Projekt:

Olsker Cupcake webshop

Gruppemedlemmer:
Michael Xu / cph-mx@cphbusniss.dk / Michaelxuw / Hold A
Muneeb Ashraf / cph-ma670@cphbusniss.dk / MuneebAshraf / Hold A
Nicolai Andersen / cph-na203@cphbusniss.dk / Legolai / Hold A

Opgave udleveret:
Mandag den 4. april 2022 kl. 9

Rapport påbegyndt:
Tirsdag den 19. april 2022 kl. 9

Afleveringsdato:
Torsdag den 21. april 2022 kl. 12

Indholdsfortegnelse

Indledning

Dette er en rapport om 'Cupcake-projekt' lavet i 2. semester i Datamatiker linjen på CPHBusiness. Projektets opgave er at lave en kørende webside inklusiv server og database, ud fra en 'kundes' krav. I dette tilfælde en lille bager, der kun sælger cupcakes, som gerne vil have en webshop.

Vi har i anledningen til dette projekt også blevet udleveret en startkode, som vi skulle starte som udgangspunkt i og videreudvikle fra. Vi har derudover også brugt programmer som Draw.io og Figma for mockups og selve programmet er udviklet i Java, MySql, HTML, CSS, Twitter Bootstrap og JavaScript som alt kører på en Tomcat webcontainer.

Baggrund

Virksomheden hedder 'Olsker Cupcakes' og er et økologisk iværksætteri fra Bornholm. Selve virksomheden sælger kun cupcakes, men kunder kan vælge mellem et varieret valg af cupcake bunde og cupcake toppings.

De krav som Olsker har er i bredde term: "En webshop hvor en kunde kan se produkter (cupcakes toppe og bunde), vælge et sortiment, og så bestille og betale for dem. Derudover skal det være muligt at oprette og logge på som enten bruger eller administrator, og der skal være ekstra funktionalitet for administratorer".

Teknologivalg

Sprog: Java 17
Build tool: Maven 3.8.1
Dependencies:

- Hamcrest 2.2
- HikariCP 5.0.1
- JUnit jupiter API 5.8.1
- JUnit jupiter Engine 5.8.1
- Java servlet API 4.0.1
- Logback Classic Module 1.2.11
- Logback Core Module 1.2.11
- MySQL Connector/J 8.0.28
- SLF4J API Module 1.7.36
- JSTL 1.2
- p6spy 3.9.1

Database: MySQL 8.0.27-arm64 og MySQL 8.0.27
IDE: Intellij 2022.1
VCS: Github - GIT 2.35.1
Server: Tomcat 9.0.60
CSS Framework: Twitter Bootstrap 5.1.3

