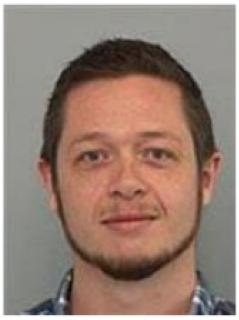
CDIO Final - 2016
02324 Videregående Programmering

Projektnavn: CDIO Final
Gruppe nr: 11
Afleveringsfrist: 17/06-2016
Denne rapport er afleveret via CampusNet (der skrives ikke under).
Denne rapport indeholder 48 sider inkl. bilag og denne side.

S136396
Christensen, Brian

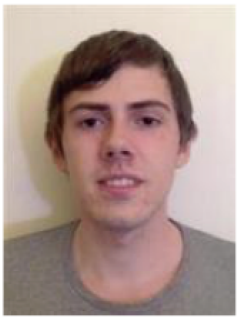


S136378
Christiansen, Morten Due

S140330
Danielsen, Casper

S153670
Larsen, Sofie Paludan

S153869
Lindberg, Cecilie



S123620
Matthiessen, Niels Ulrich

	Analyse	**Design**	**Impl.**	**Dok.**	**Test**	**Ialt**
Morten	14	14	17	13	15	73
Brian	12	15	18	15	13	73
Casper	14	13	19	14	13	73
Sofie	14	14	16	15	14	73
Cecilie	13	1	18	14	13	73
Niels	14	15	16	15	13	73
Sum	81	86	103	87	81	438

Indhold

1 Brugervejledning

Systemet består af fire komponenter: en web applikation, vægt, afvejningsstyreenhed (ASE) og en database. Databasen er opsat så både ASE'en og web applikationen kan bruge den, og det eneste, der skal bruges for at forbinde til denne, er en internetforbindelse.

Brugervejledningerne nedenfor er tilegnet systemadministratoren. Det antages, at vedkommende har kendskab til Eclipse, og selv formidler relevant information videre.

1.1 G11_CDIOFinal_GWT

Dette projekt indeholder web applikationen. Hvis der efter import af projektet er fejl, skal GWT SDK ændres i buildpath. Højreklik på projektet, vælg buildpath -> Configure Build Path. Her skal GWT biblioteket vælges, hvorefter version 2.6.1 kan vælges.

Når web applikationen startes op, åbnes en login side. Brug følgende ID og password til at logge ind:

- Administrator: ID: 1 Password: Qwer1234
- Farmaceut: ID: 2 Password: Qwer1234
- Værkfører: ID: 3 Password: Qwer1234
- Operatør: ID: 4 Password Qwer1234

Ved brug af operatør ID'et i web applikationen, vil der ikke være nogen funktionalitet, da operatøren ingen tilladelser har i web applikationen, da de kun arbejder med afvejning via ASE'en.

Fra hovedmenuen er det muligt at vælge forskellige menuer alt efter hvilken rolle man har. Rolle 1, kan se alle 5 muligheder; operator administration, recipe administration, ingredient administration, ingredientbatch administration og productbatch administration. Rolle 2 kan se alt undtagen operator administration. Rolle 3 kan se ingredientbatch og productbatch, mens rolle 4 ikke kan se noget.

1.1.1 Operator administration

Hvis man klikker på operator administration kommer der en liste frem af alle operatører. Fra dette view er det muligt at oprette en ny operatør. Her skal man være særlig opmærksom på, at man ikke skal skrive et opr_id som allerede findes i listen. Navnet skal være mellem 2 og 20 karakterer langt. CPR nummeret skal skrive med bindestreg, rolle skal være mellem 1 og 4 og passwordet skal have en længde mellem 5 og 8, og skal have opfuldt mindst

3 af følgende regler; Indeholde små bogstaver, indeholde store bogstaver, indeholder tal eller indeholde et af følgende tegn; . _ - + ! ? =. Fra listen er det også muligt at ændre på informationerne i operatørene ved at klikke på "edit-knappen. I dette view er det muligt at ændre oplysningerne på en operatør, dog skal man være opmærksom på at man ikke kan ændre id'et hvis operatøren har foretaget en afvejning. Dette sørger databasen for. Det er altid muligt at ændre active til false (Dette gælder alle menuer).

1.1.2 Recipe administration

I recipe menuen kan man se en oversigt over opskrifter. Også her er det muligt at klikke på create recipe. Når dette bliver gjort skal man først skrive et ID og navn på mindst 2 og max 20 karakterer, på opskriften man ønsker at oprette (Husk ingen dubletter af ID'er er tilladt). Herefter skal der oprettes komponenter til opskriften. Dette gør man ved at vælge et ingrediens ID og netto mellem 0.05 og 20 kg og en tolerance mellem 0.1 og 10%. Klik herefter på save component. Systemet fortæller hvis ingrediensen ikke findes eller den ikke længere er aktiv. Når alle komponenter er oprettet, skal man klikke på save recipe for at gemme det hele.

Fra listen er det også muligt at komme ind og se flere detaljer om en opskrift, ved tryk på "More...-knappen. Herinde kan man ændre de forskellige informationer, og man kan tilføje flere komponenter til opskriften. Dog skal man være opmærksom på, at alle ændringer kun bliver gemt hvis opskriften ikke har været i brug tidligere.

1.1.3 Ingredient administration

Inde i ingrediens menuen, kommer en liste over alle ingredienser. Hvis man ønsker at oprette en ny klik på create ingredient i bunden. Her inde skal man indtaste de nødvendige informationer, navnene skal være mellem 2 og 20 karakterer og klikke på save. Fra listen er det også muligt at ændre ingredienser, hvis man har lavet en fejl ved indtastning. Fra listen kan man også se hvilke ingredientbatches der er af en bestemt ingrediens, fra denne liste kan man oprette en ny ingredientbatch af den bestemte ingrediens.

1.1.4 Ingredientbatch administration

Denne menu viser en liste over alle ingredientbatches, fra listen er det muligt at ændre evt. indtastningsfejl man har lavet ved oprettelse. Oprettelse af ingredientbatches, skal ske ved indtastelse af ingredientbatch id, ingrediens id og en mængde skrevet i kg.

1.1.5 Productbatch administration

Productbatch menuen viser en liste over productbatches, fra denne er det muligt at oprette en ny productbatch, ved at indtaste et productbatch ID, som er unikt og en recipe ID. Hvis det valgte recipe ID ikke findes eller ikke længere er i brug, vil system komme med en fejlmeddelelse. Ved indtastningsfejl er det muligt at ændre de indtastede informationer, ved at trykke på edit knappen ud for det element der ønskes ændret.

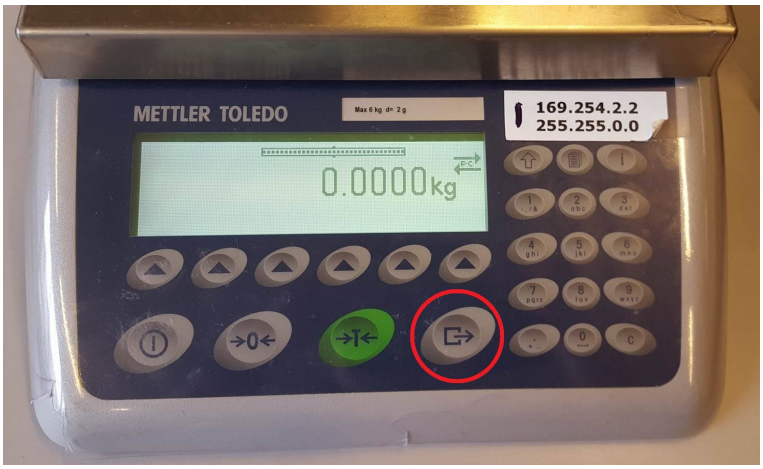
1.2 G11_CDIOFinal_ASE

Dette projekt indeholder ASE'en. Under ASE-projektet findes en tekstfil kaldet "connections". Det er i denne, at IP adresser og porte på vejeterminaler konfigureres. Skal der tilføjes flere terminaler, gøres dette på en ny linje.

Når vægten er tilsluttet, skal ASE'en startes op. Dette gøres ved at køre Java-filen "App". Her fra vil resten foregå på vægten.

Vægten skal naturligvis have de rigtige inputs. Når et input er tastet, trykker man på knappen "ok". For hver gang, vil vægten komme med en ny forespørgsel eller krav, hvor operatøren igen skal taste "ok", når disse er mødt. Når vægten printer "Press ok for weighing"tastes ok. Herefter vil vægten vise den nuværende vægt i displayet. Når den rette mængde er hældt op, og operatøren er tilfreds, tastes ikke "ok", men i stedet den knap, der er vist på figur 1.1.

Ønskes det derimod at afslutte og vende tilbage til login-stadiet, gøres dette med et "X". Bemærk, at bogstavet er stort, og man derfor først skal trykke på den hvide pil øverst til venstre. Man kan dog ikke afslutte midt i en afvejning.



Figur 1.1: Standard Interface Command Set

1.3 G11_CDIOFinal_Simulator

ASE'en kan også bruges med en vægtsimulator. Den ligger i dette projekt, og er et stykke kode, der blev udleveret i et tidligere CDIO projekt.

ASE'en skal, ligesom med alle andre enheder, tilsluttes denne. Simulatorens IP adrese og port skrives derfor i "connections". Når simulatoren skal starte, køres filen "App"under "application". Derefter startes ASE'en.

Når først Afvejningsstyreenheden er koblet til simulatoren er det ikke muligt at ændre den simulerede belastning. Derfor er simulatoren kun med til proof of concept. alle skridt i afvejningsprocessen kan gennemgås på nær det sidste (se 5.5.1)

2 Indledning

I dette CDIO projekt, der bygger på forrige delprojekter, skal der videreudvikles et softwaresystem, som håndterer og opbevarer data og dokumentation af bl.a. afvejninger og lagerbeholdning. Dette gøres med en database, et web interface, en afvejnings styrings enhed (ASE) og en vægt.

Systemet bliver bygget til en medicinalvirksomhed, hvor flere vejeterminaler står til rådighed for operatører, der skal foretage afvejninger. Disse vejeterminaler skal forbindes til en lokal ASE. ASE har til formål at styre afvejningsproceduren på vejeterminalerne, og skal derfor kunne forbinde til flere terminaler på én gang. Denne applikation skal skrives i Java.

Systemet skal kunne tilgås fra et web interface, der ligger på en ekstern server, hvor man med rette loginoplysninger kan se, rette i og oprette forskellige data. Dette interface skal implementeres i Java med Google Web Toolkit (GWT).

Både ASE og web interfacet skal have forbindelse til online databasen, hvor alle data ligger gemt. Her skal der oprettes relevante stored procedures, views, access rights osv., så databasen kan tilgås på sikker og effektiv vis.

3 Resumé

I dette projekt fik vi til opgave at færdigudvikle et softwaresystem for en medicinalvirksomhed, der skal kunne håndtere og dokumentere afvejninger og opbevare data. Det skulle være muligt at tilgå systemet som bruger via en web applikation, samt at foretage afvejninger ved vejeterminaler, der skal have forbindelse til ASE. Både ASE og web applikationen skal snakke med en online databas, som indeholder alle data.

