FOG Carports

Navne:

Ermin Dzafic

Daniel Pedersen

Jens Christian

Email:

Daniel: cph-dp134@cphbusiness.dk

Ermin: cph-ed79 @cphbusiness.dk

Jens:cph-jo163@cphbusiness.dk

Github:

Incognito95

ErminDZ

Mawl026

Klasse: B

Tidspunkt:

Mandag 05. Maj 2021

Indholdsfortegnelse:

Links

Link til vores repository:
https://github.com/Incognito95/Carport

- **Link til demovideo (på YouTube, Vimeo etc). En video på ca. 5 minutter, som viser brugen af systemet. Formålet er at Censor og eksaminator skal se videoen og få et indtryk af det færdige resultat. Demovideoen kan f.eks. optages ved hjælp af et screencast program. F.eks. Screencast-o-matic. Gruppen kan evt. sidde sammen på Zoom og dele en skærm mens de laver demoen, så flere kan kommentere undervejs.**

Link til demovideo for vores hjemmeside på YouTube:

Link: https://youtu.be/DbzB9CHomIo

- **Link til kørende version af deres website på Digital Ocean (en deployet version) og angivelse af login navn + kodeord til en demo admin bruger.**

Link til hjemmeside:

Daniel's Droplet:

Link: http://104.248.253.9:8080/sem2-startcode-1.0-SNAPSHOT/

Ermin's Droplet:

Link: http://64.227.124.226:8080/sem2-startcode-1.0-SNAPSHOT/

Jens Christian Øgaard's Droplet:

Link: http://104.248.141.139:8080/sem2-startcode-1.0-SNAPSHOT/

Login til de forskellige roller/brugere på hjemmesiderne:

email: customer
password: 1234

email: employee
password: 1234

email: admin
password: 1234

Indledning

Kort intro til hvad dette projekt omhandler. Formålet med indledning er at sætte en fagfælle i stand til at forstå resten af rapporten. For jer som studerende er en "fagfælle" en anden datamatiker studerende på 2. semester der er på samme niveau, men som ikke kender opgaven.

Projektet omhandler konstruering af en hjemmeside for et byggemarked(FOG), som gerne vil have en mulighed for, at lave skræddersyede carporte og optimere et system, der allerede er der men, som er mangelfuld og gammel. Kunderne skal kunne se en liste af de forespørgsler som de har oprettede samt en ordrebekræftelse når en bestilling er godkendt.

Baggrund

Virksomheden, som skal bruge systemet er et byggemarked, der sælger byggematerialer og i det her tilfælde carporte i færdige pakker og skræddersyede carporte. Det er her systemet skal gå ind og erstatte det eksisterende system, som er gammelt og mangelfuldt. Det eksisterende system fungere dog, det er bare ikke muligt, at tilføje nye ting og nogle ting er hardcoded.?

Kunden vil gerne kunne bestille en carport eller vælge, at lave i sin egen

Det typiske der skal med for at forklare projektet er:

- En kort beskrivelse af den virksomhed som skal bruge systemet
- Hvilke krav kunden har til systemet forklaret i brede termer, f.eks. "*kunden skal kunne bestille en carport, hvor man kan vælge både højde og bredde.*" i modsætning til "*der skal være en drop down menu med scrollbar i højre side med en liste over hvilke typer der er og hvad de koster*".

Teknologivalg

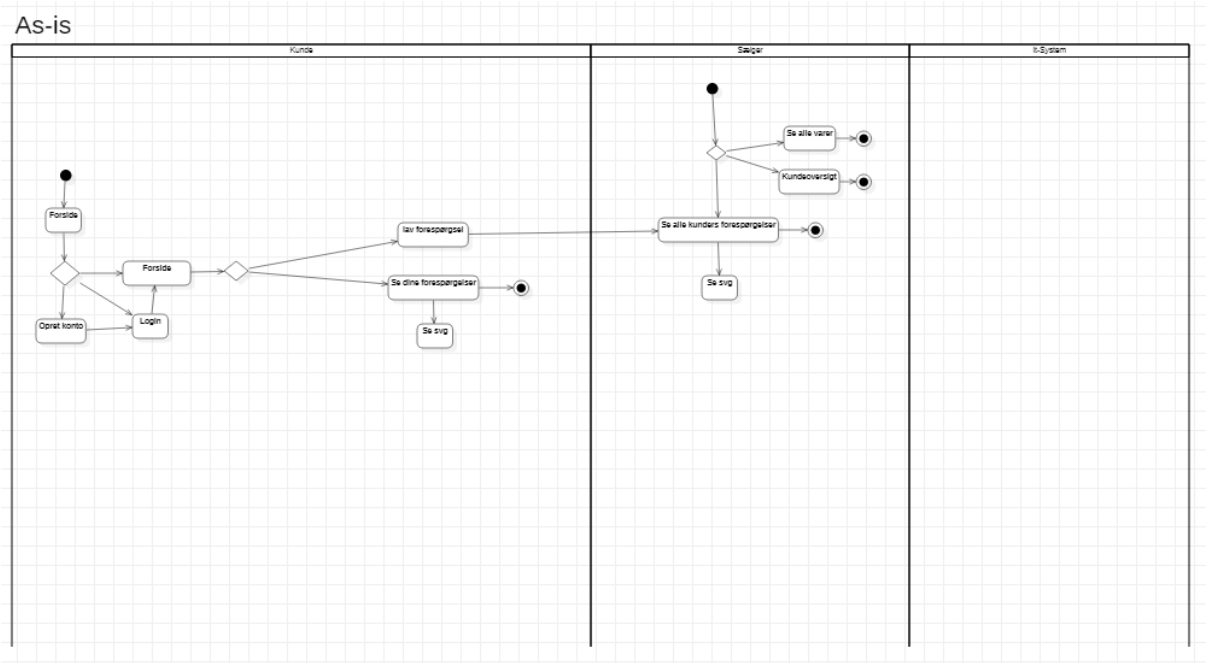
- IntelliJ IDEA 2020.3.3 (Ultimate Edition)
- MySQL Workbench 8.0 CE
- Apache Tomcat 9.0.44
- Adobe XD 39.0.12.12
- HTML5
- CSS3
- Github/Github Desktop - 2.7.2
- Git Bash/terminal
- Bootstrap 5
- Digital Ocean - Droplet
- Github Kanban Board

Krav

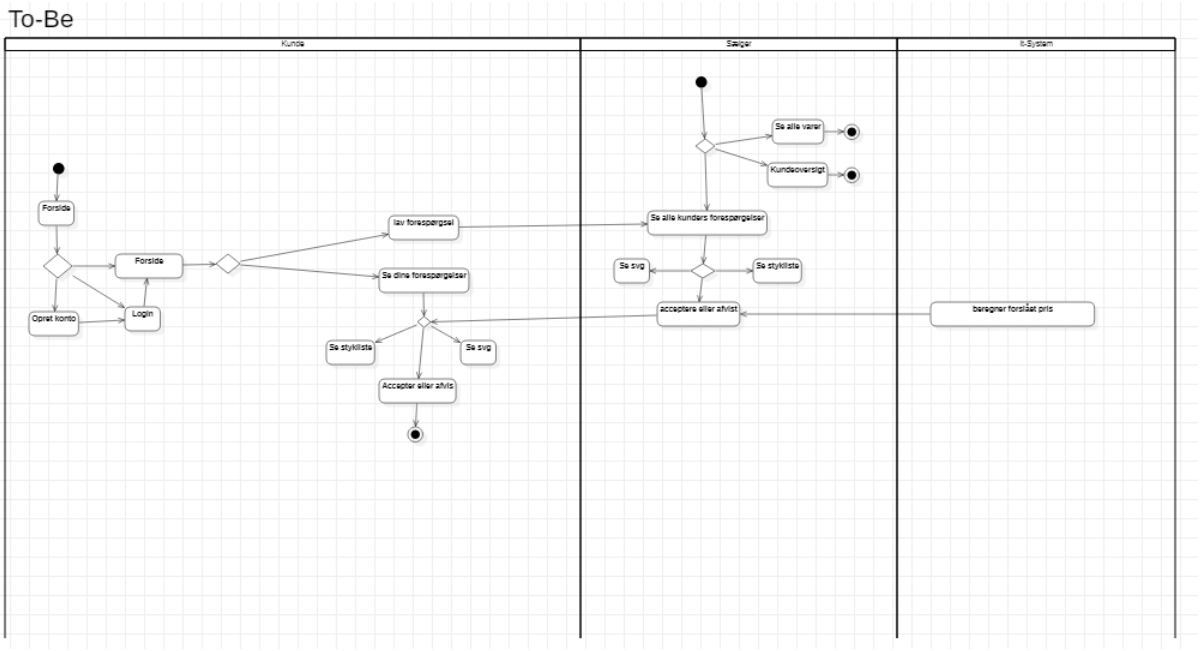
Aktivitetsdiagrammer

Arbejdsgange der skal IT-støttes. I afsnittet beskrives de overordnede arbejdsgange før og efter IT systemet, beskrevet med “as-is” og “to-be” som aktivitetsdiagrammer.

