

# FOG Carport

## Datamatikeruddannelsen

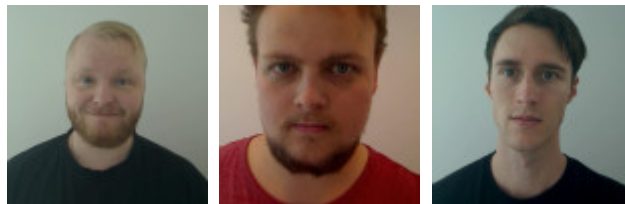
Gruppe 1, hold 20dat2e

### Gruppemedlemmer:

Alex Richardt Gottorp Wagner (GH: A-R-G Wagner)  
(cph-aw116@cphbusiness.dk)

Joakim Stensnæs (GH: JoakimKSS)  
(cph-js437@cphbusiness.dk)

Lasse Emil Støvring Larsen (GH: Lforlasse)  
(cph-ll330@cphbusiness.dk)



Copenhagen Business Academy  
29. maj 2020

## Abstract

This rapport contains a detailed description of how we created the requested product from FOG-Carport and the work-flow in which our group delivered this. It's a system for both customer and employee, wich takes a specific customers carport-request, configures it and sends it to a database. The employee can select a single request and the system will generate a list of part, a visual of the carport and a price-calculation as a salesprice.

## Links

- **GitHub:** <https://github.com/Lforlasse/FOG-Carport>
- **JavaDocs:** [INDSÆT LINK](#)
- **Demo-video:** <https://www.youtube.com/watch?v=Qg-UbBnf4ik>
- **DigitalOcean:** [INDSÆT LINK](#)

# Indhold

<b>1</b>	<b>Indledning, Lasse</b>	<b>5</b>
1.1	Baggrund . . . . .	5
1.2	Teknologivalg . . . . .	6
<b>2</b>	<b>Krav, Joakim/Lasse</b>	<b>7</b>
2.1	Overordnet beskrivelse af virksomheden . . . . .	7
2.2	Projekt krav . . . . .	7
2.2.1	Produktkrav . . . . .	8
2.2.2	Proceskrav . . . . .	8
2.2.3	Dokumentationskrav . . . . .	9
2.3	Essentielle userstories . . . . .	10
2.3.1	US nr. 96: Hjemmeside . . . . .	10
2.3.2	US nr. 36, Bestil carport: Konfigureret . . . . .	11
2.3.3	US nr. 92, Bestil carport: Stykliste . . . . .	13
2.4	Tids-estimering . . . . .	14
<b>3</b>	<b>Diagrammer, Alex</b>	<b>15</b>
3.1	ER diagram . . . . .	15
3.1.1	Sortiment . . . . .	16
3.1.2	Komponenter . . . . .	16
3.1.3	Tagelementer . . . . .	17
3.1.4	Dele . . . . .	17
3.1.5	Lagerstatus og priser . . . . .	17
3.1.6	Konfigurationer . . . . .	18
3.1.7	konfigurationsdel . . . . .	18
3.1.8	konfigurationsdel . . . . .	18
3.1.9	Dimensioner og tillæg . . . . .	19
3.1.10	Validering . . . . .	19
<b>4</b>	<b>Særlige forhold, Joakim/Alex</b>	<b>20</b>
4.1	Grænseflade . . . . .	20
4.2	Navigation med HTTP request . . . . .	22
4.3	Input validering og runtime-exception håndtering . . . . .	23
4.3.1	Validering af brugers input . . . . .	23

4.4	Styklisten	25
4.4.1	Opbygning	25
4.4.2	Opfyldning af lister	25
4.4.3	Data fra database	25
4.4.4	Data til hjemmeside	26
4.4.5	Resultat	26
4.4.6	Kodeeksempler	27
4.4.7	1.6.1 – mapper metode	27
4.5	Tilføj komponent	28
4.6	Færdige klasser	29
4.7	Plantegning	30
4.7.1	Blueprint opbygning	30
4.7.2	SVG elementer	30
4.8	Komponering af SVG string	33
4.9	Sammenkobling	34
4.10	Resultat	34
<b>5</b>	<b>Status på implementation, Joakim/Lasse</b>	<b>37</b>
5.1	Det opbyggede projekt	37
5.2	Projektets fremtid	43
<b>6</b>	<b>Test, Alex</b>	<b>46</b>
6.1	Forklaring	46
6.1.1	Siden c (calcSideC())	47
6.1.2	Siden b (calcRoofHeight())	47
6.1.3	Konvertering mellem datatyper (toInt())	47
<b>7</b>	<b>Arbejdsproces, Lasse</b>	<b>48</b>
7.1	Arbejdsprocesen faktisk	48
7.1.1	Mødestruktur	48
7.2	SPRINT kronologisk	51
7.2.1	Scrumreview- og Sprintmøde, Torsdag d. 23/04 og Fredag d. 24/04	51
7.2.2	Scrumreview- og Sprintmøde, Torsdag d. 30/04 og Fredag d. 1/05	52
7.3	Arbejdsprocessen reflekteret, Lasse	53
7.3.1	SCRUM som værktøj	53
<b>8</b>	<b>Konklusion, alle</b>	<b>55</b>
<b>9</b>	<b>Appendix</b>	<b>57</b>
9.1	Product Backlog	57
9.2	LOG, Lasse	58
9.2.1	Scrum møde 1, Torsdag d. 16/04-20	58
9.2.2	Scrum møde 2, Fredag d. 17/04-20	59
9.2.3	Scrum møde 3, Mandag d. 20/04-20	59

9.2.4	Scrum møde 4, Tirsdag d. 21/04-20	59
9.2.5	Scrum møde 5, Onsdag d. 22/04-20	59
9.2.6	Scrum møde 6, Torsdag d. 23/04-20	60
9.2.7	Scrum møde 7, Fredag d. 24/04-20	60
9.2.8	Scrum møde 8, Lørdag d. 25/04-20	61
9.2.9	Scrum møde 9, Mandag d. 27/04-20	61
9.2.10	Scrum møde 9, Tirsdag d. 28/04-20	61
9.2.11	Scrum møde 10, Onsdag d. 29/04-20	62
9.2.12	Scrum møde 11, Torsdag d. 30/04-20	62
9.2.13	Scrum møde 12, Fredag d. 1/05-20	62
9.2.14	Scrum møde 13, Mandag d. 4/05-20	63
9.2.15	Scrum møde 14, Tirsdag d. 5/05-20	63
9.2.16	Scrum møde 15, Onsdag d. 6/05-20	63
9.2.17	Scrum møde 16, Torsdag d. 7/05-20	64
9.2.18	Scrum møde 17, Fredag d. 8/05-20	64
9.2.19	Scrum møde 18, Tirsdag d. 12/05-20	64
9.2.20	Scrum møde 19, Onsdag d. 13/05-20	64
9.2.21	Scrum møde 20, Torsdag d. 14/05-20	65
9.2.22	Scrum møde 21, Fredag d. 15/05-20	65

# Kapitel 1

## Indledning, Lasse

Denne rapport beskriver gruppens arbejdsproces med introduktion af scrum/sprint-værktøjet og dertilhørende product-backlog samt systemet som FOG-carport har bestilt. Det omhandler hvorledes en kunde kan specificere dimensionerne for en carport de ønsker bygget. Systemet tager imod denne anmodning, sender den til FOG's database og når sælger derefter vælger konfigurationen der fra, sendes den igennem en konfigurator, som skaber et visuelt billede med SVG, en stykliste af komponenter og dele hertil, samt en pris-regulator. Sælger kan herved generere det perfekte tilbud til den enkelte kunde.