Vores system indeholder både web applikation, database og ASE. Databasen ligger på Amazon Web Services (AWS), og snakker med både ASE og web applikation over en internetforbindelse. Via web applikationen er det muligt at foretage en række ændringer i databasen, afhængigt af brugerens rolle, mens det via ASE'en er muligt at foretage afvejninger og gemme nye produktbatchkomponenter.

Der er naturligvis givet en række krav og regler, der skal overholdes i forbindelse med dette system, hvilket vi har analyseret og udformet en kravspecifikation ud fra. Kravspecifikationen er skrevet med MoSCoW-metoden, hvor kun enkelte krav er ændret. Alle use cases har fået en use case description - to af dem er beskrevet yderligere i analysefasen, mens resten ligger i bilag. Der er derudover også lavet et sekvensdiagram over ASE'ens forløb.

I designfasen er vi gået i dybden med de tre komponenter (Web applikation, database og ASE), hvordan de skal opbygges og struktureres. Under web applikationen er der lavet et klassediagram og beskrivelse af klient/server. Under databasen er normaliseringen beskrevet og et ER-diagram tegnet.

Sidst har vi beskrevet ASE'ens flow, states og arkitektur.

Web applikationen er skrevet i Eclipse ved hjælp af Github til versionsstyring og Google Web Toolkit til udvikling. Sproget har været Java og HTML.

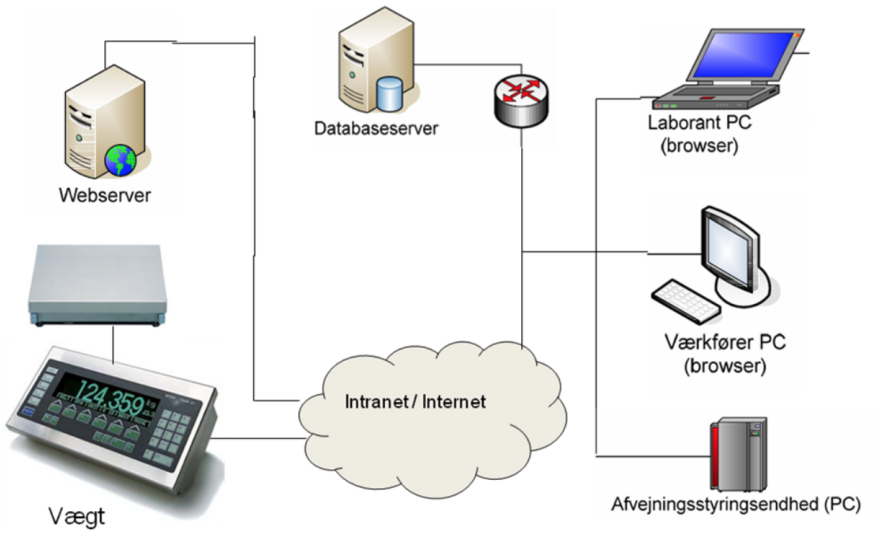
Databasen er skrevet i MySQL.

Implementeringen og den kodede struktur af web applikationen og ASE er beskrevet yderligere i implementeringsfasen.

Gennem projektets forløb er der blevet lavet code reviews, og til sidst en række J-Unit tests, der skal sikre, at koden fungerer efter reglerne. Disse er dokumenteret med testrapporter.

4 Oversigt

Hele projekt består af 4 hovedkomponenter. En web applikation, en vægtstyreenhed, en database og en fysisk vægt.



Figur 4.1: Systemoversigt [CDI]

Web applikationen er med til at give en oversigt over hvilke produkter, der er i produktion, er færdige og hvilke materialer der er til rådighed på lageret. Det er også muligt at oprette nye brugere, og se hvilke brugere, der allerede er oprettet i systemet. Det er også her, man opretter nye recepter, og sætter produktionen i gang.

Databasen bliver brugt til at gemme alle data, som for eksempel brugere, recepter og produktioner. Der gemmes ikke data uden for databasen. De data der gemmes, vil aldrig blive slettet, da man skal kunne gå tilbage for at

se, hvem der har været ansvarlig for en bestemt produktion, og hvilke komponenter der er blevet brugt til et bestemt parti. På den måde kan man finde ud af, om der har været, eller er en dårlig råvarebatch, og herved tilbagekalde produktet fra markedet.

Vægtstyreenheden er den del, der sender beskeder til vejeterminalerne. Beskederne vises i displayet, så operatøren ved hvilken handling, der skal udføres. Den vil også vise en besked, hvis der forekommer en fejl. Disse fejl betyder typisk, at man bare kan prøve igen.

Terminalerne skal have et bruger ID, inden man kan få lov til at veje de forskellige komponenter. Den kan ikke andet end at vise de forskellige beskeder, der kommer fra afvejningsstyreenheden.

Vægt og afvejningsstyreenheden behøver ikke at være på samme lokation, da de kan kommunikere over internettet.

5 Analyse

Ud fra analysen af de udleverede dokumenter, [CDI], er der blevet udformet en kravspecifikation. Aktører er blevet identificeret og beskrevet. Use cases var dog allerede blevet specificeret i oplægget. Kravene til databasen er blevet gennemgået og ændringer til databaseudlægget, bilag 3 i [CDI], er også fundet. Arkitektur til både webinterface og afvejningsstyreenhed er også udformet.

5.1 Kravspecifikation

Kravspecifikationen beskriver hvad systemet skal indeholde. MoSCoW modellen [DSD], er brugt til at prioritere de forskellige krav.

1. M: Et system skal udvikles til administration af produktionen i en medicinal virksomhed.
2. M: Der skal laves et web interface ved brug af GWT framework.
3. C: Systemet skal sættes i produktion på en web server.
4. M: Der skal udvikles en styreenhed til vægtene, som bruges af operatøren.
5. M: Systemet skal gemme data til et fast lager ved hjælp af en database.
6. S: Database skal køre på en server online.
7. M: Systemet skal understøtte forskellige roller. Rollerne er organiseret sådan at rolle 1 kan det rolle 2 kan plus mere, og rolle 2 kan det rolle 3 kan plus mere.

7.1. Rolle 1: Administrator.

7.1.1. M: Systemet skal sørge for denne rolle kan oprette og opdatere operatører.

7.2. Rolle 2: Farmaceut

7.2.1. M: Systemet skal sørge for denne rolle kan oprette og ændre recepter og receptkomponenter

7.3. Rolle 3: Værkfører.

7.3.1. M: Systemet skal sørge for denne rolle kan oprette og ændre produktbatches, og råvarebatches

7.4. Rolle 4: Operatør

7.4.1. M: Systemet skal sørge for denne rolle kan afveje ved hjælp af en vægt terminal

8. M: Der skal være en menu der gør det muligt at få adgang til de forskellige dele af programmet

8.1. M: Der skal være en operatør menu der gør det muligt at oprette, opdaterer og se alle brugere på systemet. Denne menu skal kun være tilgængelig for en administrator.

8.2. M: Der skal være en recept administrations menu der gør det muligt at oprette, se og ændre recepter. Denne menu skal kun være tilgængelig for farmaceut eller administratorer.

8.2.1. S: Det må ikke være muligt at ændre en recept hvis den bliver brugt i en produktbatch

8.3. M: Der skal være en ingrediens menu der gør det muligt at oprette, se og ændre ingredienser. Denne menu skal kun være tilgængelig for farmaceut eller administratorer.

8.3.1. S: Det må ikke være muligt at ændre en ingrediens hvis der findes en ingrediensbatch med den.

8.4. M: Der skal være en ingrediensbatch menu der gør det muligt at oprette, se og opdaterer ingrediensbatches. Menuen skal være tilgængeligt for alle på nær operatører.

8.4.1. S: Det må ikke være muligt at ændre på en ingrediensbatch hvis den bliver brugt i en produktbatch.

8.5. M: Der skal være en produktbatch menu der gør det muligt at oprette, se og ændre produktbaches. Menuen skal være tilgængelig for alle på nær operatøren.

8.5.1. S: Det må ikke være muligt at ændre i en produktbatch hvis den er gået i gang.

5.2 Aktørbeskrivelse

Kravspecifikationen beskriver, at der skal være en række brugere, som alle skal have adgang til forskellige dele af systemet.
Dette har ledt os frem til følgende:

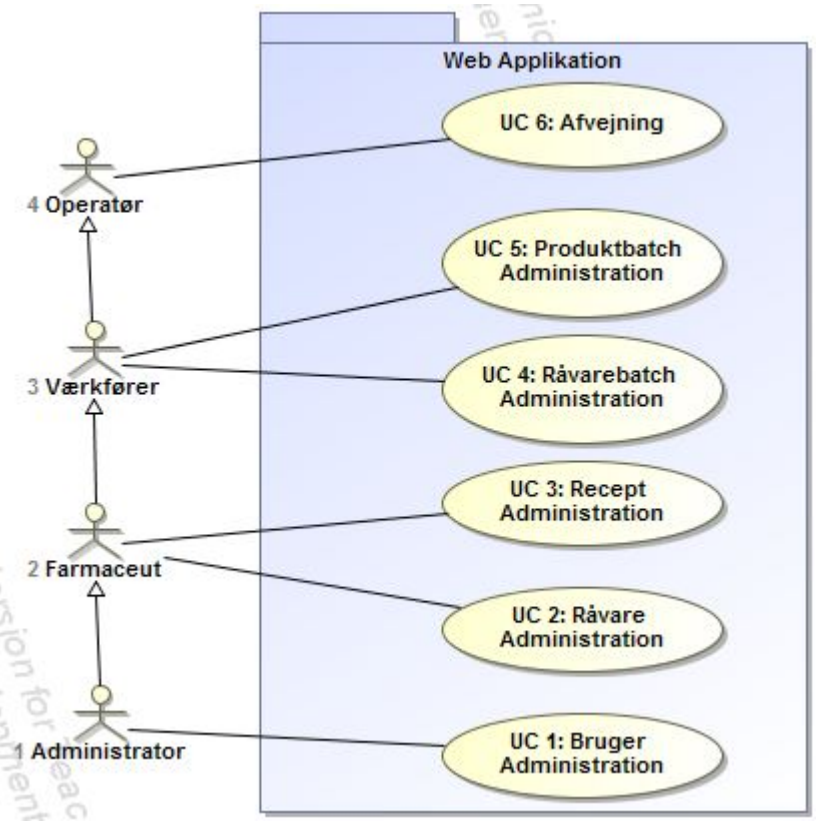
- Operatør: Denne aktør har til opgave at foretage afvejninger på vægten. Da denne aktør ikke har nogen grund til at logge på web applikationen, har vedkommende ikke adgang til den, den vil blot modtage en meddelelse om, at denne ingen rettigheder har.
- Værkfører: Denne aktør kan administrere produktbatch og råvarebatch. Det vil sige, at aktøren kan udføre CRUD operationer på produkt og råvarebatches. Arver derudover rettigheder fra Operatør.
- Farmaceut: Denne aktør kan administrere recepter og råvarer. Dvs. udføre CRUD funktioner på disse. Derudover arver en farmaceut værkførens rettigheder.
- Administrator: Denne aktør kan bruge CRUD operationer på alle brugere af systemet. Administratoren arver også rettigheder fra farmaceuten.