As-Is:



To-Be:



Scrum user stories

Dette afsnit skal beskrive de user-stories der er aftalt med product-owner. Det er vigtigt for hver user-story at man kan spore det tilbage til aktivitets diagrammerne fra forrige side, og at I har en håndfuld user stories som er lavet fuldt ud, dvs:

- der er accept kriterier
- der er brudt ned i tasks
- der er lavet et estimat

Første sprint

User stories:

1. Som **kunde** ønsker jeg, at kunne **vælge, at logge ind eller oprette konto** sådan, at **kunder kan se deres forespørgsler** på deres kundeprofiler. M

2. Som **kunde** ønsker jeg at kunne **sende en forspørgelse** på en carport hvor jeg selv kan vælge carportens dimensioner, tagtype mm. Sådan, at en **kunde selv kan designe deres carport. M**

Accept Kriterier:

Givet at kunde har adgang til hjemmesiden **når** man tilgår den **så** kan man vælge enten, at logge ind eller oprette en profil**.**

Givet at kunden har logget ind på hjemmesiden **når** kunden så indtaster dimensionerne i carport vælgeren **så** gemmes forespørgslen i databasen**.**

Review:

Daniel:
Jeg synes vi blev forvirret ret meget da vi var ikke så klar over hvilken retning vi skulle gå i forhold til projektet og vi skulle ha afklaret nogle ting flere gange.
Da vi fik afklaret det vi skulle så var det nemmere at komme i gang og arbejde sig frem til det vi skulle nå.

Jens:
Jeg har været ærgerlig over, at have misset Zoom møder med gruppen og underviser. Jeg synes, at opgavens omfang har været uklart i starten, men efter review med gruppe og undervisere, har det gjort projektet med mere klart.

Ermin:
Jeg synes, at gruppens samarbejde har været virkelig godt, men vi brugt lidt for meget tid på tekniske problemer og udfordringer fremfor at bruge mere tid på implementering af user stories og accept kriterier.

Andet sprint

User stories:

1. Som **ansat** ønsker jeg, at kunne **vælge, at logge ind** sådan, at **ansatte kan se kunders forespørgsler** på hjemmesiden i en tabel. **M**

2. Som **admin** ønsker jeg, at kunne **vælge, at logge ind eller oprette kontoer til ansatte** sådan, at **ansatte kan se deres køb og forespørgsler** på deres kundeprofiler. **M**

Accept Kriterier:

Givet at en ansat kan logge ind, **når** en ansat vil se kunders forespørgsler **så** vises der en liste over kunders forespørgsler fra en database.

Givet at en admin har kontrol over **når** nogen andre tilgår hjemmesiden **så** kan admin oprette profiler til ansatte.

Review:

Det var en virkelige svær uge med meget få fremskridt da vi stødte ind på mange problemer. Det største problem var at ansatte ikke kunne komme ind på en side som viser alle kunders forespørgsler på grund af nullpointerexception og det viste sig at problemet ligger i datatyperne som var anderledes til dem i databasen.

Tredje sprint

User stories:

1. Som **ansat** vil jeg gerne **automatisk få vist en stykliste og SVG-tegning** for en given forespørgsel, sådan at, jeg f.eks. kan **godkende forespørgslen fra kunden**. X**L**

2. Som **kunde** skal jeg kunne **se min forespørgsler sådan**, at jeg kan **vælge imellem, at gå til betaling, annullere eller redigere** min forespørgsel. **L**

3. Som a**dmin** skal jeg kunne **tilføje nye varer til databasen**(varenr., navn, dimensioner, beskrivelse, pris.). **M**

Accept Kriterier:

Givet at kunder har indtastet forespørgsler **når** de vælger carport **så** får ansatte vist en SVG-tegning.

Givet at kunder har indtastet forespørgsler **når** de vælger carport, **så** skal de få vist en liste over deres forespørgsler.

Givet at admin har adgang til hjemmesiden og **når** de er logget ind, **så** skal de kunne tilføje nye varer via hjemmesiden og **så** det bliver gemt i databasen.

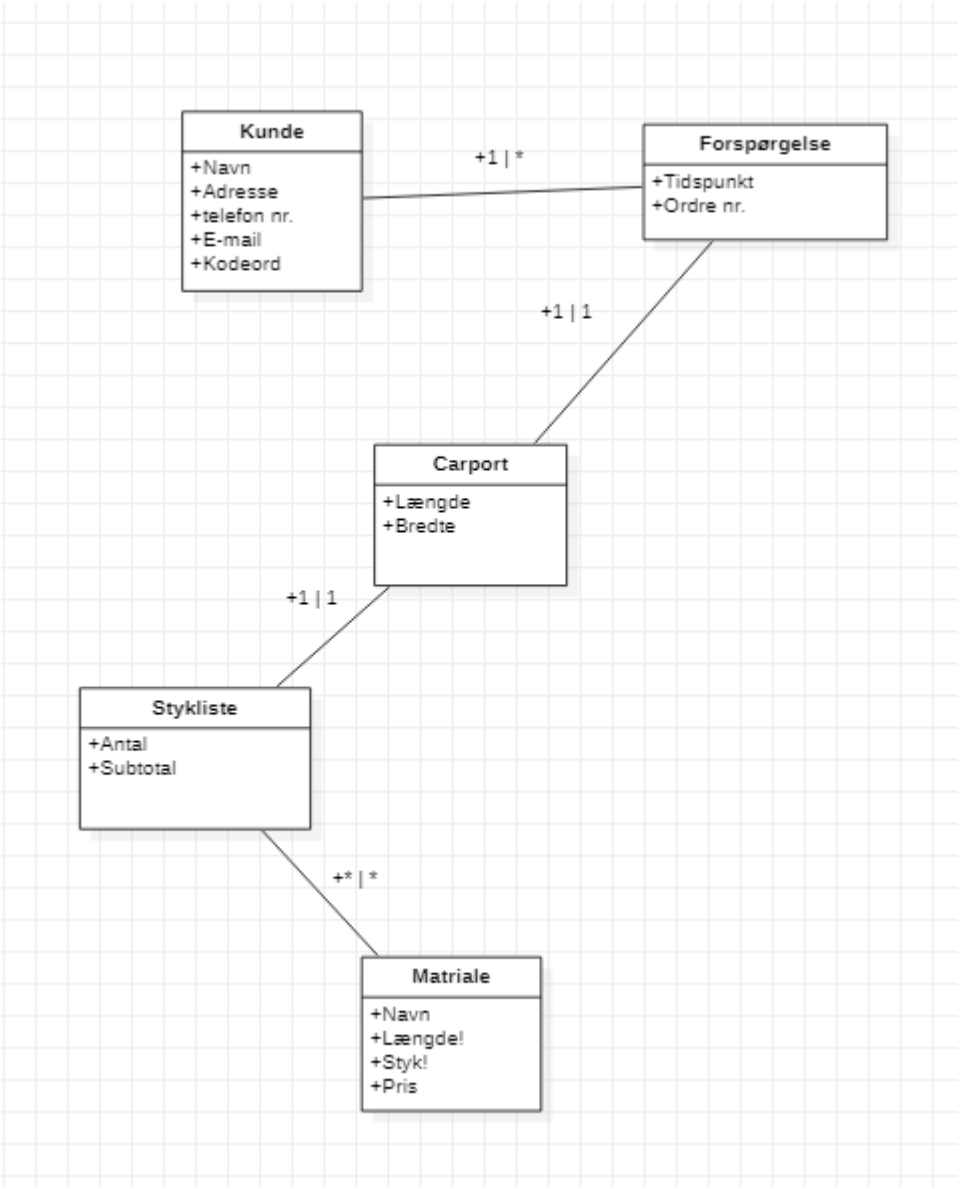
Givet at admin har adgang til hjemmesiden og **når** de er logget ind, **så** skal de kunne se en sælger oversigt, samt kunne tilføje nye sælgere via hjemmesiden og **så** det bliver gemt i databasen.

