Fog Carporte

Maj 2020

Datamatiker 2. Semester, Hold E, Gruppe 666

Alexander Pihl
Email: cph-as509@cphbusiness.dk
Github: https://github.com/AlexanderPihl

Morten Rasmussen
Email: *cph-mr462@cphbusiness.dk*
Github:
https://github.com/amazingh0rse

Mick Larsen
Email: *cph-ml616@cphbusiness.dk*
Github: *https://github.com/MivleDK*

Per Kringelbach
Email: cph-pk171@cphbusiness.dk
Github: https://github.com/cph-pk

Jean-Poul Leth-Møller
Email: *cph-jl360@cphbusiness.dk*
Github: *https://github.com/Jean-Poul*

Indhold

0. Projektformaliteter

Produktets demovideo

https://youtu.be/1ygwSqIfdZ8

Produktets website

http://134.209.243.195:8080/Fog/

Bruger: fogAdmin@fog.dk
Pass: Gruppe666/

Projektets kildekode

https://github.com/Jean-Poul/Eksamensprojekt_Fog

Projektets JavaDoc

https://jean-poul.github.io/Eksamensprojekt_Fog/

1. Indledning

Vi har fået stillet til opgave at hjælpe virksomheden Johannes Fog med at udarbejde et IT-system, hvor en kunde kan bestille en standard og byg-selv carport. Vi har valgt at gøre det med et website, som har forbindelse til en dedikeret server med en database. Fog kan derfor nemmere hente, gemme og ændre i deres data, samt danne sig et overblik over deres data.
Fog skal kunne tage imod forespørgsler fra kunder, som en medarbejder herefter kan se og rette i. Herfra kan der udarbejdes et acceptkriterie, som herefter kan ses af kunden.
Vi har i vores gruppe haft fokus på, at man som kunde skal kunne vælge mål, materialer, tagvinkel samt skur med tilhørende mål. Vi har dog forbeholdt os retten til at fastsætte højden til 2,50 meter (Hvilket er i overensstemmelse med Fogs nuværende løsninger).
Alt bliver udregnet ud fra Fogs priser, samt alt materialet som skal bruges for at kunne konstruere carporten. Mere om dette i afsnit ”2.2 A*rbejdsgange der skal IT-støttes”*.

Fogs website skal være dynamisk og vil blive kodet ved hjælp af bl.a. HTML, CSS, Bootstrap, Java og mySQL. For til sidst at afvikle sitet vha. en Tomcat webcontainer. Mere om dette i afsnittet ”2.3 *teknologi valg”*.

Vi er blevet rådgivet til at skulle udvikle det endelige produkt ud fra en skabelon, som vi har fået tildelt. Denne skabelon benytter sig af et designmønster kaldet MVC.
MVC er et framework, der står for Model, View og Controller. Det benyttes i programmering til at adskille ens kode, for at gøre det mere overskueligt og nemmere at vedligeholde. Det endelige produkt skal uploades og køre på en droplet hos udbyderen Digital Ocean. Vores kildekode er tilgængelig på GitHub, hvor vi med hjælp af javadocs har gjort det nemmere, at danne sig et overblik over projektets indhold.

Opgaven er opdelt i to dele, med en administratordel og en kunde/brugerdel.
Johannes Fog ønsker en administratorside, hvor man skal kunne se en oversigt over alle forespørgsler, og hvor en forespørgsel skal kunne rettes og godkendes. Desuden skal en administrator kunne rette i standardmål, samt priserne til materialerne.

1.1. Baggrund

Virksomheden Johannes Fog har fokus på selv-byg og de har derfor flere byggecentre, hvor kunder kan komme for at få rådgivning til deres egne byggeprojekter. Fog står for kvalitetsvarer og udover deres byggecentre har de et bolig- og designhus, hvor designvarer er til udstilling, for at kunne få inspiration og vejledning. Fog brander sig med, at de har et stort udvalg med fokus på kvalitet og de vælger kun produkter, som dækker kundens brugsbehov.

Kunder som vil gøre brug af websitet, ville kunne stille følgende krav:

- Som kunde kan jeg forespørge en carport med standardmål
- Som kunde kan jeg forespørge en carport med udvalgte mål
- Som kunde kan jeg tilføje et skur til min carport med udvalgte mål
- Som kunde kan jeg tilføje grader på mit tag
- Som kunde kan jeg selv vælge mit materiale
- Som kunde forventes der at få tilsendt en status på en afsendt forespørgsel
- Som kunde forventes der at Fog tilsender en oversigt af materiale
- Som kunde forventes der at Fog tilsender arbejdstegninger

2. Krav

De følgende sektioner har til formål at give et indblik, i hvad virksomheden Johannes Fog forventer og ønsker af deres nye IT-System, samt give et indblik, i hvad deres nuværende system har af mangler.
Der vil også blive henvist til hvordan Fog, som virksomhed, vil få gavn af vores system og få optimeret deres arbejdsgange.
Herudover vil der blive vist hvilke user-stories, vores gruppe har arbejdet med, for at kunne nå i mål med vores produkt. User-stories er oprettet ved hjælp af møder med en product owner og vedligeholdt ved hjælp af den agile udviklingsmetode kaldet scrum.

2.1 Virksomhedens håb for dette system

Vi vil ligge ud med at komme ind på, hvilke forhåbninger virksomheden Johannes Fog, har med vores produkt. Fog har et ønske om at kunne få opdateret deres IT-system, da deres nuværende system er udviklet i 1999, hvilket må siges at være gammelt på nuværende tidspunkt.

I det nuværende system er der ikke mulighed for at opdatere data, da personen som udviklede systemet for Fog, ikke var udvikler og ikke lavede systemet med henblik på videreudvikling.
Den tidligere ansatte, som udviklede systemet, havde en virksomhed ved navnet Bergman IT-service, og det formodes ikke at eksistere længere.
Fog har forsøgt sig med at opdatere deres nuværende system, hvilket kostede dem i omegnen af 1 million kr., men da det ikke kunne ”Tale” med det gamle system, var de nødsaget til at bibeholde deres nuværende system, hvilket må siges at have været en dyr investering.

Fog kan forvente at få alle deres ønsker opfyldt ved brug af vores system, da vi har vedligeholdt de funktioner de er glade for, optimeret dem og endda tilføjet nye funktioner.
Inden der vil blive forklaret hvordan vores system opfylder Fogs forventninger, vil vi først give et overblik over hvordan deres system virker på nuværende tidspunkt.

Fog har ikke mulighed for at kunne rette data i deres nuværende system. Det vil sige at man som Fog-medarbejder, kun har mulighed for at kunne ændre i den samlede pris ved hjælp af en dækningsgrad.
Der er ikke mulighed for at kunne ændre i materialevalg, mål eller antal. Udover dette, er der fejl i den tekst, som fortæller en medarbejder, hvilke mål og materiale, som skal bruges til en kundes forespørgsel på en carport. Det skal dog siges at selvom der er fejl i teksten virker systemet stadigt, som forventet, da de ansatte kender til denne fejl.

Når en forespørgsel på en carport bliver sendt til Fog, som bliver modtaget via e-mail, skal en medarbejder selv indtaste forespørgslen for at kunne generere en stykliste, arbejdstegning og pris til kunden. Herfra kan Fog tage stilling til om det er realistiske mål kunden forespørger og om det er noget kunden selv, har mulighed for at bygge. Da visse kunder ikke selv har mulighed for at bygge deres egen carport, har Fog kontakt til flere tømrere, som har været med til at bygge disse carporte i mange år, hvilket gør det nemt for Fog at udregne en pris. Dette opnås kun pga. deres store erfaring indenfor faget.

Ved at gøre brug af vores IT-system ville Fog få en opdatering af deres system, som medfører en optimering af deres nuværende proces, samt håndteringen af salg af deres carporte.

Vores system er udviklet på sådan en måde, at en kunde ikke har mulighed for at vælge urealistiske mål, da vi har indsat kriterier for diverse valg af mål, en kunde kan foretage sig ved deres forespørgsel.
Derudover skal man ikke genindtaste en kundes forespørgsel for at få udregnet en stykliste med pris, samlet pris og arbejdstegninger, da systemet tager sig af alle disse ting lige så snart en kunde sender sin forespørgsel.

Til sidst skal der kort nævnes, at en medarbejder har mulighed for at rette i alt det data der høre til en forespørgsel. Systemet har fuldt adgang til en database og derved kan der ændres i alt data.
Man skal dog have de rigtige rettigheder og adgang. Det vil sige at det kun er tilegnet en administrator af systemet.

Alt i alt kan Fog forvente at digitalisere de daglige arbejdsgange, samt optimere de ansattes brug af deres tid, hvilket vil være med til at optimere virksomhedens omsætning.

2.2 Arbejdsgange der skal IT-støttes

Inden IT-udviklingen kan finde sted, skal der designes og udarbejdes et website, da vi på denne måde får mulighed for at binde vores system sammen med en dedikeret server. Dette vil optimere hele deres proces og gøre det hele dynamisk. Vi har i dette tilfælde valgt at holde os tæt op af deres nuværende design og farvevalg, da det er Fogs identitet.

Før vores IT-system blev udarbejdet for Johannes Fog, var en bruger af systemet nødt til fysisk at være til stede på lokationen af det installerede software, da det hele blev kørt på en lokal klient.
Derudover havde brugere af systemet ikke mulighed for at kunne rette i information vedrørende deres udvalg som en kunde havde forespurgt. Dermed kunne der ikke rettes i det udvalg Fog udbyder i form af materiale, længde og antal.
Pris havde en medarbejder selv mulighed for at tilrette, men der nævnes ikke om man havde mulighed for at rette i materialepriser. Kun at man kan rette i totalprisen ved at ændre på dækningsgraden.
Dette gjorde det meget statisk og derved var der ikke mange muligheder for at rette en eventuel fejl i ordren, samt i systemets indhold. Ved udregninger af materiale, som f.eks. antal af stolper og spær, tog Fogs system udgangspunkt i forældet tabel. Det var efter Fogs mening stadig sikkert, da deres udregninger af mål og tryk var foretaget med ”livrem og seler”.
Desuden skal det nævnes at de ikke havde mulighed for at kunne sende en status på en forespørgsel.
Det vil sige at systemet ikke automatisk fik tilsendt en mail til en kunde angående deres forespørgsel på en ordre. Rettelser og bekræftelser blev gennemført telefonisk eller personligt med kunden.

Hos Fog forventer man at der kan ses en oversigt over hvilke forespørgsler de har fra deres kunder.
Hertil skal man have mulighed for at kunne se en kundes informationer. Derudover skal man kunne se status på kundens forespørgsler, da dette giver en Fog-medarbejder mulighed for at kunne tage stilling til en forespørgsel. Grunden til, at der skal være et statusfelt, er at en medarbejder skal kunne se hvor langt forespørgslen er i forløbet og vide hvornår en sælger eller byggemester skal kontakte kunden, med henblik på mersalg eller konstruktionsrettelser. Fog nævner selv denne proces kan virke gammeldags og langsommelig, men det er noget de godt kan lide at forholde sig til, hvilket vi ikke ville ændre på.
Det vil sige at selvom alt i vores system er automatiseret, skal en medarbejder stadigvæk ind og læse en forespørgsel for derved at kunne acceptere eller rette denne og dermed kunne opfylde kundens behov.

Herfra kan man sende en kvittering til en kunde, i form af en e-mail. Fog kan hermed sideløbende ringe til kunden, for at fastholde deres ønskede kontakt med henblik på mersalg og ændringer.

Da alt data kommer fra en database, har Fog også mulighed for at rette i deres data. Det vil sige, at man som Fog-medarbejder har mulighed for at ændre i materialevalg, længde, antal samt pris. Dette er muligt, da vi forholder os til de fire grundlæggende funktioner i en databaseapplikation. Kort sagt kaldes det CRUD og står for Create, Read, Update og Delete.

Før en Fog-medarbejder modtager en kundeforespørgsel bliver materiale, længder, antal samt pris udregnet. Således skal en medarbejder ikke selv regne sig frem til, eller genindtaste mål, som pt. er tilfældet. Alt er udregnet efter byggeregler og følger en opdateret tabel med mål og tryk. Derfor kan kunden være sikker på, at carporten holder når den er færdigbygget.

Efter implementeringen af vores system skal man ikke længere være fysisk til stede, hvor systemets software er installeret, for at kunne ændre i systemet. Alt foregår ved hjælp af en dedikeret server og behøver ikke software installeret på en lokal klient. Alt en bruger af systemet har brug for, er en forbindelse til internettet. Herudover er der mulighed for, at kunne ændre i mål, materialevalg, materialepris(Stykliste), totalpris og status på en forespørgsel. Det er også muligt at lave ændringer til det udvalg, som Fog vil kunne udbyde for en kunde.

2.3 Teknologivalg

Vores Projekt er programmeret i Java under et Maven framework. Alle i teamet har gjort brug af Windows OS.

Vores program tager udgangspunkt i model-view-controller, som er et software designmønster, der er til for at opdele ens kode i tre, overskuelige elementer.
Model er til for at gemme data, som skal fremvises ellers behandles.
View er til for at fremvise et output, hvilket gøres på vores jsp sider.
Controller opfanger et HTTP request og sender et HTTP response tilbage.

Vi har fået tildelt en skabelon, som vi skulle forholde os til og anvende som rygraden af vores projekt.
Her er de mest essentielle klasser Command, FrontController og brugen af Servlets. Command opretter en instans af et HashMap. Kendetegnet ved et HashMap er, at det indeholder en nøgle og en værdi.
Det er igennem denne funktion, at vi gemmer en reference til vores java-klasser og derfor kan lave en instans af de ”rigtige” klasser.
FrontController klassen er til for at lave en instans af Command og til at sende programmet videre til en jsp-side, som er givet i form af et request fra en bruger. Afslutningsvis gør vi brug af HTTPServletRequest samt HTTPServletResponse, som nedarver fra Servlet klassen. Dette gøres fordi vi arbejder med et request fra en bruger, hvorefter der tilsvarende sendes et response tilbage. Alt dette kan bruges da vi bruger et webapplikationsframework, som i vores tilfælde hedder Tomcat.

Hele projektet er versionsstyret med Git som bindeled til Github. ”Milepæle” i projektet er ”Merget” til en Master-branch imens løbende udvikling er styret på en Production-branch.

Der er desuden udarbejdet en omfattende dokumentation i javadocs, som kan ses på projektets github side.

Følgende teknologier er blevet anvendt for at kunne udarbejde projektet og holde kommunikationen intakt i vores gruppe.

Programmering og syntaks

- CSS3
- JSP
- HTML5
- Java - 1.8.0_251
- Javascript (ECMAScript 2015)
- jQuery
- MySQL - 8.0.18
- PlantUML

Frameworks og teknologier

- Apache Tomcat - 9.0.31
- Bootstrap - 4.3.1
- JavaE Web API - 7.0
- JSTL - 1.2
- JUnit - 4.12
- Maven compiler plugin - 3.1
- Maven dependency plugin - 2.6
- Maven war plugin - 2.3
- MySQL Connector Java - 8.0.19

Software

- Adobe XD - 29.0.32
- Discord - 0.0.306
- Draw.io - 11.3.0
- FileZilla Client - 3.48.1
- Firefox - 76.0.1
- Git bash - 3.0.2
- Google Chrome - 81.0.4044.138
- Google sketchup
- IntelliJ IDEA (IDE) - 2020.1.1
- MySQL Workbench - 8.0.18
- Taiga.io
- Photoshop - 20.0.7
- PuTTY - 0.73
- Ubuntu OS 19.10
- Windows 10 OS Build 18362.836
- winRAR - 5.80.0
- Word for Microsoft 365
- Zoom - 5.0.2

2.4 SCRUM User-stories

Scrum er en agil udviklingsmetode, der har stor fokus på ledelsen af projektet.
Strukturen i følgende elementer:

- *Product backlog meeting*: Med product owner hvor user-stories og sprints godkendes.
- *Sprint meeting:* Med sit team hvor user-stories og sprints planlægges.
- *Daily meeting*: Med sit team hvor gårsdagens (Samt de forestående) opgaver gennemgås.
- *Sprint review meeting:* Med sit team for at afslutte hele ens forløb med retrospective meeting.

Mere om dette i afsnit *"8 Process".*

Under vores forløb har vi haft fokus på følgende user-stories, fra vores product backlog, som er blevet godkendt af en product owner, der var tilknyttet vores gruppe. *Se* afsnittet *Appendix* for at få et overblik over alle vores user-stories og de tilhørende tasks.

1 Sprint

#21 Kunde: Forespørgsel på spec. carport (basic)
#22 Kunde: Forespørgsel på spec. carport (basic m. tag)
#23 Kunde: Forespørgsel på spec. carport (basic, tag + skur)
#34 Kunde: Forespørgsel på materialetyper

2 Sprint
#5 Fog: Se forespørgsler
#15 Fog: Generering af tegning

3 Sprint
#80 Dynamisk view a SVG
#82 Fog: Se forespørgsel
#8 Fog: Slette forespørgsler
#14 Validering af spec. løsninger
#109 Kunde: Kvittering ved forespørgsel

4 Sprint
#11 Fog: Administrering af varekatalog
#7 Fog: spec. carport (Redigering)

5 Sprint
#171 Slutspurt
#6 Fog: spec. carporte (Modtaget tilbud)
#148 Fog: CRUD
#59 Fog: beregning af pris

Af de user-stories der er aftalt med product-owneren, har vi valgt at fremhæve og beskrive fire af dem, der hver især dækker en vigtig funktion i programmet.

Der er valgt følgende user-stories:

- **#21 Kunde: Forespørgsel på spec. carport (basic)**
- **#15 Fog: Generering af tegning**
- **#11 Fog: Administrering af varekatalog**
- **#7 Fog: spec. carport (Redigering)**

#21 Dækker over, at kunden skal kunne lave en forespørgsel på en carport i specialmål.
#15 Dækker over, at der skal genereres dynamiske tegninger til en forespørgsel.
#11 Dækker over, at Fog skal kunne ændre og opdatere deres varekatalog.
#7 Dækker over, at Fog skal kunne redigere en eksisterende forespørgsel, således at fejl og mangler kan ændres.

Resten af vores user-stories kan ses i backloggen i appendix ”11.2 User stories”.

#21 Kunde: Forespørgsel på spec. carport (basic)
Som kunde
Vil jeg kunne forespørge på en carport i specialmål (L,H,B) med standard tag, u. skur
Således at jeg kan modtage et tilbud på samme.

Accept kriterier:
Kunden skal kunne angive udvalgte mål til længde, højde, bredde samt vælge fladt tag og fravælge skur.

Estimat:
Back end: L

How-to:
Home -> Bestil -> Byg Selv -> Indtast dine mål
Fra forsiden klikkes der på "Bestil" i navigationsbaren i toppen så man lander på "vælg carport" siden.
Derefter trykkes der på "Byg selv" i højre kolonne nederst, hvorefter man lander på byg-selv siden.
Her kan man angive alle målene til sin carport, tag samt skur og sende forespørgslen afsted til Fog.

Tasks
#30 Krav til dimensionering skal implementeres (Antal søjler, dimensioner mm)
#54 Kundeforspørgsels <select> skal populeres med korrekt data
Vælg tag
1 - valg tagtype: fladt / rejst
2 - hvis fladt = kun ét slags tag
3 - hvis rejst = vælg hældning
4 - hvis rejst = vælg beklædning
#39 Design konverteres til kode
Mockup skal omdannes til .jsp
#27 Database: E/R Diagram
Der skal laves et E/R diagram over vores database
#51 Beregning af lægter
CarporpCalculation skal kunne beregne antallet af lægter og afstand
#48 Database: Queries til tabelopslag på raft length/spacing/dimension
Database queries til tabelopslag skal laves i datamapper
#24 Opsætning af command-pattern skabelon og klargøring på git med MASTER og PRODUCTION branch
#26 Design: Hjemmeside design til index og customerpage
Index skal laves og customerpage skal kunne håndtere bestillinger
#37 Database og program skal forbindes (Med standard credentials)
Connetor klassen & datamapper skal ordnes så der kan oprettes forbindelse til databasen
#28 Domæne model (.uml)
Første udkast af domæne modellen skal laves
#29 Opstart på klasse diagram
Første udkast af klasse diagram skal laves

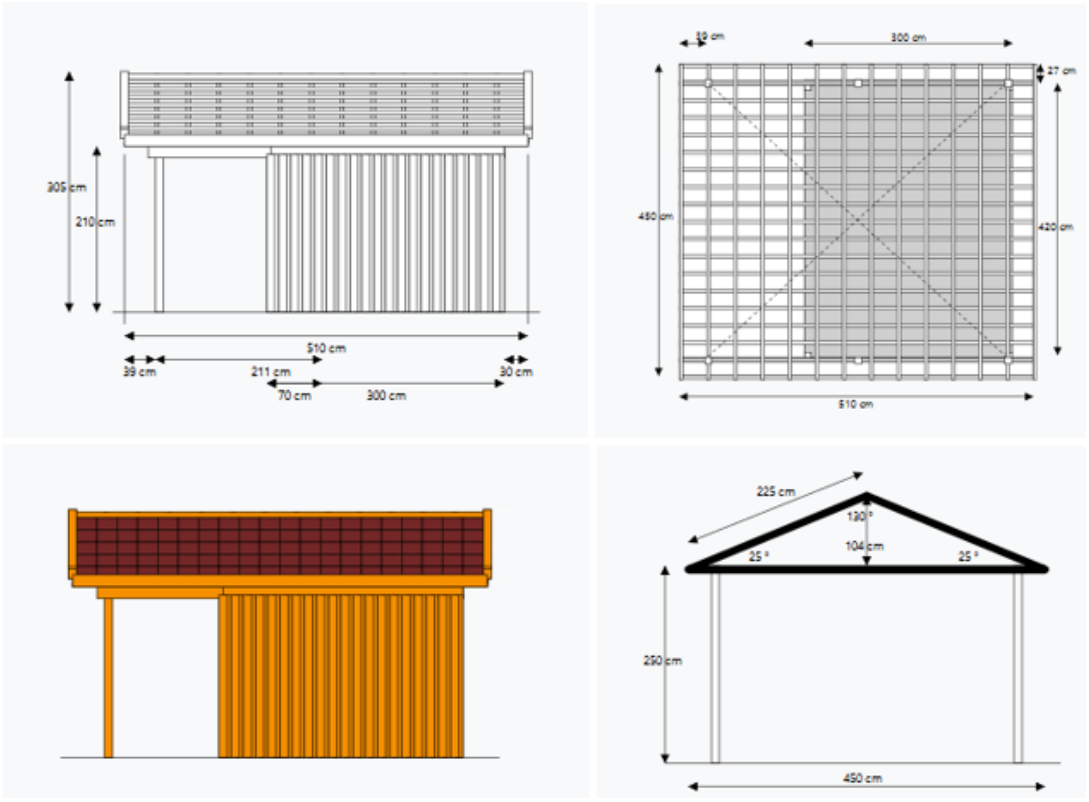
#15 Fog: Generering af tegning

Som Fog-medarbejder
Vil jeg kunne generere en tegning på kundens forespørgsel
således, at jeg kan visualisere løsningen for kunden

Accept kriterier:
Dynamiske tegninger af kundens forespørgsler skal kunne generes og vises

Estimat:
Front end: L
Back end: XL

How-to:
Home -> Log Ind -> Log ind -> Åben -> Se tegning og stykliste
Fra forsiden trykkes der på Log ind i øverste højre hjørne. Derefter indtastes admin bruger info
Brugernavn: <Se afsnit 0>
Kodeord: <Se afsnit 0>
og der trykkes på log ind.
Herefter trykkes der på "åben" ud for en forespørgsel i højre side. Nederst på forespørgselssiden trykkes der på "se tegning og stykliste". Hvorefter de dynamiske tegninger ses nederst på siden.