Ved CRUD menes der: create, read, update og delete. I dette system er delete dog fjernet og erstattet med en active, som, hvis den er sat til false, er det det samme, som at den ikke længere er i brug.

5.3 Use cases

Der er seks use cases i projektet. To af disse er valgt ud og beskrevet. De sidste use cases kan findes i bilag (13.0.2).

Der er også lavet et use case diagram, som giver et overblik over aktører og de forskellige use cases.



Figur 5.1: Use Case Diagram

5.3.1 U3: Recept administration

I use case 3 bliver det uddybet, hvordan recepter oprettes, ændres og vises. Den primære aktør er farmaceuten, som har til opgave at håndtere recepter og råvarer i systemet. Det er en precondition, at farmaceuten er logget ind på web applikationen.

For at oprette, ændre og/eller se recepterne i databasen, vælger farmaceuten "Recipe administration" i hovedmenuen. Derefter ses en liste med alle recepterne, hvor det er muligt at ændre i en recept (givet, at den ikke er knyttet til en produktbatch) eller oprette en ny. Ved at oprette en ny, klikker farmaceuten på "Create recipe". Her indtaster farmaceuten gyldig information, så som recept ID og navn, og tilføjer komponenter til recepten. Ved at have indtastet gyldig information ved en receptkomponent, klikkes "Save recipecomponent". Dette gøres, indtil alle komponenter er oprettet, og farmaceuten kan nu gemme recepten.

Som skrevet tidligere er det kun muligt at ændre i en recept, hvis den endnu ikke er blevet brugt i en produktbatch. Hvis det ikke er tilfældet, kan en recept ændres ved at klikke på "More" ud for en recept. Her bliver man

videresendt til en menu, hvor alle attributter kan ændres.

Use case:	Recept Administration
ID:	U3
Beskrivelse:	Dette use case beskriver oprettelsen af recepter. Dette gøres af værkføreren
Primære aktører:	Farmaceut
Sekundære aktører:	Ingen
Preconditions:	Farmaceuten er logget ind på web applikationen
Main flow:	1. Farmaceuten vælger menuen "Recipe Administration" 2. Farmaceuten vælger "Create recipe" 3. Recipe ID og Recipe Name indtastes 4. Farmaceuten klikker "Add New Component", og tilføjer x antal komponenter til recepten 4.1 For hver receptkomponent gemmes der 5. Farmaceuten gemmer recepten, når alle komponenter er tilføjet
Postcondtions:	Recept er oprettet
Alternative flows:	Ingen

Figur 5.2: Usecase 3: Recept administration

5.3.2 U6: Afvejning

I forbindelse med hvert trin i afvejningsprocessen laves der en masse tjek, hvor disse tjek kan deles op i to kategorier. En kategori bestående af brugerinput og en kategori bestående af database svar. I brugerinput kategorien er der inputvalidering (mismatch). I database kategorien bliver der tjekket, om der kommer et svar, og om det svar passer med de forventede ID'er, navne osv. Derudover vil der i trin 5, blive givet en besked hvis det givne råvarebatch ikke indeholder nok råvare, til at kunne lave komponenten.

I use case 6 bliver afvejningsprocessen beskrevet, hvor operatøren er den primære aktør. Til at afveje skal operatøren vælge en vejeterminal. Det er påkrævet, at applikationen kører og forbindelsen til ASE'en er gyldig. Main flowet svarer til ASE'ens proces, eftersom det er ASE'en, der styrer hele afvejningen. Ud fra hvert trin, hvor terminalen sender information tilbage, skal operatøren bekræfte i form af at trykke OK på vægten (Sker ved trin 1.1, 2.1, 4.1, 5.1, 6.2 og 7). Efter dette kan terminalen fortsætte. Når operatøren har kørt main flowet successfuldt igennem en gang, vil terminalen gå tilbage til trin 4.

Der er ingen alternative flows, eftersom man kun kan afveje med terminalen. Sker der fejl eller fremskridt rykker man blot op og ned i flowet.

Use case:	Afvejning
ID:	U6
Beskrivelse:	Operatør foretager afvejning ved en vejeterminal, som er forbundet til ASE
Primære aktører:	Operatør
Sekundære aktører:	Ingen
Preconditions:	Vejeterminalen er forbundet til ASE, og applikationen kører
Main flow:	1. Operatøren indtaster sit ID 1.1 Terminalen svarer med vedkommendes navn 2. Operatøren indtaster et produktbatch ID 2.1 Terminalen tjekker, at ID'et er gyldigt og har rette status Returnerer herefter opskriftens navn 3. Operatøren taster enter, når vægten er tom 4. Operatøren taster enter, når den første beholder er placeret 4.1 Terminalen tarerer og gemmer værdien 5. Operatøren indtaster det ønskede råvarebatch ID 5.1 Terminalen tjekker, at ID'et er gyldigt, og at produktbatch-en ikke allerede indeholder denne råvare
	6. Operatøren hælder den ønskede mængde op og taster enter for for at veje 6.1 Terminalen tjekker, at vægten svarer til net, og ikke er højere eller lavere end tolerancen 6.2 Der oprettes en ny produktbatchkomponent 7. Operatøren kan taste enter for at foretage en ny afvejning (trin 4)
Postcondtions:	Produktbatchkomponenten er oprettet
Alternative flows:	Ingen

Figur 5.3: U6: Afvejning beskrivelse

5.4 Database

I analysen af databaseopbygningen og brugen af databasen, er der reflekteret over, hvordan sletning og opdatering af objekter i databasen skal fungere. Det blev besluttet, at intet må blive slettet fra databasen. Dette grundet, at alt skal kunne dokumenteres senere, og hvis der er mulighed for at slette, kan det ikke dokumenteres.

Opdatering af detaljer på de forskellige entiteter må ske så længe, at den ikke har en fremmednøgle i en anden tabel. Dette giver fordelen, at tastefejl i oprettelse af forskellige elementer kan rettes. Men senere, når eksempelvis en råvare er brugt i en recept, er det ikke længere muligt at rette i denne råvare. Herved undgås det, at tomater eksempelvis lige pludselig bliver omdøbt til bananer. Det betyder, at farmaceutten er nødt til at oprette nye entiteter i disse tilfælde.

5.4.1 Tabelændringer

Ligesom i de forrige delprojekter er operatørens initialer blevet fjernet og der er tilføjet en rolletabel. Initialerne er ikke implementeret, da man ikke fandt anvendelse for dette. Ved udskrifter eller lignende bruges ID eller navn. Rollerne indgår som en ekstra attribut hos operatørerne, og skaber en fremmednøgle reference til rolle-tabellen. Dette er gjort for at imødekomme kravet fra CDIO Final oplægget, [CDI], hvor det skal være muligt at give varierende tilladelser blandt brugerene af systemet. Der er givet fire roller.

For at overkomme begrænsningen ovenfor, hvor det ikke er muligt at slette, er der tilføjet en attribut "active" til tabellerne. Herved er det muligt at deaktivere ikke længere brugte råvare, operatører etc.

5.5 Afvejningsstyreenhed

Det er ønsket, at afvejnings styrings enheden (ASE) skal fungere med en vægtterminal, men samtidig nemt kan udvides til at understøtte flere terminaler. En idé er, at have en liste over adresserne på terminalerne. Når ASE starter vil den oprette en række tråde (en til hver terminal) ud fra listen, hvor hver tråd har eget ansvaret for at kommunikere med de forskellige terminaler.

5.5.1 ASE og vejeterminal

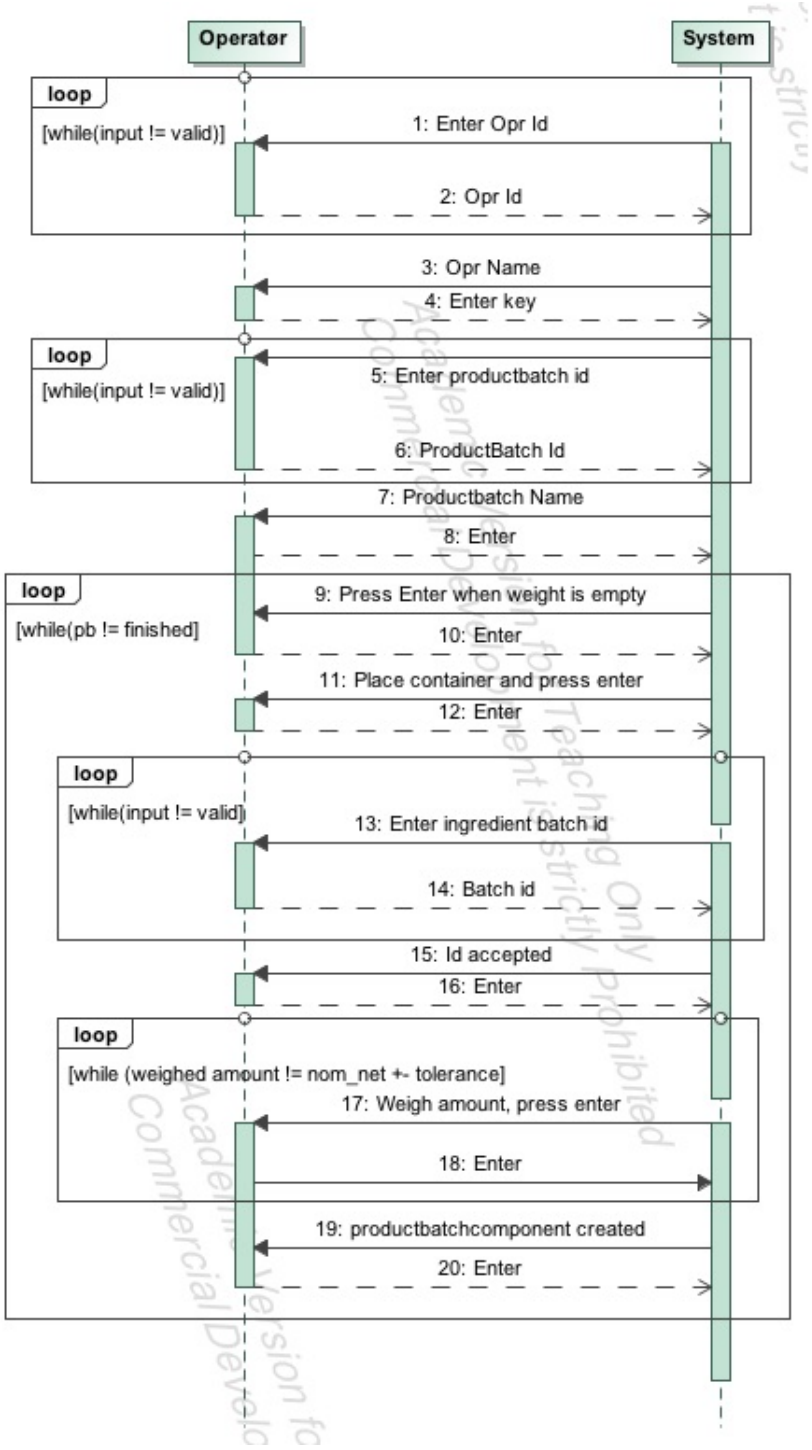
Samspillet mellem ASE'en og vejeterminalen er styret fuldstændigt af ASE'en. ASE'en sender beskeder til terminalen, som operatøren efterfølgende skal agere på.

