Fog Carports

Dette projekt er lavet af Gruppe 'F' bestående af:

Israa El-Haj Moussa - cph-ie51@cphbusiness.dk - github.com/israa1809

Malene Rolsted Christensen - cph-mc452@cphbusiness.dk - github.com/malenec

Mark Pihl Kousgaard Pedersen - cph-mp677@cphbusiness.dk - github.com/Marpeddata

Sebastian Timothy Berry - cph-sb468@cphbusiness.dk - github.com/berry1196

Martin Hougaard Lorentzen - cph-ml724@cphbusiness.dk - github.com/zendary

Link til pitch video:

https://youtu.be/Lu9FBsNEvFs

Link til demo af website:

https://youtu.be/T02184_cOAo

Link til projekt på github:

https://github.com/Israa1809/Carport

Link til hjemmeside:

Hjemmeside: http://167.172.183.202:8080/carport/

Login bruger: dit ordrenummer
kode bruger: dit telefonnummer
Login admin: Admin
Kode admin: 1234

Dette projekt blev lavet:

Fra den 28. november 2022 til den 4.Januar 2023

Fog Carports **1**

Indledning **3**
Baggrund 3

Virksomheden/Forretningsforståelse **3**

SWOT **3**

Krav **5**
User stories 8
Task udklip fra User Story 10

Domæne model og EER diagram **11**

Navigationsdiagram & Mockups **12**

Navigationsdiagram **12**

Valg af arkitektur **18**

Egne Regler **18**

Særlige forhold **19**

Udvalgte kodeeksempler **20**

Status på implementering **22**

Test **25**

Proces **27**
Arbejdsprocessen faktuelt 27
Arbejdsprocessen reflekteret 29

Indledning

Johannes Fog har behov for at få moderniseret deres website, hvor de bl.a. sælger carporte, så de kan få rationaliseret deres forretnings flow.
Vi har fået til opgave, at udvikle et system i form af et website til salg af skræddersyede carporte.
Ud fra interviewet med Martin fra Johannes Fog får vi oplyst, at systemet skal overholde nogle krav. Dette kunne være, at kunden skal kunne vælge nogle bestemte mål også få designet deres carport. Systemet skal kunne udregne hvor mange materialer kunden skal bruge til at bygge sin carport. Admin skal kunne se alle kundeordrer, samt tilføje nye materialer til databasen.
Systemets backend skal bygges med Java, Java Servlets og bruge en mySQL database. Frontend skal designes med Bootstrap 5.0 og i JSP sider. Der skal bl.a kunne tegnes en carport og dette bliver gjort ved hjælp af SVG.

Baggrund

Johannes Fog er en tømmerhandel, der har en stor boligafdeling, og de yder rådgivning med alt indenfor byggeri.
Johannes Fog hjælper sine kunder ved, at rådgive kunden til bedste løsning og bedste materialer. Deres rådgivning bruges ofte, når kunden vil bygge eller designe noget indenfor hus, carport, hegn og terrasser. Johannes Fog har både en Bolig & Design afdeling, Trælast & Byggecentre afdeling. Det er Trælast & Byggecentre afdeling, som skal have bygget et system.

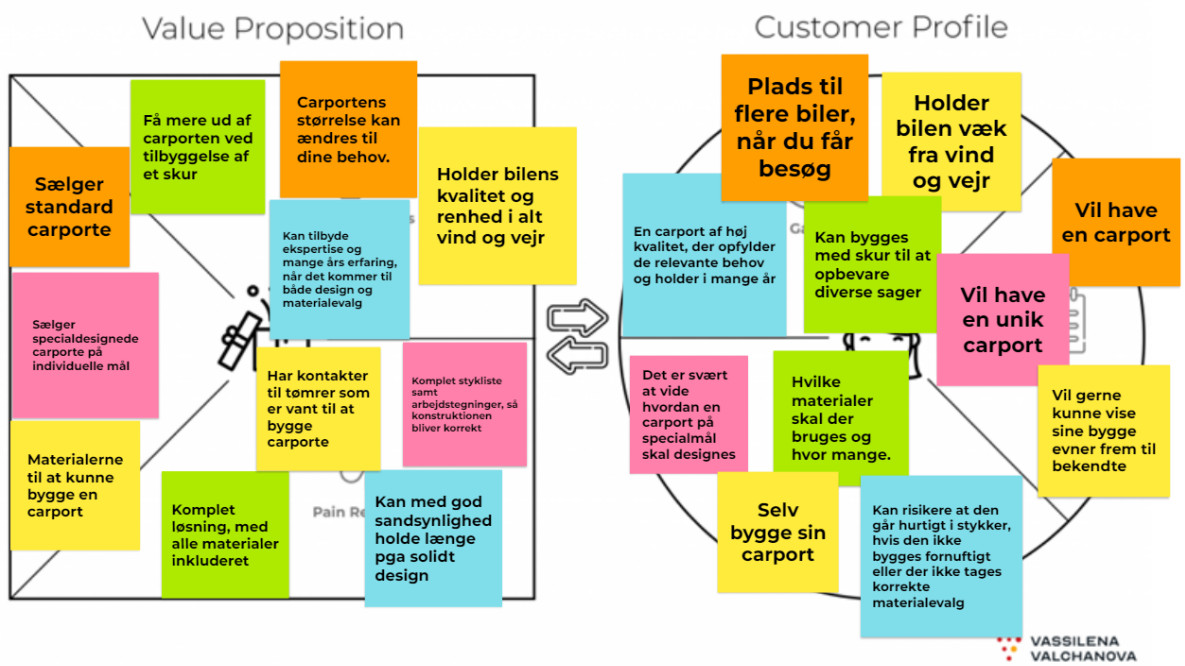
Virksomheden/Forretningsforståelse

SWOT

På vores gruppe.

Strengths	Weaknesses
1. Højt engagement hos alle gruppens deltagere 2. Mange øjne på tingene. 3. Vi er hver især dygtige til forskellige ting - og hvis ingen ved det, er vi gode til ikke at give op, og løse problemerne sammen 4. Gode til at skrive rapport	1. Alle i gruppen har svært ved at “gå glip” af noget 2. Koordinering af arbejdstider er en udfordring pga det høje antal gruppemedlemmer 3. Opdeling af arbejdsopgaver der ikke må overlappe, men samtidig skal alle også have noget at lave
Opportunities	**Threats**
1. Lærer nye arbejdsprocesser og værktøjer at kende gennem projektet. 2. Sparring med andre grupper	1. Sygdom eller syge børn 2. Brugte systemer stopper med at virke og/eller data mistes. 3. Hacking

I vores gruppe har vi alle svært ved at gå glip af noget, og det gør os højt passionerede og engageret for at lave et top kvalitets projekt, som vi alle kan føle os stolte af. Koordinering af arbejdet tænker vi kan blive svært med fem gruppemedlemmer, da mange i gruppen har børn og arbejde. Fordelen ved at være mange i gruppen er, at det ofte er hurtigere, at finde fejlen når man spørger om hjælp. Vi tænker, at det kan være svært at opdele arbejdsopgaver, da de nemlig ikke må overlappe hinanden, være for store eller være for små. Med tre forældre i gruppen har vi førhen oplevet, at syge børn oftest har sat arbejdet på pause. Igennem projektet tænker vi, at lære ud fra truslen og lære nye arbejdsprocesser som kan optimere vores arbejde. Sparring med andre grupper vil give os en bedre forståelse. Hacking og hvis programmer som vi bruger, pludselig får en update eller stopper med at virke, så kan det sætte os bagud.



Vores tanker omkring at arbejde med vpc har været, at det både er sjovt og udfordrende, da man skal tænke som kunde og firma.

Firmaet skal have fokus på hvilket produkt der tilbydes. Der skal tænkes over hvilket arbejde kunden vil have løst, hvilke smerter de kan have under processen i at anskaffe sig en carport, og hvilke fordele der kan være ved at anskaffe sig en carport. På samme tid skal man, som firma kunne afløse kundens smerter, og genererer en masse fordele der kan få overtalt kunden til at købe en carport.

Et eksempel kunne være, at en kunde vil have en unik carport, der holder hans bil væk fra vind og vejr. Kunden tænker, at det er svært at få designet en carport med specifikke mål. Dette løser firmaet ved at give en komplet stykliste samt arbejdstegninger, så konstruktionen bliver korrekt.

Teknologivalg

Programmer og udviklingsværktøjer:

- IntelliJ IDEA Ultimate 2021.2.4
- Java SDK 11
- MySql Server 8.0.30
- MySql Workbench 8.0.30
- Tomcat Webcontainer 9.0.67
- GIT og Github

Programmeringssprog, biblioteker med mere:

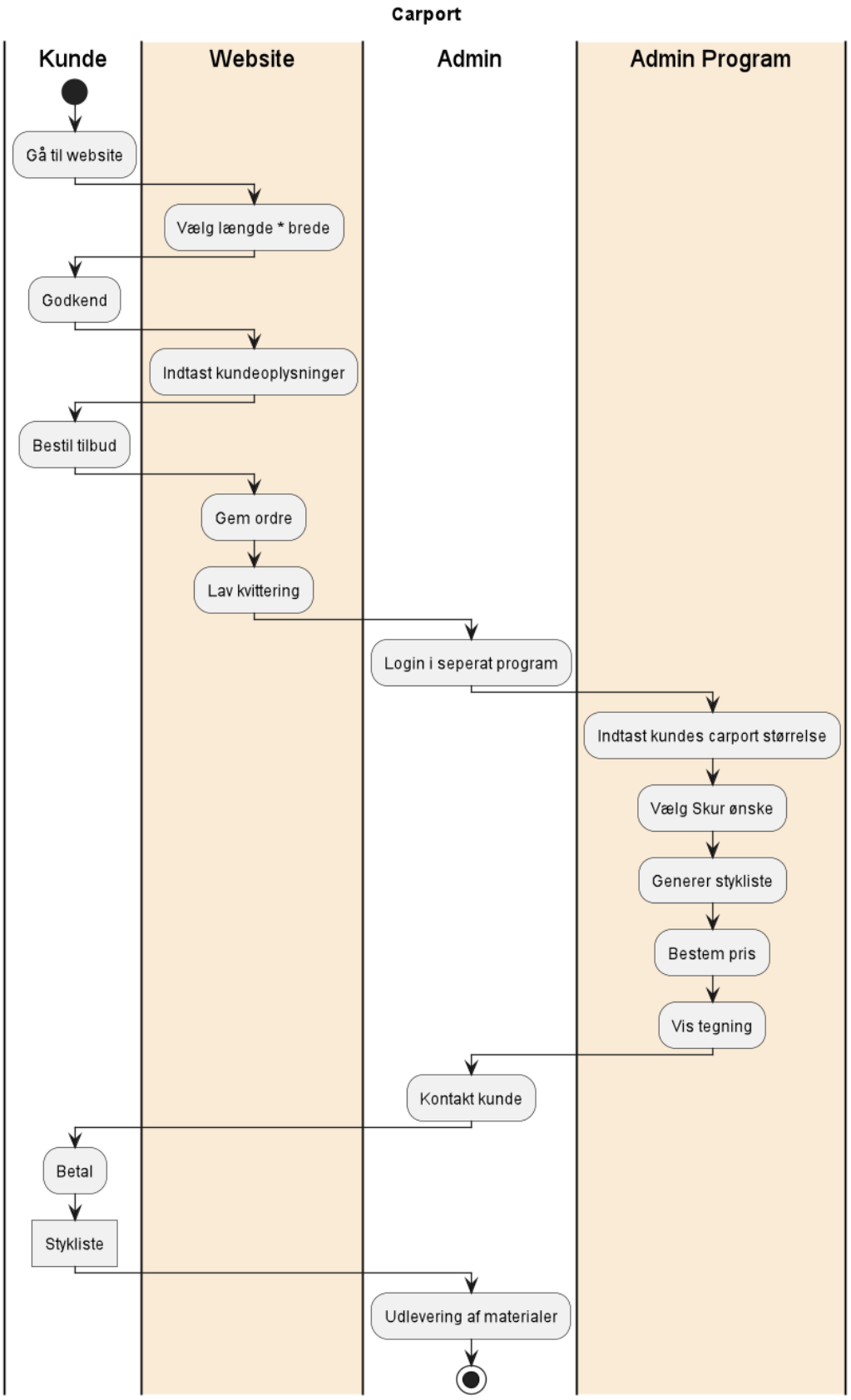
- Java / JSP / JSTL
- HTML 5
- CSS 3.0
- Twitter Bootstrap 5.0