### 1.1 Baggrund

FOG Trælaster er en virksomhed der tilbyder og formidler professionelle gør-det-selv løsninger, og som forhandler al materiale, transport m.m. til sådanne konstruktioner.

FOG Trælaster tilbyder kunder at købe carporte som de selv har givet mål på. Denne gør-det-selv løsning kræver et system på deres hjemmeside, hvor kundens dimensioneringer sendes til FOG's database. Kunden skal have mulighed for at vælge dimensioner og materiale ud fra FOG's sortiment, samt mulighed for at vælge hvilken type tag der skal anvendes, og hældningen på dette.

FOG ønsker ligeledes mulighed for at kunne overskue de specifikke anmodninger på carporte, og derved et system til sælgeren som kan redigere i både kundens anmodning hvis der er ændringer, samt holde styr på og redigere anmodningerne generelt.

Til slut vil fog gerne have muligheden for at kunne redigere i databasen fra den ansattes system.

## 1.2 Teknologivalg

- **IDE:** IntelliJ IDEA 2019.3.3
- **Database:** MySQL 8.0.18 (2019-10-14)
- **Mock-ups:** Adobe XD
- **API:** apache-tomcat-8.5.53
- **API:** JDBC
- **Hosting:** DigitalOcean Webserver
- **Server:** Virtual Ubuntu 19.10 x64
- **Graphics:** SVG
- **Rapport:** Latex Overleaf

## Kapitel 2

# Krav, Joakim/Lasse

I dette afsnit bliver der redegjort for hvad Johannes Fog er for en virksomhed, hvad de ønsker og har behov for i arbejdet med dette projekt. Ydermere bliver der sat fokus på nogle af de essentielle dele af produktet og dets udvikling.

### 2.1 Overordnet beskrivelse af virksomheden

Johannes Fog er en virksomhed der har sit fokus i byggecenter-branchen. De har flere trælast-afdelinger spredt rundt i Nordsjælland<sup>1</sup>. Et af deres produkter er carporten. I sammenhæng med dette, ønsker virksomheden en ny og moderne måde at kunne tilbyde deres kunder en skræddersyet carport, og dertil hører nogle krav.

### 2.2 Projekt krav

- Der skal genereres et produkt (produktet)
- Der skal være en dokumenteret arbejdsproces (processen)
- Der skal afleveres en rapport og der skal præsenteres et projekt (dokumentationen).

---

<sup>1</sup>Link til FOG's hjemmeside: <https://www.johannesfog.dk/byggecenter/om-fog/>



### 2.2.1 Produktkrav

#### Arbejdsgange der skal IT-støttes

Hos Johannes Fog er det vigtigt, fortsat at kunne tilbyde kunder muligheden for at konfigurere deres egen carport. Det er en funktionalitet som virksomheden i mange år allerede har kunnet tilbyde, men et gammelt og forældet system forhindrer Johannes Fog i at følge med tiden, og skabe det overskud, firmaet har brug for.

- Kunden skal kunne skræddersy deres egen carport. Med dette menes der at kunden selv kan vælge mål på carporten, hvilke materialer den skal bestå af, og hvilke faciliteter den skal have.
- Der skal være mulighed for at tilvælge tagrejsning og dertil hvilken hældning taget skal have, og hvad materiale taget være lavet i.

Ydermere skal det være muligt at tilvælge vægge, til bestemte sider af carporten, samt at kunne bestemme om der skal være indbygget et skur i carporten eller ej.

Denne funktionalitet er udgangspunktet for værktøjet, der skal give kunden flere muligheder, og gøre det attraktivt at handle hos Johannes Fog.

- Johannes Fog har yderligere et fokus på hvorledes det bagved liggende program, håndterer alt data'en, og hvordan en medarbejder har mulighed for at få indflydelse på udbuddet af materialer, mål og priserne herpå.

Der skal være et overordnet overblik over kundernes anmodninger, som den ansatte kan hente og redigere i. Sælger skal også kunne arbejde på en enkelt konstruktion, og der skal genereres en stykliste, et visuelt billede samt være en prisenregner, som sælger kan arbejde i.

### 2.2.2 Proceskrav

- Projektet skal forløbe efter SCRUM modellen - opdelt i diverse Sprints.
- Vi beslutter os selv internt for hvordan rollen som SCRUM-master varetages.
- De tilknyttede lærere vil agere som product-owner i de foreliggende SPRINT review- og tekniske møder, som afholdes torsdage og fredage.

### 2.2.3 Dokumentationskrav

- Der er krav om dokumentation i forhold til både kodning og rapport. I kodningen bruger vi Javadocs, som er linket under LINKS, øverst i rapporten. Rapporten er specificeret efter en rapportskabelon<sup>2</sup>.
- Scrum og sprint værktøjet skal dokumenteres i f.eks. Taiga.io

---

<sup>2</sup>Link til rapport skabelon: <https://datsoftlyngby.github.io/dat2sem2020Spring/projekt/RapportSkabelonFinal.html>

## 2.3 Essentielle userstories

Det aftalte product-backlog med product-owner bliver gennemgået i dette afsnit. Specifikt struktureringen af individuelle user-stories og estimering af disse.

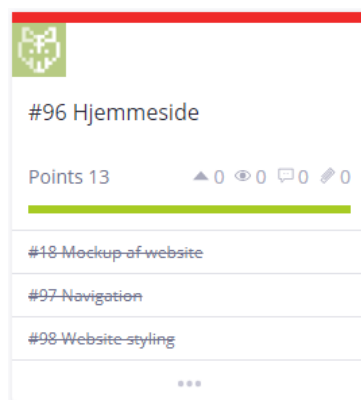
I product-backlog fremgår enkelte userstories der er vigtigere end andre for den essentielle funktionalitet i programmet.

### 2.3.1 US nr. 96: Hjemmeside

Denne userstory beskriver og omhandler mockup af hjemmeside, samt den primære navigation på siden.

Acceptance-criteria for denne user-story lyder: “Som bruger skal jeg kunne tilgå en hjemmeside hvorpå jeg kan tilgå en carport konfigurator, således at jeg kan personliggøre en carport.”

Som den første task i denne user-story blev der lavet mockup af hjemmesiden. Ved at lave mockup, kan man planlægge sin hjemmesides visuelle udtryk og funktionalitet, før man udvikler dette. Dette kan være et godt værktøj til hurtigt at lave en prototype man kan iterere på, uden at skulle bruge ressourcer på at bygge hjemmesiden, før end at man ved hvad der skal laves.