Ved hjælp af afvejningsprocessen (bilag 4 i [CDI]) samt use case detail for U6, er vi kommet frem til følgende trin, som skal udføres i softwaren:

1. ASE forespørger opr id, og søger efter operatoren i databasen og returnere navn til vejeterminal hvis det er et gyldigt ID.
2. ASE forespørger Produktbatch ID. Inputtet kontrolleres i databasen, og returnere recept navn.
3. ASE forespørger om vægten er ubelastet, Operatør trykke enter.
4. ASE tarerer vægten og sætter produktbatchstatus til 1 (igang).
5. ASE forespørger at beholder sættes på vægten, operatør placerer beholder og trykker ok.
6. ASE gemmer vægt af beholder og tarere vægten.
7. ASE forespørger råvare batch ID.
8. ASE forespørger at operatør afvejer passende mængde og trykker ok

9. ASE registrere det afvejede, og kontrollere det holder sig inden for definerede tolerance.

Dette har videre kunne bruges til at lave et sekvens diagram over systemets forløb (5.5.1). Her ses det endnu tydeligere, at det er ASE'en der forespørger og styre forløbet i afvejningsprocessen. Det er endvidere bestemt, at operatøren kan afbryde forløbet i afvejningsprocessen ved at trykke x, på et hvilket som helst tidspunkt.



Figur 5.4: ASE Sekvens diagram

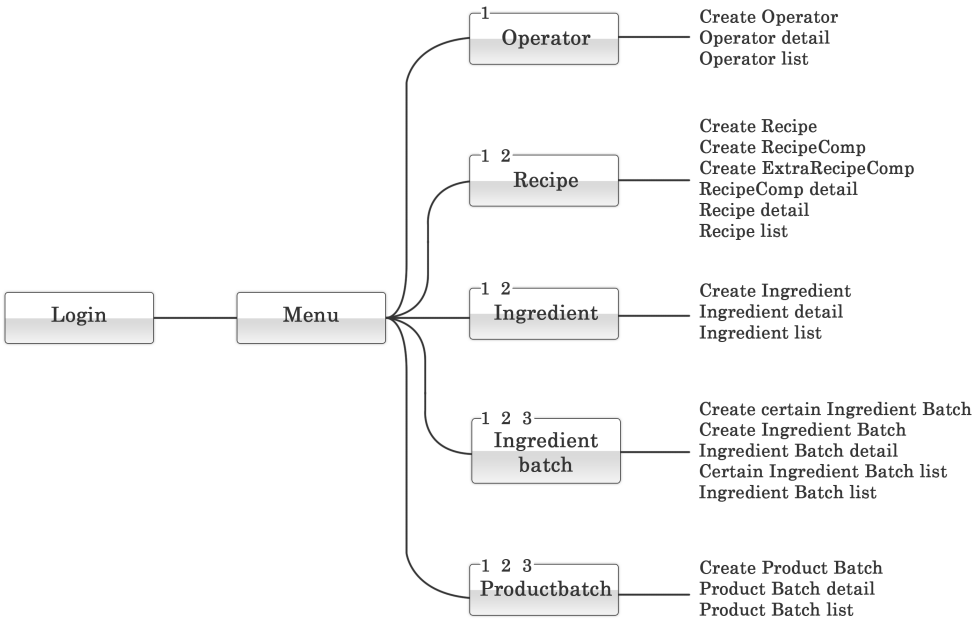
5.6 Web Applikation

Web applikationen skal udvikles ved brug af GWT Framework. I denne skal der være en klient side og en server side. Server siden står for kommunikationen med databasen, samt session håndtering af brugere, der er logget ind.

5.6.1 Validering af input

Det ønskes, at al input bliver valideret, hvilket vil sige, at der ikke videresendes ugyldig data. Valideringen skal foregå flere steder, både på klient og server siden. Dette er for at skabe en større sikkerhed, for at alle input er korrekte. Et ID skal for eksempel være mellem 1 og 9999999 - dette gælder for alle ID'er. Navnene skal alle have en længde mellem 2 og 20 karakterer. Password skal leve op til de regler, som også gælder for password på DTU's campusnet, og have en længde, på mellem 5 og 8 karakterer. De resterende regler angående input står i bilag (13.0.1).

5.6.2 Storyboard



Figur 5.5: GWT

For at få en oversigt over, hvilke sider der skal være i applikationen, er der lavet ovenstående storyboard (se figur 5.5).

Alt starter med, at en bruger logger ind. Herefter er antallet af muligheder begrænset, afhængigt af brugerens rolle, hvilket også er illustreret på figur 5.5 med tal på de forskellige menuer.

Operatøren (rolle 4) kan logge ind, men har ingen rettigheder i web interfacet, og vil derfor ikke kunne foretage sig noget. Rollerne er som følger:

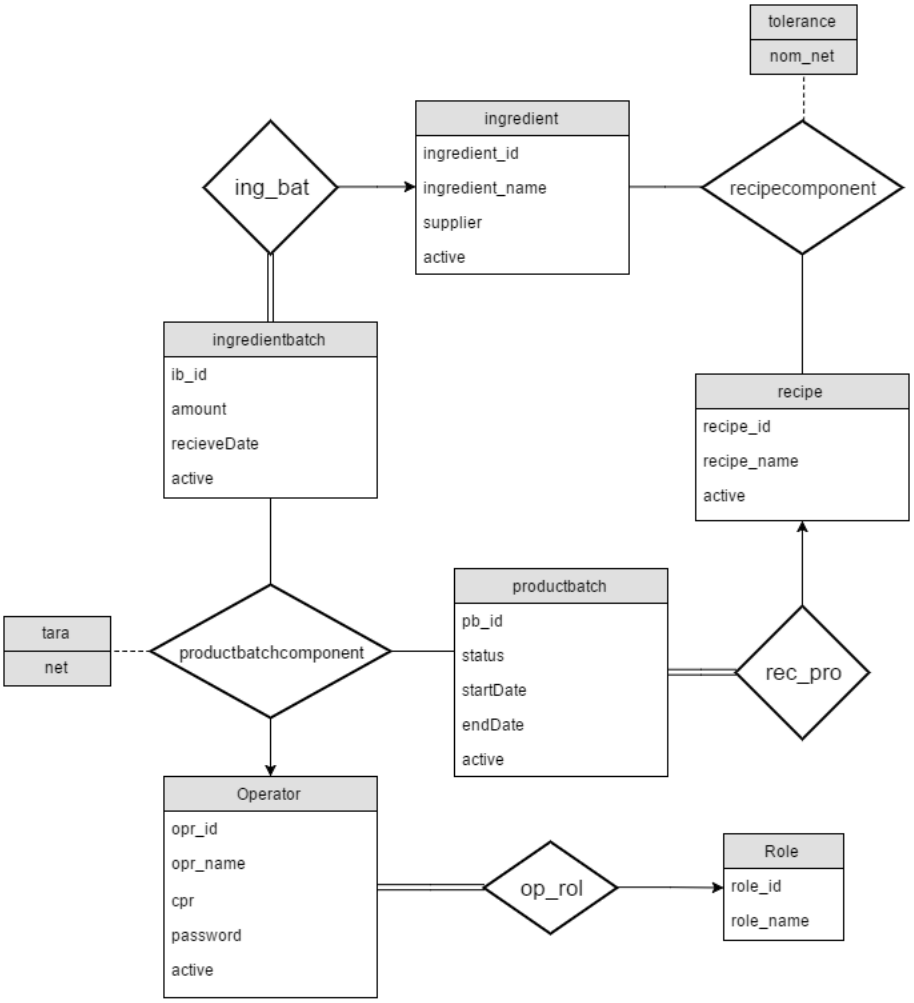
1. Administrator (rolle 1)
2. Farmaceut (rolle 2)
3. Værkfører (rolle 3)
4. Operatør (rolle 4)

Der ses et generelt system i opbygningen, hvor der fra hovedmenuen kan vælges en række views, der viser lister over forskellige ting. Dette kunne være operatører, råvare, produktbatches etc. Fra disse er der mulighed for at se detaljer om de forskellige elementer i listerne, samt rette i dem. Der er også mulighed for at oprette nye elementer.

6 Design

6.1 Databasedesign

Databasen har syv tabeller, som indeholder de forskellige informationer, som programmet skal bruge. I ER-diagrammet nedenfor (figur 6.1) kan forholdet mellem de forskellige tabeller ses.



Figur 6.1: ER-diagram

I databasen er der lavet to brugere: ASE'en og serveren, som har adgang til forskellige ting i databasen. Dette er gjort for at skabe en større sikkerhed i forhold til, hvem der må ændre hvad i databasen.

Der er lavet en række views, som bruges til at hente den information, som vi er interesseret i. Derudover bliver nogle tabeller brugt, som de er bl.a. operator og recipe. Brugerne må se forskellige views, hvilket er gjort

ved at give adgang (grant select) på de views og tabeller, som de må se data fra.

I stedet for, at brugere må indsætte og opdatere data i tabellerne direkte, er der lavet stored procedures, som kan bruges til at indsætte og opdatere data. I stored procedure er der nogle tjek, som gør, at der ikke ændres noget i databasen, som ikke må ændres. For eksempel, hvis man ønsker at opdatere en ingrediens, må dette kun gøres, hvis ingrediensen ikke er blevet brugt i en opskrift. Dette tjek laver databasen før den opdaterer ingrediensen. Disse tjek bliver altid lavet før data bliver opdateret. Det eneste, der altid må ændres på, er om den er aktiv eller ej.

Den eneste create procedure, der laver et tjek, er create_recipeComponent, som først tjekker om opskriften bliver brugt. Hvis den gør dette, må man ikke oprette en ny komponent i opskriften.

6.1.1 Normalisering

Tabellernes normalisering er beskrevet nedenfor

- Role: Denne tabel overholder første normalform, idet de enkelte celler kun består af singulære værdier. Anden normalform er også overholdt, da primærnøglen kun består af en attribut. Tabellen overholder også tredje normalform da attributterne ikke er transitive.

- Operator: Denne tabel overholder både første og anden normalform. Dog overholder tabellen ikke tredje normalform, det gør den ikke fordi både opr_id og cpr vil være unikke, og man kan derfor finde informationer om operatører ved hjælp af cpr, som ikke er primærnøgle. Hvis man ønskede at tabellen var i tredje normalform, kunne man dele den op, så al personlig information var gemt i en tabel, som have opr_id som fremmednøgle. På den måde ville der ikke være nogen transitive attributer.

- Ingredient: Tabellen overholder første normalform, da cellerne kun består af atomiske værdier, anden normalform er også overholdt, da der kun er én primærnøgle. Den tredje normalform overholdes også, da et råvarenavn ikke kan føre til en leverandør eller omvendt. Flere leverandører kan nemlig producere den samme råvare. Vi har altså ingen transitive attributter.

