Danmarks Tekniske Universitet

3-ugers projekt

02324 - Videregående programmering

Gruppe 11

25-06-2020

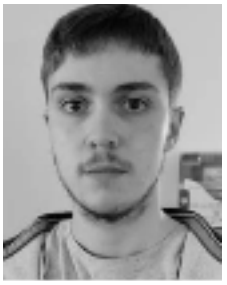
Mikkel Danielsen
s183913

Frederik Koefoed
s195463

Muhammad Talha
s195475

Volkan Isik
s180103

Lasse Strunge
s195486



Mark Mortensen
s174881

Github: https://github.com/Kameldrengene/cdioFinal_f2020

Indholdsfortegnelse

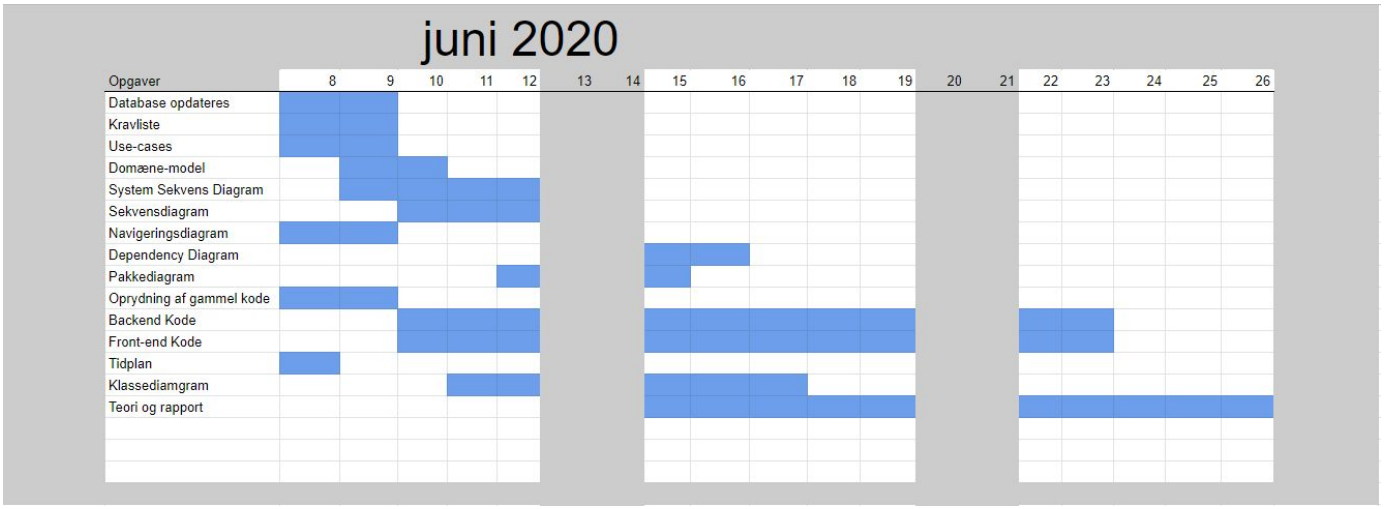
Indledning

.....
.....

Noter

- Vi har brugt onclick i buttons til at aktivere js i stedet for at via js at target de enkelte elemtenter. På den måde kan kode også nemt genbruges
- .getRole(String role)
- Default.css (brugertabeller)
- Vores brug af async..
- Hvorfor “rediger” -> “gem”
- ProduktbatchDTO indeholder kun en linje, så det er nemmere at opdatere.. skulle være kaldt produktbatchkomponent ifølge opgaven
- createUser retunerer user så det er nemmere at teste med postman

Tidsplan



Senest opdateret 10/06-2020

Kravliste[1]

Systemkrav

K?? M - Der skal være et web interface (GUI)
K?? M - Dataen skal gemmes i en database
K?? S - Alle input til GUI skal valideres og hvis der gives et ugyldigt input skal systemet skrive at der er en fejl
K?? C - En bruger må aldrig slettes fra databasen, men kun “inaktiveres”

Bruger administration

K3 M - Det skal være muligt for en administrator at oprette en bruger
K4 M - Når man opretter en bruger skal man kunne angive deres navn, initialer, cpr-nummer samt deres roller
K9 M - Der skal være 4 roller (Admin, Pharmacist, Foreman, Operator)
K10 M - Brugernavnet skal være mellem 2 og 20 karakterer
K11 M - Initialerne skal være mellem 2 og 4 tegn
K12 M - Passwordet skal indeholde mellem 6 og 50 tegn
K6 M - Brugerens ID skal være mellem 11 og 99
K16 M - Når en ny bruger oprettes skal de have det laveste ledige ID
K5 S - Man skal kunne se data for alle oprettede brugere på en samlet liste
K7 S - Man skal kunne ændre i brugerens oplysninger (Navn, initialer, cpr-nummer, password)
K15 C - Når en ny bruger oprettes skal der genereres et password til dem

Råvare administration

K?? M - *farmaceuten* skal kunne oprette råvarer
K?? M - Råvarer skal oprettes med id, navn og leverandører.
K?? M - En *råvare* defineres ved et råvareNr (**brugervalgt** og entydigt) og navn. (d.v.s. råvareNr skal IKKE autogenereres.)
K18 M - Systemet kan kunne definere *Råvare*: Kommer fra forskellige leverandører i *RåvareBatches* med given mængde af råvarer.

[1] Bilag 1

K?? S - Farmaceuten skal kunne se råvarer i en samlet liste
K?? C - Farmaceuten skal kunne opdatere råvarer i den samlede liste

Recept-administration

K?? M - En recept skal have et receptnummer og en eller flere receptkomponenter
K?? M - En receptkomponent består af en råvare, en mængde og en tolerance
K?? M - En farmaceut skal kunne oprette en recept, hvori der bestemmes hvilke råvare der skal bruges samt mængden og tolerancen af disse.
K?? S - Råvaren skal findes i systemet for at være en del af en recept

Råvarebatch-administration

K?? M - Produktionslederen skal kunne oprette et råvarebatch
K?? M - Et råvarebatch skal have et råvarebatchnummer, en mængde og en leverandør
K?? S - Produktionslederen skal kunne vises alle råvarebatches
K?? C - Mængden i råvarebatchet skal automatisk opdateres når der bliver brugt noget af en råvare i et produktionsbatch

Produktbatch-administration

K?? M - Produktionslederen skal kunne oprette produktbatch
k?? M - Produktbatch skal oprettes med id, receptid, status, userid, råvarebatch id, tara i kilogram og netto vægt i kilogram
k?? M - Afvejningsresultater for de enkelte *receptkomponenter* gemmes som en *produktbatchkomponent* i *produktbatchen* i takt med at produktet fremstilles.
k?? S - Produktsionslederen skal kunne opdatere en produktbatch
k?? C - Produktionslederen skal kunne printe produktbatch i et dokument.

Afvejning

K?? M - Systemet skal fortælle laboranten hvilken ingrediens, samt mængde og tolerancen før han begynder at veje
K?? M - Laboranten skal kunne skrive vægt ind på websiden
K?? M - Hvis vægten ikke er tæt nok på den vægt der er givet i recepten, skal websiden melde fejl og laboranten må veje igen
K?? C - Laboranten skal identificere sig selv før de kan bruge systemet

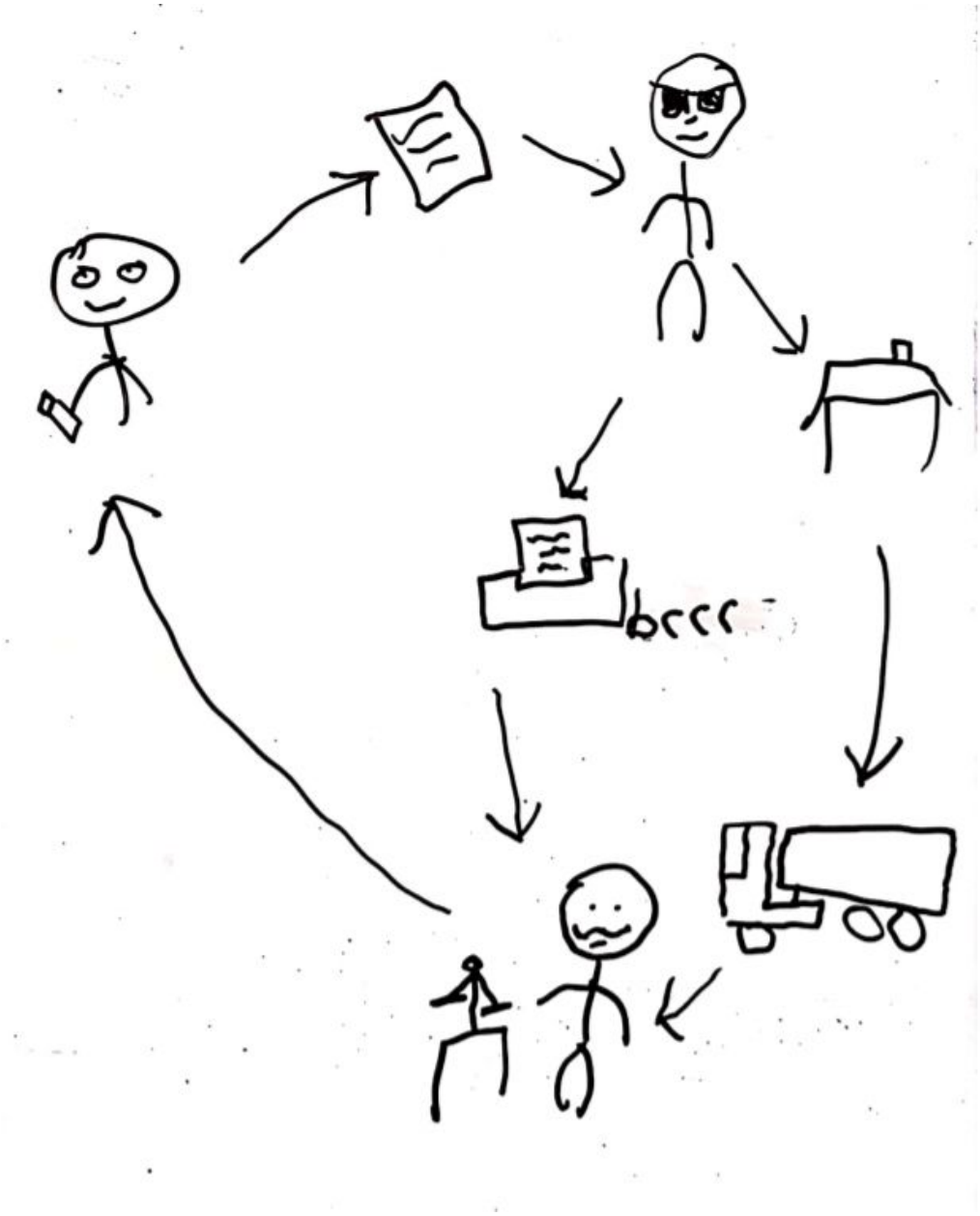
K?? M - Når laboranten indtaster produktbatch-id'et skal sytemet kunne finde det pågældende produktbatch og laboranten redigerer herefter i netop det produktbatch
K?? S - Laboranten skal kunne bestemme produktbatchets tara.
K?? M - Før afvejningen startes skal produktbatchets status være ikke påbegyndt
K?? M - Når den første afvejning er sket skal produktbatchets status gå fra ikke påbegyndt til under produktion
K?? M - Når alle receptkomponenter er afvejet skal produktbatches status gå fra under produktion til afsluttet

Ikke funktionelle krav

- Datalaget i applikationen kan implementeres i MysSql database.
- Der skal laves et data-acces laget, skal der kastes en exception, der beskriver fejlen til øvre lag.