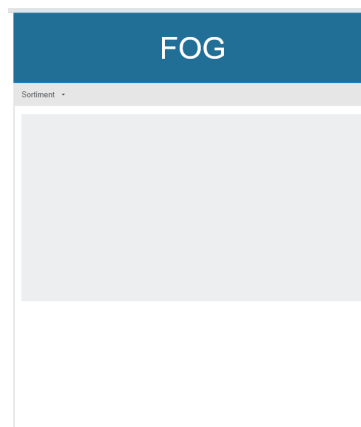


Figur 2.1: Userstory nr 96

#### Mock-up

I denne mockup er der blevet tegnet det første udkast til hvorledes hjemmesiden visuelt skal se ud, og dette leder ind i den næste task i user-storien.

Navigation-tasken tager udgangspunkt i mockup'en til at udvikle en fungerende hjemmeside hvorpå man kan navigere til forskellige sider. Sammen med denne task, blev hjemmesiden også stilet. Dette er repræsenteret i user-storien af en anden task “Website-styling”.



Figur 2.2: Mock-up af hjemmesiden

I illustrationen herunder ses resultatet af denne user-story og følgende iterationer. Bid mærke i forskellen fra det originale mockup til det færdige produkt, efter flere userstories har introduceret mere funktionalitet.



Figur 2.3: Færdige illustration af hjemmesiden

### 2.3.2 US nr. 36, Bestil carport: Konfigureret

Userstory nr. 36 fokuserer på kundens behov for at kunne konfigurere deres egen carport efter specifikke mål og materialer. Acceptance-criteria i denne feature lyder:

“Som kunde skal jeg kunne anmode om at få et tilbud på en carport med personlige mål på højde, længde, bredde samt materialer.”

Udover ovenstående lyder nogle specifikke kriterier for denne user story:

- “Ved kundens anmodning om et tilbud, skal kunden også kunne indtaste kontakt-informationer.”
- “Ved kundens anmodning om et tilbud, skal den specifikke konfiguration, samt kontaktinformation, gemmes i systemet som en nyt anmodning.”



Figur 2.4: Userstory nr 36

Tidsestimering: 21 point (se afsnit ‘tids-estimering’ ref.)

Tasks i denne userstory er opdelt først og fremmest i frontend delen der består af det visuelle udtryk af siden, hvor knapperne sidder, og hvilken tekst der fremgår på siden.

Ydermere udgør frontend hvordan indtastet data fra kunden bliver kontrolleret og sendt til backend ved brug af HTML requests. Det specifikke resultat af denne user-story fremgår af figuren forneden:

The image shows a web application titled "Carport konfigurator" with a "fag" logo in the top left and "Konfigurator" and "Værktøj" buttons in the top right. The interface is divided into two main sections: "Carport konfiguration" and "Kontakt information".

**Carport konfiguration**

- Længde:** A slider control with a unit dropdown set to "cm".
- Bredde:** A slider control with a unit dropdown set to "cm".
- Højde:** A slider control with a unit dropdown set to "cm".
- Materiale:** A dropdown menu.
- Tagrejsning:** A checkbox followed by a slider control with a unit dropdown set to "grader °".
- Materiale:** A dropdown menu.
- Væg beklædning:** Three checkboxes labeled "Højre side", "Venstre side", and "Bagside".

**Kontakt information**

- Navn:** Two input fields labeled "fornavn" and "efternavn".
- Kontakt:** Two input fields labeled "email" and "telefon".
- Adresse:** One input field labeled "postnummer".
- Anmod tilbud:** A dark button located at the bottom right of the contact section.

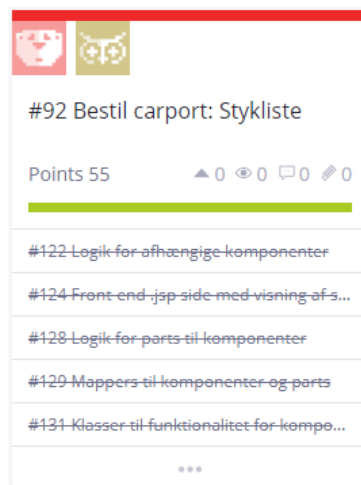
Figur 2.5: carport konfiguratoren

### 2.3.3 US nr. 92, Bestil carport: Stykliste

I denne userstory bliver der sat fokus på en af de vigtigste funktioner for de ansatte der benytter systemet. Styklisten giver en komplet oversigt til den medarbejder, af alle de materialer, længder og små-dele som en given carport består af.

Denne userstory har følgende acceptance-criteria: “Som ansat skal jeg have vist en stykliste således at jeg kan få et overblik over hvilke materialer som kunden skal bruge.” Specifikke krav til denne sætning lyder som følger:

- “Hjemmesiden skal vise en liste over hvilke komponenter og hvor mange som bliver genereret ud fra den konfigurerede carport.”
- “Listen skal genereres automatisk af systemet.”
- “På listen skal indkøbsprisen for komponenter og vejledende salgspris, totalt, fremgå.”



Figur 2.6: Userstory nr 92

Tidsestimering: 55 point.

Sammenlignet med tidligere beskrevet userstories, er denne story estimeret til at være meget mere omfattende. Størstedelen af denne userstory består af udvikling af backend-delen af programmet, som står for at udregne og opdele materialer og længder, alt efter hvad kunden har bestilt ved brug af konfiguratoren. Denne opdeling bliver gjort i produkter som skal være registreret og gemt i FOG's database.

Dermed lyder en af opgaverne i denne userstory på at kunne gemme og hente denne data til og fra databasen. En mindre task omhandler visningen af resultatet fra den bagvedliggende database og logik.

I illustrationen, figur 2.7 (næste side), ses resultatet af denne user-story.

Stykliste					
Vare nr.	Type	Materiale	Længde	Antal	Beskrivelse
2001	Stolpe	Egetræ	315	6	-
2003	Rem	Egetræ	290	4	-
2005	Spær	Egetræ	300	13	Grundspær
2011	Stern, over	Egetræ	96	12	-
2013	Stern, under	Egetræ	96	12	-
2011	Stern, over	Egetræ	150	4	-
2013	Stern, under	Egetræ	150	4	-
2015	Liste	Egetræ	145	4	-
2015	Liste	Egetræ	145	4	-
2015	Liste	Egetræ	290	2	-
2017	Beklædning	Egetræ	225	83	Højre side
2017	Beklædning	Egetræ	225	83	Venstre side
2017	Beklædning	Egetræ	225	43	Bagsiden
Vare nr.	Type	Højde	Bredde	Antal	Beskrivelse
4000	PLASTMO	121	155	10	5cm overlap
Vare nr.	Type	Enhed	Antal	Beskrivelse	
6007	Brædebolt 10 x 120mm	Sæt	6	Samler rem og stolpe	
6012	Firkantskive 40 x 40 x 11mm	Styk	6	Samler rem og stolpe	
6009	Universalbeslag 190mm højre	Styk	13	Samler spær og rem	
6010	Universalbeslag 190mm venstre	Styk	13	Samler spær og rem	
6005	Skruer 4,0 x 50mm 250stk	Pakke	1	Anvendes med Universalbeslag	
6003	Skruer 4,5 x 60mm 200stk	Pakke	2	Samler stern og spær	
6006	PLASTMO bundskruer 200stk	Pakke	3	Skruer til PLASTMO plader	
6002	Skruer 4,5 x 70mm 200stk	Pakke	1	Til ydre brædder	
6004	Skruer 4,5 x 50mm 350stk	Pakke	1	Til inder brædder	
6011	Vinkelbeslag	Styk	20	-	

Figur 2.7: Carport konfiguratoren

## 2.4 Tids-estimering

I afsnit (2.3.1, 2.3.2, 2.3.3) blev tre essentielle userstories gennemgået. Fælles for alle tre er at de er blevet tids-estimeret ud fra en relativ vurdering.