Krav

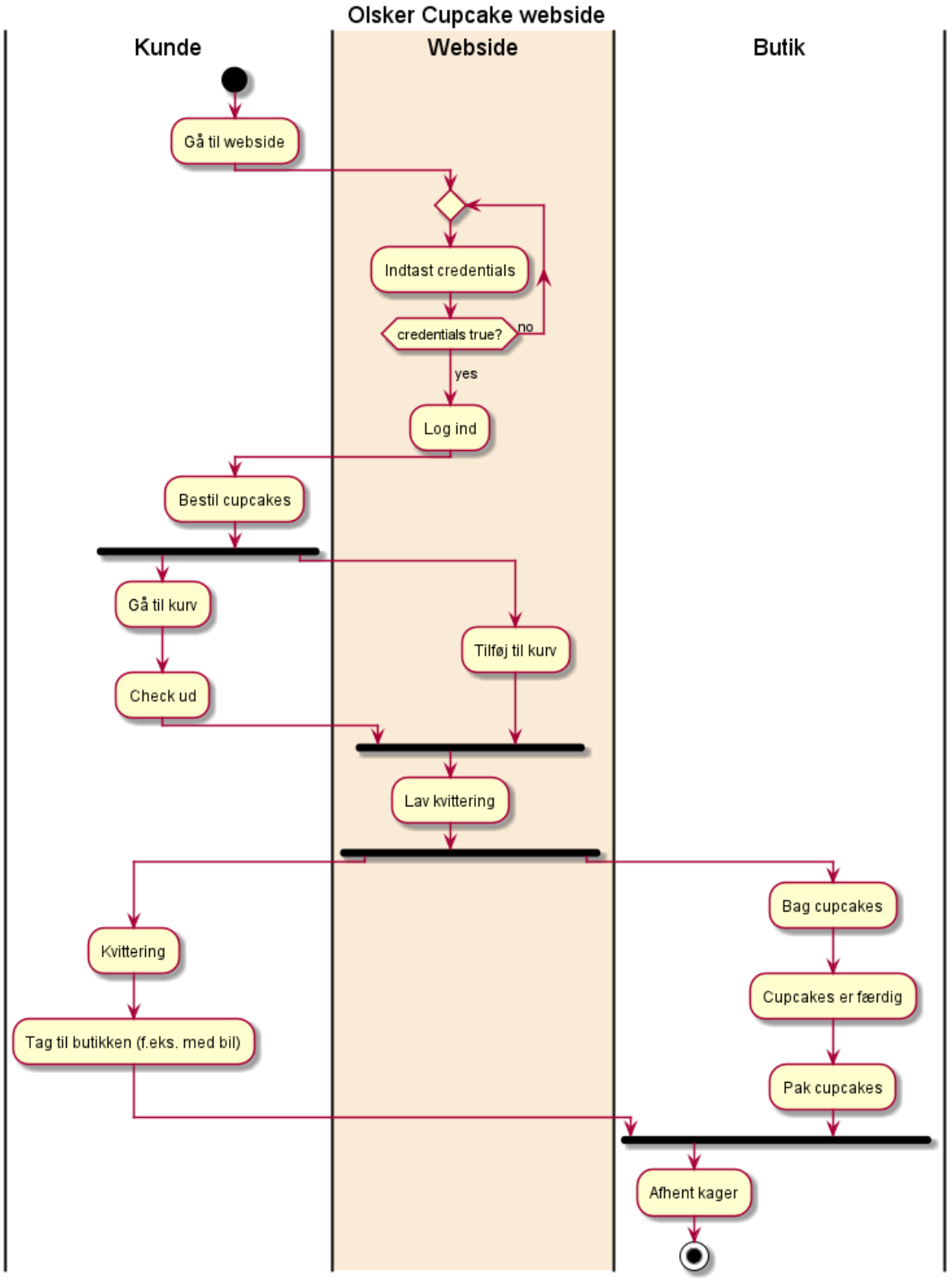
Firmaets håb med systemet er en webshop der kan hjælpe med at reklamere og forøge butikkens synlighed, og også hjælpe kunder med at kunne bestille ordre online til en bestemt tid for afhentning.

User stories der blev udleveret til os:

- US-1: Som kunde kan jeg bestille og betale cupcakes med en valgfri bund og top, sådan at jeg senere kan køre forbi butikken i Olsker og hente min ordre.
- US-2 Som kunde kan jeg oprette en konto/profil for at kunne betale og gemme en en ordre.
- US-3: Som administrator kan jeg indsætte beløb på en kundes konto direkte i MySql, så en kunde kan betale for sine ordrer.
- US-4: Som kunde kan jeg se mine valgte ordrelinjer i en indkøbskurv, så jeg kan se den samlede pris.
- US-5: Som kunde eller administrator kan jeg logge på systemet med email og kodeord. Når jeg er logget på, skal jeg kunne se min email på hver side (evt. i topmenuen, som vist på mockup'en).
- US-6: Som administrator kan jeg se alle ordrer i systemet, så jeg kan se hvad der er blevet bestilt.
- US-7: Som administrator kan jeg se alle kunder i systemet og deres ordrer, sådan at jeg kan følge op på ordrer og holde styr på mine kunder.
- US-8: Som kunde kan jeg fjerne en ordre fra min indkøbskurv, så jeg kan justere min ordre.
- US-9: Som administrator kan jeg fjerne en ordre, så systemet ikke kommer til at indeholde ugyldige ordrer. F.eks. hvis kunden aldrig har betalt.