Rigt billede

Skal nok i bilag på et tidspunkt, men for nu ligger den her så vi alle kan nyde kunst.



En farmaceut (øverst til venstre) laver og sender en recept. Produktionslederen (ham med solbriller) printer et produktbatch og afleverer det til laboranten (ham med skægget). Produktionslederen sørger også for at bestille nogle råvarebatches fra nogle leverandører som laboranten kan bruge til hans afvejninger. Når laboranten har lavet alle afvejningerne sendes de til farmaceuten.

Analyse

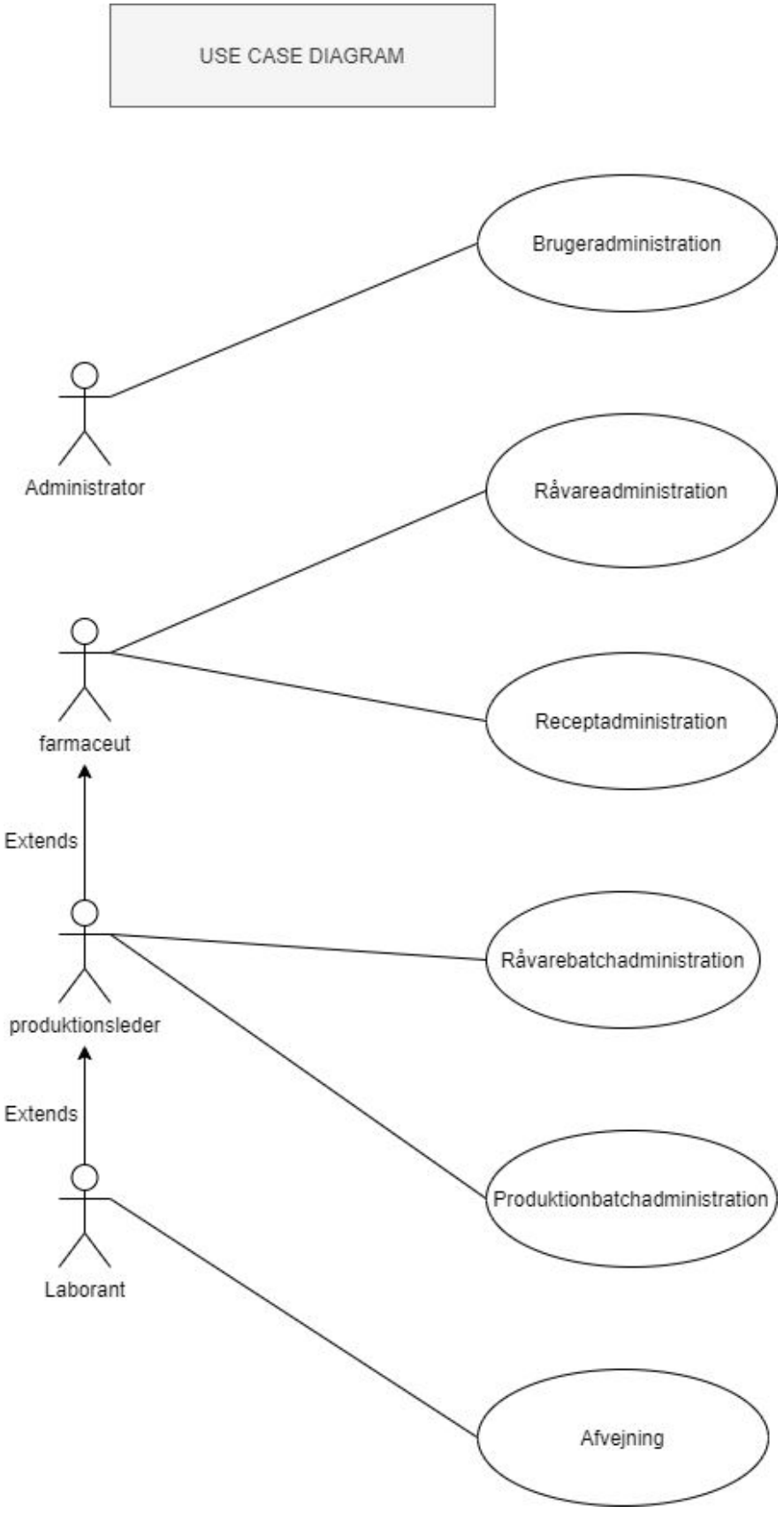
Aktørliste

Aktør	Formål
Admin	Admin skal kunne udføre fuld brugerhåndtering:

	● Oprette en bruger ● Deaktiver en bruger ● Se brugernes oplysninger samt rette i dem
Farmaceut	Farmaceut skal kunne udføre fuld råvare- og recepthåndtering ● Oprette en råvare ● Opdatere en råvare ● Oprette en recept ● Opdatere en recept
Produktionsleder	Produktionsleder skal kunne udføre fuld råvarebatch- og produktionbatchhåndtering ● Oprette en råvarebatch ● Opdatere en råvarebatch ● Oprette en produktionbatc ● Opdatere en produktionbatc ● Printe en produktionbatc
Laborant	Laborant skal kunne udføre afvejning af en produktbatch: ● Afvejning af råvarer i et produktbatch ● Opdaterer status i et produktbatch
Tid	Siger hej, hver tredje sekund til vores database!!!!

Use cases

- U01 Administrere bruger
- U02 Administrere råvarer
- U03 Administrere recept
- U04 Administrere produktbatch
- U05 Administrere råvarebatch
- U06 Afvejning


USE CASE DIAGRAM
Brugeradministration
Administrator
Råvareadministration
farmaceut
Receptadministration
Extends
Råvarebatchadministration
produktionsleder
Extends
Produktionbatchadministration
Laborant
Afvejning

Aktør Tabel

Aktører:	U01	U02	U03	U04	U05	U06
Admin	x					
Farmaceut		x	x	x	x	x
Projektleder				x	x	x
Laborant						x
Tid ?						

Use case beskrivelse

Fully dressed

Use Case Section	Comment
Navn	Opret bruger
Scope	CDIO final
Level	User-goal
Primær aktør	Bruger
Preconditions:	Hjemmesiden er startet og brugeren er logget ind.
Postconditions:	En ny bruger er oprettet
Hoved succes scenarie:	1) Brugeren trykker på knappen ''Opret Ny Bruger'' 2) Navn bliver indtastet. 3) Initial bliver indtastet. 4) CPR-nummer bliver indtastet. 5) Password bliver indtastet. 6) En rolle eller flere roller vælges fra menuen. 7) Brugeren trykker på knappen ''Opret''. 8) Brugeren bekræfter oprettelse ved at trykke på ''ok''. 9) Brugeren bliver oprettet.
Alternative scenarier:	2-6: Brugeren ikke indtaster på nogle af de felter, 7: Brugeren trykker på knappen ''Opret''. 8: Brugeren får fejlmeddelelse at den skal udfylde de tomme felter og bruger trykker på ''ok'', og udfylder de tomme felter. 7: Brugeren trykker på knappen ''Opret''. 8a: Brugeren bekræfter oprettelse ved at trykke på ''ok''. 8b: Brugeren trykker på ''cancel'', der sker ikke noget man bliver på samme side. 2-6: Brugeren indtaster alle oplysninger 7: Brugeren trykker på knappen ''Opret''. 8a: Brugeren trykker på ''cancel'' og der sker ikke noget man bliver på samme side. 8b: Brugeren trykker på ''ok'' og brugeren bliver oprettet.

Use Case Section	Comment
Navn	Vis brugere
Scope	CDIO 2

Level	User-goal
Primær aktør	Bruger
Preconditions:	Hjemmesiden er startet
Postconditions:	Liste af brugere bliver vist
Hoved succes scenarie:	1) Brugeren indtaster brugernavn og password. 2) Brugeren trykker på ''login'' 3) Brugeren er logget ind. 4) Brugeren kan nu se alle de oprettet bruger i systemet.

Fully dressed

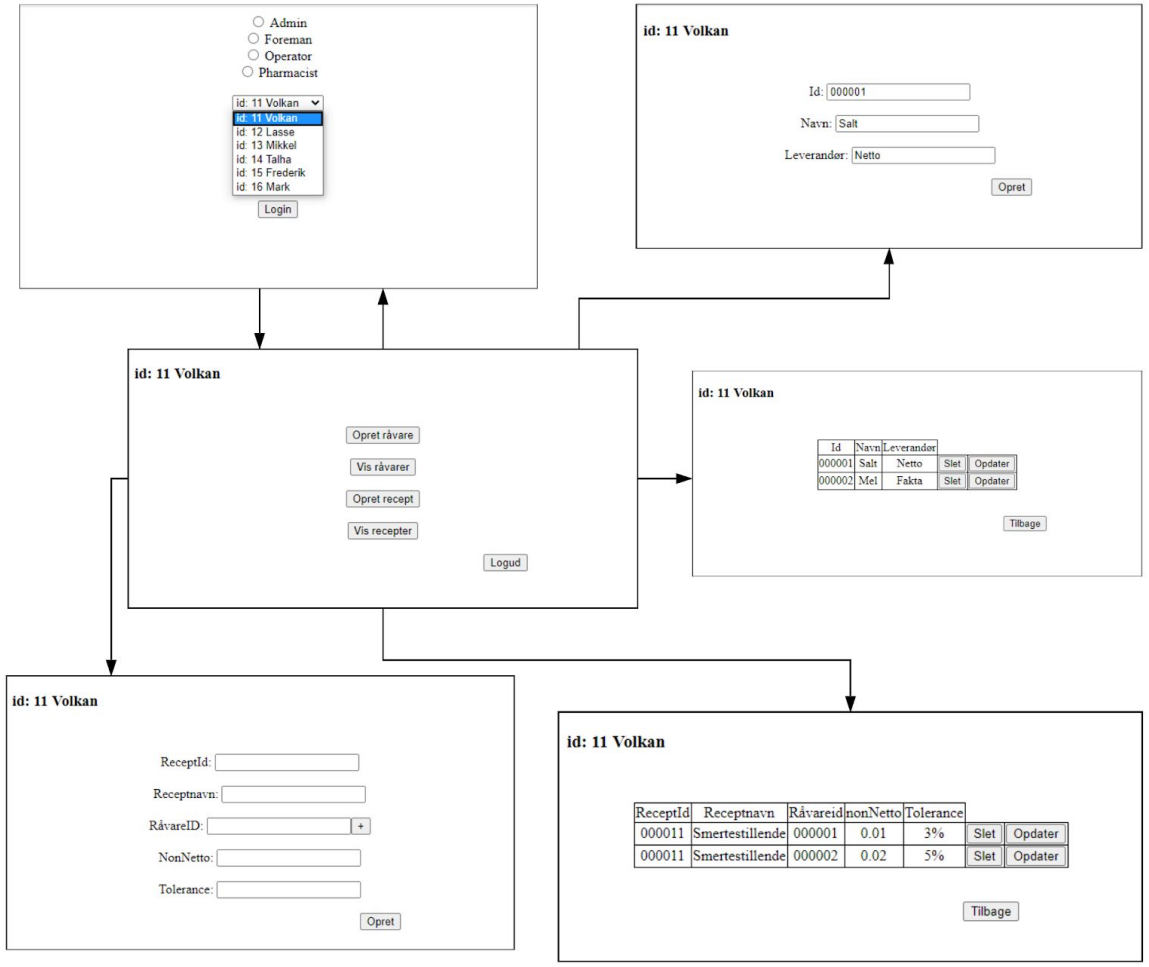
Use Case Section	Comment
Navn	Opdater bruger
Scope	CDIO 2
Level	User-goal
Primær aktør	Bruger
Preconditions:	Programmet startet og en database valgt
Postconditions:	En brugere bliver ændret
Hoved succes scenarie:	1) Brugeren trykker på knappen ''update'' bruger. 2) Brugeren indtaster ændringerne som skal ændres(Brugernavn, initialer , cpr, password, role). 3) Brugeren trykker på knappen ''update''. 4) Brugeren bekræfter sine ændringer og trykker ''ok''.
Alternative scenarier:	4.a: Brugeren trykker på ''OK'' og nu er brugeren har ændret sine oplysninger. 4.b: Brugeren trykker på ''CANCEL'' og opdater siden bliver gentaget igen.