Figur 1 - Dynamisk genererede tegninger

Tasks
#66 SVG-tegning forfra
Der skal laves en dynamisk SVG-tegning af carporten set forfra med mål og vinkler
#64 SVG oppe fra
Der skal laves en dynamisk SVG-tegning set oppe fra med mål
#63 SVG-mål
SVG-klasserne skal kunne modtage mål fra carportcalculation
#67 SVG-tegning fra siden
Der skal laves en dynamisk SVG-tegning set fra siden med tag med mål
#69 xd design til SVG-landing page
Der skal laves et design til den side der viser alle SVG-tegningerne
#56 Hente data fra database (dimensioner tabel) til program
Der skal laves database queries til at hente forespørgsels data

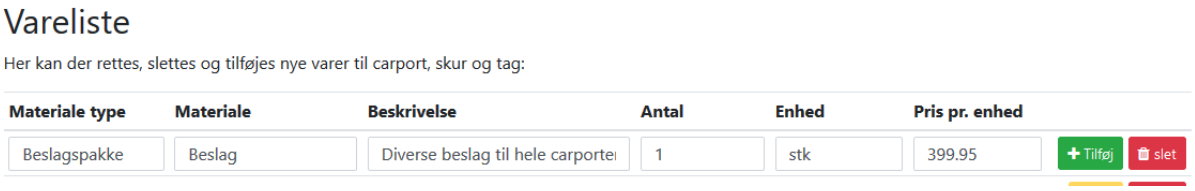
#11 Fog: Administrering af varekatalog

Som Fog-medarbejder
Vil jeg kunne opdatere varer og priser i mit produktudvalg
således, at varekataloget altid er up-to-date

Accept kriterier:
Vareliste og priser skal kunne opdateres

Estimat:
Front end: M
Back end: XL

Home -> Log Ind -> Log ind -> Admin -> Vareliste
Fra forsiden trykkes der på Log ind i øverste højre hjørne. Derefter indtastes admin bruger info
Brugernavn: <Se afsnit 0>
Kodeord: <Se afsnit 0>
og der trykkes på log ind.
Herefter trykkes der på admin i navigationsbaren og derefter på Vareliste.
Der vises derefter en liste over alle varer samt priser i produktudvalget.
Trykker man på ret kan man opdatere alt på den enkelte vare



Figur 2 - Vareadministration

Tasks
#113 Opdater: material, description, quantity, unit, price_per_unit
Som Fog-medarbejder skal jeg kunne opdatere en vare for at kunne rette i eventuelle fejl
#128Fog: Fremvisning af pris
Som fog medarbejde skal jeg kunne se en samlet pris for at kunne redegøre/redigere i den
#129 SVG view dynamisk af forespørgsel
Som for medarbejder skal jeg kunne se et billede af en forespørgsel

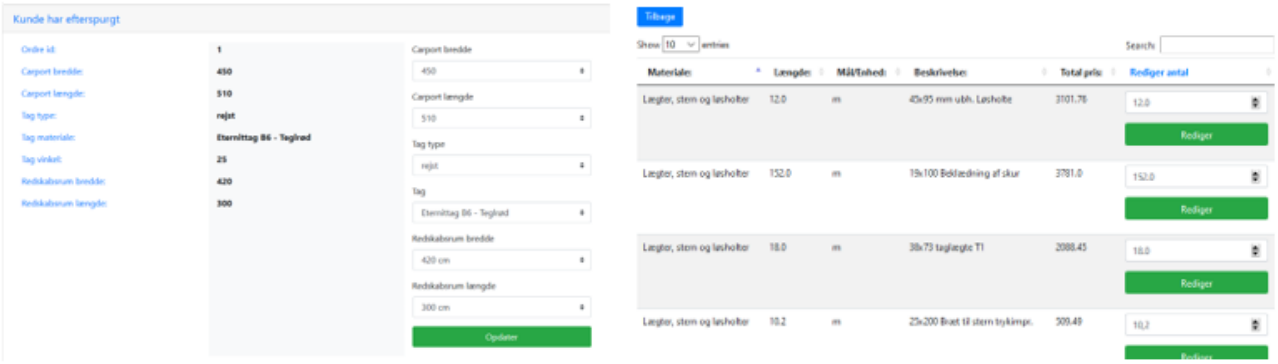
#7 Fog: spec. carport (Redigering)

Som Fog-medarbejder
Vil jeg kunne redigere i en kundeforespørgsel således, at en kundens rettelser bliver tilføjet

Accept kriterier:
Carportens mål, vinkler og valgmuligheder skal kunne ændres på den individuelle forespørgsel

Estimat:
Front end: L
Back end: XL

How-to:
Home -> Log Ind -> Log ind -> Åben -> Se tegning og stykliste
Fra forsiden trykkes der på Log ind i øverste højre hjørne. Derefter indtastes admin bruger info
Brugernavn: <Se afsnit 0>
Kodeord: <Se afsnit 0>
og der trykkes på log ind.
Herefter trykkes der på ”åben” ud for en forespørgsel i højre side.
På forespørgsel siden kan man derefter opdatere kundens valg af mål, materier samt info.
Derefter kan man nederst på forespørgsel siden trykke på ”se tegning og stykliste”. Hvorefter man også kan opdatere styklisten.



Figur 3 - Tegning og stykliste

Tasks

#140 Select option værdier

Dropdown menuer skal have værdier

#137 Fog: Opdatering af stykliste mål

Stykliste siden skal kunne redigere i mål og antal af varer

#114 redigere mål på en forespørgsel

Forespørgselsiden skal kunne redigere i mål og valg i en forespørgsel og sende dem videre til datamapper

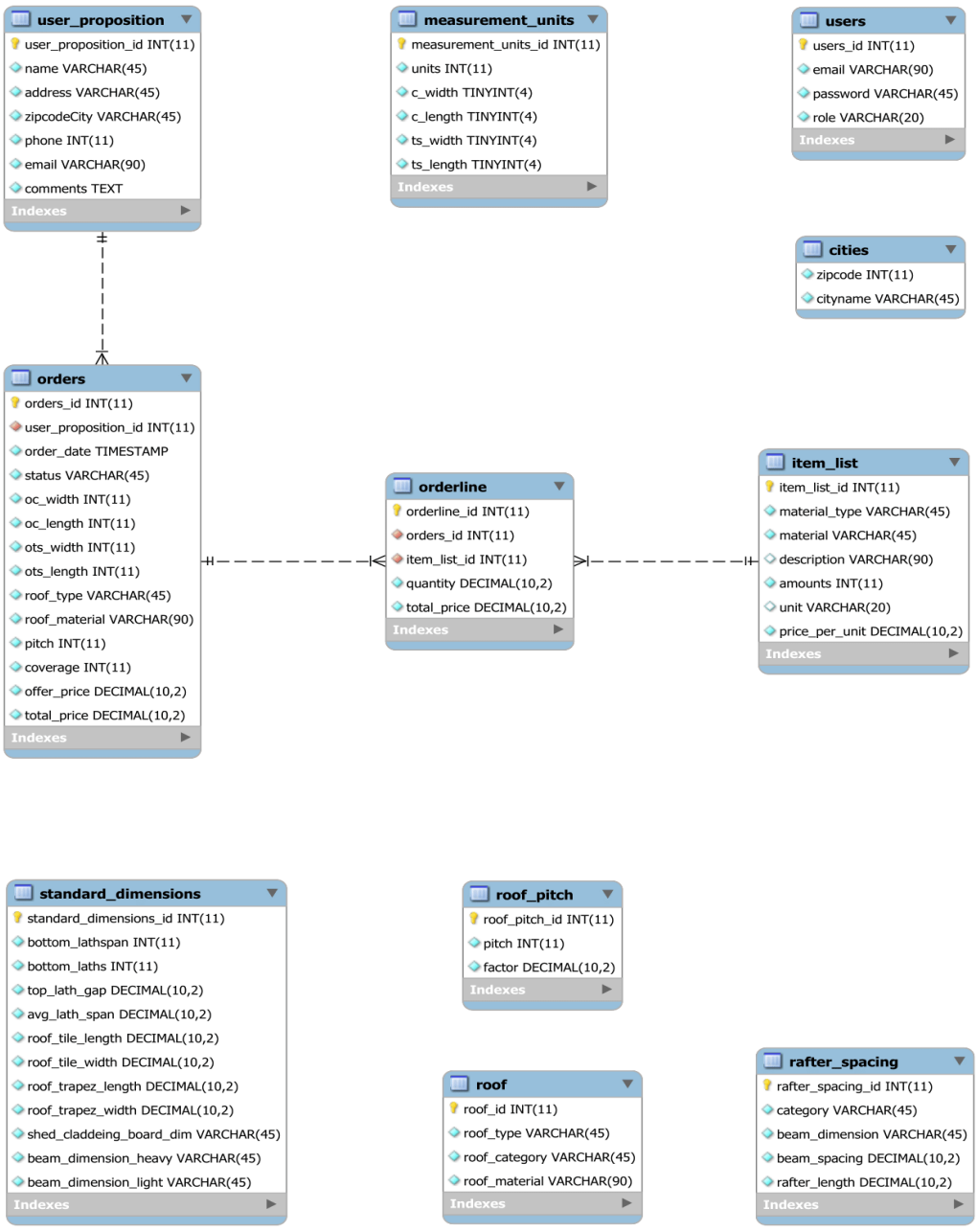
#138 Fog: Opdatering af stykliste antal fra orderline id

Databasequery skal laves for at opdatere mål udfra orderline id

3 Diagrammer

3.1 ER Diagram

Databasen er designet med tabeller og relationer, som overholder 3. normalform. Vores databasetabeller er delt op i to. En relationstabel og en del løse tabeller, der indeholder forskellige måleenheder til brug for udregning af carport.



Figur 4 - ER Diagram

Det er ikke alle tabellerne, der overholder 3. normalform. I selve `orders`-tabellen har vi valgt at indsætte værdierne fra `measurement_units`, `roof` og `roof_pitch`, som derfor vil give redundans.
En af grundene til valg af denne metode er for at gemme kundens historik. Så hvis Fog retter eller sletter et felt i en af tabellerne `measurement_units`, `roof` eller `roof_pitch`, vil ændringerne ikke indvirke på kundens valg.
En anden grund vil være, at en fuldt normaliseret database kunne medføre, at selv de mest simple udtræk skal ske over mange joins, hvilket ofte vil være langsommere at eksekvere end en tabel, som ikke overholder alle normalformer.
Hvis `orders`-tabellen skulle være uden redundans, vil man være nødt til at joine ni tabeller og ikke kun tre for hver gang, der skal trækkes data ud. Man vil også være nødt til enten at skille `measurement_units`-tabellen i fire, én til hver måleenhed eller lave en tabel, der linker mellem `orders` -og `measurement_units`-tabellen.

Tabellerne `user_proposition` og `orders` er sat op som 1-mange relation, men bliver i vores system kun brugt som 1-1 relation. Grunden til, at det kun bliver brugt som 1-1 relation, er, at en kundeforespørgsel er unik og ikke bliver genbrugt. Man vil i vores tilfælde sagtens kunne slå begge tabeller sammen, men vi vælger her at separere informationerne, så kundens personlige data som navn, adresse, telefonnummer etc. ikke blandes med de tekniske data fra kundens indtastning.
Hvis Fog på et tidspunkt vælger at bruge kundeoplysningerne til at oprette en bruger, vil det hermed være muligt at tilknytte flere ordrer til `user_proposition`-tabellen.

Den eneste tabel, som ikke har et automatisk generet ID, er `cities`-tabellen. Det var meningen, at den skulle bruges i `user_proposition`-tabellen, men Fog har valgt, at postnummer og by skal være et frit skrivefelt og ikke separeres i to felter.
Vi har dog tilføjet tabellen `cities`, som indeholder alle danske postnumre med dertilhørende bynavne. Hvis Fog senere hen vælger at tilføje en mere dynamisk indtastning af postnummer og bynavn, er databasen klar.

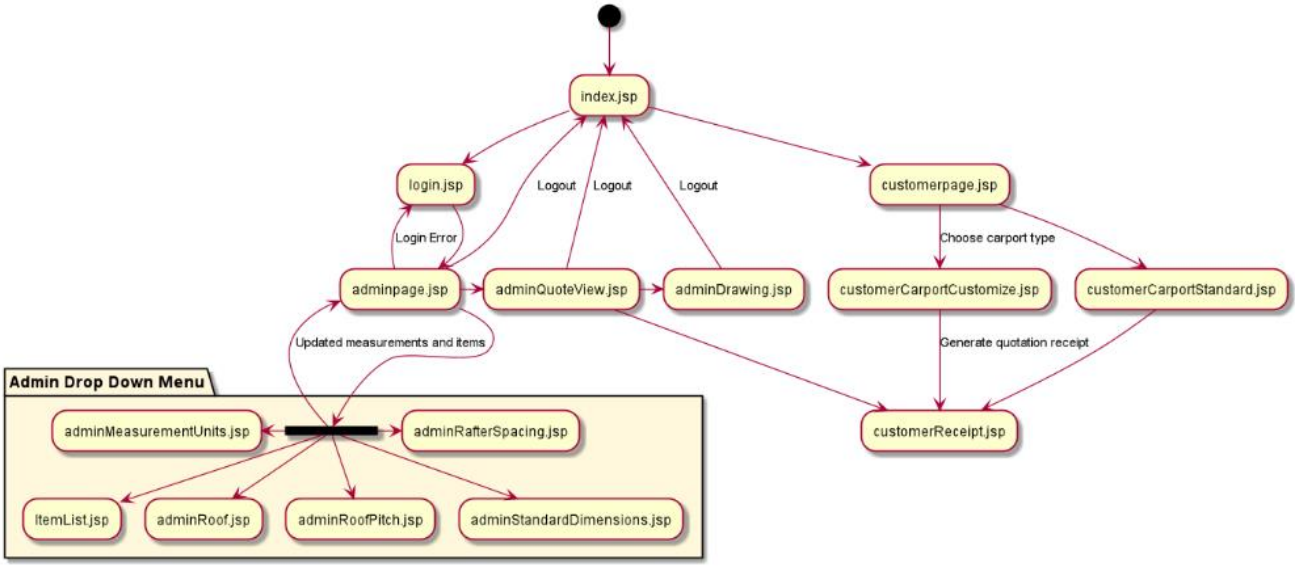
Alle tabellerne i databasen har en PRIMARY KEY, som er auto increment, undtagen `cities`-tabellen.
Ved at bruge auto increment skaber man en unique nøgle for hver række i den enkelte tabel. Denne nøgle kan hermed bruges til at ændre, slette eller få vist en specifik række i tabellen.

I vores relationstabel er der FOREIGN KEYs, der refererer til PRIMARY KEYs.
FOREIGN KEYs tillader krydshenvisende relaterede data på tværs af tabeller, og ved brug af CONSTRAINT, holde de relaterede data konsistente.
Da data i `user_proposition`, `orders` og `orderline` refererer til hinanden, er der tilføjet ON DELETE CASCADE og ON UPDATE CASCADE. Det vil sige, at alle FOREIGN KEYs, der referer til slettede, også skal slettes eller at alle FOREIGN KEYs, der referer til ændrede felter, skal ændres.

Hvis f.eks. en Fog-medarbejder sletter en kundeforespørgsel fra `user_proposition`-tabellen, slettes også den tilhørende ordre i `orders`-tabellen og alle ordrelinjer i `orderline`-tabellen.
Tabellen `item_list` har også en relation til `orderline`-tabellen, men her skal der hverken slettes eller ændres, hvis f.eks. en kundeforespørgsel bliver slettet.
For at undgå, at der ændres eller slettes i `item_list`- tabellen, tilføjer vi ON DELETE RESTRICT og ON

UPDATE RESTRICT.
På denne måde bliver det let at ændre og slette i tabellerne, da man kun behøver én query i stedet for tre.

3.2 Navigationsdiagram



Figur 5 - Navigationsdiagram

Ovenstående diagram afspejler navigationen gennem Fogs hjemmeside, som enten kunde eller Fog-medarbejder. Dette er beskrevet for at få en bedre forståelse af brugeroplevelsen på siden.

Brugeren starter på velkomstsiden index.jsp hvor der vises en navigation bar i toppen af siden.
I denne navigation bar er det muligt for brugeren at trykke Home, Bestil eller Log ind.
Trykker brugeren Home bliver siden re-directed til velkomstsiden igen og fungerer på samme måde på samtlige jsp sider.

Er brugeren en kunde har brugeren mulighed for at trykke på Bestil i navigationsbaren og sendes derved hen til customerpage.jsp hvor der er følgende to valgmuligheder.

- Den ene er “Bestil - pr. stk. 23.998,-” (customerCarportStandard.jsp) hvor det er tiltænkt at kunden skal kunne bestille carporte på standardmål. Denne mulighed er dog ikke implementeret da product owner ikke ønskede dette på nuværende tidspunkt.

- Klikker brugeren derimod på “Byg selv” (customerCarportCustomize.jsp) sendes brugeren til en side hvor det gør det muligt for brugeren at bestille en carport med specielmål samt tagtype- og farve. Efter kunden har tastet alle mål, valg, oplysninger og bemærkninger har kunden mulighed for at trykke på send. Derved sendes kunden hen til en kvitteringsside (customerReceipt.jsp) hvor kunden modtager en kvittering på at deres forespørgelse er gennemført. Denne kvittering kan også printes hvis kunden vil det.

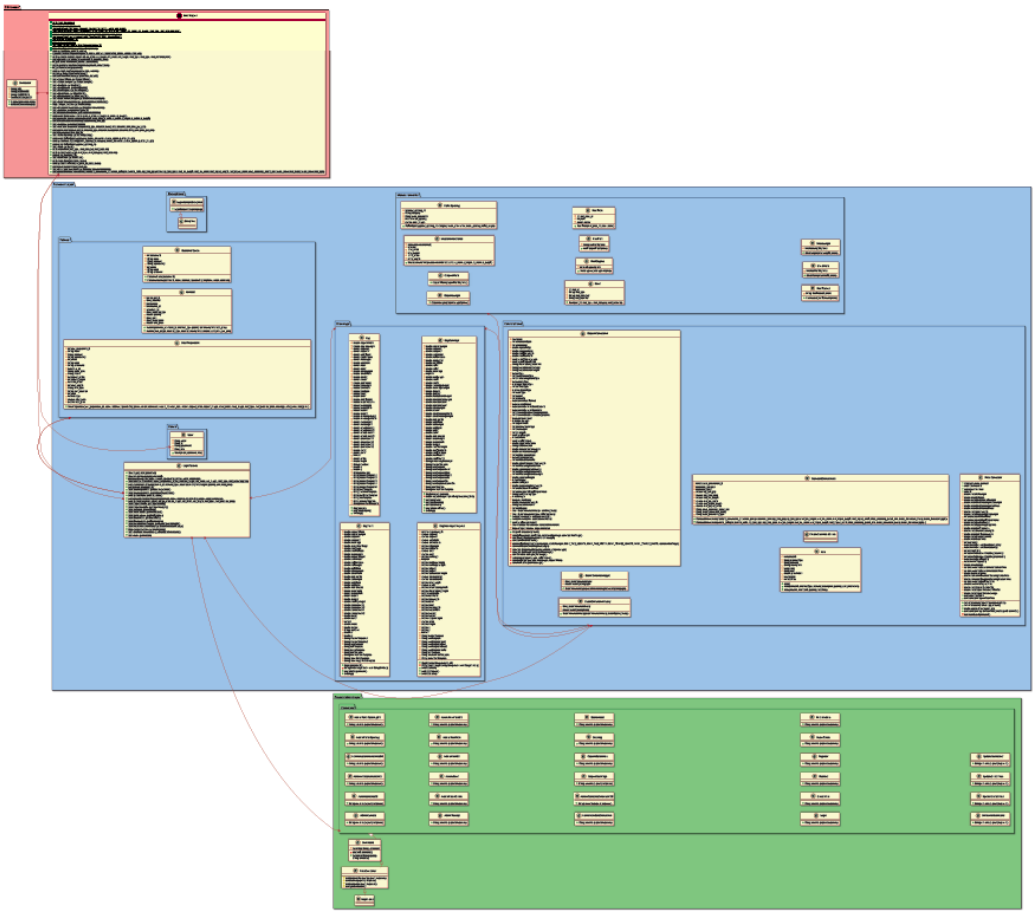
Går vi tilbage til udgangspunket index.jsp og lader nu en Fog-medarbejder logge ind på hjemmesiden, bliver Fog-medarbejderen sendt hen til adminpage.jsp hvor der gives en række valgmuligheder.
Navigationsbaren ændrer sig og indeholder nu en forespørgelsesoversigt samt en drop-down menu.
Hvis Fog-medarbejderen vil kigge, rette eller slette en bestemt forespørgelse kan han trykke på knapperne "Åben" for at se og rette eller "Fjern" for at fjerne en forespørgelse.

