Danmarks Tekniske Universitet

CDIO Final

Diplom Software

15. juni 2018


Rasmus Gregersen


Thomas Mattsson


Patrick Madsen


Søren Rasmussen


Dan S. Drachmann


Hemen Al-Ribaty

Gruppe 22

Gruppemedlemmer:

Thomas Mattsson - s175206
Rasmus Gregersen - s175221
Patrick Marcell Madsen - s175209
Søren Kaare Rasmussen - s175202
Dan Sørensen Drachmann - s155384
Hemen Al-Ribaty - s133476

Tidsregistrering

En oversigt af vores tidsregistrering over hele projektet.

CDIO Final							
Time-regnskab	Ver. 2008-09-03						
Dato	Deltager	Design	Impl.	Test	Dok.	Andet	I alt
Hele projektet	Dan	4,5	48	8,5	5,5	3,5	70
-	Rasmus	4,5	22,5	12	14	5	58
-	Patrick	5,5	26,5	9,5	8,5	7,5	57,5
-	Thomas	2,5	34	11	5	6	58,5
-	Hemen	4	17	4	2,5	0	27,5
-	Søren	2	36,5	13	8	1	60,5
Sum		23	184,5	58	43,5	23	332

Burgerguide

Opsættelse

- Importer project som “Existing Maven Projects” i Eclipse.
- Sørg for at Tomcat er installeret.
- Sørg for at have adgang til enten den fysiske vægt eller vægt simulatoren.

Kør hjemmeside

- Inde i Eclipse kør vores project run as (run on server), kør på tomcat 8 server.
- Når Eclipse kører projektet burde vise hjemmesiden. Ellers kan denne indlæses gennem en web-browser ved at indlæse http://localhost:8080/22_CDIOFinal/index.html
- “22_CDIOFinal” skal ændres til hvad projektet hedder i Eclipse i URL’et.

Forbindelse med vægt

- Sørg for at enten den fysiske vægt eller simulatoren er tændt og har forbindelse til computeren[1].
- Kør `main` som java applikation hvis du bruger den fysiske vægt eller `SimMain`, hvis du bruger simulatoren.

Brug af system

På hjemmesiden kan du f.eks. oprette brugere. Her er en liste over allerede oprettede brugere:

BrugerID	Brugernavn	Initialer	CPR	Rolle	Status
22	hemen	he	1111111	Laborant	false
9	Jesse P	JE	010110-1001	Foreman	true
8	Donald T	DT	020220-2002	Laborant	true
7	Pablo E	PE	060660-6006	Foreman	true
6	Arnold S	AS	050550-5005	Laborant	true
5	Harry P	HP	040440-4004	Laborant	true
4	Walter W	WW	030330-3003	Admin	true
3	Luigi C	LC	090990-9009	Foreman	false
2	Antonella B	AB	080880-8008	Laborant	true
1	Angelo A	AA	070770-7007	Admin	true

Figur 1: Bruger oprettet i databasen

Ligeledes er produktbatch 1-8 og råvarebatch 1-7 allerede oprettet.
En simpel kørsel af vægten kan gøres med værdierne pb_id = 5, rb_id = 1 og netto = 5.

[1] Slå op i relevant dokumentation for at sikre dette.

Indholdsfortegnelse

Indledning

Vi har fået en opgave, der går ud på at programmere noget software, som kan forbinde til en vægt vi fik udleveret. Brugeren af vægten kan bruge vores system, til at afveje ingredienserne til et defineret farmaceutisk produkt og gemme dette i en database.
Vores software skal indhold både en frontend og backend del, i frontend delen har vi en web applikation, hvor medarbejderne i det pågældende farmaceutiske selskab kan logge ind, og alt efter deres sikkerhedsgrad se hvilket lægemiddel der skal produceres, oprette produkt, se lagerstatus osv.
Projektet er blevet udarbejdet i Eclipse, gruppen har brugt GitHub i høj grad for at uddelegere opgaver mellem gruppens medlemmer.

Problemformulering

Fra tidligere projekter har vi et program, som er i stand til at kunne kommunikere med vægten. Programmet kan sende besked frem og tilbage med vægten, og det bagvedliggende program. Dette program skulle omskrives til at passe ind i dette projekt, nogle bugs skulle løses før det kunne opfylde kravene sat op i kravspecifikation. Programmet ”afvejnings-processen” skulle også ændres til at kunne kommunikerer med webapplikationen, da alle ordrerne som skal produceres bliver oprettet der. I vores webapplikation blev vi nødt til at tage højde for sikkerhed, med henblik på en given brugers rolle. En bruger vil kun have rettigheder til det dens rolle tillader, og må ikke benytte sig af funktioner som kun er tilladt for andre brugere. En stor del af projektet kunne suppleres med dele af tidligere projekter, som vi skulle redigere lidt for at passe ind i vores ny krav. Det gjorde vores proces hurtigere og nemmere, end vi skulle lave det hele fra bunden. Vi fik også udleveret en masse DTO og DAO klasser, som vi skulle implementere og det fungeret godt med vores program.

Analyse

Projektplanlægning

Vi besluttede at det ville være nemmest at lave et lag ad gangen, så vi startede med frontenden, derefter backenden, og til sidst vores datalag. Vi valgte denne fremgangsmåde frem for en usecase ad gangen, fordi vi mente at det ville være hurtigere, da kunne være 100% fokuseret på en samling teknologier ad gangen. Efter at have fået vores frontend og vores REST API til at spille sammen med en arrayList datalag, fandt vi ud af at vores vægt applikation ikke kunne tilgå dette datalag, da det var en separat applikation. Derfor valgte vi at implementere en SQL database, så vi kunne tilgå vores data fra alle steder i vores system. Kig i bilag 1, for at se en tidsplan over vores arbejdsfordeling.

Kravspecifikation

Nedenstående ses vores funktionelle-, og ikke funktionelle kravspecifikationer som kunden har bedt os om at opfylde til systemet, ydermere har vi vores egne krav. Vi har defineret et af vores egne krav til at hjemmesiden skal være en single page application, da vi ønsker at hjemmesiden kører hurtigt, samt at vi synes det giver et bedre flow og en bedre oversigt over hjemmesiden. Desuden, så har vi også valgt at hjemmesiden skal have et simpelt design, samt at vi gerne vil have at databasen ligger i skyen.

