



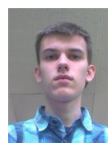
02324

VIDEREGÅENDE PROGRAMMERING

CDIO - Final

Gruppe 17

s185118
Aleksander Lægsgaard Jørgensen



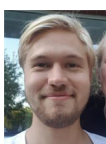
s175561
Andreas Østergaard Schliemann

s185096
Jonatan Amtoft Dahl



s164177
Josephine Weirsøe

s185103
Søren Hother Rasmussen



s185121
Theodor Peter Guttesen

21. juni 2019

DTU Compute

Institut for Matematik og Computer Science

DTU Diplom

Center for Diplomingeniøruddannelse

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

Git - Vægten: https://github.com/theogutt/Vaegt_CDIOfinal

Abstract

- *Theodor*

In this report we describe the parts of the medicine administration system we have made. We have made a weighing system and a webpage for administration of recipes, users, commodities and so on. Our goal was to create a user friendly webpage and an error-tolerant weighing system. We think these goals have been fulfilled. The report details the process we have followed, divided into the sections Requirements, Analysis, Design, Implementation, Test, Project planning and Conclusion. Furthermore it contains diagrams to illustrate the design of our program and it also gives detailed descriptions of our program. In the report there is a test chapter, where we document that our program meets the requirements specified in the first part of the report. The program uses the programming languages Java, JavaScript, HTML, CSS and MySQL. The IDEs we have used are IntelliJ and MySQL Workbench.

1 Indledning

- *Theodor*

I dette projekt har vi fået til opgave at lave en softwareløsning til medicin administration. Projektet består af to dele. Et afvejningssystem til laboranter, der skal forbindes til en vægt. Og en hjemmeside til administration og visning af brugere, råvarer, produktbatches, recepter og råvarebatches til Administratorer, Farmaceuter og Produktionsledere. Vi har brugt Java og MySQL til vægtsystemet og backenden, vi har brugt HTML, CSS og javascript til hjemmesidens frontend. Sideløbende med udviklingen af systemet, har vi udarbejdet denne tekniske rapport, for at dokumentere vores proces, vise vores diagrammer og beskrive programmet og de overvejelser vi har gjort.

Indhold

1	Indledning	1
2	Instruktioner for opstart	3
3	Timeregnskab	5
4	Krav	6
4.1	Vision	6
4.2	Kravliste	6
4.3	Domænemodel	8
5	Analyse	10
5.1	Aktører	10
5.2	Use case	10
5.3	Casual og fully-dressed	12
5.4	Traceability matrix: Use cases og krav	14
5.5	System sekvensdiagram	15
6	Design	16
6.1	Hjemmeside	16
6.2	Klassediagram	16
6.3	Pakkediagram	20
6.4	Sekvensdiagram	21
6.5	Database design	22
7	Implementering	23
7.1	Overblik	23
7.1.1	Hjemmeside	23
7.1.2	Afvejning	24
8	Test	25
8.1	Positiv test	25
8.2	Negative test	26
8.3	Unit-test	26
8.4	Coverage test	26
9	Projektplanlægning	28
10	Konklusion	30
11	Bilag	31

2 Instruktionser for opstart

- Jonatan

Vægt

1. For at kunne bruge vores vægt program, skal man først downloade program filerne fra github hjemmesiden, linket til githubben kan ses på rapportens forside. Disse filer skal være i et nyt projekt, her bruger vi IntelliJ som vores IDE.
2. Herefter skal der vælges en SDK, projektet er skrevet i Java 1.8 derfor anbefales der, at man bruger samme SDK. SDK'en ligger på DTU-computerne i mappen *C: / Program Files / Java / jdk.1.8.0_111*. Man kan sætte dette inde i *Project Structure*, som ligger under *Filer*, og under *Project*, se bilag 11..
3. Her skal *Project language level* også sættes til 8.
4. Efter JDK'en er sat op, skal man under *Project Structure*. I Project Structure gå ind på Modules og Sources, og sæt java mappen som source mappe. Java mappen ligger under *Projektnavn/src/main* se bilag 12.
5. Derefter skal vi have importeret de rigtige filer fra Maven. Dette gøres ved at trykke på *Maven Projects* ovre i højre side, og derefter klik *Reimport all Maven Projects* se bilag 13.
6. Derefter kan programmet buildes og køres gennem *WeightController* klassen.
7. For at kunne starte afvejnings proceduren, skal man først sætte et ethernet stik fra vægten til computeren. Hvis man vil bruge simulatoren så gå til trin 12.
8. Herefter skal der tjekkes, hvorvidt det er den rigtige IP-adresse programmet prøver at connecte med.
9. De to vægte har hver deres IP-adresse, den kan enten være 169.254.2.2 eller 169.254.2.3, vi forslår at man bruger puTTY, og prøver at pinge begge IP-adresser, den man får svar fra er den rigtige.
10. Når den rigtige IP-adresse er fundet, skal man gå ind i *WeightConnector* klassen og ændre variabelen *VAEGT* i linje 13 til den rigtige IP-adresse se bilag 14.
11. Herefter skal vægten startes, og når den er startet helt op, så skal java projektet køres. Herefter er vægten den eneste del man skal interagere med.
12. Hvis man bruger vægt simulatoren, skal man ændre i linje 14 hvor der står *VAEGT*, til at der står *LOCALHOST*.
13. Herefter kan man kører simulatoren, og derefter kan man kører vægt programmet.

Webpage

1. På samme måde som i vægt programmet, skal projektet først downloades fra github, linket til projektet kan findes på forsiden.
2. Igen bruger vi IntelliJ som IDE og starter med at sætte SDK'en til 1.8, igen skal language level også sættes til 8.
3. På samme måde sætter vi også *Projekt navn/src/main/java* til source mappen. For at kunne kører testene skal mappen, som hedder test, sættes til test mappe.

4. Præcis på samme måde som det forrige program, skal Maven projects imports, dette gøres på samme måde.
5. Nu mangler vi kun at opsætte Tomcat. For at gøre det klikkes knappen *Add Configurations* se bilag 15.
6. I det vindue, som åbner op vælges *Add New Configurations* (det er den lille plus knap), der skulle nu åbnes en drop down menu.
7. Kom ned i bunden af menuen og klik *x items more*. Dette forlænger drop down menuen, find i den menu knappen *Tomcat Server*. Klik på knappen og vælg *Local*, dette ændrer hele popupvinduet se bilag 16.
8. Klik på knappen *Configure...*, som åbner et nyt popupvindue. I tekst feltet *Tomcat Home* klik på folderen ude i siden, i denne skal mappen som hedder *C: / Program Files / apache-tomcat-8.5.11*. Klik ok på popupvinduet se bilag 17.
9. Derefter vælg knappen *Fix* og tryk derefter på knappen *War Exploded* se bilag 18.
10. Herefter klik ok og kørs programmet, dette vil åbne start fanen i en internet browser.

3 Timeregnskab

- *Josephine, Theodor*

Rapport: Dækker over de diagrammer og den tekst, der er og til opsætning af struktur i LaTeX. Hvem der har skrevet de forskellige afsnit eller lavet diagrammer fremgår af lige under overskrifterne i rapportens.

Database: Dækker over opsætning af database i MySQL og Java-JDBC laget i form af DAO'er og DTO'er

Webpage: Dækker over arbejde på hjemmesiden især HTML, CSS og JavaScript, men også REST i java.

Vægt: Dækker over arbejdet med WeightController og WeightConnector klasserne.

Projektplanlægning: Dækker over tidsplan og diskussion af proces.

Test: Dækker over diverse test

Time regnskabet er angivet i timer.

Opgave	Josephine	Aleksander	Søren	Jonatan	Andreas	Theodor	Sum
Rapport	24.5	0.3	8	20	21	32	105.8
Webpage	10.5	74	47	0	5	0	136.5
Database	0	0	7	20	5	2	34
Vægt	2.5	0	0	15	6.5	22	46
Projektorganisering	2	2	2	2	2	2	12
Test	4	0	0	11	8.5	6	29.5
Sum	43.5	76.3	64	68	48	64	363.8

4 Krav

4.1 Vision

- *Andreas*

En medicin virksomhed ønsker et softwaresystem, de kan bruge til afvejning af råvarer 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 virksomhedens 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.

%begincenter

4.2 Kravliste

- *Andreas og Josephine*

ID	Funktionelle krav
Must have	
M01	Systemet skal kunne administrere råvarer
M02	Systemet skal kunne administrere råvarebatches
M03	Systemet skal kunne administrere produktbatches
M04	Systemet skal kunne administrere recepter
M05	Systemet skal kunne administrere 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 operatorens 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.