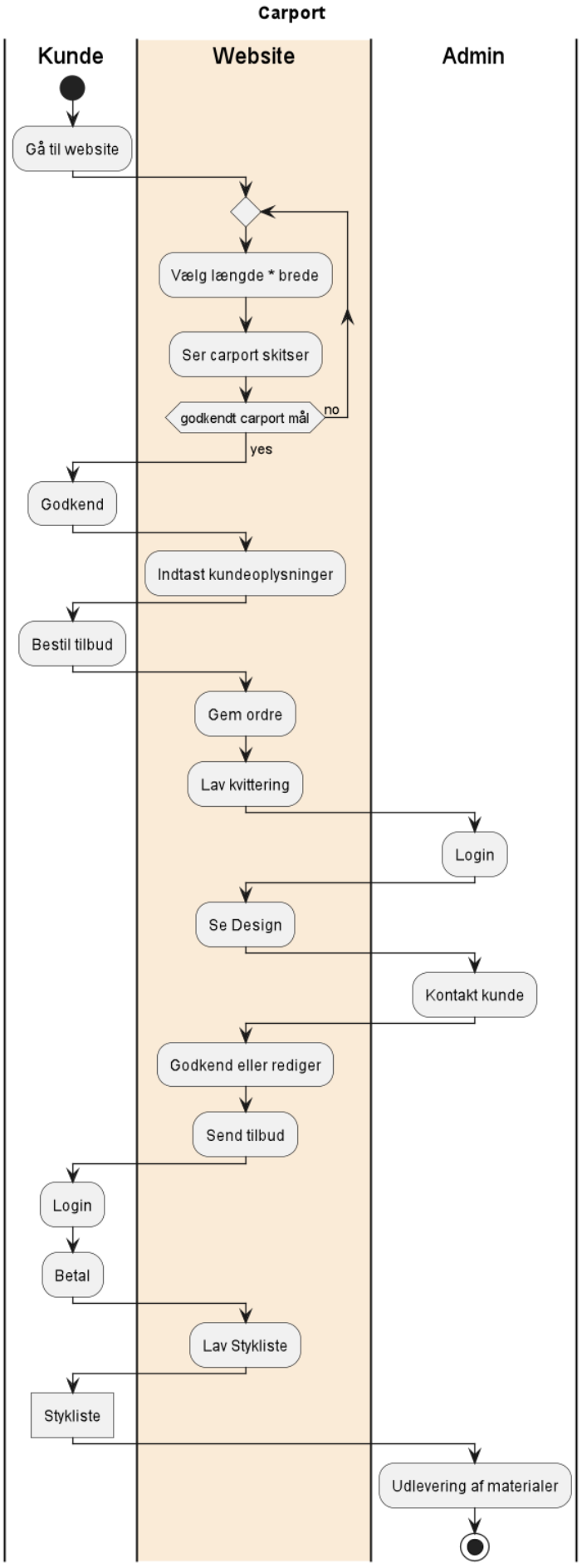
Planlægning og Rapportskrivning:

- Google Docs
- Trello - til projektstyring
- Draw.io og PlantUML til diagrammer
- Github - til versionering og deling af kode
- Figma - til Mockups
- Discord - til remote samarbejde

Krav

De følgende 2 sider viser først det nuværende aktivitetsdiagram for Fog efterfulgt at det nye aktivitetsdiagram vores løsning skaber.
Den helt store forskel ses i, at vi tager hele det eksterne program ud, der kræver at admin indtaster alle de oplysninger kundens allerede har valgt en gang.
I stedet generer hjemmesiden nu automatisk en stykliste, pris og tegning ud fra de oplysninger kunden allerede har valgt.
Dette gavner også kunden, da de allerede kan se tegningen for carporten, når de vælger den.
Admin har stadig adgang til styklisten, men kunden får den først udleveret automatisk efter at carporten er betalt for.


Carport
Kunde
Website
Admin
Admin Program
Gå til website
Vælg længde * brede
Godkend
Indtast kundeoplysninger
Bestil tilbud
Gem ordre
Lav kvittering
Login i seperat program
Indtast kundes carport størrelse
Vælg Skur ønske
Generer stykliste
Bestem pris
Vis tegning
Kontakt kunde
Betal
Stykliste
Udlevering af materialer


Carport
Kunde
Website
Admin
Gå til website
Vælg længde * brede
Ser carport skitser
godkendt carport mål
no
yes
Godkend
Indtast kundeoplysninger
Bestil tilbud
Gem ordre
Lav kvittering
Login
Se Design
Kontakt kunde
Godkend eller rediger
Send tilbud
Login
Betal
Lav Stykliste
Stykliste
Udlevering af materialer

User stories

US-1: Som kunde ønsker jeg at kunne indsætte mine egne mål, så systemet beregner hele konstruktionen af min carport.

AC-1: Givet at jeg bliver præsenteret for et antal felter til at udfylde carportens mål og kan trykke på en “Beregn” knap, så kan jeg på den efterfølgende side tilgå alle carportens informationer inklusiv dens mål, pris samt den samlede stykliste.

Size-1: Large

US-2: Som kunde ønsker jeg at se en tegning af min carport, som hjælper mig med at visualisere mit design, så jeg kan redigere målene indtil jeg er tilfreds

AC-2: Givet at jeg bliver præsenteret for en tegning, når jeg indsætter mine egne mål i de givne felter, så jeg kan blive præsenteret for en tegning af carporten, med mulighed for at trykke på tilbage knap, hvis jeg ikke er tilfreds.

Size-2: Large

US-3: Som kunde ønsker jeg at kunne godkende mit design, så jeg kan bestille et tilbud

AC-3: Givet at jeg bliver præsenteret for en knap, når jeg har valgt de mål, der giver en tegning af en carport, som jeg er tilfreds med, så kan jeg trykke godkend og få mulighed for at bestille et tilbud på det design.

Size-3: Small

US-4: Som kunde ønsker jeg at kunne indtaste mine kontaktoplysninger, så jeg kan modtage et tilbud på den carport, jeg er kommet frem til passer til lige præcis mit behov.

AC-4: Givet at jeg bliver præsenteret for et antal felter til at udfylde kontaktoplysninger, når jeg trykker på ‘Godkend design’-knappen, så kan jeg indtaste mine informationer og trykke på en ‘Bestil tilbud’-knap.

Size-4: Medium

US-5: Som kunde ønsker jeg at blive præsenteret for en bekræftelses-side, når jeg har trykket ‘Bestil tilbud’, så jeg ved at bestillingen er blevet sendt afsted, og jeg kan se mit ordrenummer.

AC-5: Givet at jeg bliver præsenteret en ‘Bestil tilbud’-knap, når jeg har udfyldt mine kontaktoplysninger, så bliver jeg ved at trykke på knappen sendt videre til en bekræftelsesside, der viser mit ordrenummer.

Size-5: Small

US-6: Som admin ønsker jeg at kunne logge ind, så jeg kan få adgang til de relevante sider min rolle indebærer.

AC-6: US-6: Givet at jeg bliver præsenteret for en mulighed for at trykke ’Log ind’, når jeg går ind på web applikationer, så kan jeg som admin logge ind på en del af systemet, som kun jeg har adgang til vha. mit unikke brugernavn og adgangskode, hvor jeg bliver præsenteret for et antal sider dedikeret til administration af systemet og kundeordrer

Size-6: Small

US-7: Som admin ønsker jeg at kunne se en kundes bestilling, så jeg enten kan godkende bestillingen eller kontakte kunden gennem mail/telefon og yde rådgivning angående designet.

AC-7: Givet at jeg bliver præsenteret for et antal sider, når jeg jeg er logget ind på systemet som admin, så kan jeg trykke på den specifikke side, der viser mig hvilke bestillinger systemet har taget imod fra kunder samt har en knap hvor jeg kan trykke 'Send tilbud', når jeg har vurderet og godkendt designet i samråd med kunden

Size-7: Medium

US-8: Som kunde ønsker jeg at kunne tilgå en personlig side, så jeg kan se et tilbud, når designet er godkendt af admin.

AC-8: Givet at jeg bliver præsenteret for min ordrer side, når jeg er logget ind med min ordrenummer account, så kan jeg se en pris på et tilbud stillet af admin, efter de har godkendt mit carport design og opstillet et tilbud til mig.

Size-8: Medium

US-9: Som kunde ønsker jeg at kunne betale for min carport, så styklisten til min carport også vises på min personlige side.

AC-9: Givet at jeg bliver præsenteret for min ordrer side, når jeg er logget ind med min ordrenummer-account, så jeg kan trykke på en betalingsknap, der herefter tager mig videre til en side der viser mig styklisten på materialerne til min carport.

Size-9: Medium

US-10: Som admin ønsker jeg at kunne se alle oplysninger omkring de tilgængelige materialer, der ligger i databasen, så jeg kan se om det stemmer overens med lageret.