Funktionelle krav[2]:

Kundens krav:

1. Der skal være en webinterface til farmaceut, produktionsleder og administrator.
2. Der skal være 4 roller: “Administrator”, “Farmaceut”, “Produktionsleder”,”Laborant”.
3. En brugers rolle skal kunne definere brugerens adgang og rettigheder i systemet.
4. En administrator skal kunne administrere brugere: vise-, oprette-, opdatere brugere.
5. En produktionsleder skal kunne administrere produkt-batch og råvare-batch.
6. En farmaceut skal kunne administrere produkt-batch, råvare-batch og recepter.
7. En laborant skal kunne foretage afvejninger.
8. En bruger, råvare, recept, produktbatch, råvarebatch skal kunne oprettes og administreres i systemet.
9. Tolerancen skal definere hvor meget under/over den givne mængde af råvare en recept kan tåle.
10. En ordre skal have følgende statusser: “oprettet”, “under produktion” og “afsluttet”
11. Data skal hentes og skrives på enten en SQL database eller arrayList som et transient lager.
12. Operatør id skal ikke kunne ændres.

[2] Kravene står i prioriteret rækkefølge.

13. En bruger skal kunne sættes inaktiv.

Vores krav

14. Hjemmesiden skal reagere på interaktion uden mærkbare forsinkelser[3].

Ikke funktionelle krav

Kundens krav

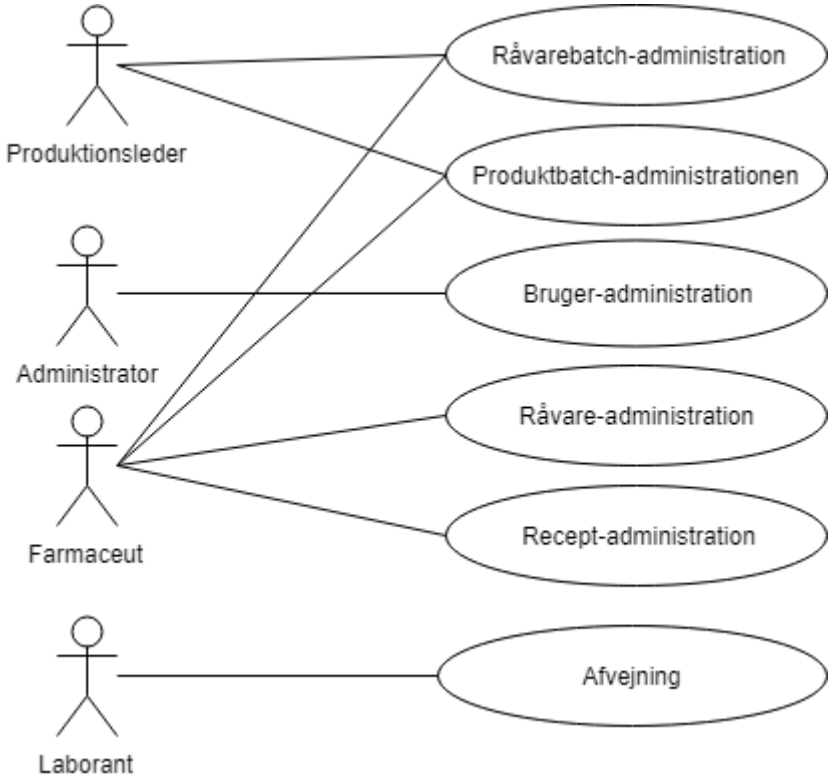
15. Programmet skal kunne understøttes af DTU's databarer.

Vores krav

16. Webinterface skal have et simpelt design.
17. SQL databasen skal køre i skyen.
18. Webinterface skal implementeres som en single page application.

[3] Mærkbare forsinkelser er alt over 300 millisekunder.

Use case diagram



Figur 2: Use case diagram med aktører og relaterede use cases.

Ovenstående diagram illustrerer vores use cases og relationen mellem aktørerne.
Produktionslederen skal kunne administrere råvare-batches og produkt-batches, hvilket inkludere at oprette og vise dem i en liste.
Administrator foretager bruger-administrationen og kan derfor oprette-, vise- og opdatere brugere.
Farmaceuten står for råvare-administration og recept-administration og kan derfor se og oprette begge ting. Samt har farmaceuten de samme rettigheder som produktionslederen.
Laborant arbejder med vægten og er med til at afveje råvare til produkt-batch.

Usecase beskrivelse

Bruger Administration
Beskrivelse: Administrerer alle bruger i systemet
Primær aktør: Administrator
Sekundær aktør: Ingen
Pre-Conditions: Server kører
Main flow: Opret bruger
Post Conditions: Bruger er opret
Alternative flow: Fjerne, retter og se bruger i systemet

Råvare administration
Beskrivelse: Administrerer råvare i systemet
Primær aktør: Farmaceut
Sekundær aktør: Ingen
Pre-Conditions: Server kører
Main flow: Opret råvare I systemet
Post Conditions: Råvare oprettet
Alternative flow: Se råvare i systemet

Recept administration
Beskrivelse: Administrerer recepter i systemet
Primær aktør: Farmaceut
Sekundær aktør: Ingen
Pre-Conditions: Server kører
Main flow: Opret recept I systemet
Post Conditions: Recept er oprettet

Alternativ flow: Se recepter I systemet
Råvarebatch administration
Beskrivelse: Administrerer råvarebatch i systemet
Primær aktør: Produktionsleder
Sekundær aktør: Ingen
Pre-Conditions: Server kører
Main flow: Opret råvarebatch I systemet
Post Conditions: Råvarebatch oprettet
Alternative flow: Se Råvarebatch i systemet