Trykker Fog-medarbejderen på "Åben", sendes medarbejderen hen til adminQuoteView.jsp hvor der kan rettes og slettes i de informationer kunden har indtastet. Derudover har Fog-medarbejderen også mulighed for at trykke på knappen "Se tegning & stykliste", som viser tegningen af den pågældende carport.
Her sendes medarbejderen hen til adminDrawing.jsp hvor tegningerne og stykliste bliver genereret og vist.

Hvis Fog-medarbejderen vil se samme kvittering som kunden kan medarbejderen trykke på "Se tilbud" og bliver dermed sendt til customerReceipt.jsp.

Til sidst har Fog-medarbejderen mulighed for at gøre brug af drop-down menuen hvor der kan rettes følgende ting: Måleenheder (adminMeasurementUnits.jsp), Vareliste (ItemList.jsp), Spærafstand, (adminRafterSpacing.jsp), Tagbeklædning (adminRoof.jsp), Taghældning (adminRoofPitch) og Standardmål (adminStandardDimensions.jsp).

3.3 Klassediagram



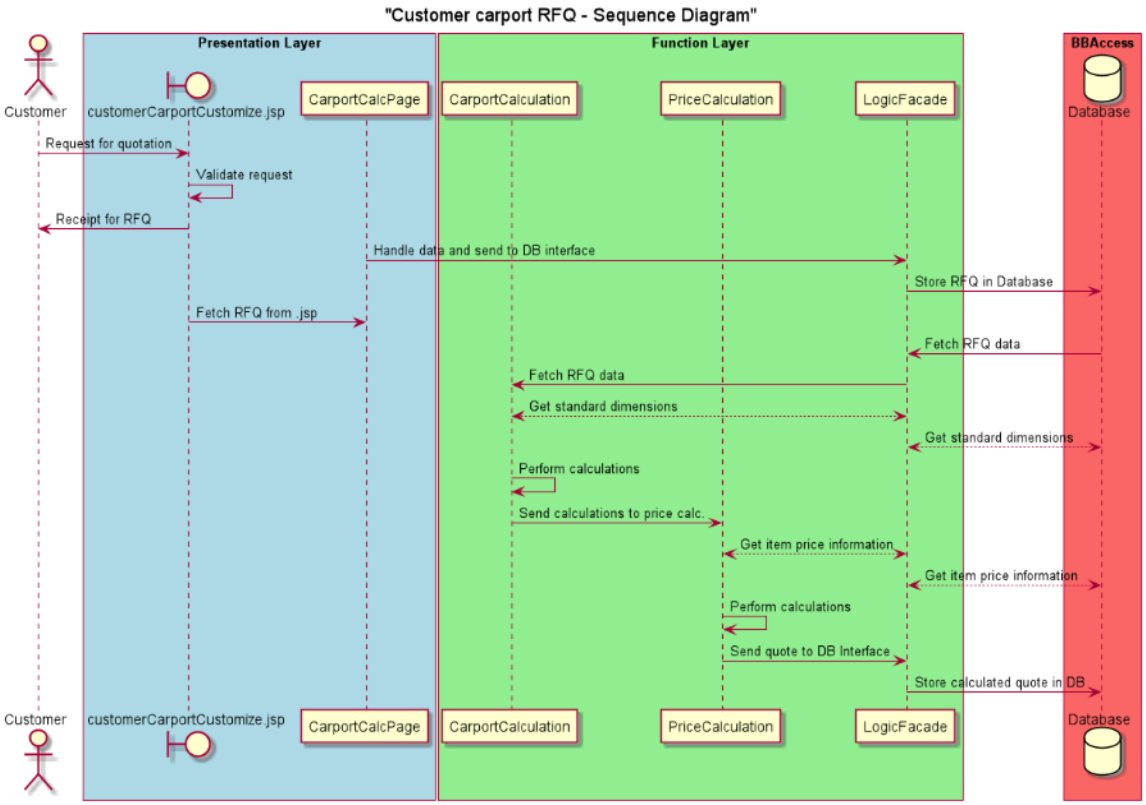
Figur 6 - Klassediagram (Se bilag)

Klassediagrammet er taget med i rapporten for at skabe et overblik over systemets klasser, alle deres metoder og packages. Klassediagrammet er blevet udarbejdet i plantUML og er vedlagt som bilag.

Klasserne er inddelt i forskellige packages i forskellige lag af systemet. Grunden til at vi har inddelt i underpackages i 3 lag (DBAccess, FunctionLayer & PresentationLayer) af command pattern skabelonen, er for at skabe en struktureret, overskuelig opbygning og læsning af koden.

Koden er som sagt designet efter en command pattern designskabelon der gør brug af *servlets*, *frontcontroller* samt en *command* klasse. Dette er visualiseret ved hjælp af et klassediagram. Dette ses blandt andet ved de røde pile på klassediagrammet, der binder de enkelte klasser sammen og viser deres afhængighed af hinanden.

3.4 Sekvensdiagram



Figur 7 - Sekvensdiagram - RFQ

I ovenstående sekvensdiagram er kundeforespørgselsprocessen (RFQ – Request for quotation) illustreret. Diagrammet beskriver trin-for-trin, hvad der sker når kunden lægger en forespørgsel.

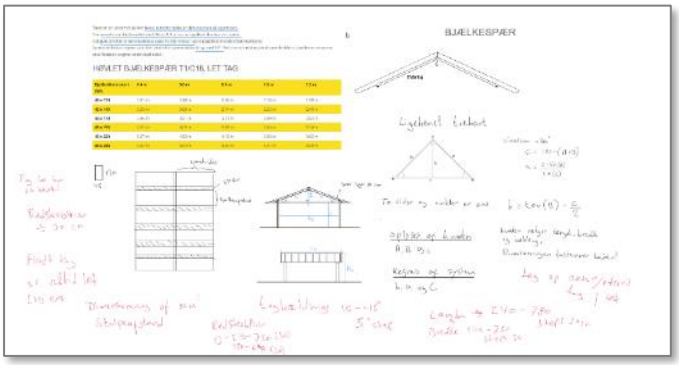
1. *Request for quotation*: Kunden opretter forespørgslen
2. *Validate request*: jQuery script på siden validerer kundens indtastning.
3. *Receipt for RFQ*: Kunden modtager en kvittering for hans forespørgsel
4. *Fetch RFQ from .jsp*: Klassen CarportCalcPage henter de indtastede data fra formularen på .jsp siden.
5. *Handle data and send to DB interface*: Klassen sender kundens forespørgselskriterier til LogicFacade, som håndterer data imellem databasen og resten af klasserne.
6. *Store RFQ in Database*: LogicFacade gemmer forespørgslen i databasen.
7. *Fetch RFQ data*: Henter RFQ data fra igennem LogicFacade til CarportCalculation klassen
8. *Get standard dimensions:* Udveksler løbende data mellem databasen, igennem LogicFacade, for at foretage beregninger.
9. *Perform Calculations*: Udregninger samtlige mål og antal emner til carporten.
10. *Send calculations to price calc:* Sender udregninger til PriceCalculator
11. *Get item price information:* Henter aktuelle priser på samtlige varer igennem LogicFacade
12. *Perform calculations*: Udregner prisen for den samlede carport
13. *Send quote to DB interface*: Sender udregningen til LogicFacade
14. *Store calculated quote in DB:* Gemmer den samlede pris inkl. stykliste i databasen

4 Særlige forhold

4.1 Carport kalkulationen

Gruppen har fra starten arbejdet med følgende forudsætning for øje:

"Vi vil ikke levere mindre funktionalitet, end Fog har i dag".



Figur 8 - Noter og brainstorming ifbm. udregninger

Derfor blev det tidligt besluttet, at komme så tæt på de rigtige (Og direkte anvendelige) beregninger fra start. Dette medførte et forholdsvis stort research-arbejde og mundede ud i konkrete udregninger programmet skal foretage når carporten skal beregnes.
Gevinsten heraf er, at alle generede SVG-tegninger er dynamiske, ligesom størstedelen af carportens dimensioner og stykliste tager højde for statikberegninger samt brugerens ønsker.
De få ting som ikke er dynamiske er baseret på realistiske antagelser fra Fog's eget produktkatalog.
Det drejer sig bl.a. om størrelser på forskellige tagsten og trapezplader. Disse er udeladt da andre beregninger og opgaver blev prioriteret frem for disse. Projektet indeholder flere mål og fagudtryk som er illustreret i appendix 11.7.

For Carport kalkulationen dokumenteres udregningerne samt et eksempel derfor, i dette afsnit.
Udvalgte sektioner af den tilhørende, funktionelle kode vil blive gennemgået i afsnit "5.1 Carport Calculation"

Om dimensionering
Når man bygger en carport skal de bærende søjler samt tagspærene dimensioneres. Parameteret for dimensionering udgøres af tagkonstruktionens vægt, ofte regnet i kilo newton [kN].
En tommelfingerregel[1] siger at:

- Et let tag vejer maks. 25 kg / m^2
- Et tungt tag vejer maks. 45 kg / m^2

Typen af tagkonstruktionen falder derfor indenfor kategorien "Let" eller "Tung".
For Fogs udbudsmateriale er tilfældet typisk, at en carport med "Rejst" tag falder i kategorien "Tung" og at en carport med et fladt tag falder i kategorien "Let".

[1] http://www.icopal.dk/~/media/UploadFolder/Products/DK/IcopalDK/ProductLibrary/Bitumen%20Membranes/SBS%20TopBase/6001altomtag0612.pdf

Dimensionering af søjler

Afhængigt af, om taget er let eller tungt anvender man desuden forskellige søjler og spærdimensioner.
For søjler kan man slå op i en dimensioneringstabel[2] for at finde passende dimensioner til sit byggeprojekt.
Til dette projekt anvendes følgende søjler:

- Til let tag: 100x100x25000 mm
- Til tungt tag: 125x125x25000 mm

Alle søjler er 2,5 m høje, nøjagtigt ligesom Fogs udbud. Søjlernes dimension betyder desuden, at de er dimensioneret til hhv. let og tungt tag, inkl. en sikkerhedstolerance.
Standardstørrelserne Fog udbyder er magen til de størrelser programmet kalkulerer med. Det betyder, at en Carport ikke kan blive større end maksimalt 750 x 780 cm – Denne størrelse kan med de, dimensionerede søjler understøttes af fire stolper. Af samme grund er programmet bygget op således, at en carport altid vil bestå af fire søjler, medmindre at kunden har tilvalgt skur – I så fald vil carporten bestå af otte søjler.

Dimensionering af spær

At dimensionere spær er en proces der kræver flere trin.
Afhængigt af tages hældning udregnes spærets længde vha. simpel trigonometri, som afhænger af kundens specifikationer. Disse udregninger resulterer i en spærlængde. Spærafstanden skal herefter beregnes for at sikre et holdbart tag. Til dette ganges spærets længde med en hældningsfaktor, hvorefter resultatet af dette anvendes til at slå op i en tabel, som både angiver den nødvendige spærafstand samt spærets dimensioner.
Heri skal den korteste spærafstand vælges, uden at være lavere end tabelopslaget. Et eksempel på anvendelsen af tabellerne findes sidst i dette afsnit.
De tre tabeller[3] er angivet nedenfor:

	Spærafstand ved spærlængde (Let tagkonstruktion)				
Bjælkedimension i mm	**0,4 m**	**0,6 m**	**0,8 m**	**1,0 m**	**1,2 m**
45 x 120	2,81	2,48	2,26	2,10	1,98
45 x 195	4,52	4,02	3,68	3,44	3,24

	Spærafstand ved spærlængde (Tung tagkonstruktion)				
Bjælkedimension i mm	**0,4 m**	**0,6 m**	**0,8 m**	**1,0 m**	**1,2 m**
45 x 120	2,43	2,13	1,93	1,79	1,68
45 x 195	3,94	3,48	3,18	2,95	2,78

[2] http://flexwood.dk/wp-content/uploads/2012/03/Flexwood-beregningstabel.pdf
[3] https://www.ringstedspaer.dk/konstruktioner/bjaelkespaer

Gangefaktor for spærafstand							
Hældning	**<=15°**	**20°**	**25°**	**30°**	**35°**	**40°**	**45°**
Faktor	1	0,97	0,94	0,89	0,84	0,79	0,72

Med antallet af spær og søjler, er de mest komplicerede udregninger nu foretaget, og programmet kan udregne resten. Specifikt udregner programmet følgende:

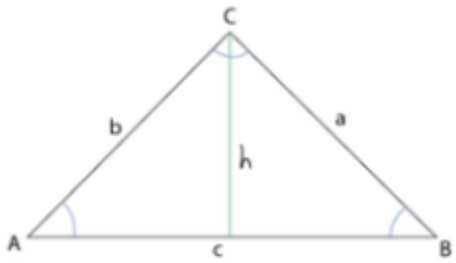
Vare (Vareteksten er simplificeret)	**Udregningsmetode (Dimensioner)**	**Udregningsmetode (Stk. antal)**	**Tilhører (Rejst, fladt, skur)**
Diverse beslag til hele carporten	Antaget	Antaget	Alle
38x73 taglægte T1	Udregnet	Udregnet	Alle
25x200 Bræt til stern trykimpr.	Udregnet	Udregnet	Alle
45x195 Rem ubh.	Udregnet	Udregnet	Alle
Skruepakke til universalbeslag + toplægte	Antaget	Antaget	Alle
45x120 Tagspær trykimpr.	Udregnet	Udregnet	Alle
100x100 Stolpe trykimpr.	Udregnet	Udregnet	Alle
109 x 240 tagplade	Antaget	Udregnet	Fladt tag
Skruer til tagplader	Antaget	Antaget	Fladt tag
Skruepakke til taglægter	Antaget	Antaget	Rejst tag
20,4 x 23,6 tagsten	Antaget	Udregnet	Rejst tag
Komplet dør sæt til skuret	Antaget	Antaget	Skur
Skruepakke til mont. af bræt ved beklædning	Antaget	Antaget	Skur
Skruepakke til mont. af bræt ved beklædning	Antaget	Antaget	Skur
45x95 mm ubh. Løsholte	Udregnet	Udregnet	Skur
19x100 Beklædning af skur	Udregnet	Udregnet	Skur

Ovenstående linjer anvendes til at generere stykliste, pris samt dynamiske SVG-tegninger.

Betragter man carportens tag forfra vil man se at den er udformet som en ligebenet trekant.
Derfor kan længden af spær regnes ud ved at anvende formler der gælder for en ligebenet trekant.

Kunden oplyser selv følgende mål:

- Tagets hældning (Vinkel) illustreret nedenfor i vinkel "A" og "B".
- Carportens, og dermed tagets, bredde illustreret nedenfor med "c".



Figur 9- Ligebenet trekant

Programmet skal derfor udregne:

- Længden på spær illustreret på figur 9 med hhv. "b" og "a". Vi ved at trekanten er ligebenet hvorfor "a" må svare til "b".
- Højden på den samlede konstruktion illustreret i figur 9 med "h".

Tagets kip (Vinkel C) udregnes: $C = 180 - (A - B)$

Spærets længde: $a = \frac{c \cdot sin(A)}{sin(C)} \cdot hældningsvinkel$

Tagkonstruktionens højde: $h = tan(B) \cdot \frac{c}{2}$

Med disse formler kan vi nu udregne det væsentligste.

Udregnet eksempel

Lad os antage at kunden bestiller en carport med målene 350 x 510 cm, rejst tag på 15° samt et skur på 270 x 240 cm.

Udregning af tagets kip: $C = 180 - (15 - 15)$

$C = 150$

Udregning af spærets længde: $a = \frac{350 \cdot sin(15)}{sin(150)}$

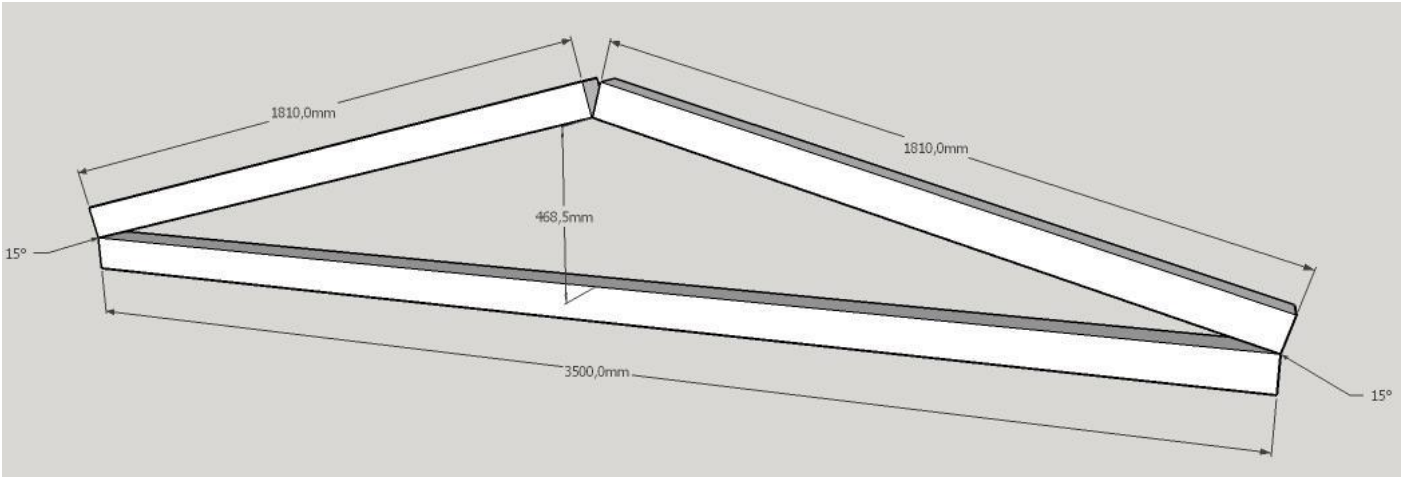
$a = 181{,}17 \cdot 1$

$a = 181{,}17\ cm\ /\ 1\ ,81m$

Udregning af tagkonstruktionens højde: $h = tan(15) \cdot \frac{350}{2}$

$h = 46{,}85\ cm\ /\ 0{,}46m$

Eksemplet kan bl.a. verificeres ved at tegne spærkonstruktionen i målfast 3D

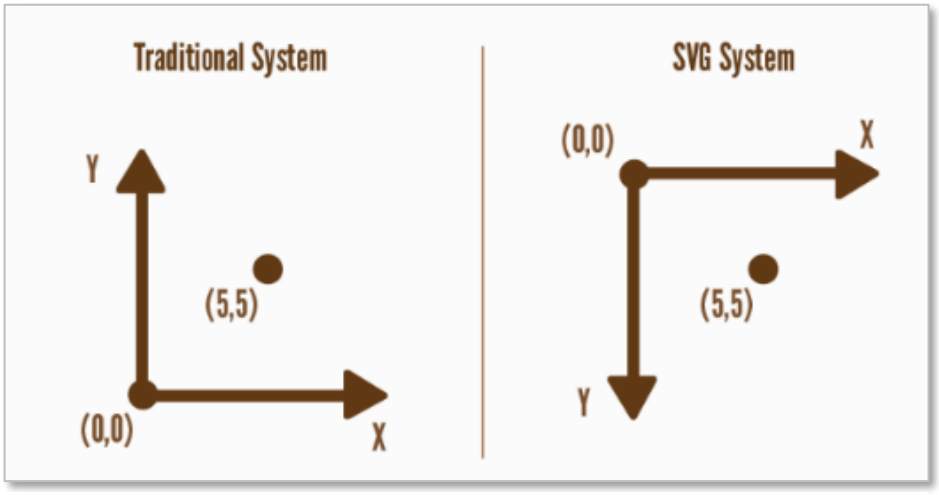


Figur 10 - Målfast 3D tegning

De øvrige mål for carporten er forholdsvis simple, sammenlignet med de ovennævnte udregninger hvorfor de ikke gennemgås i rapporten.

4.2 Generering af tegninger

Vi har i projektet gjort brug af Scalable Vector Graphics også kalde SVG, som er vector baseret grafik i XML format. Det bruger vi i dette tilfælde til at tegne dynamiske tegninger ved hjælp af firkanter og linjer i et koordinatsystem. I modsætning til et almindeligt koordinatsystem, fungerer SVG systemer lidt anderledes hvilket gør at man roterer aksen som vist på figur 11.
Det smarte ved SVG er at det skalerbart og kan vises i høj kvalitet i enhver opløsning, hvilket gør det ideelt til tegninger af carportene i vores projekt.
De specifikke forhold der gør sig gældende er beskrevet i afsnittet om udvalgte kodeeksempler.



