



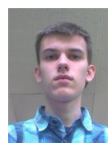
02324

VIDREGÅENDE PROGRAMMERING

CDIO - Final

Gruppe 17

s185118
Aleksander Lægsgaard Jørgensen



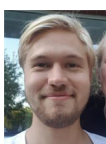
s175561
Andreas Østergaard Schliemann

s185096
Jonatan Amtoft Dahl



s164177
Josephine Weirøse

s185103
Søren Hother Rasmussen



s185121
Theodor Peter Guttesen

10. juni 2019

DTU Compute

Institut for Matematik og Computer Science

DTU Diplom

Center for Diplomingeniøruddannelse

Kode og Git: https://github.com/aleksanderlj/17-CDIO_Final-2Sem

Abstract

Indhold

1	Indledning	3
2	Timeregnskab	4
3	Krav	5
3.1	Vision - Andreas	5
3.2	Navneordsanalyse - Aleksander	5
3.3	Kravliste - Andreas og Josephine	5
3.4	Domænemodel - Theodor	7
4	Analyse	9
4.1	Aktører	9
4.1.1	Use case diagram - Andreas	9
4.2	Use case - Theodor og Aleksander	10
4.2.1	Brugeradministration	10
4.2.2	Råvareadministration	10
4.2.3	Receptadministration	10
4.2.4	Råvarebatchadministration	10
4.3	Traceability matrix: Use cases og krav - Josephine	12
4.4	System sekvensdiagram - Theodor, Søren, Jonatan og Andreas	13
5	Design	14
5.1	Hjemmeside - Theodor	14
5.2	Klassediagram - Theodor	14
5.3	Pakkediagram - Theodor	16
5.4	Sekvensdiagram	16
5.5	Database design - Søren	16
6	Implementering	17
7	Test	18
7.1	Positiv test	18
7.2	Negative test	18
7.3	Unit-test	18
7.4	Coverage test	18
8	Projektplanlægning	19
8.1	Planlagt forløb - Josephine	19
8.2	Reelt forløb	19
9	Konklusion	20
10	Bilag	21

1 Indledning

I dette projekt har vi lavet en softwareløsning, der kan bruges til dokumentation af råvarebatch forbrug og afvejning af råvarer.

2 Timeregnskab

Opgave	Josephine	Aleksander	Søren	Jonatan	Andreas	Theodor	Sum
Rapport	6	0.3	4	6	7	14	37.3
Webpage	0	16	2	0	0	0	18
Database	0	0	7	5	0	0	12
Vægt	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
Projektorganisering	2	2	2	2	2	2	12
Test	0	0	0	0	0	0	0
Sum	8	18.3	15	13	9	16	79.3

3 Krav

3.1 Vision - Andreas

Medicin virksomheden, Panther Pharma, ønsker et softwaresystem, de kan bruge til afvejning af råvare og dokumentation af afvejningen og råvarebatch-forbrug. I dette system vil det være muligt at definere råvare og indskrive råvarebatches, som sendes fra leverandører og stilles på Panther Pharmas lager. Hvert råvarebatch vil indeholde en mængde råvarestof, der bruges af til produktion af produktbatches. Det vil også være muligt at definere recepter, som indeholder en liste af råvare og den mængde, der skal bruges til produktion. Disse recepter vil blive anvendt til at producere produktbatches, som indeholder information om, hvilken recept, der er blevet brugt, hvilke råvarebatches, som er blevet brugt og den aktuelle mængde af dem, der blev brugt. I systemet er fire roller, administrator, produktionsleder, farmaceut og laborant, som brugere kan have, der hver har forskellige rettigheder.

3.2 Navneordsanalyse - Aleksander

Råvarer, Råvarebatches, Recepter, Produktbatches.
Administrator, Farmaceut, Produktionsleder, Laborant.

3.3 Kravliste - Andreas og Josephine

ID	Funktionelle krav
Must have	
M01	Systemet skal kunne oprette og redigere råvarer
M02	Systemet skal kunne oprette og redigere råvarebatches
M03	Systemet skal kunne oprette og redigere produktbatches
M04	Systemet skal kunne oprette og redigere recepter
M05	Systemet skal kunne oprette og rediger bruger
M06	Systemet skal ikke kunne slette bruger men skal markerer dem som inaktive
M07	Systemet skal kunne afveje råvarer gennem en vejeterminal
M08	Systemet skal kun tillade de rigtige roller adgang til de rigtige funktioner
M09	Systemet skal kunne vise en brugerliste
M10	Systemet skal kræve at brugeren angiver rolle på forsiden
M11	Systemet skal garanterer at råvare, råvarerbatch, recepter, produktbatches og brugere alle har unikke ID
M12	Systemet skal holde styr på produktbatches status (Oprette/Under produktion/Afsluttet)
M13	Systemet skal kunne åbne og lukke forbindelse til enten en virtuel eller en virkelig vægt, og disse skal kunne snakke frem og tilbage mellem vægten og systemet
M14	Vægten skal kunne bruge kommandoerne S, T, D, DW, p111, RM20 8
M15	Vægten skal kunne ud fra operatørnummre finde operatørens navn