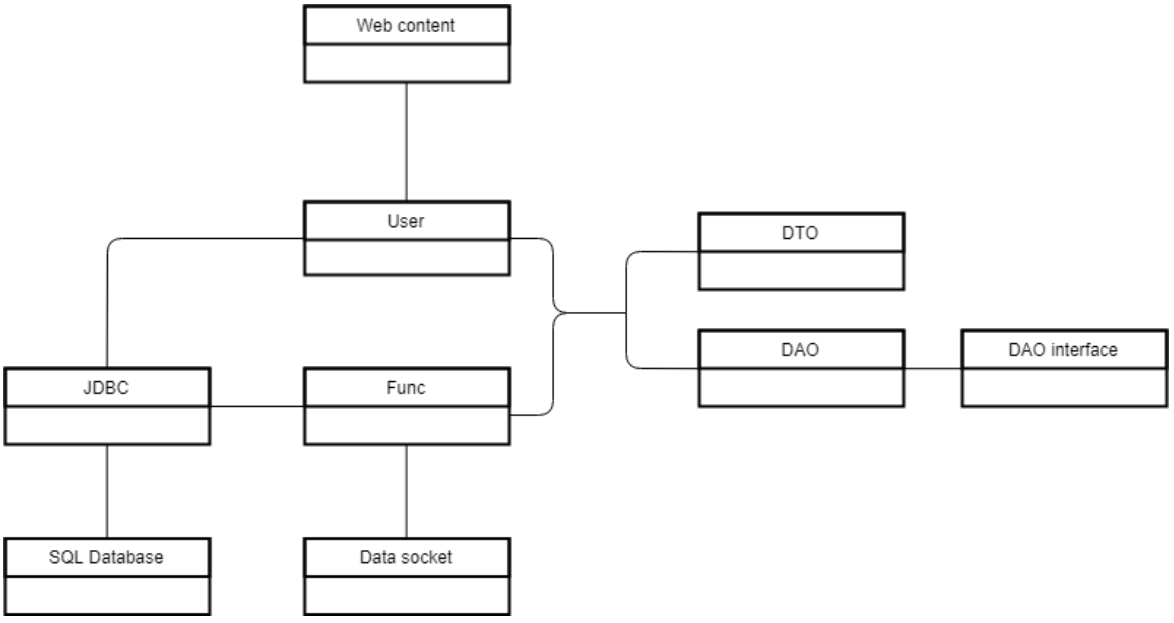
Produktbatch administration
Beskrivelse: Administrerer produktbatch i systemet
Primær aktør: Produktionsleder
Sekundær aktør: Ingen
Pre-Conditions: Server kører
Main flow: Opret produktbatch I systemet
Post Conditions: Produktbatch oprettet
Alternative flow: Se produktbatch i systemet

Afvejning
Beskrivelse: Afvejning af ordre
Primær aktør: Laborant
Sekundær aktør: Vægt
Pre-Conditions: Vægt virker
Main flow: Identifikation overfor terminalen
- Indtastning af laborant nr. på terminalen
- Terminalen svar tilbage med laborants navn

Laborant indtaster nr. på produktbatch, som skal produceres.
Terminalen beder laborant om at placer en tom beholder på vægten, beholderens vægt registreres i produktbatch komponenten. Vægten tareres, tara belastning gemmes.
Terminalen oplyser om hvilken råvare skal på vægten.
Laboranten oplyser, hvilket batch nr. den pågældende råvarebatch har.
Afvejning foretages på terminalen.
Afvejnings-resultatet registreres i produktbatchkomponenten efter recept.

Post Conditions: Ordren er afvejet
Alternative flow: Ingen

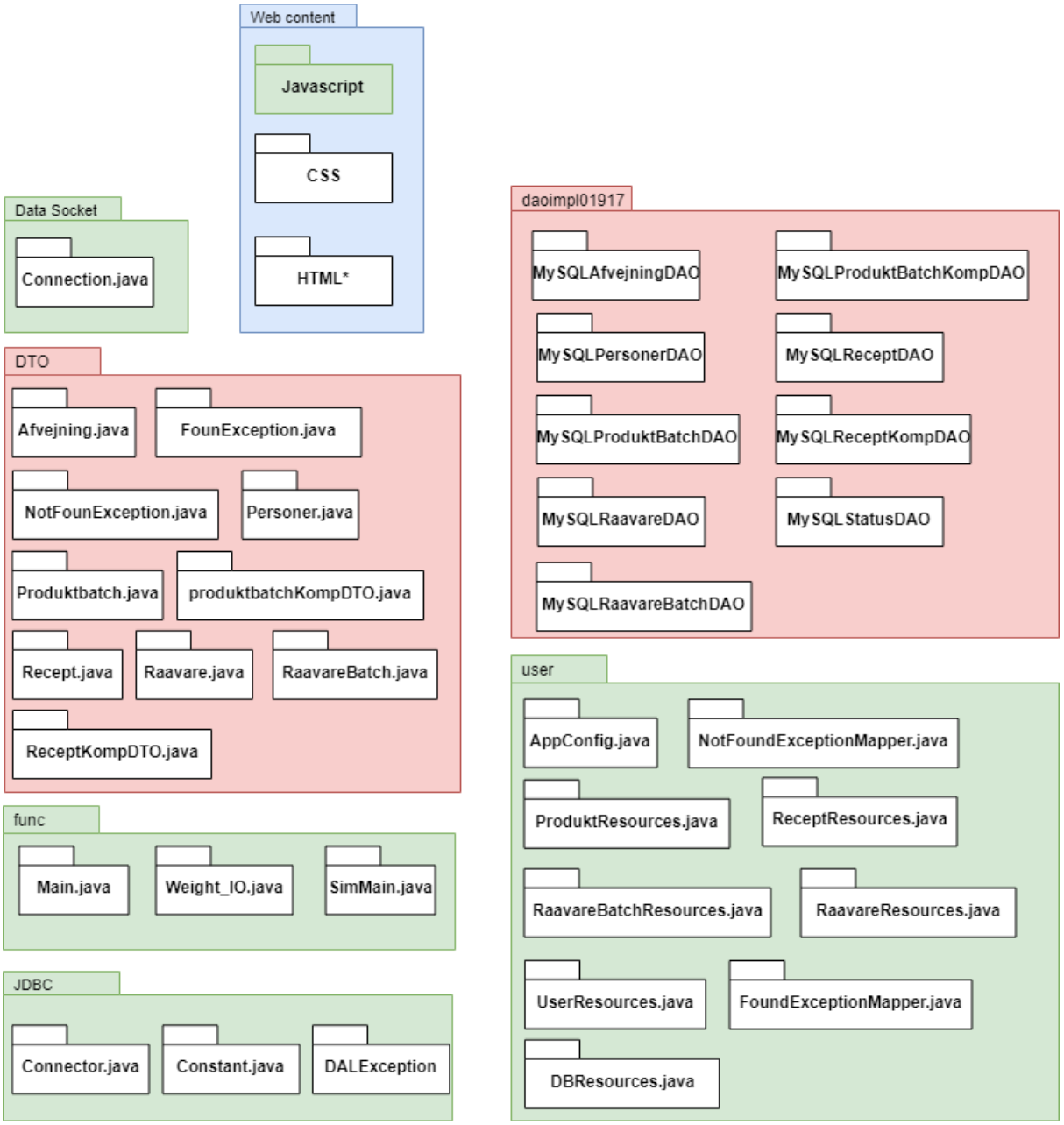
Domæne model



Figur 3: Domæne modellen for systemet.

Illustreret står domæne modellen, som viser relationerne mellem de forskellige komponenter. De forskellige komponenters indhold står i pakke diagrammet (afsnittet nedenfor).