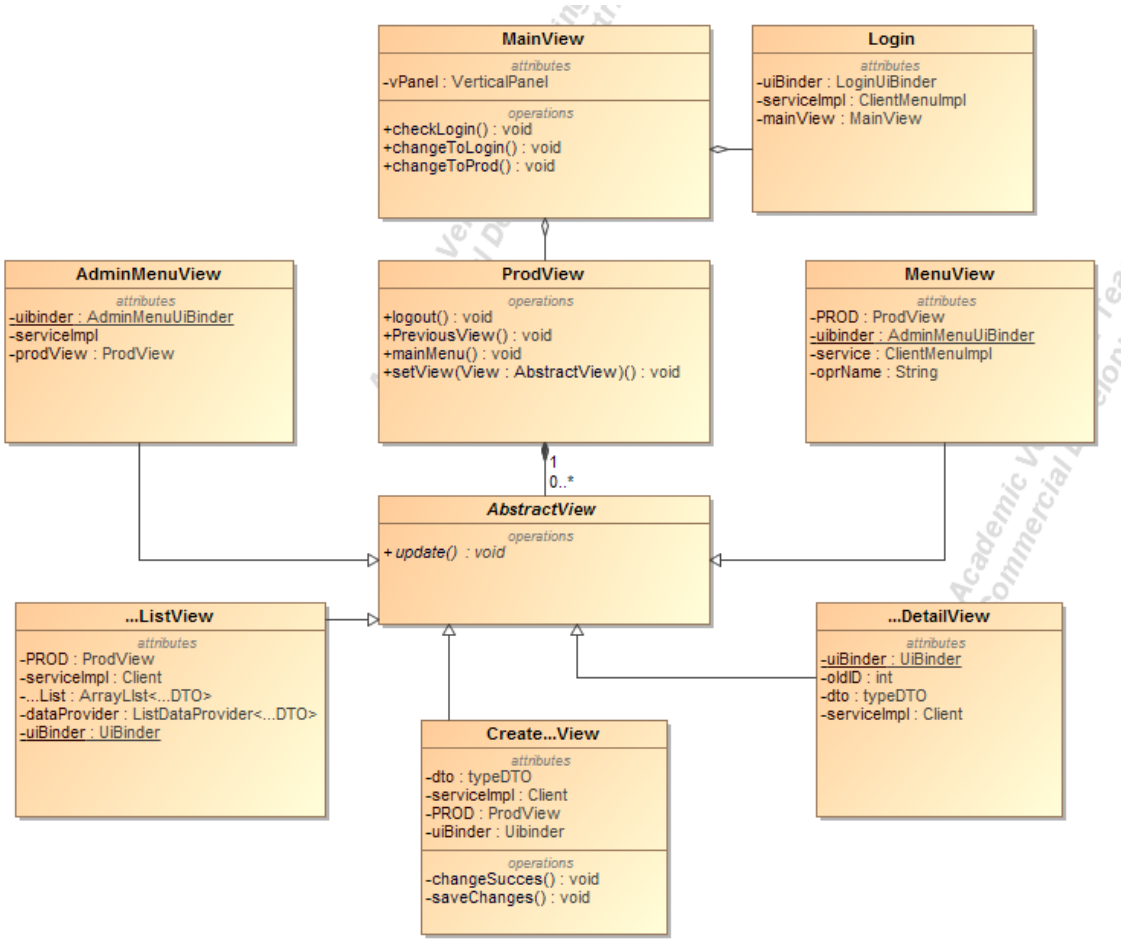
- Ingredientbatch: Overholder første normalform, da der kun er atomiske værdier, derudover er den også i anden normalform, da der kun er én primærnøgle. Tredje normalform overholdes, da attributten "amount" ikke kan finde frem til et ingredient_id og omvendt. Der er derfor ingen transitive attributter.

- Recipe: Denne tabel overholder alle tre normalformer. Der er kun atomiske elementer, kun en primærnøgle, og ingen transitive attributter.
- Recipecomponent: Overholder første normalform, kun atomiske værdier. Anden normalform overholdes også, da hverken recipe_id eller ingredient_id kan finde frem til nom_netto eller tolerance alene, dette kan man kun med kombinationen. Tredje normalform opnås, da nom_net ikke kan angive tolerancen og omvendt.
- Productbatch: Overholder alle tre normalformer. Grunden er den samme som i "Ingredientbatch".
- Productbatchcomponent: Første normalform overholdes, da der kun er atomiske værdier i cellerne. Anden normalform overholdes også da, productbatch_id eller ingredient_id ikke alene kan finde frem til tara, netto, dato eller opr_id, disse er ikke unikke, og operatører kan have afvejet flere forskellige productbachcomponent. De andre attributter kan samtidigt heller ikke finde frem til hinanden, hvilket vil sige, at tredje normalform også overholdes.

6.2 Web applikation

Web applikationen er designet med fokus på simpel og ensformigt udseende, så det er let for brugeren at finde rundt på hjemmesiden. Der har også været fokus på brugervenlighed, da ønsket var, at det skulle være nemt at bruge vores system.

6.2.1 Klient del



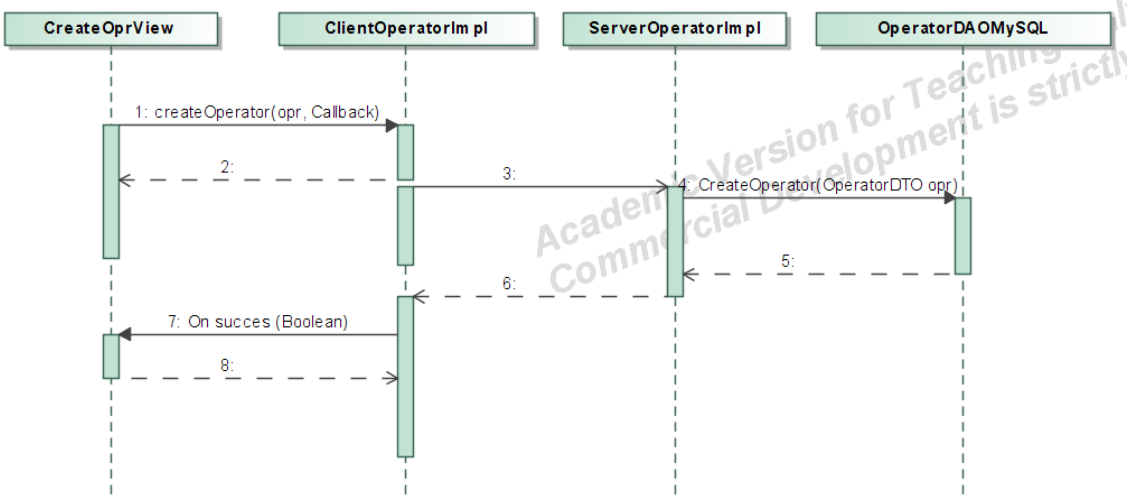
Figur 6.2: Klasse diagram

Klassediagrammet på figur 6.2 er en generel beskrivelse, hvor de enkelte navne er erstattet med "...". Dvs. "...ListView" kan eksempelvis være operatorListView. Dette er lavet for at simplificere klassediagrammet og gøre det mere overskueligt. "MainView" er hovedsiden, som det aktive view bliver lagt ovenpå. Den styrer også, om man er i prodview eller i login. Klassen "Prodview" tager sig af de forskellige skift, der er imellem menuerne i Administrationsinterfacet. Den håndterer også de skift, der sker på knapperne "Back", "Log out" og "Main menu". Alle de enkelte views indeholder også en instans af prodview, da de skal være i stand til at bruge funktionen setView for at gå videre. Alle de enkelte views bortset fra Main, login og prodView arver fra den abstrakte klasse AbstractView. AbstractView klassen er oprettet som en base klasse, for at alle arvende klasser indeholder en

update metode. Hvilket betyder, at update bliver kaldt på et view, hvis det skal vises igen.

Dette gøres for, at funktionen setView kan sætte alle de forskelige views. Alle views har også en funktion update, som kaldes når man går tilbage til en side ved hjælp af back knappen.

6.2.2 Server del



Figur 6.3: Diagram for callbacks

Serverdelen af vores Java program skal fungere asynkront af det, der foregår på klientdelen. Dette skyldes, at der kan være en lang kommunikationstid, og det vil ikke være hensigtsmæssigt at vente på et svar. Dette betyder også, at når kaldet laves fra klienten, skal den have et callback med, så serveren ved hvor den skal sende svaret hen. Al kommunikationen mellem server og klient foregår ved hjælp af remote procedure calls.

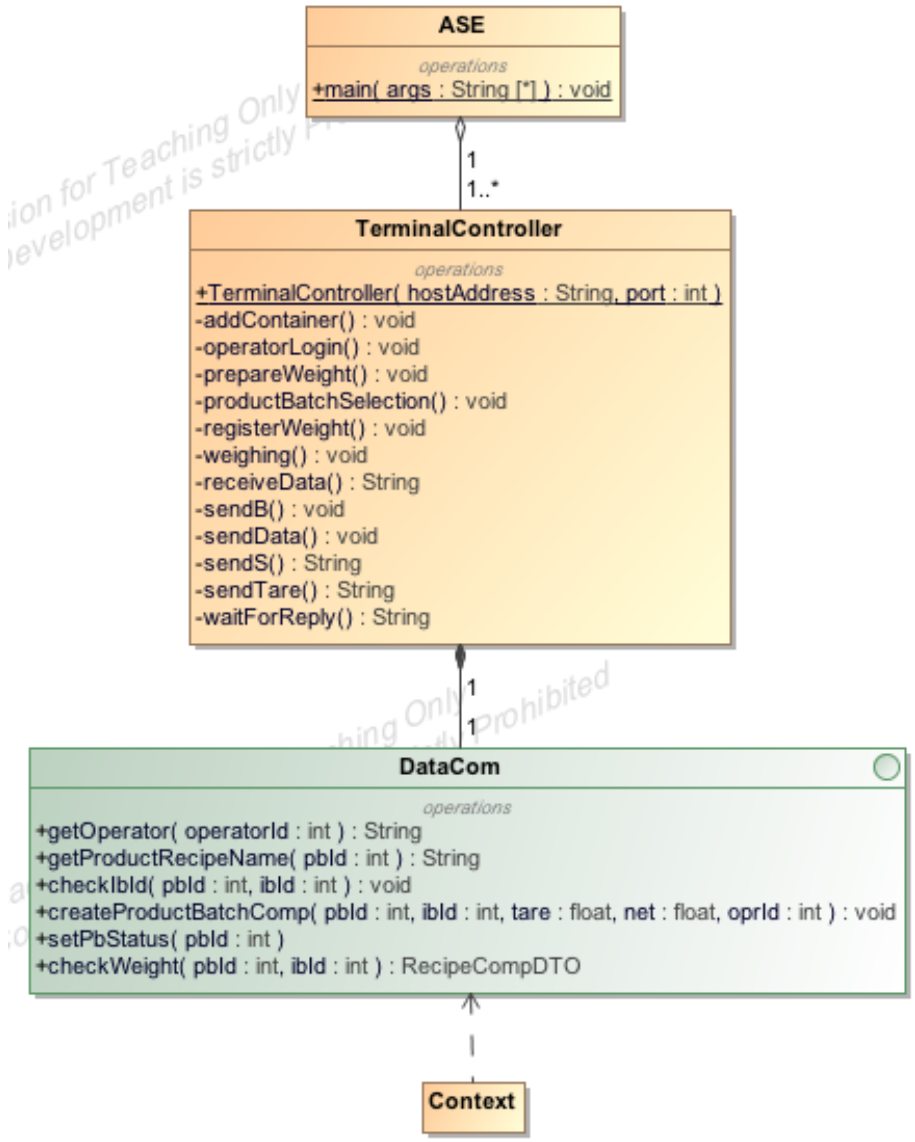
6.3 Afvejningsstyreenhed

Afvejningsstyreenheden er designet ud fra tanken om, at der nemt skal kunne udvides med flere vejeterminaler. Dette har krævet, at der skulle tænkes i en multitråd arkitektur.