Should have	
S01	Systemet skal holde styr på den aktuelle mængde råvare der er blevet brugt i et givent produktbatch
S02	Systemet skal kunne holde styr på tilbageværende mængde af råvarerbatch
S03	Systemet skal sørger for at recepter har tolerance på alle råvarer
Could have	
C01	Systemet skal holde styr på produktbatches oprettelses dato
C02	Systemet skal lave en produktionsforskrift som kan printes
Won't have	
W01	Systemet skal kræve login fra bruger

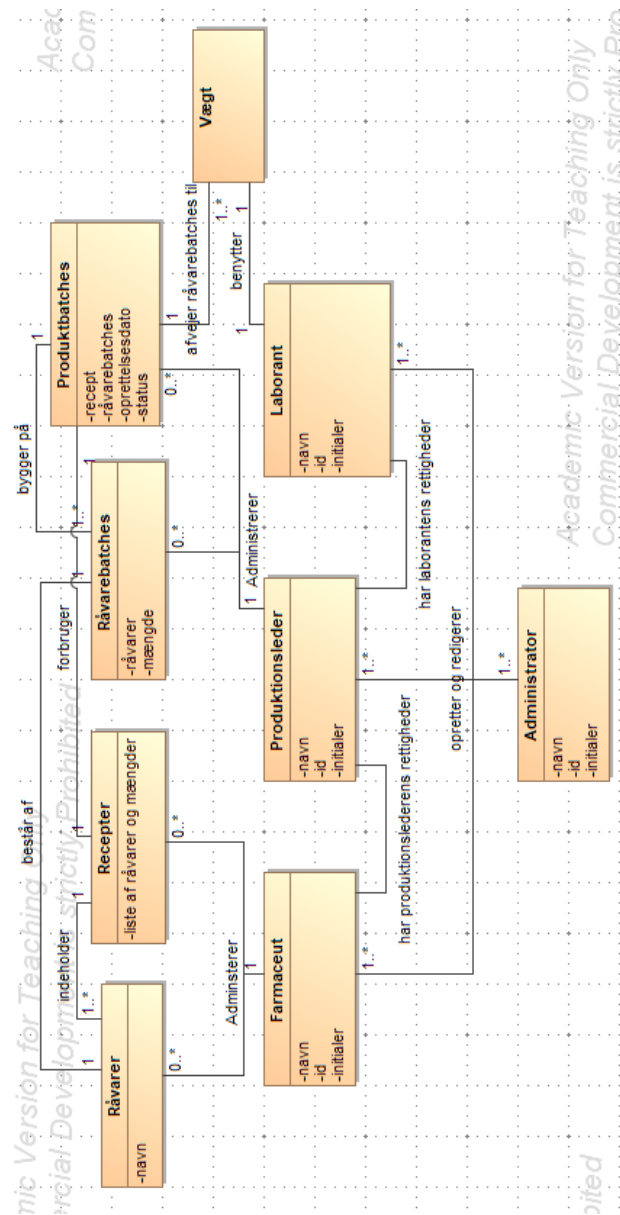
ID	Ikke-funktionelle krav
Must have	
IF01	Systemet skal være implementeret som en Single page application med en REST-backend og en HTML/CSS/JavaScript baseret frontend.
IF02	Data skal lagres i en persistent database mellem sessioner.
IF03	Der skal sendes til/fra Rest-backenden serialiseret på en måde (eks. JSON) så andre kan anvende servicen
IF04	Modulet skal kunne tilgås via. en webbrowser
IF05	Vægten skal tale med databasen gennem websidens REST API.

3.4 Domænemodel - Theodor

Vi har foretaget en navneordsanalyse og fundet alle navneord relevante for opgaven. Derefter har vi sorteret dem i klasse-kandidater og attribut-kandidater.

Klasser	Attributter
Råvarer	leverandører
Råvarebatches	mængde
Recepter	id
Produktbatches	navn
Farmaceut	initialer
Produktionsleder	tolerance
Laborant	dato
Administrator	status
Vægt	

Domænediagrammet viser klasserne og deres attributter, som de er i virkeligheden og beskriver forholdet mellem dem. Domænediagrammet ligger til grund for bl.a. designklassediagrammet.