Review:

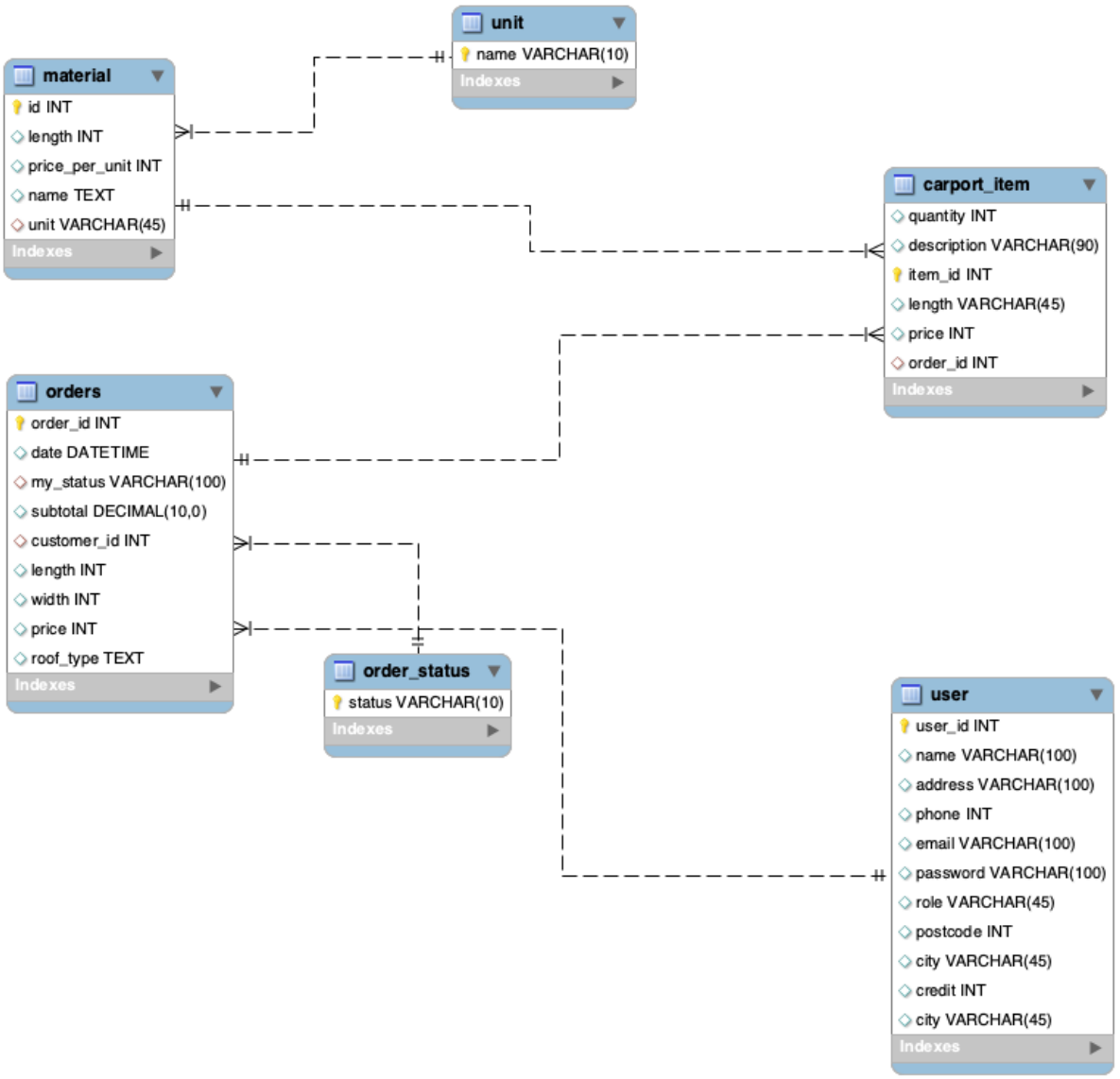
Det blev meget bedre i dette sprint med gode fremskridt. Vi fik sat alle dataene fra databasen ind i de forskellige tabeller samt fik vi lavede det statiske svg dog ikke det dynamiske da vi ikke kom til måls med materialerne.

Domæne model og ER diagram

Domæne model:



ER diagram:



Det interessante ved domænemodel og databasen, er at de langt hen af vejen er grundlaget for resten at systemet. Tabeller og relationer siger noget om *hvad* systemet arbejder med, ikke *hvordan*. Så det er godt sted at starte. Husk at domænemodellen kommer før ER diagrammet, og at domænemodellen ikke handler om implementering. Det er kun konceptuelle klaser med attributter, der beskriver den virkelige verden.

Som led i beskrivelsen af Domæne eller ER diagram skal man have følgende med:

- **Diagram over hele modellen. Det er vigtigt at få plads til alle tabeller og alle relationer.** ***Det kan så betyde at man ikke kan få plads til alle attributter på de enkelte tabeller. Dem kan man slå op i databasen, så det er ikke så vigtigt***

Vi har fået plads til det hele.

- **Hvis nogle af tabellerne ikke er på 3. normal form vil det være almindeligt at nævne det, og forklare hvorfor det er gjort (tidspres eller anden overvejelse).**

Vi havde ikke brugt 3. normal form på grund af tidspres. Vi havde andre overvejelser, vi ville gerne lave vores tabeller og relationer mere simpelt, så arbejdet går hurtigere og nemmere når vi skal lave vores SQL sætninger.

- **Hvis der anvendes 1-1 relationer kan man beskrive hvorfor man ikke blot har en tabel.**

Vi har ikke anvendt 1-1 relationer.

- **Hvis nogle tabeller implementerer en mange-mange relation vil det være normalt at beskrive det.**

Vi har ikke anvendt mange til mange relationer.

- **Hvis der er flere veje at nå fra et sted til et andet vil det nemt gøre det svært at holde databasen konsistent. Hvis I har gjort det alligevel så skal I forklare hvorfor.**

Det har vi ikke.

- **Hvis der er tabeller hvor man benytter andet end et automatisk genereret ID som nøgle skal man forklare det.**

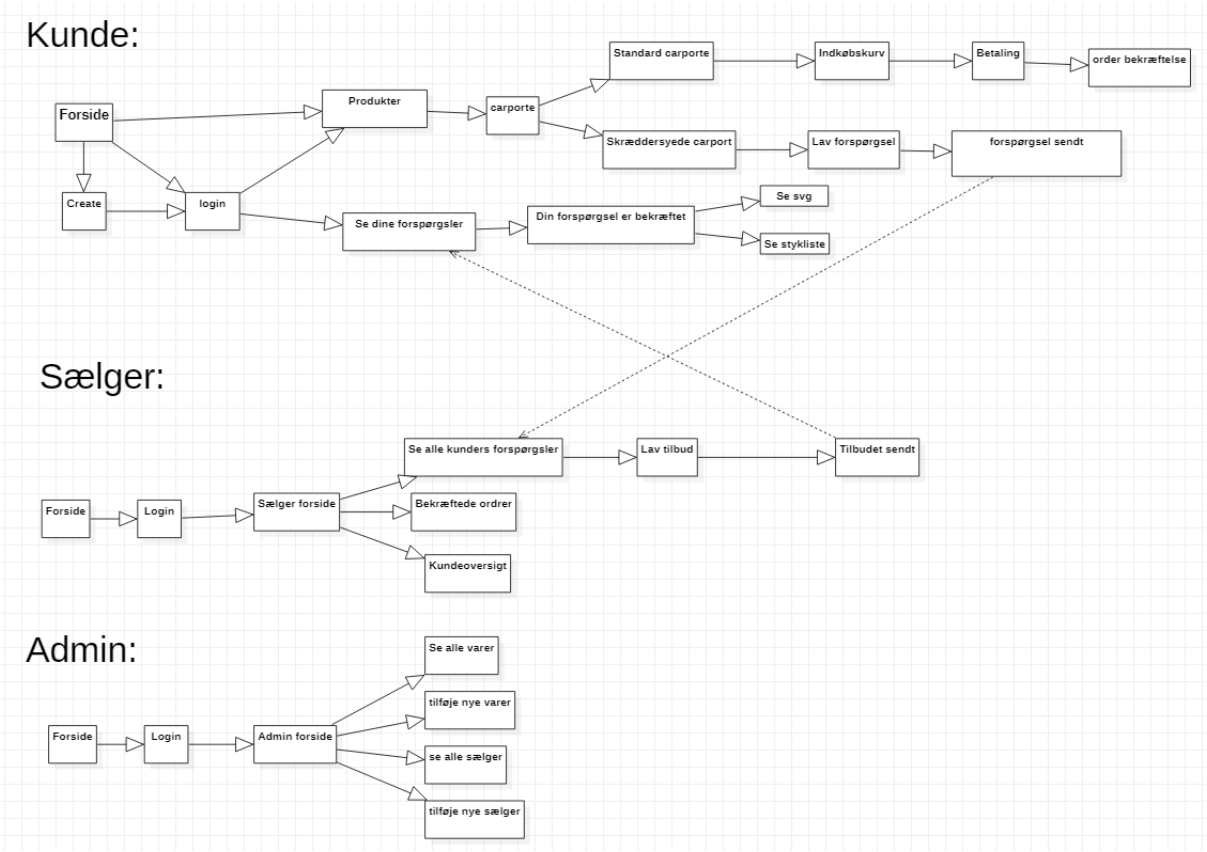
Det er der ikke.