Fully dressed

Use Case Section	Comment
Navn	Slet bruger
Scope	CDIO 2

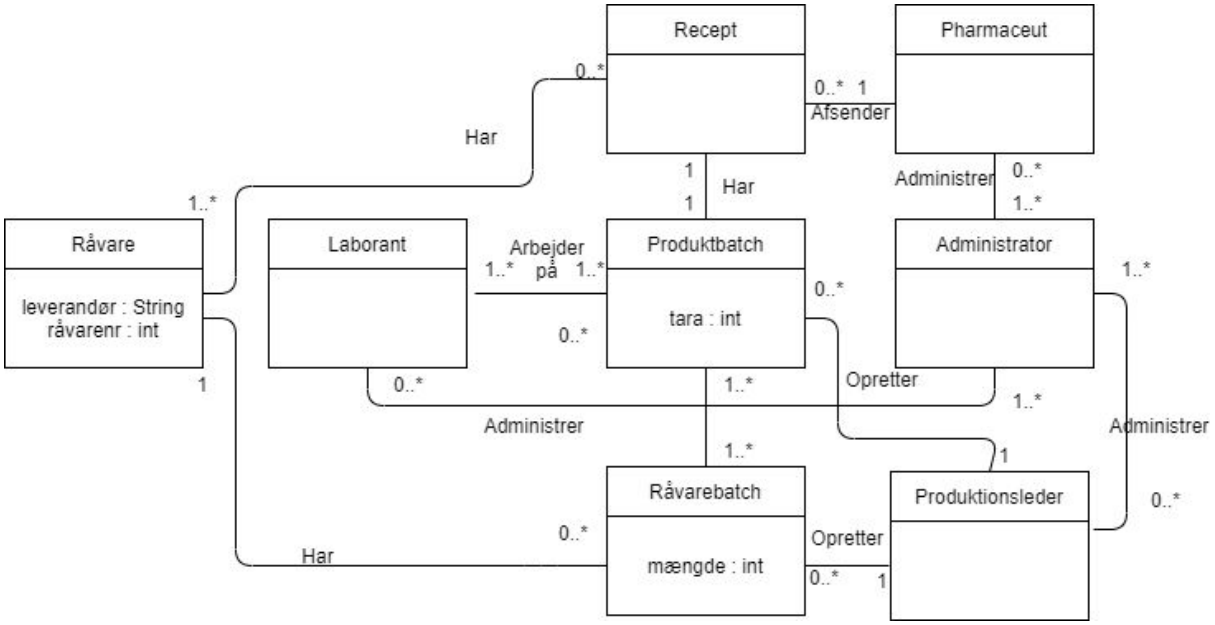
Level	User-goal
Primær aktør	Bruger
Preconditions:	Hjemmesiden startet og en brugeren er logget ind
Postconditions:	En bruger er blevet slettet
Hoved succes scenarie:	1. Brugeren trykker på ''slet'' knappen. 2. Brugeren får en besked på toppen af skærm, om de sikker på at slette den bruger. 3. Brugeren trykker på ''ok'' 4. Den valgte bruger bliver slettet fra systemet
Alternative scenarier:	1. Brugeren følger trin 1-2 fra hoved succes scenarie. 3.a: Brugeren trykker på ''cancel''og der sker ikke noget 3.b: Brugeren trykker på ''ok'', og brugeren er slettet fra system.

Navigeringsdiagram



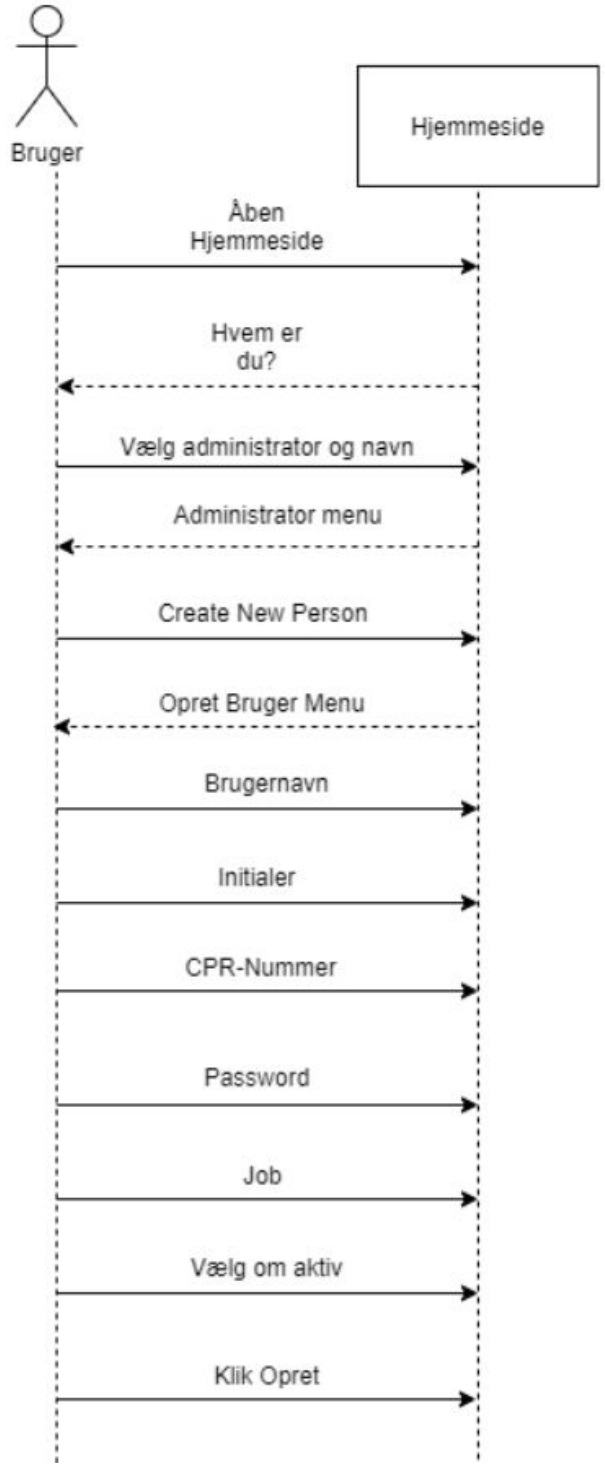
Aktivitetsdiagrammer

Domæne model

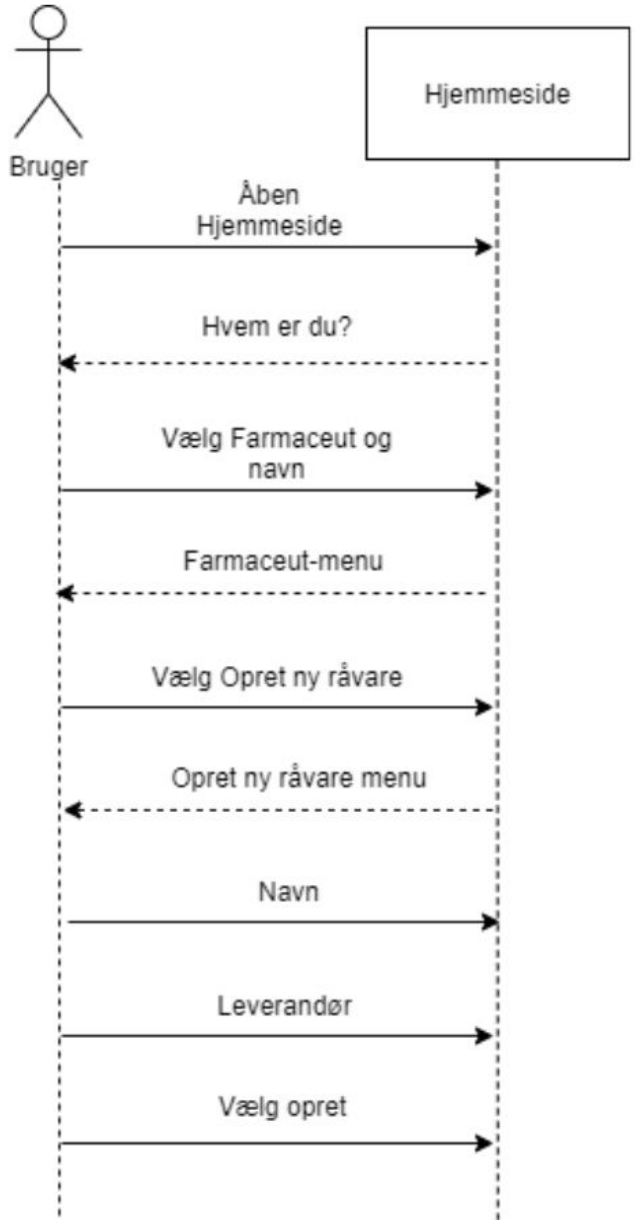


Systemsekvensdiagram

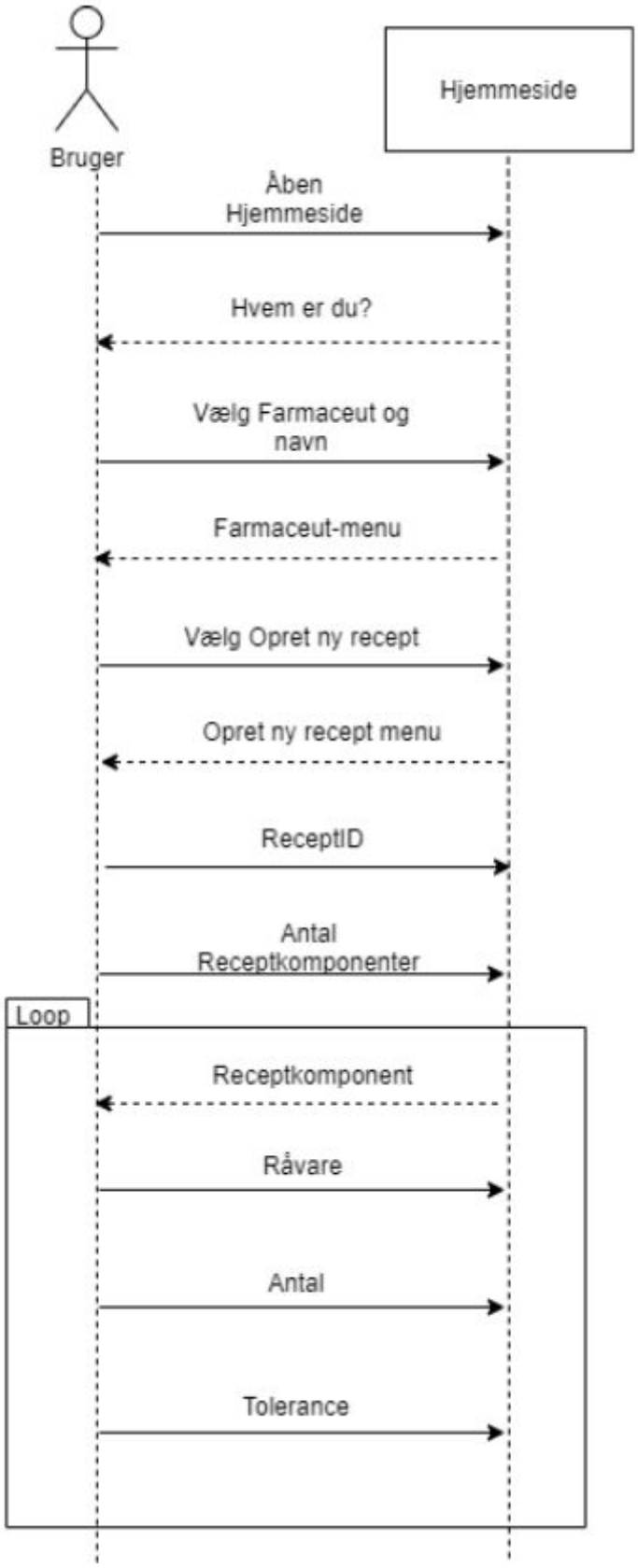
Nedenfor ses et systemsekvensdiagram over use casen “opret bruger”:


Bruger
Hjemmeside
Åben Hjemmeside
Hvem er du?
Vælg administrator og navn
Administrator menu
Create New Person
Opret Bruger Menu
Brugernavn
Initialer
CPR-Nummer
Password
Job
Vælg om aktiv
Klik Opret

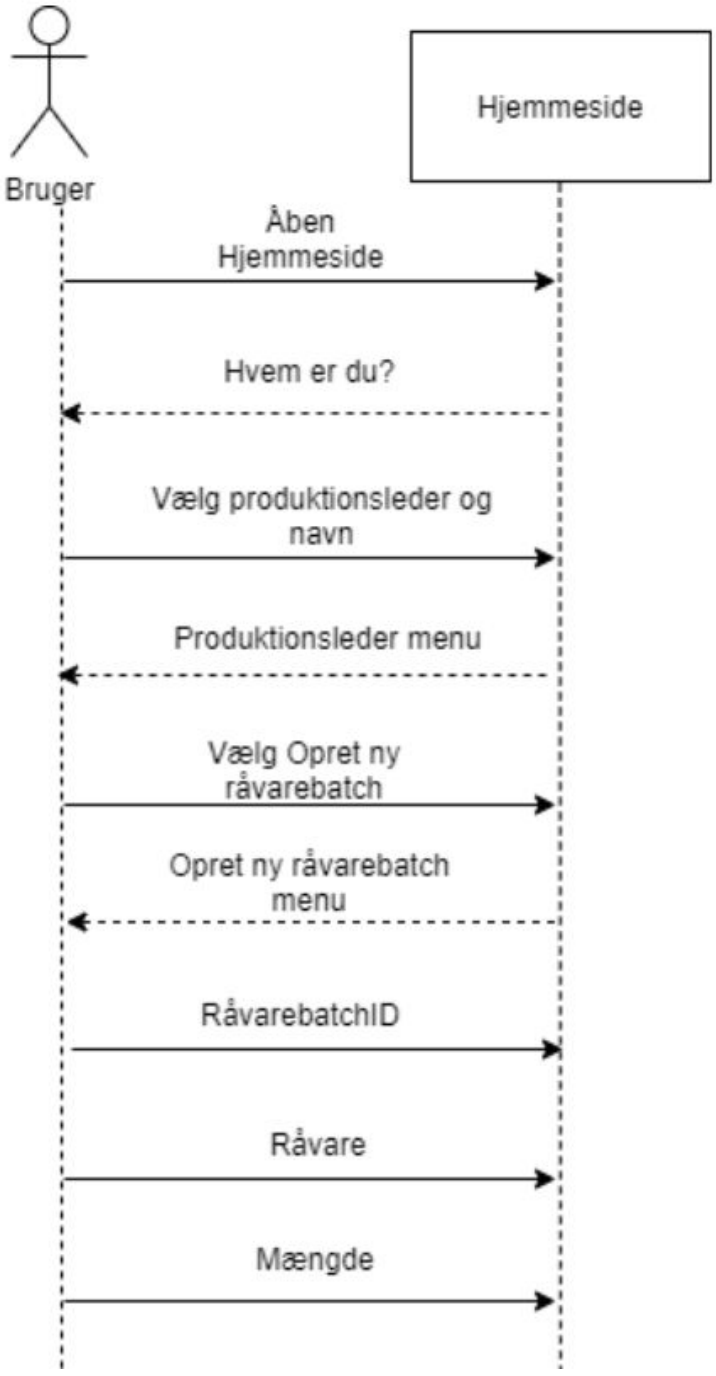
Opret råvare



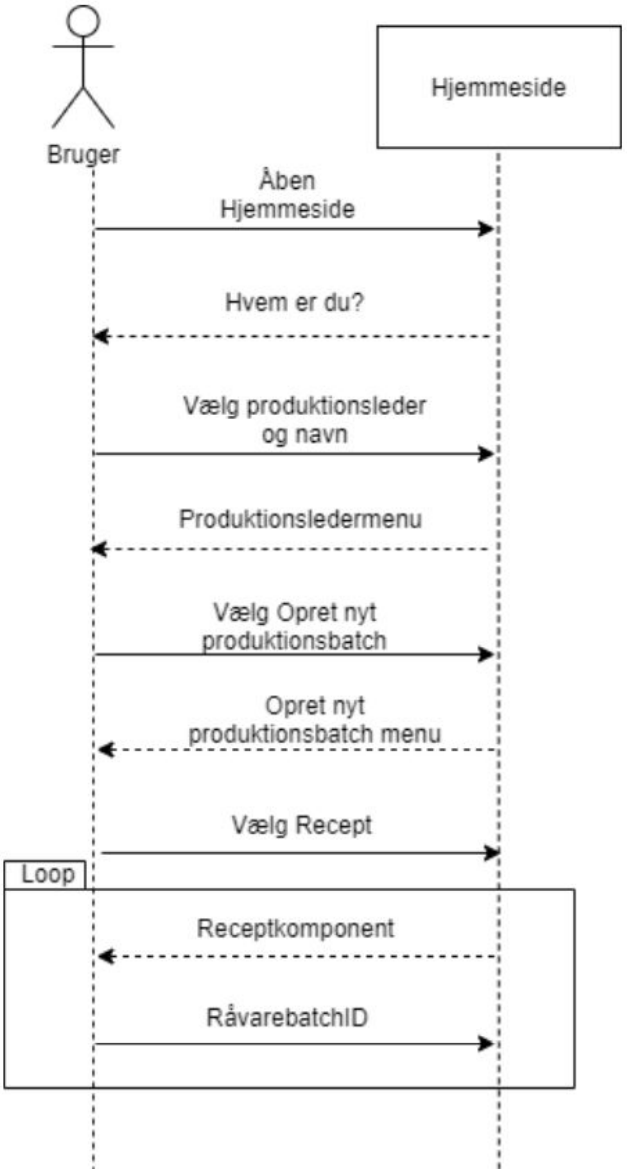
Opret Recept



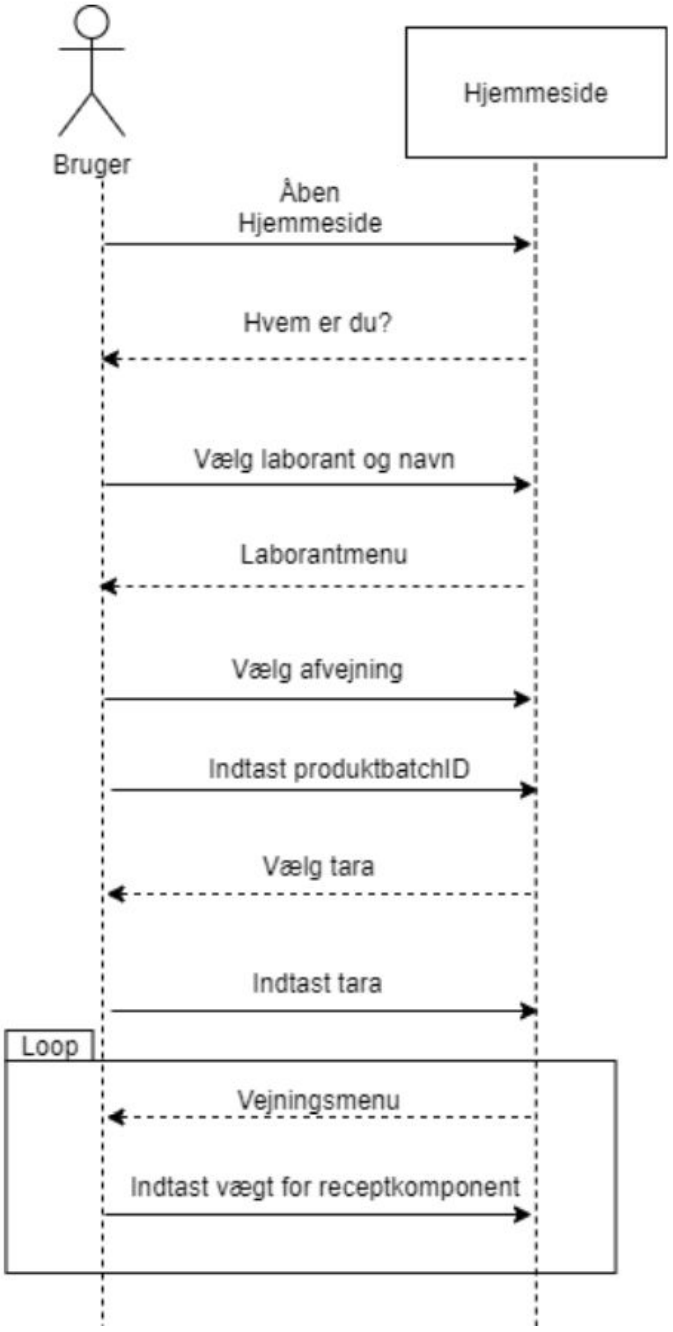
Opret Råvarebatch



Opret Produktbatch



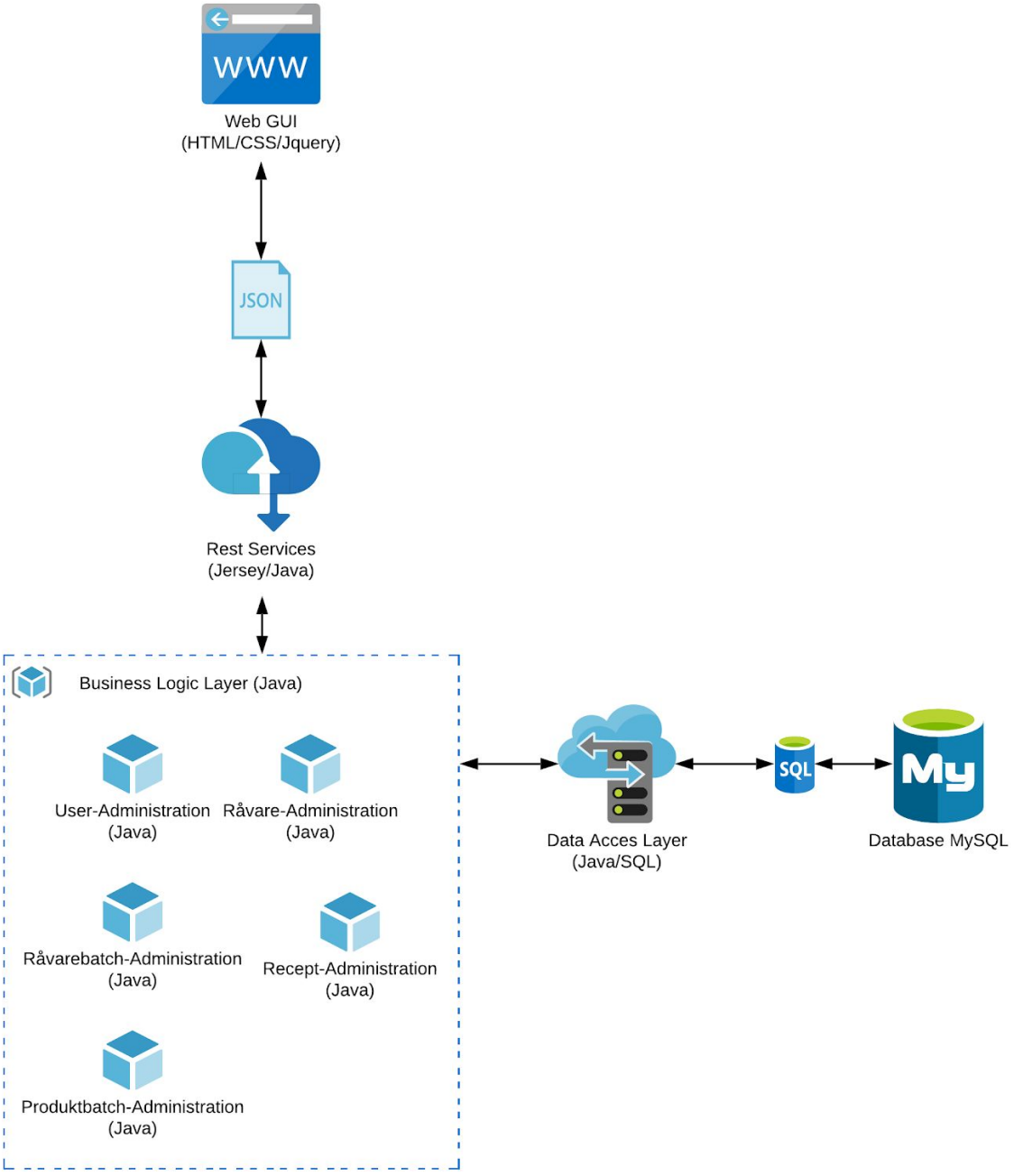
Afvejning



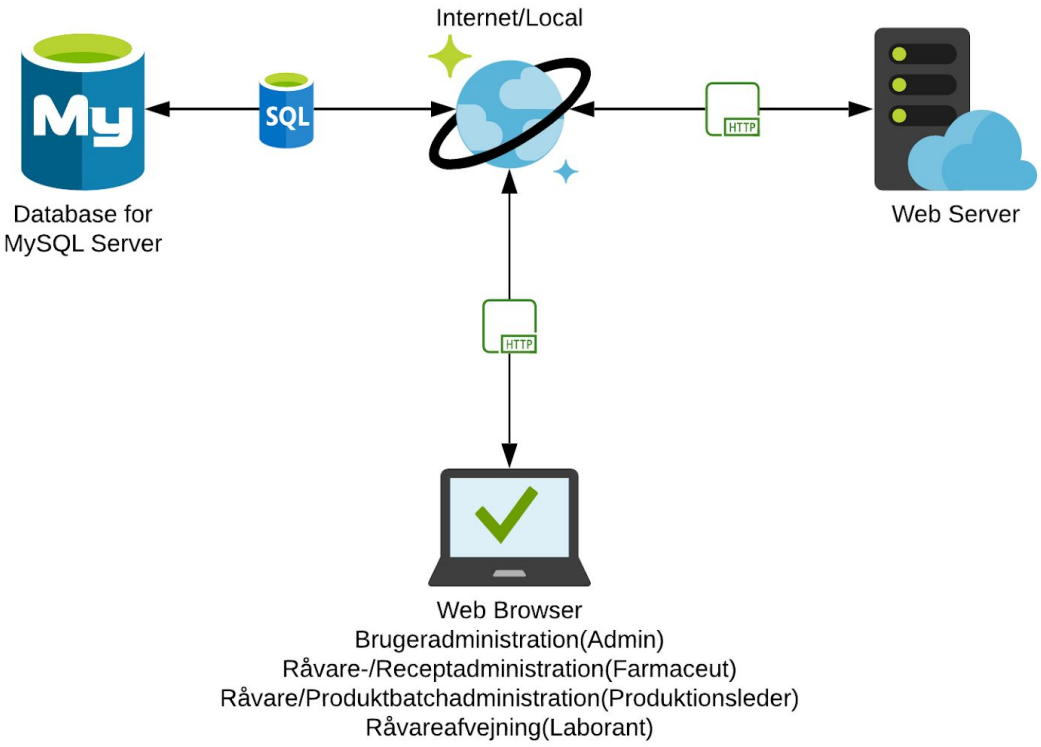
Lav et pr. use case

Design

Systemarkitekturdiagram



System Deployment diagram



Klassebeskrivelser

Functionality: Logikken for validering af input og andre funktioner.
AppConfig: Defination af applikation stien.
UserAPI: Indeholder Rest webservices med Jersey for UserDTO
RåvareAPI: Indeholder Rest webservices med Jersey for RåvareDTO
RåvareBatchAPI: Indeholder Rest webservices med Jersey for RåvareBatchDTO
ReceptAPI: Indeholder Rest webservices med Jersey for ReceptDTO
ProduktbatchAPI: Indeholder Rest webservices med Jersey for ProduktBatchDTO
SQLDatabaseIO: Kommunikation af SQL databasen (query/manipulation)
UserDTO: UserDto er objektet der bliver transmitteret igennem vores system
UserController: Manipulerer og henter data fra UserDAOSQL ved hjælp af UserAPI