Figur 1: Domænediagram

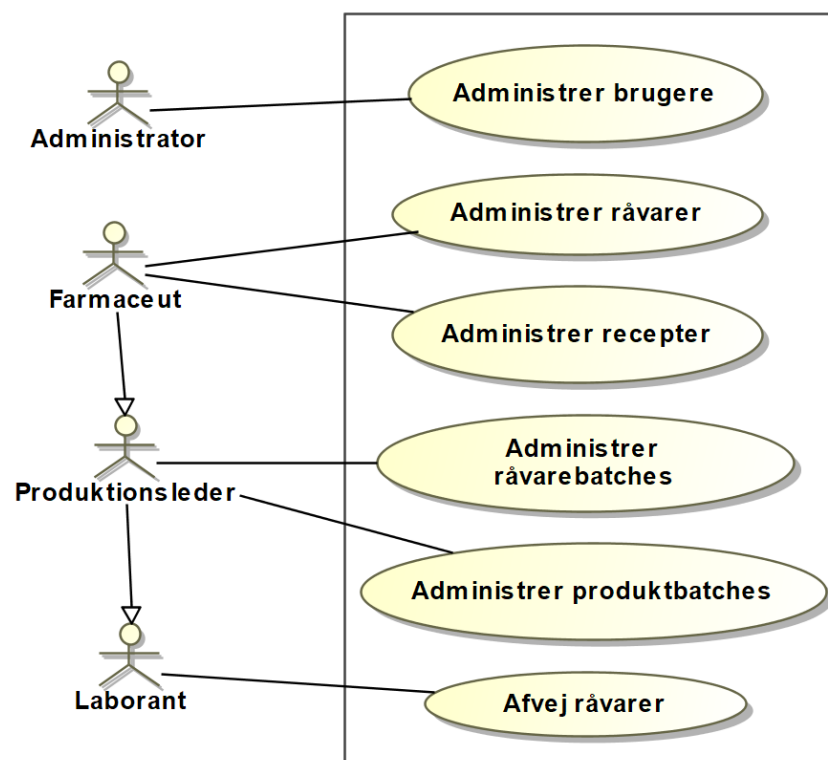
4 Analyse

4.1 Aktører

Aktørtype	Aktører
Primære	Administrator, farmaceut, produktionsleder og laborant
Supporterende	
Offstage	

4.1.1 Use case diagram - Andreas

Vi har i alt fire aktører, administrator, farmaceut, produktionsleder og laborant. Administratoren sørger for at oprette, redigere og deaktivere brugere i systemet. Farmaceuten kan oprette, redigere og deaktivere råvarer og recepter. Farmaceuten kan også det samme som produktionslederen, som kan bestille nye råvarebatches og oprette dem i systemet og igangsætte produktionsbatches. Produktionslederen kan også det samme som laboranten, der kun kan afveje råvare.



Figur 2: Usecase diagram

4.2 Use case - Theodor og Aleksander

4.2.1 Brugeradministration

Den overordnede administration af brugere i systemet foretages af administrator aktøren. Denne skal kunne oprette, rette, fjerne og vise brugere i systemet. Ved oprettelsen angives brugerens ID (entydigt), navn, initialer.

En bruger, der én gang er oprettet i systemet, kan ikke slettes men kan inaktiveres.

4.2.2 Råvareadministration

Administrationen af råvarer i systemet foretages af farmaceut aktøren. Denne skal kunne oprette, rette samt vise råvarer i systemet.

En råvare defineres ved et råvareNr (brugervalgt og entydigt) og navn. (d.v.s. råvareNr skal IKKE autogenereres.)

4.2.3 Receptadministration

Administrationen af recepter foretages af farmaceut aktøren. Denne skal kunne oprette samt vise recepter i systemet.

En recept defineres ved et receptNr (brugervalgt og entydigt), navn samt en sekvens af receptkomponenter.

En receptkomponent består af en råvare (type), en mængde samt en tolerance.

4.2.4 Råvarebatchadministration

Administrationen af råvarebatches i systemet foretages af produktionslederen. Denne skal kunne oprette samt vise råvarebatches i systemet.

En råvarebatch defineres ved et råvarebatchNr (brugervalgt og entydigt) samt mængde og leverandør.

Use case: Administrere produktbatches
ID: UC5
Kort beskrivelse: Produktionslederen opretter produktbatches i systemet
Primære aktører: Produktionslederen
Preconditions: råvarerne og råvarebatches er oprettet i systemet.
Main flow: 1. Produktionslederen definerer produktBatchNr, receptNr, dato for oprettelse, status for batchen hhv. oprettet/under produktion/ afsluttet. 2. Afvejningsresultater for enkelte receptkomponenter gemmes som en produktbatchkomponent, i takt med produktet fremstilles. 3. Når produktbatchet er oprettet udprintes det på papir og uddeles til en udvalgt laborant, der herefter har ansvaret for produktionen.
Postconditions: Når produktbatchet er oprettet kan laboranten gå i gang med at producere det.
Alternative flows:

Fully dressed

I arbejdet med use cases kan der benyttes tre forskellige grader af beskrivelse. Her er der tale om "brief", "casual" og "fully-dressed". Brief er den mindst omfattende og fully-dressed er den mest omfattende.

