råvarer, råvarebatches, recepter*, samt planlagte og færdigproducerede *produktbatches*.
- Et webinterface (GUI), hvor farmaceuten kan administrere *recepter* og *råvarer*, og hvor værkføreren kan oprette *produktbatch*, ssamt se en status over disse.
- En række vejeterminaler udstyret med vejeplade, tastatur og display.
- Et program (ASE, se afsnit 4.2), som vi selv skriver, der styrer alle vejeterminalerne og gemmer data fra disse i databasen.
- Datalaget i applikationen er implementert som en MySql database, der som minimum bør være på 3. normalform.

Service

Vi har implementeret Service-klasser til at håndtere forbindelse til WCU'en og databasen. Initialt ligger de enheder uden for umiddelbare programmets rækkevidde, men med en Service-klasse som extender RemoteService tillades ekstern adgang for vores program til WCU og database. WeightService og Async implementerer funktionerne for vægten og tillader asynkrone callbacks til WCU. Databaseservice og Async sørger for at implementere de funktioner vi skal bruge til database-implementation og tillader asynkrone callbacks tilbage databasen, så den er opdateret. Servicene starter op med applikation og starter forbindelsen til databasen og WCU.

DAL, DTO og DAO

Data Access Layer (DAL), er laget der befinder sig mellem applikationen og databasen. Laget består af to dele, et Data Access Objekt (DAO) og en Data Transfer Objekt (DTO). Formålet med DAL er at adskille applikationen og databasen, og have al logik placeret i dette lag.

DAOernes funktion er at skabe adgang informationerne gemt i databasen vha SQL forespørgsler. Interfacene DAO har til opgave at hente tabeller og tupler til visning for brugeren, samt opdatere tupler, hvor der er felter der skal ændres. Slette og lave nye tupler i tabellerne. Disse omfatter CRUD-funktionerne³, som vores system skal gøre brug af. Her følger en gennemgang af vores UserDAO: Interfacet sørger for adgang til operatørens data i DTO. En systemadministrator skal kunne oprette, opdatere og slette operatører, og se en liste over samtlige operatører, der findes i databasen. En operatør - uafhængig af rolle - identificerer sig selv overfor systemet med et *ID* og et *password* for at kunne logge ind og foretage funktioner tilpasset hans brugerrolle. Derudover eksisterer information om brugernes *navn*, *CPR*, *rolle* og om han/hun er *aktiv/inaktiv*.