Aktivitetsdiagram

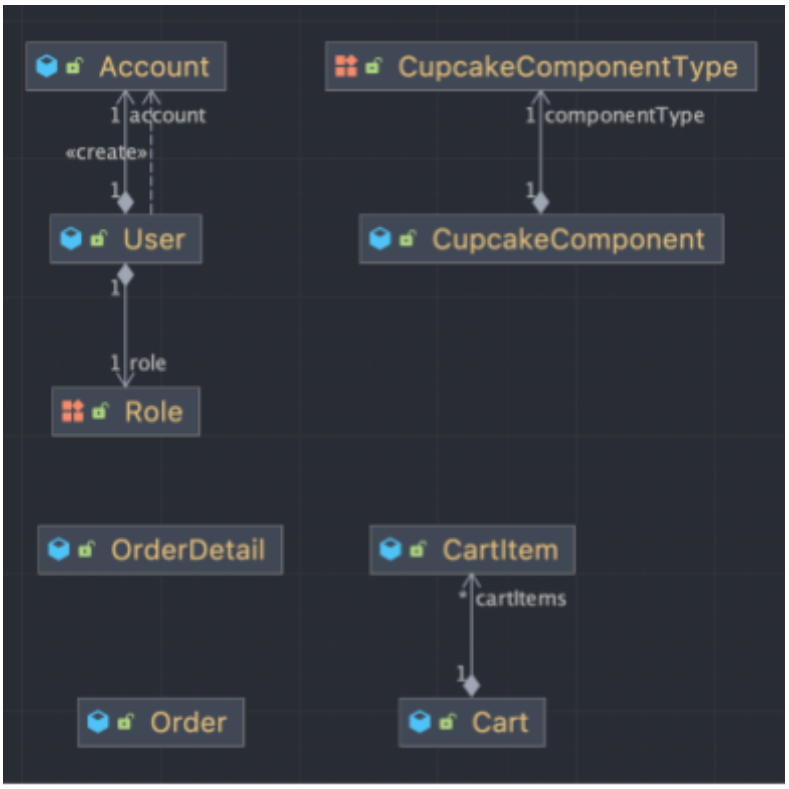
Aktivitetsdiagram over den overordnede workflow i butikken når butikken får sin hjemmeside:



Figur 1: [cupcakeProject/ActivityDiagram.png at main · Legolai/cupcakeProject (github.com)]

Domæne model og ER diagram

Domænemodel:

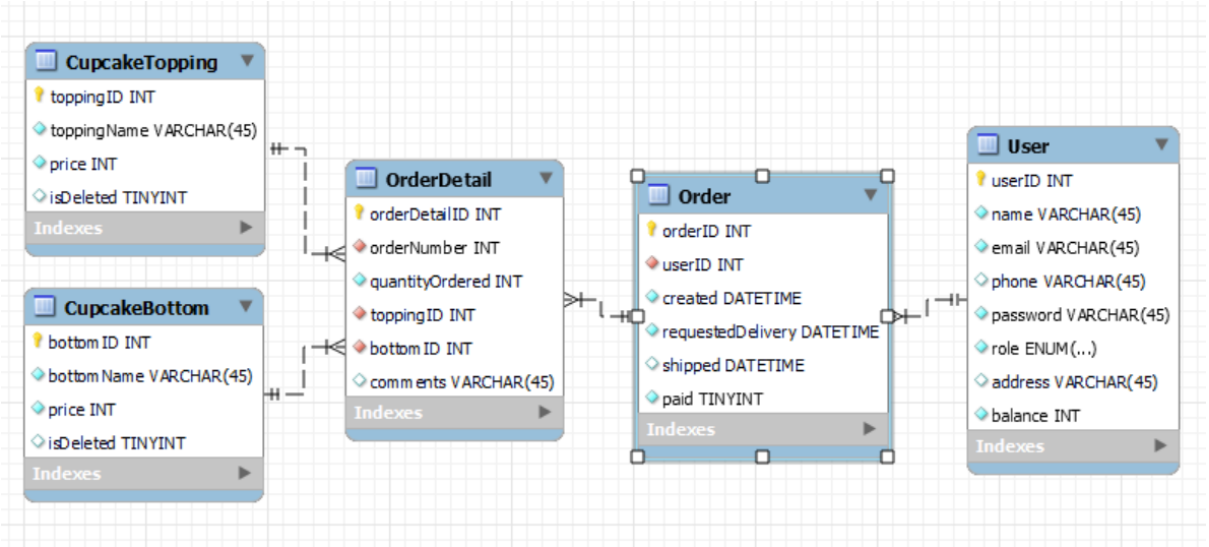


Figur 2: Domænemodel

Der blev ikke lavet en domænemodel da det blev hurtigt klart at systemet skulle kunne håndtere en bruger(User), som kan have et antal ordre(Order) der indeholder en liste af ordre detaljer (OrderDetail), og i ordre detaljerne står der hvilke cupcake toppings og bunde der er blevet bestilt.

Vi sprang derfor over dette step, direkte til at lave en EER diagram. Det blev dog senere klart at det ville være smart at have en indkøbskurv, hvilket så blev lavet. Og en indkøbskurv (cart) skulle være med i en domænemodel, som der kan ses på figur 2, som lavet ved projektets afslutning.

EER diagram:



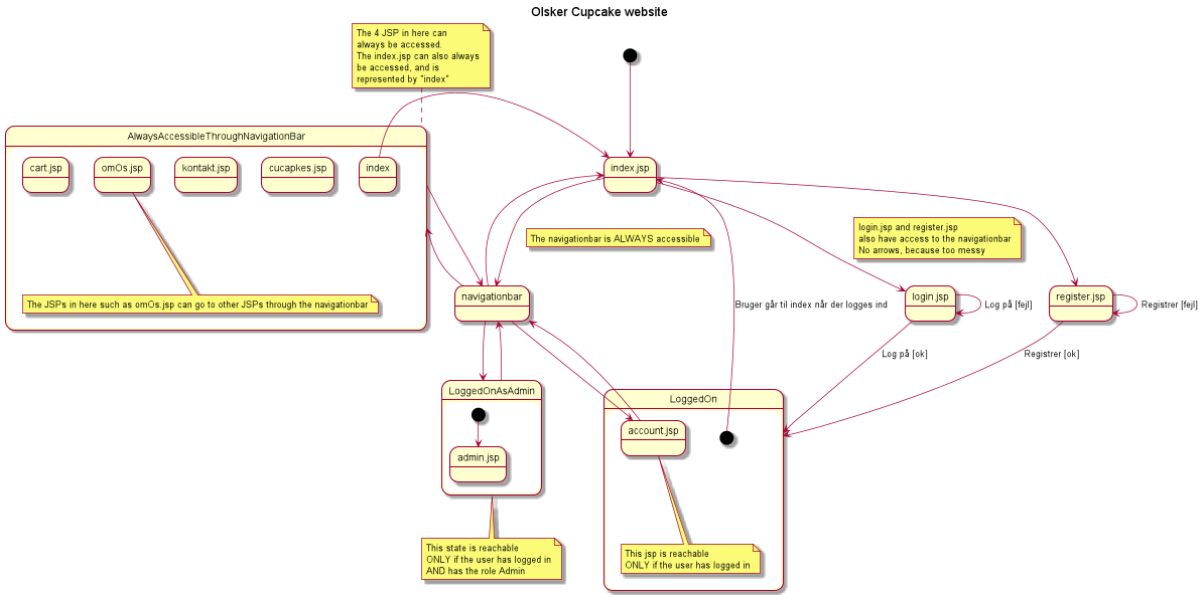
Figur 3: [cupcakeProject/cupcakeModel.mwb at main · Legolai/cupcakeProject (github.com)]

Alle tabellerne er på 3. normal form og der er ingen 1-1 relationer eller mange-mange relationer.