Pakke diagram



Figur 4: Pakke Diagram af systemets pakkestruktur farvekoordineret til illustrering af tre-lags-modellen.

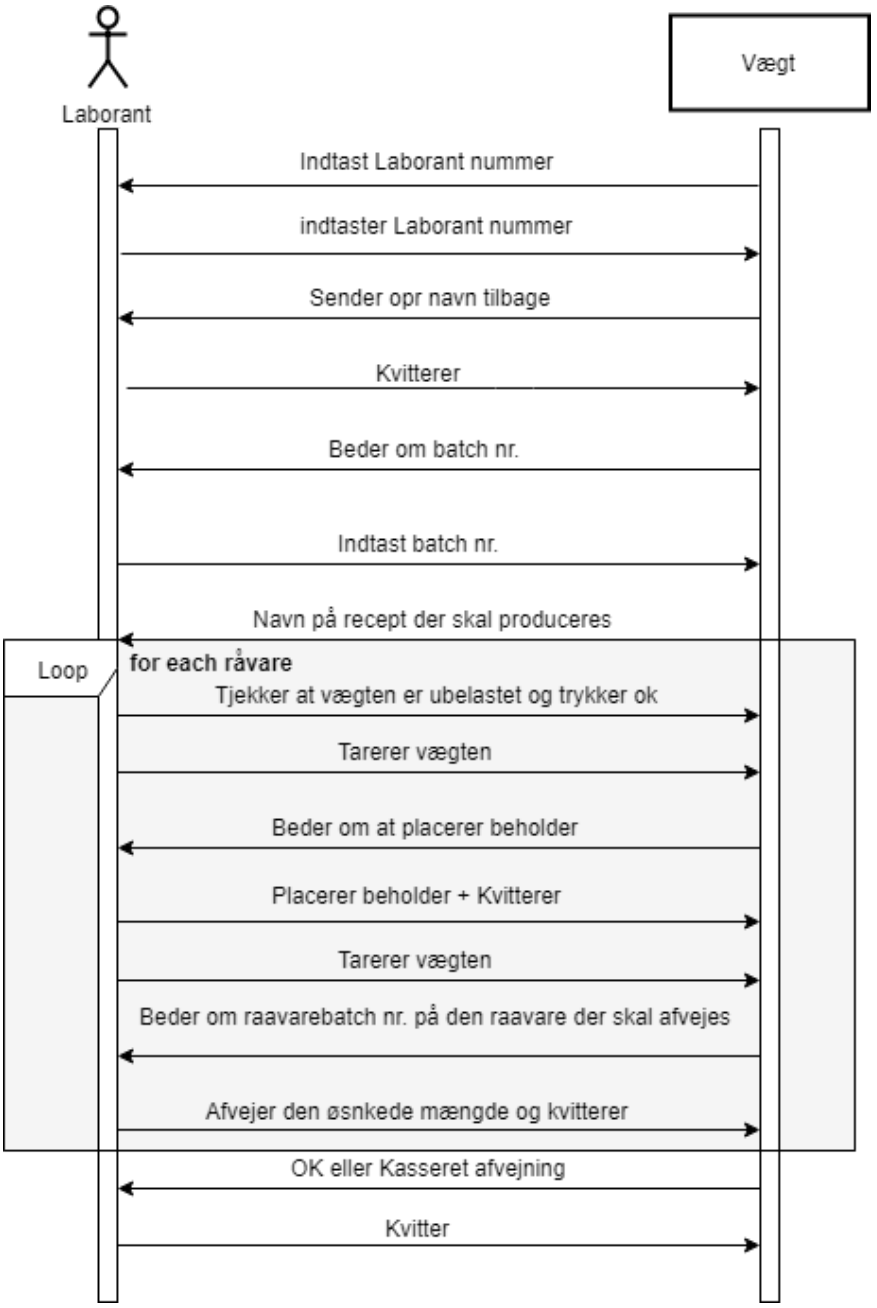
Ovenstående billede viser arkitekturen af afvejnings systemet. Vi har ikke inkluderet interfaces og web content ind i pakke diagrammet. Inde i “user” pakken ligger vores RESTful application. “func” og “data socket” bruges til vægten (simulator). “JDBC” er vores forbindelse til vores MYSQL database, mens “DTO” og “DAOImpl” er til at referere til de nødvendige data som vi benytter os af.
Vi benytter os af tre lags arkitektur som illustreret med farver på billedet:

- Blå: Dette er præsentationslaget, hvor data og grafik bliver præsenteret til brugeren.

- Grøn: Dette er funktionslaget, som sørger for information og logik mellem klienterne og databaselaget.
- Rød: Til sidst har vi datalaget, som håndterer sig af al data i systemet. Endvidere har vi en MySQL server i skyen, som lagrer vores data.

De små hvide filer tilhører den samme farve som deres tilhørende pakke farve.