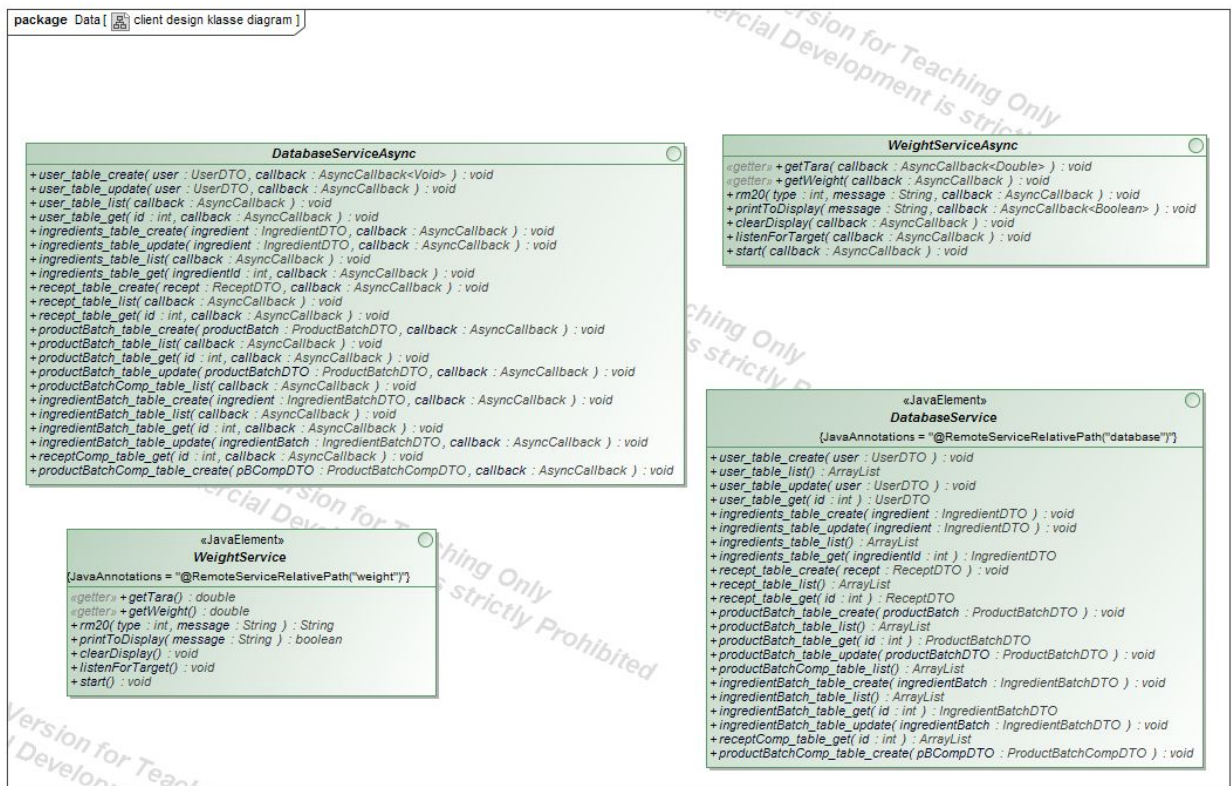
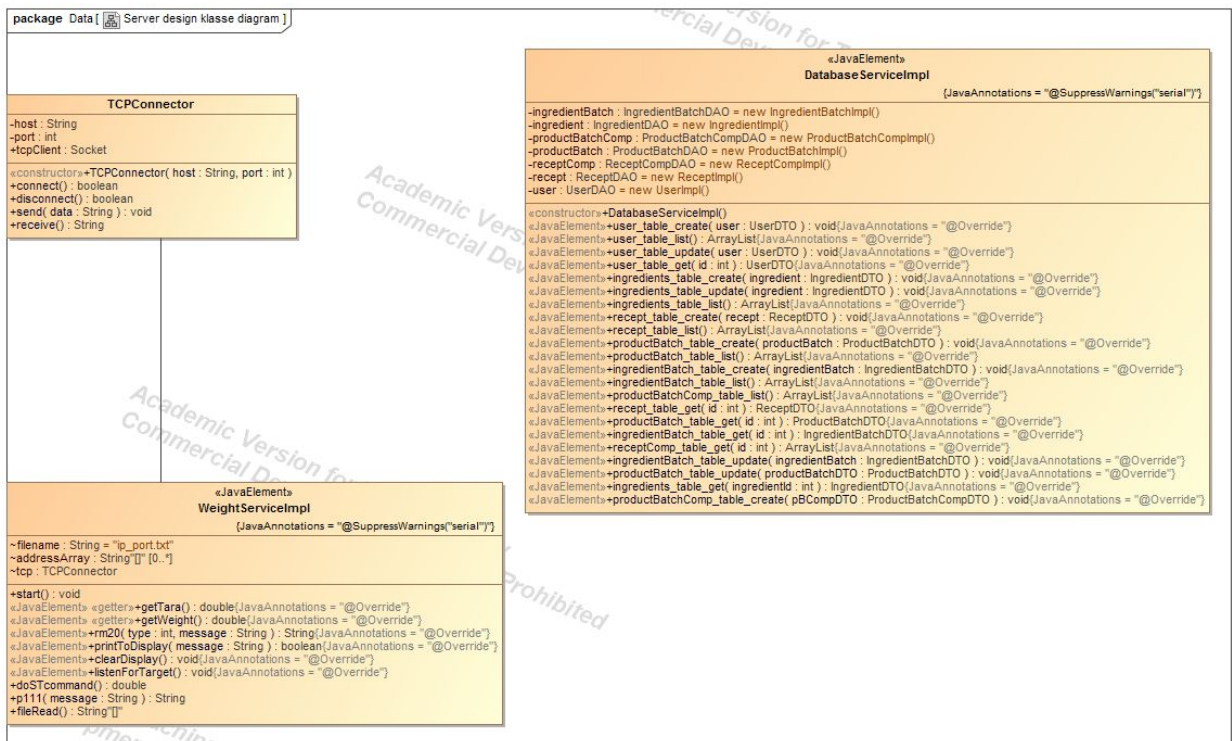
For let at kunne manipulere med dataene i de forskellige tabeller, har vi et DTO for hver enkelt tabel, som fungerer som et java objekt der repræsenterer tuplerne i tabellerne. Informationerne bliver hentet ind fra databasen, vha SQL forespørgsler, og gemt i et DTO, hvis informationerne fra en tuppel bliver opdateret vil attributterne fra DTO, blive gemt i databasen. Dette giver os en form for objekt persistens. Vi har følgende DTO'er i vores projekt:

³ Create, Read, Update, Delete

UserDTO, IngredientDTO, IngredientBatchDTO, ReceiptDTO, ReceiptCompDTO, ProductBatchDTO, ProductBatchCompDTO.

Klassediagram





Samtlige diagrammer findes i læselige udgaver i eclipse projektet

Om database

På grund af projektets begrænsede omfang er der nogle rent tekniske ting, vi har kunne komme let rundt om eller helt undlade, men vi vil alligevel give en kort beskrivelse af hvilke ting, man bør tage højde for, når man arbejder med et database projekt.

I dette projekt har vi udelukkende arbejdet med databasen placeret lokalt på vores maskiner for lettere omskrivning til vores egne behov, trods muligheden for en server-database også har været der. Denne har vi så benyttet i vores endelige implementation. Vi har haft SQL-dump af databasen liggende i projektmappen, som løbende er blevet opdateret, når det har været brug for det. På den måde har vi sikret, at databaserne er identiske, når vi kører og arbejder med dem. Foruden hvor og hvordan man "hoster" sin database, bør man også gøre sig overvejelser om sikkerhed og evt. pladsforbrug og forbindelses begrænsninger

Sikkerhed i databasen har på grund af projektets natur heller ikke været noget, vi skulle tage hånd om. I et større projekt med mange brugere, bør man dog gøre sig nogle overvejelser om rettigheder til databasemiljøet. Det er god skik at give udviklere, administratorer og brugere forskellige rettighedsniveauer for at undgå fejl og udnyttelse af systemet. Vi har tilføjet variablen *rolle* til UserDTO, som holder styr på, hvad brugeren har lovet til at se og gøre med systemet efter login.

I projektet har vi ikke arbejdet med stored procedures, som er sætninger der er gemt i databasen og, som bl.a. kan kontrollere, og sørge for at data bliver sat ordentligt ind i databasen, uden at applikationen der kalder databasen nødvendigvis kender denne. Vi har ikke implementeret dette, da størrelsen af databasen er beskeden, og vi derfor ikke har fundet dette relevant.

Views

Views omfatter alt det GWT, vi har implementeret i projektet, som kommer til udtryk i vores web-interface. Der bruges views til visning af tabeller, opdatering og oprettelse. Views er den grafiske interaktion, der sørger for at vejlede brugeren til retmæssig brug af webinterfacet. GWT spiller sammen med HTML på websiden som javaScripts.

Exceptions

I forbindelse med arbejdet på at lave interfaces til kommunikation til SQL kan der opstå en række exceptions. For at håndtere disse korrekt, kigges der på hvilken type exceptions der kan komme til at opstå i processen, og hvordan vi vil behandle dem.

Det forvente at der skal kunne håndteres SQLException. Disse opstår blandt andet hvis man angiver ukorrekte tabel eller feltnavne, eller hvis der opstår fejl ved forbindelsen. Derudover vil vi bruge den exception der hedder DALEXception til at behandle custom exceptions. Dette kunne være exceptions som bliver rejst på baggrund af vores SQL forespørgsler. For eksempel vil det være smart at rejse en DALEXception hvis vores forespørgsler ikke returnere noget resultat eller at resultatet er uden for de forventede værdier. Exceptions bliver kastet videre af de metoder de findes i, så de kan behandles på højere niveau i den software-arkitektur de bliver implementeret i.

3-lags-model

Vi har forsøgt at overholde 3-lags modellen i implementation af vores projekt. DTO og deres tilhørende interface DAO omfatter datalaget. Vores Controllere agerer funktionalitets-laget, mens vores Views optræder som grænsefladen.