Use case: Afveje råvarer
ID: UC6
Kort beskrivelse: Laboranten afvejer råvarer fra råvarebatches til fremstilling af produktbatches
Primære aktører: Laborant
Preconditions: Produktionslederen har defineret et nyt produktbatch, der er blevet udprintet og uddelt til laboranten.
Main flow: 1. Laboranten indtaster sit laborantNr i vejeterminalen. 2. Vejeterminalen svarer tilbage med laborantens navn. 3. Laboranten indtaster nummeret på produktbatchet. Produktbatchets status ændres fra 'oprettet' til 'under produktion'. 4. Vejeterminalen beder laboranten om at placere en beholder. 5. Beholderens vægt registreres. 6. Vægten tareres og tarabelastningen gemmes. 7. Vejeterminalen oplyser hvilken råvare der skal afvejes. 8. Laboranten henter råvaren og oplyser råvarens batchnummer. 9. Laboranten afvejer råvaren indenfor tolerancen. 10. Vægten udfører brutto-kontrol og registrerer afvejningsresultatet i <i>produktbatchkomponenten</i> . Trin 7-10 gentages for alle råvare i produktbatchet.
Postconditions: produktbatchets status ændres fra 'under produktion' til 'afsluttet'.
Alternative flows: 2.1 laborantNr eksisterer ikke og terminalen giver en fejlmeddelelse og beder om laborantNr igen. 3.1 Produktbatchet findes ikke eller er allerede afsluttet, så vejeterminalen giver fejlmeddelelse og beder om produktbatch nummeret igen. 8.1 Batchnummeret er ukendt eller batchet er opbrugt. så vejeterminalen giver fejlmeddelelse og beder om batchnummeret igen. 10.1 Brutto-kontrollen giver ikke det ønskede resultat. Laboranten kontrollerer for fejl.

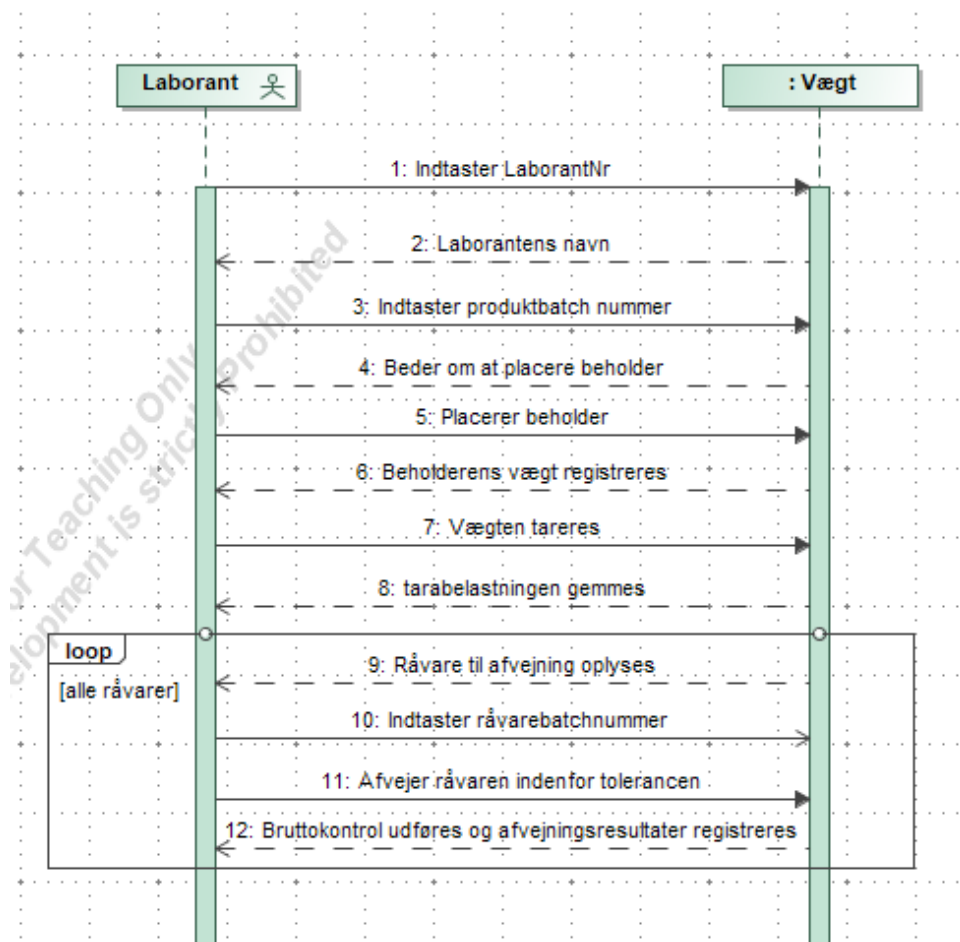
4.3 Traceability matrix: Use cases og krav - Josephine

I følgende tabel ses sammenhængen mellem use cases og krav, de grønne er dem der er implementeret, de røde er endnu ikke implementeret.