Heraf skal der noteres at User attributen role er en ENUM, fordi at vi besluttede at der kun var 2 relevante roller (en customer og en admin).

Under processen hvor vi lavede EER diagrammet blev der diskuteret om mange forskellige måder diagrammet kunne laves på.
Heraf om CupcakeTopping og CupcakeBottom skulle slås sammen til én tabel kaldet f.eks. Cupcake (senere hen da vi lavede programmet, blev det dog klart at det ville have været bedre at slå dem sammen, men ved det tidspunkt var det lidt for sent, og det ville have taget meget tid at ændre det).
Derudover var der også diskussion om OrderDetail skulle have en dedikeret primary key eller en primary key bestående af flere sammenslåede attributter.
Vi var dog meget enige om at have en Order og OrderDetail for at separere OrderDetail som fortæller om cupcake valg og antal, mens Order indeholder en liste af OrderDetail og info såsom created, requestedDelivery og paid.

Navigationsdiagram



Figur 4: [cupcakeProject/NavigationDiagram-Olsker_Cupcake_website.png at main · Legolai/cupcakeProject (github.com)]

Ekstra kommentarer:
- cupcakes.jsp har en funktion hvor man kan tilføje cupcakes til cart. Dette gøres gennem en command (ikke en servlet!) og commanden redigerer tilbage til cupcakes.jsp
- cart.jsp har en checkout knap hvor commanden redigerer til account.jsp.

Særlige forhold

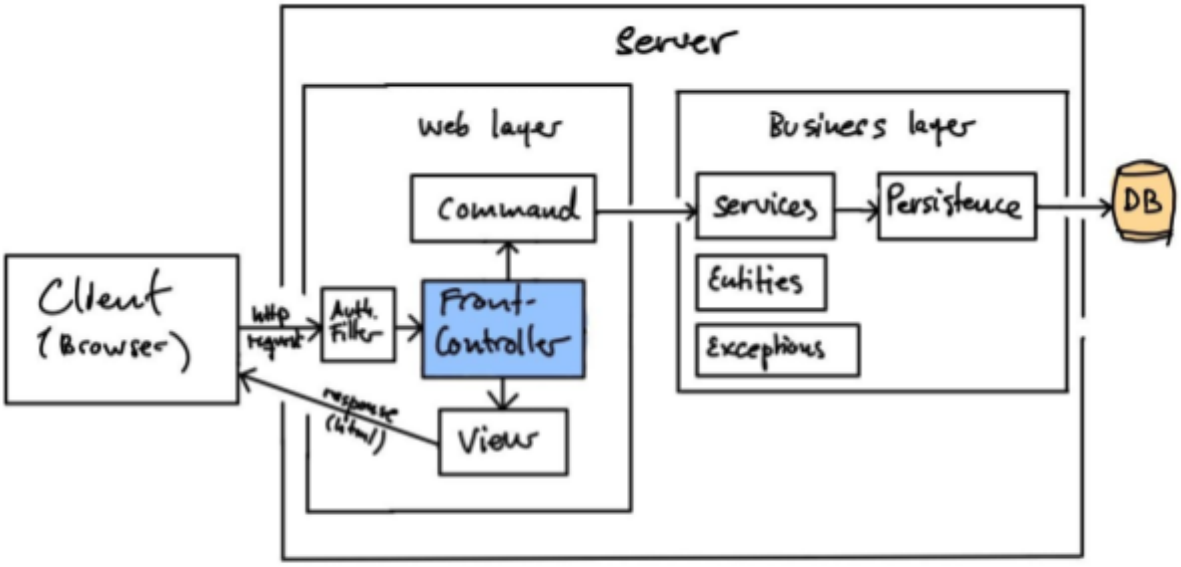
Når en session oprettes vil der blive genereret et Cart objekt som lægges på session som et attribut, dette sker via en web-listener. Derudover vil der også lægges et User objekt på session når et login forsøg er blevet valideret og godkendt.