GRASP

General Responsibility Assignment Software Patterns er en udviklingsstrategi som omhandler forskellige håndtaringsmønstre; høj samhørighed og lav kobling i objekt-orienteret programmering. Vi har i dette projekt for så vidt muligt forsøgt at udvikle efter *grasp*, idet det burde give mere stabil og robust kode.

Dog er der et punkt hvorpå dette ikke er overholdt, navnlig lav kobling; alle *views* er i stridigheder hér, ved at tage parameteren *MainController*. Altså er disse direkte afhængige af *MC*, og hvis denne ændres eller fejler på nogen vis vil samtlige *views* ikke fungere.

Implementation

Da programmet er en omfattende implementation af mange forskellige klasser, interfaces, views, GWT, SQL og AsyncCallback, som alle skal spille sammen, for at det virker ordentlig, går vi ikke i detaljer med nogle af disse elementer i dette afsnit. Til den forestående eksamen i kurset 02324 vil vi derimod fremvise en mere specifik gennemgang af enkelte implementations-dele, som er helt centrale og nye for dette semester.

Test

I dette projekt er det blevet testet formelt set med JUnit test. Vi har løbende foretaget brugertests på vores views, at de rent faktisk opfører sig som de bør, både med forkerte og sande input, så systemet hele tiden checker, at input er på en form, som tillades for DTO'erne og databasen. I bilagene kan ses en række af billeder for at dokumentere vores JUnit-tests validitet.

Fejl og mangler

Da gruppen var i tidspres op til aflevering, er der en række ting, vi ikke har været i stand til korrekt at implementere i projektet, hvorfor vi adresserer dem i dette afsnit. Det er dog ikke vitale ting for programmets funktionalitet, men rettere information til kunden om, hvilke krav, der ikke er blevet imødekommet eller vurderet af projektgruppen til at være insignifikante.

1. ListUsersView kan ikke redigere en bruger. Det kunne den tidligere i projektforsøget, men er undervejs blevet korrupt inden aflevering.
2. Vi har prøvet at tage hånd om Exception-handling i WCU'en, men det ryger ud af vægten som en P111-besked, hvilket medfører, at relevante skridt i afvejningsproceduren skal køres igen.
3. Nogle views opfører sig ustabil. Det kan skyldes
4. Projektet kunne klart have et lavere niveau af kobling og dermed følge GRASP i højere grad. Det kunne også have medvirket til færre fejl i vores versionsstyring og synkronisering af projektet, hvilket har været et problem i de sidste dage op til aflevering. Dette havde også medført en højere grad af genbrugelighed.
5. SQL-forespørgsler er implementeret korrekt i databasekald-klasserne, men vi kunne have undgået mange problemer med kald igennem views med Stored Procedures.

I forhold til kommunikation med WCU'en, har der været en del problemer, som først blev løst til sidst i projektets forløb. Der har været stor efterspørgsel på vægten blandt alle projektgrupper i de afsluttende dage op til aflevering og simulatoren, som skulle fungere som substitute for den fysiske version, har ikke været særlig pålidelig. Det har medført en del ventetid på at verificere