6.3.1 Arkitektur

ASE softwaren har en klasse "ASE", som gør forbindelserne og dermed programmet klar. Denne klasse bruger en tekst fil, ved navn "connections", til at oprette en række terminal controllers. Connection-filen indeholder kommaseparerede rækker, hvor der er IP adresse og port nummer på de forskellige

terminaler. Disse adresser og porte bliver læst ved opstart og forbindelse skabt til dem.



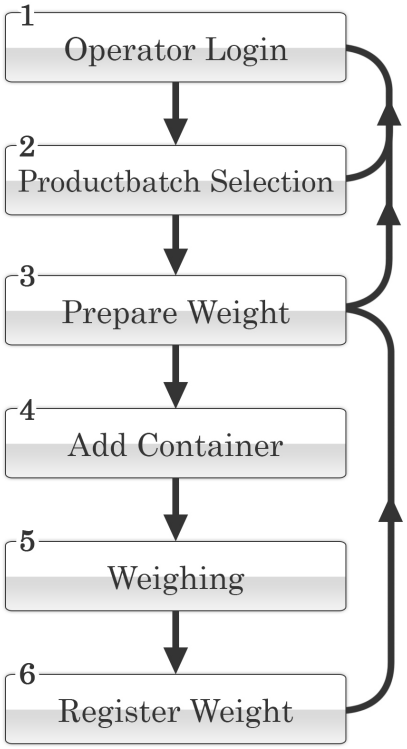
Figur 6.4: ASE klasse diagram

6.3.2 ASE flow og states

ASE'ens processer kan opdeles i states, hvor hver state har deres egen opgave. Når hver state er afsluttet på succesfuld vis, vil ASE'en gå videre til næste state. Bliver der derimod trykket "x", der er bestemt som afslutningsknappen, vil ASE'en gå tilbage til det første state, "Operator Login". Dette

sker dog ikke efter, at operatøren har klargjort vægten. Her er det ikke muligt at afbryde før, at afvejningen er fuldført og man automatisk bliver sendt tilbage til stadiet "Prepare Weight".

På figur 6.5 ses en illustration over, hvordan de forskellige states hænger sammen, og hvordan flowet i ASE'en er.



Figur 6.5: ASE Flow

7 Implementering

Der er blevet brugt Google Wep Toolkit (GWT), med en tilføjelse af GWT-Bootstrap, som giver yderligere funktionaliteter, for eksempel modals. Det giver også et pænere layout til GWT. Layoutet på alle views i systemet er lavet ved hjælp af UIBinders, hvilket er en java-fil og en xml-fil, der er kædet sammen. Disse UIBinders findes bl.a. i de knapper og felter, vi har brugt.

7.1 Web Applikation

Web applikationens front-end består af en række views. Disse views bruger remote procedure calls til at transportere information til og fra backend.

Ud over dette er der udviklet et simpelt navigationssystem, som gør det muligt for brugeren at bevæge sig hurtigt og nemt rundt i applikationerne.

7.1.1 Navigation

Da der er flere forskellige views, er det fordelagtigt at kunne gå fra det aktuelle view og tilbage til de tidligere views i rækkefølge. Denne funktionalitet er blevet implementeret ved hjælp af en stack kaldet "AbstractViews".

AbstractViews stakken kan indeholde klasser, som arver fra klassen AbstractView. AbstractView klassen har en abstract metode kaldt "Update".

ProdView er det styrende view i administrationsdelen, og dette view sørger for at udskifte det aktuelle UI. Når man skifter til et nyt view, bliver det før aktive view tilføjet til stakken. Når man trykker på "Back" knappen, vil det øverste view blive pop'et, og den føromtalte update metode blive kørt. Update sørger for at opdatere det gamle view, med eventuelle nye data, hvis dette er nødvendigt. Dette er eksempelvis når der oprettes nye objekter, og man vender tilbage til listen, der skal være opdateret med den nye information. Herefter bliver viewet sat som det view, der skal vises. Referencen til viewet brugeren kom fra, bliver nu ikke brugt mere, og garbage collectoren vil rydde op i dette, på et passende tidspunkt.

7.1.2 View opbygning

Mange af de views som er lavet, er opbygget på den samme måde. I hver kategori (Råvare, produktbatch og osv.) er der et list-, et detail- og et create-view, som viser henholdsvis en liste over elementer i kategorien, detail viser detaljerne om et bestemt element og create viewet bruges, når man ønsker at lave et nyt element af den pågældende kategori.

- List viewet: Dette view er lavet, så det viser udvalgt informatoner om alle elementer i kategorien. Dette bliver gjort ved en celltable, som indeholder DTO'er af det, som skal vises i listen. Tabelen modtager data fra en dataprovider, som modtager data fra remote procedure calls, som

opdaterer tabellen med de infromationer, der er i databasen. Fra dette view er det muligt at gå til de to andre views. For at gå til detail view, skal man vælge at klikke på knappen More... Når dette bliver gjort, bliver viewet ændret til detail view ved hjælp af ProdView klassen, dette view skal have sendt en DTO, som den skal vise informationer fra, dette får den fra tabelen. Nederst i dette view kan man vælge at lave et nyt element, af den liste man er inde i. Denne klasse indeholder også en update metode, som bliver kaldt, hver gang viwet skal vises, den opdater listen, så de nyeste informationer også bliver vist.

- Detail viewet: Dette view modtager den DTO, som skal vises. Viewet udtrækker de forskellige informationer, og viser dem. Klassen indeholder en masse click handlers, som håndterer funktionen af de knapper, brugeren klikker på. Når der bliver klikket på en edit knap, bliver en modal åbnet, denne modal's id bliver ændret, så den passer, til det der er blevet ændret. På denne måde kan alle ændringer blive udført af den samme metode, nemlig saveChanges(). Denne metode bruger en switch case, som tjekker hvilket id en modal har, og ud fra dette tjekker den om ændringerne lever op til de regler som den skal. Hvis dette er tilfældet bliver DTO'en opdateret, og ved hjælp af en serviceImpl opdaterer den informationerne i databasen. Dog skal man være opmærksom på, at selvom det ligner, at det lykkedes at opdatere, kan det være databasen, ikke ændre informationerne alligevel, da der kan være nogle andre ting, som umuligøre en ændring.

- Create viewet: Denne klasse er bygget op, så den viser nogle tekstbokse, hvor brugeren skal indtaste den information de ønsker. Når felterne er udfyldt, sørger click handleren for at sætte Fildverifier klassen i gang med at tjekke de indtastede data og sikre sig, at de lever op til reglerne. Efter dette tjek, bliver der lavet en DTO, som bliver sendt til serviceImpl, med en callback, som sørger for at fortælle brugeren, om det er lykkedes at oprette de aktuelle ting.

7.1.3 Server backend

Serverbackenden kan tilgås af alle views. Denne tilgås ved hjælp af remote procedure calls. Største delen af metoderne i backend bruger databasen til enten at hente informationer eller gemme informationer.

Enkelte arbejder med sessions. Når en bruger logger ind, bliver der gemt en session på serveren, som registrerer hvilken rolle den bruger, der logger ind har. Når isLoggedIn() bliver kaldt, returnerer den brugerens rolle. På den måde kan programmet sikre sig, at det kun er de tilladte brugerroller, der får adgang til aktuelt indhold.

Når en bruger logger ud bliver sessionsvariablen ændret til 0. Null og 0 forståes begge som om, at man ikke er logget ind i systemet. Alt over 0 og

under 5 er en rolle, som kan logge ind i systemet.

7.1.4 Validering

Der er lavet to klasser, som står for at validere de input som brugeren skriver. Den ene er FieldVerifier, den står for at validere input på klient siden. Dette gør den ved hjælp af en række metoder, som tjekker om inputene er korrekte. Hvis de input, som bliver tjekket er korrekte, returnerer disse metoder true.

cprValid tjekker om brugeren har skrevet et gyldigt cpr nummer, dette bliver gjort ved at tjekke om det første tal er mellem 0 og 3, da det er det eneste, som dagen kan starte med i en dato. Det andet tal kan være alt mellem 0 og 9. Det tredje tal skal være mellem 0 og 1, og det fjerde mellem 0 og 9. tallene som repræsenterer året, må være alle tal, der skal bare repræsenteres med 2 decimaler. Herefter skal der være en bindestreg. De sidste 4 tal, må være alt fra 0 til 9. Denne måde at tjekke cpr på, finder ikke fejl i datoen, hvis brugeren har skrevet datoer som er over 31 og heller ikke om datoen ligger ude i fremtiden.

Den anden klasse er DTOVerifier, som bruges på server siden. Denne klasse validerer de DTO, som den modtager, dette gør den ved hjælp af FieldVerifier klassen, som er beskrevet tidligere. På denne måde bliver alle input tjekket to gange for fejl, og vi er derfor sikre på at input er korrekt.

7.2 Afvejningsstyreenhed

Afvejningsstyreenheden er en separat applikation. Den skal stadig have forbindelse til den fælles database, hvilket gøres via en connector.

7.2.1 TerminalControllere

Når Afvejningsstyreenheden startes via klassen ASE, læser den i connection tekstfilen. For hver linje opretter den en instans af TerminalControllerklassen.

Terminalcontrollerens konstruktør tager imod en IP adresse og en port. Begge skal passe med en vejeterminal. Herefter opretter TerminalControlleren en forbindelse til vejeterminalen ved hjælp af TCP protokol.

7.2.2 Databaseforbindelse

For at kommunikere med databasen er der lavet et interface (DataCom), som deklarerer de metoder, der skal bruges. Context klassen implementerer DataCom interfacet i forhold til MySQL. Context klassen bruger klassen connector, som er en simpel wrapper til MySQL connectoren. Denne står for at oprette og fjerne statements variable, og den har en enkelt public metode kaldet doQuery. Denne tager imod en string, og returnere et result set fra databasen.

8 Test

I gennem hele projektet er der løbende udført test for mindske antallet af fejl i det færdige produkt. Testene har været af forskellig karakter, der strækker sig fra unit test og monkey test, til brugertest.

Ansvar for udvikling af forskellige delkomponenter er blevet delt ud blandt gruppens medlemmer, men for at sikre kvalitet i udvikling og kode, er der også løbende blevet udført code reviews.