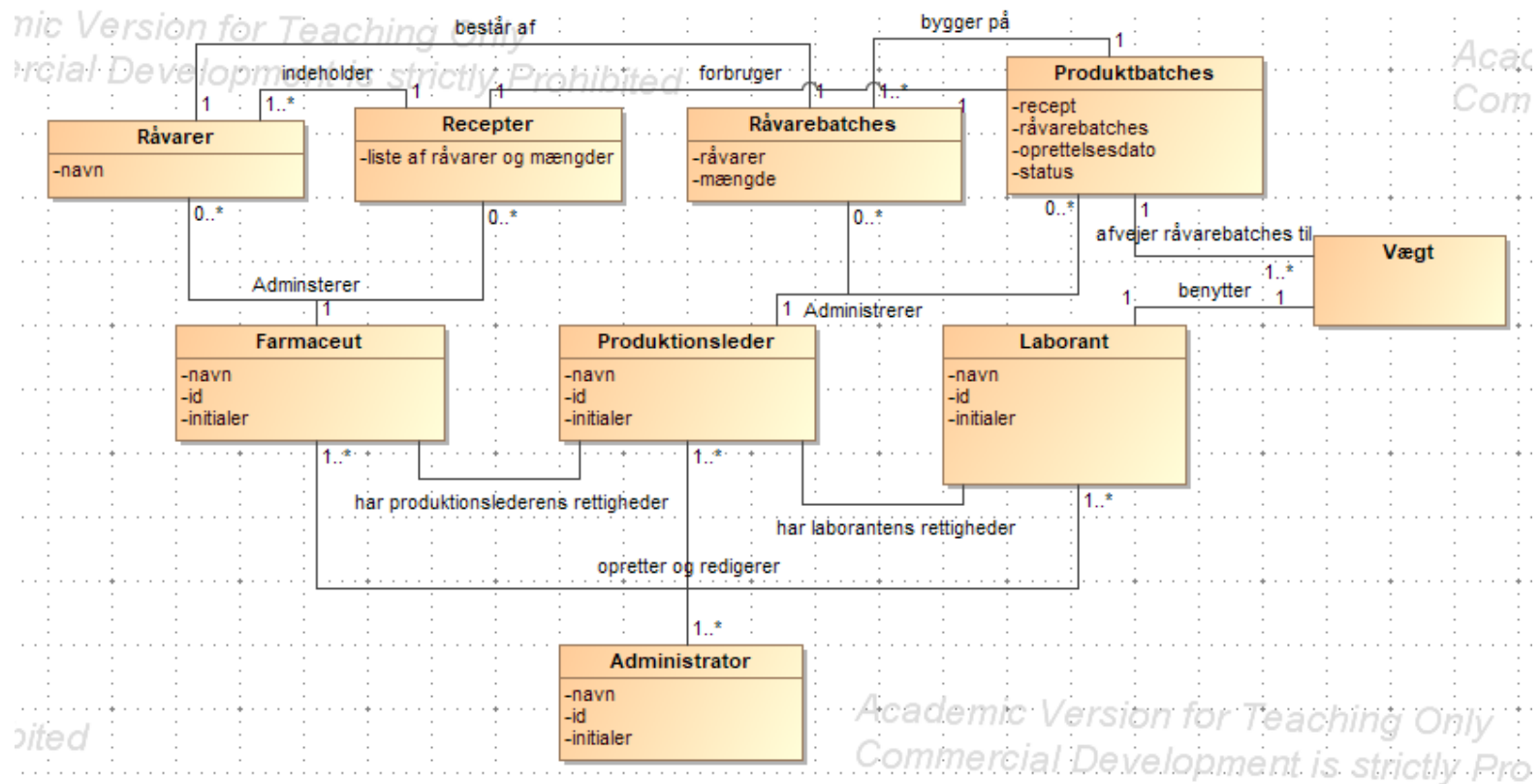
4.3 Domænemodel

- *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	Leveandører
Råvarerbatches	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

5 Analyse

5.1 Aktører

- Jonatan, Josephine

Nedenfor ses en analyse af hvilke aktører der berøres af systemet og hvordan de interagerer med systemet.

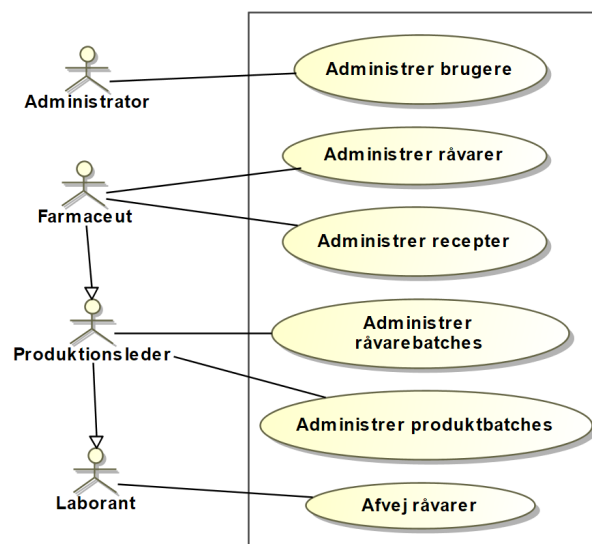
Aktørtype	Aktører
Primære	Administrator, farmaceut, produktionsleder og laborant
Supporterende	
Offstage	

5.2 Use case

- Aleksander

Use cases er taget fra opgavebeskrivelsen og er blevet gentegnet som "fully dressed" efter relevans.

Vi har i alt fire aktører, administrator, farmaceut, produktionsleder og laborant. Administratoren sørger for at oprette, redigere og deaktivere brugere i systemet. Farmaceuten kan administrere 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 - Andreas

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.

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.)

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.

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.

5.3 Casual og fully-dressed

- *Theodor*

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.

Casual
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, receiptNr, 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
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.

5.4 Traceability matrix: Use cases og krav

- *Josephine*

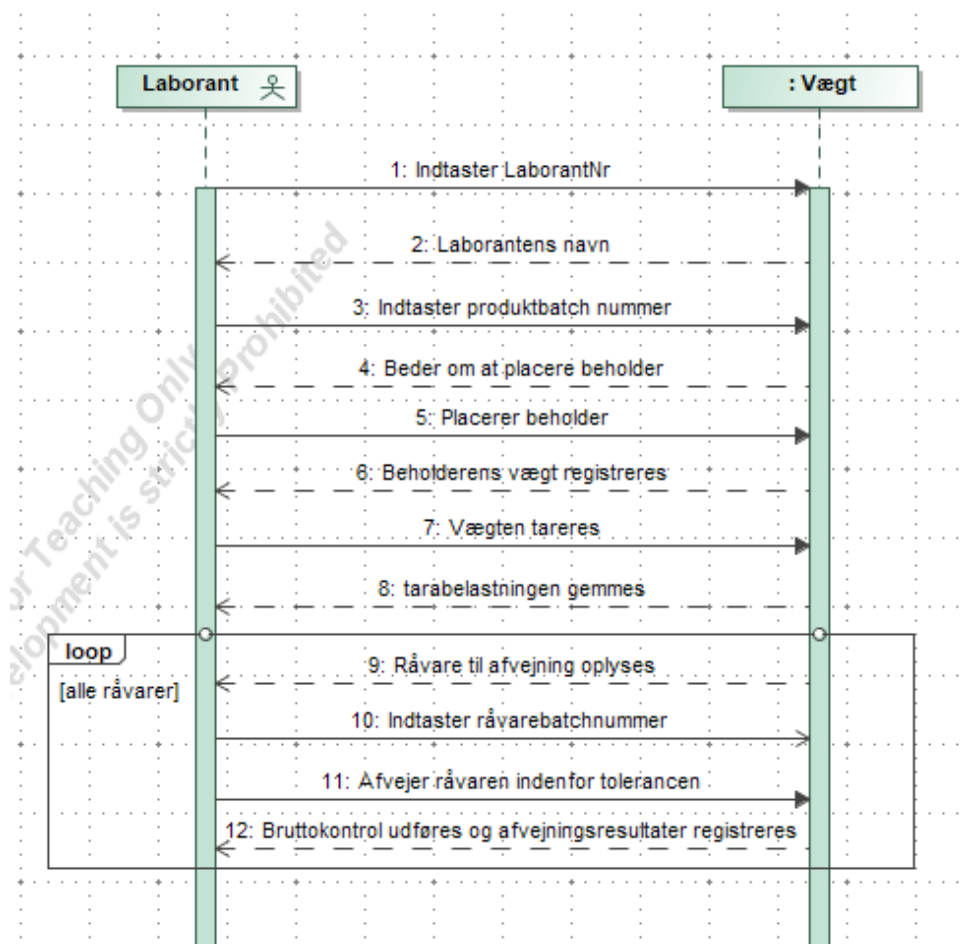
I følgende tabel ses sammenhængen mellem use cases og krav, de grønne er dem, der er implementeret, de røde er endnu ikke implementeret.

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	

5.5 System sekvensdiagram

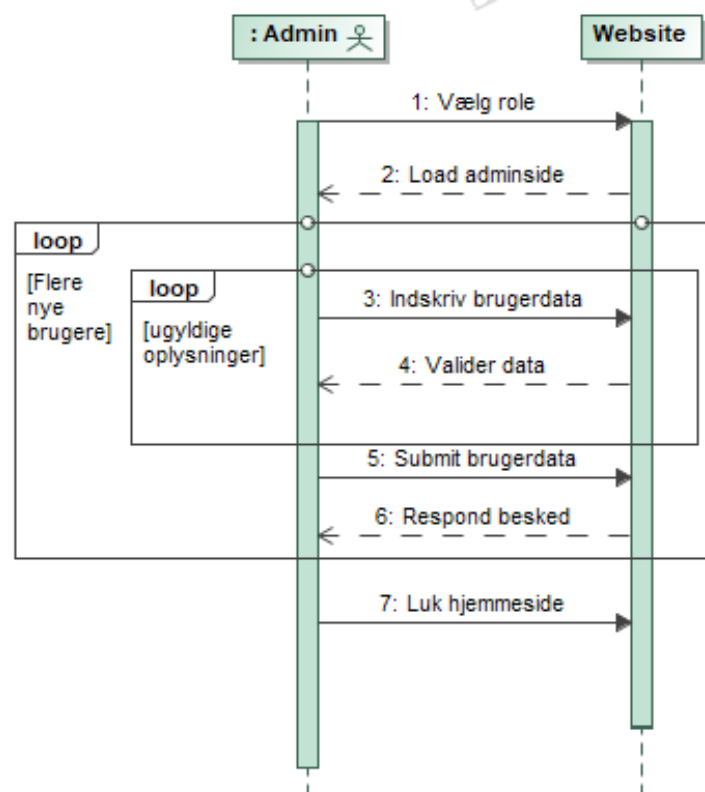
- *Andreas*

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



Figur 3: Vægt systemsekvens - Theodor

I dette systemsekvensdiagram vises en administrators interaktion med websiden. Først vælger brugeren at være en administrator, hvorefter administrator siden vises. Her indtaster administratorer 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. Administratorer lukker websiden, når de er færdige.



Figur 4: Admin opretter bruger - Jonatan og Søren

6 Design

6.1 Hjemmeside

- Theodor og Andreas

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. Vi startede med tegne, hvordan vi tænkte hjemmesiden skulle se ud, på et styk papir, for at få overblik over layoutet. Vi overvejede om vi skulle lave vores hjemmeside singlepage eller ej. Vi endte med at beslutte os for ikke at lave en singlepage applikation, da der ikke ville være et stort pres på siden, da det er et begrænset antal brugere, som vil benytte sig af siden samtidig.