Figur 11 - SVG Koordinatsystem

5 Udvalgte kodeeksempler

5.1 Carport Calculation

Teorien bag "CarportCalculation" klassen blev gennemgået i afsnit "4.1 Carport Kalkulationen" og vil blive yderligere forklaret i, kodemæssigt i dette afsnit.

Standardmål

Tabellerne i afsnit "4.1 Carport Kalkulationen" med standardmålene er oprettet som tabeller i databasen, således at programmet kan slå op i disse for at generere en korrekt, dynamisk beregning. Der er bl.a. oprettet én tabel til spærafstande og en til "Vinkelfaktoren" (Også gennemgået i afsnit "4.1 Carport Kalkulationen"). Disse tabeller blev oprettet med følgende SQL:

```
CREATE TABLE IF NOT EXISTS `fogdb`.`rafter_spacing` (
  `rafter_spacing_id` INT(11) NOT NULL AUTO_INCREMENT,
  `category` VARCHAR(45) NOT NULL,
  `beam_dimension` VARCHAR(45) NOT NULL,
  `beam_spacing` DECIMAL(10,2) NOT NULL,
  `rafter_length` DECIMAL(10,2) NOT NULL,
  PRIMARY KEY (`rafter_spacing_id`))
ENGINE = InnoDB
AUTO_INCREMENT = 1
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
```

Figur 12 - tabel "rafter_spacing"

```
CREATE TABLE IF NOT EXISTS `fogdb`.`roof_pitch` (
  `roof_pitch_id` INT(11) NOT NULL AUTO_INCREMENT,
  `pitch` INT(11) NOT NULL,
  `factor` DECIMAL(10,2) NOT NULL,
  PRIMARY KEY (`roof_pitch_id`))
ENGINE = InnoDB
AUTO_INCREMENT = 1
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
```

Figur 13 – Tabel "roof_pitch"

rafter_spacing_id	category	beam_dimension	beam_spacing	rafter_length
1	let	45 x 120	0.40	2.81
2	let	45 x 120	0.60	2.48
3	let	45 x 120	0.80	2.26
4	let	45 x 120	1.00	2.10
5	let	45 x 120	1.20	1.98
6	let	45 x 195	0.40	4.52
7	let	45 x 195	0.60	4.02
8	let	45 x 195	0.80	3.68
9	let	45 x 195	1.00	3.44
10	let	45 x 195	1.20	3.24
11	tung	45 x 120	0.40	2.43
12	tung	45 x 120	0.60	2.13

Figur 14 - Spærafstand tabel data (Udsnit)

roof_pitch_id	pitch	factor
1	15	1.00
2	20	0.97
3	25	0.94
4	30	0.89
5	35	0.84
6	40	0.79
7	45	0.72

Figur 15 - Vinkelfaktor tabel data

Når systemet har udregnet spærlængde og determineret om der er tale om et "Let" eller "Tungt" tag, dikterer det byggetekniske spærafstanden. Denne undersøges ved at gange spærlængden med vinkelfaktoren som beskrevet i afsnit 8 og lave et opslag i tabellen. Programmet skal finde en *rafter_length* som kommer tættest på den udregnede spærlængde, uden at være under denne.

```java
/**
 * Calculates the raft length
 *
 * @param carportWidth        Customer selected carport width.
 * @param customerRoofAngle   Customer selected roof slant angle.
 * @param calculatedRoofAngle The calculated upper roof angle (Comes from calcRoofAngle().
 */
public void calcRaftLength(double carportWidth, int customerRoofAngle, int calculatedRoofAngle) {

    //Determine sloped raft length
    double custRoofAngleRadian = Math.toRadians(customerRoofAngle);
    double calcRoofAngleRadian = Math.toRadians(calculatedRoofAngle);
    double calcRaftLength = (carportWidth * Math.sin(custRoofAngleRadian)) / (Math.sin(calcRoofAngleRadian));
    calcRaftLength = calcRaftLength * angleAndFactor.get(customerRoofAngle);
    this.raftLength = calcRaftLength;
```

Figur 16 - metode "calcRaftLength"

Metoden *calcRaftLength* tager tre parametre. Carportens længde, samt hhv. kundens hældningsvinkel og den udregnede vinkel i tagets kip.
For at anvende sinus i Javas *Math* klasse skal vinklerne konverteres til radianer.
Herefter ganges spærlængden med hældningsfaktoren som er hentet fra databasen og gemt som et *"HashMap"*.
Ved at "Matche" kundens hældningsvinkel med tilsvarende i dette hashmap, findes vinkelfaktoren nemt.

Programmet skal nu finde spærafstanden i databasetabellen *rafter_spacing*.

```java
if (roofHeavy) {
    raftLengthAdjust = raftLength / 100;
    raftDistance = LogicFacade.getBeamDimensionHeavy(raftLengthAdjust).get(0).getBeamSpacingHeavy();
    raftDimension = LogicFacade.getBeamDimensionHeavy(raftLengthAdjust).get(0).getBeamDimensionHeavy();
    raftType = 21;
} else {
    if(!raisedRoof){
        raftDistance = 1;
        raftDimension = "45 x 195";
        raftType = 20;
    } else {
    raftLengthAdjust = raftLength / 100;
    raftDistance = LogicFacade.getBeamDimensionLight(raftLengthAdjust).get(0).getBeamSpacingLight();
    raftDimension = LogicFacade.getBeamDimensionLight(raftLengthAdjust).get(0).getBeamDimensionLight();
    raftType = 20;
    }
}
```

Figur 17 - determinering af spærafstand

Programmet har undersøgt om der er tale om en tung eller let tagkonstruktion på dette tidspunkt.
I Fogs tilfælde, vil udbuddet af carporte med flade tage, altid udgøre lette tagkonstruktioner.

Fladt tag

Er der tale om et let, fladt tag er det givet, at spærafstanden svarer til carportens bredde.
Spærafstanden sættes til 1m og spæret dimensioneres til et standardmål der passer til samme.
raftType svarer direkte til spærets varenummer i databasen.

Rejst tag

Er der tale om et rejst tag skal programmet slå op i databasen efter den rigtige spærafstand. Først divideres spærlængden med 100 for at konvertere den til meter så den svarer til databaseformatet. Herefter laves opslaget igennem *LogicFacade* som indeholder den SQL der håndterer søgningen, og returneringen fra databasen. Der kaldes to metoder som returnerer hhv. spærets dimension og spærafstanden.

```
/**
 * Get beam-dimension and beam-spacing for light roof
 *
 * @param rafterLength rafter length
 * @return beamDimensionLight object
 * @throws LoginSampleException LoginSampleException
 */
public static List<BeamDimensionLight> getBeamDimensionLight(double rafterLength) throws LoginSampleException {
    List<BeamDimensionLight> beamDimensionLight = new ArrayList<>();
    try {
        Connection con = Connector.connection();
        String SQL = "SELECT beam_dimension,beam_spacing FROM rafter_spacing WHERE category = 'let' and rafter_length >= ?" +
                "ORDER BY rafter_length ASC LIMIT 1";
        PreparedStatement ps = con.prepareStatement(SQL);
        ps.setDouble( parameterIndex: 1, rafterLength);
        ResultSet rs = ps.executeQuery();
        while (rs.next()) {
            String beamDimension = rs.getString( columnLabel: "beam_dimension");
            double beamSpacing = rs.getDouble( columnLabel: "beam_spacing");
            BeamDimensionLight bd = new BeamDimensionLight(beamDimension, beamSpacing);
            beamDimensionLight.add(bd);
        }
    } catch (ClassNotFoundException | SQLException ex) {
        throw new LoginSampleException(ex.getMessage());
    }

    return beamDimensionLight;
}
```

Figur 18 - SQL statement til spær

I ovenstående figur ses et SQL opslag i kategorien ”Let” (Metoden er ens for kategorien ”Tung”). Metoden søger, som tidligere beskrevet, efter den spærlængde som kommer tættest på den udregnede, uden at være under. Resultatet, i form af spærdimension og spærlængde gemmes i en *ArrayList* som returneres til CarportCalculation klassen igennem LogicFacade.

På denne måde returneres den rette spærafstand altid.

5.2 Price Calculation

Priskalkulationsklassen foretager forholdsvis simple udregninger. Carportkalkulationen har på forhånd udregnet antal og / eller mængder af de forskellige, nødvendige materialer. Klassen ganger derfor, afhængigt af varen, antal, mængde samt størrelse sammen med prisen per enhed, hvilket resulterer i en totalpris. Der regnes moms på totalprisen og efterfølgende en dækningsgrad, fastsat af Fog-medarbejderen.
Samtlige priser hentes altid fra databasen og kan nemt opdateres i programmets tilhørende CRUD system.

Et væsentligt aspekt ved priskalkulationen er måden hvorpå varen findes i databasen for efterfølgende at blive anvendt. Til dette er der skrevet en "Søgefunktion".

```
public Item itemSearch(int itemID) throws LoginSampleException {

    for (int i = 0; i < log.getItemList().size(); i++) {

        if (itemID == log.getItemList().get(i).getItemListID()) {
            this.item = log.getItemList().get(i);
        }
    }
    return item;
}
```

Figur 19 - PriceCalculator search

Metoden modtager et varenummer, som er fastsat i carportkalkulationsklassen.
Den søger herefter i en liste som består af samtlige varenumre, hentet via LogicFacade i databasen.
Når varenummeret matcher returneres hele varen, *item*, fra listen som indeholder informationer om pris, navn mm. Varen kan nu bruges til udregninger.

5.3 Dynamisk SVG-generering

Måden hvorpå vi har bygget vores kode op omkring SVG-motoren er ved hjælp af en constructor, en metode til at tegne og en StringBuilder hvor vi appender strenge som indeholder de SVG-informationer der skal bruges for at fremvise de enkelte elementer i tegningen.

```
private StringBuilder svg = new StringBuilder();
```

StringBuilder'en bliver altså fodret med en masse strenge som til sidst danner en sammenhængende SVG-tegning på hjemmesiden.

```
private final String rectTemplate       = "<rect transform=\"translate(100,100)\" x=\"%f\" y=\"%f\" height=\"%f\" width=\"%f\" " +
                                          "style=\"stroke:#000000; fill: #ffffff\" />";
```

Ovenstående Streng er en af flere templates vi har lavet for at kunne generere de enkelte strenge såsom denne, der danner en firkant og forskyder den på et kanvas så det bliver placeret det rigtige sted.
Det kunne f.eks. være en lægte som skal placeres på et tag.

```
//Laths
svg.append(String.format(rectTemplate, lathX, lathY, lathLength, lathWidth));
for (int i=0; i <noOfLaths; i++) {
    lathY=lathY+(carportWidth /noOfLaths);
    svg.append(String.format(rectTemplate, lathX, lathY, lathLength, lathWidth));
}
```

I ovenstående eksempel laver vi et for loop for at tegne samtlige lægter på en gang, for så til sidst at appende dem sammen med rectTemplate til stringbuilderen svg. På den måde bliver det tilføjet til de mange strenge som til sidst danner en stor streng med alt SVG-indholdet.
For at bygge hele carporten har vi en metode kaldet addCarport() som indeholder alle vores strenge.
Det er en stor metode som gør brug at samtlige templates samt variablerne fra CarportCalculations.
Disse variabler bliver smidt ind på x og y's pladser i de strenge der skal appendes til stringBuilderen.
Derved får strengene de rigtige værdier med og derved SVG'en de rigtige mål.

```
    c = new CarportCalculation(orderID);

    this.carportWidth = c.getCarportWidth();
    this.carportLength = c.getCarportLength();
    this.noOfRafts = c.getNoOfRafts();
    this.raftDistance = c.getAvgRaftDistance();
    this.raftDistance2 = c.getAvgRaftDistance();
    this.raftLength = c.getCarportWidth();
    this.shedLength = c.getShedWidth();
    this.shedWidth = c.getShedLength();
    this.noOfLaths = c.getNoOfLaths();
    this.lathWidth = c.getCarportLength();
    this.lathSpan = c.getLathSpan();
    this.noOfBeams = c.getNoOfBeams();
    this.roofBargeWidth = c.getCarportLength();

    //If else for handling viewbox size
    if(carportWidth<400 && carportLength<400) {
        svg.append(String.format(headerTemplate1));
    }else if(carportWidth<500 && carportLength>500) {
        svg.append(String.format(headerTemplate2));
    }else if(carportWidth<600 && carportLength<600) {
        svg.append(String.format(headerTemplate3));
    }else if(carportWidth<700 && carportLength<700) {
        svg.append(String.format(headerTemplate4));
    }else if(carportWidth<800 && carportLength<800) {
        svg.append(String.format(headerTemplate5));
    }
}
```

For at kunne fylde vores SVG-motor med alle de mål der skal til, for at tegne en carport, laves der en instans af CarportCalculation klassen. Herefter tildeler vi de lokale variabler i Svg.java, værdier fra CarportCalculations, så vi får de dynamiske mål, som passer på kundes ønsker. Alt dette bliver gjort i en constructor som laver en headerTemplate baseret på carportstørrelsen således at vores viewbox på hjemmesiden bliver tilpasset i størrelse.

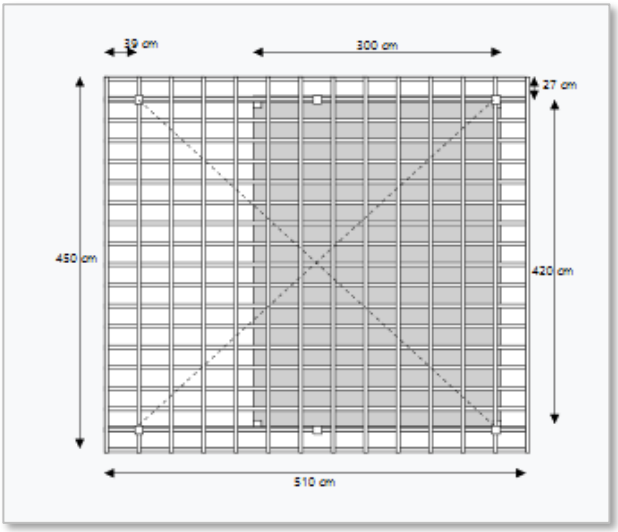
Ret hurtigt rendte vi ind i et browserproblem med hensyn til kommaer og punktummer. Dette løste vi ved hjælp af en replace som tager alle kommaer og erstatter dem med punktummer for at tegningen kunne vises i en browser. Dernæst har vi nogle faste værdier i vores templates som igen skulle konverteres tilbage til kommaer for at de fungerede i SVG formatet i browseren.

```
@Override
public String toString() {
    String res = svg.toString().replace( target: ",", replacement: ".");
    res = res.replace( target: "translate(100.100)", replacement: "translate(100,100)");
    res = res.replace( target: "M0.0 L12.6 L0.12 L0.0", replacement: "M0,0 L12,6 L0,12 L0,0");
    res = res.replace( target: "M0.6 L12.0 L12.12 L0.6", replacement: "M0,6 L12,0 L12,12 L0,6");
    res = res.replace( target: "0.0.550.550", replacement: "0,0,550,550");
    res = res.replace( target: "0.0.650.650", replacement: "0,0,650,650");
    res = res.replace( target: "0.0.750.750", replacement: "0,0,750,750");
    res = res.replace( target: "0.0.850.850", replacement: "0,0,850,850");
    res = res.replace( target: "0.0.950.950", replacement: "0,0,950,950");
    return res + "</svg>" ;
}
```

På denne måde genererede vi vores tegninger dynamisk i forhold til kundens indtastninger. Det smarte ved at gøre brug af SVG er at det som sagt tidligere er i XML format, og derfor ikke behøver at være en ekstern fil man hiver ind. Vi kan derimod generere dem dynamisk og skalerbart i selve koden, hvilket vi har formået at gøre ved hjælp af CarportCalculation klassen.
Vi valgte at tegne 4 typer tegninger.

- Fra siden med farver – En ”Visualiseringstegning”
- En arbejdstegning fra siden
- En arbejdstegning oppefra
- En arbejdstegning forfra.

Dette gjorde vi ud fra de papirer vi har fået af Fog som viser deres nuværende arbejdstegninger og hvad Fog finder vigtigt for kunden at vide. Vi valgte at tilføje en tegning fra siden med farver så kunden havde mulighed for at få visualiseret deres carport. Netop fordi at man som kunde kunne have svært ved at forstille sig det endelige resultat ud fra en arbejdstegning.



6 Status på implementation

Da vi er fem i vores gruppe, lagde vi ud med at sætte høje forventninger til os selv og hvad vores system skulle kunne udføre. Det skulle dog vise sig, at flere uventede ting dukkede op gennem vores udvikling af projektet.

Færdigbygget carport

På .jsp-siden customerCarportStandard, ønskede vi at en kunde skulle kunne bestille en færdigbygget carport med fastlagte mål. Kunden skulle kunne være i stand til at se en side med tekst og pris til den valgte carport. Herfra skulle kunden kunne navigere sig videre til en anden side, som skulle give en større og bedre oversigt over vores udvalg, samt en beskrivelse af den pågældende carport.

Da standard carporten ikke var en user-story blev der ikke afsat tid til at implementere denne del.
Vi havde stor fokus på den dynamiske del ved at kunne bygge sin egen carport og at kunne få genereret arbejdstegninger dynamisk til den pågældende forespørgsel.

For at komme i mål med denne opgave skulle der laves input-felter, på .jsp siden, for at en kunde kunne indtaste information om personens navn, adresse, telefon, e-mail og en eventuel kommentar til købet. Herfra skulle dataen fra input-felterne sendes til LogicFacade, som sender kaldet videre til DataMapper klassen, for til sidst at ramme en metode med en INSERT query, som vi allerede havde lavet til vores byg-selv forespørgsler. Den nødvendige information, som skulle sendes videre, sendes via metodeparameterne, som rammer en metode i DataMapper klassen, med fastlagte variabler for den pågældende standardcarport.

Styling

Der kunne have været anvendt mere tid på den visuelle del af vores hjemmeside. F.eks. kunne der bruges mere tid på vores styling af alle .jsp siderne, samt et par justeringer til det overordnede design af hjemmesiden. Hjemmesiden er dog aldeles brugervenlig og næsten 100% responsiv. Dette er opnået ved hjælp af frameworket Bootstrap.
Vi kunne også have tænkt mere over brugervenligheden på vores landing page(adminpage.jsp), når en bruger/administrator logger ind i systemet. Her menes der f.eks., at man til oversigtstabellen kunne have tilføjet status på diverse forespørgsler. Det vil sige om forespørgslen er godkendt eller afvist, hvilket er en funktion man kan se og redigere på oversigts siden(adminQuoteView.jsp).

Refactoring

Der burde være anvendt mere tid på refactoring af vores kode. Det har bestemt været en læringsproces, i form af ens kodeopsætning og fremgangsmåde. Vi skulle have fået gjort det løbende og ikke tænkt, at vi kunne tage tid til det sidst i processen. Et godt eksempel kunne være vores kalkulationsklasse, hvor der er mange elementer som bliver instansieret i constructoren. Det skulle være delt op i metoder, da det ville gøre koden overskuelig og nemt tilgængelig.

Oprettelse af medarbejdere
Vi har også haft i tankerne, at en administrator skal kunne have mulighed for at oprette en ny medarbejder. Det er der gjort plads til og på adminpage.jsp er koden også gjort klar til brug. Alle funktionerne virker hertil, men vi har valgt at udkommentere koden, da det ikke har været en del af en user story.
Samme stykke kode kunne også være brugt til at oprette kunder til systemet, og blot ændre i ”role” fastlæggelsen, som gemme i databasen, hvis dette skulle blive relevant for Fog i fremtiden.
Det skal dog nævnes, at det på intet tidspunkt har været en user-story, men fra starten har været en del af den anvendte MVC-skabelon.

Tekniske aspekter
Get/Post funktionerne har også været til diskussion i vores gruppe og vi har valgt i vores gruppe at holde os til Post i alle vores <form>. Det kan dog diskuteres om SELECT funktioner ikke skulle være Get, men her mangler vi en større viden om brugen af GET/POST. Det har dog ingen effekt på brugen eller brugen af vores system.

Exceptions kunne vi også have tænkt os at have brugt mere tid på, da vi kun forholder os til LoginSampleException, SQLException og ClassNotFoundException. Vi har flere gange vendt det internt i gruppen, mhb. på at udarbejde vores egne brugerdefinerede exceptions, som ville blive opsat ligesom LoginSampleException. Derved kunne man også lave en brugervenlig besked, ved en fejl, samt gøre bedre brug af log systemet, hvilket leder os til det sidste i denne sektion.

Der var gruppens ønske om at få implementeret brugen af en logger, så vi kunne logge vores fejl i systemet og få segmenteret vores fejl(exceptions). På grund af tidsmangel har det dog ikke været muligt. Undervisningsmateriale omkring dette har været set og er bestemt noget vi vil have i tankerne til næste projekt.

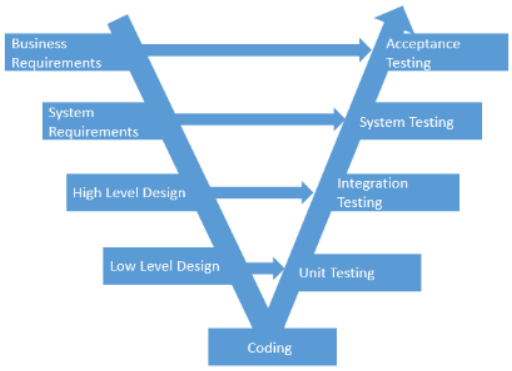
7 Test

Ovenstående tabel beskriver måden hvorpå vi har testet vores system, samt hvilke niveauer vi dækker i forhold til V-modellen som set på figur 20.