Ved estimeringen af product-backlog'en blev alle userstories estimeret med et given antal point, der følger fibonacci-skalaen. Da det kan være udfordrende at estimere præcist hvor lang tid et stykke arbejde tager, kan det være bedre at vurdere hvor meget arbejde en task tager i forhold til en anden task man har defineret. Når man efterfølgende har gennemført disse tasks, re-evaluerer man ens estimater, hvorefter man ved mere præcist hvor lang tid en given opgave tager.

Med dette udgangspunkt er hele product-backlog blevet estimeret et antal point, som bruges til at prioritere, delegere og planlægge udviklingen af product-backlog, i samarbejde med product-owner.

## Kapitel 3

# Diagrammer, Alex

### 3.1 ER diagram

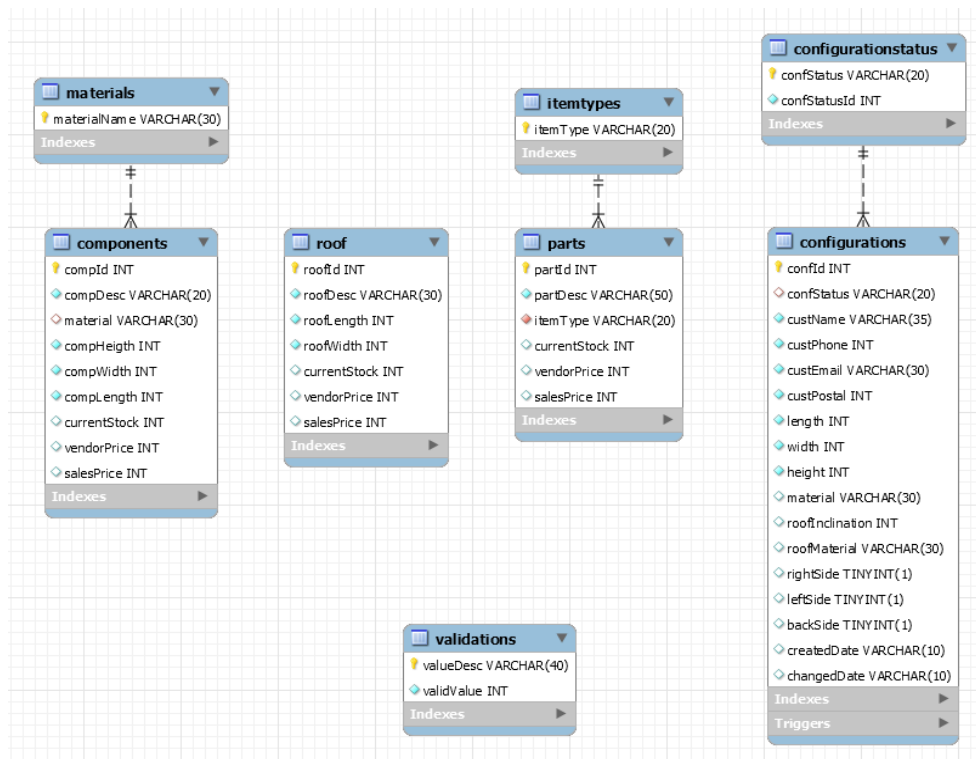


Figure 3.1: ER diagram



## ER diagram forklaret

Vi har lavet en database, hvis funktion er at opbevare FOG's varesortiment og FOG's kunders carport konfigurationer. Databasen kan derved opdeles i to dele, sortiment delen og konfigurationsdelen. Vi vil i dette afsnit forklare intentionen for disse dele og hvad de indeholder. Der eksisterer en ekstra del som vi kalder validering, men den er ikke blevet aktiveret på nuværende tidspunkt.

### 3.1.1 Sortiment

Vores sortiment indeholder tre dele, komponentdelen som består af de forskellige brædder, stolper og andre byggematerialer som bruges til at konstruere en carport, tagdelen som består af forskellige typer tagelementer og til sidst, "deledelen" som indeholder de metalelementer som sætter komponenterne sammen. De tre kolonner (currentStock), (vendorPrice) og (salesPrice) vil blive forklaret i punkt 4.1.5.

### 3.1.2 Komponenter

compId	compDesc	material	compHeight	compWidth	compLength	currentStock	vendorPrice	salesPrice
2000	Stolpe	Trykimprægneret	10	10	400	74	250	475
2001	Stolpe	Egetræ	9	9	400	56	400	750
2002	Rem	Trykimprægneret	5	19	400	45	135	205
2003	Rem	Egetræ	5	18	400	4	145	215

Figur 3.2: Komponent tabel

- **compId:** Komponentens varenummer, dette er unikt og kan bruges til at identificere en specifik komponent.
- **compDesc:** Komponentens beskrivelse, her angives komponentens type i forhold til hvad den bruges til i carporten.
- **compMaterial:** Komponentens materiale bestemmer træsorten på carporten, men hvilke 'komponentsæt' som skal benyttes til de forskellige konfigurationer. Denne kolonne har en relation til (materials) tabellen, så der ikke findes komponenter som har et ukendt materiale.
- **compHeight:** Komponentens højde dimension i centimeter.
- **compWidth:** Komponentens bredde dimension i centimeter.
- **compLength:** Komponentens længde i centimeter. Denne er sat til DEFAULT 400, da det er den ubehandlede længde på komponenten hos FOG(i vores projekt).

### 3.1.3 Tagelementer

roofId	roofDesc	roofLength	roofWidth	currentStock	vendorPrice	salesPrice
4000	PLASTMO	200	120	71	75	130
4001	Betontagsten, sort	42	33	224	25	45

Figur 3.3: Tagelement tabel

- **roofId:** Tagelementets varenummer, dette er unikt og kan bruges til at identificerer et specifikt tagelement.
- **roofDesc:** Tagelementets beskrivelse, her angives tagelementets type.
- **roofLength:** Tagelementets højde dimension i centimeter.
- **roofWidth:** Tagelementets bredde dimension i centimeter.

### 3.1.4 Dele

partId	partDesc	itemType	currentStock	vendorPrice	salesPrice
6001	Skruer 4,5 x 120mm 200stk	Pakke	120	15	45
6007	Bræddebolt 10 x 120mm	Sæt	128	10	19
6013	Bræddeplade 200 x 150 x 5mm	Styk	264	15	28

Figur 3.4: Dele tabel

- **partId:** Delens varenummer, dette er unikt og kan bruges til at identificere en specifik del.
- **partDesc:** Delens beskrivelse, her angives beskrivende navne med hvilken slags del det er, størrelsen og antallet hvis det er en pakke.
- **itemType:** Delens enhedstype, her angives om delen fås i pakker med mange enkeltelementer, er et sæt af mindre dele eller bare eksisterer stykvis. Denne kolonne har en relation til (itemTypes) tabellen, så der ikke tilføjes enhedstyper som ikke findes.