6.2 Klassediagram

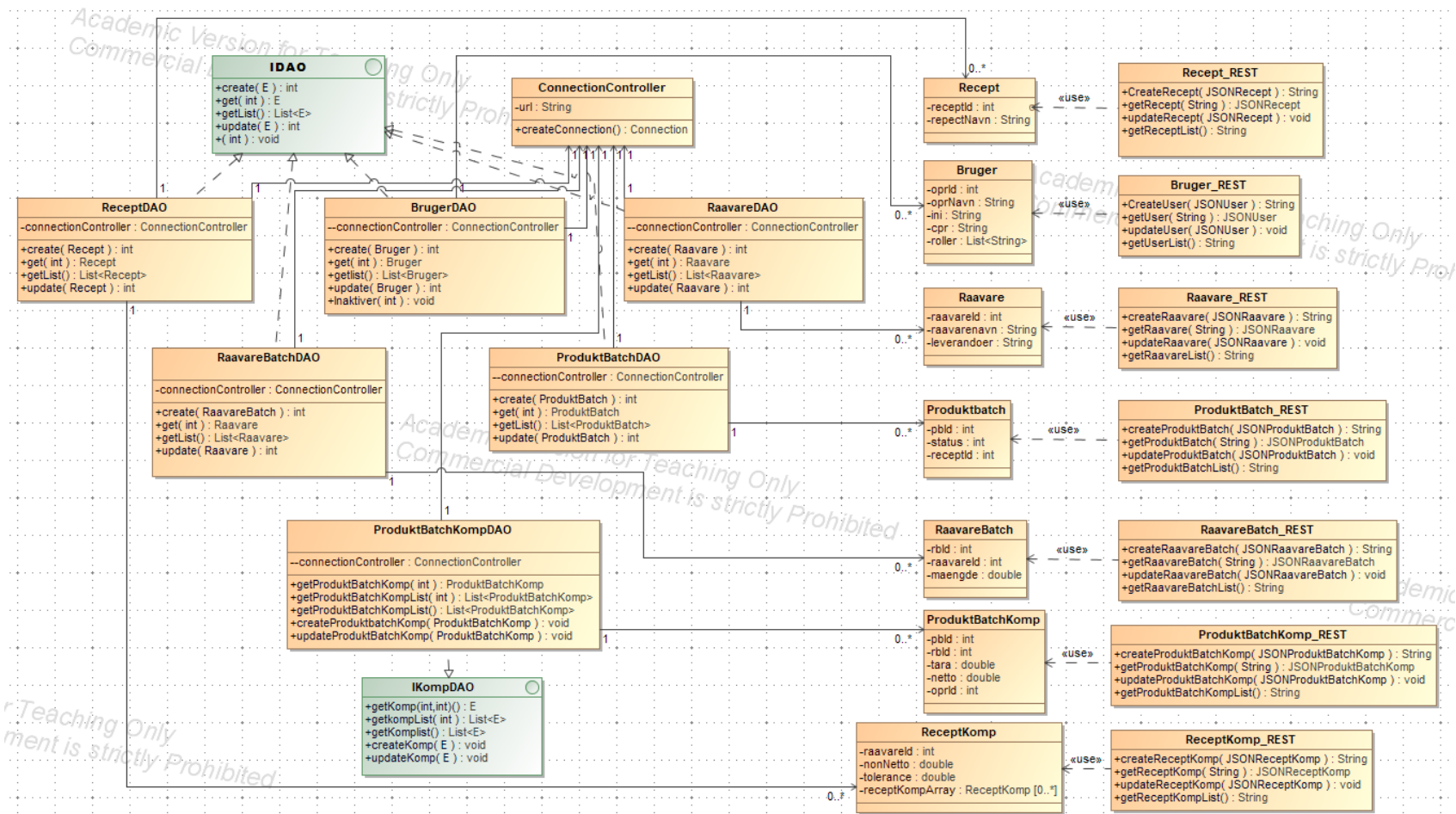
- Theodor

Design klassediagrammerne er udgangspunktet for programmerne. Design klassediagrammerne er udarbejdet ud fra domænemodellen og use cases. Vi har inkluderet attributter og metoder, såvel som klassens navn og multiplicitet i vores klassediagram. Desuden viser design klassediagrammet relationerne klasserne imellem.

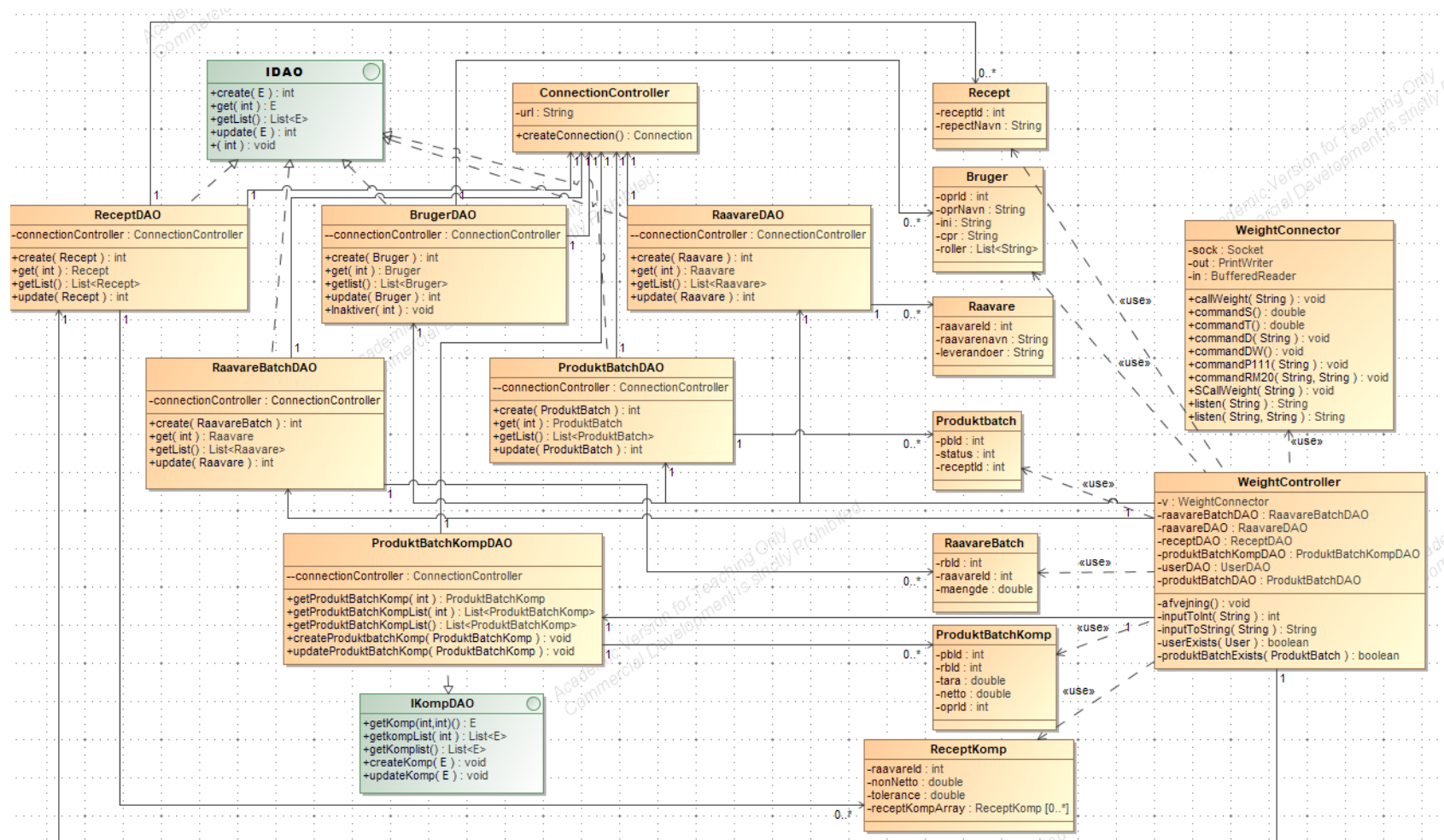
Hjemmeside programmet kan deles op i DAO'er, DTO'er, REST og nogle hjælpeklasser. DAO'erne

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.

Vægt programmet kan deles op i DAO'er, DTO'er, WeightController og WeightConnector. DAO'erne overholder interfaces hhv. IDAO og IKompDAO. DAO'erne får deres connections til databasen fra en ConnectionController klasse. DTO'erne bruges også af WeightControlleren til at indsætte data i.