- **Der er interessant at beskrive hvilke overvejelser der ligger til grund for de konkrete valg der er i ER modellen (fremmednøgler, constraints, triggers, osv)**

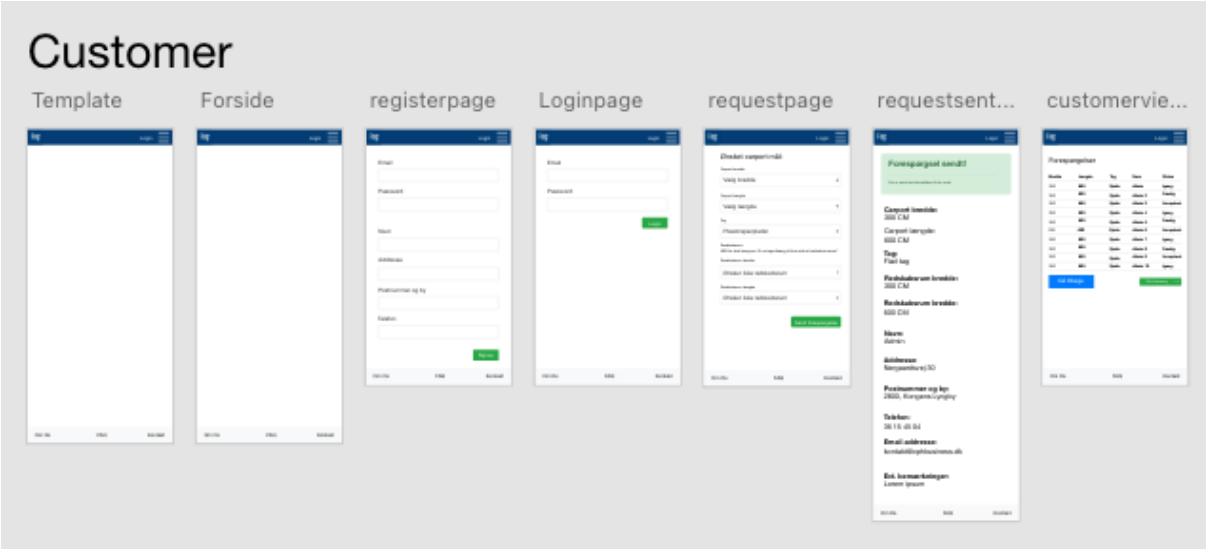
Har ikke haft nogen bestemte konkrete overvejelser, som vi kan pege til omkring hvorfor vi har truffet diverse valg.

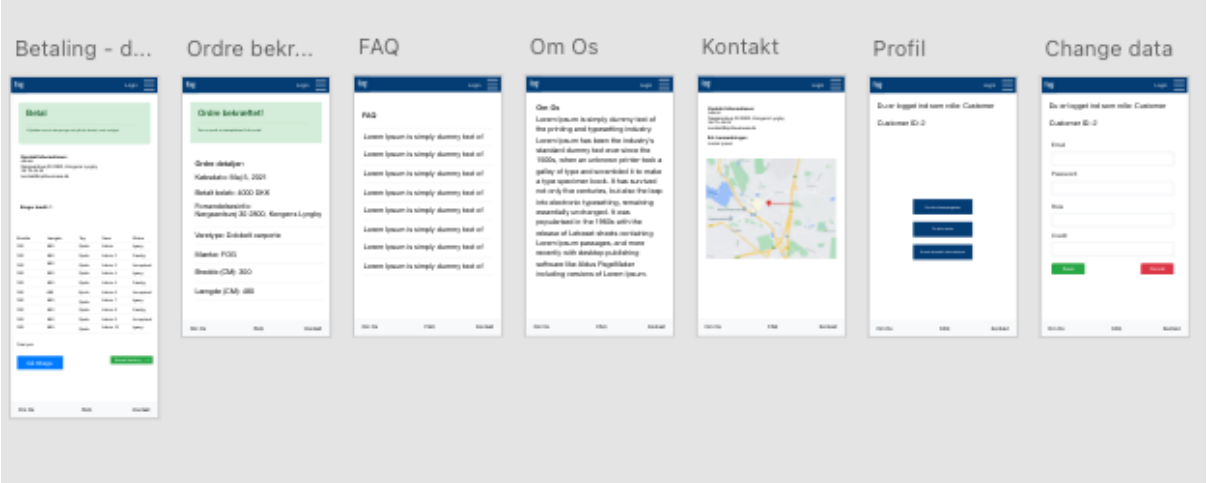
Navigation Diagram & Mockups

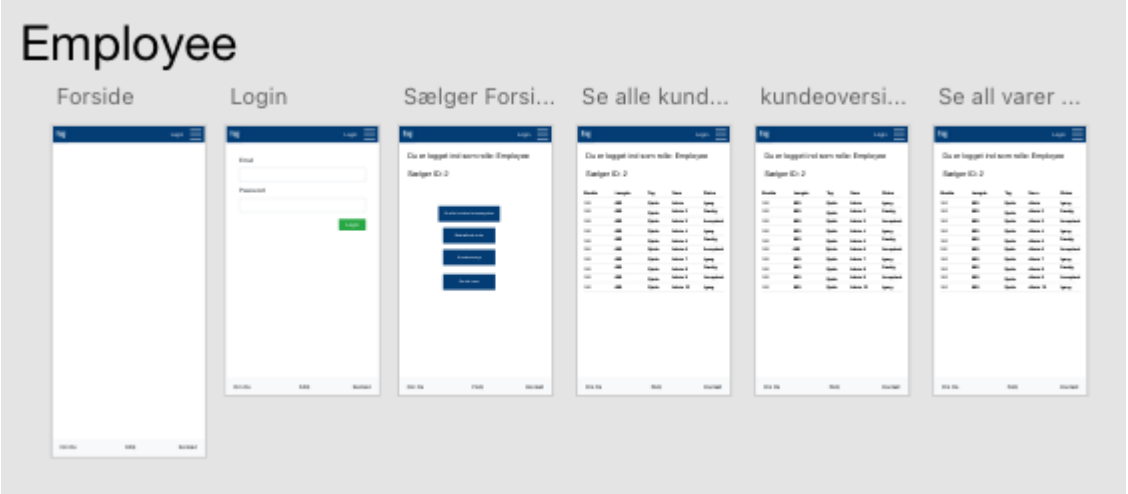
Navigation Diagram:

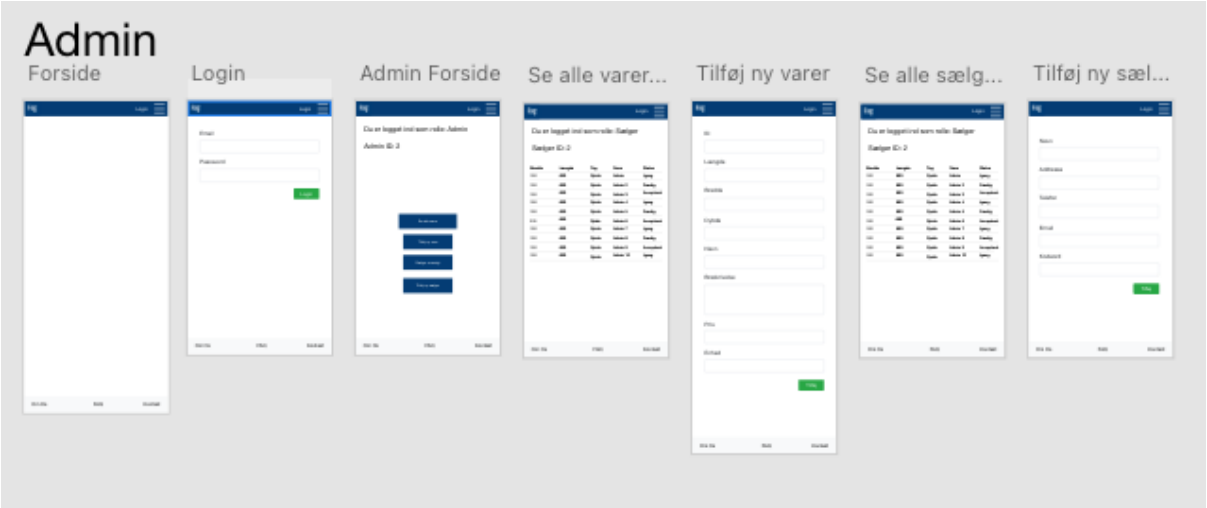


Mockups:


Customer
Template
Forside
registerpage
Loginpage
requestpage
requestsent...
customervie...


Betaling - d...
Ordre bekr...
FAQ
Om Os
Kontakt
Profil
Change data


Employee
Forside
Login
Sælger Forsi...
Se alle kund...
kundeoversi...
Se all varer ...



Adobe XD Mockup filen ligger i documents mappen:
carport/documents/Mockups/Mockup.xd

Det som brugeren oplever er en række websider hvor man kan indtaste oplysninger gå videre til andre sider. I større systemer kan det være svært at bevare overblikket over hvilke sider der er, og hvordan man kommer fra den ene til den anden. Navigations Diagrammet er beregnet på at vise dette på en mere overskuelig måde. Som led i beskrivelsen af navigations diagrammet skal følgende med:

- **Oversigts diagrammet. Hvis det bliver for stort må man dele det op. Men det er vigtigt at der er et overordnet diagram.**

Vi har delt det op.

- **Hvis man har benyttet sig af en "fælles navigations bar" i toppen af alle sider skal man forklare det.**

Vi har brugt et fælles navigations bar på toppen som indeholder alle de links til de sider vi skal hen til, og i bunden af alle siderne, har vi også en footer, som indeholder links til faq, om os og kontakt.

- **Hvis nogle sider kun kan nås af nogle brugere (dem der har konto, dem der er logget ind, dem der arbejder i butikken,…), så skal det fremgå.**

Man kan kun lave en forspørgelse hvis man har en konto.
Som kunde kan man lave en forespørgsel og se alle sine forespørgsler og ordrer.
Som ansat kan man se oversigt over alle fogs kunder samt alle deres forespørgsler.
Så kan ansatte også bekræfte de kommende forespørgsler som bliver sendt tilbage til kunde som en ordre.
Så kan ansatte også se en oversigt over alle deres varer.

Som admin kan man se oversigt over alle fogs varer samt tilføje nye varer.
Så kan admin også se oversigt over alle deres sælger samt tilføje nye sælger.

- **Navne på jsp sider skal fremgå, og hvilke servlet der bringer en fra den ene side til den næste.**

customerpage -> requestpage -> thankyoupage -> customerpage -> customerviewrequestpage -> customerpage ->customervieworderspage -> customerpage -> orderconfirmationpage

employeepage -> seallcustomersrequest -> employeepage -> seallconfirmedorders -> employeepage -> seallcustomerpage -> employeepage -> seallproducts -> employeepage !! hvor skal se stykliste og se svg være!!!

adminpage -> seeallproductspage -> adminpage -> addnewproductpage -> adminpage -> seallsellerspage -> adminpage -> addnewsellerpage -> adminpage

Valg af arkitektur

Beskrivelse af den forelagte skabelon, som er konstrueret ud fra et command-pattern og en 3-lags arkitektur. Facade-lag, singletons etc.

Det mappe struktur som vi har arbejdede ud fra er et flerlags arkitektur, hvor vi bruger Mapper filer inde i vores persistence mappe hvor vi lavede alle vores SQL sætninger til at både indsætte og læse data som kommer fra databasen.

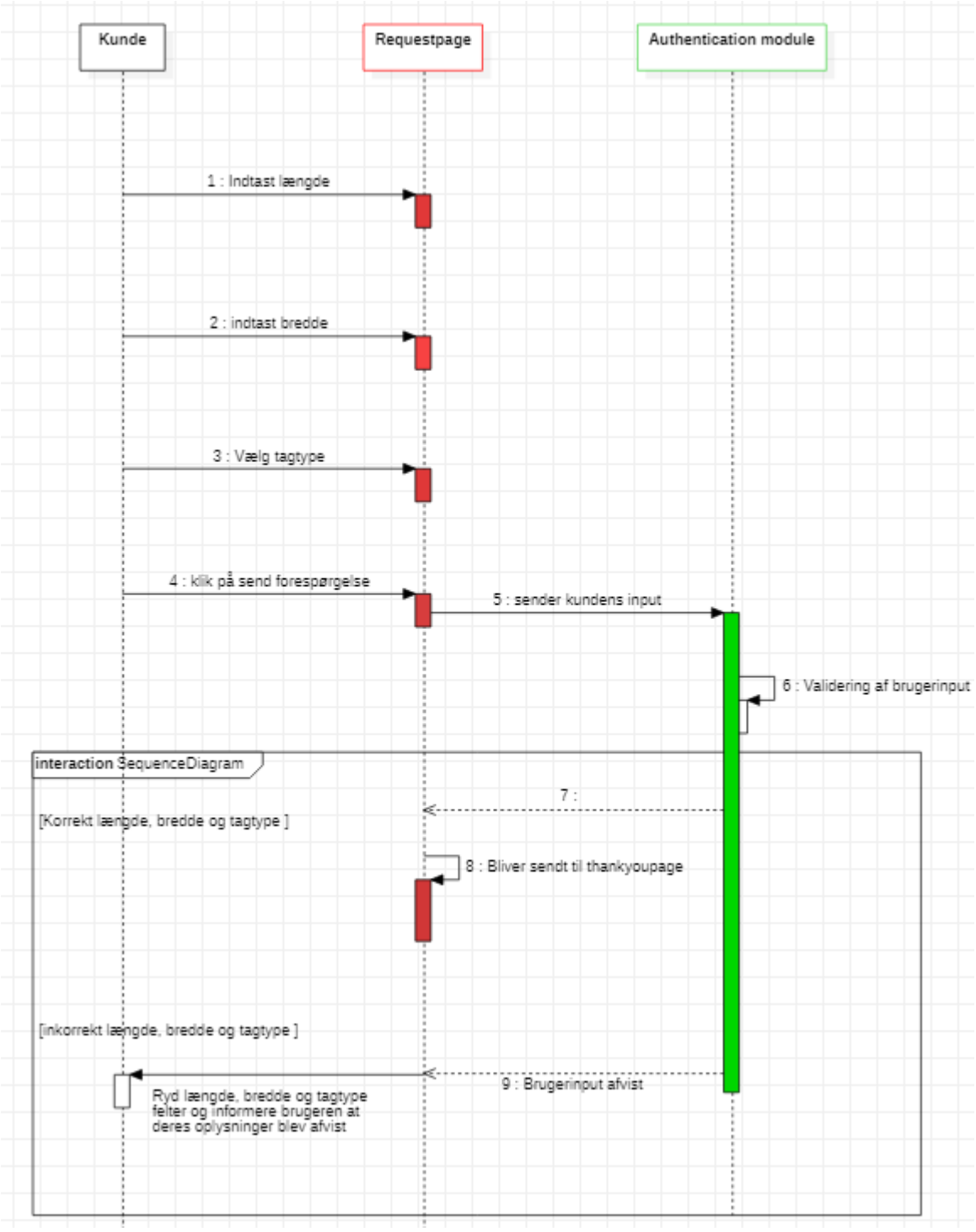
Vi bruger Facade filerne inde i services mappen til at kalde vores metoder som kommer fra mapper filer som er brugt til at køre SQL'en og tilhører de data som er blevet brugt inde i entities mappen i attributterne, konstruktor og getters og setters.