Krav	Use Case	UC01	UC02	UC03	UC04	UC05	UC06
M01			x				
M02					x		
M03						x	
M04				x			
M05		x					
M06		x					
M07							x
M08		x	x	x	x	x	x
M09		x					
M10		x					
M11		x	x	x	x	x	x
M12						x	
M13							x
M14							x
M15							x
S01			x				
S02					x		
S03				x			
C01						x	
C02						x	

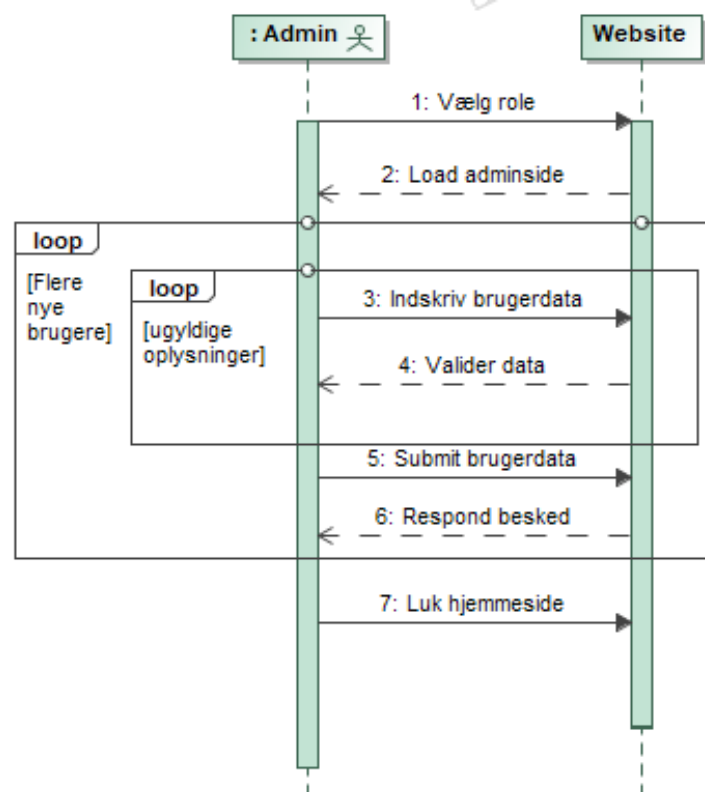
4.4 System sekvensdiagram - Theodor, Søren, Jonatan og Andreas

Dette systemsekvensdiagram viser, hvordan en laborant interagerer med vægten. Først vil laboranten indtaste sit id, hvorefter vægten viser laborantens navn. Derefter indtastes produktbatch id, hvor vægten så beder om, at en beholder stilles på vægten. Når en beholder er stillet på vægten gemmes beholders vægt og vægten tarares. Herefter indskrives den råvare, der skal afvejes ved at indtaste råvarebatchid. Så afvejes råvaren indefor tolerancen, bruttokontrol udføres og vægten af råvaren registreres. Gentag for alle råvare i recepten.



Figur 3: Vægt systemsekvens

I dette systemsekvensdiagram vises en administrators interaktion med websiden. Først vælger brugeren at være en administrator, hvorefter administrator siden vises. Her indtaster administrator brugerdata for en ny bruger. Derefter validerer websiden den indtastede data, hvis den ikke er valid, skal administratoren prøve igen. Hvis dataen er valid kan administratoren gemme den, hvorefter websiden returnerer en om besked om, at den er blevet gemt. Gentag for eventuelle andre nye brugere. Administratoren lukker websiden, når de er færdig.