I hele vores projektforløb har der ikke været fokus på TDD grundet de kompetencer som vi har erhvervet os på 2. semester, ikke stemmer helt overens med TDD. Metoderne er dog testet så snart det kunne lade sig gøre efter nedenstående testtyper.

Spørgsmål	Teststrategi
Virker vores beregner?	Unit testing: White-box
Passer antallet af stolper?	Unit testing: White-box
Passer antallet af spær?	Unit testing: White-box
Andre beregninger?	Unit testing: White-box
SVG-motor	
Kan jeg tegne et rektangel med en vis størrelse? Virker vores metode?	Unit testing: White-box
Kan jeg tegne en pil som f.eks. er 300px lang?	Unit testing: White-box
Datamapper	
Er der overhovedet hul igennem til databasen?	Integrationstest: Black box
Kan jeg indsætte en ny bruger?	N/A
Kan jeg hente et produkt / liste af produkter?	Integrationstest: Black box
Bliver en ordres status gemt i databasen når man opdaterer den på jsp siden?	Integrationstest: Black box
Hvad med vores brugergrænseflade?	
Kan en kunde finde ud af at bestille en carport? Hvordan afgør vi det?	Testet på potentielle kunder
Kan de ansatte hos Fog finde ud af at bruge systemet?	Testet på ”Ikke-softwarekyndige”.
Hvordan håndterer vi input validering?	jQuery på front-end. Diverse validering på back-end.
Generelt	
Er vores kode skrevet, så vi automatisk kan teste den?	Ingen selenium tests, men god struktur og base for unit- og integrationstests.
Hvordan sikrer vi os en ensartet kodekvalitet?	God dialog og daglig præsentation af gårsdagens arbejde.
Hvordan sikrer vi os at vi ikke tjekker fejlagtig kode ind på vores master- eller developerbranch i GitHub?	Ved altid at bruge *git pull* inden et commit og den daglige dialog.

Vi har altså løbende i projektperioden testet og afprøvet tænkte scenarier, i et begrænset omfang, hvor en kunde eller Fogmedarbejder kunne indtaste nogle forkerte værdier eller få programmet til at crashe.
Derved har vi formået at fejlfinde på mange af de problemer et ikke-testet system kunne forvolde.
Derfor har test af systemets funktionalitet altid været en del af vores arbejdsgang hvilket har sikret os en god struktur og et kompetent fundament for unit- og integrationstest.



Figur 20 - V-model

I projektet er der både brugt white- og black box testning i form af Unit- og integrationstest, testet på "ikke softwarekyndige individer" og jQuery på front-end med diverse validering til back-end. På den måde har vi fået afhjulpet mange af de fejl en bruger eventuelt måtte forårsage allerede på front-end.

Unit & integrations testresultater

Element	Class, %	Method, %	Line, %
Svg	100% (1/1)	20% (3/15)	75% (107/141)
SvgFront	100% (1/1)	20% (3/15)	77% (76/98)
SvgSideways	100% (1/1)	46% (6/13)	91% (110/120)
SvgSidewaysBlueprint	100% (1/1)	41% (5/12)	89% (116/130)

Element	Class, %	Method, %	Line, %
BeamDimensionHeavy	100% (1/1)	60% (3/5)	75% (6/8)
BeamDimensionLight	0% (0/1)	0% (0/5)	0% (0/8)
CarportCalculation	100% (1/1)	34% (26/76)	60% (138/228)
Item	100% (1/1)	4% (1/21)	4% (2/44)
PriceCalculator	100% (1/1)	3% (1/28)	14% (20/141)
StandardDimensions	100% (1/1)	46% (12/26)	47% (24/51)

Element ▲	Class, %	Method, %	Line, %
Connector	100% (1/1)	66% (2/3)	25% (4/16)
DataMapper	100% (1/1)	7% (4/54)	9% (75/798)

I ovenstående tabeller ses dækningsgraden af de tests vi har lavet til projektet. Disse tests er nøje udvalgt grundet deres vigtighed og kompleksitet i opbygningen af projektet.
Hele svg-motoren danner grundlag for alt det visuelle for både kunden og for Fog-medarbejderen og er derfor ret interessant at teste om giver det forventede udfald.

Svg motoren er i vores projekt afhængig af CarportCalculation samt danner CarportCalculation også grundlag for den udregnede pris. Derfor valgte vi også at teste CarportCalculation og PriceCalculator.

Connecter og DataMapper er forbindelsen til vores database, som indeholder alle informationer om materialer og er derfor utrolig vigtig i forhold til PriceCalculator og CarportCalculations. Derfor har vi lavet en række udvalgte integrationstest, på en testdatabase vi har sat op i MySQL WorkBench, som er vigtige for projektet.

8 Process

8.1 Arbejdsprocessen faktuelt

Sprint gennemgang

Under projektet har vi arbejdet med 5 sprints, der hver gik fra mandag til søndag. Vi valgte at uddelegere Scrum Master titlen med en uge til hver, således at vi alle kunne få erfaring med dette.
Udover scrum-masterens normale ansvar blev han også tildelt ansvaret for at føre en log over vores "Daily scrum" møder. De daglige scrum møder tog alt mellem 15 minutter og op til flere timer. Det skal dog tilføjes at de lange møder, oftest var grundet at de endte ud i et samarbejde om en specifik problemløsning.
Under de daglige møder blev statussen på user-stories, tasks, eventuelle rettelser, problemer og tilføjelser gennemgået samt uddelegeret hvis dette var nødvendigt.
Ved hvert fredags scrum møde blev ugens sprint retrospektivt gennemgået, hvor scrum masteren stod for at gennemgå status på ugens user-stories samt tasks med gruppen. Eventuelle rettelser og tilføjelser til weekenden blev derefter tilføjet til ugens sprint.
Dernæst blev den efterfølgende uges sprint planlagt, hvor scrum masteren stod for at oprette user-stories og uddelegere tasks til gruppen.

Vores "Daily Scrum log" samt alle user-stories kan findes i henholdsvis appendix 11.1 og appendix 11.2

Sprint 1
Scrum Master: Mick

Sprint 1 var fokuseret på at få forespørgselsdelen af projektet op at køre.
Da dette var en af de grundlæggende ting i projektet, hvis funktionalitet skulle benyttes til mange af de efterfølgende sprint, var det for os en vigtig ting at få op at køre som det første.

Sprintet indeholdt følgende user-stories:

#21 Kunde: Forespørgsel på spec. carport (basic)
#22 Kunde: Forespørgsel på spec. carport (basic m. tag)
#23 Kunde: Forespørgsel på spec. carport (basic, tag + skur)
#34 Kunde: Forespørgsel på materialetyper

Sprint 2:

Scrum Master: Jean-Poul

Sprint 2 var fokuseret omkring Fogs mulighed for at se en kundes forespørgsel, samt få systemet til at kunne genere en standard tegning af en carport.

Sprintet indeholdt følgende user-stories:

#5 Fog: Se forespørgsler
#15 Fog: Generering af tegning

Under sprint 2 havde vi desuden både et scrum review samt et scrum planning møde med product owner, følgende er taget fra vores "Daily Scrum log", som set i appendix 11.1:

Uddrag fra
"30-04-2020 - Scrum review med P.O
Vi havde lidt misforståelse med product owner, som ikke troede vi havde fået lavet det som var planlagt, men det viste sig, at vi var længere end antaget og kunne fremvise en tegning af vores carport. Gruppen mener vi kan nå i mål med de fleste user-stories, men vil mødes igen til aften for at få et bedre overblik over eventuelle mangler.
"Se forespørgsler" og "slet forespørgsler" er de to user-stories, som mangler mest arbejde og vi kan blive nødsaget til at skubbe dem til næste sprint. Alt afhænger af hvad gruppen når at få kodet i løbet af dagen.

Der skal desuden sættes point på de næste user-stories. Dette vil blive fremlagt til scrum planning mødet med vores P.O i morgen."

Misforståelsen opstod da product owner forventede at se en eksisterende løsning på dette tidspunkt, i form af færdige tegninger. Gruppen forstod, at der var et ønske om at se en fungerende dynamisk løsning som på daværende tidspunkt ikke var klar.

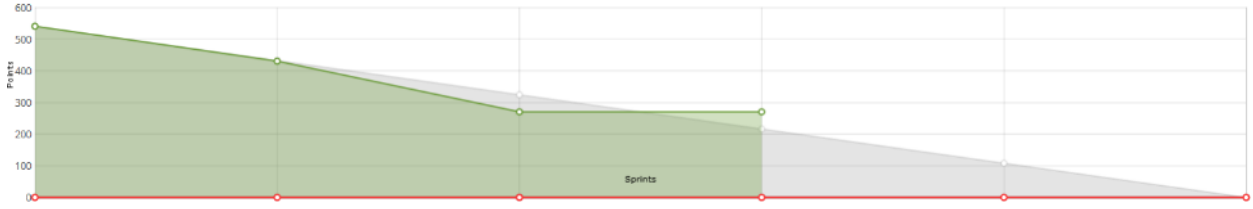
Det viste sig at P.O. blot forventede at se færdige standard SVG-tegninger som gruppen sagtens kunne fremvise.

De næste user-stories blev prioriteret og pointsat som den efterfølgende dag blev godkendt med ros fra P.O. om projektets fremskridt.

Uddrag fra
"30-04-2020 - Scrum planning med P.O
Andet sprint er gennemført og vi nåede i mål med det meste.

Vi havde en User story som lød på "Som Fog-medarbejder vil jeg have at systemet beregner en pris således at jeg kan klargøre et tilbud til kunden." Denne case måtte vi sande, at vi ikke kunne nå i mål med, da den var større end først antaget. Teamet har derfor flyttet den til næste uges sprint.

Vi kan derfor se et dyk i vores performance graf, men vil være stigende når vi får færdiggjort næste uges sprint.



Sprint 3 vil bestå af 210 point, i stedet for de 180 point, da vi som sagt har flyttet en user story over fra sprint 2. Efter gennemgang af user-stories, task og en hel del mere erfaring med estimater, er vi, i teamet, enige om at vi kan nå i mål med Sprint 3. Det skal dog noteres at flere task kan opstå i det vi dykker ned i koden."

Som det fremgår af uddraget, forventede gruppen at det næstkommende sprint (Sprint 3) godt kunne nås med de planlagte point. Det viste sig endnu engang, at prismodulet måtte rykkes frem hvorfor der ses et "Dyk" i grafen for sprint 3. Tilsvarende blev der løst flere opgaver i sprint 4 da gruppen satte ekstra timer ind på projektet for ikke at komme bagud sidst i udviklingsprocessen. Denne indsats gav pote og den efterfølgende udvikling fulgte en stabil kurve.

Sprint 3:
Scrum Master: Alexander
Sprint 3 var fokuseret på et lidt bredere aspect. Et fokus var på at lave SVG tegningerne dynamiske, således at de kunne vise en tegning der passede til den enkelte forespørgsel.
Derudover var der fokus på at få yderligere håndtering af forespørgslerne til at virke, herunder at kunne slette dem, samt give en kvittering når en kunde sender en forespørgsel videre til Fog.
Til sidst var der fokus på, at få en validering af mål og valgmuligheder inkorporeret i drop-down menuerne, således at en kunde ikke kan vælge en carport, hvis mål skaber en carport der ikke kan realiseres.

Sprintet indeholdt følgende user-stories:

#80 Dynamisk view a SVG
#82 Fog: Se forespørgsel
#8 Fog: Slette forespørgsler
#14 Validering af spec. løsninger
#109 Kunde: Kvittering ved forespørgsel

Sprint 4:
Scrum Master: Morten
Sprint 4 var fokuseret på at CRUD. Herunder administrering af varekataloget samt forespørgsler, således at Fog kan redigere og tilføje varer, samt ændre i eksisterende forespørgsler direkte fra hjemmesiden af.

Sprintet indeholdt følgende user-stories:

#11 Fog: Administrering af varekatalog
#7 Fog: spec. carport (Redigering)

Sprint 5:
Scrum Master: Per
Sprint 5 var fokuseret på at få lukket løse ender, fixe diverse fejl og få den sidste funktionalitet op at køre. #171 Slutspurt var meget bred og indeholdt mange vidt forskellige tasks, der kan ses i "user-stories" i appendix 11.2:

Sprint 5 gik også ud på at få de sidste dele af forespørgselsdelen af projektet til at virke, samt opdatere funktionaliteten og designet til administreringen af varekataloget.
Sidste del af sprintet var at få tilbudsprisen samt dækningsgrad til at fungere, således at Fog kan sende et tilbud til kunden.

Sprintet indeholdt følgende user-stories:

#171 Slutspurt
#6 Fog: spec. carporte (Modtaget tilbud)
#148 Fog: CRUD
#59 Fog: beregning af pris

8.2 Arbejdsprocessen reflekteret

Efter at have gennemgået den faktuelle arbejdsproces, giver det anledning til efterfølgende at reflektere over processen. Nedenfor har vi udvalgt nogle emner og citater der giver mening at reflektere over.

Uddelegering af arbejde
Nedenfor er et uddrag af vores Daily scrum log fra appendix 11.1:

Uddrag fra
" 05-05-2020 - Daily scrum
Herudover blev der i gruppen evalueret på effektivitet og uddeling af arbejdsopgaver. Her tænkes der på opgaven med at gøre tegninger dynamiske godt kunne have været varetaget af en person i stedet for to personer. Gruppen kom til enighed om at have særlig fokus på uddeling af Task's fremover, så der ikke behøver at være flere om en Task, medmindre den enkelte Task kræver det."

To af gruppemedlemmerne havde siddet med hver deres SVG-klasse, hvor man nok godt kunne have sat en enkelt person på tasken, i stedet for at to personer hver især satte sig ind i materialet.
Da den ene klasse var færdig, kunne man i stedet have taget og ændret den til at passe til de andre tegninger, i stedet for at have to personer til individuelt at skulle opfinde den dybe tallerken.

Nedbrydning af user-stories
Flere gange i løbet af projektet har vi haft lidt udfordringer med at nedbryde user-stories til tasks. Vi havde en tendens til at vores tasks blev for store, i stedet for korte og præcise. Dette havde vi oppe og vende flere gange under vores Daily scrum møder.
Nedenfor er et par uddrag af vores Daily Scrum log fra appendix 11.1:

Uddrag fra
" 20-04-2020 Daily scrum
Generelt mener vi, at det er vanskeligt at planlægge tasks og især klassediagram da nogle opgaver pt. er uoverskuelige. Vi forsøger os frem og evaluerer løbende."

Uddrag fra
"22-04-2020 - Daily Scrum
Vi ser en tendens til, at vores tasks ikke er "Små nok" og prøver derfor, fremover, at præcisere dem en del mere.
Tasks skal være korte og præcise så vi pludselig ikke har "Uskrevne" opgaver i mellem linjerne."

Som skrevet ovenfor så er det rigtig vigtigt at tasks ikke gaber over for meget, og er så korte og præcise som muligt. Dette giver en et bedre workflow, og man sidder ikke og finder flere og flere "To do", således at tasken trækker ud.

Estimering
Estimering af user-stories var ikke noget vi altid ramte plet med.
Det er ikke nemt at skulle estimere en arbejdsopgave på forhånd om opgaver man ikke har prøvet før.
Det kan samtidig være svært på forhånd at vide hvilke komplikationer det kan bringe og hvilke udfordringer der ligger i opgaven. Nedenfor har vi et par eksempler på komplikationer vi løb ind i under projektet.
Følgende er uddrag fra "Daily Scrum log" appendix 11.1.

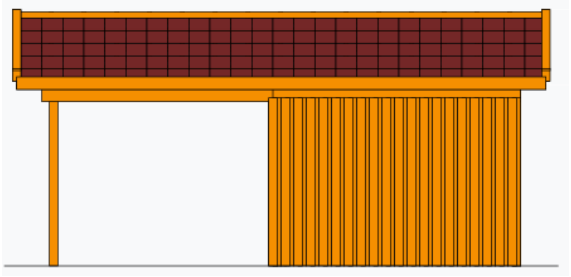
Uddrag fra:
"17-04-2020 Daily Scrum
Gennemgik gårsdagens tasks. Ikke alle blev færdige da de viste sig at være mere komplekse end først antaget. Især dimensionering af carport m. tilbehør krævede mere tid. "

Uddrag fra:
"30-04-2020 - Scrum planning med P.O
Vi havde en User story som lød på "Som Fog-medarbejder vil jeg have at systemet beregner en pris således at jeg kan klargøre et tilbud til kunden." Denne case måtte vi sande, at vi ikke kunne nå i mål med, da den var større end først antaget. Teamet har derfor flyttet den til næste uges sprint. "

Vores daglige scrum møder, samt vores fredags møder har derfor været en rigtig god måde at føre projektet på, da det har givet os en løbende status, som vi dertil har kunne indrette og ændre efter behov.

Produkt owner møde
Uddrag fra:
"07-05-2020 - Daily scrum
Product owner ville gerne have at man kunne se teglsten på tegningen fra siden så det bliver tilføjet. P.O gav udtryk for at vi er kommet rigtigt langt med vores opgave og har derfor givet os mere frie hænder til at bestemme hvilke user-stories vi gerne vil implementere i næste sprint. Derudover sagde han at hvis vi ville implementere andre funktioner som ikke nødvendigvis stod beskrevet i opgaven, måtte vi gerne det."

Beskeden om tagsten blev taget godt imod af teamet som på relativ kort tid implementerede dette. Forventningen var, at tegne tagstenene mere "Realistisk" ved at sørge for, at hvert enkelte tagsten havde en kurve, men det viste sig at være for stor en udfordring på den givne tid. I stedet forblev tagsten firkantede, hvilket fint illustrerer carporten.



Teamet havde fra starten et ønske om at udvikle en CRUD til Fog og da der nu var frie hænder til dette begyndte de indledende diskussioner om dette system, som senere blev implementeret.

Perspektivering

Uddrag fra:

" 18-05-2020 Daily scrum + opsamling

Herefter var der behov for at "Træde tilbage" og se projektet i et større perspektiv hvorfor vi besluttede at holde et "Helikoptermøde" for at se projektet i "Helikopterhøjde".

Agendaen blev aftalt på forhånd, gemt på taiga og gennemgået med gode resultater.
Vi talte med vores Product Owner (Palle) om at slutte med kodning og starte på dokumentation. Blev dog enig om at få de sidste funktioner til at virke."

WIKI FOG CARPORTE

HELIKOPTERMØDE

Agenda 18-05-2020

Helikopterhøjde

1. Kort gennemgang af gennemførte user-stories: "Er de 100% i mål"
2. Gennemgang af tilbageværende user-stories: "Er de relevante? Mangler vi nogle?"
3. Planlægning af næste user-stories - Planlægning af kommende tasks, både test, programmering og dokumentation

- Forespørgselsoversigt (OK for mig)
- "Godkendt forespørgsel" - hvad skal der ske
- "Se tilbud" - hvad skal der ske
- "Se tegning & stykliste"
- mangler vi noget?

Diverse follow-ups

- adobe xd fuld design og prototype
- Mangler vores egen throw beskeder til kunden
- javadoc mangler rigtig fremgangsmåde
- Mangler comments til skabelon klasser (command, frontcontroller osv)
- Mangler billeder til standardbyg og mål
- Slette andre drawings .jsp og drawings fra presentationlayer
- standard design html (model.jsp)
- Færdig design af hele projektet
- Den rigtige raft-distance laver null-pointers -> Convert to task
- toString på SVG driller for vertikale elementer
- Fladt tag bør resultere i "0" på customerAngle i carportCalculation (For at undgå 'null')
- Diverse styling
- N2H: Server 'logger' til fejlsøgning

9 Konklusion

9.1 Fog Trælast

Det kan konkluderes at det nye program vil blive en mærkbar forbedring for Fog Trælast.

Hvor programmet før var en lokal klient uden en umiddelbar centraliseret databaseplatform imellem de forskellige centre, er programmet nu lagt "I skyen" hvor det kan driftes centralt.
De lokale klienter udgår og samtlige medarbejdere kan trænes til, og forventes at anvende samme system.

I det gamle system blev forespørgsler modtaget på mail, telefon eller mundtligt og medarbejderen måtte manuelt taste oplysningerne i deres system. Fremadrettet er denne proces lagt ud til den potentielle kunde i et brugervenligt system og dermed er denne del af processen fjernet hos Fog hvilket resulterer i en optimeret arbejdsgang.

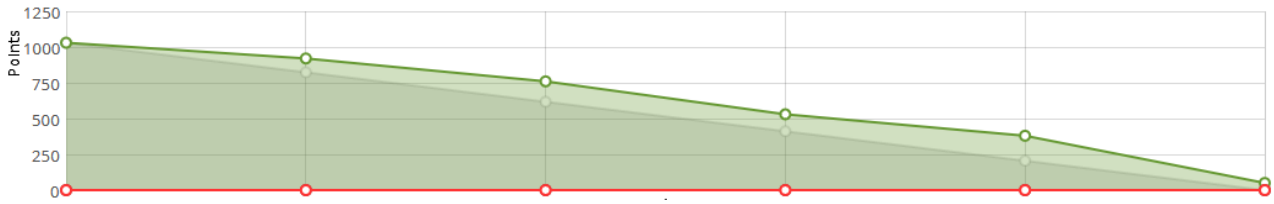
Hvor det gamle system bestod af fejlbehæftede varebeskrivelser, uden mulighed for at opdateres disse er det nye system optimeret til en lokal drift. Sidstnævnte kan lade sig gøre da en evt. opdatering af varer, dimensioner og priser ikke længere kræver systemudviklere, men kan løses af medarbejdere uden stor teknisk indsigt.
I Fogs nuværende løsning er standarddimensionerne foretaget "Med livrem og seler" hvilket resulterer i, at carporten ofte er overdimensioneret som medfører en højere kostpris. Fremover vil det være lettere at styre dimensioneringen og tilrette standardmål hvilket reducerer kostprisen som må formodes at generere flere salg.