AC-10: Givet at jeg bliver præsenteret for et antal sider, når jeg jeg er logget ind på systemet som admin, så kan jeg trykke på den specifikke side, der viser mig en liste over alle materialer hentet op fra databasen.

Size-10: Small

US-11: Som admin ønsker jeg at kunne slette en vare i de tilgængelige materialer, der ligger i databasen, så jeg kan sørge for at det stemmer overens med lageret.

AC-11: Givet at jeg bliver præsenteret for en liste over materialer, når jeg jeg er logget ind på systemet som admin, så kan jeg trykke på den specifikke side, her kan jeg trykke på en knap der sletter varen fra databasen.

Size-11: Small

US-12: Som admin ønsker jeg at kunne oprette en ny vare i de tilgængelige materialer, der ligger i databasen, så jeg kan sørge for at det stemmer overens med lageret.

AC-12: Givet at jeg bliver præsenteret for en liste over materialer, når jeg jeg er logget ind på systemet som admin, så kan jeg trykke på den specifikke side, her kan jeg trykke på en knap der tager imod input så nye materialer kan tilføjes til databasen.

Size-12: Small

US-13: Som admin ønsker jeg at kunne ændre oplysningerne på en vare i de tilgængelige materialer, der ligger i databasen, så jeg kan sørge for at det stemmer overens med lageret.

AC-13: Givet at jeg bliver præsenteret for en liste over materialer, når jeg jeg er logget ind på systemet som admin, så kan jeg trykke på den specifikke side, her kan jeg trykke på en knap der redigerer allerede eksisterende materialer i databasen.

Size-13: Medium

US-14: Som admin ønsker jeg at kunne se prisen på den givne carport, så jeg kan ændre i den pris der bliver sendt i tilbuddet til kunden

AC-14: Givet at jeg bliver præsenteret for en 'Rediger pris'-knap, når jeg er inde på en side for en specifik kunde ordre, så bliver jeg sendt til en side, der tillader at jeg kan ændre og opdateret den samlede pris, som kunden får i sit tilbud

Size-14: Small

US-15: Som admin ønsker jeg at kunne ændre målene på en given kundes carport, så designet bliver mere hensigtsmæssigt til kundens behov

AC-15: Givet at jeg bliver præsenteret for en 'Rediger design'-knap, når jeg er inde på en side for en specifik kunde ordre, så bliver jeg sendt til en side, der tillader at jeg kan ændre og opdatere målene på carporten, som kunden derefter i stedet vil kunne se på sin ordre

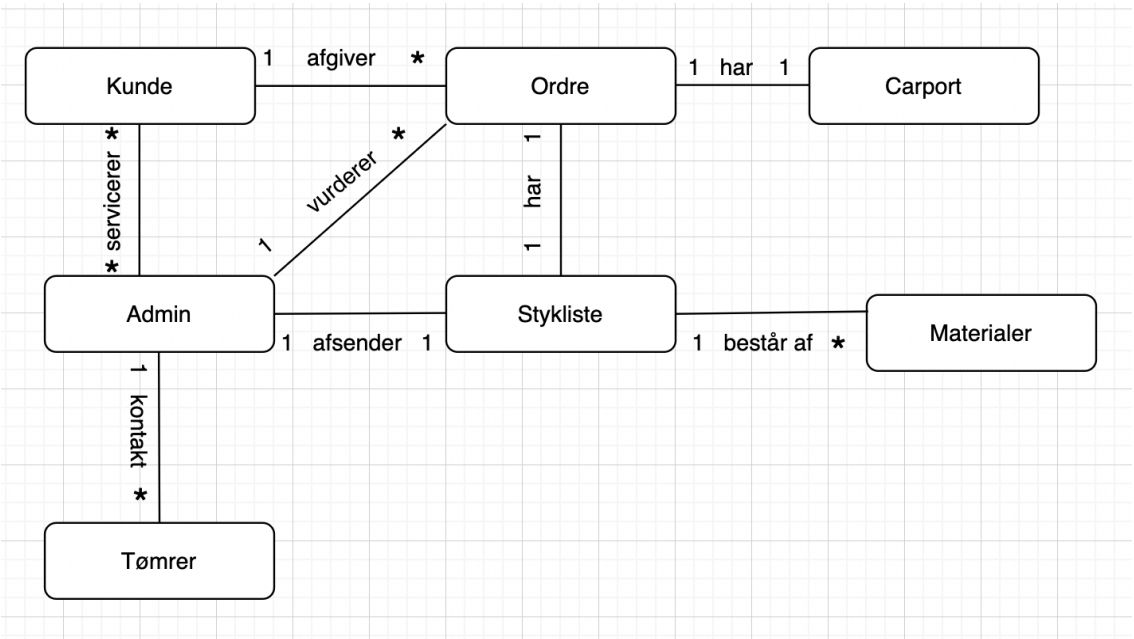
Size-15: Medium

Task udklip fra User Story

Her ses et par stykker af vores tasks vi lavede til userstories.
Det har været ret forskelligt i hvor stor en mængde af tasks der kan oprettes til hver user story. I det tilfælde jeg viser her er nogle user stories også afhængige af den samme jsp side, hvor vi så har givet dem en fælles task.

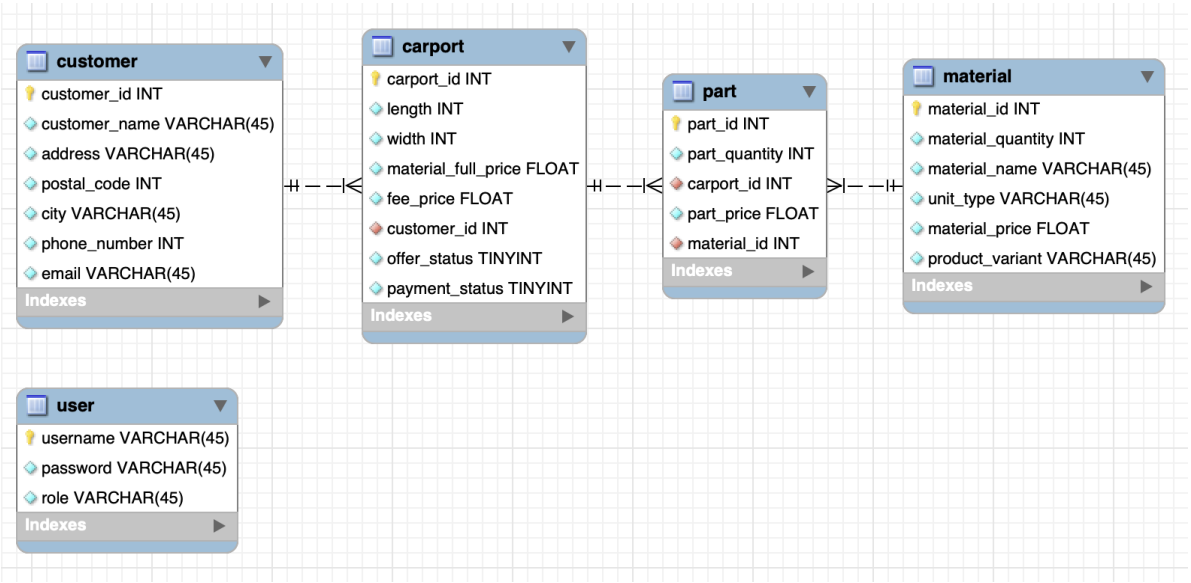
Domæne model og EER diagram

Domænemodel



Herover ses vores domænemodel som er produceret gennem en analyse af Fog som leverandør af carporte. vi har har samlet hvilke elementer der indgår i forbindelsen med producering af en færdig carport og sat dem i relation til hinanden.

EER-diagram

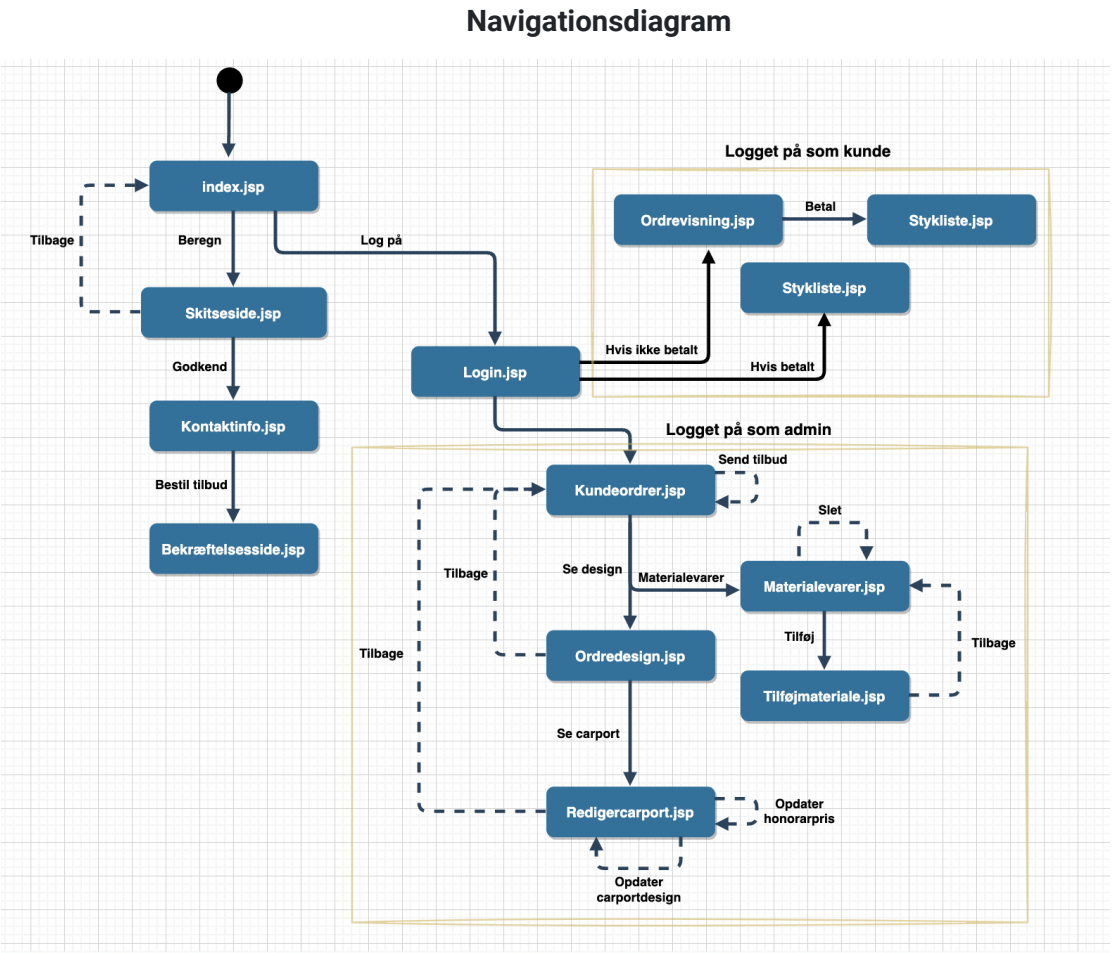


I vores design af databasen har vi taget følgende valg som også ses på vores EER diagram herover. Postal code, Unit type som er en mængde betegnelse(cm, stk, kasse osv) og Product variant, kunne alle være designet efter 3 normal form, men vi har her valgt ikke at

gøre det for at spare tid, så vi kan nå frem til et færdigt produkt hurtigere samt gøre vores database mere simpel til en begyndelse.