Figur 4: Admin opretter bruger

5 Design

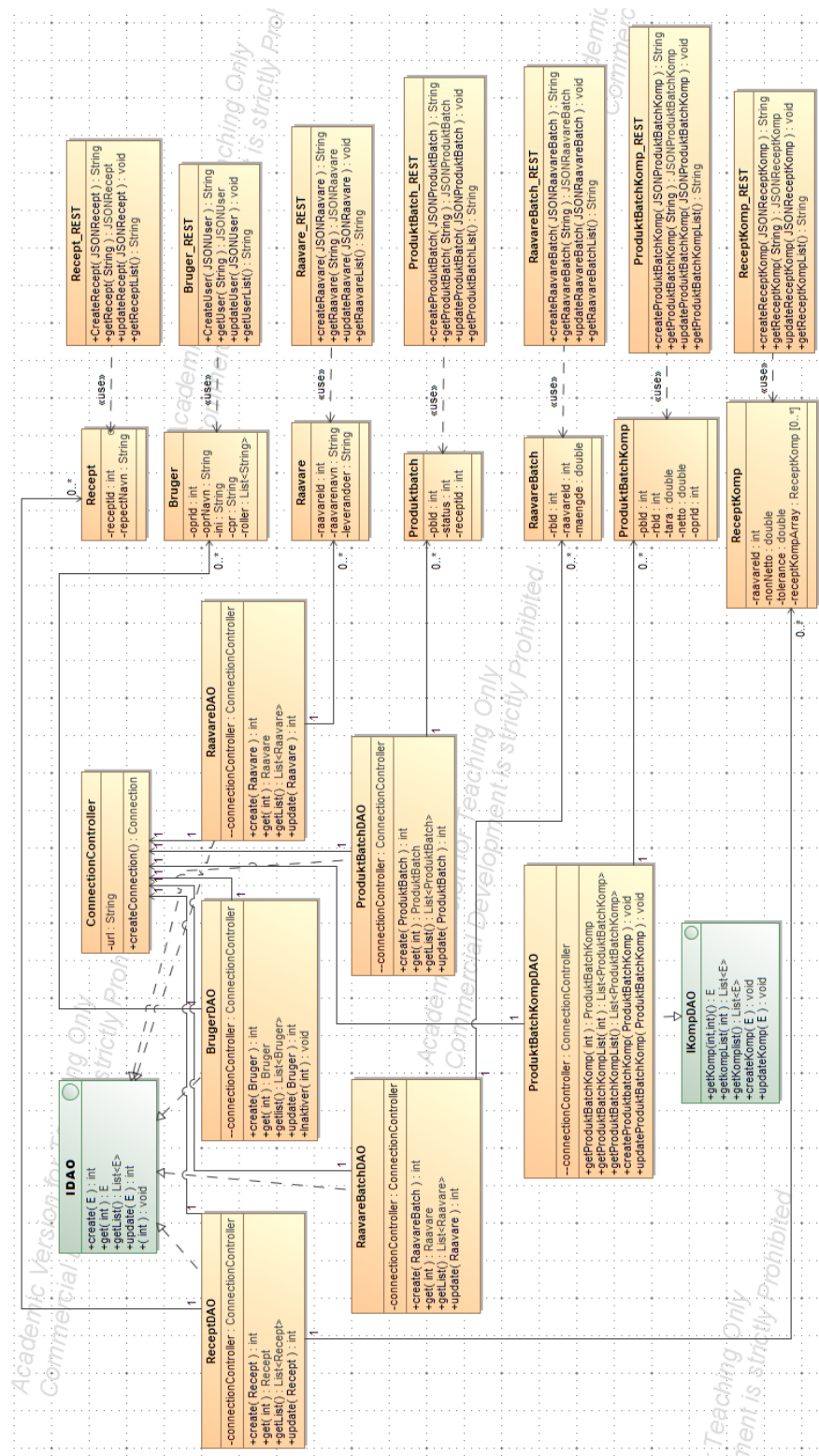
5.1 Hjemmeside - Theodor

Vi har nøje overvejet brugervenlighed i vores design af hjemmesiden. Vores mål har været, at gøre det nemt at få fat i de oprettelses og redirektionsmenuer, hver rolle må bruge. Derfor har vi brugt en singlepage application.

5.2 Klassediagram - Theodor

Design klassegrammet er udgangspunktet for programmet. Designklassediagrammet er udarbejdet ud fra domænemodellen og use cases. Det er et statisk UML diagram. Vi har inkluderet attributter og metoder, såvel som klassens navn og multiplicitet i vores klassediagram. Desuden viser designklassediagrammet relationerne klasserne imellem.