De valgte teknologier er moderne og har været på markedet i mange år, hvorfor det formodes at de fortsat vil være relevante fremadrettet. Fog kan således anvende og drifte systemet i adskillige år frem.
Desuden er de valgte systemer skalérbare hvis Fog skulle ønske yderligere udvikling på produktet.
Produktet er veldokumenteret med tekniske diagrammer og leveres med JavaDoc dokumentation som vil lette arbejdet for eventuelt fremtidige udviklere som skal arbejde med programmet.

9.2 SCRUM projektformen

Teamet har arbejdet efter SCRUM principperne for første gang med overvejende gode resultater, men også med nogle faldgruber.
Det skønnes at SCRUM har været nødvendigt for at lave et system af dette omfang på den givne tid da fem udviklere om den samme kode (Som hidtil) ikke havde nået at lave projektet. Derfor har uddelegering vha. user-stories og tasks været en god løsning.

Specielt de daglige møder (Daily scrum), fastsat til 15 min har været uundværlige. Dog har teamet måtte sande, at 15 min ofte blev til 1-3 timer, da vi løbende evaluerede og løste problemstillinger sammen. Allerede efter to sprints vurderede gruppen, at ca. 200 point i hvert sprint var realistisk og har kunne holde dette niveau, som det fremgår af grafen i nedenstående figur.



Figur 21 - Scrum backlog graf

Teamet fastsatte fra start, en række user-stories som over fem uger har været uændrede, dog med ganske få tilføjelser.
Tilføjelserne var nødvendige, da projektet var svært at overskue i opstartsfasen hvorfor få nye user-stories blev tilføjet frem til sprint tre. Desuden var det tilfældet, at der i alle sprints blev tilføjet ekstra tasks løbende, som teamet blev bedre til at nedbryde de enkelte opgaver.
Teamet er løbende blevet bedre til at præcisere user-stories og de tilhørende tasks hvilket er nødvendigt for at sikre et godt projektforløb. Specielt daily meets skal være korte – eventuelle problemløsninger må løses i par, så resten af gruppens ressourcer forbliver fri.

Desuden er det alles indtryk, at det er umådeligt svært, at planlægge et udviklingsforløb over flere uger således at product owners deadlines overholdes 100% - Der er simpelthen for mange ukendte parametre i projektforløbet.
Teamets product owner har løbende haft en god dialog med scrum-master og udviklingen har heldigvis ikke været præget af store kundeændringer som kunne komplicere forløbet.

Teamet kan konkludere, at alle medlemmer har opnået en god føling med SCRUM og fremadrettet vil være i stand til at arbejde, endnu mere effektivt, efter denne projektform.

11 Appendix

11.1 Daily SCRUM log

SPRINT 1:
SCRUM master: Mick
15-04-2020 Opstart
Indledende diskussion og overvejelser i forlængelse af projektoplæg og kundemøde.
Rammer for mødetider, daglig arbejdstid- og form blev diskuteret og blev som følger:

WIKI FOG CARPORTE

Før en kort log over det hele til rapporten

SCRUM planning møde

- Opdatér product backlog
- Prioriteret user stories
- Estimeret user stories i t-shirt størrelse
- (Acceptance kriterier)

SCRUM review

- Opdateret sprint backlog
- Demo af User Stories

SCRUM retrospective

- Hvad gik godt
- Hvad gik ikke godt

Daily SCRUM (15 min)

- Hvem gør hvad
- Er der problemer?

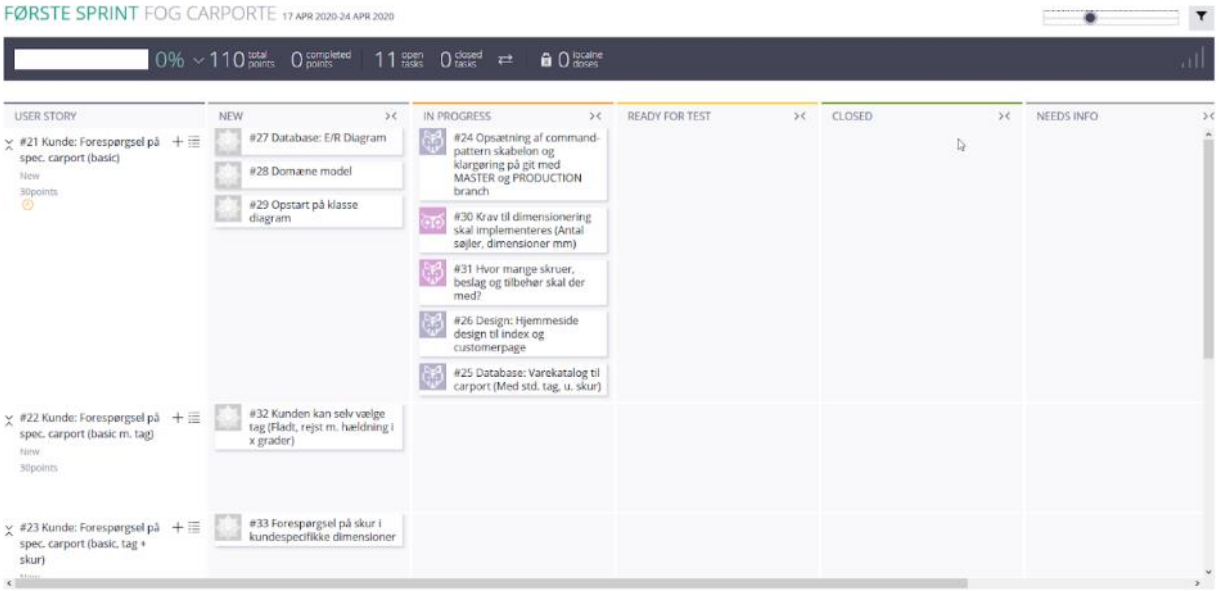
Tech meetings

- Updated Domain Model
- E/R-diagram
- Source code
- Test cases

Har oprettet backlog af ca. 19 user-stories hos taiga.io og prioriteret samt tilføjet deadlines på de indledende opgaver. Nogle user-stories har pt. tilhørende sub-tasks.
Har oprettet github repo.
Har opdateret "wiki" inde på taiga så vi ved hvad der skal forberedes til diverse SCRUM møder.

16-04-2020 SCRUM planning
Første sprint planning møde med product owner, Palle Bech.
En user-story om bestilling af special carporte blev brudt ned til adskillige mindre user-stories. Der vil være fokus på user-stories fra kundeperspektiv og opstarten vil handle om kundens indlende "Kontakt" med sitet.

Planlægningssytemet i taiga.io blev opdateret med nye user-stories samt tilhørende tasks. Disse user-stories blev lagt i et sprint og de underliggende tasks blev efterfølgende fordelt i gruppen.



Vi forventer fremover at afholde daglige scrum teammøder efter nedenstående tidsplan.

Mandag	Tirsdag	Onsdag	Torsdag	Fredag	Lørdag	Søndag
1230	0900	1230	0900	0900	Efter behov	Efter behov

I mellemtiden prioriterer hvert gruppemedlem sin tid, således at tasks bliver løst efter aftale.

17-04-2020 Daily Scrum

Gennemgik gårsdagens tasks. Ikke alle blev færdige da de viste sig at være mere komplekse end først antaget. Især dimensionering af carport m. tilbehør krævede mere tid.
Udkast til database som passer til det igangværende blev gennemgået med øje for, at der vil være ændringer.
E/R diagram og domæne model startes og vi følger op, på næste daily meet (Mandag).
I mellemtiden arbejdes der hver især hjemme, i weekenden.

20-04-2020 Daily scrum
Gennemgik opgaverne fra fredags.
Design er klar til at blive programmeret.
Tilbehør (Skruer, beslag mm) er dokumenteret.
Første version af domænemodel er udarbejdet og konverteres efterfølgende til .uml.
Første version af databasen inkl. E/R er lavet og hægtet på command-pattern skabelonen. Der arbejdes videre med smårettelser og tilføjelser.
6 siders dokumentation om beregning af spær, stolper, rem mm. er lavet inkl. eksempler. Nogle formodninger har været nødvendige - Product owner er informeret om dette. Næste skridt er, at konvertere beregningerne til metoder i java.

Generelt mener vi, at det er vanskeligt at planlægge tasks og især klassediagram da nogle opgaver pt. er uoverskuelige. Vi forsøger os frem og evaluerer løbende.

Vi aftalte desuden allerede nu, at tænke tests med i alle relevante metoder undervejs.

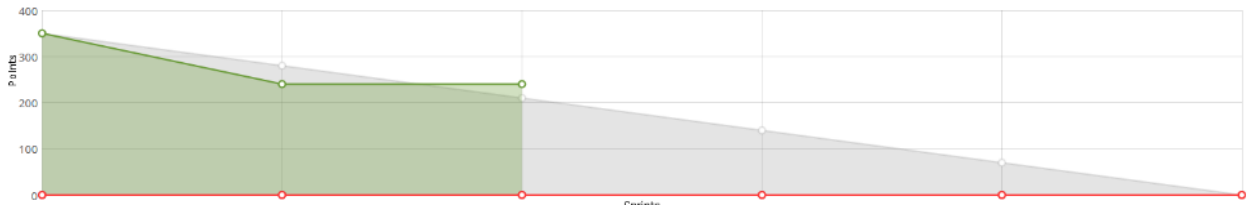
Wiki til projektet på taiga.io er opdateret og indeholder nu mødeplan samt mødestruktur for de enkelte scrum møder.

21-04-2020 - Daily Scrum
Gennemgik præliminært design og beregningsklasse. Ingen er 100% klar endnu og vi talte derfor om udformning og forventninger til disse.
Vi fortsætter, hver især, med tasks og forventer endnu flere opgaver efter 2D-tegningsworkshop i morgen.

22-04-2020 - Daily Scrum
Deltog i 2D tegningsworkshop (.svg) og gennemgik efterfølgende tasks.
Vi ser en tendens til, at vores tasks ikke er "Små nok" og prøver derfor, fremover, at præcisere dem en del mere.
Tasks skal være korte og præcise så vi pludselig ikke har "Uskrevne" opgaver i mellem linjerne.
Vi talte desuden om hvordan vi vil håndtere flow fra forespørgsel til program, til database og tilbage. Vi har aftalt en struktur som vi forventer kan løse dette og arbejder videre med denne.

23-04-2020 - Scrum review med P.O.
Sprint user-stories blev gennemgået og PO så vores design med ros tilovers.
User-stories er næsten opfyldt, men vi har lavet en del mere end hvad user story ligger op til.
Vi skal blive bedre til at holde os til den præcise user story, og bryde dem endnu mere ned hvis det er nødvendigt.
Vi forventer at blive færdige med de planlagte user-stories til imorgen, således at første sprint er 100% gennemført efter planen og lidt til.

24-04-2020 - Scrum planning



Første sprint blev gennemført efter planen. PO oplyser, at i takt med at vi får erfaring med SCRUM forventes det at de efterfølgende sprints kan indeholde flere point.
Vores oplæg til næste sprint accepteres af PO. Vores første sprint bestod af 110 point. Sprint nummer to bliver 39% større med 180 point og vi aftaler i teamet, at vi gør en ekstra indsats for at nedbryde de enkelte user-stories til tasks.

SPRINT 2
SCRUM MASTER: Jean-Poul
27-04-2020 - Daily scrum
Dagens møde startede med at få et hurtigt overblik over eventuelle tilføjelser fra weekendens arbejde. Derefter fik vi valgt en ny SCRUM master. Vi har valgt at inddele det sådan, at hvert medlem bliver SCRUM master for hvert sprint vi har. På denne måde får alle prøvet kræfter med det. Dernæst fik vi set på ugens sprint. Vi har fået oprettet nye task samt fået uddelegeret diverse opgaver til alle gruppens medlemmer. Da vi nu for alvor går i gang med backend programmering har vi også i sinde, har vi aftalt at gøre os tanker omkring test af vores metoder, da det er noget vi ser nærmere på til sprint 3.

28-04-2020 - Daily scrum
Gårsdagens opgaver blev vendt. Tilføjelser til design af administrators forespørgsels side er blevet diskuteret og vil blive færdiggjort i dag. Varekatalog og priser er blevet lavet og er klar til at blive overført til databasen. SVG tegning, set ovenfra, mangler små rettelser hvad angår elementer og deres placeringer. Udkast til SVG tegning, set fra siden, er godt igang. Der vil blive lavet tilhørende java klasser, så vi holder os til separation of concern.

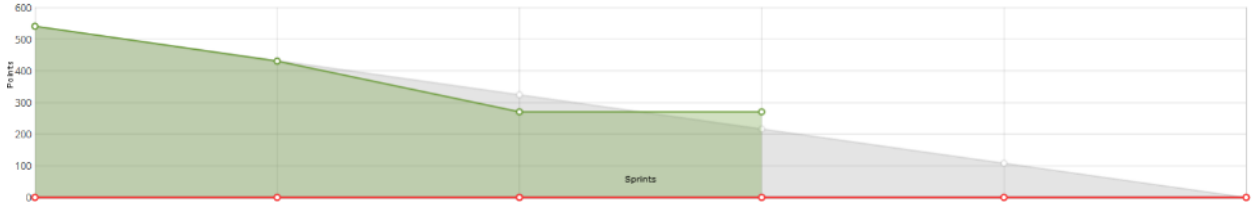
29-04-2020 - Daily scrum
Flere i teamet havde spørgsmål vedrørende vores database kald og hvordan vi skulle foretage kald igennem vores layers. Derudover har vi også stillet spørgsmålstegn ved om vi skal foretage os test af kode i denne uge. Det vil formentlig blive påbegyndt i næste uge, da vi har fået lidt undervisning i det, her til morgen. Det blev påpeget, at vi kun skal forholde os til unittest og positiv-negativ test. Yderligere test bliver først inddraget i projektforløbet til næste semester. Bl.a. fordi vi ikke har nok erfaring til at arbejde med TDD.
Vi havde vores første **technical review** med Arne. Her blev der spurgt ind til om vi skulle opdele vores Mapper klasse i flere dele, da vi indså denne klasse kunne blive meget stor. Det samme galt LogicFacade klassen. Arne mente dog ikke det ville være relevant for os i denne omgang. Til sidst spurgte vi ind til SVG og negative koordinater, hvor vi har haft problemer med at ramme de rigtige mål. Vi har endda været i stand til at bruge negative værdier, hvilket ikke skulle være muligt i et cartesian koordinatsystem. Arne kunne ikke give et entydigt svar, men vi skulle forholde os til at starte fra (0, 0).

30-04-2020 - Scrum review med P.O
Vi havde lidt misforståelse med product owner, som ikke troede vi havde fået lavet det som var planlagt, men det viste sig, at vi var længere end antaget og kunne fremvise en tegning af vores carport. Gruppen mener vi kan nå i mål med de fleste user-stories, men vil mødes igen til aften for at få et bedre overblik over eventuelle mangler.
Se forespørgsler og slet forespørgsler er de to user-stories, som mangler mest arbejde og vi kan blive nødsaget til at skubbe det til næste sprint. Alt afhænger af hvad gruppen når at få kodet i løbet af dagen.

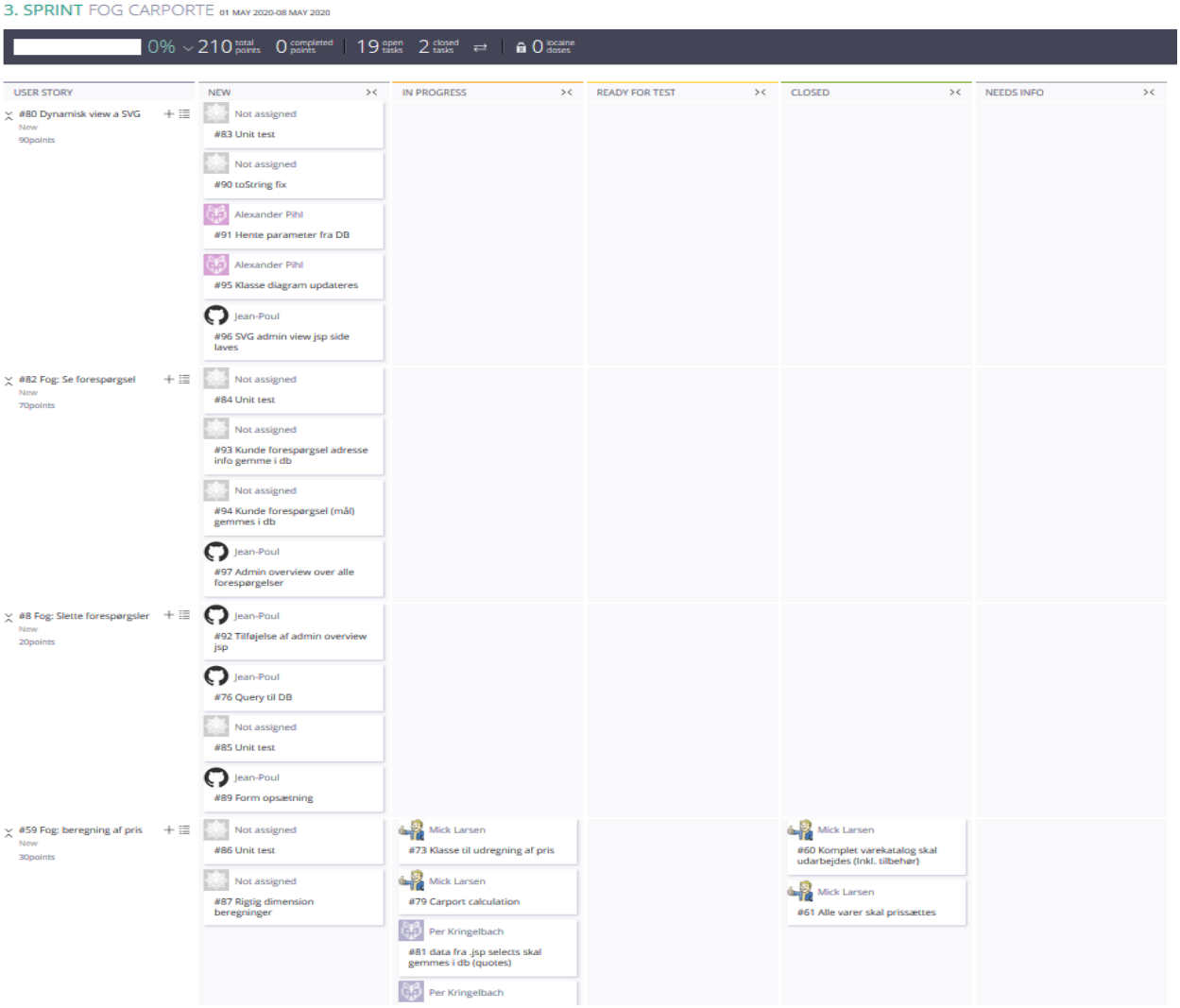
Der skal desuden sættes point på de næste user-stories. Dette vil blive fremlagt til scrum planning mødet med vores P.O i morgen.

30-04-2020 - Scrum planning med P.O
Andet sprint er gennemført og vi nåede i mål med det meste.
Vi havde en User story som lød på *"Som Fog-medarbejder vil jeg have at systemet beregner en pris således at jeg kan klargøre et tilbud til kunden."* Denne case måtte vi sande, at vi ikke kunne nå i mål med, da den var større end først antaget. Teamet har derfor flyttet den til næste uges sprint.

Vi kan derfor se et dyk i vores performance graf, men vil være stigende når vi får færdiggjort næste uges sprint.



Sprint 3 vil bestå af 210 point, i stedet for de 180 point, da vi som sagt har flyttet en user story over fra sprint 2. Efter gennemgang af user-stories, task og en hel del mere erfaring med estimater, er vi, i teamet, enige om at vi kan nå i mål med Sprint 3. Det skal dog noteres at flere task kan opstå i det vi dykker ned i koden.
På nuværende tidspunkt ser sprint 3 ud på følgende måde:



SPRINT 3
SCRUM MASTER: Alexander
04-05-2020 - Daily scrum
Ny scrum master blev valgt.
Den daglige status omhandlende ændringer/tilføjelser fra weekenden som blev gennemgået. Herunder en masse validering på selects fra jsp, opdatering af UML klassediagram, klargøring til modtagelse af data til dynamisk generering af svg tegning samt design ændringer, tilføjelser til jsp sider herunder sletning af forespørgsel fra admin side og ændringer i kalkulation af carport.
Dernæst blev der lavet code review, med detaljeret gennemgang af klasser/jsp sider osv. Så alle er med og up to speed, med koden indhold. Derudover blev der aftalt fortsat arbejde til næste dag.

05-05-2020 - Daily scrum
På statusmødet blev der gennemgået:
Forespørgsel implementeret så den kan vises på adminsiden. Kategorier på admin siden er blevet mere samlet. Derudover er tegningerne i fuld gang med at blive dynamiske og er kommet et godt stykke af vejen. Der er startet på en prisliste til forespørgsel herunder drøftet fremgangsmåden mht. Varenumre.

Herudover blev der i gruppen evalueret på effektivitet og uddeling af arbejdsopgaver. Her tænkes der på opgaven med at gøre tegninger dynamiske sagtens kunne have været varetaget af en person i stedet for to personer. Gruppen kom til enighed om at have særlig fokus på uddeling af Task's fremover, så der ikke behøver at være flere om en Task, medmindre den enkelte Task kræver det.

Alle i gruppen fortsætter med de Tasks til i morgen som tilhøre dem.