Vi gør brug af fremmednøgler i vores 1 til mange relationer på carport som har fremmednøglen customer_id som genereres automatisk i customer tabellen, vores brug af fremmednøgler bliver dog særligt spændende på tabellerne carport, part og materiale. Vi vælger at bruge Part både som en samler tabel men også til at holde styr på mængden af et givent materiale til styklisten til carporten. Part har en fremmednøgle carport id, som den får fra carport id på carport tabellen, denne autogenereres og er den vi kommer til at bruge som ordrenummer i vores design af det færdige system. På denne måde defineres hvilken ordre en "part" høre til. Vi ved at data på materialet er konstante og ikke skal ændres dig igennem et carport design, men vi har samtidig behov for at vide hvor mange af et givent materiale der skal benyttes på et carport design, så vi har lagt en fremmednøgle på part tabellen kaldet material id, så et materiale kan defineres på en part. På denne måde sikre vi materialets værdier forbliver konstante, men vi kan samtidig gemme en samlet pris såvel som mængden af det pågældende materiale i databasen.

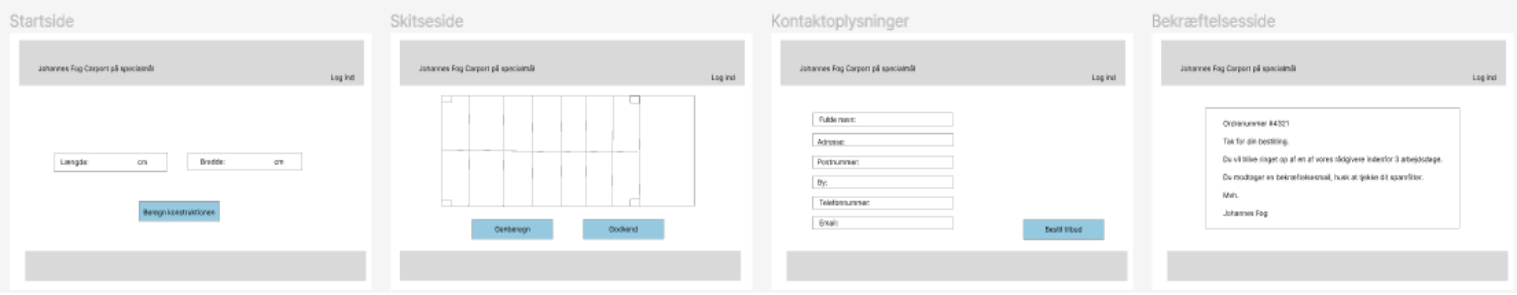
Navigationsdiagram & Mockups



JSP	Reference	Servlet	Destination JSP
index.jsp	Beregn	CalculateCarport.java	skiteseside.jsp
skitseside.jsp	Tilbage	CalculateCarport.java	index.jsp
skitseside.jsp	godkend	ApproveCarport.java	kontaktinfo.jsp
kontaktinfo.jsp	bestil tilbud	RequestCarportOffer.java	bekræftelsesside.jsp
login.jsp	hvis betalt	ToBillOfMaterials.java	stykliste.jsp
login.jsp	hvis ikke betalt	ToOrderView.java	ordrevisning.jsp
ordrevisning.jsp	betal	TogglePayment.java	stykliste.jsp
login.jsp	log på som admin	ToCustomerOrders.java	kundeordre.jsp
kundeordre.jsp	send tilbud	ToggleOffer.java	kundeordre.jsp
kundeordre.jsp	materialevarer	ToMaterialList.java	materialevarer.jsp
materialevarer.jsp	slet	DeleteMaterial.java	materialevarer.jsp
materialevarer.jsp	tilføj	ToAddMaterial.java	tilføjmateriale.jsp
tilføjmateriale.jsp	tilbage	ToMaterialList.java	materialevarer.jsp
kundeordre.jsp	se design	ToDesign.java	ordredesign.jsp
ordredesign.jsp	tilbage	ToCustomerOrders.java	kundeordre.jsp
ordredesign.jsp	se carport	ToEditCarport.java	redigercarport.jsp
redigercarport.jsp	opdaterhonorarpris	EditFeePrice.java	redigercarport.jsp
redigercarport.jsp	opdatercarportdesign	EditCarportMeasurements.java	redigercarport.jsp
redigercarport.jsp	tilbage	ToCustomerOrders.java	kundeordre.jsp

Navigationen rundt på vores hjemmeside ses herover samt en samlet liste af benyttede servlets til at binde siden sammen. Vi bruger i vores system, en fælles navigations bar som vi benytter til login/logout, såvel som når man er logget ind på siden som admin, får man her mulighed for at gå til en liste over materiale fra lageret samt en liste over kundeordre.

Mockup



Her ses bestillings flowet for kunden, der vil bestille en carport. Først indtaster kunden sine mål og får efterfølgende en skitse af carporten på skitsesiden. Her kan kunden så tage stilling til om alt ser ud som ønsket og enten vælge at foretage en genberegning med nye mål, eller godkende for så at komme videre i processen. I næste del skal kunden så indtaste oplysninger såsom navn, adresse, telefonnummer og email, derefter kan der bestilles et tilbud på carporten så sendes til admin. Kunden præsenteres da for en ordrebekræftelse med et ordrenummer samt info om bestillingen.

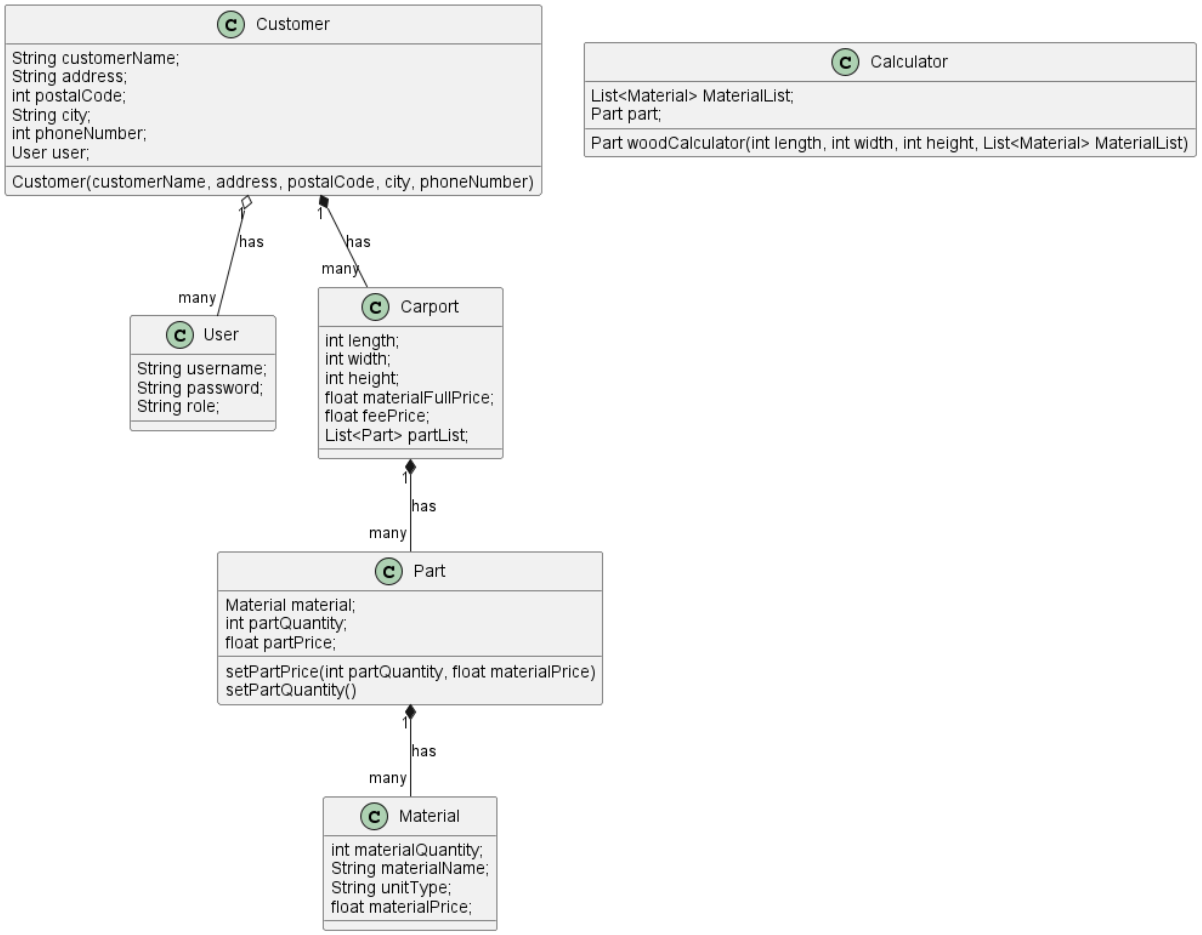
Efter bestillingen kan brugeren logge ind med ordrenummer og telefonnummer for at komme ind på sin bestillingsside. På dette stadie afventer kunden et tilbud fra administrator, og så snart ordren er godkendt får brugeren mulighed for at betale carporten via betal-knap. På efterfølgende side vil kunden får mulighed for at se sin stykliste, hvoraf alle materiale fremgår samt den endelige skitse.

Vores admin flow starter samme sted som bruger login, her kan admin med et specielt admin login, logge sig ind på admin siden og se alle kundeordre, der er her mulighed for at se specifikationerne på de carporte der er bestilt. Det er muligt at sende et tilbud, så brugeren kan betale sin carport via brugersiden. Der vil også være informationer om hvorvidt betalingen er gennemført eller ej, såvel som pris og en knap til at åbne en given ordre. Åbnes en ordre vil styklisten kunne ses samt en tegning, såvel som brugerens data. Hvis der er behov for at finpudse designet kan knappen “rediger carport” klikkes. På efterfølgende side kan designet redigeres ved at indtaste nye mål på carporten. Admin har her også mulighed

for at redigere sit honorar for håndteringen samt evt vejledning, for dermed at kunne give brugeren en mere fordelagtig pris, hvis nødvendigt.