UserDAOSQL: Manipulerer og henter data fra fjerner fra User tabel i mySQL database
RåvarerDTO: RåvarerDTO er objektet der bliver transmitteret igennem vores system
RåvareController: Manipulerer og henter data fra RåvareDAOSQL ved hjælp af RåvareAPI
RåvarerDAOSQL: Manipulerer og henter data fra fjerner fra Råvare tabel i mySQL database

RåvarebatchDTO: RåvarebatchDTO er objektet der bliver transmitteret igennem vores system
RåvarebatchDAOSQL: Manipulerer og henter data fra fjerner fra Råvarebatch tabel i mySQL database
RåvarebatchController: Manipulerer og henter data fra RåvarebatchDAOSQL ved hjælp af RåvarebatchAPI
ReceptDTO: ReceptDTO er objektet der bliver transmitteret igennem vores system
ReceptController Manipulerer og henter data fra ReceptDAOSQL ved hjælp af ReceptAPI
ReceptDAOSQL: Manipulerer og henter data fra fjerner fra Recept tabel i mySQL database
ReceptkompDTO??
ProduktbatchDTO:: ProduktbatchDTO er objektet der bliver transmitteret igennem vores system

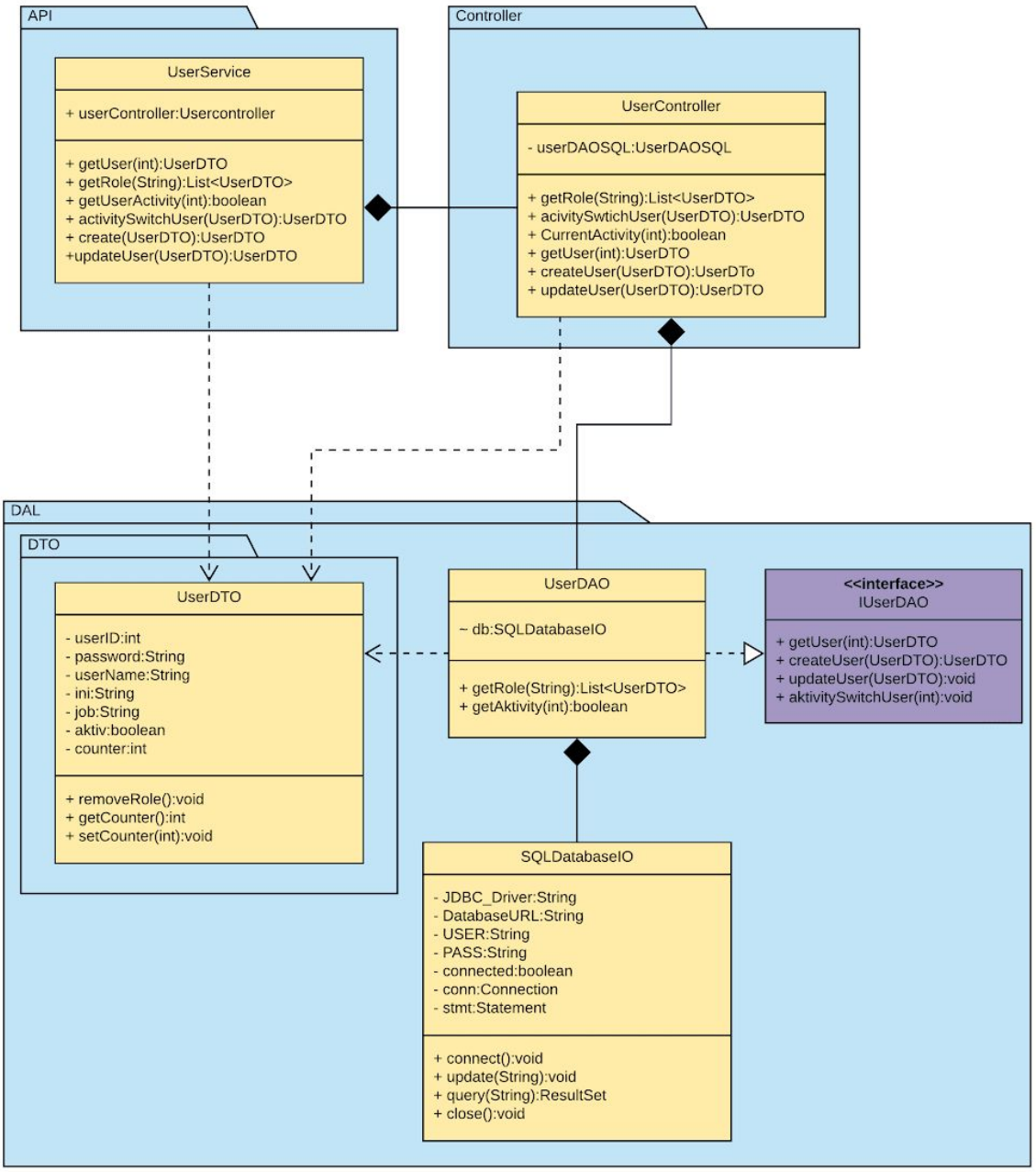
ProduktbatchController: Manipulerer og henter data fra ProduktbatchDAOSQL ved hjælp af ProduktbatchAPI

ProduktbatchDAOSQL: Manipulerer og henter data fra fjerner fra Produktbatch tabel i mySQL database