System sekvens diagram

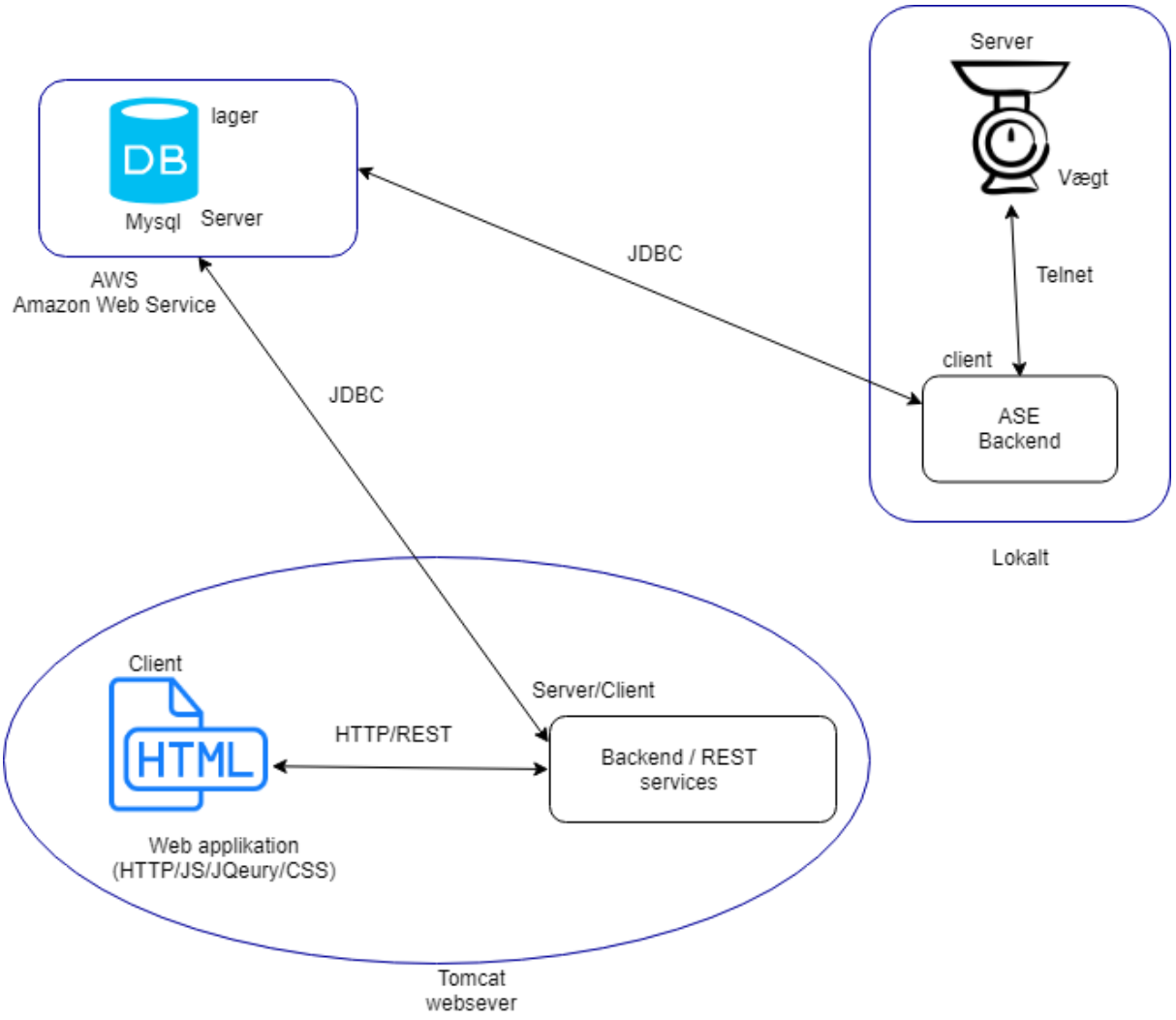


Figur 5: Systemsekvensdiagram over afvejnings use case.

Ovenstående billede illustrerer system sekvensdiagrammet, som beskriver flowet mellem en laborant og vægten for afvejnings processen når en et produktbatch produceres.

Design

Her ses et deployment diagram. Som det kan ses har vi 2 dele af vores backend, der snakker med henholdsvis vægten og webapplikationen. Begge dele af vores backend snakker så også med en fælles SQL server.



Figur 6: Deployment diagram

GRASP:

Controllers:
Vi betragter en kombination mellem vores javascript ajax kald og vores rest services som vores controllers da de videresender information mellem frontend og databasen.
High cohesion:
Vi har sørget for at opdele de forskellige elementer af koden op i hver deres mappe og kun kode som er relateret til den klasse er implementeret deri.
Low coupling:
I vores kode er har vi forsøgt at sørge for at der er “low coupling” i systemet
Et sted hvor vi kan få et problem, er hvis man ændrer et “class name” et sted vil det også påvirke vores javascript filer. Dette skete også hvis vi ændrede “Package name”.
Men der har været fokus på at klasserne kan genbruges i en vis omfang.

Patterns:

Under udvikling har der ikke været fokus på andre eksterne patterns end GRASP. Vi har haft fokus på at sørge for at så mange klasser som overhovedet muligt afstammer fra et interface men projektet har ikke vist nødvendighed for polymorfy eller andre generelle patterns.

Valg af design: Hjemmeside

Hjemmesiden er bygget op efter et simpelt og intuitivt bruger design hvor der ikke kræves login da dette er kundens ønske.
Der har været fokus på at knapperne skulle være selvforklarende for det firma som vil bruge systemet.
De individuelle sider er designet til kun at vise det absolut mest nødvendige i følge med kundens krav. dvs, at man ikke har mulighed for små ekstra unødvendige features.
siderne er bygget op ved at indkapsle de forskellige hjemmeside elementer så man fremadrettet nemt kan modificere designet eller evt. tilføje flere elementer så længe man kender sit CSS.
Ved opretning af recepter, har vi valgt at implementere det dynamisk, så man kan tilføje *n* komponenter til en recept som ønsket af kunden.
I henhold til baggrund har vi valgt at hjemmesiden skal afspejle en blid repræsentation af medicin, da det skal stemme overens med det simple layout.
Til sidst har vi valgt at man kun viser tilknyttede komponenter til recepter eller produktbatches respektivt når det ønskes.
En oversigt over hjemmesidens struktur kan ses i bilag 2.

Design overvejelser: Fremtid

Fremtidsorienteret kan hjemmesiden implementeres med CCS grids da dette rygtes at være mere dynamisk og begyndende at blive mere udbredt.
Derudover kan designet forbedres og baggrunden kan blive bedre implementeret.
Ved aflevering af dette projekt er baggrunden stadig en samlet svg fil men der var overvejelser om at splitte den i to, dog ikke nået. Desuden skal teksten separeres bedre fra baggrunden da den kan være svær at læse.

Implementering

Teknologier

En kort gennemgang af de teknologier vi har gjort brug af og hvorfor.

Web applikation

Vores webapplikation er implementeret som en SPA (single page application), hvilket betyder at vores system består kun af en side. Indholdet bliver loadet dynamisk ind, dvs. der ikke bliver åbnet ny url side hver gang. Dette minimerer load tiden, og det har stor betydning for hastigheden i systemet, det bliver generelt hurtigere.

Javascript/jQuery

Med hensyn til vores JavaScript har vi brugt jQuery biblioteket, hvorved hensigten har været at få gjort mere med mindre kode men også simplificere vores rest services kald. da AJAX er et asynkront kald tillader vi os at sende/hente data til/fra serveren uden at skulle fryse alt andet aktivitet på samme tid.

Backend

Backenden til hjemmesiden og vægten er implementeret i Java og står for business logic i projektet. Det er også i vores backend at vores REST API ligger.

REST API

Vi har lavet en backend som kan bruges af vores webapplikation. Backenden er opbygget som et stateless Rest Api, der holdes altså ikke styr på sessioner, men returneres, oprettes eller ændres det data som du har bedt om.
Systemet bruger JSON format både som input og output, hvilket vi bruger grundets dets simplicitet og det at det er let at læse og forstå.