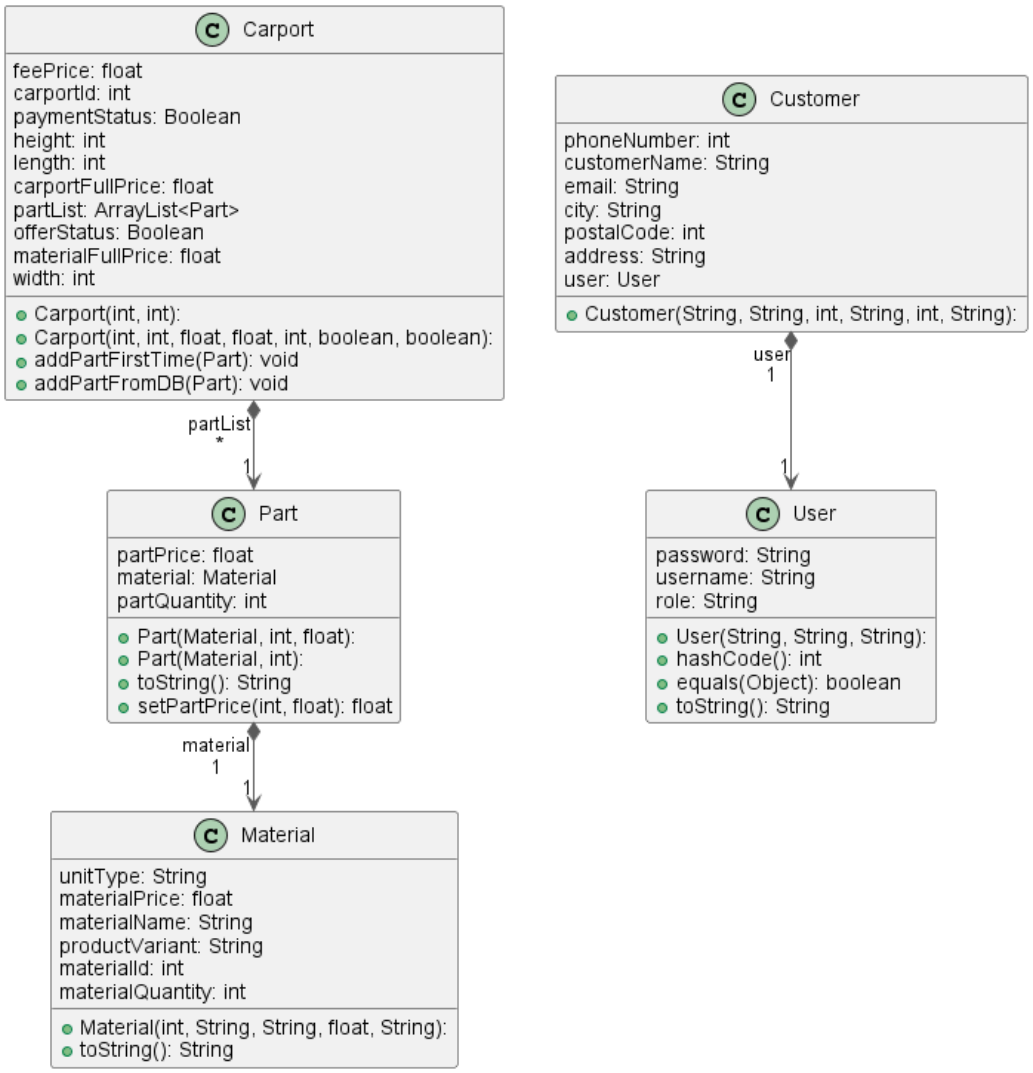
Administrator har også mulighed via admin-login at klikke på “materialevarer” i top menuen, og se den fulde liste over materiale i FOG’s sortiment. Det er muligt at fjerne en vare der er udgået eller lignende, samt klikke på “opret nyt materiale”, for at tilføje et nyt produkt til databasen som gør det muligt at foretage carport beregninger hvoraf disse også vil indgå. Klikkes der på “kundeordrer” i topmenuen startes samme flow som beskrevet i admin forløbet.

Klassediagram

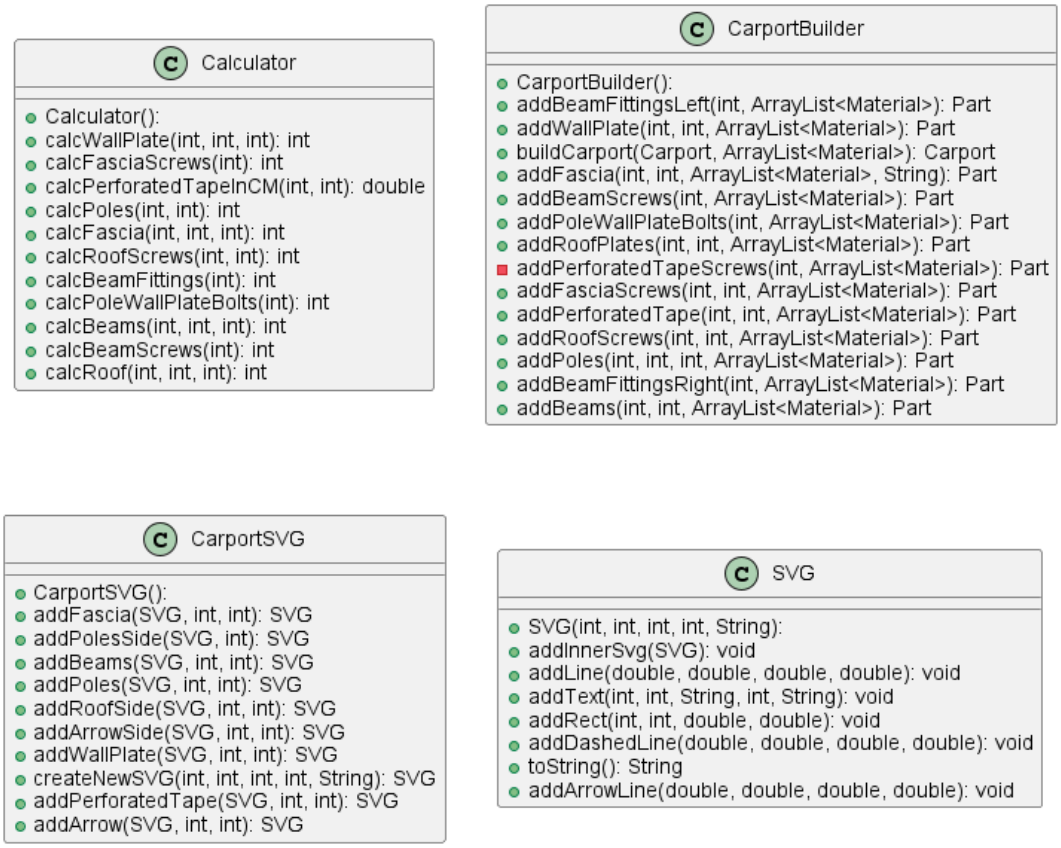


Herover ses et billede af vores indledende klassediagram. Ud fra analysen af kunden og deres behov kom vi frem til, at der ville være behov for en Material-klasse til at bygge og holde styr på materialerne Fog tilbyder. Der er også behov for en Carport-klasse til at holde styr på de data Fogs kunder indtaster som vi kan bruge til senere beregninger af stykliste for kundens carport. En Customer-klasse til at tage holde kundedata under bestilling. Hver carport har sin egen stykliste som vi benytter en List til at samle materialerne i. Årsagen til valget af oprettelsen af en Part klasse også, som mellemmand mellem Materiale og Carport klasserne, skyldes at vi ønsker at kunne differentiere mellem et fastsat materiale fra Fogs lager, og det samme materiale efter beregninger. Så vi ex. ved hvor lang et bræt til spær er, samt hvor mange der skal bruges af dette materiale i styklisten. Så derfor samles materialet på Part-klassen for her at kunne sætte et Materiale, en Mængde af dette materiale samt en samlet pris, hvis der eksempelvis skal bruges 5 af det pågældende materiale, ganges materialPrice fra Material-Klassen.

Vi ser ligeledes et behov for at have en Calculator-klasse til at bygge metoder til beregning af materialer til stykliste af dele(Part) som gemmes på Carport-klassen, ud fra de fastsatte materialer(Material) som vi henter fra databasen med Materiale.



Vores endelige klassediagram er som vist herover. Overordnet set er opbygningen gået som forventet, den største forskel fra indledende oprids af klassediagrammet til det endelige her, er afkoblingen af customer fra Carport-Part-Material flowet. Dette skete da vi gennem udviklingsarbejdet ikke så nogen grund til at Carport havde en customer i sig, da denne samles i carport tabellen i databasen og vi ikke havde behov for at lave denne sammenkobling i programmet for at nå det ønskede mål. Customer er til gengæld koblet til user i form af at Customer har en user, dette fandt vi nødvendigt i forbindelse med den del af vores program hvor en kunde kan anmode om et tilbud på carport.



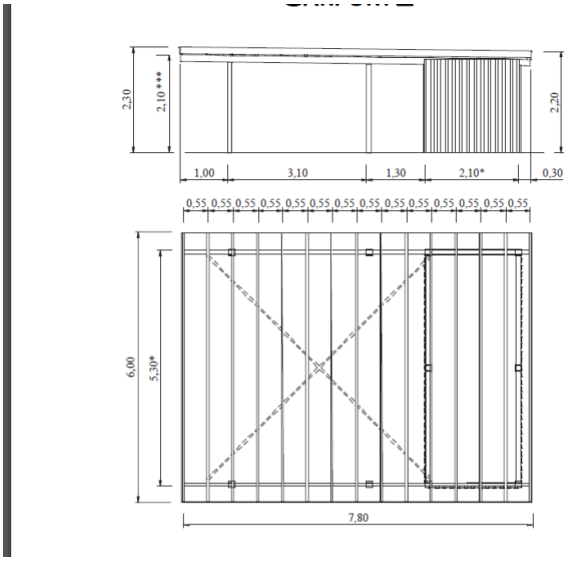
Her ses klassediagrammet fra vores service klasser, da vi indledende nævner behovet for en calculator klasse, så ses her det endelige resultat med metoder vi har brugt i forbindelse med beregningerne af en tilpasset carport. Vi var som udgangspunkt klar over behovet for SVG klasser, da ønsket fra FOG var at der kunne tegnes en skitse af carporten i bestillingsprocessen. Dog var SVG en teknologi vi måtte lære i forbindelsen med udviklingsprocessen, så vi havde ikke den fornødne viden herom til at give et overslag på hvordan en SVG klasse skulle opsættes, så vi undlod at skitsere dette i vores første klasse diagram. Vi valgte at afgrænse vores beregninger i projektet, så calculator klassen kun beregner antal af et givent materiale baseret bla. på carportens længde, bredde og materialet længde. I vores CarportBuilder-Klasse bruger vi metoderne fra calculator klassen, til at finde det mest egnede materiale til den valgte carport og tilføjer denne til styklisten.