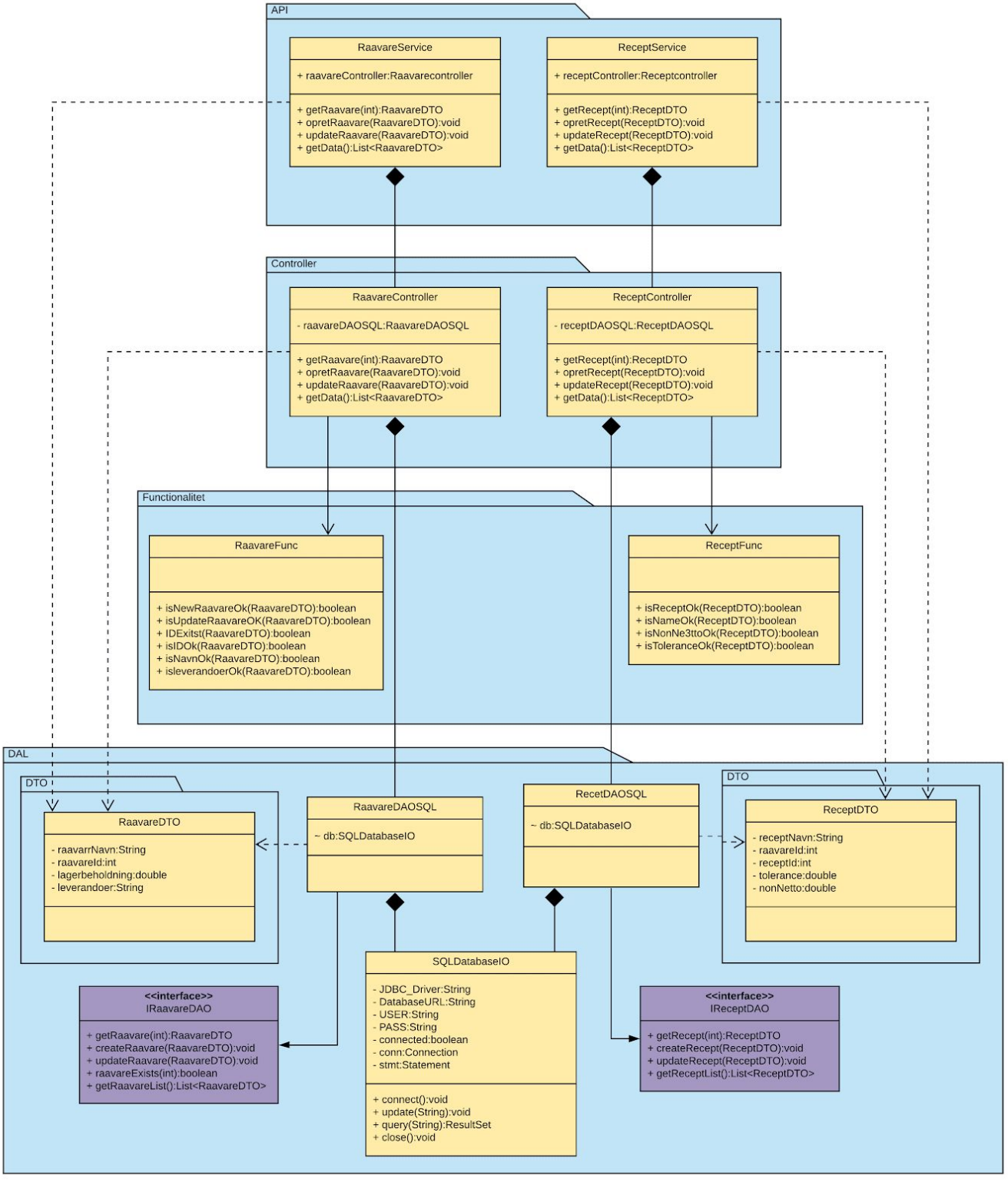
ProduktbatchkompDTO:??

Design Class Diagram

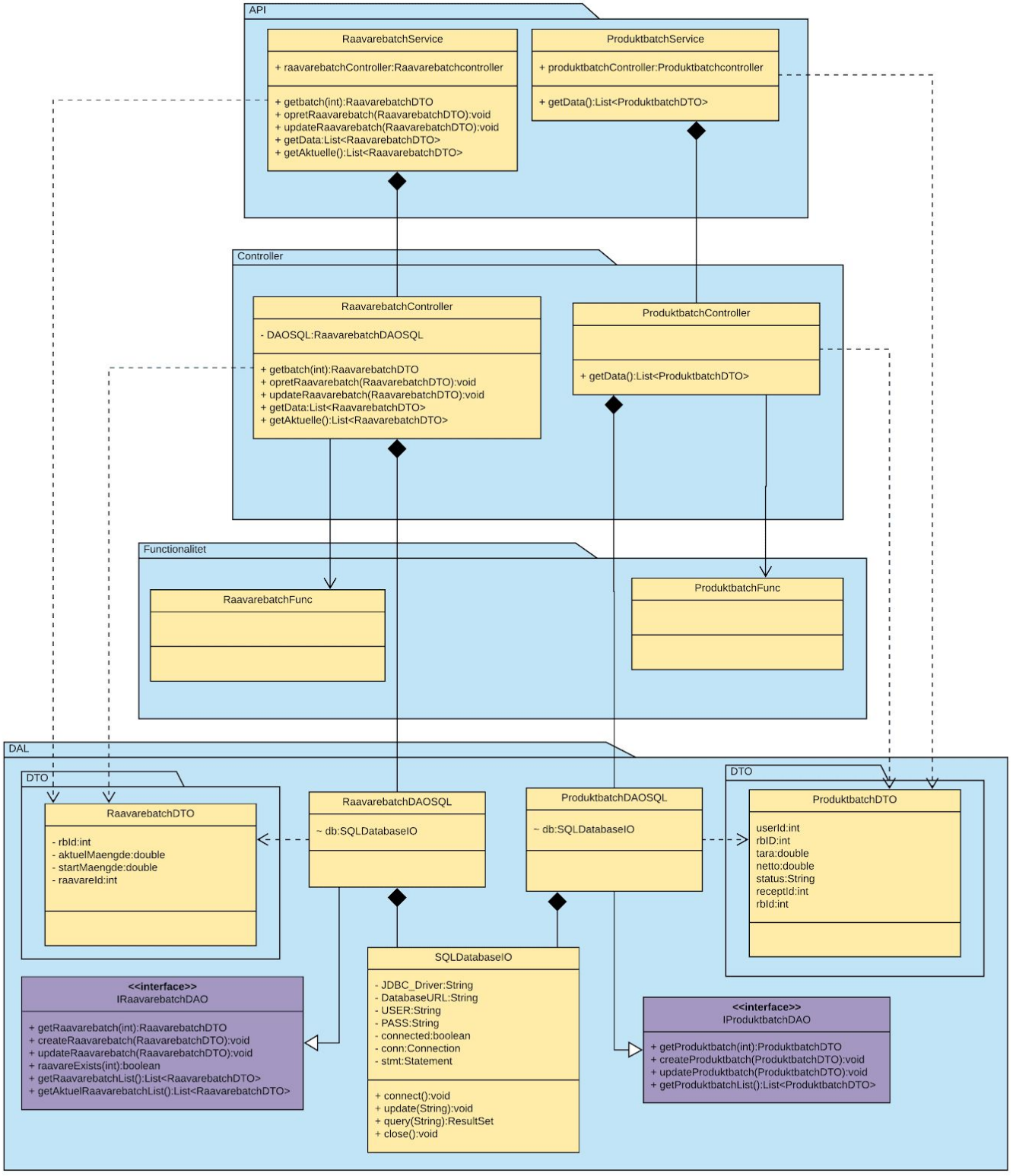
Admin klassediagram



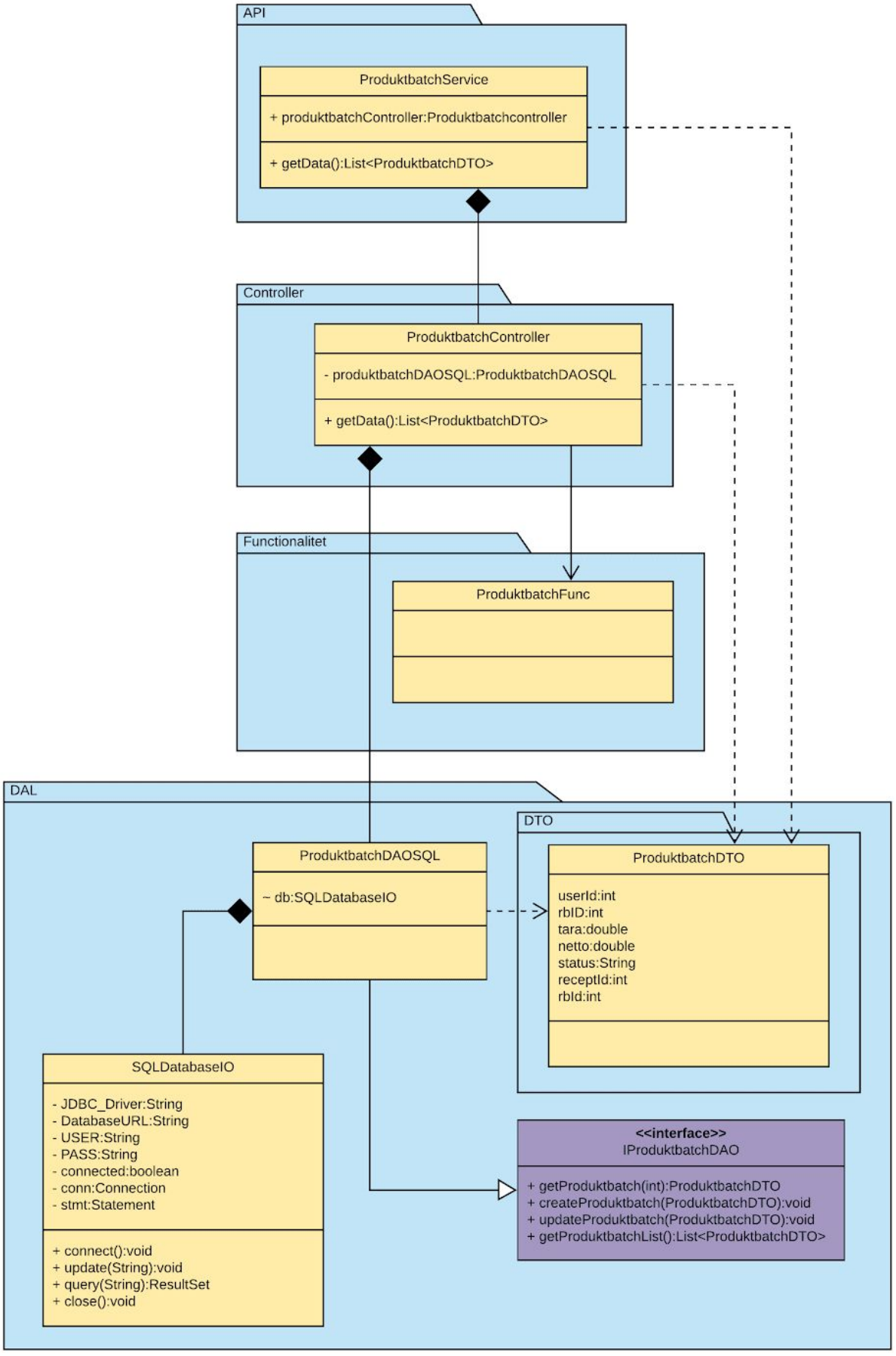
Farmaceut Diagram



Produktleder diagram



Laborant diagram


API
ProduktbatchService
+ produktbatchController:Produktbatchcontroller
+ getData():List<ProduktbatchDTO>
Controller
ProduktbatchController
- produktbatchDAOSQL:ProduktbatchDAOSQL
+ getData():List<ProduktbatchDTO>
Functionalitet
ProduktbatchFunc
DAL
DTO
ProduktbatchDAOSQL
~ db:SQLDatabaseIO
ProduktbatchDTO
userId:int
rbID:int
tara:double
netto:double
status:String
receptId:int
rbId:int
SQLDatabaseIO
- JDBC_Driver:String
- DatabaseURL:String
- USER:String
- PASS:String
- connected:boolean
- conn:Connection
- stmt:Statement
+ connect():void
+ update(String):void
+ query(String):ResultSet
+ close():void
<<interface>>
IProduktbatchDAO
+ getProduktbatch(int):ProduktbatchDTO
+ createProduktbatch(ProduktbatchDTO):void
+ updateProduktbatch(ProduktbatchDTO):void
+ getProduktbatchList():List<ProduktbatchDTO>