Figur 5: Hjemmeside designklassediagram - Theodor

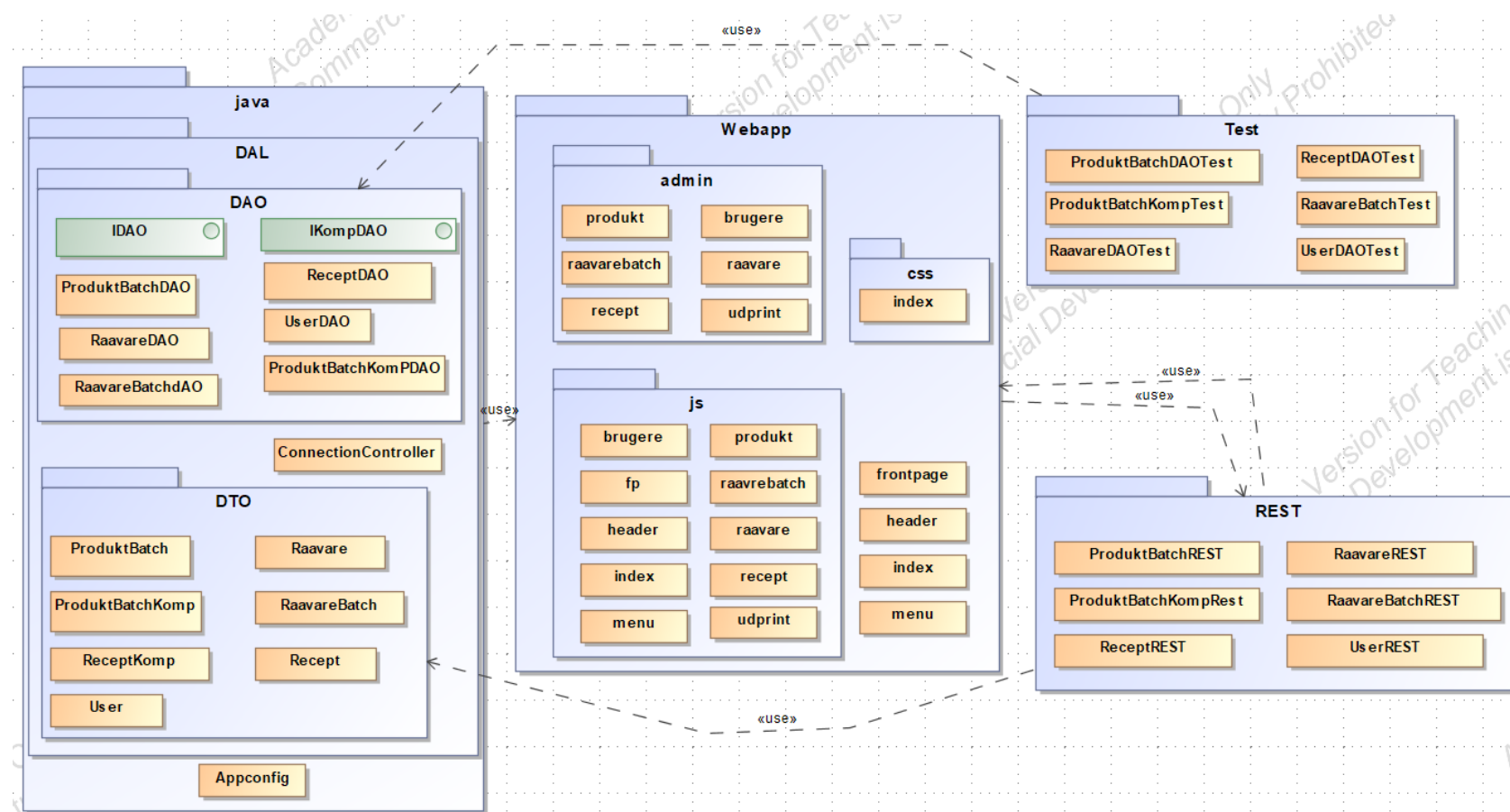


Figur 6: Vægt designklassediagram - Theodor

6.3 Pakkediagram

- Theodor og Andreas

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

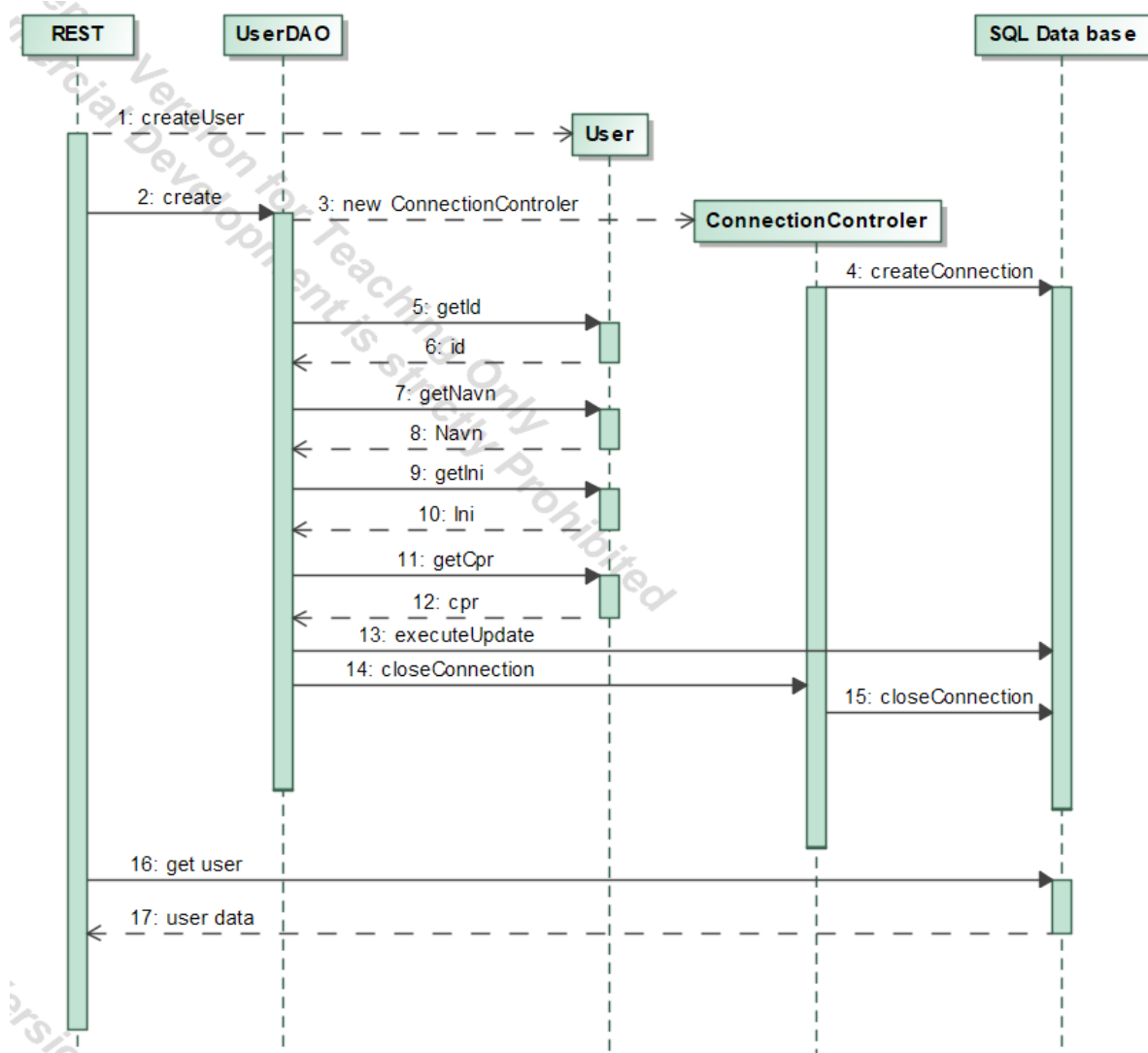


Figur 7: Pakkediagram - Andreas

6.4 Sekvensdiagram

- *Andreas*

Dette sekvensdiagram viser flowet i den del af programmet, hvor en bruger bliver oprettet. Først bliver alle indskrevne data fra hjemmesiden sendt til Java delen af programmet, hvor et User objekt bliver oprettet ud fra de indskrevne data. Herefter bliver create metoden i UserDao kaldt, der opretter et ConnectionController objekt, som så opretter en forbindelse til SQL databasen. Derefter kalder create metoden gettere, i User klassen, for id, navn, ini og cpr, som alle returnerer de forskellige data. Efter dette overføres dataen til SQL databasen. Til sidst henter hjemmesiden dataen fra SQL databasen.



Figur 8: Sekvensdiagram - Andreas

6.5 Database design

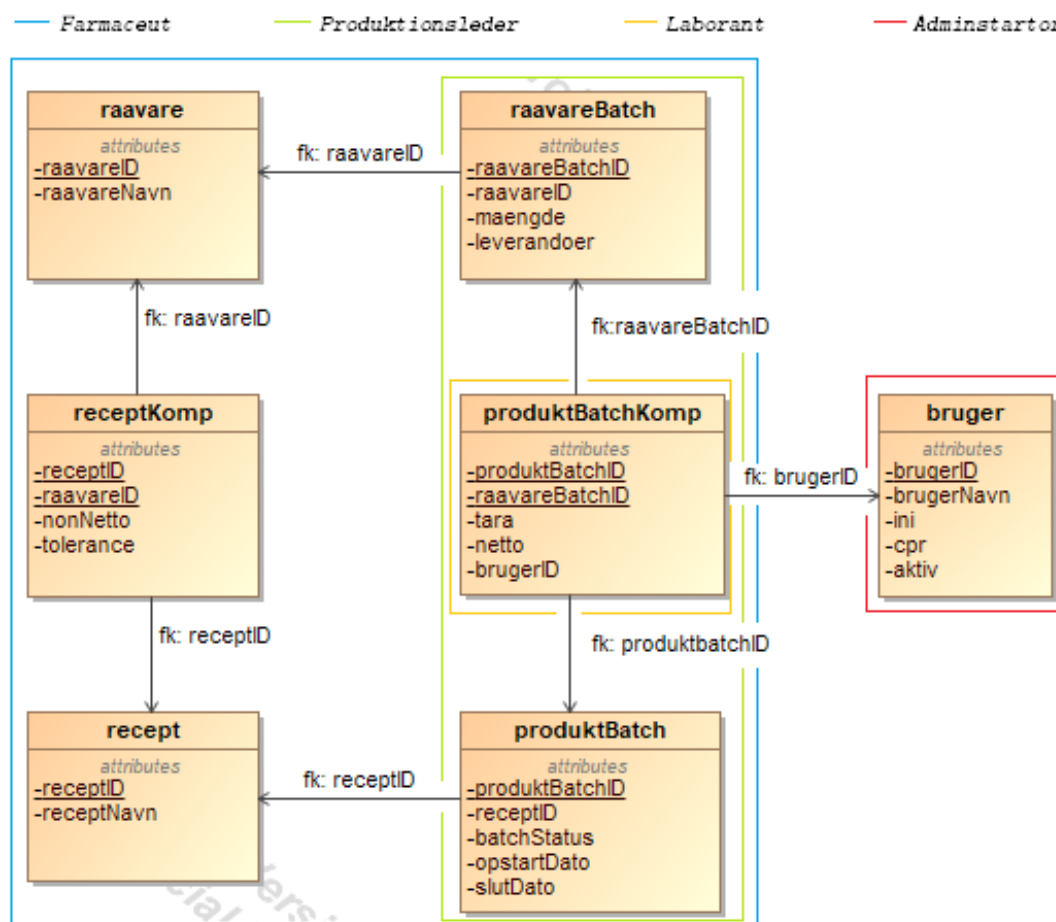
- Søren, Josephine

For at lave et fornuftigt design af databasen, er der blevet undersøgt, hvad der skal administreres. Det som skal administreres er følgende: brugere, råvarer, recepter, råvarebatches og produktbatches. Derfor er disse fem oprettet som tabeller med relevante tabelnavne. Hver recept har dog flere råvare, så der er oprettet endnu en tabel for at holde styr på det kaldet receptKomp. Det samme gælder for produktbatches og råvarebatches, så derfor er der oprettet tabellen produktBatchKomp.

Hver af de fem ting, der skal administreres, benytter deres ID som primary key. Dvs. at man ikke kan have flere råvarer med samme ID og derfor kan finde hvert råvare unikt ud fra ID. ReceptKomp benytter receptID og raavareID som dens primary key, og produktBatchKomp benytter produktBatchID og raavareBatchID som dens primary key.

I opgave beskrivelsen bliver der foreslået, at Raavare tabellen skal have en attribut som hedder "leverandør". Denne attribut har vi valgt at flytte til RaavareBatch tabellen, da man godt kan købe den samme råvare fra forskellige leverandører.

På det understående skemadiagram er der benyttet farver til at illustrere hvad der hører til hvilke roller. Hvor den blå er farmaceuter, grøn er produktionsledere, orange er laboranter og rød er adminstartorer.



Figur 9: Skemadiagram - Søren, Josephine

7 Implementering

- *Theodor*

7.1 Overblik

Vi har fulgt opgaveformuleringens forslag og lavet DTO(Data Transfer Object) og DAO(Data Access Object) klasser. DTO og DAO er i hver sine packages med tilsvarende navne, de packages ligger i en DAL(Data access layer) package. Alle DAO'erne undtagen ProduktBatchKompDAO implementerer IDAO interfacet. ProduktBatchKompDAO implementerer interfacet IKompDAO, fordi IKompDAO bruger method overloading, dvs. at interfacet indeholder to forskellige getList metoder, i modsætning til IDAO som kun indeholder en. IDAO og IKompDAO gør brug af generics, så vi kan bruge de samme interfaces til forskellige klasser. DAO'erne indeholder den kode, der skal til for at hente data fra databasen, og derefter indsætte det data i java objekter og omvendt. De metoder er create, get, getList, update og delete. DTO'erne definerer de java objekter, vi har brug for, for at modellere de objekter, der findes i medicinproduktionen.