Command mappen er det sted hvor vi indeholder alle vores command filer som er brugt til at kalde facaden fra den tilhørerne facade fil og den metode som tilhører den metode i mapper filen.

WEB-INF er hvor vi ligger alle vores JSP, JSP siderne er brugt til at vise indholdet som kommer fra databasen og de JSP sider bliver sa kaldt inde i Command.java filen så det kan blive vist ude i browseren.

Test mappen er hvor vi lægger alle vores test af vores kode til at se om hvordan vores kode fungerer og at det gøre det som det skal. Vi laver tests så vi er mere sikker på at det virker.

Sekvensdiagrammer



Dokumentation af udvalgte dele af koden. De fleste programmører kan læse de enkelte metoder i et program, mens det kan være svært at skabe sig et overblik over hvordan programmet virker på overordnet plan. Et sekvensdiagram bruges til at vise hvordan et typisk forløb foregår, eller til at vise et særligt svært særtilfælde. *Man kan aldrig dokumentere hele programmet med sekvensdiagrammer, man vælger altid nogle interessante eksempler*.

Et eksempel på et typisk forløb kunne være at brugeren præsenteres for indkøbssiden.

Her skal der vises følgende:

- **Selve diagrammet, startende med jsp-siden eller servlet.**
- **Brug de rigtige klassenavne og metode navne**
- **Undlad argumenter til metoderne**

I forklaringen til diagrammet skal du særligt lægge vægt at beskrive hvilke grene af if-sætninger der er brugt i de enkelte metoder.

Særlige forhold

Dette afsnit bruges til at beskrive særlige forhold der benyttes i programmet. Det kan f.eks. være:

- **Hvilke informationer gemmes i session**

Information som gemmes i session er dataene som kommer fra databasen som så bliver lagt ind i klasserne inde i entities mappen ved brug af konstruktor og getters and setters.

- **Hvordan håndterer man exceptions og logger.**

Vi håndterer exceptions ved at bruge try catch til at tjekke om et stump kode virker og hvis det ikke virker så bliver det kastede videre ned til catch hvor man outputter en fejlbesked hvis der sker en fejl.

- **Hvordan man på har valgt at lave brugerinput validering**

Vi har valgt at lave brugerinput validering ved at bruge en if statement til at tjekke om brugernavn eller kodeord er indtastede korrekt eller forkert, hvis den er forkert så bliver man re-direktede til den siden man allerede er på, og hvis man skriver den rigtig så bliver man sendt videre til den side man vil ind på.

- **Hvordan man har valgt at lave sikkerhed i forbindelse med login**

Vi har valgt at lave sikkerhed ved at bruge en if statement så man kan sikre sig at der ikke er mulighed for tilfældige personer til at kunne logge ind.

- **Hvilke brugertyper der er valgt i databasen, og hvordan de er brugt i jdbc**

De datatyper som vi har valgt at bruge i databasen er VARCHAR og INT. Den må vi har valgt at bruge dem i jdbc er ved at sætte dem i vores Mapper klasser ind i persistence mappen og bruge String som henter alle de data fra databasen som tilhører varchar og bruge INT som henter alle data som tilhører INT.

- **… andre elementer – i Fog projektet kan det være:**
    - **Tegning**

Vi har valgt at gøre brug af datatyperne int og String og så lave forskellige template som så indeholder en html sætning f.eks. <rect.../> som så kan blive brugt i en anden klasse hvor vi giver den parametrene.

    - **Stykliste beregner**

Vi har haft mange uforventede forhindringer i vores forsøg på, at kode stykliste beregneren. Derfor har vi ikke kunne færdiggøre den del af projektet.

Husk: det er bedre med 2 linjers dokumentation end ingen.

Udvalgte kodeeksempler

SVG.java

```java
package business.services;

public class SVG
{
    StringBuilder svg = new StringBuilder();

    private int x;
    private int y;
    private String viewBox;
    private int width;
    private int height;

    private final String headerTemplate = "<svg height=\"%d%%\" "  +
            "width=\"%d%%\" "  +
            " viewBox=\"%s\" "+
            "x=\"%d\" "+
            "y=\"%d\" "+
        " preserveAspectRatio=\"xMinYMin\">";

    private final String rectTemplate = "<rect x=\"%d\" y=\"%d\" height=\"%d\" width=\"%d\" style=\"stroke:#000000; fill: #ffffff\"/>";

    private final String lineTemplate = "<line x1=\"%d\" y1=\"%d\" x2=\"%d\" y2=\"%d\" style=\"stroke:#000000;\"/>";

    private final String textvandretTemplate = " <text style=\"text-anchor: middle\" x=\"%d\" y=\"%d\">%s</text>";

    private final String textlodretTemplate = "<text style=\"text-anchor: middle\" transform=\"translate(%d,%d) rotate(-90)\">%s</text>";

    private final String stippledelineTemplate = "<line stroke-dasharray=\"5, 5\" x1=\"%d\" y1=\"%d\" x2=\"%d\" y2=\"%d\" stroke=\"black\" />";

    public SVG(int x, int y, String viewBox, int width, int height) {
        this.x = x;
        this.y = y;
        this.viewBox = viewBox;
        this.width = width;
        this.height = height;
        svg.append(String.format(headerTemplate, height, width, viewBox, x, y));
    }

    public void addRect(int x, int y,int height, int width)
    {
        svg.append(String.format(rectTemplate, x, y, height, width));
    }

    public void addLine(int x1, int y1, int x2, int y2) { svg.append(String.format(lineTemplate, x1, y1, x2, y2)); }

    public void addSvg(SVG innerSVG) { svg.append(innerSVG.toString()); }

    public void addvandretText(int x,int y, String text) { svg.append(String.format(textvandretTemplate,x,y,text)); }

    public void addlodretText(int x,int y, String text) { svg.append(String.format(textlodretTemplate,x,y,text)); }

    public void addstippledeline(int x1, int y1, int x2, int y2) {...}

    @Override
    public String toString() { return svg.toString() + "</svg>" ; }
}
```

Det er ikke sikkert at censor (eller eksaminator) finder alle jeres guldkorn i selve koden. Derfor er det en god ide at vælge særlige kode stumper ud og vise dem i rapporten.

De eksempler der er givet under “særlige forhold” afsnittet kan man godt tage og illustrere med kode direkte i rapporten.

Det kommer til at virke særligt overbevisende hvis den kode man vælger ud indgår som led i et af sekvensdiagrammerne.

Der er mange af jer der vil skrive jeres ting i Word eller google docs. Vær opmærksom på hvordan i formaterer jeres kode. Man vælger ofte en lidt mindre font, en der er “monospaced” (alle bogstaver optager samme bredde). Der er også nogle der sætter små skærmbilleder fra IntelliJ ind. Det er OK, men så husk at vælge et tema fra IntelliJ med hvid baggrund og mørke/farvede bogstaver da nogle censorer skriver rapporten ud på blækprintere som ikke gengiver lyse bogstaver på sort baggrund særligt godt.

Status på implementering

Dette afsnit skal gøre rede hvor langt I er nået med implementeringen. Typiske ting man kan have sprunget over er:

- **Man har ikke nået at lave alle de jsp sider man har med i navigations diagrammet.**

Vi har ikke nået at kunne vise stykliste beregneren(det output den skulle lave) og heller ikke vise den dynamiske SVG-tegning.

- **Man har ikke nået at lave alle CRUD metoderne til alle tabellerne**

Vi har noget C (Create) og noget R (Read), men vi har ikke U (Update) og vi har ikke D (Delete).

- **Man har ikke fået pynte sine sider**

Vi har pyntet vores sider til perfektion.

- **Man har fundet en fejl “i sidste øjeblik”, men har ikke haft tid til at rette det. - F.eks. at man har brugt session forkert sådan at man på en af siderne kan komme ind uden at være logget in.**

Vi har ikke Delete og Update metoder.