06-05-2020 - Daily scrum
På statusmødet blev der gennemgået:
Tegning oppefra er mere eller mindre færdig, dog små rettelser, men den genereres dynamisk ud fra forespørgsler nu. Tegning fra siden er også næsten færdig, mangler tekst, men er også dynamisk ud fra forespørgelse.
Forspørgelse kvittering side lavet så kunden får en kvittering når forespørgslen er gennemført.
Hver forespørgsler kan nu åbnes fra adminsiden. Afvis forespørgelse knap også implementeret.

Der har været teknisk review med Arne hvor vi snakkede om følgende:
Gennemgang af relationer i EER diagram i forhold til at slette materialer som tilhøre ordre. Arne snakke om at anomalia. Det kan skabe problemer hvis en forespørgsel bliver hægtet op på materialer som ikke længere findes i databasen. Vi gennemgik sidens implementeret funktioner.

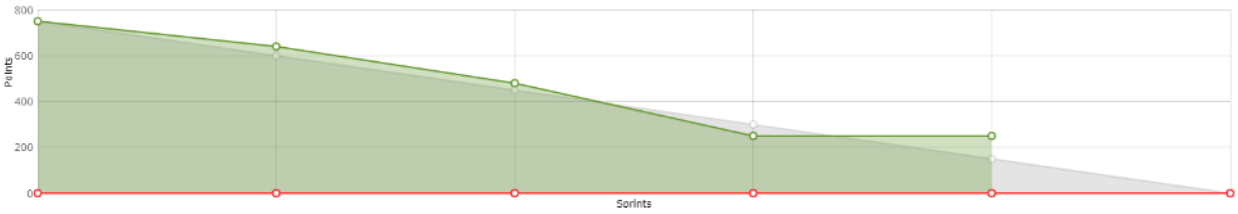
07-05-2020 - Daily scrum
På statusmødet blev der gennemgået:
Planlægning af nyt sprint til næste uge samt snak om nuværende sprint og forberedelse til review med Palle.

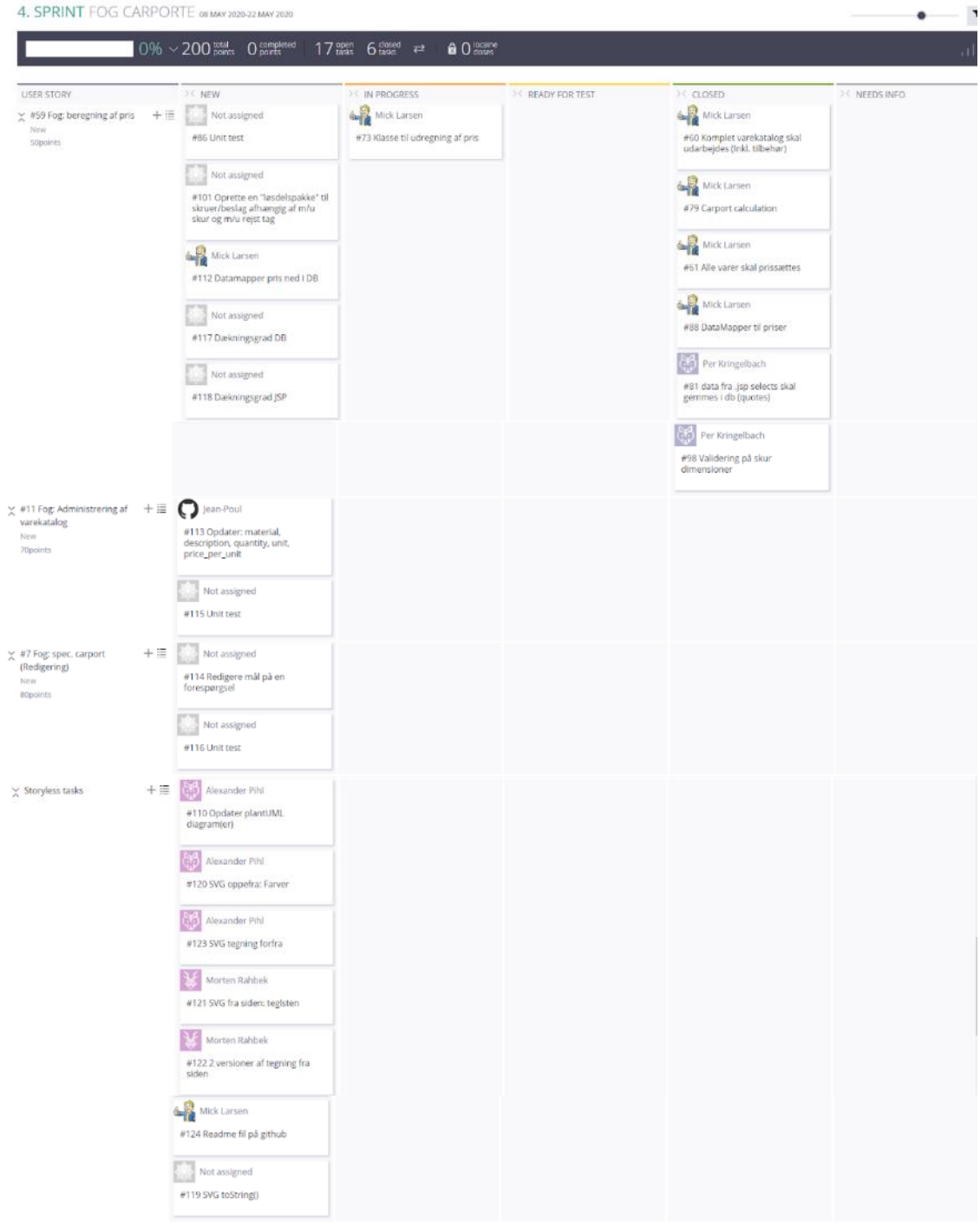
På mødet med Palle blev der snakket om følgende:
Vi viste task frem fra nuværende sprint samt hjemmesiden med de funktioner som er blevet implementeret siden sprint 2.
Palle ville gerne have at man kunne se teglsten på tegningen fra siden så det bliver tilføjet. Palle gav udtryk for at vi er kommet rigtigt langt med vores opgave og har derfor givet os mere frie hænder til at bestemme hvilke user-stories vi gerne vil implementere i næste sprint. Derudover sagde han at hvis vi ville implementere andre funktioner som ikke nødvendigvis stod beskrevet i opgaven, måtte vi gerne det.

08-05-2020 - Daily scrum
3. Sprint blev lukket med 100% og

På statusmødet blev der gennemgået:
Ændringer/tilføjelser fra dagen før og gennemgang heraf. Prisklassen er næsten færdig og spytter en pris ud, men mangler nogle småting.
Ny funktion på admin siden med søgning og show entries af kundeforespørgsler.

Herudover fik vi tilføjet alle task's til de user-stories vi skal igang med i sprint 4:





4. Sprint vil bestå af 200 point fordelt på 3 user-stories med ovenstående task's.

Der blev aftalt arbejde til weekenden og minor fixes sammen. Der var en fejl med mål på tegningen oppefra som blev fikset og pushed til github.

SPRINT 4
SCRUM Master: Morten
11-05-2020 - Daily scrum
På statusmødet blev der gennemgået:

Efter teknisk konsultation med Arne, fik vi valgt en databasestruktur for mere normaliseret tabeller, forslag 1 valgt.
SVG Sideview har fået dynamiske røde teglsten, mangler "bue" i bunden

Vi har diskuteret vores CRUD system (Create, Read, Update, Delete) samt antallet af tegninger på ordre samt udseende. Det blev valgt at vi skulle have 4 tegninger:
Byggetegning set oppefra
Byggetegning fra siden uden teglsten
Byggetegning fra siden med teglsten
Byggetegning forfra

Vi har planlagt at begynde på Unit test torsdag

13-05-2020 - Daily scrum

Der arbejdes videre med svg tegninger samt alexander arbejder med uml diagrammer.
Der kigges på design og opdatering af admin page.
Der arbejdes på at løse problemer vedrørende ordre id.
Vi aftalte samlet at lave unit test torsdag

14-05-2020 Daily scrum

Blueprint tegning tilføjet med mål og uden farver.

JP gør klar til at pushe større ændringer til admin siden.

Gennemgang af projektets mangler
- Kunne være en idé at lave custom exception handling
- Der skal laves bedre kommentarer
- Der skal laves javadoc

15-05-2020 Daily scrum
Vi gennemgik en kort status over weekendens opgaver.

SPRINT 5
SCRUM Master: Per
18-05-2020 Daily scrum + opsamling

Vi gennemgik en kort status over weekendens opgaver.
Herefter var der behov for at "Træde tilbage" og se projektet i et større perspektiv hvorfor vi besluttede at holde et "Helikoptermøde" for at se projektet i "Helikopterhøjde".

Agendaen blev aftalt på forhånd, gemt på taiga og gennemgået med gode resultater.
Vi talte med vores Product Owner (Palle) om at slutte med kodning og starte på dokumentation. Blev dog enig om at få de sidste funktioner til at virke.

WIKI FOG CARPORTE

HELIKOPTERMØDE

Agenda 17-05-2020

Helikopterhøjde

1. Kort gennemgang af gennemførte user-stories: "Er de 100% i mål"
2. Gennemgang af tilbageværende user-stories: "Er de relevante? Mangler vi nogle?"
3. Planlægning af næste user-stories - Planlægning af kommende tasks, både test, programmering og dokumentation

- Forespørgselsoversigt (OK for mig)
- "Godkendt forespørgsel" - hvad skal der ske
- "Se tilbud" - hvad skal der ske
- "Se tegning & stykliste"
- mangler vi noget?

Diverse follow-ups

- adobe xd fuld design og prototype
- Mangler vores egen throw beskeder til kunden
- javadoc mangler rigtig fremgangsmåde
- Mangler comments til skabelon klasser (command, frontcontroller osv)
- Mangler billeder til standardbyg og mål
- Slette andre drawings .jsp og drawings fra presentationlayer
- standard design html (model.jsp)
- Færdig design af hele projektet
- Den rigtige raft-distance laver null-pointers -> Convert to task
- toString på SVG driller for vertikale elementer

19-05-2020 Daily scrum
Alle deltog i undervisning vedrørende rapport-skrivning og hvad der forventes af lærer/sensor at indeholde. Der blev også fortalt hvordan eksamen skulle afvikles. Bagefter arbejde vi sammen om at få nogle af de funktioner som drillede til at virke.

Inden vi stoppede, aftalte vi at lave jUnit-test sammen onsdag.

21-05-2020 Daily scrum

Vi startede med at lave Unit test sammen indtil kl. 12, men havde problemer med at få code coverage til at virke ordentlig. Vi valgte derfor at lave rettelser på div. Kode, og kikke på code coverage senere.

22-05-2020 Daily scrum

Vi startede dagen med kode gennemgang fra de enkelte medlemmer i gruppen, hvorefter vi skrev carport testsuite sammen. Test på svg, pricecalc og integrationstest på database blev uddelegeret til medlemmer i gruppen. Aftale at snakkes ved i weekenden og starte op på rapporten mandag.

11.2 User-stories

Sprint 1

#21 Kunde: Forespørgsel på spec. carport (basic)
Som Kunde
Vil jeg kunne forespørge på en carport i specialmål (L,H,B) med standard tag, u. skur
Således at jeg kan modtage et tilbud på samme.
Acceptkriterier:
Kunden skal kunne angive udvalgte mål til længde, højde, bredde.

Tasks
#30 Krav til dimensionering skal implementeres (Antal søjler, dimensioner mm)
#54 Kundeforspørgsels <select> skal populeres med korrekt data
Vælg tag
1 - valg tagtype: fladt / rejst
2 - hvis fladt = kun ét slags tag
3 - hvis rejst = vælg hældning
4 - hvis rejst = vælg beklædning
#39 Design konverteres til kode
#27 Database: E/R Diagram
#51 Beregning af lægter
#48 Database: Queries til tabelopslag på raft length/spacing/dimension
#24 Opsætning af command-pattern skabelon og klargøring på git med MASTER og PRODUCTION branch
#26 Design: Hjemmeside design til index og customerpage
#37 Database og program skal forbindes (Med standard credentials)
#28 Domæne model (.uml)
#29 Opstart på klasse diagram

#22 Kunde: Forespørgsel på spec. carport (basic m. tag)
Som kunde
Vil jeg kunne forespørge på en carport i specialmål (L,H,B) med enten rejst eller fladt tag
Således at jeg kan modtage et tilbud på samme.
Accept kriterier:
Kunden skal kunne vælge tagtype og hældning

Tasks
#38 Carport beregninger laves om til metoder
#32 Kunden kan selv vælge tag (Fladt, rejst m. hældning i x grader)

#23 Kunde: Forespørgsel på spec. carport (basic, tag + skur)
Som kunde
Vil jeg kunne forespørge på en carport i specialmål (L,H,B) med rejst eller fladt tag inkl. skur i specialmål
Således at jeg kan modtage et tilbud på samme.
Accept kriterier:
Kunden skal kunne angive udvalgte mål til længde, højde, bredde til skur og sende en forespørgsel til fog.

Tasks
#47 Design til spec. byg (Customer)
#33 Forespørgsel på skur i kundespecifikke dimensioner

#34 Kunde: Forespørgsel på materialetyper
Som kunde
Vil jeg kunne vælge blandt materialetyper og farver
således at jeg får præcis den carport jeg vil have
Accept kriterier:
Kunden skal kunne vælge de materialer han ønsker

Tasks
#36 Hvilke byggematerialer tilbyder fog (Tagtyper, Træsorter, metaller, skruer, beslag mm)

Storyless tasks
#45 Design til customer landing page
#43 Layoutside til admin forespørgsel
#44 Fog SVG logo
#42 Opstart på 2D SVG (Learning to walk)

Sprint 2

#5 Fog: Se forespørgsler
Som fog-medarbejder
Vil jeg modtage forespørgsler på en carport
Således, at jeg kan se forespørgslen på hjemmesiden
Accept kriterier
Forespørgsel inkl. beregning (Mål - ikke pris) skal være vist

Tasks
#70 Database: Stumpe liste
#78 Main mapper klasse
Efter vejledningen fra Arne fik vi af vide, at der kun skal bruges en mapper klasse og ikke flere
#55 Forespørgselsside – designoplæg
#77 Forespørgselsside – jsp

#15 Fog: Generering af tegning
Som fog-medarbejder
Vil jeg kunne generere en tegning på kundens forespørgsel
således, at jeg kan visualisere løsningen for kunden
Accept kriterier:
Dynamiske tegninger af kundens forespørgsler skal generes og være vist

Tasks
#66 SVG tegning forfra
#64 SVG oppe fra
#63 SVG mål
#67 SVG tegning fra siden
#69 xd design til SVG landing page
#56 Hente data fra database (dimensioner tabel) til program

Sprint 3

#80 Dynamisk view a SVG
Som Fog-medarbejder skal jeg kunne se en dynamisk tegning af en forespørgsel
Accept kriterier
Dynamiske tegninger af kundens forespørgsler skal generes og være vist

Tasks
#91 Hente parameter fra DB
#95 Klasse diagram opdateres

#82 Fog: Se forespørgsel
Som en fogmedarbejder vil jeg kunne se en kundes forespørgsel således at jeg kan starte på et tilbud
Accept kriterier
Kunders forespørgsler skal kunne vises

Tasks
#102 Validering af select options
#96 SVG admin view jsp side laves
#93 Kunde forespørgsel adresse info gemme i db
#107 Åbne kunde quote dynamisk
#99 Quote overview redesign
#94 Kunde forespørgsel (mål) gemmes i db
#97 Admin overview over alle forespørgelser
#108 request info fra specifik kunde til overview.jsp

#8 Fog: Slette forespørgsler
Som fog-medarbejder
vil jeg kunne slette en forespørgsel således at fejlkøb, fortrydelser mm. ikke er i systemet
Accept kriterier
Forespørgsler skal kunne slettes samt være slettet i databasen

Tasks
#89 Form opsætning
#76 Query til DB

#14 Validering af spec. løsninger
Som fog-medarbejder
vil jeg hjælpes af systemet til at vurdere om kundens løsning er realistisk
således at vi sælger en løsning der kan lade sig gøre
Accept kriterier:
Mål og valgmuligheder skal valideres således at en carport kan realiseres udfra en forespørgsel

Tasks
#106 Validering på bestillingssiden

#109 Kunde: Kvittering ved forespørgsel
Som kunde
Vil jeg modtage en kvittering når jeg sender en forespørgsel afsted til fog
Accept kriterier:
Kvittering med de valgte mål, materialer og valgmuliger skal vises efter afstendt forespørgsel
#111 Vis kvittering

Storyless tasks

#104 Spær rettes til
#103 Opret getters til carportcalc
#105 Databasetabel til ”assumption” dimensioner

Sprint issues

#100 eternit tag på rejst tag skal fjernes
Man skal ikke kunne vælge eternittag på rejst tag?

Sprint 4

#11 Fog: Administrering af varekatalog

Som fog-medarbejder
vil jeg kunne opdatere varer og priser i mit produktudvalg
således, at varekataloget altid er up-to-date
Accept kriterier:
Vareliste og priser skal kunne opdateres

Tasks

#113 Opdater: material, description, quantity, unit, price_per_unit
Som Fog-medarbejder skal jeg kunne opdatere en målinger for at kunne rette i eventuelle fejl
#128Fog: Fremvisning af pris
Som fog medarbejde skal jeg kunne se en samlet pris for at kunne redegøre/redigere i den
#129 SVG view dynamisk af forespørgsel
Som for medarbejder skal jeg kunne se et billede af en forespørgsel

#7 Fog: spec. carport (Redigering)

Som fog-medarbejder
vil jeg kunne redigere i en kundeforespørgsel
således, at kundens rettelser bliver tilføjet
Accept kriterier:
Mål, vinkler og valgmuligheder skal kunne ændres på den individuelle forespørgsel

Tasks

#140 Select option værdier
#137 Fog: Opdatering af stykliste mål
#114 redigere mål på en forespørgsel
#138 Fog: Opdatering af stykliste antal fra orderline id

Storyless tasks

#133 Design clean up
#134 Stykliste fremvisning
#136 Exception clean up i DB
#135 Code: redesign code til at søge på order id og ikke user id
#124 readme fil på github
#110 opdater plantUML diagram(er)
#123 SVG tegning forfra
#119 SVG toString()
#121 SVG fra siden: teglsten
#122 2 versioner af tegning fra siden

Sprint issue:

#125 metoder skal hentes statisk og ikke som objekter

Sprint 5

#171 Slutspurt

Få færdiggjort de sidste ting inden deadline
Accept kriterier:
Skal være lukket inden fredag d.29/5

Tasks

#190 CSS oprydning/style
#186 jUnit skal laves på udvalgte klasser/metoder
#193 N2H: SVG crasher når man går fra rejst til fladt tag
#187 JavaDoc kommentarer på alt
#191 Lorem ipsum skal rettes
#194 ”Se tilbud” funktion
#197 form group button fix
#189 fladt tag og u. Skur skal init. Til 0 for at undgå null-pointer
#184 løsdelspakker skal tilføjes til priskalkulationen
#183 pris skal i database inden den vises på kundeforespørgselssiden
#188 ”Ryd op efter dig selv”- klasser og testmetoder

#6 Fog: spec. carporte (Modtaget tilbud)

Som Fog-medarbejder
Vil jeg kunne godkende et tilbud på spec. carporte
således, at jeg kan fremsende ordrebekræftelse, stykliste og byggevejledning
Accept kriterier:
En kundens forespørgsel skal kunne godkendes, og der skal vises en ordrebekræftelse, stykliste samt byggetegninger.

Tasks

#199 antal og pris hænger ikke sammen
#195 SVG fixes
#198 fravælge redskabsrum
#196 item list crash
#130 godkend quote
#144 dropdown til status på kundeforespørgsel

#148 Fog: CRUD

Som Fog-medarbejder
Vil jeg kunne rette, slette og opdatere priser og enheder i databasen
Således at systemet er langtidsholdbart og fleksibelt
Accept kriterier:
Der skal kunne rettes, slettes og opdateres priser og enheder i databasen fra hjemmesiden af.