Lager

Vi startede med bruge arrayList som datalager, men vi skiftede hurtigt til en lokal SQL server, for at alle dele af systemet kunne tilgå dataen. Vi lavede små modifikationer til den database, som vi udviklede i kurset 02327. Mod slutningen af projektet blev vi tilbudt at få vores SQL server op i skyen på Amazon Web Services af hjælpelærerne. Dette valgte vi at gøre, da det er nemmere at administrerer 1 server i skyen end at holde styr på 6 lokale servere fordelt på vores computere. Vi valgte det også fordi det er et mere realistisk setup i forhold til implementation og iht. et rigtigt produkt til en kunde.

Exceptions

Exceptions er implementeret i to forskellige omfange beskrevet herunder.
Vi har selv defineret exceptions med brug af exceptions mappers og så har vi implementeret exceptions baseret på outputs fra SQL statements. exception bliver sendt til Ajax kaldet og smidt op på hjemmesiden som en alert.
Vi har ikke haft mulighed for at implementere ordentlig exception handling iht. mysql UPDATE statements inde i stored procedures grundet manglende viden. Fokus har været på at exception handling er fokuseret primært i MySQL klasserne hvorved de returnere en string tilbage til start kaldet. Problemet med update statements i stored procedure kunne løses ved først at eksekvere en select statement som fortæller os hvorvidt vi vil update et element hvis det eksisterer.
Derudover har vi endvidere forsøgt at fange så mange exceptions som muligt ved at bruge specifikke mysql error codes I stedet for bredere sql exception. I bilaget kan der findes en oversigt over vores exceptions og hvad årsagen er til at de er blevet triggered.

Vægt Flow

For at snakke med vægten har vi udviklet et flow i controller-klassen `WeightIO`, der sørger for at sende de rigtige oplysninger til vægten, i form af kommandoer som vægten kan forstå. Dertil sørger `WeightIO` klassen for at “oversætte” de svar som vægten kommer med til data, der kan sendes til vores DTO’er og til databasen via vores DAO’er.
Flowet er designet med de former for input validering, som vi besluttede var vigtigst. Først og fremmest sikrer vi, at man ikke kan indtaste et ugyldigt laborant-, produktbatch- eller råvarebatch-nummer. Disse værdier bliver tjekket imod databasen, hver gang man indtaster dem. En anden vigtig validerings-feature er at man ikke kan veje den samme ting to gange i samme recept, da alle råvare man afvejer, bliver tjekket imod recepten i databasen.
I forhold til brugervenlighed har vi også gjort at vægten viser hjælpsom information i displayet, såsom hvilken råvare brugeren skal afveje. Vi har også implementeret en “vil du fortsætte?” funktion, som giver brugeren mulighed for at afveje det næste produktbatch med det samme, efter det forrige er færdigt.
Vi er tilfredse med vores implementering ift. kommunikation mellem vægten og databasen samt vores flow. Vi opfylder alle de krav, der står beskrevet i afsnit 2.6 i opgavebeskrivelsen.

Dertil har vi lavet ekstra funktionalitet, som f.eks. “vil du fortsætte?”-funktionen. Havde vi haft mere tid ville bedre fejlhåndtering og evt. en mulighed for at logge ud i stedet for at man skal slukke vægten eller vores program.

Input validering

Fra vores hjemmeside laver vi primært input validering i form af html input begrænsninger. For eksempel at man kun kan indtaste heltal større end 0, eller antal tegn i initialer. Denne form for input validering fungerer mere som hjælp til folk med gode hensigter, den forhindrer ikke angreb. Men selv hvis man indtastede, noget man ikke skulle kunne, så burde vores system smide nogle exceptions på grund af type mismatch. Ud over det så er vores database beskyttet mod SQL-injections, ved at gøre brug af prepared statements. Så selv hvis en hacker ikke blev mødt af en exception, så kan han stadig ikke få adgang til databasen.
Med hensyn til vægten så kan brugeren kun indtaste positive hele tal fra vægten og vi håndtere inputtet ved at tjekke i databasen om værdien findes, hvis ikke så gentag processen. Dog hvis et problem ville forekomme under vægt processen, så vil en exception håndtere det.

Test

Vi har importeret vores projekt på en databar, hvor vi fik bekræftet at maven hentede de ting som den skulle. Den hentede en tomcat server, så vi kunne få kørt vores frontend, samt vores backend Rest API. Når vi starter serveren opretter den forbindelse til vores SQL server, så vi kan se dataen. Hvis vi kører vægt simulatoren, og derefter kører vores `SimMain` som en java applikation, kan simulatoren igennem vores backend også tilgå SQL serveren.

Vi har også testet vores kode på den rigtige vægt, og der virkede alt som forventet, dog var det ikke muligt at se hvor meget man afvejer. Vores kode er udviklet til simulatoren, og der var der ikke noget problem med at se/vælge hvor meget man ville afveje. Men på den fysiske vægt viser skærmen bare at man skal afveje sin nettovægt, den viser ikke den aktuelle vægt man har lagt op på vægten. Vi er ikke helt sikre på hvordan vi skal løse dette problem, da det har været svært at få adgang til en fysisk vægt, så vi kan teste forskellige løsninger. Et løsningsforslag vil være at prøve sig frem med nogle kommandoer fra dokumentationen til vægten. For eksempel så skulle kommandoen “DW” vist nok returnere til vægt visning, men vi havde ikke mulighed for at teste denne eller andre eventuelle kommandoer for at finde en løsning.
Alternativt kunne man køre et loop, der viser den aktuelle vægt hvert sekund indtil der trykkes OK. På den måde vil laboranten kunne se hvor meget han/hun har afvejet indtil videre.

Black box test

Som en del af vores tests har vi udført en række black box tests. Disse test har vi valgt, da de er hurtige at lave og dokumentere. Vi har testet funktionalitet relateret til vores use-cases, for at sikre at de virker. Nedenfor vises et eksempel på en black box test. Resten af vores test findes i bilag 3.
Farver:
Grøn - Lykkedes
Gul - Til dels lykkedes (accepteret)
Rød - Fejlet

Test Case ID (Kørsel)	Resumé	Aktuelt output	Forventet output	Status
#BB01 (16-06-2018)	Bruger oprettes når form submittes	Oprettes i databasen	Bruger oprettes i databasen	

Figur 7: Eksempel på en af de udførte black box tests.