Dependency diagram over web

Implementering

Overvejelser

3-lags modellen

Vi har i vores projekt valgt at benytte os af trelagsmodellen, for at gøre systemets struktur mere overskuelig og de enkelte moduler nemmere at forstå. Samtidig bidrager modulariseringen til et system der er mere genanvendeligt, nemmere at vedligeholde samt lettere at udvide i fremtiden, da de enkelte delelementer kan betragtes isoleret.[2]

Vores grænseflade findes i webapp mappen og består af en række .html, .css og .js filer. HTML og CSS styrer strukturen og layoutet på hjemmesiden og JavaScript står for kommunikation med de forskellige webservices. Vi har desuden brugt Jquery biblioteket for at manipulerer og kommunikere med backenden.

Vores controllerlag indeholder de forskellige webservices, der er dekorerede med vha. Jersey til at definere forskellige resources og actions. Vha. Jackson kan vi nemt oversat JSON til klassiske Java-objekter og omvendt.

Til sidst har vi datalaget, som controllerlaget arbejder tæt sammen med. Her finder vi de nødvendige klasser for at oprette en forbindelse til og kommunikere med vores database. Bemærk hvordan vi til at overføre information mellem de forskellige lag benytter en såkaldt DTO (data transfer objekt), til at overskuelige processen.

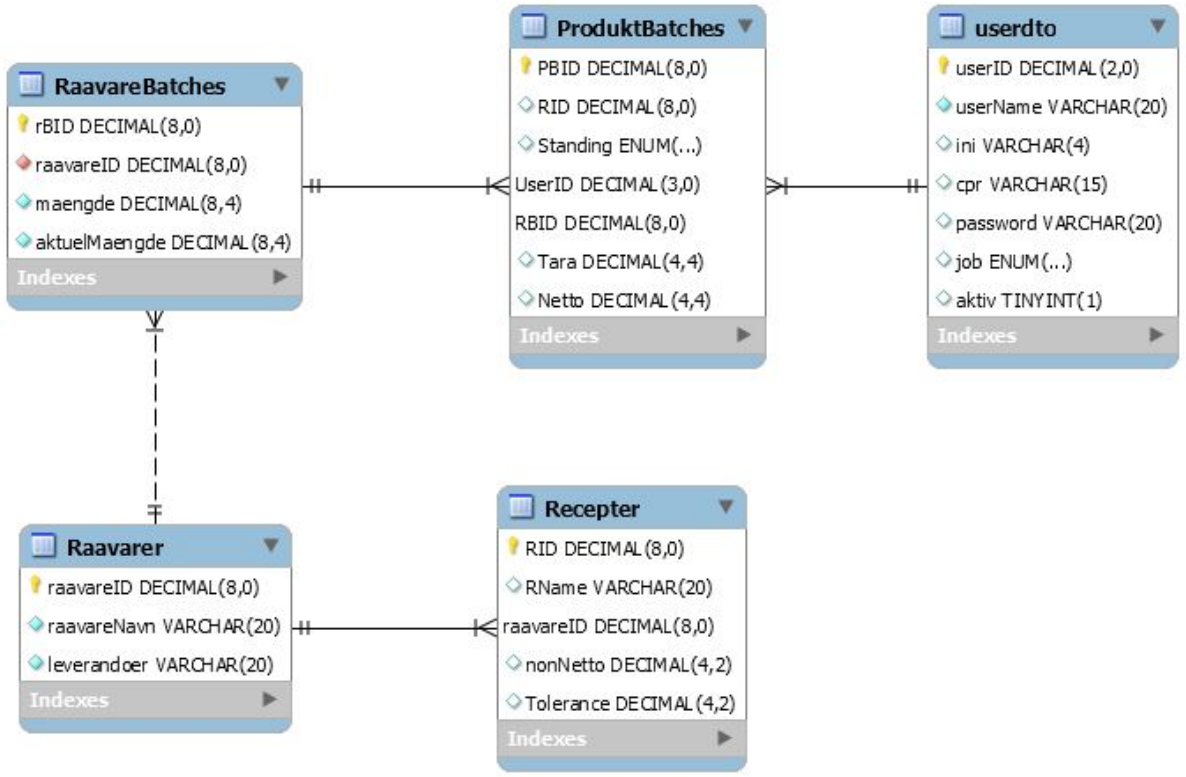
Som det kan ses, indeholder datalaget forskellige interfaces , hvilket vi med fordel også kunne have anvendt i de andre lag. Dette ville have forstærket effekt af modulariseringen yderligere og gjort vores software mere fleksibel.

Da der i vores løsning kun forekommer rene datamanipulationsopgaver, var der ikke har behov for et funktionalitetslaget. Skulle vi også have implementeret f.eks. kodeordsgodkendelse, havde det været en fordel med et funktionalitetslag. Bemærk også, hvordan vi benytter en såkaldt DTO (data transfer objekt) til at overføre information mellem de forskellige lag, for at overskueliggøre processen.

[2] "3 lags modellen" af Mads Nyborg og Uffe Kofoed, 2013, s. 1-2.

Database server

EER diagram



Vi har i vores løsning valgt at benytte en online MYSQL server som database, for at undgå at brugeren skal have en lokal database server liggende. Databasen er opsat med tre tabeller: userdto (brugeren), roles (roller) og userdto_roles (Hvilke userdto brugere har hvilke roller).

userdto

userID	userName	ini	cpr	password

roles

role_name

userdto_roles

userID	role_name

Demodata

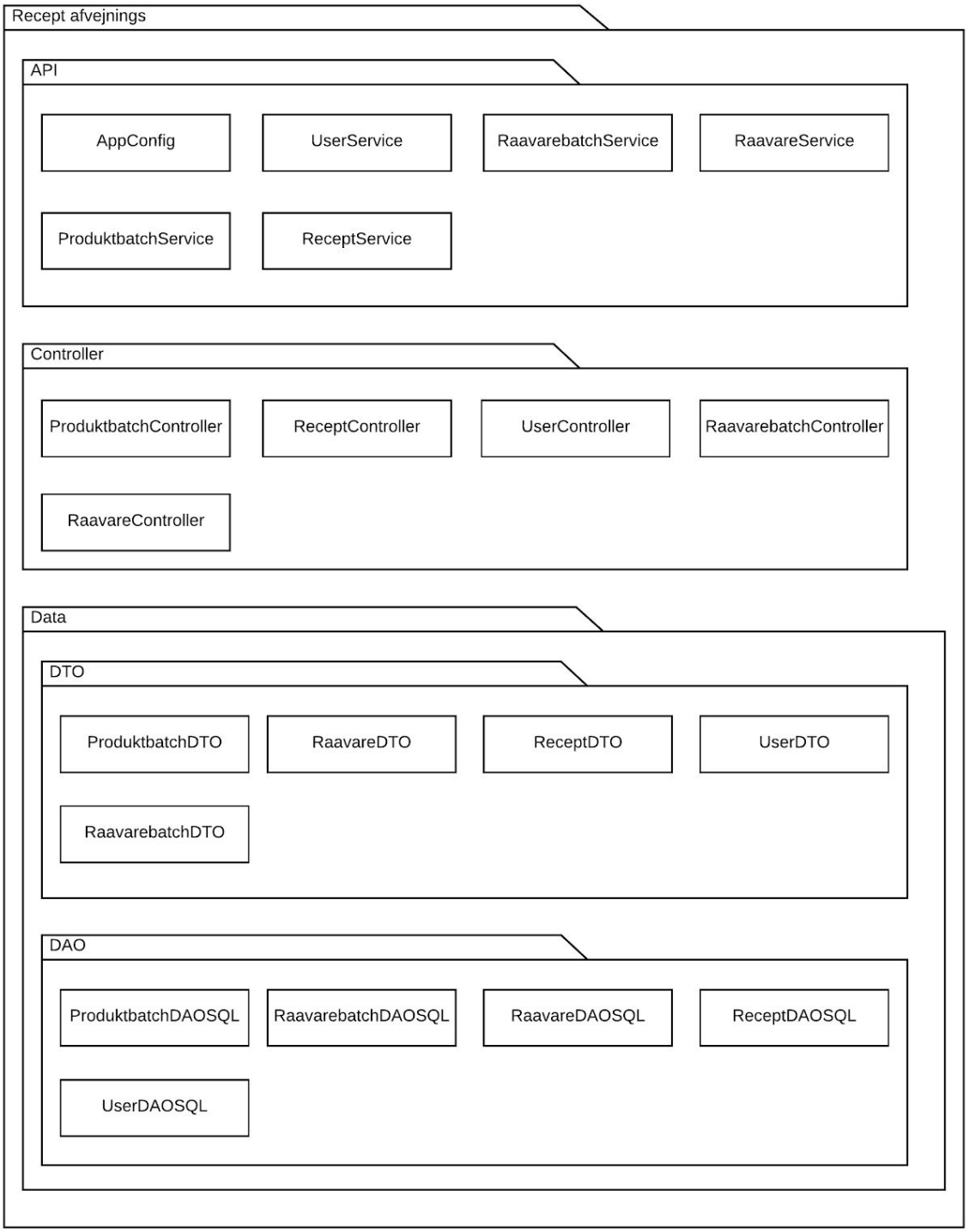
Nedenfor ses den demodata som alle systemets databaser er blevet initialiseret med

ID	Navn	Ini.	CPR	Kodeord	Roller
11	Volkan	VOL	0912941235	SkiftMig!	Admin
12	Mikkel	MIK	3007902345	SkiftMig!	Pharmacist
13	Mark	MAR	1603783455	SkiftMig!	Foreman
14	Talha	TAL	2301974565	SkiftMig!	Operator
15	Frederik	FRE	0501025675	SkiftMig!	Pharmacist, Operator
16	Lasse	LAS	1104966785	SkiftMig!	Admin, Pharmacist, Foreman, Operator

Platform

Operativsystemer testet på	Windows 10 Version 1903 Build 18362.476 MacOS 10.15.1 Ubuntu 20.04
Java	Java 1.8.0_161
IntelliJ	IntelliJ IDEA 2019.2.3 (Ultimate Edition) Build #IU-192.6817.14, built on September 24, 2019
Jquery	jquery-3.5.1.min
MySql	mariaDB 10.3
JAX-RS	javax 2.1.1
Jersey	Jersey 2.27
Servlet	Javax servlet 4.0.1
Junit	Junit 5.7

Pakkediagram:


Recept afvejnings
API
AppConfig
UserService
RaavarebatchService
RaavareService
ProduktbatchService
ReceptService
Controller
ProduktbatchController
ReceptController
UserController
RaavarebatchController
RaavareController
Data
DTO
ProduktbatchDTO
RaavareDTO
ReceptDTO
UserDTO
RaavarebatchDTO
DAO
ProduktbatchDAOSQL
RaavarebatchDAOSQL
RaavareDAOSQL
ReceptDAOSQL
UserDAOSQL


webapp
AdminScripts
currentUserScript
default
index
jqScripts
AdminScreen
Admin
index
jquery
jquery-3.5.1.min
Brugeroversigt
index
users
Opretbruger
index
index
createUser
Opretbruger
index
index
updateUser
fetchUser
FarmaScreen
index
NyRaavare
index
NyRecept
index
VisRaavare
index
UpdateRaavare
index
VisRecept
index
UpdateRecept
index
PLeadScreen
index
Login
Login
Login
Login
LabScreen
index
HTML
CSS
JavaScript

Brugervejledning

Installation og kørsel af CDIO1.jar

- Programmet kræver Java 8.
- Programmet er bygget til at køre på en Windows 10 maskine.
- Apache Tomcat 8.5.54 Core downloades her[3] og pakkes ud.
- Udpak 11_del2.zip.
- Mappen udpakket åbnes som projekt i IntelliJ Ultimate edition 2020.1.1+
- Mavenfilen pom.xml genimporteres
- Øverst til højre vælges “Edit Configuration”
- Tryk på plus tegnet øverst til venstre i vinduet og Og tomcat > local vælges.
- Application server Configureres ved at vælge mappen apache-tomcat-8.5.54
- Hold alt andet standard of tryk på “fix” nederst til højre.
- Her vælges artifaktet “BoilerPlate:war exploded” og der trykkes “Apply” og “OK”
- Programmet køres ved tryk på den grønne pil øverst til højre.

[3] https://tomcat.apache.org/download-80.cgi

Unit Test

Test case ID	TC01
Summary	Tester om SQLDatabaseIO klassen forbinder til databasen, sender efterspørgsel og manipulerer data korrekt
Procedure	1. connect() metoden kaldes og variablen "isConnected" kontrolleres. 2. En efterspørgsel oprettes og query metoden kaldes. 3. Forventet data sammenlignes med aktuel efterspørgselsdata. 4. En manipulation oprettes og update metoden kaldes. 5. Forventet data sammenlignes med aktuel efterspørgselsdata ved hjælp af query metoden.
Data	sqlDatabase forbindelse: user: kamel password: dreng url :runerne.dk port: 8003 **forventet = true , aktuel = true** query: SELECT * FROM Recepter WHERE RID=10 **forventet=10 , aktuel = 10** update(manipulation): insert into Recepter (RID, RName, raavareID, nonNetto, Tolerance) VALUE ('99', 'Mojito' ,'1' ,'6.2' ,'1.5') **forventet=99 , aktuel=99**
Status	Bestået
Created by	Mark Rune Mortensen
Tested by	Volkan Isik
Creation date	10/05/2020
Last tested	15/06/2020
Environment	IntelliJ IDEA 2019.2.3 (Ultimate Edition) Build #IU-192.6817.14, built on September 24, 2019 on Windows 10 Enterprise Version 1903 OS build 18362.418 Google Chrome

Test case ID	TC02
Summary	Tester om UserDAOSQL klassen manipulerer userdto tabellen i databasen
Procedure	1. Initalisering af UserDAOSQL objekt. 2. testUser variablen initialiseres ved hjælp af getUser metoden med 11 som parameter. 3. Forventet userID sammenlignes med den initialiseret testUser. 4. Liste af UserDTO initialiseres med getRole metoden som returnerer en liste af specifik UserDTO objekter af en given rolle. 5. Listens størrelse sammenlignes med forventet data. 6. En liste af UserDTO initialiseres med getData som returnerer alle rækker i userDTO tabellen i vores database. 7. Listens størrelse sammenlignes med forventet data. 8. En UserDTO objekt initaliseres og bruges som parameter i createUser metoden. 9. Den oprettede bruger læses fra databasen og sammenlignes med forventet userID. 10. En UserDTO objekt initaliseres og bruges som parameter i updateUser metoden. 11. Den opdateret bruger læses fra databasen og sammenlignes med forventet job. 12. Aktual boolsk variable initialiseres med getActivity metoden. 13. Aktuel variablen kontrolleres for sandhed. 14. userDTO aktivitet data skiftes til at være sand med metoden aktivitySwitchUser. 15. aktiv variablen kontrolleres for ændring. 16. Alt manipulation i databasen slettes.
Data	getUser() **forventet = 11 , aktuel = 11** getRole() **forventet=1 , aktuel = 1** getData() **forventet=FD , aktuel=FD** **forventet=Lasse , aktuel=Lasse** createUser() UserName: "Test" Ini: "TES" Job: "Laborant" Password: "passNew" Aktiv: false **forventet=Test , aktuel=Test** updateUser() Job: Produktionsleder **forventet=Produktionsleder , aktuel=Produktionsleder**

	getActivity() **forventet=false , aktuel=false** aktivitySwitchUser() **forventet=true , aktuel=true**
Status	Bestået
Created by	Mark Rune Mortensen
Tested by	Volkan Isik
Creation date	10/05/2020
Last tested	15/06/2020
Environment	IntelliJ IDEA 2019.2.3 (Ultimate Edition) Build #IU-192.6817.14, built on September 24, 2019 on Windows 10 Enterprise Version 1903 OS build 18362.418 Google Chrome

Test case ID	TC03
Summary	Tester om ReceptDAOSQL klassen manipulerer Recepter tabellen i databasen
Procedure	1. Initalisering af ReceptDAOSQL objekt. 2. testRecept variablen initialiseres ved hjælp af getRecept metoden med 10 som parameter. 3. Forventet receptID sammenlignes med den initialiseret testRecept. 4. En liste af receptDTO initialiseres med getReceptList som returnerer alle rækker i Recepter tabellen i vores database. 5. Element nummer 4 i listen sammenlignes med forventet data. 6. En receptDTO objekt initialiseres og bruges som parameter i createRecept metoden. 7. Den oprettede recept læses fra databasen og sammenlignes med forventet receptID. 8. En ReceptDTO objekt initaliseres og bruges som parameter i updateRecept metoden. 9. Den opdateret recept læses fra databasen og sammenlignes med forventet nonNetto værdien. 10. Alt manipulation i databasen slettes.
Data	getRecept() **forventet = 10 , aktuel = 10**

	getReceptList() **forventet=11 , aktuel = 11** createRecept() ReceptId: 99 ReceptNavn: "Morfin" RaavareId: 3 NonNetto: 5.5 tTolerance: 9.5 **forventet=99 , aktuel = 99** updateRecept() NonNetto: 3.5 **forventet=3.5 , aktuel = 3.5**
Status	Bestået
Created by	Mark Rune Mortensen
Tested by	Volkan Isik
Creation date	10/05/2020
Last tested	15/06/2020
Environment	IntelliJ IDEA 2019.2.3 (Ultimate Edition) Build #IU-192.6817.14, built on September 24, 2019 on Windows 10 Enterprise Version 1903 OS build 18362.418 Google Chrome

Test case ID	TC04
Summary	Tester om RaavareDAOSQL klassen manipulerer Raavarer tabellen i databasen
Procedure	1. Initalisering af RavarerDAOSQL objekt. 2. testRaavare variablen initialiseres ved hjælp af getRaavare metoden med 1 som parameter. 3. Forventet raavareID sammenlignes med den initialiseret testRaavare. 4. En liste af raavareDTO initialiseres med getRaavareList som returnerer alle rækker i Raavarer tabellen i vores database. 5. Element nummer 1 i listen sammenlignes med forventet data. 6. En raavarerDTO objekt initialiseres og bruges som parameter i createRaavare metoden. 7. Den oprettede råvare læses fra databasen og sammenlignes med forventet raavareID.

	8. En RaavareDTO objekt initaliseres og bruges som parameter i updateRaavare metoden. 9. Den opdateret råvare læses fra databasen og sammenlignes med forventet leverandoer værdien. 10. Boolsk variabel initialiseres med raavareExists metoden med parameter 99 som raavareID nummer. 11. Variablen kontrolleres for forventet resultat. 12. Alt manipulation i databasen slettes.
Data	getRaavare() **forventet = 1 , aktuel = 1** getRaavareList() **forventet=2 , aktuel = 2** createRecept() RaavareID: 99 RaavareNavn: "Vodka" Leverandoer: "Leo" LagerBeholdning: 10.5 **forventet=99 , aktuel = 99** updateRecept() Leverandoer: "Novo" **forventet=Novo , aktuel = Novo** raavareExists() **forventet=true , aktuel = true**
Status	Bestået
Created by	Mark Rune Mortensen
Tested by	Volkan Isik
Creation date	10/05/2020
Last tested	15/06/2020
Environment	IntelliJ IDEA 2019.2.3 (Ultimate Edition) Build #IU-192.6817.14, built on September 24, 2019 on Windows 10 Enterprise Version 1903 OS build 18362.418 Google Chrome

Test case ID	TC05

Summary	Tester om RaavarebatchDAOSQL klassen manipulerer Raavarerbatch tabellen i databasen
Procedure	1. Initalisering af RavarerbatchDAOSQL objekt. 2. testRaavarebatch variablen initialiseres ved hjælp af getRaavarebatch metoden med 1 som parameter. 3. Forventet raavarebatchID sammenlignes med den initialiseret testRaavarebatch. 4. En liste af raavarebatchDTO initialiseres med getRaavarebatchList som returnerer alle rækker i Raavarerbatches tabellen i vores database. 5. Element nummer 1 i listen sammenlignes med forventet data. 6. En liste af raavarebatchDTO initialiseres med getAktuelRaavarebatchList som returnerer alle rækker i Raavarerbatches tabellen som har aktuelMaengde større end 0 i vores database. 7. Element nummer 2 i listen sammenlignes med forventet data 8. En raavarerbatchDTO objekt initialiseres og bruges som parameter i createRaavarebatch metoden. 9. Den oprettede råvarebatch læses fra databasen og sammenlignes med forventet raavarebatchID. 10. En RaavarebatchDTO objekt initaliseres og bruges som parameter i updateRaavarebatch metoden. 11. Den opdateret råvarebatch læses fra databasen og sammenlignes med forventet aktuelmængde værdien. 12. Alt manipulation i databasen slettes.
Data	getRaavarebatch() **forventet = 1 , aktuel = 1** getRaavarebatchList() **forventet=2 , aktuel = 2** getAktuelRaavarebatchList() **forventet=3 , aktuel = 3** createRaavarebatch() RbId: 99 tRaavareId: 2 AktuelMaengde: 85.5 StartMaengde: 100.0 **forventet=99 , aktuel = 99** updateRaavarebatch() AktuelMaengde: 70.5 **forventet=70.5 , aktuel = 70.5**
Status	Bestået
Created by	Mark Rune Mortensen
Tested by	Volkan Isik

Creation date	10/05/2020
Last tested	15/06/2020
Environment	IntelliJ IDEA 2019.2.3 (Ultimate Edition) Build #IU-192.6817.14, built on September 24, 2019 on Windows 10 Enterprise Version 1903 OS build 18362.418 Google Chrome

Black Box Test

Ordbog

Konklusion

Bilag

Bilag 1: CDIO-D1 Rapport:
https://docs.google.com/document/d/1a9eWnNCtYzhWIhl8TpB49kwtZdvbW_2mMQBvW_alCTg/edit?usp=sharing