Vi har en hjemmeside, der udover DAL package består af en REST package med REST klasser, for hver DTO klasse. Vores løsning bruger en persistent database i form af s185103 MySQL database. Vi har en test package med DAO tests. Der er en webapp mappe med HTML filer, CSS filer og JavaScript filer, der udgør vores hjemmeside.

Vi har også et afvejningsprogram, der udover DAL består af en weight package. I denne weight package ligger der to java klasser, som sammen kan stå for alt kommunikation fra og til vægten.

I vores implementering har vi to af alle DAO'er, fordi både webpage og vægten har deres egen kopi. På grund af dette så når DAO'erne blev opdateret i et projekt, blev de ikke opdateret i det andet. Konsekvensen af det er at vi har to forskellige sæt af DAO'er, som fungerer på lidt forskellige måder. Måden vi ville have fikset dette, var ved REST så vi kunne java klasser til begge programmer. Dette ville gøre at vægt programmet direkte ville gå ind i det andet projekt, og kører dens klasser. Dog fik vi ikke implementeret dette.

7.1.1 Hjemmeside

På hjemmesiden loades alle html filer ind i index.html når de skal bruges. Vi har brugt CSS til at ændre html grafiske udtryk og øge brugervenligheden. Vi har brugt JavaScript til at lave funktionalitet og få den rette data ind på siden og ind i databasen.

Hjemmesiden er opbygget, så man først bliver præsenteret for 3 roller: administrator, farmaceut og produktionsleder, man vælger så en.

Administratorer kommer ind på en side med en form til oprettelse af brugere og en tabel med en oversigt over brugere, hvis man klikker på en brugers navn i tabellen kan man redigere brugeren.

Farmaceuter har fire muligheder i den menu som er i toppen af hjemmesiden: råvarer, recepter, råvarebatches og produktbatches.

Vælger man råvarer, er der en form til oprettelse af råvarer, her kan man indskrive den nye råvares id og navn, samt se en liste over alle råvarer med deres id og navne.

Vælger man recepter, er der en form til oprettelse af råvarer, her kan man igen vælge id og navn. Ne-deunder oprettelses tabellen er der nogle linjer, her kan man fjerne og tilføje råvare til en recept. Også på denne hjemmeside, er der en tabel som viser alle recepter, trykker man på en recept, kommer der en anden tabel frem, som viser alle råvarer i den recept.

Vælger man råvarebatch, er der en form til oprettelse af råvarer med 4 felter: id, en dropdown menu med

råvare id, mængde og leverandørnavn og en tabel med de oprettede råvarebatches.

Vælger man produktbatch, er der en form til oprettelse af produktbatches, hvor der er et felt, en dropdown menu med recepter og en tabel med de oprettede produktbatches med id, recept, status, oprettelsesdato og afsluttelsesdato. Hvis man klikker på et produktbatch kommer der en ny tabel frem med produktbatchkomponenter, der viser råvare, mængde, tolerance, tara og netto.

Produktionslederen er som farmaceut, men har ikke adgang til råvarer og recepter.

På brugersiden er der i brugeropretningsmenuen gjort brug af regular expressions i JavaScript, for at forhindre, at man kan indtaste f.eks. bogstaver i sit cpr-nummer eller tal i sit navn. Derudover kræver siden også, at alle felter er udfyldt, for at man ikke kan indsætte ukomplet data i databasen. Man kan yderligere redigere brugere ved at klikke på den relevante bruger i brugertabellen, så kan man ændre alt undtagen brugers ID.

7.1.2 Afvejning

Afvejningsprogrammet består, ligesom hjemmesiden, af en DAL pakke med DTO'er og DAO'er, der er identiske med hjemmesidens DTO'er og DAO'er. Forskellen er en weight package, hvori der er en WeightConnector klasse og en WeightController klasse. WeightConnector klassen har en konstruktør der opretter en socket til localhost eller vægten på port 8000. Den indeholder metoden callWeight, der laver PrintWriter, som kommunikerer med vægten og en BufferedReader, som modtager en inputStream fra vægten. Derudover udskriver den kommandoerne i konsollen. Fordi vægten ikke er som simulatoren og laver overflødig kommunikation, har vi lavet en listen metode, der benytter et while loop og et if-statement med en contains metode til at finde den String, der skal bruges. Der er også en modificeret callWeight metode kaldet SCallWeight. Den findes fordi commandS() bruger den, da den skal returnere en double og dermed skal bruge input fra vægten, så hvor der i callWeight() er en in.readLine() er den flyttet ned i commandS(). Resten af klassen består af metoder, som bruger callWeight til at udføre en af kommandoerne beskrevet i:

https://docs.google.com/presentation/d/1Z0ELvdqur66q1fYCKN6YDwwqqr2ISwxB5B_7iL9zVYM/edit

Den anden klasse er WeightController, der indeholder den store metode afvejning(). Derudover er der 4 hjælpemetoder. 2 af dem laver vægtens kommunikation om til brugbare variable og de andre 2, tjekker om en DTO findes i databasen. Afvejning() deklarerer alle simple variable og objekter i toppen af metoden. Det første der sker i metoden er at brugeren, bliver bedt om at indtaste sit ID, så tjekkes det om ID'et matcher med et ID i databasen. Så findes navnet, der passer med ID'et frem. Så skal laboranten be- eller afkræfte om det er det rette navn. Hvis ikke starter hele processen om igen, ved hjælp af et do while loop. Samme struktur går igen for produktBatchID'et. ProduktBatchID'et tjekkes for om det er under produktion eller afsluttet. Og brugeren får besked om det. Så findes den navnet på den recept, der skal produceres. Så udregnes hvilken receptKomponent, der skal afvejes først. Og selve afvejningen går i gang. Først skal beholderen placeres, så skal råvaren hentes og råvarebatchID'et indtastes og så skal råvaren afvejes. Så udføres bruttokontrol. Hvis det går noget galt i den proces, starter afvejningsprocessen om. Processen gentages for alle receptKomponenter og til sidst kan man afslutte programmet eller afveje et nyt produktBatch.

8 Test

8.1 Positiv test

- Jonatan, Søren

Disse positive test dækker over alle funktionelle krav i systemet, og viser hvorvidt vi mener at kravene er bestået. Alle krav bliver vist her på nær vores won't have krav, da vi vidste fra starten af, at dette var noget vi ikke ville implementere. Vi kom igennem alle krav inklusiv should have og could have kravene. Vi havde fra starten af projektet forventet, at nogle af disse krav ikke ville være inde for vores rækkevidde.

Funktionelle krav	Positiv test	Bestået
M01	Systemet skal kunne administrere råvarer	Bestået
M02	Systemet skal kunne administrere råvarebatches	Bestået
M03	Systemet skal kunne administrere produktbatches	Bestået
M04	Systemet skal kunne administrere recepter	Bestået
M05	Systemet skal kunne administrere bruger	Bestået
M06	Systemet skal ikke kunne slette bruger men skal markerer dem som inaktive	Bestået
M07	Systemet skal kunne afveje råvarer gennem en vejeterminal	Bestået
M08	Systemet skal kun tillade de rigtige roller adgang til de rigtige funktioner	Bestået
M09	Systemet skal kunne vise en brugerliste	Bestået
M10	Systemet skal kræve at brugeren angiver rolle på forsiden	Bestået
M11	Systemet skal garanterer at råvare, råvarerbatch, recepter, produktbatches og brugere alle har unikke ID	Bestået
M12	Systemet skal holde styr på produktbatches status (Oprette/Under produktion/Afsluttet)	Bestået
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	Bestået
M14	Vægten skal kunne bruge kommandoerne S, T, D, DW, p111, RM208	Bestået
M15	Vægten skal kunne ud fra operatørnummre finde operatorens navn	Bestået
S01	Systemet skal holde styr på den aktuelle mængde råvare der er blevet brugt i et givent produktbatch	Bestået
S02	Systemet skal kunne holde styr på tilbageværende mængde af råvarerbatch	Bestået
S03	Systemet skal sørger for at recepter har tolerance på alle råvarer	Bestået
C01	Systemet skal holde styr på produktbatches oprettelses dato	Bestået
C02	Systemet skal lave en produktionsforskrift som kan printes	Bestået

8.2 Negative test

Vægt

- Theodor og Andreas

I de negativetest til vægten prøvede vi at få programmet til at crashe, ved at trykke på cancel i stedet for ok, og ok mens der ikke var indtastede noget data. Vi fandt ud af, at hvis man trykkede på cancel i stedet for ok, ville man få at vide, at man skal prøve igen. Hvis man trykkede ok uden, at der var indtastede noget, crashede programmet. Dette fiksede vi ved at se om programmet returnede null og hvis det gjorde ændrede vi det til -99, som der ikke er nogen, som har det ID i databasen. Dette fik programmet til at bede brugeren om at prøve igen, i stedet for at crashe.

Vi testede også om de do while vi havde implementeret virkede, som vi troede og det gjaldt for fejlinput i produktBatchID og for produktbatchets status.

Vi testede om et afsluttet produktbatch kunne produceres, det kunne det ikke.

8.3 Unit-test

- Theodor, Jonatan

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.

Vi har lavet forskellige unit-tests, blandt andet en for hver DAO klasse, for at teste create, get, getlist, update og delete metoderne. Vi har valgt hovedsageligt at fokusere vores Unittest på DAO'erne, fordi de er en så stor drivende faktor i alle dele af programmet. Det er også denne del af programmet, som nemmest kan komme til at sende forkert data frem og tilbage.