### 3.1.5 Lagerstatus og priser

- **currentStock:** Lagerstatus på de individuelle varenumre.
- **vendorPrice:** Prisen på et varenummer, som FOG betaler for at få varen i kroner.
- **salesPrice:** Prisen på et varenummer, som FOG forventer at sælge for i kroner.

### 3.1.6 Konfigurationer

Når en kunde besøger FOG's hjemmeside, bygger en carport og bestille en anmodning om et tilbud, så gemmes denne konfiguration i databasen. For overskuelighedens skyld opdeles bestillingerne, her i forklaringen, i relevante kategorier.

### 3.1.7 konfigurationsdel

confId	confStatus	createdDate	changedDate
224466	Afsluttet	2020-05-24	2020-05-24
224467	Behandles	2020-05-24	2020-05-24

Figur 3.5: Konfigurationsdel tabel

- **confId:** Konfigurationens identifikations nummer. Dette er unikt og kan bruges til at identificere en specifik bestilling.
- **confStatus:** Bestillingens status viser hvor i processen en bestilling befinder sig, fra den bliver oprettet, til den bliver betalt og afsluttet. Denne kolonne har en relation til (configurationStatus). Dette er for at holde styr på trinene i bestillingsprocessen.
- **createdDate:** Dette er et autogenerated timestamp som viser hvornår kunden oprettede bestillingen.
- **changedDate:** Dette er et autoopdaterende felt som angiver hvornår den seneste ændring til bestillingen har fundet sted.

### 3.1.8 konfigurationsdel

custName	custPhone	custEmail	custPostal
Abbott	44884488	abbott@bot.com	5020
Costello	22662266	costello@ost.com	2550

Figur 3.6: Kundedel tabel

- **custName:** Navnet på kunden.
- **custPhone:** Telefonnummeret på kunden.
- **custEmail:** E-mail adressen på kunden.
- **custPostal:** Postnummeret på kunden. Dette er relevant i forhold til kundens leveringsomkostninger.

### 3.1.9 Dimensioner og tillæg

length	width	height	material	rightSide	leftSide	backSide	roofInclination	roofMaterial
300	580	225	Egetræ	1	1	1	0	PLASTMO
300	450	225	Trykimprægneret	0	1	1	0	Betontagsten, sort

Figur 3.7: Dimensioner og tillæg tabel

- **length:** Længden på carporten i centimeter.
- **width:** Bredden på carporten i centimeter.
- **height:** Højden på carporten uden tag i centimeter.
- **material:** Materiale typen på carporten.
- **rightSide:** Her indikeres om der skal en væg på siden af carporten.
- **leftSide:** Her indikeres om der skal en væg på siden af carporten.
- **backSide:** Her indikeres om der skal en væg på bagsiden af carporten.
- **roogInclination:** Her indikeres om taget på carporten skal have rejsning, målt i grader.
- **roofMaterial:** Tagbelægningstypen.

### 3.1.10 Validering

Vi havde en ambition om at indføre validering på mange af carporten, kundens og materialernes størrelser og forhold, således at når en konfiguration overtrådte disse valideringskriterier, så vil det genererer en advarsel eller notits hos sælger.

valueDesc	validValue
Carspace height	185
Carspace length	300
Carspace width	250

Figur 3.8: Validering tabel

- **valueDesc:** Hvis vi ønsker et udfald på en validering, eksempelvis i forhold til højde, så skrives det her.
- **validValue:** Her indikeres grænseværdien for hvornår valideringen skal træde i kraft.

## Kapitel 4

# Særlige forhold, Joakim/Alex

I dette afsnit bliver der sat fokus på bestemte tiltag og forhold i produktet, som har en udpræget effekt på resten af programmet.

### 4.1 Grænseflade

Produktet tager udgangspunkt i at Johannes Fog i forvejen har en fungerende brugergrænseflade med et login-system til både deres kunder og medarbejdere.

Der er i arbejdet med dette projekt derfor sat fokus på de specifikke krav, se Kapitel 3, som virksomheden har stillet. Dermed er hjemmesiden opdelt i to; ‘Konfigurator’ og ‘Værktøj’. Disse to dele skal anses som uafhængige af hinanden, og kan dermed implementeres i Johannes Fog’s nuværende system.

Konfigurator (kunde platform) Når man tilgår “Konfiguratoren”, får man adgang til de muligheder der er tilgængelige for Johannes Fog’s kunder. Dette gør, man som tidligere beskrevet, uden at skulle logge ind. Dette ses på figur 5.1 (næste side).

Fog
Konfigurator
Værktøj

Carport konfiguration

Længde
cm
Bredde
cm
Højde
cm
Materiale

Tagrejsning
grader °
Materiale

Væg bekledning
Højre side
Venstre side
Bagside

Kontakt information

Navn
fornavn
efternavn
Kontakt
email
telefon
Adresse
postnummer
Anmod tilbud

Figur 4.1: carport konfiguratoren

Værktøj (medarbejder platform) Denne afdeling af produktet giver medarbejderen adgang til de faciliteter der ikke er tilgængelig til den almene kunde. Heriblandt oversigt over anmodninger, værktøj til ændring af priser m.m. Se figur 5.2.

Anmodningsoversigt

Se alle anmodninger
Offer Request ID
Søg

Se nye anmodninger
Se anmodninger i behandling
Se afsluttede anmodninger

Komponent værktøj

Komponent
id
Redigér
Ny komponent

Figur 4.2: Værktøj for ansatte

## 4.2 Navigation med HTTP request

Navigationen mellem de forskellige .JSP sider er bygget på HTTP requests. Grundet intet login system, er der heller ikke identificeret et tilfælde hvori det har været nødvendigt at åbne og benytte en HTTP session.

Et HTTP request bliver benyttet til at overføre data fra en side til den næste. Denne funktionalitet udnyttes, til at lave en dynamisk side, som kan ændre sig afhængig af funktionaliteten.

Komponent værktøj			
Komponent	2000	Redigér	Ny komponent
Komponent	Type	Stolpe	Materiale
			Trykimprægneret
Salg	Pris	475 kr.	Ny pris
Opdater			

Figur 4.3: Komponent værktøj

Med sammenkobling af JSTL (teknologivalg ref.) opstilles en række if-statements som afgør hvilke dele af siden der skal synliggøres for brugeren, som det fremgår af koden på figur 5.3, 5.4 og 5.5.

```
185 <c:if test="${requestScope.component != null}">
186 <!-- JSTL START: IF STATEMENT TO SHOW ONLY IF component-attribute is NOT NULL -->
187 <div class="card-body border-top">
```

Figur 4.4: If test 1

```
174 <c:if test="${requestScope.newComponent != null}">
175 <!-- JSTL START: IF STATEMENT TO SHOW ONLY IF newComponent-attribute is NOT NULL -->
176 <div class="card-body border-top">
```

Figur 4.5: if test 2

En alternativ men også almen tilgang til dette ville være at benytte JavaScript (teknologivalg ref.), til at lave dynamiske elementer på en hjemmeside, uden at skulle indlæse siden på ny.