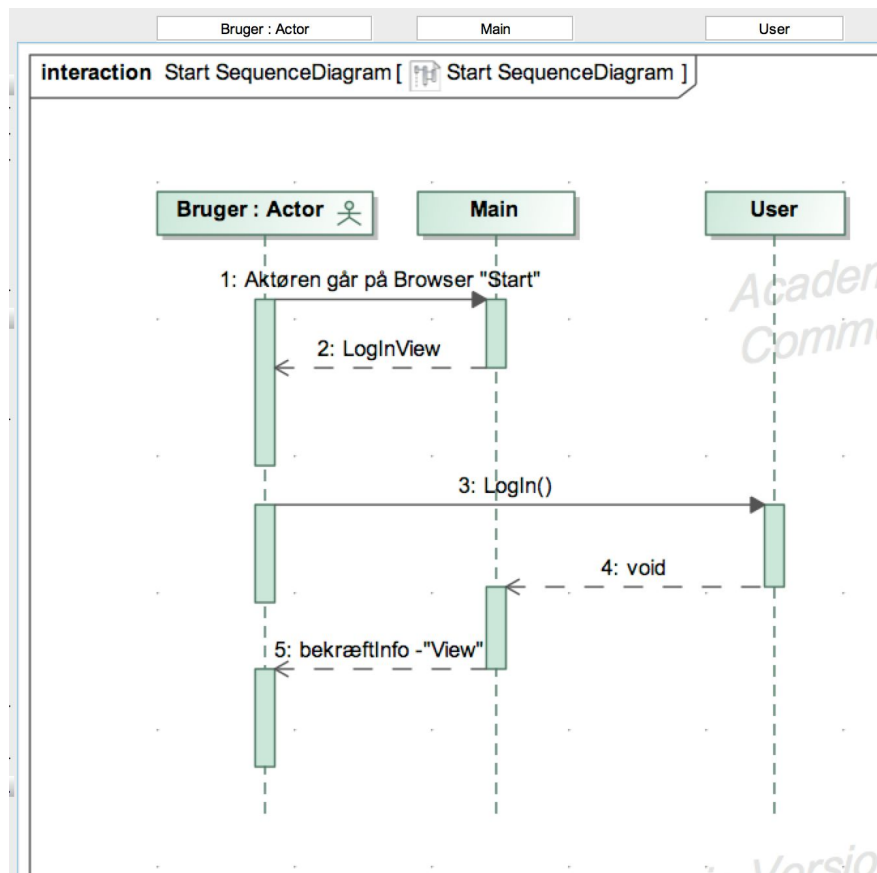
vores kode, hvilket har påvirket vores effektive arbejdstid på nogle områder. Skønsmæssige forbedringer og kommentering af kode, designdiagrammer og rapport har nogle mangler.

Konklusion

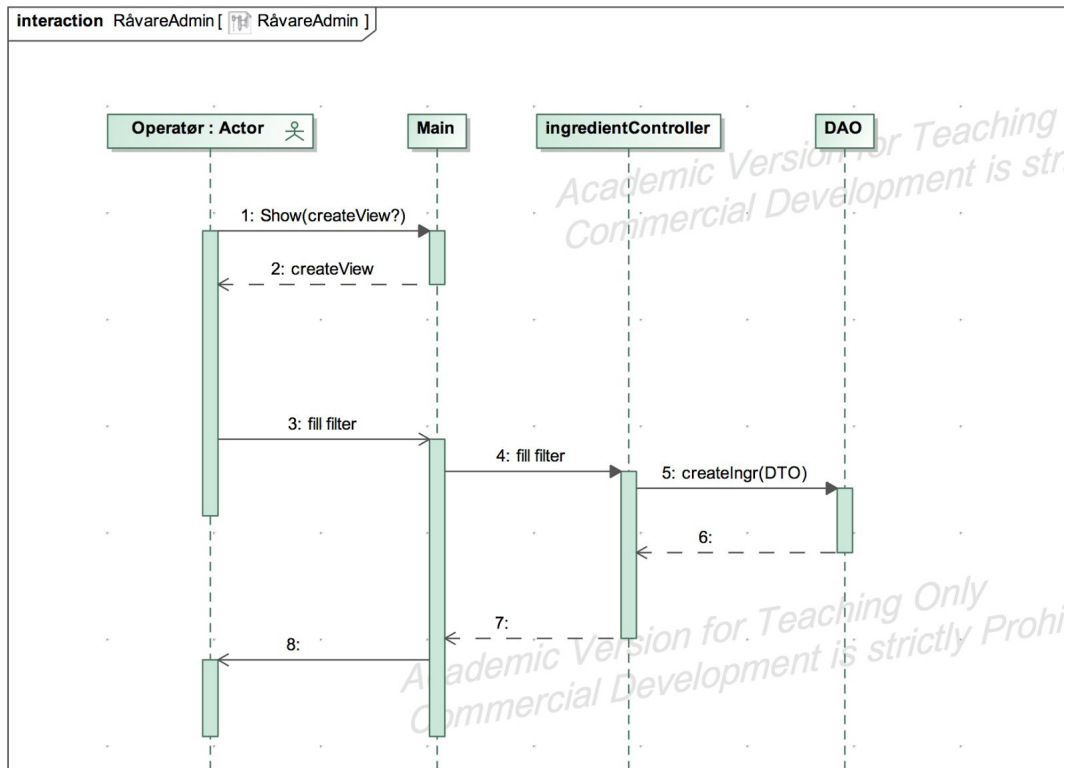
Afslutningsvis kan vi evaluere på projektforløbet og konkludere, at vi har afleveret et nogenlunde tilfredsstillende produkt, der overordnet set implementerer de vigtigste facetter fra kravspecifikation, men også har seriøse mangler. Disse mangler skyldes en række af ting, først og fremmest mange problemer med ASE og manglende erfaring og stabilisering med GWT. Vi har formået at lave et program, der kan: forbinde til en vægtterminal og foretage afvejningsproceduren; forbinde til en database og foretage forespørgsler, som henter information fra og skriver til databasen: vise disse forespørgsler i en grafisk interaktion med en aktør i form af et GWT-webinterface. Vi mener, at vi er sluppet godt fra opbygning af vores program, på trods af at fx main-controlleren strider med GRASP og koblingen kunne være lavere.