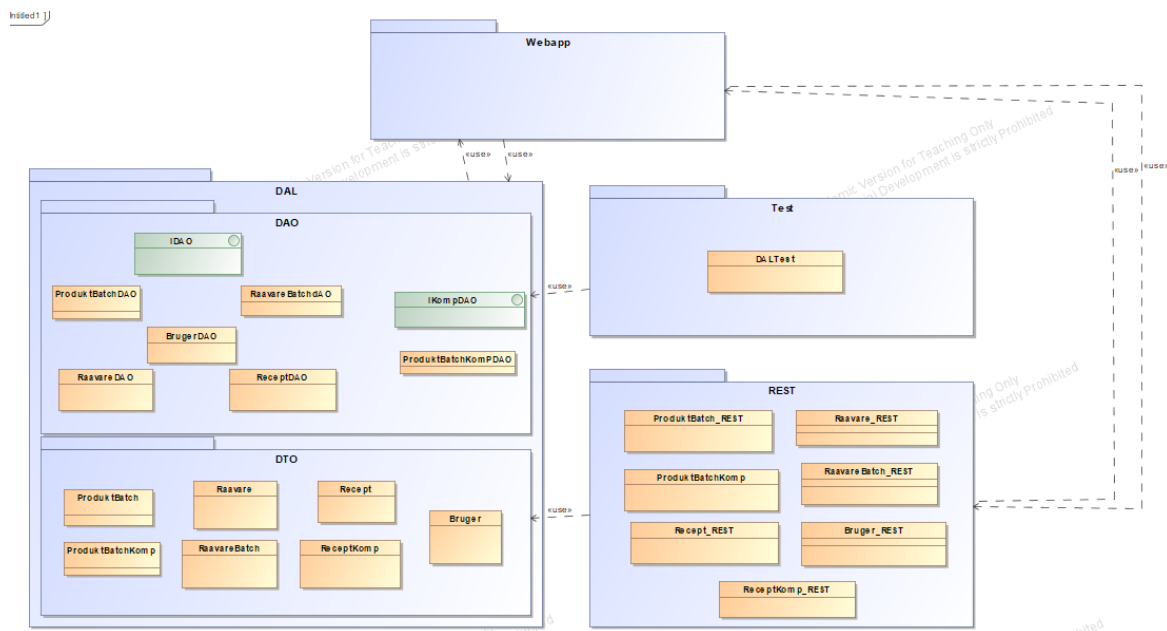
Programmet kan deles op i DAO'er, DTO'er, REST og nogle hjælpeklasser. DAO'erne overholder interfaces hhv. IDAO og IKompDAO. DAO'erne får deres connections til databasen fra en ConnectionController klasse. DTO'erne bruges også af REST klasserne til at få data fra.



Figur 5

5.3 Pakkediagram - Theodor

Pakkediagrammet giver overblik over de forskellige pakker i programmet og hvilke pakker der har med hinanden at gøre.