HTTP request bliver også brugt således at den data der fremgår på siden, altid er opdateret direkte fra det der er registreret i den bagvedliggende database. Dette er et skridt på vejen til at sikre medarbejderens arbejde, ikke er baseret på forældede data og forhold.

## 4.3 Input validering og runtime-exception håndtering

Der redegøres i dette afsnit for hvordan programmet sikre at det input brugeren indtaster er af en korrekt værdi-type, som programmet kan arbejdes med. Endvidere hvordan exceptions er en del af dette arbejde.

### 4.3.1 Validering af brugers input

Ved brug af programmet, både for en medarbejder men også for Johannes Fog's kunder, er det vigtigt at validere de værdier som brugerne indtaster, således at programmet kan arbejde med de rigtige værdier, og give et resultat der har hjemmel i praksis, og som det er udtænkt fra virksomheden.

På konfigurator-siden bliver kunden begrænset til nogle meget specifikke værdier. Dette bliver eksempelvis håndhævet af HTML koden. På figur 5.6 bliver der benyttet en "number"-type til input-elementet, således at man ikke kan indtaste andet end tal.

```
70 <div class="form-row mb-3">
71   <div class="col px-0">
72     <label for="heightRange">Højde</label>
73     <c:set var="heightLimitMin" value="{requestScope.heightLimits[0]}" />
74     <c:set var="heightLimitMax" value="{requestScope.heightLimits[1]}" />
75   </div>
76   <div class="col">
77     <input type="number" class="form-control text-right px-0 h-100 border-secondary"
78           placeholder="cm" name="heightInput" id="heightInput"
79           min="{heightLimitMin}" max="{heightLimitMax}"
80           oninput="heightRange.value=heightInput.value">
81   </div>
82   <div class="col-8 border border-secondary rounded">
83     <input type="range" class="custom-range h-100" value="0"
84           name="heightRange" id="heightRange"
85           min="{heightLimitMin}" max="{heightLimitMax}"
86           oninput="heightInput.value=heightRange.value">
87   </div>
88 </div>
```

Figur 4.6

I Logic-Facade klassen er der ydermere defineret et interval hvori man kan tilpasse sin carports mål. Når man tilgår konfigurator siden, bliver elementerne begrænset af dette interval. Som det fremgår af koden på figur 5.7 er der ikke koblet nogen database på denne data, endnu. Det er klargjort således at man kan oprette en tabel i databasen og hente intervallet derfra.



```

40 @
41
42 public static List<Integer> widthLimits() {
43     //TODO Test data, replace with DBAccess method
44     List<Integer> list = new ArrayList<>();
45     list.add(300);
46     list.add(600);
47     // Test data end
48     return list;
49 }

```

Figur 4.7

Derudover er det nødvendigt for Johannes Fog at få kontaktoplysninger på den kunde der ønsker et at få tilsendt et tilbud. Derfor er der en række felter som har et “required”-tag i elementet i HTML-koden. Dette tvinger kunden til at indtaste en værdi i feltet, før end at vedkommende kan indsende anmodningen, se figur 5.8.

Figur 4.8

Såfremt at brugeren alligevel skulle formå at indsende data af en ugyldig type, eksempelvis et bogstav i et felt hvor der skal stå et tal, vil dette blive fanget i java-koden af en “try-catch”-blok. Se figur 5.9.

```

97
98
99
100
101
102
103
104
105
106

```

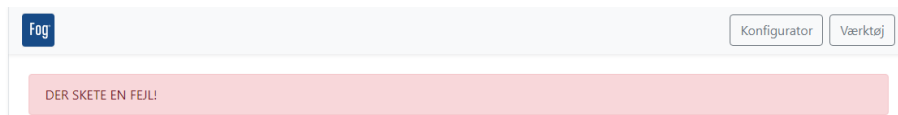
```

try {
    request.setAttribute( s: "offerRequestID", LogicFacade.submitOfferRequest(
        lengthInput, widthInput, heightInput, confMaterial,
        roofInput, roofMaterial,
        custName, custPhone, custEmail, custPostal, right, left, back));
    request.setAttribute( s: "actionSuccess", o: true);
} catch (Exception e) {
    request.setAttribute( s: "error", o: "Der skete en fejl.");
    System.out.println(e);
}

```

Figur 4.9

Hvis dette er tilfældet, har programmet mulighed for at sætte en “error”-attribut i HTTP request, som ved brug af JSTL (teknologivalg ref.), viser en fejlmeddelelse til brugeren. Se figur 5.10.



Figur 4.10

## 4.4 Styklisten

Styklisten i vores projekt er en liste over hvor mange og hvilke materialer som skal bruges til at bygge den carport som en kunde gerne vil bestille. Genereringen af denne liste sker automatisk og processen for det går fra databasen, gennem backend og ud til frontend af vores program. Listen indeholder alt lige fra træ og byggekomponenter, tagelementer og de værktøjsdele som skal bruges for at sætte det hele sammen. Vores stykliste indeholder også den samlede omkostning og salgs pris for sit indhold.

### 4.4.1 Opbygning

Styklisten eksisterer som tre hashmap attributer på vores offerRequest klasse. Disse hashmaps bliver fyldt op med objekter af henholdsvis komponenter(Component), tagelementer(RoofUnit) og værktøjsdele(Part) i deres Map-key og antallet af denne, som integer, bliver sat i Map-value.

### 4.4.2 Opfyldning af lister

OfferRequest klassen indeholder et hav af metoder, hvis formål det er at tage imod diverse mål fra konfiguraionen og så ud fra den individuelle metode, skabe et objekt af den rigtige type, med de rigtige attributer og det rigtige antal. Objektet som skabes indsættes som sagt i den dertilhørende Hashmap liste, som Key-værdi og antallet af objektet som skal bruges, tilføjes som Value-værdi.

### 4.4.3 Data fra database

Når vi opretter objekter til vores stykliste, så bruger vi data fra databasen. Vi beder vores Mapper-metoder på DBAccess laget om at hente denne data og sende os et objekt som vi kan tage fat i og bearbejde, inden vi putter det i vores hashmap liste.

#### 4.4.4 Data til hjemmeside

HashMap attributterne på bestillingsobjektet(OfferRequest) er nu fyldt med objekter(hhv. Component, RoofUnit, Part). Når en ansat åbner en bestilling på hjemmesiden, vil vores front-end kigge listerne igennem og indsætte de specifikke data som styklisten skal indeholde.