Det er vores intention, at vi til projektgruppens mundtlige eksamen kan fremvise en opdateret og fuldstændig funktionsdygtig version af programmet være tilgængelig, og vi vil være i stand til at fremvis programmet uden de småfejl, som desværre er opstået til sidst i projektet.

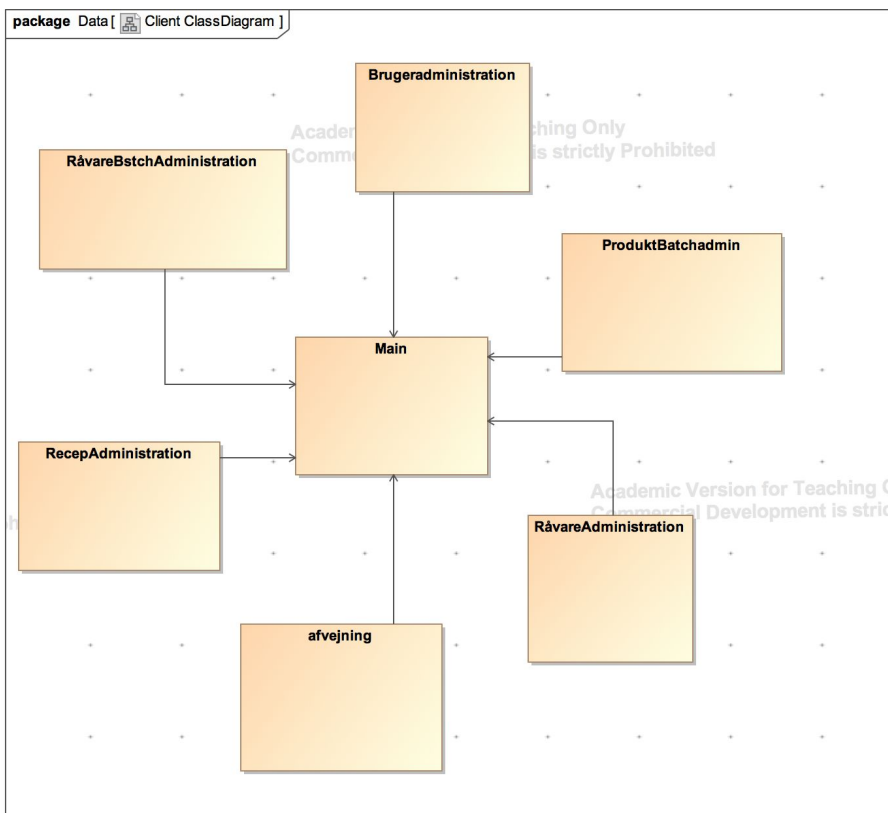
Bilag 1 : Sekvens diagram af start



Bilag 2 : Diagram af RåvareAdmin































































































































































































































Bilag 3 : Analyse klassediagram



Bilag 4 : Afvejningsprocedure

- 1: Operatøren har modtaget en produktionsforskrift på papir fra værkføreren.
- 2: Operatøren vælger en afvejningsterminal.
- 3: Operatøren indtaster operatør nr.
- 4: Vægten svarer tilbage med operatørnavn som så godkendes.
- 5: Operatøren indtaster produktbatch nummer.
- 6: Vægten svarer tilbage med navn på recept der skal produceres (eks: saltvand med citron)
- 7: Operatøren kontrollerer at vægten er ubelastet og trykker 'ok'
- 8: Systemet sætter produktbatch nummerets status til "Under produktion".
- 9: Vægten tareres.
- 10: Vægten beder om første tara beholder.
- 11: Operatør placerer første tarabeholder og trykker 'ok'
- 12: Vægten af tarabeholder registreres
- 13: Vægten tareres.
- 14: Vægten beder om raavarebatch nummer på første råvare .
- 14 (a):
- 15: Operatøren afvejer op til den ønskede mængde og trykker 'ok'
- 16: Pkt. 7 – 14 gentages indtil alle råvarer er afvejet.
- 17: Systemet sætter produktbatch nummerets status til "Afsluttet".
- 18: Det kan herefter genoptages af en ny operatør.

Bilag 5 : Tests

Test 01 af Recept	Test 02 af ReceptComp
                                  Finished after 0,103 seconds Runs: 4/4 Errors: 0 Failures: 0  code.TestReceptDAO [Runner: JUnit 4] (0,085 s) Failure Trace  testCreateRecept (0,072 s)  testUpdateRecept (0,004 s)  testGetRecept (0,004 s)  testGetReceptList (0,005 s)	                 Finished after 0,115 seconds Runs: 4/4 Errors: 0 Failures: 0  code.TestReceptCompDAO [Runner: JUnit 4] (0,085 s) Failure Trace  testUpdateReceptComp (0,081 s)  testGetReceptCompList (0,006 s)  getReceptCompListWithReceptID (0,008 s)  testGetReceptComp (0,005 s)