Valg af arkitektur

Vi har arbejdet ud fra den startkode vi er blevet udleveret på studiet, da dette blev sat op for os, som hvad der skulle bruges.
Derfor er vores valg af arkitektur sat til at være Model-view-controller med Page Controller.
Vi arbejder med en facade/mapper-lag struktur, igen en del af startkodens opsætning, men det bliver gjort for nemt, at kunne adskille de forskellige lag mellem backend og frontend.

Egne Regler

For at kunne bygge vores carport op har vi været nødt til, at sætte nogle regler vi kunne arbejde ud fra.
Her har vi ud fra den carport vi blev udleveret i opgavebeskrivelsen.



Vi har derfor sat følgende regler op:
En Carport er altid 210cm høj.
Der kan maximum værer 310cm mellem hver stolpe.
Et skur er altid 210 langt.
Et skur følger altid bredden af carporten.
De 2 ydre remme sættes altid 35 cm ud fra den ende de ligger nærmest.
Der må max være 530cm mellem 2 remme.

Særlige forhold

Vi har grundigt overvejet hvornår vi gemmer data i session og requestscope. Den data vi gemmer i session scopet, er ofte oplysninger vi gerne vil bruge mere end en gang. Det kunne f.eks. være når admin er logget på, og kan få vist en side med materialevarer og kundeordrer. De lister skal kunne blive vist hele tiden, og derfor er de hængt op på session scopet og ikke request, da vi så ville være nødsaget til at hænge den op i alle servlets vi kommer igennem. Et andet eksempel kunne være, at vi gemmer kundens oplysninger om den udregnede carport i sessionen, indtil der bliver bestilt et tilbud, først der bliver det gemt i databasen.

Ved de metoder, hvor noget kan gå galt, f.eks. når vi henter data fra databasen, så har vi håndteret exceptions. Det kunne f.eks. være der hvor vi laver brugerinput validering, når en kunde skal logge ind. Ved at omfavne koden med en try-catch og kaste en exception, kan vi undgå at systemet crasher. Dette kunne være hvis kunde logger på med et brugernavn der ikke eksisterer, eller hvis koden ikke er rigtig.

Hvert brugernavn er unikt, da kunden logger på med et ordrenummer, der bliver autogenereret, når de bestiller en carport, og koden til deres login er også unikt, da det er deres telefonnummer. På den måde har vi lavet en form for sikkerhed i forbindelse med login, så man kun kan tilgå sin ordre eller stykliste, ved at kende de to.

I databasen er der lige nu to brugertyper. Der er en user og en admin. Dette var en del af startkoden, men vi har ændret lidt på det, så det ikke længere er en boolean, men nu en String, så vi på længere sigt også kan udvide.

Til at tegne carporten har vi brugt SVG ud fra nogle beregninger vi har lavet. Vi har forsøgt at lave metoderne helt dynamiske, så de kan bruges til forskellige mål. Styklisten til carporten bliver også genereret ud fra vores beregninger der bygger en carport

Udvalgte kodeeksempler

Eksempel 1:

```
try {
    if(request.getParameter( s: "length") != null || request.getParameter( s: "width") != null) {


        Boolean hasShed = false;
        int length = Integer.parseInt(request.getParameter( s: "length"));
        int width = Integer.parseInt(request.getParameter( s: "width"));

        ArrayList<Material> materialArrayList = MaterialFacade.getMaterials(connectionPool);

        request.setCharacterEncoding("UTF-8");
        response.setCharacterEncoding("UTF-8");
        Locale.setDefault(new Locale( language: "US"));

        String checkbox = request.getParameter( s: "checkbox");
        if(checkbox != null) {
            SVG svgCarportTopViewWithShed = CarportSVG.drawCarportTopViewWithShed(length, width);
            SVG svgCarportSideViewWithShed = CarportSVG.drawCarportSideViewWithShed(length, width);
            request.setAttribute( s: "svgCarportTopViewWithShed", svgCarportTopViewWithShed);
            request.setAttribute( s: "svgCarportSideViewWithShed", svgCarportSideViewWithShed);
            hasShed = true;

        } else {
            SVG svgCarportTopView = CarportSVG.drawCarportTopView(length, width);
            SVG svgCarportSideView = CarportSVG.drawCarportSideView(length, width);
            request.setAttribute( s: "svgCarportTopView", svgCarportTopView);
            request.setAttribute( s: "svgCarportSideView", svgCarportSideView);
        }

        Carport carport = CarportBuilder.buildCarport(new Carport(length, width, hasShed), materialArrayList);
        session.setAttribute( s: "carport", carport);

        request.getRequestDispatcher( s: "skitseside.jsp").forward(request, response);
    }
} catch (Exception e) {
    e.printStackTrace();
    String errormessage = "Husk at vælge både længde og bredde";
    request.setAttribute( s: "errormessage", errormessage);
    request.getRequestDispatcher( s: "index.jsp").forward(request, response);
}
```

Dette eksempel er taget fra en servlet, der hedder “calculate carport”. Det er den servlet man går igennem, når man har valgt sine længde- og breddemål i dropdown menuen på forsiden (index.jsp) og trykker på beregn-knappen. Her “bygges” en carport via en “CarportBuilder”, der ligger i “Sevices”-pakken. Man sendes derefter videre til en “skitseside”, hvor man får vist to tegninger af en carport på de valgte mål, både én fra siden og én oppefra. I bestilling processen er det for brugeren muligt at vælge via. en checkbox, om der er ønske om skur til carport eller ej. Der tages i servletten forbehold for om carporten skal tegnes med, eller uden skur. Derudover beregnes carportens stykliste også ud fra valget om skuret og ekstra materialer tilføjes, såfremt skur er valgt på forsiden.
Carporten tages med videre på sessionen indtil man placerer en ordre - med mindre man laver en ny beregning vha genberegn-knappen, så overskrives den eksisterende carport af en ny carport på andre mål. Når man er tilfreds og placerer en ordre gemmes carport- samt kundeoplysninger i databasen og man tildeles en bruger, hvorefter man kan logge ind med

sit ordrenummer som brugernavn og telefonnummer som adgangskode. Alle informationer om carporten kan herefter tilgås af både kunde og administratoren via 'login' knappen.

Eksempel 2:

```
public static int calcPoles(int carportLength, int carportWidth) {     //'poles' er 'stolper' på dansk

    // alle de følgende antagelser er baseret på en skitse af en standard carport på 780 cm x 600 cm udleveret af Fog

    // vi har besluttet at længde er det man også ville kalde dybden på carporten, når man står ved indgangen hvor bilen kan køre ind. I eksemplet er det 7,8 m
    // vi har besluttet at bredden er målet fra højre til venstre, når man står ved indgangen hvor bilen kan køre ind. I eksemplet er det 6,0 m

    // vi har besluttet at længde og bredde måles i cm, og vi antager at hvis længden overstiger 310 cm skal der indsættes en ekstra stolpe
    float lengthInterval = 310f;
    // vi har besluttet at længde og bredde måles i cm, og vi antager at hvis bredden overstiger 530 cm skal der indsættes en ekstra stolpe
    float widthInterval = 530f;
    // vi har besluttet at der lægges 100 cm til foran på carporten samt 60 cm ved bagenden af carporten
    int lengthStandOff = 160;
    // vi har besluttet at der lægges 35 cm til i hver side af carporten
    int widthStandOff = 70;
    int resLength;
    int resWidth;

    float lengthDiv = (carportLength-lengthStandOff)/lengthInterval;
    float widthDiv = (carportWidth-widthStandOff)/widthInterval;

    // der tilføjes en ekstra stolpe efter længden er divideret med intervallet, da den forrige linje udregner antallet af mellemrum, ikke antallet af stolper
    // og der skal dermed sættes en stolpe efter sidste mellemrum
    resLength = (int) lengthDiv + 1;
    // hvis der var en rest da carportens længde blev divideret med det givne interval, skal der tilføjes en ekstra stolpe, så intervallet ikke overstiges
    if(lengthDiv%1 > 0){
        resLength = resLength + 1;
    }

    resWidth =  (int) widthDiv + 1;
    if(widthDiv%1 > 0){
        resWidth = resWidth + 1;
    }

    int resTotal = resLength * resWidth;

    return resTotal;
}
```

Dette eksempel er taget fra en beregningsmetode, der ligger i "Calculator"-klassen i "Services"-pakken. Det er en metode, der udregner hvor mange stolper en given carport-konstruktion vil skulle bruge. Der er fastsat et antal "regler" ud fra et eksempel på en af Fog's carporte. Disse regler er skrevet ind som kommentarer her, da det er den første udregning der sker, når man kalder metoden "buildCarport", som vist i Eksempel 1 ovenfor. "buildCarport"-metoden ligger i "CarportBuilder"-klassen og indeholder udover denne metode et antal metoder, der kalder 'calc' metoder fra 'Calculator' på et specifikt materiale og tilføjer det antal materialer, der skal bruges til en carports stykliste. "CarportBuilder" er ansvarlig for at tjekke i databasens materialetabel og vælge det mest hensigtsmæssige materiale blandt de tilgængelige, hvorimod 'Calculator' er ansvarlig for at forholde sig til hvor mange gange et specifikt materiale skal bruges på en carport med de givne mål.