Ved operationer som login eller andre der fx berører databasen, kan der opstå fejl som skal kunne behandles som de opstår i applikationens levetid. For at kunne dette bliver der benyttet try/catch, hvor en eventuel fejl håndteres således at brugeren bliver sendt tilbage siden. Hvis det var sket på baggrund en fejl i indtastning fra brugerens side, vil der blive givet en passende fejlbesked så indtastning kan bliver korrigeret.
I andre tilfælde vil brugeren bliver ført over til error.jsp, hvor fejlkoden og besked gives.

I forbindelse med cupcake topping og bundene, har vi valgt at gøre det således at disse bliver loadet ved applikation start og derefter lagt på application scope. Det har vi gjort for at spare på kald til databasen, da det alternative ville være at skulle hente data fra databasen hver gang der bliver navigeret hen til siden hvor man kan lave ens cupcakes og når man lægger en cupcake i kurven .
Men for så at løse problemet med hvornår dataen som ligger på application scope skal opdateres, har vi valgt igen at lave en web-listener, som bliver aktiveret når en trigger variable i application scope bliver ændret.

Vi har taget udgangspunkt i startkoden som var givet, men vi har også taget inspiration fra en tidligere version af startkoden, hvor måden af MVC arkitekturen er blevet sat op på er lidt anderledes end det oprindelige udgangspunkt.
Som det kan ses på figur 5, bruger vi et filter til først og fremmest til at håndter om requesten som klienten sender har tilladelse til at tilgå en ønsket side eller funktionalitet.

Requesten kan bliver afvist på grund af at man ikke er logget ind, i det tilfælde bliver klienten ført over til login.jsp siden, men i tilfælde hvor den rolle som klientens konto har ikke er tilstrækkelig for at tilgå en bestemt side eller funktionalitet vil man bliver ført til index.jsp siden.



Figur 5: https://github.com/jonbertelsen/sem2-startcode, Jon Bertelsen

Et request vil som det kan ses på figur 5, efter at være blevet kontrolleret af filteret, bliver håndteret af en front controller, som er en servlet.
FrontControlleren vil så ved hjælp af en singleton kaldet CommandController, kunne finde kommandoen som strien fra requesten angiver. Kommandoen vil så blive udført hvis stien eksisterer, ellers sendes klient til error.jsp og det vil så fremgå, at siden klient prøvede at tilgå ikke eksisterer.

Status på implementation

Det meste af systemet er lavet men heraf mangler vi dog:

- Delete operationer for databasen. Heraf følte vi at det ikke var det mest relevante/vigtige operation, men der er dog en 'soft-delete' for cupcake toppe og bunde (isDeleted).
- Kunden kan ikke betale for deres ordre (eller indsætte penge på deres konto, administratorer kan indsætte penge)
- Vi mangler én enkelt jsp side, hvilket er en kvitteringsside. Dog kan denne funktionalitet blive gjort på 'cart.jsp' som er indkøbskurv siden. Dette er dog heller ikke blevet gjort.
- Og den sidste, men en meget stor mangel, er vores styling af websiderne ifølge af vores mockup.

Proces

Vi startede dagene med status og planen for dagens forløb, heraf mål for hvad der skal laves og gøres klare til dagen efter. Der blev så delt opgaver ud til de fremmødte.
Opgaverne blev delt ud i ny branches i vores GIT projekt, for at vi ikke skulle komme ud i større konflikter eller uheld med overskrivning af hinandens kode.
Merge med main blev altid aftalt inden det blev gjort, således alle var enig og klar på det.
Et merge af en vores branches forløb ved brugen af rebase og squash ind i main, hvilket gjorde det smertefrit i langt de fleste tilfælde.

En form for Kanban tavle ville nok havde hjulpet med at overskueliggøre projektet og hvor langt tingene var i forløbet, men på grund af mængden af tid til rådighed og skala af projektet. Af den grund blev det valgt fra, da mængden af tid der skulle bruge på fremstillingen og vedligeholdelsen af tavle vil tage for meget af tiden tilrådighed.Men som projektet forløbe kunne vi hurtigt mærke at overblikket mangle, også selvom vi havde EER-diagram og lignende.