Test 03 af ProductBatch	Test 04 af ProductBatchComp
                                  Finished after 0,123 seconds Runs: 4/4 Errors: 0 Failures: 0  code.TestProductBatchDAO [Runner: JUnit 4] (0,085 s) Failure Trace  testGetProductBatch (0,071 s)  testUpdateProductBatch (0,006 s)  testGetProductBatchList (0,005 s)  testCreateProductBatch (0,010 s)	                 Finished after 0,113 seconds Runs: 4/4 Errors: 0 Failures: 0  code.TestProductBatchCompDAO [Runner: JUnit 4] (0,085 s) Failure Trace  testUpdateProductBatchComp (0,081 s)  testProductBatchCompList (0,004 s)  testGetProductBatchComp (0,006 s)  testGetProductBatchCompListPbld (0,004 s)
Test 05 af User	Test 06 af Ingredient
                                  Finished after 0,113 seconds Runs: 4/4 Errors: 0 Failures: 0  code.TestOperatorDAO [Runner: JUnit 4] (0,085 s) Failure Trace  updateOperator (0,072 s)  getOperatorList (0,006 s)  createOperator (0,012 s)  TestGetOperator (0,006 s)	                 Finished after 0,111 seconds Runs: 4/4 Errors: 0 Failures: 0  code.TestIngredientDAO [Runner: JUnit 4] (0,085 s) Failure Trace  testGetIngredList (0,075 s)  testUpdateIngred (0,005 s)  testGetIngredient (0,005 s)  TestCreateIngred (0,009 s)
Test 07 af FieldVerifier	Test 08 af IngredientBatch
                                  Finished after 0,018 seconds Runs: 10/10 Errors: 0 Failures: 0  code.TestFieldVerifier [Runner: JUnit 4] (0,000 s) Failure Trace  testInitialsWith1Char (0,000 s)  testIsValidName2Short (0,000 s)  testIsValidNameLength (0,001 s)  testIsValidName (0,000 s)  testInitialsWith19Char (0,000 s)  testInitials (0,000 s)  testIsCPRWithLetter (0,000 s)	                 Finished after 0,116 seconds Runs: 4/4 Errors: 0 Failures: 0  code.TestIngredientBatchDAO [Runner: JUnit 4] (0,085 s) Failure Trace  testGetIngredientBatchList (0,077 s)  testCreateIngredientBatch (0,013 s)  testUpdateIngredientBatch (0,005 s)  testGetIngredientBatchId (0,005 s)