Test Case ID	Beskrivelse	Status
TC01	Test af UserDAO	Bestået
TC02	Test af ReceptDAO	Bestået
TC03	Test af RaavareDAO	Bestået
TC04	Test af RaavareBatchDAO	Bestået
TC05	Test af ProduktBatchKompDAO	Bestået
TC06	Test af ProduktBatchDAO	Bestået

8.4 Coverage test

Vægt

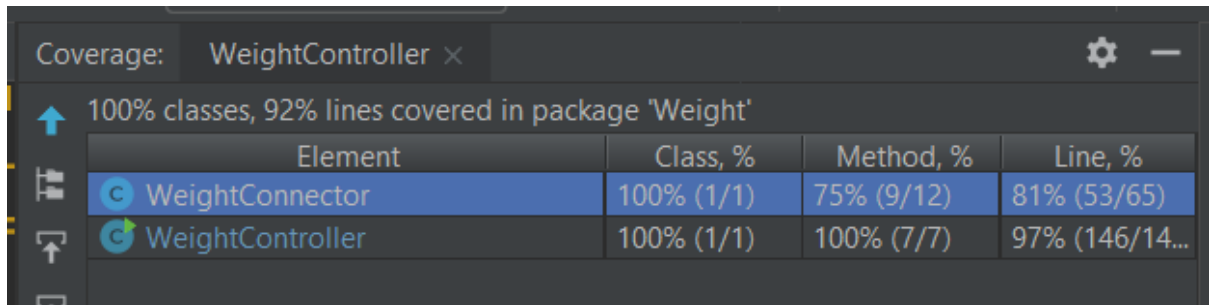
- Theodor

En coverage afdækker, hvor mange procent af koden, der bliver kørt, på flere niveauer. Den tester om klassen bliver brugt, hvor mange af metoderne i en klasse bliver brugt og hvor mange af linjerne der bliver kørt.

Vi har kørt coverage test på weight package. Begge klasser bliver kørt. Alle metoderne i WeightController bliver kørt og 97% af linjerne bliver kørt, de sidste 3% har at gøre med, at et råvarebatch ikke

har nok tilbage til at producere produktet.

I WeightConnector klassen er der tre metoder der ikke bliver brugt og det er kommandoer til vægten, som opgaven siger vi skal have med, men vi ikke bruger. 81% af linjerne bliver kørt.



The screenshot shows the 'Coverage' window in IntelliJ IDEA for the package 'Weight'. It displays a table with coverage data for two classes: WeightConnector and WeightController. The table has four columns: Element, Class, %, Method, %, and Line, %. The WeightConnector class shows 100% class coverage (1/1), 75% method coverage (9/12), and 81% line coverage (53/65). The WeightController class shows 100% class coverage (1/1), 100% method coverage (7/7), and 97% line coverage (146/14...).

Element	Class, %	Method, %	Line, %
WeightConnector	100% (1/1)	75% (9/12)	81% (53/65)
WeightController	100% (1/1)	100% (7/7)	97% (146/14...)

Figur 10: coverage test - Andreas

9 Projektplanlægning

Planlagt forløb

- *Josephine og Theodor*

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 starten 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. Desuden vil vi undgå solokodning (at et gruppemedlem bruger dag og nat og får skrevet en masse kode i et hug).

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 i dem. Lave kravliste med færre "must have" krav end tidligere. Ud fra krav opstilles test tidligt, så vi løbende kan teste vores program.

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

Design klassediagram 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. System sekvensdiagram 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, JavaScript.

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

Reelt forløb

- *Jonatan og Andreas*

Starten af vores projekt gik præcis som planlagt, vi startede med at lave analyse delen og derefter design delen. I den periode begyndte folk at flytte fokus fra rapporten, og koden fik højere og højere prioritet. Med koden begyndte nogle at starte med frontend primært Aleksander, mens resten startede med database kodning og lave java objekterne. Før arbejdet på databasen og backend, blev der lagt god tid i arkitekturen, for at undgå fejl og store ændringer senere i processen. Da DTO'erne og DAO'erne var begyndt at tage form, så startede vi allerede der, med at lave UnitTest til DAO'erne. I perioden her, fik vi også sat hele databasen op med tabeller, variable, keys og constraints. Ved dette punkt kom Milestone 1 mødet. Efter dette møde fik vi en bedre ide om, hvor projektet stod, og hvor godt med vi var.

Efter dette møde har vi fået arbejdet videre med frontenden, og vi har dermed fået nogle af hjemmesiderne, til at kunne snakke med java koden, og dermed også databasen. Vi har fokuseret på brugervenlighed og overskuelighed, og havde en grundlæggende ide om, hvordan hjemmesiderne skulle se ud, før vi startede med at lave dem. Derudover har vi fået lavet UnitTest til alle vores DAO'er, så vi ved, at linket mellem java og databasen, ikke kan give os nogle fejl længere henne.

Efterfølgende har vi arbejdet på at få vægtcontrolleren til at virke, som vi til, at starte med, fik til at virke med den virtuelle vægt. Derudover har vi fået de fleste af siderne på hjemmesiden til at snakke sammen med databasen og indsætte data på den. Vi har også fået lavet et nyt design klassediagram, et sekvensdiagram og fået lavet nogle positive test

Ved milestone 2 manglede vi nogle få hjemmesider, og vi manglede at få lavet vægten helt færdig. Vores vægt program fik vi til at kunne virke, på den rigtige vægt, efter vi havde været til mødet. Derudover blev den sidste del af hjemmesiden prioriteret meget højt, så vi havde en fuld funktionel frontend. På nær frontenden var alle andre dele af programmet færdig arbejdet, og manglede kun nogle negative test og positive test, til at tjekke, hvor mange krav vi havde fået udført, og hvor hårdført vores program er. I de sidste dage af projektførløbet formåede vi at færdig gøre frontenden og vægtcontrolleren. Vi fik testet begge dele af systemet og alle positive test bestod. Vi fik også set på de sidste dele af rapporten, og fik læst rapporten igennem.

10 Konklusion

- *Jonatan*

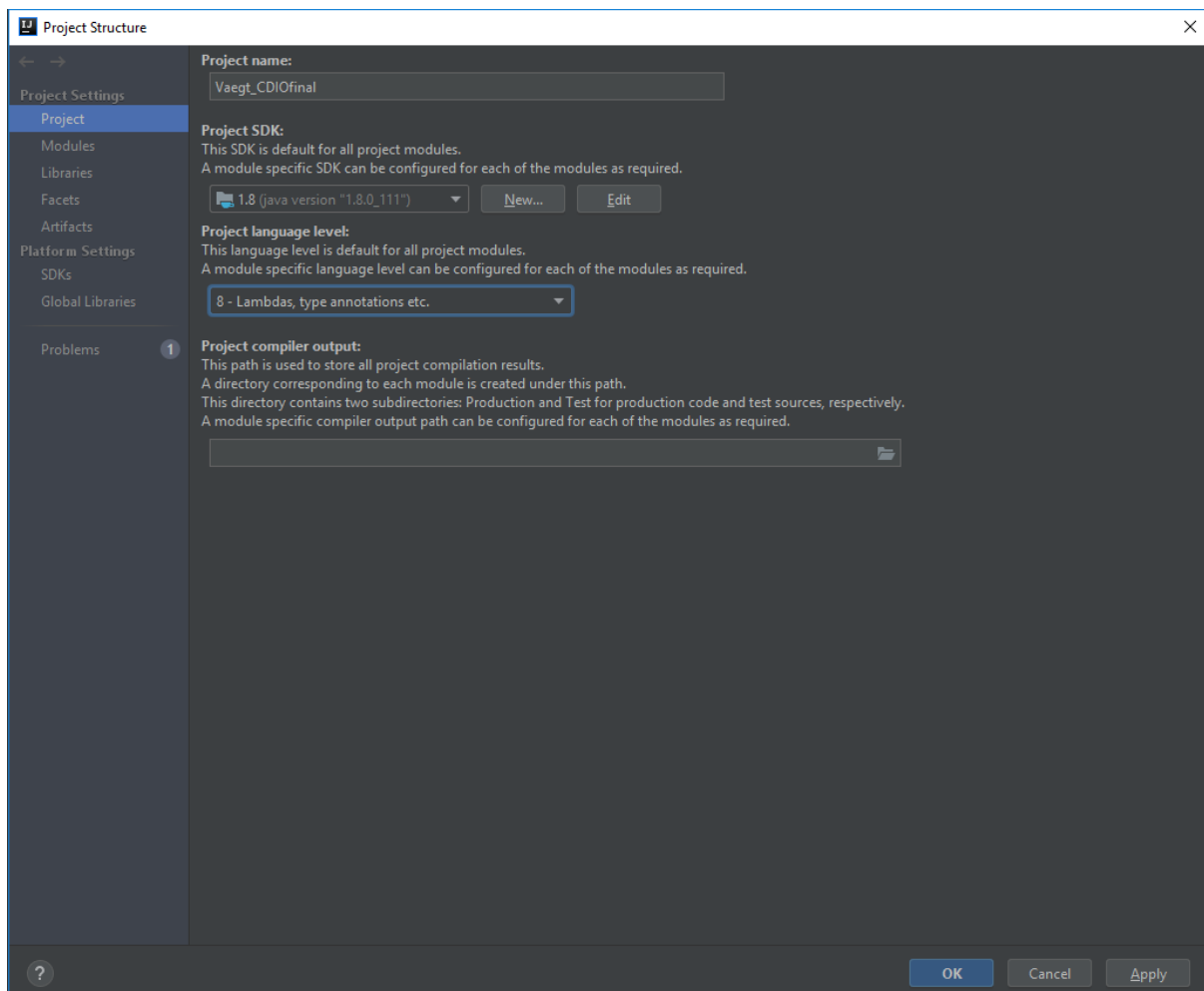
Vores proces har i dette CDIO projekt været overvejende god, vi havde lagt en plan i starten, og vi kunne holde os til planen. Dog var der dele, som trak mere ud end først forventet, dette inkluderede f.eks. frontenden og vægten. På den anden side tog database opsætning og database implementeringen mindre tid end forventet. Med dette blev projektet færdigt mere eller mindre på den forventede tid, og vi nåede at implementere alle dele, vi ville implementere. I modsætning til forrige CDIO projekter, hvor blandt andet sygdom har sat processen bagud i forhold til planen, så har dette projekt været mere smertefrit. Her til sidst kan vi konkludere, at have en plan fra dag til dag, giver markant mere overskud og struktur gennem hele projektet.

Selve produktet vi stod med i sidste ende, er vi blevet godt tilfredse med, og vi har nået de krav vi forventede, vi kunne nå. Programmet har som forventet tre større dele i sig, disse er databasen, vægt-delen og frontenden i form af websiden. Databasen var delen, vi først fik færdigt, dette inkluderer DAO'erne, som snakker med databasen. Vi lavede den først, da de resterende dele af programmet, begge to bruger kommunikation til databasen. Den næste del, vi blev færdige med var vægten, her fik vi lavede et program, som kunne klare de krav vi havde til den, men kun virkede når vi brugte simulatoren. Derefter begyndte vi at bruge den rigtige vægt, og fik programmet til at fungere på den også.