- **tests der fejler på afleveringstidspunkt**

Vi har ingen tests der fejler på afleveringstidspunktet.

- **… andre mangler**

Vi mangler beregning af materialerne og gøre svg dynamisk og at kunne betale for sin carport.

Test

Der skal være lavet test. Du kan dokumentere tests ved at beskrive i tabelform:

- **Hvilke klasser er testet**

Den klasse som vi har testede er MaterialMapper.

- **Hvilke metoder der er testet**

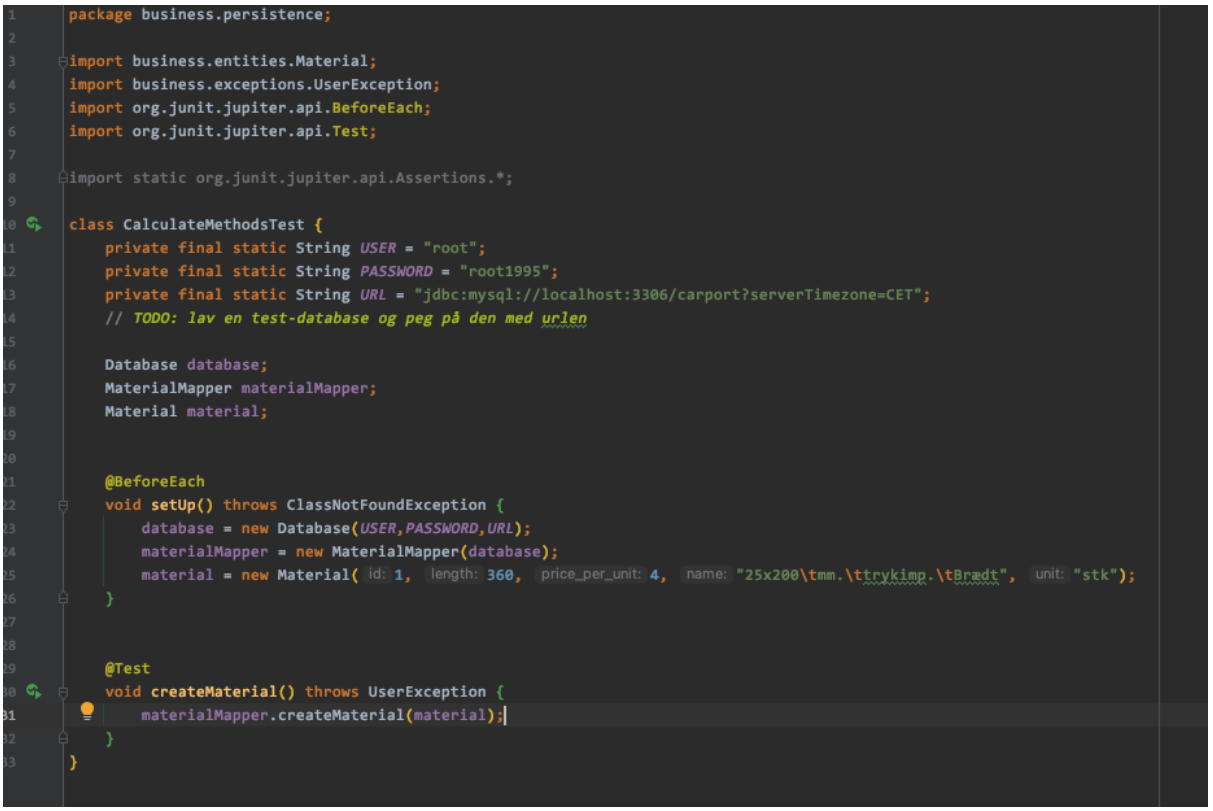
Den metode som blev testede er createMaterial.

- **Dækningsgrad af dine tests for de valgte metoder og klasser**

Vi har 100% dækningsgrad af vores testede metoder og klasser.

- **Desuden kan du beskrive hvordan i systematisk har arbejdet med at teste koden før den er blevet gjort til en del af master branch.**

Vi har ikke haft synderligt meget systematisk arbejde i forhold til, at teste koden før den blev gjort en del af master branchen. Vi har i det sidste sprint fokuseret en stor del af arbejdskraften på stykliste beregneren.

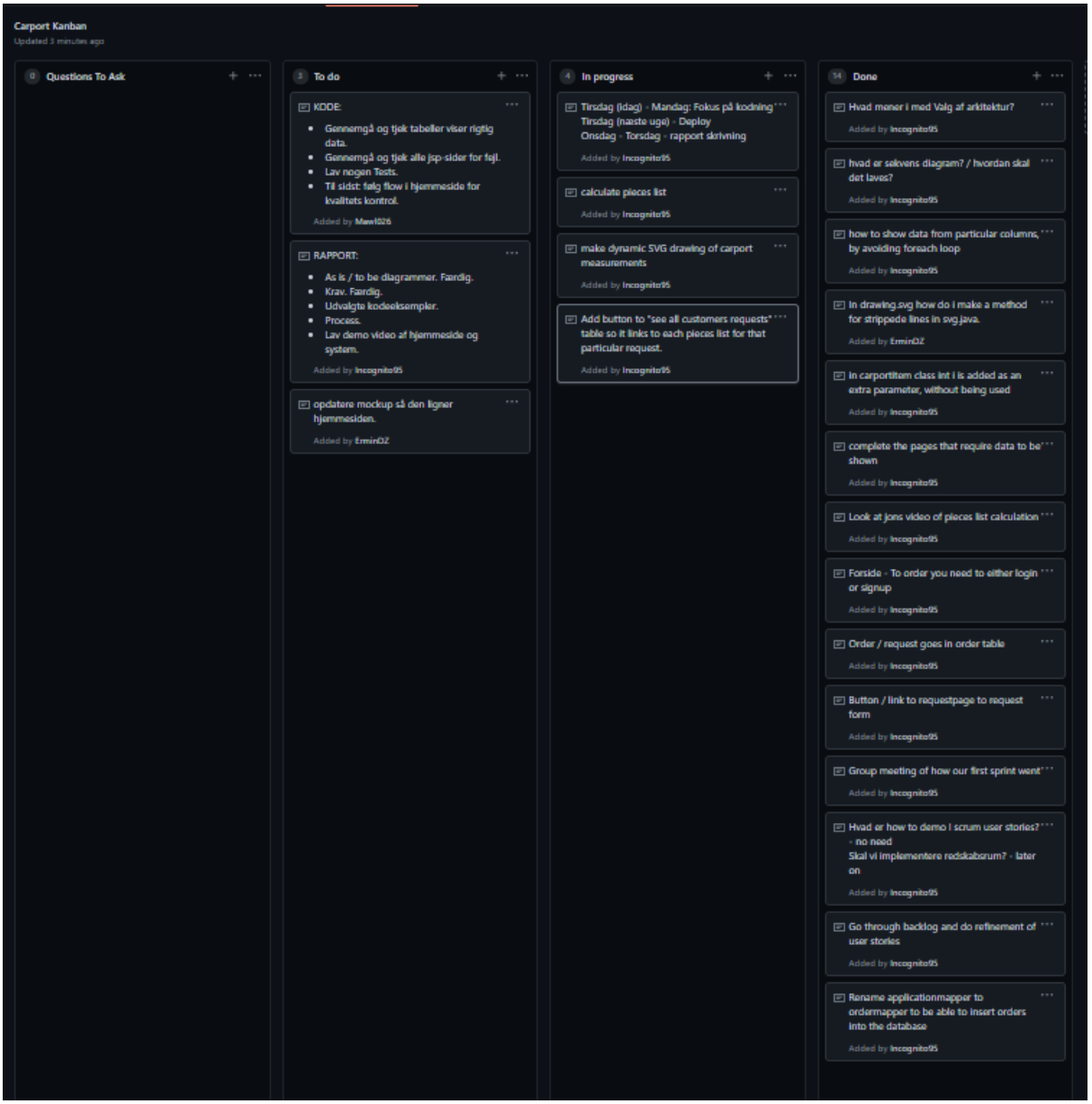


```
package business.persistence;

import business.entities.Material;
import business.exceptions.UserException;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

class CalculateMethodsTest {
    private final static String USER = "root";
    private final static String PASSWORD = "root1995";
    private final static String URL = "jdbc:mysql://localhost:3306/carport?serverTimezone=CET";
    // TODO: lav en test-database og peg på den med urlen

    Database database;
    MaterialMapper materialMapper;
    Material material;


    @BeforeEach
    void setUp() throws ClassNotFoundException {
        database = new Database(USER,PASSWORD,URL);
        materialMapper = new MaterialMapper(database);
        material = new Material( id: 1,  length: 360,  price_per_unit: 4,  name: "25x200\tmm.\ttrykimp.\tBrædt",  unit: "stk");
    }


    @Test
    void createMaterial() throws UserException {
        materialMapper.createMaterial(material);
    }
}
```

Proces

Der skal være et afsnit hvor I beskriver jeres arbejdsproces i projektperioden. Der skal dels være et faktuelt afsnit og et refleksionsafsnit. Nævn f.eks. også hvilke værktøjer I har brugt til at styre processen, og indsæt gerne skærmdumps til at vise det. Det kunne være GitHub Issues etc.