#149 Design til ”måleenheder”
#154 klasse & funktionalitet til ”måleenheder”
#158 klasse & funktionalitet til ”standardmål”
#156 klasse & funktionalitet til ”raftermål”
#169 klasse & funktionalitet til ”taghældning”
#155 klasse & funktionalitet til ”vareliste”
#157 klasse & funktionalitet til ”tagbeklædning
#159 Design til ”måleenheder”
#150 design til ”vareliste”
#152 Design til ”tagbeklædning”

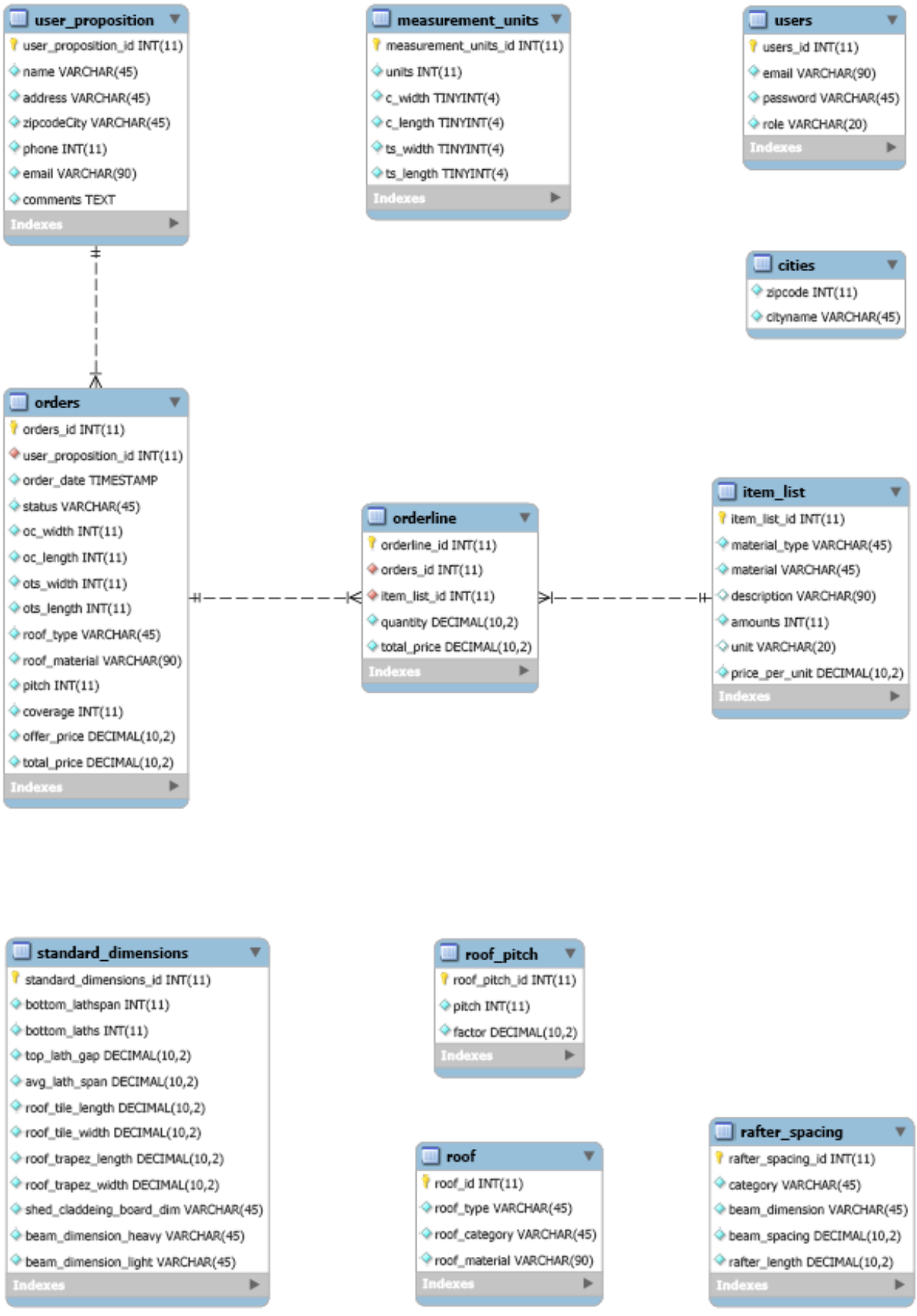
#153 design til ”standardmål”
#166 klasse & funktionalitet til ”rafterafstand
#165 klasse & funktionalitet til ”vareliste”
#151 design til ”rafterafstand”

#59 Fog: beregning af pris
Som Fog-medarbejder
vil jeg have at systemet beregner en pris
således at jeg kan klargøre et tilbud til kunden
Accept kriterier:
Systemet skal kunne beregne en pris på et tilbud til kunden
#60 komplet varekatalog skal udarbejdes(inkl tilbehør)
#86 unit test
#118 dækningsgrad JSP
#112 totalpris ned i DB
#132 tabel til prisforslag
#101 oprette en ”løsdelspakke” til skruer/beslag afhængig af m/u skur og m/u rejst tag
#73 klasse til udregning af pris
#79 carport calculation
#141 priser i DB skal rettes så de er realistiske
#117 dækningsgrad DB
#61 alle varer skal prissættes
#88 datamapper til priser
#81 data fra .jsp selects skal gemmes i db(quotes)
#98 validering på skur dimensioner

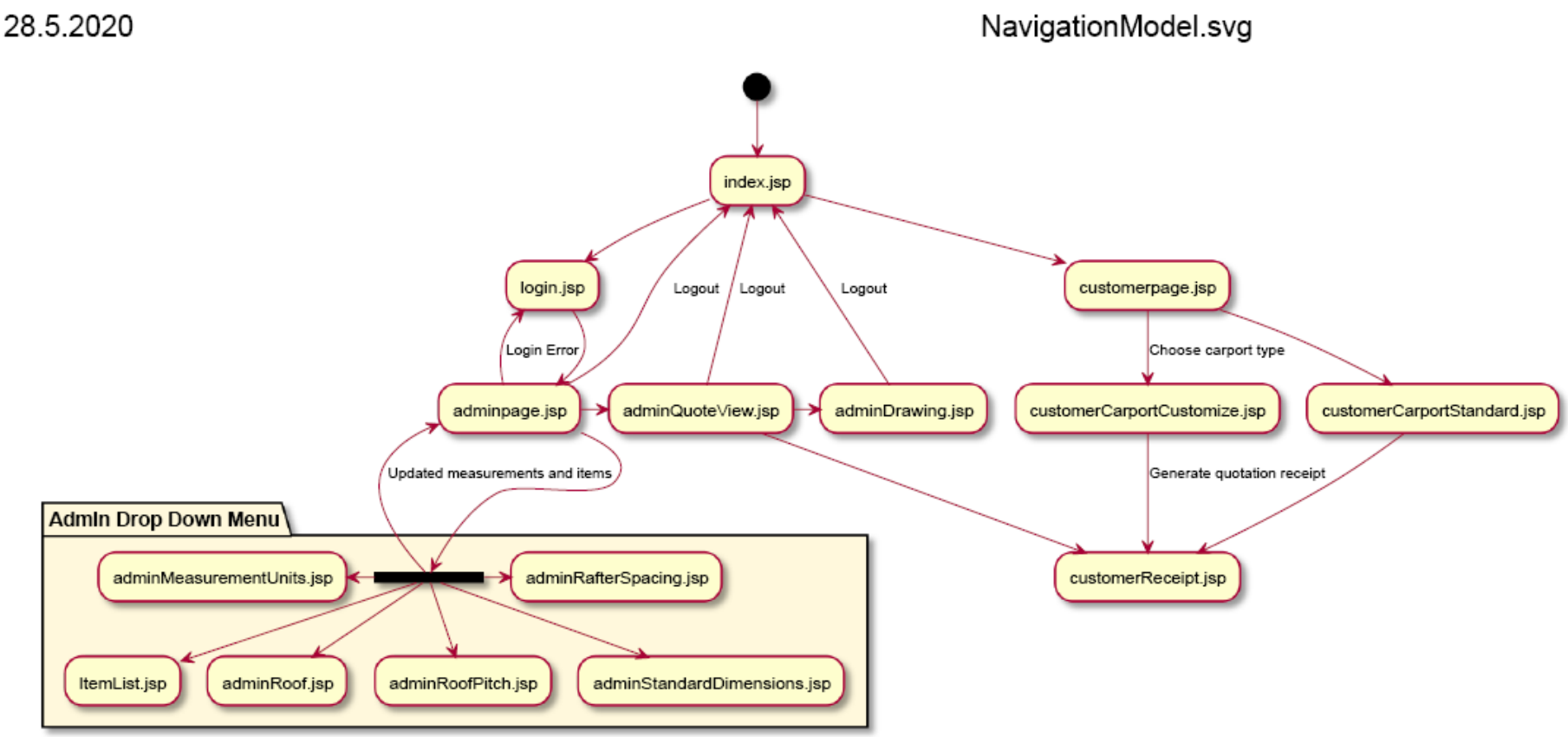
Storyless tasks
#203 rapport skabelon
#205 svg dokumentation af hver pil
#202 billeder til index carousel
#201 mangler underdeling af packages
#204 svg mål skal være ens mellem alexander og morten

Sprint issues
#143 Tag og skur skal sættes til ’0’ i carportcalculation(forhindre null)
#142 raftdistance from db results in “out of bounds” error

11.3 Diagram: EER diagram

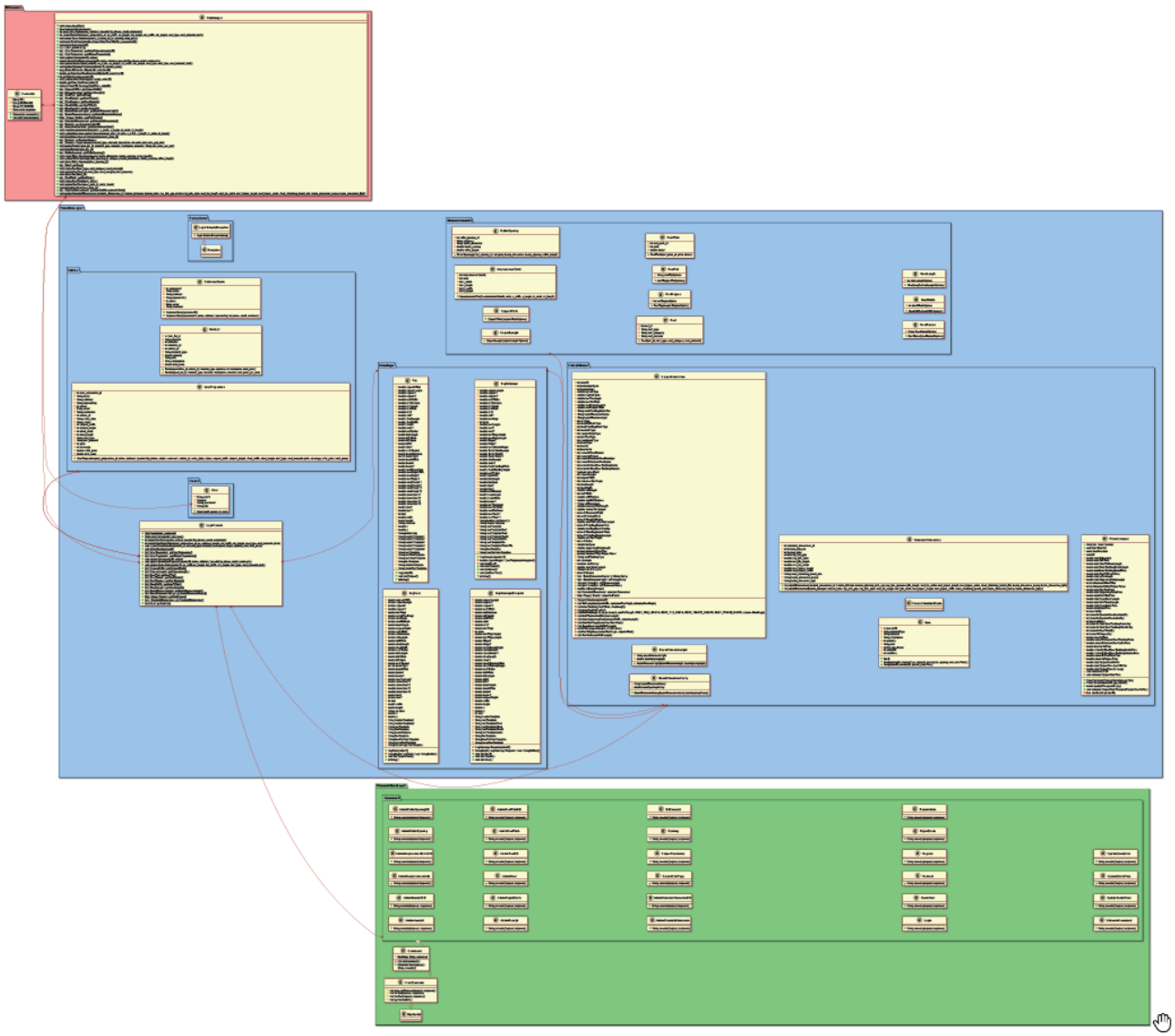


11.4 Diagram: Navigationsdiagram

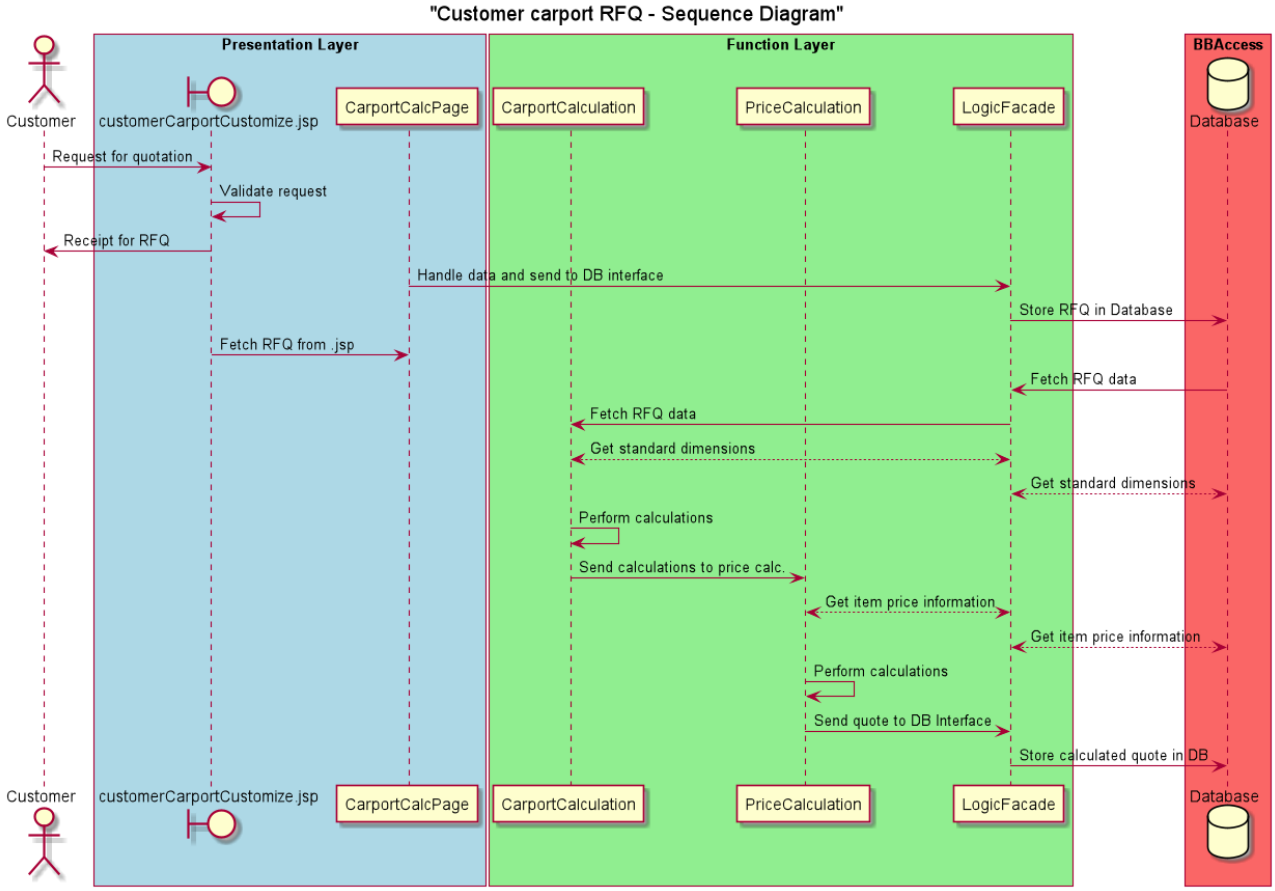


11.5 Diagram: Class diagram

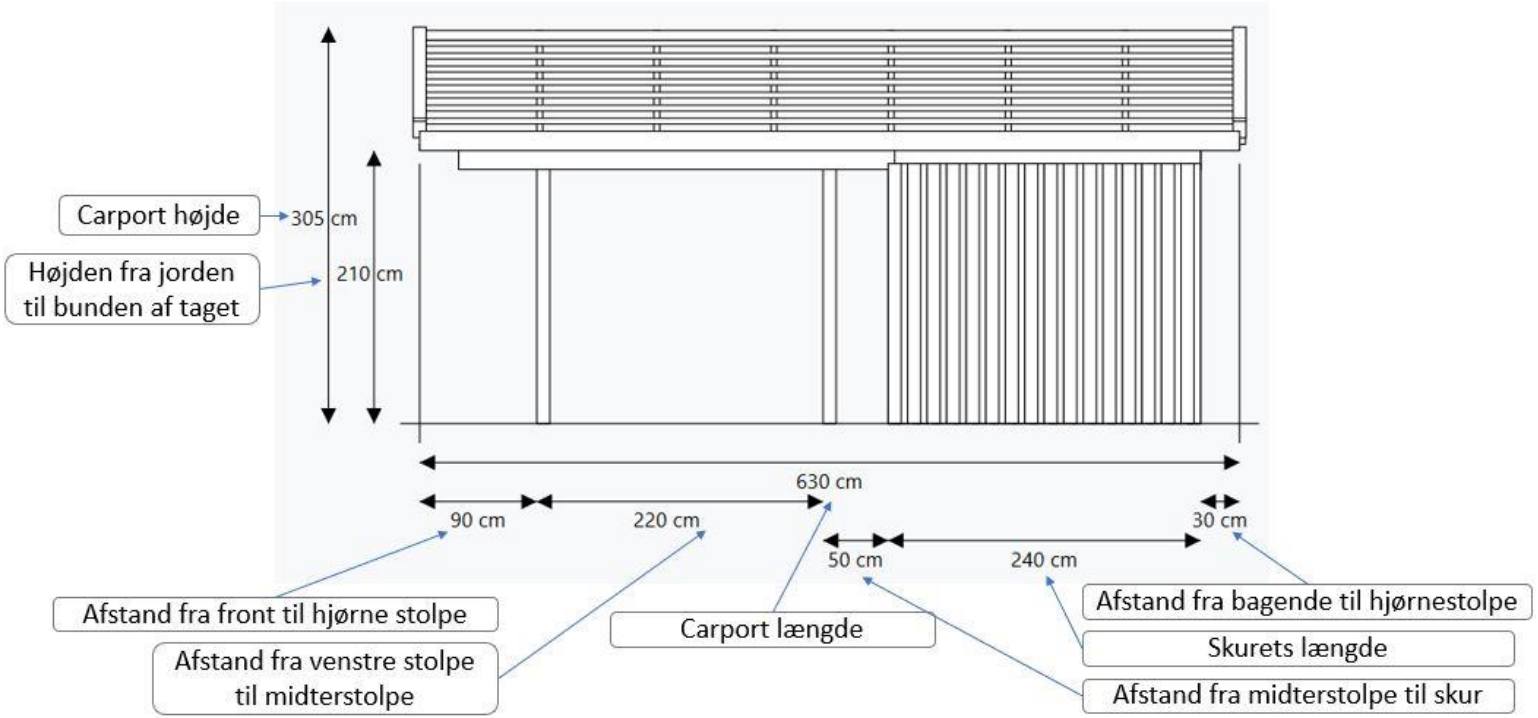
Se github for stor opløsning (Henvisning i afsnit 0).



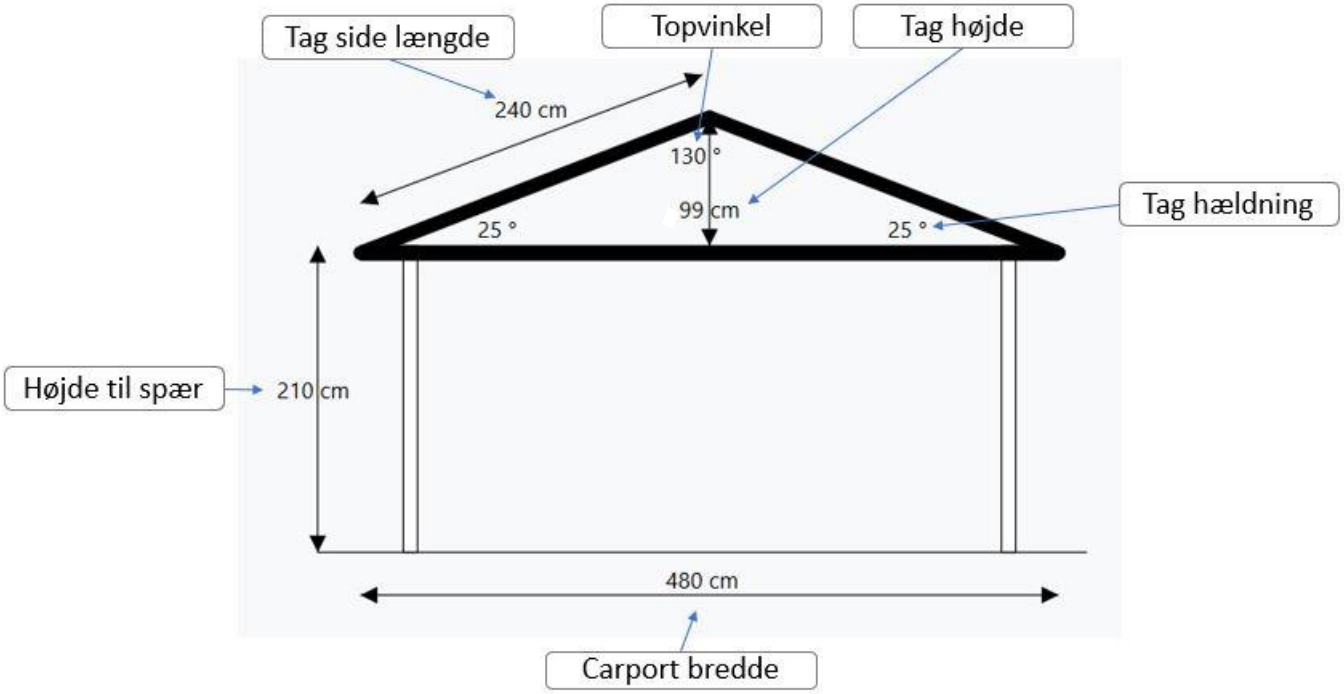
11.6 Diagram: Sekvensdiagram



11.7 Forklaringstegning til Carport



11.7 Forklaringstegninger til Carport



11.7 Forklaringstegninger til Carport