8.1 Database

Databasen består af tabeller, views og stored procedures. For at bekræfte at views returnere det forventede data, er det test data, der er indsat i database blevet analyseret, så det har været muligt at beregne det forventede output. Stored procedures er testet ved at kalde en bestemt stored procedure, og herefter kontrollere om data er blevet gemt korrekt i databasen. Create procedurerne er simple, derfor kan det hurtig verificeres om data gemmes korrekt. Update procedurerne har mere logik, og kræver derfor grundig test. Logikken i databasen er opbygget på en sådan måde, at når en tuples primær nøgle er blevet refereret til, i en anden tabel, må denne tuple ikke længere opdateres. Derfor kontrolleres andre tabeller først, og hvis disse ikke indeholder en matchende fremmednøgle, kan tuplen opdateres.

8.1.1 Test og rapporter

Følgende tester om en ingrediensbatch, der ikke er brugt endnu, kan opdateres.

Forudsætninger:
Ingrediensbatchen bliver ikke brugt nogle andre steder i databasen.
En bruger har lov til at opdatere ingrediensbatches.

Test:
En bruger opdaterer en ingrediensbatch id.

Succesbetingelser:
Ingrediensbatchen's id er opdateret i databasen.

Test rapporten lister dato for sidste test, og om denne er en success.
Test rapport:

Test 2	Dato	Succes
Forudsætninger		
Ingrediensenbatchen bliver ikke brugt nogle andre stedet i databasen	16-06-2016	Ja
En bruger har lov til at opdatere ingrediensbatches	16-06-2016	Ja
Test		
En bruger opdaterer en ingrediensbatch id	16-06-2016	Ja
Succesbetingelser		
Ingrediensbatchen's id er opdateret i databasen	16-06-2016	Ja

Figur 8.1: Test rapport 2

flere testrapporter kan ses i bilag 13.1.

8.2 Web Applikation

Web applikationen er testet ved manuelle test af systemet, samt unit test af valideringsmetoderne. De manuelle test er udført hovedsageligt af udviklerne selv, men der er lavet en enkelt brugertest, for at se hvordan lægmænd interagerer med systemet. Dette for at højne brugervenligheden af det udviklede system.

8.2.1 Test og rapport - Brugertest

Forudsætninger:
Brugeren kan logge på hjemmesiden.

Test:
Brugeren kan se alle lister.
Brugeren opretter nye elementer i alle kategorier.
Brugeren kan se detaljer af de forskellige elementer.
Brugeren kan opdatere forskellige dele i databasen.
Brugeren kan se batches af en bestemt ingrediens.
Brugeren kan tilføje flere komponenter til en recept.

Succesbetingelser:
Listerne kan ses i displayet.
Der er nye elementer i databasen.
Detaljer bliver vist på skærmen.
Opdateringerne er gemt i databasen.
Listen over batches til en bestem ingrediens er vist på skærmen.
Ekstra komponenter er gemt i databasen.

Testbeskrivelsen for brugertesten viser de forventninger der helst skulle være

for en normal bruger af systemet. Det er kun testet på en bruger, dog ville der ved et større antal brugertest forventes, at ikke alle ville kunne udføre hele testen med succes. Resultatet af testen kan ses i testrapport under bilag 13.1.

For at sikre, at validering af input virker efter hensigten, er der lavet en række unittest af dette. Disse tester de forskellige metoder i FieldVerifier klassen. Her er der lavet et udsnit af mulige korrekte og forkerte input. Det er forsøgt at komme i gennem de mest gængse kombinationer.

8.2.2 Test og rapport - UnitTest

Testbeskrivelser og testrapporter for unit tests kan ses under bilag 13.1.

8.3 Afvejningsstyreenhed

I forbindelse med test af ASE'en er der oprettet en klasse "ListImpl", som implementerer interfacet "DatabaseCom". ListImpl er en stub klasse som implementere metoderne fra interfacet med hardcodede retur værdier. På denne måde har det været muligt at teste funktionaliteten i Afvejningsstyreenheden, uden at skulle være afhænging af en tilkoblet database.

9 Diskussion

Det var ikke alle punkter på kravspecifikationen, som blev implementeret. Projektet var ønsket lagt op på en server, dette blev ikke lavet, da vi ikke havde tiden til det. Ud fra oplægget CDIO Final bilag 5 [CDI], var udprintede produktioner ønsket, disse er ikke blevet implementeret, dette kunne være gjort, hvis vi havde haft mere tid. Noget af det sidste som vi har diskuteret, er nogle ekstra features til web applikationen. Blandt andet dropdown menu'er i valg af rolle og lister som kunne vise aktive og inaktive.

10 Konklusion

Systemet endte med at bestå af fire hovedkomponenter, som tilsammen udgør det samlede afvejningssystem til en medicinalvirksomhed.

Der er blevet oprettet en database, som er lagt op på en online server. Denne database indeholder informationer om alt fra operatører i virksomheden til informationer om de enkelte medicinske produkter. Tabellerne er blevet analyseret og sikret, at de overholder første, anden og tredje normalform.

Der er endvidere blevet udviklet en web applikation vha. Google web toolkit (GWT). Denne applikation er således i stand til gemme og vise informationer om en lang række aktiviter i virksomheden. Aktiviteter, så som at gemme informationer om indkøbte råvarer. For eksempel, hvor de er brugt og hvem der har leveret varen.

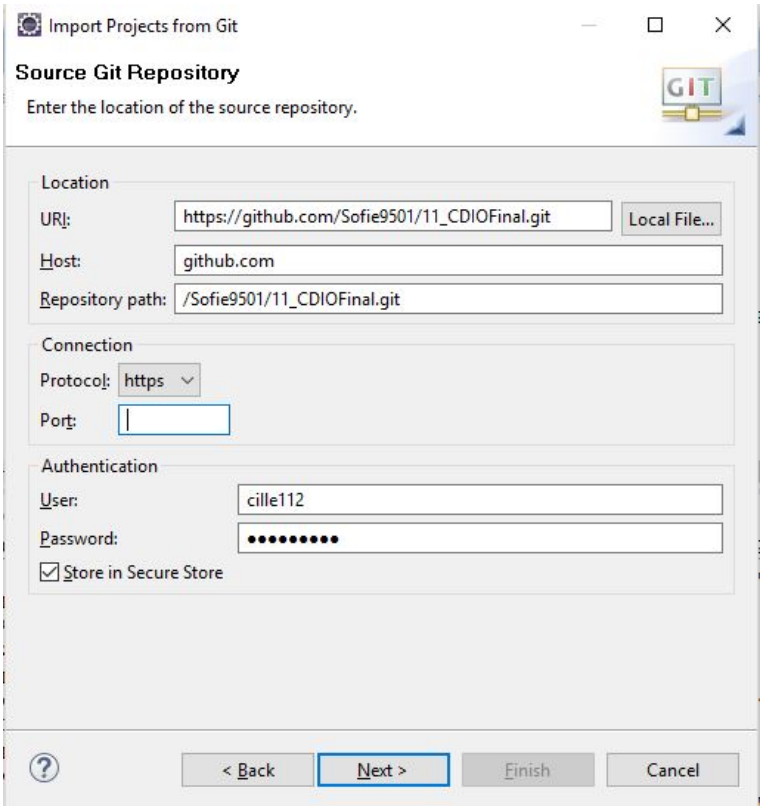
Der er blevet udviklet en afvejningsstyreenhed til at styre en vægts afvejningsproces. Ud fra de givne dokumenter, er det lykkedes at udvikle en flertrådet applikation til styring af en eller flere vejeterminaler. I forbindelse med afvejning vil der på hvert trin i afvejningen, blive kontrolleret, om det indtastede og vejede data er korrekt.

11 Appendiks: Konfiguration

11.1 Klon Repository

Det færdige program er lagt op på github og kan hentes derfra
I Eclipse skal der vælges at importere og klone fra git.
https://github.com/Sofie9501/11_CDIOFinal.git skal indsættes på URI feltets plads. I Host feltet indtastes "Github.com"og i Repository path indtastes "/Sofie9501/11_CDIOFinal.git

Når alt er indtastet, skulle vinduet gerne se ud som på figur 11.1 dog med en anden user og password.



Figur 11.1: Screenshot af, hvordan importvinduet skal se ud

11.2 PC konfiguration

Programmet er kørt og testet på følgende opsætning

- Styresystem: Windows 8 Pro 64 bit.
- CPU: Intel Core I5-4300U 1.9 Ghz

- RAM: 4 GB
- Java jdk version 1.8.0-u60
- Java version 8 Update 60
- GUI version 2.0.1
- Eclipse Mars Version: Mars Release (4.5.0), Build id: 20150621-1200

12 Litteraturliste

[CDI] CDIO. *CDIO Final Opgave beskrivelse*. https://docs.google.com/document/d/1Bcq1GHdgZh79TnQQqW2n3T3p8rwQBewad-J3p_6HWoE/edit. [Online; hentet 9-Juni-2016].

[DSD] DSDM. *MoSCoW Prioritisation*. https://www.dsdm.org/content/moscow-prioritisation. [Online; hentet 25-februar-2016].

13 Bilag

13.0.1 Database Entiteter og krav

Bilag 3

Database entiteter og attributter

Entitet: ***Råvare***			
Attributter:	***Datatype:***	***Beskrivelse:***	***Område:***
raavareId	Heltal	Råvare id	1-99999999
raavareNavn	Tekst	Råvare navn	2-20 karakterer
leverandoer	Tekst	Leverandør af råvare	2-20 karakterer

Entitet: ***RaavareBatch***			
Attributter:	***Datatype:***	***Beskrivelse:***	***Område:***
rbId	Heltal	Raavare batch id	1-99999999
raavareId	Heltal	Råvare id	1-99999999
maengde	Decimal	Lagerbeholdning i kilogram, med 4 decimaler	

Entitet: ***Recept***			
Attributter:	***Datatype:***	***Beskrivelse:***	***Område:***
receptId	Heltal	Recept Id	1-99999999
receptNavn	Tekst	Recepr navn	2-20 karakterer
(1..N)			
raavareId	Heltal	Råvare id	1-99999999
nonNetto	Decimal	Mængden i kilogram, med 4 decimaler	0,05-20,0 kg
tolerance	Decimal	Tolerancen i procent på nominel mængde.	0,1 til 10,0%

Entitet: ***Produktbatch***			
Attributter:	***Datatype:***	***Beskrivelse:***	***Område:***
pbId	Heltal	Produkt Batch id	1-99999999
receptId	Heltal	Recept Id	1-99999999
status	Heltal	Ikke påbegyndt = 0/ Under Produktion = 1/ Afsluttet = 2	0-2
(1..N)			

Figur 13.1: krav del 1

oprId	Heltal (Fremmednøgle)	Operatør id	1-99999999
rbId	Heltal	Raavare batch id	1-99999999
tara	Decimal	Tara i kg med 4 decimaler	
netto	Decimal	Netto i kg med 4 decimaler	

Entitet: ***Operatør***			
Attributter:	***Datatype:***	***Beskrivelse:***	***Område:***
oprId	Heltal	Operatør id	1-99999999
oprNavn	Tekst	Operatør navn	2-20 karakterer
ini	Tekst	Operatør initialer	2-4 karakterer
cpr	Tekst	Operatør cpr nr	10 karakterer
password	Tekst	Password der anvendes ikke kryptering	5 -8 karakterer