Vi har valgt at gøre brug af github kanban board, her under vises et skærmdump af det. Det findes også inde på vores github.



Arbejdsprocessen faktuelt

Dette afsnit skal beskrive:

- **Hvilke sprints der var, og hvilke user stories der blev arbejdet med.**

De sprints og user stories, som vi har arbejdet med, er dem som findes i Scrum User Stories afsnittet.

- **Hvem der evt. var Scrum Master i hvilke dele af projektperioden. Giv gerne nogle eksempler på hvad SM gjorde i udvalgte sprint.**

Scrum master i første sprint var Ermin. Ermin fokuserede på, at få databasens struktur rigtigt fra start og have resten af den grundlæggende struktur klar. Vi gik i gang med alle de diagrammer der skal indgå i opgaven. Hertil var Adobe XD mockup en vigtig del af hvordan vi senere hen opsatte vores jsp-sider.

Scrum master i andet sprint var Jens. Jens fokuserede på, at starte med koderiet. Vi startede med, at kode med udgangspunkt i den mockup vi havde lavet i første sprint, som så blev brugt til inspiration for koden til vores JSP-sider. Her startede vi også med, at kigge frem mod beregningen af stykliste.

Scrum master i tredje sprint var Daniel. Daniel fokuserede på, at vi skulle lave vores SVG-tegning og starte med, at kode beregningen af stykliste. Her blev alt vores energi lagt i hele sprintet. Der stødte vi alle på store problemer, som holdte os tilbage. Vi prøvede mange forskellige løsninger og tror vi kom tæt på, men kunne i sidste ende ikke færdiggøre beregning af styklisten.

- **Et eksempel på et af PO-møderne, hvad der var planlagt fra jeres side, og hvordan det gik.**

I det næstsidste PO-møde med Palle havde vi planlagt, at vi ikke skulle lave SVG-tegning og vi skulle fokusere alt vores tid på beregning af styklisten. På grund af uforudsete forhindringer kunne vi ikke komme videre med beregning af stykliste ligemeget hvor meget vi prøvede. Så det endte med, at vi gjorde det præcist modsatte af hvad der var planlagt og lavede vores statiske SVG-tegning, da vi ikke kunne komme videre med beregning af stykliste.

PO-Møder

Første PO-møde:

Vi aftalte med product-owner om at vi skal implementere så man kan oprette en konto eller logge ind i sin konto samt at kunden kan se sine forespørgsler inde på hjemmesiden.
Vi aftalte med product-owner om at vi skal implementere så kunden kan sende en forspørgelse på en carport med kundens ønskede dimensioner.

Anden PO-møde:

Vi aftalte med product-owner, at vi skal sætte data ind i vores database og rette til i vores database struktur til næste PO-møde.
Vi aftalte med product-owner at vi skal lave vores EER-diagram færdig til det næste PO-møde.
vi aftalte med product-owner at vi skulle begynde på rapport og dokumentering af vores arbejde til næste PO-møde.

Tredje PO-møde:

Vi aftalte med product-owner, at vi skal begynde med beregning af stykliste.
Vi aftalte med product-owner, at vi skal begynde med tegning af svg.

- **Hvordan i afholdt jeres Daily Scrum Meetings**

Vi aftalte at mødes hverdage, weekender og helligdage kl. 10:00.
Når vi så var i forsamling på vores fortrukne kommunikationsmedie, discord, begyndte vi med, at evaluere hvilke opgaver der lå for dagen, sætte dem i rækkefølge efter prioritet.
Herefter begyndte selve arbejdet og om vi fandt evt. ud af om vi havde brug for rådgivning af tutor eller undervisere.
Hvis vi stødte på fejl eller tekniske problemer blev disse sat, som førsteprioritet hvis det hastede. Hvis vi ikke lige kunne få hjælp, fortsatte vi selvfølgelig arbejdet på andre dele af projektet for at minimere tidstab. Hvis vi blev færdig med noget klappede vi hinanden på skuldrene, talte om hvad der gik godt og dårligt, planlagde hvad næste opgave var og så fortsætte arbejdet.
Dette var vores daglige scrum møder.

- **Hvornår I holdt Scrum Retrospectives.**

Vi afholdte møde efter hver arbejdsdag, hvor vi samlede vores branches commits i main branch og planlagde hvad vi havde fået lavet, hvad vi havde problemer med, hvad vi skulle klargøre til næste dag

Arbejdsprocessen reflekteret

Dette afsnit skal beskrive jeres overvejelser over hvilke dele der har fungeret godt og hvilke dele der måske er faldet lidt på gulvet. I kan f.eks. beskrive:

- **Om Scrum Master rollen fungerede, hvilke problemer I så i den, og hvad I gjorde for at rette op på det.**

Scrum Master rolle gik på skift mellem medlemmerne i vores studiegruppe. Vi støtte alle på problemer med enten kode eller bare gode gammeldaws' tekniske problemer. Når vi så stødte på disse virvar af problemer, var vi alle sammen gode til at, supplere den regerende Scrum Master, ved at gå sammen, 3 hjerner om ét problem så det kunne blive løst hurtigt og effektivt - så vi alle kunne komme videre med vores individuelle opgaver, som var bestemt af Scrum Master med rådgivning fra resten af studiegruppen.

- **Hvad der var de væsentligste emner på jeres retrospektiv møder**

De væsentligste emner på vores retrospektiv møder var links, arbejdsgange der skal IT-stottes, udvalgte kodeeksempler, test og proces.

- **Om I havde problemer med at nedbryde user stories i tasks og hvad I gjorde for at løse dem**

I starten af projektet havde vi nogle få problemer med at forstå omfanget af opgaven og de user stories der hørte med. Vi gennemgik opgavebeskrivelsen og de tilhørende videoer, som Jon havde lavet. Dette hjalp gevaldigt og sammen med et møde med underviser, som gjorde det nemmere for os, at forstå projektet. Næst begyndte vi med formulering og indsnævring af user stories til opgaven for, at give bedre overblik. Derefter begyndte vi på, at nedbryde user stories, for at kunne få nogle konkrete tasks, som vi kunne arbejde ud fra.

- **Om I var spot-on med jeres estimeringer og hvad der mon var årsagen eller hvad I gjorde hvis I ikke var spot on**

Vi undervurderet estimering på beregning af stykliste. Vi havde en nogenlunde estimering af tid på beregningerne af vores forskellige tasks. Vi undervurderede dog hvor meget faktisk tid vi skulle allokere til, at lave stykliste beregneren.

- **Om der var problemer med vejledningen og PO-møderne**

I løbet af de forskellige sprints i projektet havde vi gode møder med PO og fik god vejledning i de møder. Vi havde dog lidt udfordringer med, at forstå hvordan stykliste beregneren skulle laves når vi prøvede, at få vejledning ved siden af møder med PO.

- **Hvor langt inde i processen I fandt en rytme der var produktiv og om meget gerne identificere hvorfor**

Vi begyndte at finde en rytme i arbejdsprocessen der var produktiv i starten af andet sprint. Her havde vi lige færdiggjort alle JSP-sider og fik bedre overblik over hele projektet. Dette resulterede i vi bedre kunne se mangler i koden og nemmere kunne se hvor vi kunne forbedre koden.

- **Andre elementer der har at gøre med at forsøge at arbejde i et Scrum team**

Når man arbejde i et Scrum team er det rigtig nemt, at kommunikere effektivt og supplere hinandens forskellige opgaver når man støder på forhindringer.