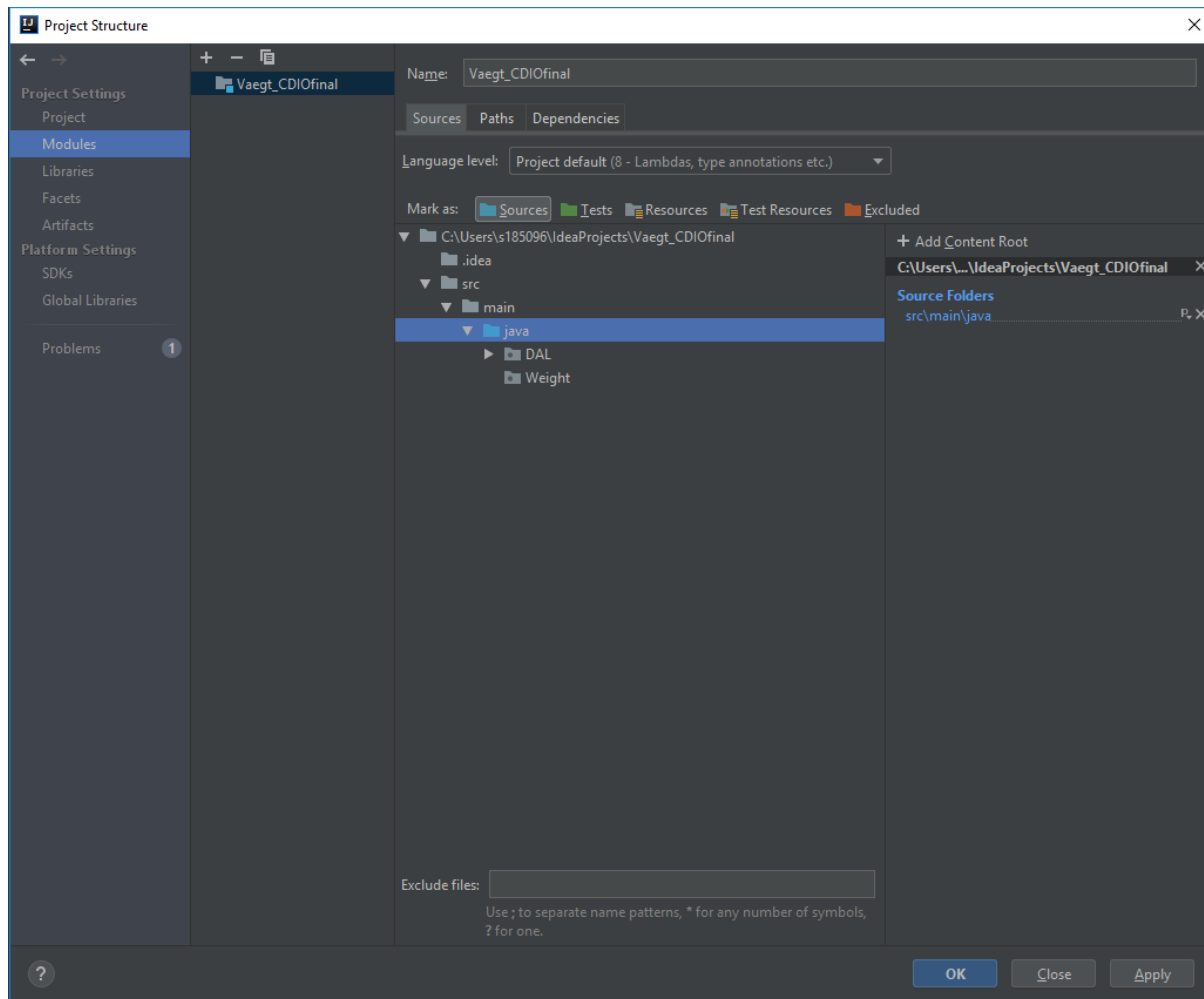
Den del, som var sværest og krævede mest tid var websiden, men i enden fik vi lavet et program, som kunne illustrere data forståeligt, og hentede den data fra databasen. Under hele udviklingen af websiden, lagde vi stor vægt på brugervenlighed.

Sammenligner man dette CDIO forløb i forhold til de forrige CDIO projekter i dette kursus, så har dette projekt været meget større, og haft mange flere dele i sig. I de forrige projekter har det været svært, at finde nok arbejde så hele gruppen, havde noget at arbejde på. Denne gang har det ikke været et problem, da der har været massere af individuelle ting, som forskellige mennesker har kunne arbejdet på samtidigt. Det, at vi har kunne siddet en hel gruppe, og arbejdet på den alle sammen samtidigt, kunne nemt havde gjort projektet mere uoverskueligt og roddet. Men fordi vi kommunikerede med hinanden, så alle vidste, hvad der blev arbejdet på, så havde vi overskud over, hvilke dele der stadigvæk manglede i projektet. I forrige CDIO projekter har vi haft problemer med blandt andet sygdom, dette har dog ikke været et problem i denne opgave, og vi har i klart de fleste af dagene været fuld hold.