Status på implementering

US-1: Som kunde ønsker jeg at kunne indsætte mine egne mål, så systemet beregner hele konstruktionen af min carport.	Kunden kan indsætte sine egne mål og trykke beregn vha dropdown menuer og en knap. Herefter beregnes hele carporten på de mål inklusiv pris og stykliste. Hvis kunden ikke er tilfreds med målene kan man vha en genberegn-knap blive sendt tilbage til den første side og indtaste nye mål, der beregner og overskriver den forrige carport med en ny.	100%
US-2: Som kunde ønsker jeg at se en tegning af min carport, som hjælper mig med at visualisere mit design, så jeg kan redigere målene indtil jeg er tilfreds.	Kunden får vist to tegninger af en carport på de valgte mål. Der vises én tegning fra siden og én oppefra. Hvis kunden vælger at trykke genberegn og indtaste nye mål på carporten opdateres tegningerne også.	100%
US-3: Som kunde ønsker jeg at kunne godkende mit design, så jeg kan bestille et tilbud.	Kunden kan trykke godkend på sit design og bliver derefter sendt videre til en side, der kan tage i mod kundens kontaktoplysninger.	100%
US-4: Som kunde ønsker jeg at kunne indtaste mine kontaktoplysninger, så jeg kan modtage et tilbud på den carport, jeg er kommet frem til passer til lige præcis mit behov.	Kunden kan indtaste sine kontaktoplysninger i et antal felter, der passer til de informationer der er brug for. Når kunden trykker på 'Bestil tilbud'-knappen gemmes alle informationer om både kunden og carporten i databasen og kan derefter tilgås af både admin og kunden via 'Login'-funktionen. Kunden tildeles i denne forbindelse en bruger i databasen, som kan bruges til at logge ind med, hvor brugernavnet er ordrenummeret og adgangskoden er kundens telefonnummer.	100%

US-5: Som kunde ønsker jeg at blive præsenteret for en bekræftelses-side, når jeg har trykket 'Bestil tilbud', så jeg ved at bestillingen er blevet sendt afsted, og jeg kan se mit ordrenummer.	Kunden bliver sendt til en bekræftelsesside, hvor ordrenummeret bliver vist, hvilket efterfølgende også er hvad kunden skal bruge som brugernavn til at logge ind og se alt info vedr sin ordre.	100%
US-6: Som admin ønsker jeg at kunne logge ind, så jeg kan få adgang til de relevante sider min rolle indebærer.	Når admin logger ind bliver de ført direkte til siden der viser alle kundeordrer, med muligheden at skifte til materialeliste gennem vores link i headeren samt videre til den enkelte kundes bestilling i tabellen vist på siden.	100%
US-7: Som admin ønsker jeg at kunne se en kundes bestilling, så jeg enten kan godkende bestillingen eller kontakte kunden gennem mail/telefon og yde rådgivning angående designet.	Admin vises en side omkring den gældende bruger, der både har brugerens data samt information omkring deres carport.	100%
US-8: Som kunde ønsker jeg at kunne tilgå en personlig side, så jeg kan se et tilbud, når designet er godkendt af admin.	Kunden kan logge ind på sin personlige ordreside vha sit brugernavn og telefonnummer. Her kan kunden se sit design og en pris samt en betalingsknap, når admin åbner for den mulighed efter at have tjekket og eventuelt ændret i designet eller honorarprisen, hvis nødvendigt.	100%
US-9: Som kunde ønsker jeg at kunne betale for min carport, så styklisten til min carport også vises på min personlige side.	Kunden kan trykke på en 'Betal'-knap og derefter vises også styklisten på den givne carport.	100%
US-10: Som admin ønsker jeg at kunne se alle oplysninger omkring de tilgængelige materialer, der	Admin får vist alle materialer fra databasen.	100%

ligger i databasen, så jeg kan se om det stemmer overens med lageret.		
US-11: Som admin ønsker jeg at kunne slette en vare i de tilgængelige materialer, der ligger i databasen, så jeg kan sørge for at det stemmer overens med lageret.	Admin kan slette en varer i databasen. Man kan argumentere for, at der mangler noget sikkerhed i form af en "er du sikker på du vil slette" popup. User storien er dog gennemført.	100%
US-12: Som admin ønsker jeg at kunne oprette en ny vare i de tilgængelige materialer, der ligger i databasen, så jeg kan sørge for at det stemmer overens med lageret.	Admin bliver ført til en ny side der tager imod alle de informationer der kræves for at gemme nye materiale i databasen.	100%
US-13: Som admin ønsker jeg at kunne ændre oplysningerne på en vare i de tilgængelige materialer, der ligger i databasen, så jeg kan sørge for at det stemmer overens med lageret.	Denne user story mangler helt at blive implementeret, det en mindre del, vi ikke har sat fokus på at få op og kører, da admin allerede kan slette og tilføje nye materialer.	0%
US-14: Som admin ønsker jeg at kunne se prisen på den givne carport, så jeg kan ændre i den pris der bliver sendt i tilbuddet til kunden.	Admin kan på redigering af kundens ordre side både se den samlede materiale pris, deres rådgivningshonorar og ændre i honoraret for at updatede den fulde pris.	100%
US-15: Som admin ønsker jeg at kunne ændre målene på en given kundes carport, så designet bliver mere hensigtsmæssigt til kundens behov	Admin kan ændre i længden og bredden på en given kundes carport, hvis det er nødvendigt, hvorefter de oprindelige mål samt pris og stykliste overskrives i databasen med den opdaterede versions informationer. Kunden beholder sit ordrenummer, så loginoplysningerne forbliver de samme.	100%

Test

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

- Hvilke klasser er testet
- Hvilke metoder der er testet
- Dækningsgrad af dine tests for de valgte 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 main branch.

Pakke	**Klasse**	**Metode**	**Dækningsgrad**
Services			20% methods, 15% lines
	Calculator		100% methods, 84% lines
		calcPoles	9% methods, 17% lines
		calcWallPlates	9% methods, 14% lines
		calcPoleWallPlateBolts	9% methods, 1% lines
		calcBeams	9% methods, 13% lines
		calcBeamScrews	9% methods, 1% lines
		calcBeamFittings	9% methods, 1% lines
		calcPerforatedTapeInCM	9% methods, 5% lines
		calcFascia	9% methods, 11% lines
		calcFasciaScrews	9% methods, 1% lines
		calcRoof	9% methods, 14% lines
		calcRoofScrews	9% methods, 1% lines
Entities			60% methods, 27% lines
	Carport		25% methods, 32% lines
		addPartFirstTime	20% methods, 30% lines
		setPartList	20% methods, 37% lines
		getMaterialFullPrice	
	Part		42% methods, 42% lines

		getPartList	
		getPartPrice	
	Material		14% methods, 40% lines
		getMaterial	
		getMaterialName	

Som en start på projektet arbejdede vi os imod et funktionelt system, med mulighed for at indtaste et mål og få beregnet en carport. I denne process tilsidesatte vi midlertidigt unit testing og lænede os i stedet primært op ad Debug som testværktøj. Vores målsætning var her at opbygge standardmetoder til at beregne dele af carporten og få indsat materialer på styklisten samt at få styklisten vist både for admin og kunden. Dette betyder at vores første metoder i 'Calculator'-klassen ikke blev lavet i form af test driven development, som uden tvivl havde været idealet havde vi haft mere tid til det samlede projekt, eller havde vi siddet med et område som resten af systemet og hermed også resten af gruppen var yderst afhængige af for at kunne bygge ovenpå denne funktionalitet, der sådan set er hjertet af af programmets kunnen. Da systemets grundlæggende funktionalitet var på plads, og vi kunne begynde at tilføje flere elementer til styklisten, begyndte vi at tilføje unit testing på metoderne som blev skabt herefter.

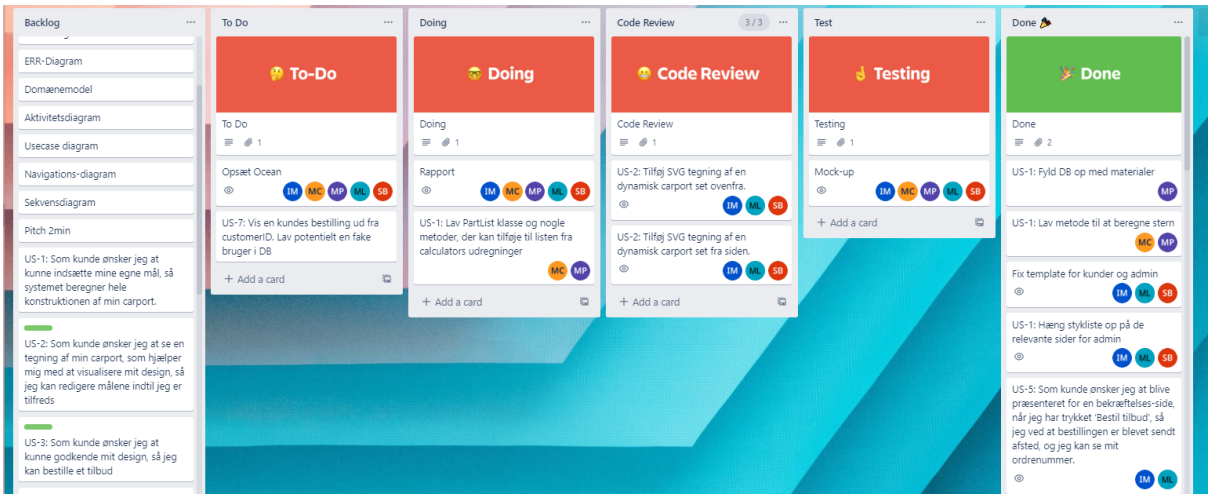
Vi har lavet unit tests på 'Calculator'-klassen, der ligger i 'Service'-pakken. Her testes hver enkelt 'calc' metode ved at give forskellige mål på en carport, for derefter at teste om vi får det forventede antal af et givent materiale ud. Debugger'en her som før nævnt været essentiel i vores process, og den har langt hen ad vejen været en større hjælp til at teste om metoderne vi har bygget opfører sig præcis som vi forventer linje for linje. Dette skyldes at vi i flere tilfælde i løbet af byggeprocessen har været ude for at en test godt kunne bestås på målene på den carport vi fik udleveret samtlige oplysninger på af Fog, men at vi pludselig ikke fik det forventede resultat hvis vi bevæger os udenfor standardmålene. Testene har til gengæld været brugbare ift at isolere et problem, hvis der opstod en fejl i styklisten, og vi havde brug for at vide om problemet lå i 'Calculator'-klassen eller i 'BuildCarport'-klassen. 'BuildCarport'-klassen er den klasse der er ansvarlig for at tale med databasen og hente materialelisten op, for derefter at køre en 'add' metode på et givent materiale og finde den mest hensigtsmæssige variant - med dette menes den variant der bruger færrest stykker af et givent materiale for at opfylde behovet til konstruktionen - dermed også mest det billigste valg for kunden. Vi blev af vores vejleder rådgivet til ikke at køre tests på 'BuildCarport'-klassen, da der ikke er bygget en connection pool baseret på singleton princippet, og vi tester derfor ikke den del af systemet, men har i den forbindelse brugt debugger'en intensivt for at sikre sig at resultatet blev korrekt i samtlige tilfælde vores system kan tage i mod.

Det er begrænset hvor meget vi har testet på andet end beregninger. Det har primært drejet sig om unit tests på styklisten, når denne komponeres for første gang samt samt hvis admin ønsker at ændre i en specifik kundes carport mål og dermed også ønsker at ændre i styklisten. Resten af programmet taler med databasen og har derfor ikke umiddelbart kunne testes uden det førnævnte singleton design pattern. Her har vi i stedet brugt debugger'en flittigt som beskrevet ovenfor.

Proces

Arbejdsprocessen faktuelt

For at kunne arbejde i en gruppe på vores størrelse uden at komme i for store problemer har vi holdt styr på hvad der skulle lave gennem et par processer.
Vi har brugt Trello kanban board til at holde styr på opgaver i et størrer omfang.