Konklusion

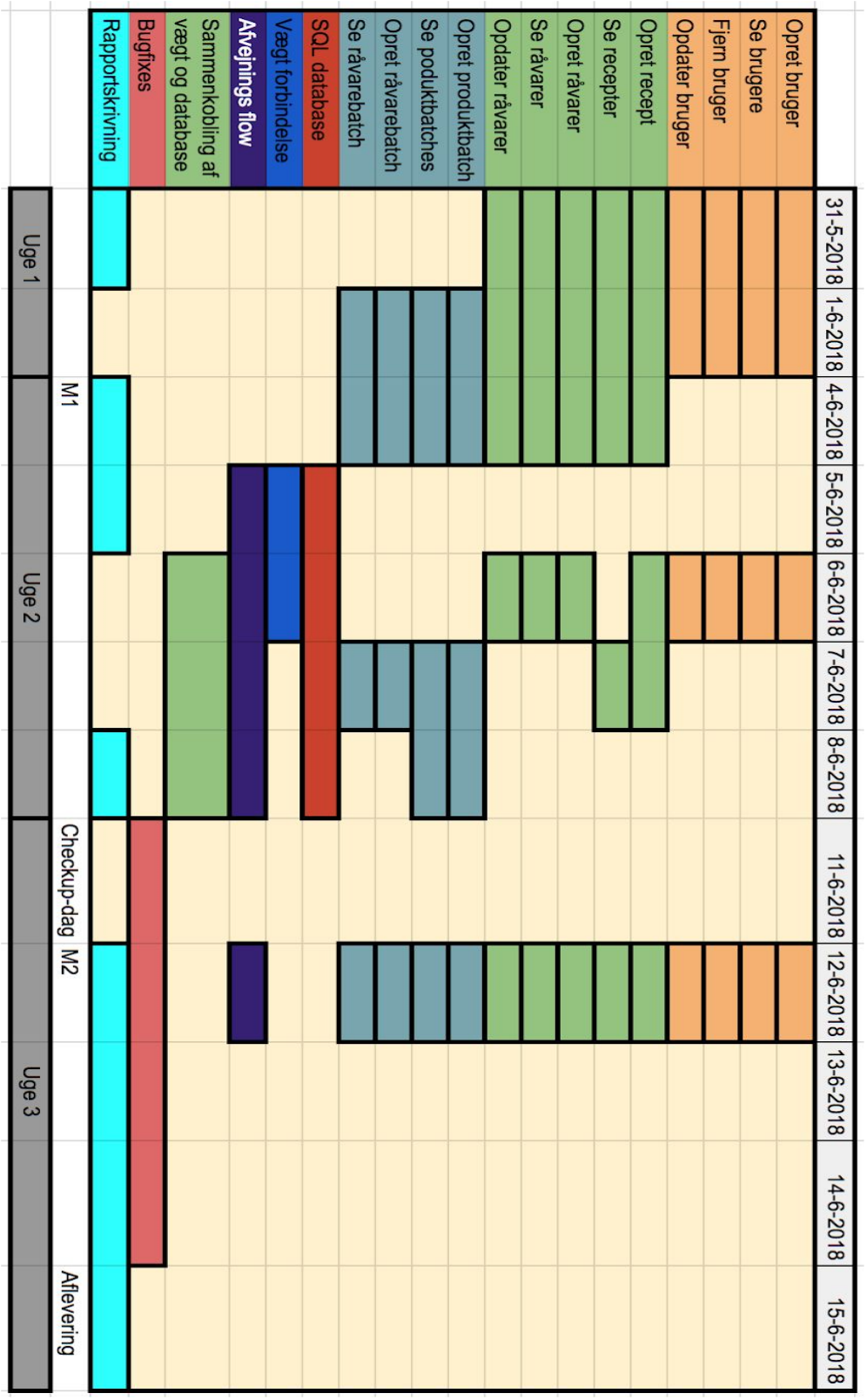
Vi kan konkludere at vi har fået lavet et produkt som opfylder kundens krav til systemet, men at der er flere steder, hvor produktet ville kunne blive forbedret, hvis der var allokeret mere tid til det.

Vi har forsøgt at følge den tidsplan, som vi havde sat op i starten af projektet, men der kom komplikationer under udførelsen af nogle af punkterne, hvilket ledte til at nogle punkter tog længere tid end andre. Udskiftningen fra arraylist til MYSQL database har specielt haft flere komplikationer end forventet under planlægningen.

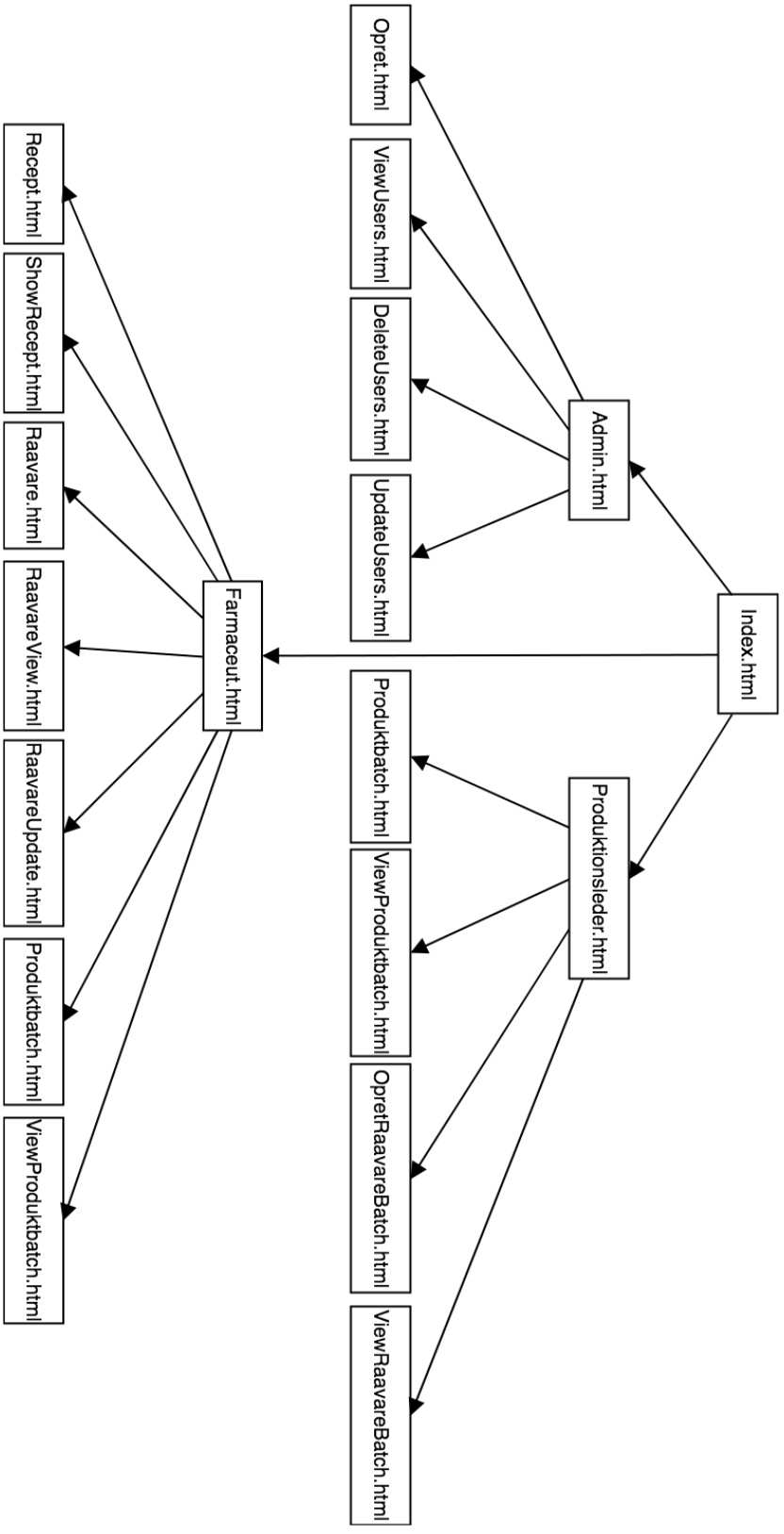
Vi oplevede også komplikationer ved at vi havde kommet til at bruge forkerte/ugunstige navne til nogle af vores pakker samt en af vores klasser. Da vi prøvede at løse det så kom der problemer der gjorde at vi ikke kunne ændre navnene. Denne oplevelse kan nu bruges til i fremtiden, at sikre at alle pakke navne og klasse navne er helt korrekt angivet i starten af et projekt, siden det forårsager komplikationer at ændre dem senere.