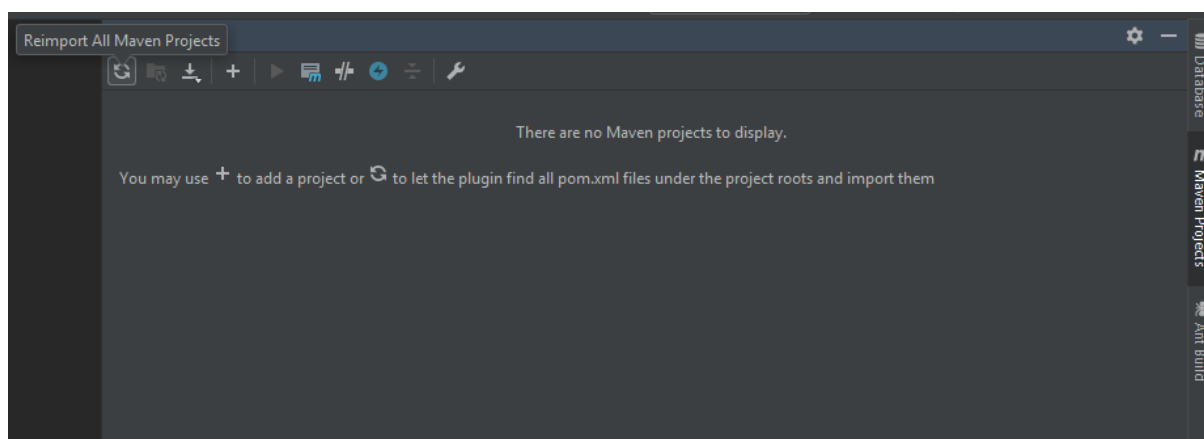
11 Bilag



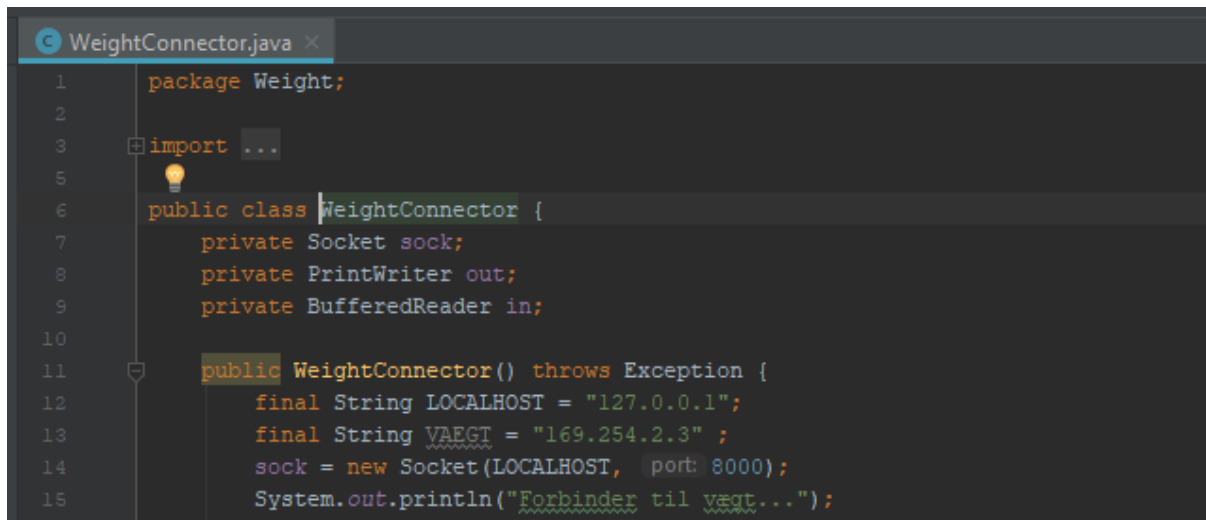
Figur 11: JDK



Figur 12: Source

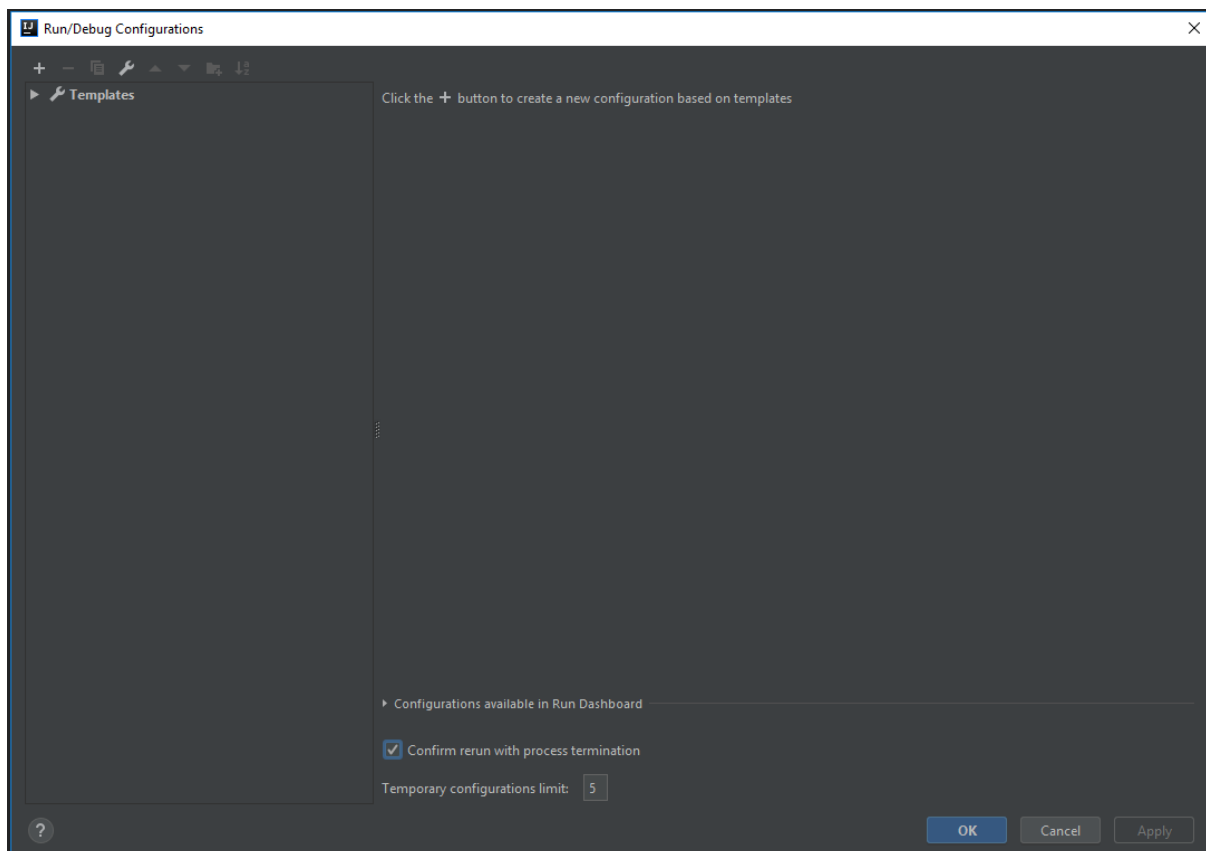


Figur 13: Maven

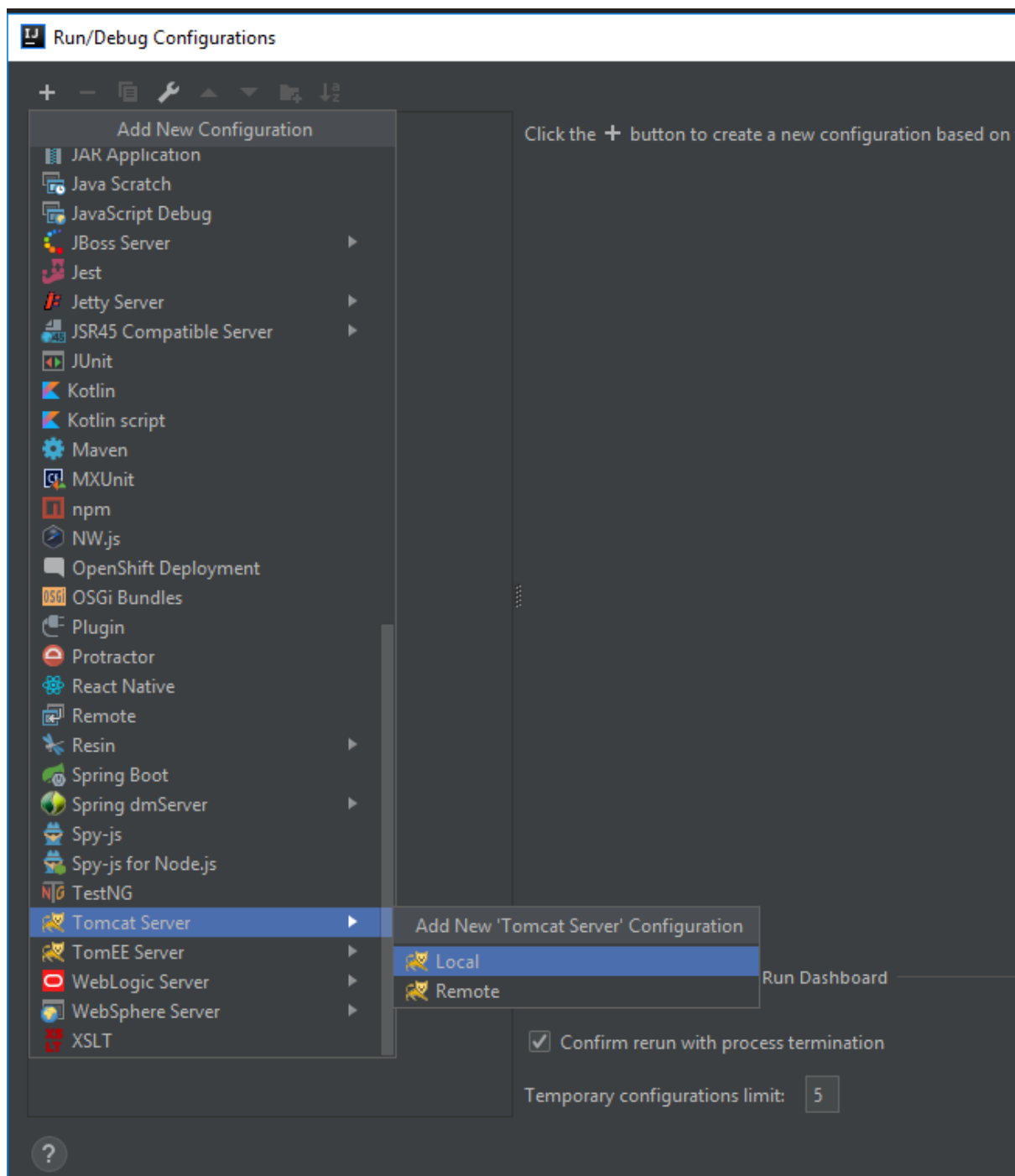


```
1 package Weight;
2
3 import java.net.Socket;
4 import java.io.PrintWriter;
5 import java.io.BufferedReader;
6
7 public class WeightConnector {
8     private Socket sock;
9     private PrintWriter out;
10    private BufferedReader in;
11
12    public WeightConnector() throws Exception {
13        final String LOCALHOST = "127.0.0.1";
14        final String VAEGT = "169.254.2.3";
15        sock = new Socket(LOCALHOST, 8000);
16        System.out.println("Forbinder til vægt...");
17    }
18 }
```

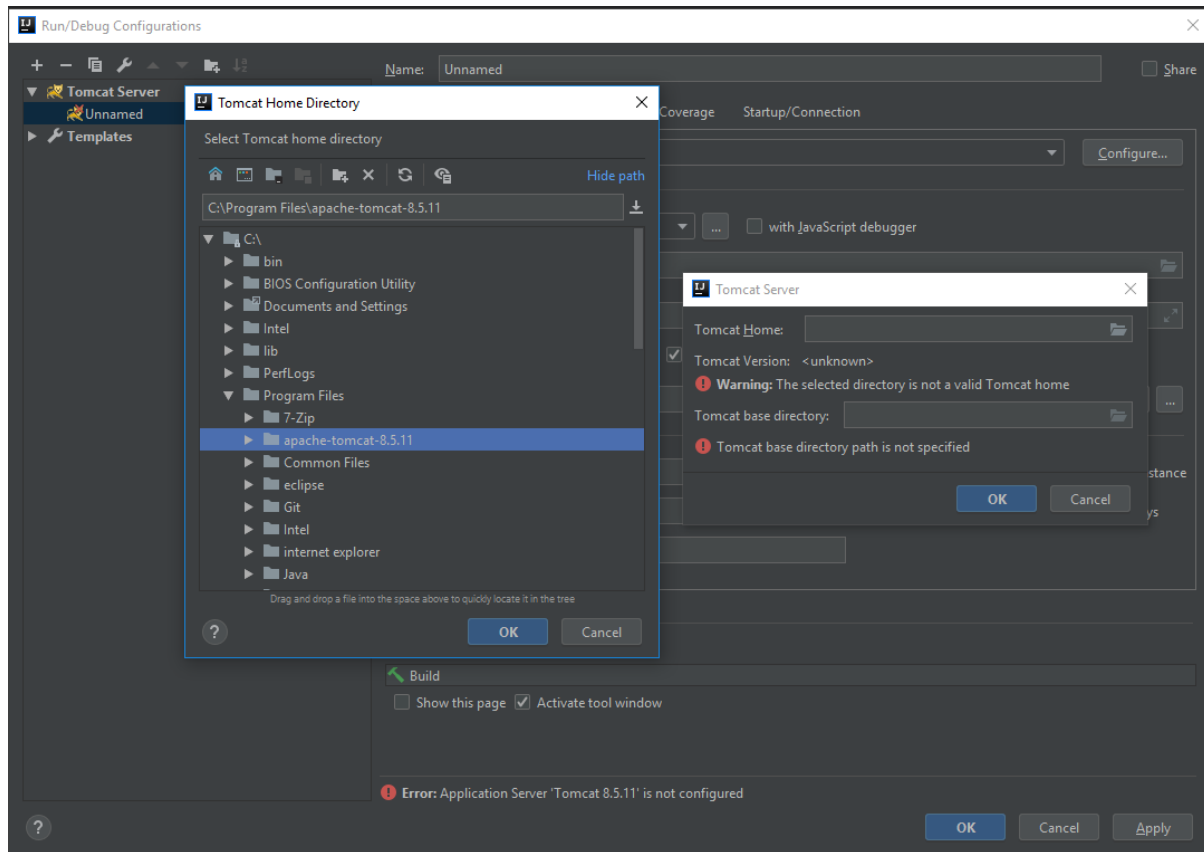
Figur 14: IP



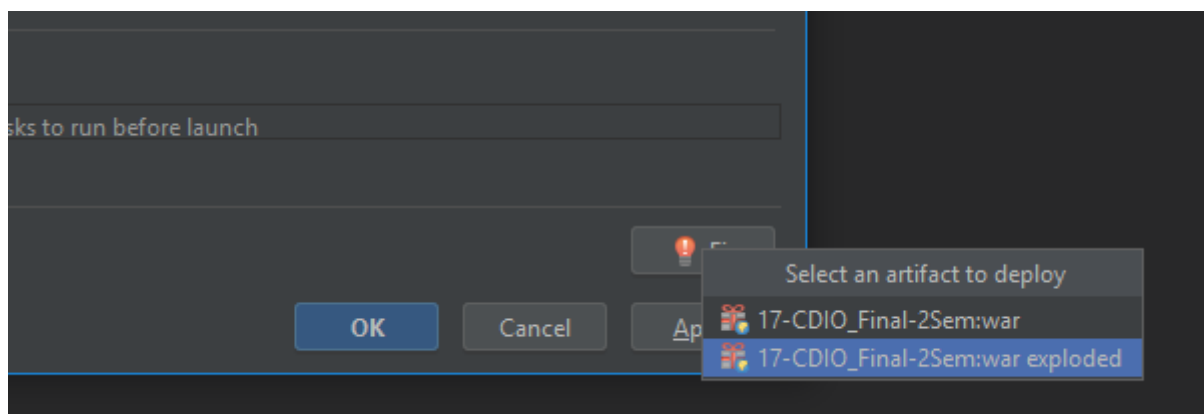
Figur 15: Add configurations



Figur 16: Tomcat Local



Figur 17: Tomcat dir



Figur 18: War Exploded