#### 4.4.5 Resultat

Denne proces gør det muligt at starte med en database record som ser sådan ud:

confid	confStatus	custName	custPhone	custEmail	custPostal	length	width	height
224470	Ny	Ray, Joey	12312300	VildMedCarporte@nu.dk	2730	390	330	254

material	roofindination	roofMaterial	rightSide	leftSide	backSide	createdDate	changedDate
Trykimprægneret	15	PLASTMO	1	1	1	2020-05-27	2020-05-27

Figur 4.11

Og ende med et billede hos den ansatte som ser sådan ud:

Vare nr.	Type	Materiale	Længde	Antal	Beskrivelse	Vare nr.	Type	Højde	Bredde	Antal	Beskrivelse
2000	Stolpe	Trykimprægneret	344	4	-	4000	PLASTMO	170	135	12	5cm overlap
2002	Rem	Trykimprægneret	330	2	-						
2004	Spær	Trykimprægneret	390	8	Grundspær						
2004	Tagspær	Trykimprægneret	390	8	Singlecut 15" Tagspær	6007	Brædebolt 10 x 120mm	Sæt	4	4	Samler rem og stolpe
2004	Midterspær	Trykimprægneret	33	8	Topcut 15" Midterspær	6012	Firkantskive 40 x 40 x 11mm	Styk	4	4	Samler rem og stolpe
2006	Lægte	Trykimprægneret	330	10	-	6009	Universalbeslag 190mm højre	Styk	8	8	Samler spær og rem
2016	Gavlbrædde	Trykimprægneret	195	12	Singlecut 15", længde	6010	Universalbeslag 190mm venstre	Styk	8	8	Samler spær og rem
2008	Stern, over	Trykimprægneret	165	4	-	6005	Skruer 4,0 x 50mm 250stk	Pakke	1	1	Anvendes med Universalbeslag
2010	Stern, under	Trykimprægneret	165	4	-	6007	Brædebolt 10 x 120mm	Sæt	96	96	Samler spær, tagspær og midterspær
2008	Stern, over	Trykimprægneret	390	2	Singlecut 15"	6013	Brædeplade 200 x 150 x 5mm	Styk	8	8	Samler tagspær og midterspær
2010	Stern, under	Trykimprægneret	390	2	Singlecut 15"	6000	Skruer 5 x 100mm 100stk	Pakke	2	2	Samler lægte og spær
2014	Liste	Trykimprægneret	165	2	-	6002	Skruer 4,5 x 70mm 200stk	Pakke	1	1	Skruer til gavlbrædder
2014	Liste	Trykimprægneret	165	2	-	6003	Skruer 4,5 x 60mm 200stk	Pakke	1	1	Samler stern og spær
2014	Liste	Trykimprægneret	165	2	-	6006	PLASTMO bundskruer 200stk	Pakke	3	3	Skruer til PLASTMO plader
2016	Beklædning	Trykimprægneret	254	47	Højre side	6002	Skruer 4,5 x 70mm 200stk	Pakke	1	1	Til ydre brædder
2016	Beklædning	Trykimprægneret	254	47	Venstre side	6004	Skruer 4,5 x 50mm 350stk	Pakke	1	1	Til inder brædder
2016	Beklædning	Trykimprægneret	254	56	Bagsiden	6011	Vinkelbeslag	Styk	12	-	-

(a) Komponentliste

(b) Tag- og deleliste

Figur 4.12: Den samlede stykliste

#### 4.4.6 Kodeeksempler

Her viser vi et par udvalgte kodestykker som viser hvordan dele af programmet fungerer.

#### 4.4.7 1.6.1 – mapper metode

```
public static Part getPart(String partIdentifier){

    int partId = 0;
    String partDesc = "Ingen del fundet";
    String itemType = "0";
    int vendorPrice = 0;
    int salesPrice = 0;

    try {
        Connection con = Connector.connection();
        String SQL = "SELECT partId, partDesc, itemType, vendorPrice, salesPrice " +
            "FROM parts WHERE partDesc =?";
        PreparedStatement ps = con.prepareStatement(SQL);
        ps.setString( parameterIndex: 1, partIdentifier);

        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            partId=rs.getInt( columnIndex: 1);
            partDesc=rs.getString( columnIndex: 2);
            itemType=rs.getString( columnIndex: 3);
            vendorPrice=rs.getInt( columnIndex: 4);
            salesPrice=rs.getInt( columnIndex: 5);
        }
    } catch (ClassNotFoundException | SQLException ex) {

    }

    Part part = new Part(partId, partDesc, itemType, vendorPrice, salesPrice);
    return part;
}
```

Figur 4.13: Part mapper

I denne kode har vi en metode i PartMapper klassen.

- Metoden tager imod en string fra metodekaldet, som beskriver hvilken væktøjdsdel som vi vil have fat i.
- Koden sender et sql request til databasen, som sender resultset tilbage til metoden.
- Metoden piller resultset'et fra hinanden og indsætter værdierne i de dertil egnede variabler.
- Disse vabler bruges når metoden nederst opretter et objekt af klassen Part.
- Metoden sender objektet tilbage til metodekaldet.

## 4.5 Tilføj komponent

```
private void addSper() {
    boolean hasInclination;
    if (carport.getRoof().inclination == 0) {
        hasInclination = false;
    } else {
        hasInclination = true;
    } //else
    int max = 400;
    int maxSpread = 50;
    int countUnit = 1;
    int addUnit = 1;

    for (int i = carport.getConflength(); i > 0; i -= maxSpread) {
        countUnit += 1;
    }

    addUnit += carport.getConflwidth() / max;

    Component sper = ComponentMapper.getComponent( type: "Sper", carport.getConflmat());
    sper.setCompLength(carport.getConflwidth() / addUnit);
    sper.setCompInfo("Grundsper");
    complist.put(sper, addUnit * countUnit);

    if (hasInclination) {
        Component sperTag = ComponentMapper.getComponent( type: "Sper", carport.getConflmat());
        sperTag.setCompLength(carport.getConflwidth() / addUnit);
        sperTag.setCompDesc("Tagsper");
        sperTag.setCompInfo("Singlecut " + carport.getRoof().inclination + "° Tagsper");
        complist.put(sperTag, addUnit * countUnit);

        Component sperMidt = ComponentMapper.getComponent( type: "Sper", carport.getConflmat());
        int sperCutLength = carport.getRoof().calcSideC(carport.getRoof().inclination, (sper.getCompWidth() * 2));
        sperMidt.setCompLength(carport.getRoof().roofHeight - sperCutLength);
        sperMidt.setCompDesc("Midtersper");
        sperMidt.setCompInfo("Topcut " + carport.getRoof().inclination + "° Midtersper");
        complist.put(sperMidt, countUnit);
    }
} //if
} //addSper
```

Figur 4.14: Metoden tilføjer et spær

- Her har vi en metode som tilføjer en komponent type som objekt til vores komponentliste og et antal.
- Metoden er skræddersyet til at tilføje spær til komponentlisten, så vi er i stand til at indsætte standard værdier som er relevante for netop spær.
- Vi har en 'if condition' som aktiveres hvis carporten har tagrejsning.
- Generelt kan man se hvordan metoden spørger en Mapper metode om at få et komponent objekt.
- Dette objekt bliver så ændret i længden så det passer til konfigurationen og i beskrivelsen.
- Den færdige komponent sættes så ind i mappet, med det samlede antal som Value.