Figur 13.2: krav del 3

13.0.2 Use cases

Use case:	Bruger Administration
ID:	U1
Beskrivelse:	Dette use case beskriver oprettelsen af brugere i systemet. Dette gøres af administratoren
Primære aktører:	Administrator
Sekundære aktører:	Ingen
Preconditions:	Administrator er logget ind på web applikationen
Main flow:	1. Administrator vælger menuen "Operator Administration" 2. Farmaceuten vælger "Create New Operator" 3. Operator ID, Name, CPR number, Password og Role indtastes, og administratoren trykker "Save", når den er færdig
Postcondtions:	Bruger er oprettet
Alternative flows:	Ingen

Figur 13.3: Usecase 1: Bruger administration

Use case:	Råvare Administration
ID:	U2
Beskrivelse:	Dette use case beskriver oprettelsen af råvarer. Dette gøres af værkføreren
Primære aktører:	Farmaceut
Sekundære aktører:	Ingen
Preconditions:	Farmaceuten er logget ind på web applikationen
Main flow:	1. Farmaceuten vælger menuen "Ingredient Administration" 2. Farmaceuten vælger "Create ingredient" 3. Ingredient ID, Ingredient Name og Supplier indtastes, og farmaceuten klikker "Save" når den er færdig
Postcondtions:	Råvare er oprettet
Alternative flows:	Ingen

Figur 13.4: Usecase 2: Råvare administration

Use case:	Råvarebatch Administration
ID:	U4
Beskrivelse:	Dette use case beskriver oprettelsen af råvarebatches. Dette gøres af værkføreren
Primære aktører:	Værkfører
Sekundære aktører:	Ingen
Preconditions:	Værkføreren er logget ind på web applikationen
Main flow:	1. Værkføreren vælger menuen "Ingredient Batch Administration" 2. Værkføreren vælger "Create Ingredient Batch" 3. Ingredient Batch ID, Ingredient ID og amount indtastes, og værkføreren trykker "Save"
Postcondtions:	Råvarebatch er oprettet
Alternative flows:	Ingen

Figur 13.5: Usecase 4: Råvarebatch administration

Use case:	Produktbatch Administration
ID:	U5
Beskrivelse:	Dette use case beskriver oprettelsen af produktbatches, hvilket gøres af værkføreren
Primære aktører:	Værkfører
Sekundære aktører:	Ingen
Preconditions:	Værkføreren er logget ind på web applikationen
Main flow:	1. Værkføreren vælger menuen "Productbatch Administration" 2. Værkføreren vælger "Create Product batch" 3. Productbatch ID og Recept ID indtastes, og værkføreren trykker "Save"
Postcondtions:	Produktbatch er oprettet
Alternative flows:	Ingen

Figur 13.6: Usecase 5: Produktbatch administration

13.1 Test beskrivelser og rapporter

13.1.1 Test 1 stored procedure create_ingredient

Forudsætninger:
Ingrediensen findes ikke i forvejen
En bruger må oprette nye ingredienser
Test:
En bruger opretter en ny ingrediens
Succesbetingelser:
Ingrediensen er oprettet i databasen

Test rapport:

Test 1	Dato	Succes
Forudsætninger		
Ingrediensen er ikke oprettet i forvejen	16-06-2016	Ja
En bruger må oprette nye ingredienser	16-06-2016	Ja
Test		
En bruger opretter en ny ingrediens	16-06-2016	Ja
Succesbetingelser		
Ingrediensen er gemt i databasen	16-06-2016	Ja

Figur 13.7: Test rapport 1

13.1.2 Test 2 stored procedure update_ingredientbatch

Forudsætninger:
Ingrediensenbatchen bliver ikke brugt nogle andre stedet i databasen
En bruger har lov til at opdatere ingrediensbatches
Test:
En bruger opdaterer en ingrediensbatch id
Succesbetingelser:
Ingrediensbatchen's id er opdateret i databasen

Test rapport:

Test 2	Dato	Succes
Forudsætninger		
Ingrediensenbatchen bliver ikke brugt nogle andre stedet i databasen	16-06-2016	Ja
En bruger har lov til at opdatere ingrediensbatches	16-06-2016	Ja
Test		
En bruger opdaterer en ingrediensbatch id	16-06-2016	Ja
Succesbetingelser		
Ingrediensbatchen's id er opdateret i databasen	16-06-2016	Ja

Figur 13.8: Test rapport 2

13.1.3 Test 3 view productbatch_administration

Forudsætninger:
Mindst en productbatch er oprettet i databasen
En bruger har adgang til dette view
Test:
En bruger åbner productbatch listen
Succesbetingelser:
Listen viser; pb_id, recipe_id, recipe_name, finished, total, start date, end date og active

Test rapport:

Test 3	Dato	Succes
Forudsætninger		
Mindst en productbatch er oprettet i databasen	16-06-2016	Ja
En bruger har adgang til dette view	16-06-2016	Ja
Test		
En bruger åbner productbatch listen	16-06-2016	Ja
Succesbetingelser		
Listen viser; pb_id, recipe_id, recipe_name, finished, total, start date, end date og active	16-06-2016	Ja

Figur 13.9: Test rapport 3

13.1.4 Test 4 fieldverifier cpr

Forudsætninger:
Ingen, da testen tester en statisk metode
Test:
260184-1234 valideres
430372-4321 valideres
1b0865-3125 valideres
Succesbetingelser:
260184-1234 skal returnere true
430372-4321 skal returnere false
1b0865-3125 skal returnere false

Test rapport:

Test 4	Dato	Succes
Forudsætninger		
Ingen		
Test		
260184-1234 valideres	15-06-2016	Ja
430372-4321 valideres	15-06-2016	Ja
1b0865-3125 valideres	15-06-2016	Ja
Succesbetingelser		
260184-1234 skal returnere true	15-06-2016	Ja
430372-4321 skal returnere false	15-06-2016	Ja
1b0865-3125 skal returnere false	15-06-2016	Ja

Figur 13.10: Test rapport 4

13.1.5 Test 5 fieldverifier rolle

Forudsætninger:
Ingen, da testen tester en statisk metode
Test:
0 valideres
1 valideres
2 valideres
3 valideres
4 valideres
5 valideres
max værdi for int valideres
-1 valideres
min værdi for int valideres
Succesbetingelser:
0 skal returnere false
1 skal returnere true
2 skal returnere true
3 skal returnere true
4 skal returnere true
5 skal returnere false
max værdi for int skal returnere false
-1 skal returnere false
min værdi for int skal returnere false

Test rapport:

Test 5	Dato	Succes
Forudsætninger		
Ingen		
Test		
0 valideres	15-06-2016	Ja
1 valideres	15-06-2016	Ja
2 valideres	15-06-2016	Ja
3 valideres	15-06-2016	Ja
4 valideres	15-06-2016	Ja
5 valideres	15-06-2016	Ja
max værdi for int valideres	15-06-2016	Ja
-1 valideres	15-06-2016	Ja
min værdi for int valideres	15-06-2016	Ja
Succesbetingelser		
0 skal returnere false	15-06-2016	Ja
1 skal returnere true	15-06-2016	Ja
2 skal returnere true	15-06-2016	Ja
3 skal returnere true	15-06-2016	Ja
4 skal returnere true	15-06-2016	Ja
5 skal returnere false	15-06-2016	Ja
max værdi for int skal returnere fals	15-06-2016	Ja
-1 skal returnere false	15-06-2016	Ja
min værdi for int skal returnere false	15-06-2016	Ja

Figur 13.11: Test rapport 5

13.1.6 Test 6 fieldverifier password

Forudsætninger:
Ingen, da testen tester en statisk metode
Test:
1aB! valideres
ABCDEF valideres
abcdef valideres
123456 valideres
abcdeF valideres
abcde1 valideres

ABCDE1 valideres
12345. valideres
123ten$ valideres
Abcde1 valideres
Succesbetingelser:
1aB! skal returnere false
ABCDEF skal returnere false
abcdef skal returnere false
123456 skal returnere false
abcdeF skal returnere false
abcde1 skal returnere false
ABCDE1 skal returnere false
12345. skal returnere false
123ten$ skal returnere false
Abcde1 skal returnere true

Test rapport:

Test 6	Dato	Succes
Forudsætninger		
Ingen		
Test		
1aB! Valideres	15-06-2016	Ja
ABCDEF valideres	15-06-2016	Ja
abcdef valideres	15-06-2016	Ja
123456 valideres	15-06-2016	Ja
abcdeF valideres	15-06-2016	Ja
abcde1 valideres	15-06-2016	Ja
ABCDE1 valideres	15-06-2016	Ja
12345. valideres	15-06-2016	Ja
123ten\$ valideres	15-06-2016	Ja
Abcde1 valideres	15-06-2016	Ja
Succesbetingelser		
1aB! skal returnere false	15-06-2016	Ja
ABCDEF skal returnere false	15-06-2016	Ja
abcdef skal returnere false	15-06-2016	Ja
123456 skal returnere false	15-06-2016	Ja
abcdeF skal returnere false	15-06-2016	Ja
abcde1 skal returnere false	15-06-2016	Ja
ABCDE1 skal returnere false	15-06-2016	Ja
12345. skal returnere false	15-06-2016	Ja
123ten\$ skal returnere false	15-06-2016	Ja
Abcde1 skal returnere true	15-06-2016	Ja

Figur 13.12: Test rapport 6

13.1.7 Test 7 bruger test

Forudsætninger:
Brugeren kan logge på hjemmesiden
Test:
Brugeren kan se alle lister
Brugeren opretter nye elementer i alle kategorier
Brugeren kan se detaljer af de forskellige elementer
Brugeren kan opdatere forskellige dele i databasen

Brugeren kan se batches af en bestemt ingrediens
Brugeren kan tilføje flere komponenter til en recept
Succesbetingelser:
Listerne kan ses i displayet
Der er nye elementer i databasen
Detaljer bliver vist på skærmen
Opdateringerne er gemt i databasen
Listen over batches til en bestemt ingrediens er vist på skærmen
Ekstra komponenter er gemt i databasen

Test rapport:

Test 7	Dato	Succes
Forudsætninger		
Brugeren kan logge på hjemmesiden	16-06-2016	Ja
Test		
Brugeren kan se alle lister	16-06-2016	Ja
Brugeren opretter nye elementer i alle kategorier	16-06-2016	Ja
Brugeren kan se detaljer af de forskellige elementer	16-06-2016	Ja
Brugeren kan opdatere forskellige dele i databasen	16-06-2016	Ja
Brugeren kan se batches af en bestemt ingrediens	16-06-2016	Ja
Brugeren kan tilføje flere komponenter til en recept	16-06-2016	Ja
Succesbetingelser		
Listerne kan ses i displayet	16-06-2016	Ja
Der er nye elementer i databasen	16-06-2016	Ja
Detaljer bliver vist på skærmen	16-06-2016	Ja
Opdateringerne er gemt i databasen	16-06-2016	Ja
Listen over batches til en bestemt ingrediens er vist på skærmen	16-06-2016	Ja
Ekstra komponenter er gemt i databasen	16-06-2016	Ja

Figur 13.13: Test rapport 7