Bilag

Bilag 1: Projektplanlægning



Bilag 2: Hjemmesidestruktur



Bilag 3: Black Box Test

Grøn - Lykkedes
Gul - Til dels lykkedes (accepteret)
Rød - Fejlet

Test Case ID (Kørsel)	Resumé	Aktuelt output	Forventet output	Status
#BB01 (13-06-2018)	Bruger oprettes når form submittes	Oprettes i databasen	Bruger oprettes i databasen	
#BB02 (13-06-2018)	Bruger oprettes form nulstilles når form submittes	Den nulstiller	Form nulstilles	
#BB03 (13-06-2018)	Liste af brugere opdateres med det samme når bruger opdates	Opdateres	Liste genindlæses med ændringer	
#BB04 (13-06-2018)	Bruger kan sættes til inaktiv i databasen	Status ændres til false	Bruger status opdateres i databasen	
#BB05 (13-06-2018)	Recept oprettes med råvare	Oprettes i databasen	Råvare er tilknyttet på recept oversigt	
#BB06 (13-06-2018)	indtast id i recept oversigt og få de tilknyttede råvare	Den henter relateret råvarer	Relateret råvare hentet og vist på hjemmeside	
#BB07 (13-06-2018)	Opret råvare i databasen via submit form	Lykkedes	Råvare oprettet i databasen og vist på hjemmesiden	
#BB08 (13-06-2018)	Opdater råvare og få liste opdateret	Ikke fungerende, opdaterer råvare men ikke liste direkte	Listen opdateres	
#BB09 (14-06-2018)	Opret et råvare i recept som ikke eksisterer i råvarelisten	Receptet bliver oprettet og den kaster	Recepten bliver ikke oprettet.	

		forkert exception		
#BB10 (13-06-2018)	Opret produktbatches	Lykkedes	Produktbatch oprettet uden tilknyttede komponenter	
#BB11 (13-06-2018)	Opret råvareBatchens	Lykkedes	Oprettet i databasen	
#BB12 (14-06-2018)	Opret råvarebatchens med råvare id som ikke eksisterer	ikke muligt	Ikke muligt at oprette råvarebatches	
#BB13 (14-06-2018)	Opret produktbatch med recept id som ikke eksisterer	opretter ikke	Kan ikke oprette produktbatch id	
#BB14 (14-06-2018)	Oplys ingen tilknyttede komponenter hvis et produktbatch ikke har nogle	oplyser information som forventet	Lykkes	
#BB15 (13-06-2018)	Bruger opdateres ikke hvis forkert kombination af cpr og id.	Forventet	Bruger ændres ikke	
#BB16 (14-06-2018)	Recept oprettes uden ravarre	ikke muligt, får besked	Få en alert som stater det ikke kan lade sig gøre	

Bilag 4: Error codes og deres betydning

Exception handling:

- Create user
    - Error code Ux01
        - Mysql stores procedure returnerer et nummer som indikere hvorvidt et bruger id allerede eksisterer.
    - Error code Ux02
        - Samme koncept som ovenstående, returnere dog et tal som indikere en anden ikke specifikt defineret SQL fejl
- Deaktiver user
    - Error code Ux03
        - Sker hvis bruger id ikke eksisterer i databasen
- Updater bruger
    - Error code Ux04
        - Kan ikke udløses i øjeblikket
            - Burde håndtere ikke eksisterende bruger. Dette grundet problem med exception handling i stored procedure relateret til SQL update

- Create recept
    - Error code Rx01
        - Sker hvis bruger forsøger at oprette en recept hvor råvare = 0
    - Error code Rx02
        - Sker hvis bruger forsøger at lave en Recept som allerede eksisterer.
    - Error code Rx03
        - Sker hvis SQL fejl urelateret til ovenstående fejl
- View receptKomponenter
    - Error code Rx04
        - Udløst af en tom liste relateret til et id
- Create Raavare
    - Error code Rax01
        - Sker hvis råvare allerede eksisterer.
    - Error code Rax02
        - Sker for urelateret SQL fejl, kontakt nærmeste tekniske afdeling
- View Raavare
    - Error code Rax03
        - Sker grundet database fell eller at projektet er ved at opdatere sig selv (burde ikke ske under uptime så længe der ikke sidder noget og rodder i koden direkte)
- Update Raavare
    - Error code Rax04
        - Sql fejl ikke relateret til eksisterende raavare id - kontakt nærmeste tekniske afdeling
    - Error code Rax05
        - Sker ved manglende raavare i systemet

        -
- Create produktbatch
    - Error code Pbx01
        - MySQL error code 1062
            - Duplicate entry key
    - Error code Pbx02
        - MySQL error code 1452
            - foreign key constraint fails
    - Error code Pbx03
        - Sql error ikke relateret til code 1062 eller 1452 (check console)
- View produktbatchKomponenter
    - Error code Pbx07
        - Udløst af Tom liste relateret til produktbatch id