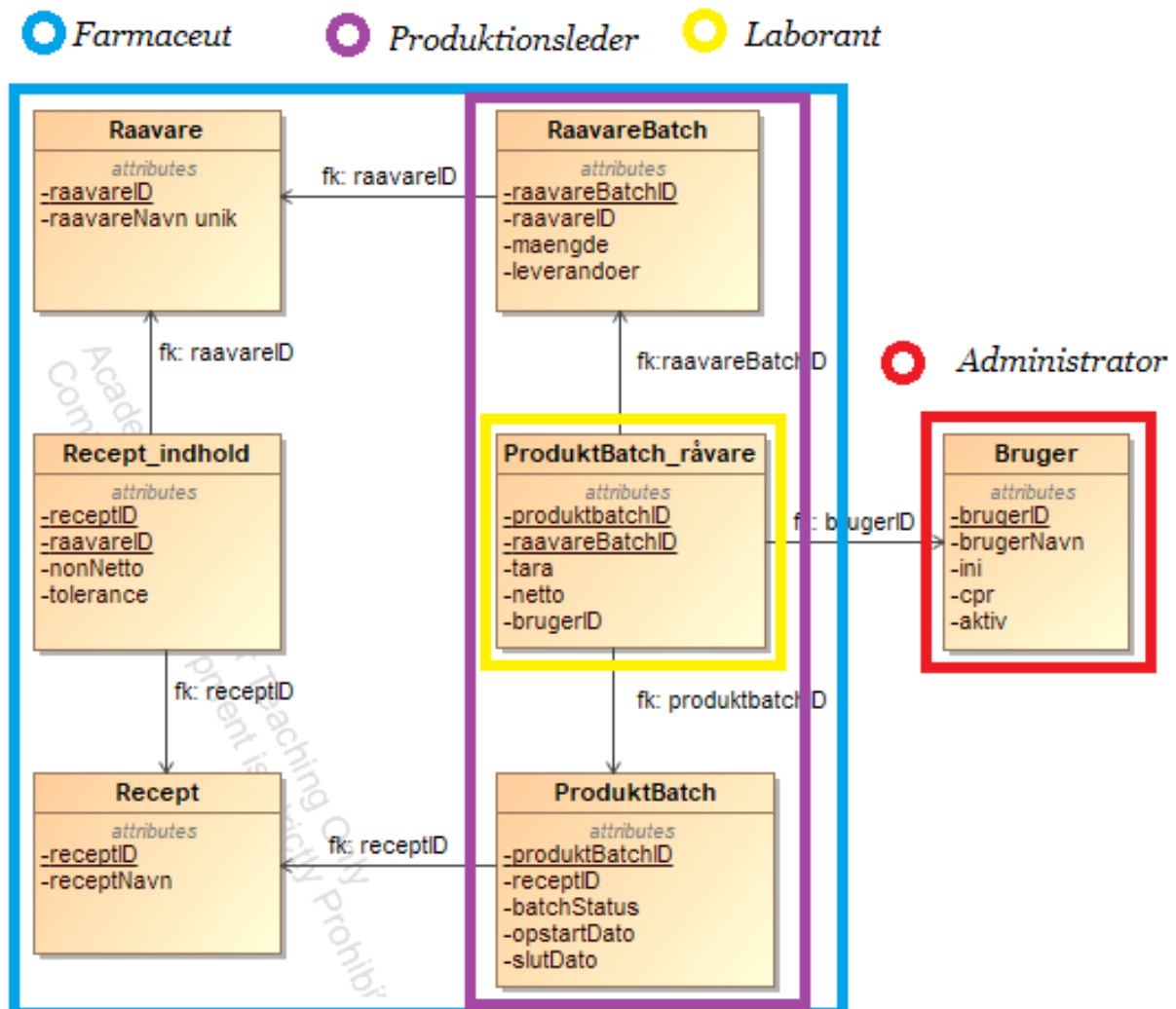
Figur 6

5.4 Sekvensdiagram

5.5 Database design - Søren

For at lave et fornuftigt design af databasen, er der undersøgt hvad der skal administreres. Dette er brugere, råvarer, recepter, råvarebatchs og produktionsbatchs, derfor er disse 5 oprettet som tabeller med relevante tabelnavne. Hver recept har dog flere råvarer for hver recept, så der er oprettet endnu en tabel for at holde styr på dette kaldt Recept_indhold. Det samme gælder produktbatchs og råvarebatchs, så derfor er der oprettet tabellen Produktbatch_indhold (navnet er rettet efter billedet)

Hver af de 5 ting der skal administreres, benytter kun en primary key, som kandidat nøgle, så det er let at



Figur 7: MIDLERTIDIGT SKEMADIAGRAM! UDSKIFT MED DET UDEN FARVER EFTER MIL 1

6 Implementering

Element	Class	Method	Line
	%	%	%
	%	%	%
	%	%	%
	%	%	%
	%	%	%

Figur 8: Coverage Test

Tabel 5: Coverage Test

7 Test

7.1 Positiv test

Funktionelle krav	Positiv test	Bestået
M01		
M02		
M03		
S01		
S02		
S03		
C01		
C02		
C03		

7.2 Negative test

7.3 Unit-test

Unit-tests er automatiske tests. Vi har brugt JUnit4 i IntelliJ, til at lave vores unit-tests. En typisk test bruger en assertEquals metode, for at se om metodens output er det samme som forventet output. Hvis det virkelige resultat og det forventede stemmer overens er testen bestået. Unit-tests kan findes i mappen Test under src i programmet.

Test Case ID	Beskrivelse	Status
TC01		
TC02		

7.4 Coverage test

8 Projektplanlægning

8.1 Planlagt forløb - Josephine

I dette projekt har vi besluttet os for en fremgangsmåde som gerne skulle gøre processen mere flydende end hvad vi tidligere har haft. Vi vil fra start af i projektet gennemgå og diskutere diagrammerne der skal indgå i vores rapport. Desuden vil vi for at opnå en god rapport, gode diagrammer og undgå waterfall metoden starte med at lave analyse afsnittet derefter design afsnittet, før vi begynder at lave koden, vi vil undgå solokodning.

Vi vil arbejde iterativt og prioritere de vigtigste use cases og krav og lægge dette ind i vores tidsplan. Vi vil så lave vores Master og Development branch i IntelliJ, og undlade og pushe kode der giver errors til dem. Lave kravliste med færre must have krav end tidligere. Ud fra krav opstilles tests tidligt, så vi løbende kan teste vores program.

For at sikre kvaliteten af vores diagrammer har vi planlagt at gøre som følger:

Designklassediagram før og efter implementering med attributter, metoder, multiplicitet og forklarende tekst. Package diagram for at give overblik og designklasse for at uddybe. Package diagram indeholder MVC og Test osv. Systemsekvensdiagram for en central vej igennem programmet og ligeledes for sekvensdiagrammet.

Til milestone 1 vil vi have følgende klar: Skemadiagram, use cases, krav, midlertidigt design af hjemmesiden, domæne model, flowchart over web processen, tegning af frontend, frontend hjemmeside med felter semi implementeret, design klasse diagram, systemsekvens diagram, dele af rapport.

Til milestone 2 vil vi have følgende klar: Database, DAO'er, DTO'er, frontend, JUnit begyndt, vægt, design klasse diagram, java Script.

Før afleveringsfrist: Test og finpudsning, JUnit test færdig, Konklusion.

8.2 Reelt forløb

9 Konklusion

10 Bilag