Alle vores Userstories, diagrammer og andet der overordnet skulle dækkes er lagt i backlog. Ud fra backloggen har vi så opdelt især usestorien ind i mindre dele så de kunne tackles i forhold til hvor langt vi var med opgaven.
Vi har så i takt med at opgaverne blev valgt, startet og løst rykket dem videre til et punkt i Code Review, hvor nogle af gruppemedlemmerne der ikke har arbejdet på den del af opgaven skulle vises koden.
Til slut skulle det testes i vores devbranch inden den kunne rykkes til vores main branch og sættes som "done" på trello.

Vi har haft et tidsplan docs dokument, hvor vi har kunnet holde styr på hvornår alle var tilgængelige samt aftalte mødetider med begge vores vejledere. Samt vores daglige gruppemøder, hvor vi aftaler hvad der specifikt skal laves af hver mindre gruppe vi opdeler vores team i.

Her har vi også brugt vores discord tråd til at skrive ned hvad der skal laves for dagen i stikordsform, så man kan gribe fat i det næste når en opgave er løst.

Uge 48	Gruppe	Malene	Martin	Berry	Israa	Mark
Mandag	Scrum meeting kl 9:30					
Tirsdag	Vejledning: Kim 11:30 Scrum meeting kl 9:30	Skal gå klokken 12, er fri om aftenen			Skal gå 11:45-13:30	Kl 12 - Arbejde
Onsdag	Vejledning: Nikolaj 13:00					
Torsdag	Scrum meeting kl 9:30					
Fredag	Scrum meeting kl 9:30					

User stories blev skrevet i fællesskab, ved at gennemgå alt hvad vi mente kunden krævede, efter også at have udspurgt kunden(underviser) om hvad vi var usikre på indgik i deres krav. Her efter skulle vi hver især skrive ned hvor stor en opgave vi så hver enkelt user story. Efterfulgt af at der så skulle tales en opgave op eller ned i størrelse til vi var enige, hvis alle ikke var enige fra start af.
Til sidst skulle vi så sammen finde ud af hvad vores acceptance criteria for vores user stories var.

Vi valgte at dele hver af de 4 uger projektet har kørt op i en fase hver.

1. Første uge som en ren SYS uge, hvor vi fik organiseret alt opgaven krævede, lavede mockup, diagrammer, research af fog og pitch.
2. Anden uge med kode opsætning, navigation og påbegynde de første beregninger. Altså skabe et overblik over hele vores hjemmeside i forhold til vores mock-up.
3. Tredje uge var det en fælles lektie at lærer SVG så vi kunde begynde at få koblet vores voksende antal udregninger op sammen med billederne af de tegnede carporte.
4. Fjerde uge skulle så bruges til at færdiggøre rapport, diagrammer samt at lukke alle de ender der stadig fandtes, og fixe alle de små fejl der stadig var.

En store del af vores arbejde med userstories handler om opdeling af arbejdet på en måde der produktivt giver mening i forhold til ikke at skulle arbejde ind over hinanden. Dette har krævet at opgaverne så vidt muligt har været opdelt i form af.

1. Navigation
2. Beregninger
3. Customer sider (hele processen i at bestille en carport efterfulgt af opret kunde)
4. Kunde sider (processen efter man er logget ind som kunde
5. Admin sider

Arbejdes der i de forskellige områder burde vi kunne omgå store merge konflikter.

Fremfor at have en kanban mester har vi i stedet arbejdet ud fra at dele os op i mindre grupper når der arbejdes, og at de små grupper har til opgave at opdatere kanban undervejs.
Dette for at give alle mulighederne for at skulle tage stilling til at få opdateret kanban undervejs.

Arbejdsprocessen reflekteret

Vores ide med ikke at have en klar kanban mester og i stedet at hver gruppe havde til ansvar at opdatere kanban har både virket godt og skidt.
Nogle medlemmer har været meget på med at få opdateret kanban efter hver afsluttet opgave, hvilket har resulteret i at man tydeligt har kunnet se deres arbejde efter en afsluttet dag.
Der er dog andre medlemmer der har været mindre stærke med dette, og er endt med ikke at have opdateret kanban særlig meget.
Skulle man gå med denne metoder i fremtiden, ville det nok kræve at det skulle blive prioriteret højere af alle gruppemedlemmerne.
Samtidig er det dog også svært at have en dedikeret mester, da vi alle lever forskellige liv, der pludselig kan hive en væk fra arbejdet og pludselig kan det være svært at nå at informere en tredjepart(kb mester) om, hvad man har lavet gennem dagen.
I sidste ende har vi skullet vælge vores prioriteter på vores tid, og her er trello klart blevet set som en mindre vital del for at komme i mål med projektet.

Vores evalueringsmøder opdeles i de 4 faser der blev nævnt tidligere.

1. Første uge fik vi projekt start kl 15, efterfulgt af at have en møde med vores sys lærer, næste dag omkring middag. Her var der et håb eller forventning om at vi allerede havde noget klar til fremvisning i form af user stories, der dog ikke eksisterede endnu. I stedet fik vi en samtale, der lidt skød os i øst og vest i forhold til hvad der skulle prioriteres.
   Med vores Kode lærer dagen efter havde vi vores user stories klar og fik spurgt om hjælp til mindre problemer. Især med sys vejledningen havde det været bedre for os, hvis ikke den lå så kort efter projektstart.
2. Anden uge startede vi med at kode, her fik vi dog muligheden for at stille sys vejlederen spørgsmål til de forskellige diagrammer, kanban osv vi havde startet på. Igen følte det som om vi gik med flere bekymringer end vi kom med.
   Vores kode vejledning var tæt på ikke eksisterende da vi kun lige var startet, og allerede var godt igang.
   De største bekymringer vi gik med var hvor stor en forskel der var i den info vi fik fra begge vejledere, samt hvad de mente skulle være vores fokus på nuværende tidspunkt.
3. Tredje uge havde vi ikke meget at bringe på bordet til vores sys lærer, vi havde i den foregående uge valgt at prioritere 100% på at vores kode, på samme måde som vores første uge havde været fuld fokus på sys. Her fik vi dog ros for vores brug af trello, og en generel ide om hvordan vi skulle lægge vores fremtidige sys fokus.
   Vores kode lærer hjalp os med at planlægge, hvad vi skulle sætte fokus på at nå at

færdiggøre denne uge, og vi kunne se vi var godt med på den front.
En uge med mere fokuseret vejledning, og en arbejdsmoral i topform.

4. Fjerde uge startede startede med kode vejledning, vi var stort set færdige her, og kunne vi kom frem til de sidste ting der skulle prioriteres for projektet.

Processen med at nedbryde vores user stories gik godt, det var endnu en opgave til de enkelte grupper at stå for at bryde arbejdet ned inden de startede med at arbejde i et område.
Det gik super godt, forskellige størrelser på dem, gjorde selvfølgelig at de blev delt op i forskellige antal, men generelt gav det os en meget strømlinet arbejdsproces, der gjorde at vi hurtigt kunne lyse hver del den enkelte user story indebar.

Vores estimeringer af user stories passede i forhold til den tid vi brugte på hver opgave. Forskellen på en stor og en lille opgave var tydelig, vores originale pitches til user stories inkludere en en epic størrelse, som vi fik splittet ned i 2 large og en small, hvilket var det helt korrekte valg for at den enkelte opgave kunne overskues på et niveau der gav mening.

En af de store fordele vi havde som gruppe har klart været, at vi allerede i et tidligere projekt havde arbejdet på vores produktive rytme. Så vi havde nogle klarer aftaler da vi gik i gang med projektet. Vi vidste fra start af, for at være produktive, krævede det at, vi planlagde alting. Dette har også resulteret i rigtig meget tid er gået på vores morgenmøder, så alle var klar over hvad der præcis skulle ske.
Det er rigtig nemt at spænde ben for hinanden når man arbejde 5 i et så relativt lille projekt som vi har. Hvilket også gør, at en stor del af det vi har skullet lærer er, netop den planlægning det kræver for ikke, at ramme en masse problemer undervejs.
Git kan nemt værer et benspænd, hvis man ikke bruger det rigtigt når man er flere der deler et lille arbejdsområde.
Netop fordi vi havde de forskellige arbejdsområder opdelt i 5 dele, gjorde det at vi meget smertefrit har kunnet arbejde med git.
Ud fra vores main branch lavede vi en devbranch, for at sikre sig at vores program altid havde en fungerende main, ud fra devbranchen oprettede vi så alle vores små gruppe brancher på, hvad end vi nu startede med at arbejde på.
Når vi mergede det gjorde vi det sammen for at sikre os at alt stadig virkede inden det blev smidt op på main branchen. Dette workflow gjorde virkelig, at vi slap for problemer.
Ikke at vi ikke ramte nogen problemer, der var meget få der opstod, og det var generelt fordi vi havde misforstået hinanden, det gode her er, at når først sådan en misforståelse var sket og fanget, kom der ikke lignende fejl igen. Helt klart en process hvor vi har udviklet os hen ad vejen.

```
silve@LAPTOP-5A5LDHV4 MINGW64 ~/OneDrive/Skrivebord/Carport (devbranch)
$ git logg
* f446037 (HEAD -> devbranch, origin/devbranch, main) minor changes to comments in carport builder
* def4d84 adds comments to calcPoles method and erases index2.jsp
*   f40c0b5 Merge branch 'calcRoof' into devbranch
|\
| * 457e114 (origin/calcRoof) addRoofPlates, addRoofScrews and addPerfTapeScrews metodes
| * a332dcf deleting test from carportbuilder
| * b92f9a4 calculate roof
| * b02a512 roof Material update
* | a9b8cc7 (origin/tryCatches) updates small details in formgroups
* | e8e1d7f updates form for kontaktinfo jsp
* | 8c2d41c updates footer
* | d7986e4 imeage href when logged in as user
* | e827ec8 adding try catches to login, index, redigerCarport jsp
* | a83cc78 fixing dublicate
* |   68c7ac7 fixing mergeconflict
|\ \
| * | a4e1eea (origin/addingSideView, addingSideView) adds svg to jsp
| |/
* | 3d77475 (origin/cssOnAllJsp) css and bootstrap on several pages2
* | 1d3986d css and bootstrap on several pages
|/
*   0c02d4e Merge branch 'calcV2' into devbranch
|\
| * bdac187 (origin/calcV2) creates add method for perforated tape in carport builder
| * dcd4559 calcPerforatedTapeInCM function created in calcualtor
| * 118293c directs tilbage button to kundeordre.jsp
| * f752ba5 refactors BOM to services
```

Her ses en lille del af vores git. Det sker også at vi har arbejdet direkte i vores devbranch, men det er kun gjort når vi har været igang sammen.