## 4.6 Færdige klasser

Til slut står vi med et OfferRequest objekt som kan sendes videre til LogicFacade klassen(hvis metoder bad om objektet til at begynde med). OfferRequestet har yderligere to objekter i sig, nemlig Carport og Blueprint.

```
public class OfferRequest {  
  
    private Carport carport;  
    private int confId;  
    private LinkedHashMap<Component, Integer> complist;  
    private LinkedHashMap<Part, Integer> partList;  
    private LinkedHashMap<RoofUnit, Integer> roofUnitList;  
    private int vendorPrice;  
    private int salesPrice;  
    private Blueprint blueprint;  
}
```

Figur 4.15: OfferRequest object

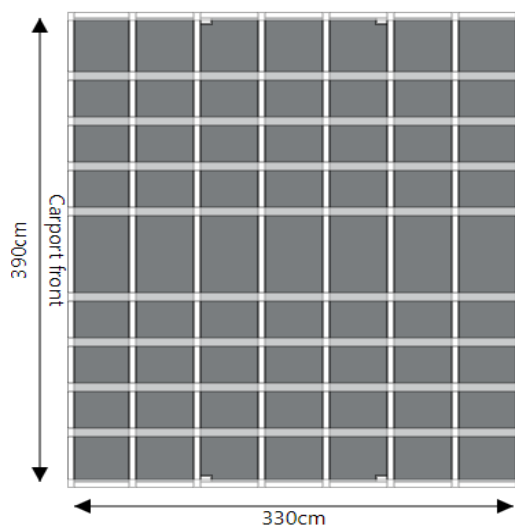
<pre>public class Carport {      private int confId;     private String custName;     private int custPhone;     private String custEmail;     private int custPostal;     private int confLength;     private int confWidth;     private int confHeight;     private String confMat;     private Roof roof;     private boolean carSpace;     private String confStatus;     private final Date CREATED_DATE;     private Date changedDate; }</pre>	<pre>public class Blueprint {      private String canvasFront;     private String canvasBack;     private String canvasText;     private String canvasFill;     private ArrayList&lt;String&gt; stolpe;     private ArrayList&lt;String&gt; rem;     private ArrayList&lt;String&gt; sper;     private ArrayList&lt;String&gt; legte;     private ArrayList&lt;String&gt; stern;     private String markerHead;     private String markerX;     private String markerY;     private String blueprintSVG; }</pre>
--	--

(a) Carport objekt

(b) Blueprint objekt

Figur 4.16: Objekter, offerRequest tager med sig

## 4.7 Plantegning



Figur 4.17: Plantegning

Plantegningen er et SVG billede som automatisk generes på hjemmesiden når sælger åbner en bestilling.

Tegningen indeholder alle komponenter, på nær tagbelægning, midterspær, væg belægninger og lister. En plantegning ser ud som eksemplet ovenfor, men varierer alt efter carportens mål og tagrejsningsvalg.

### 4.7.1 Blueprint opbygning

Blueprint klassen indeholder en række attributter som hver indeholder et lille tekst udsnit af det samlede SVG billede. Se figur 5.16 (b).

Nogle attributter er strings, mens andre er lister af strings (`ArrayList String`). En særlig attribut er `blueprintSVG`, den indeholder den færdige string som resten af attributterne puttes ind i.

### 4.7.2 SVG elementer

Hvert attribut er tom fra starten, men fyldes op af metoder når et `offerRequest` objekt bliver skabt. De skitserelaterede elementer, så som kanvas, pile, tekst og lignende bliver skabt af en enkelt (dog individuel) metode, hvor carportens dele, så som stolper og remme, skal igennem flere metoder som definerer komponentens højde, bredde og placering.

## Simpelt element

Et eksempel på et simpelt element kan være den tekst som indikerer carportens front. Vi betragter koden nedenfor, se figur 5.18.

Først har vi skrevet hele SVG teksten og sat den ind i en setter metode for den tilhørende attribut i Blueprint klassen. Herefter har vi identificeret de dele af SVG koden som skal være dynamisk og udskiftet værdier med variabler. Disse variabler sætter vi så til at tage imod data fra carport objektet eller som statiske data, hvilke skal være overskuelige at rette ved behov.

```
private void assignText(int positionRigth, int positionDown) {  
  
    int x1 = positionRigth - 30;  
    int y1 = positionDown + (carport.getConfWidth() / 2);  
  
    int x2 = positionRigth + (carport.getConfLength() / 2);  
    int y2 = positionDown + carport.getConfWidth() + 30;  
  
    String front = "Carport front";  
  
    int value1 = carport.getConfWidth();  
    int value2 = carport.getConfLength();  
  
    blueprint.setCanvasText("    <text style=\"text-anchor: middle\" transform=\"translate(" + x1 + "," + y1 + ") rotate(-90)\"> " + value1 + "cm </text>\n" +  
        "    <text style=\"text-anchor: middle\" x=\"" + x2 + "\" y=\"" + y2 + "\"> " + value2 + "cm </text>\n" +  
        "    <text style=\"text-anchor: middle\" transform=\"translate(" + (x1 + 17) + "," + y1 + ") rotate(90)\"> " + front + "</text>");  
}  
//assignText
```

Figur 4.18: Koden bag assignText

## Carport element

Carport elementerne fungerer på samme måde som de simple elementer, bortset fra at vi er nød til at definere målene på den figur som repræsenterer komponenterne.

Vi begynder med en lille start-metode som opsamler dimensionerne og bruger dem i den endelige metode.

```
private void assignRem() {  
  
    int height = defineRemHeight();  
    int width = defineRemWidth();  
  
    blueprint.setRem(placeRem(height, width));  
}  
//assignRem
```

Figur 4.19: assignRem metoden



```
private int defineRemWidth() {

    int width = 0;

    for (Map.Entry<Component, Integer> entry : compList.entrySet()) {
        if (entry.getKey().getCompDesc().equalsIgnoreCase( anotherString: "Rem")) {
            width = entry.getKey().getCompLength();
        }
    }
    return width;
}
//defineRemWidth
```

(a) Definerer bredden på remmen

```
private int defineRemHeight() {

    int height = 0;

    for (Map.Entry<Component, Integer> entry : compList.entrySet()) {
        if (entry.getKey().getCompDesc().equalsIgnoreCase( anotherString: "Rem")) {
            height = entry.getKey().getCompHeight();
        }
    }
    return height;
}
//defineRemHeight
```

(b) Definerer højden på remmen

Figur 4.20: Illustrationer af de to define-metoder

Illustration af den endelige metode, som sender en liste med strings tilbage til den lille startmetode, som sender den videre med en setter metode. Hvad er anderledes her i forhold til et simpelt element, er dette består af et antal af det samme element som varierer og skal stå forskellige steder i kanvasset. Se figur 5.21 (næste side).

```

private ArrayList<String> placeRem(int heigth, int width) {
    ArrayList<String> remme = new ArrayList<>();
    int quantity = 0;
    int pushRight = 0;
    int pushDown = carport.getConfWidth() - heigth;

    for (Map.Entry<Component, Integer> entry : compList.entrySet()) {
        if (entry.getKey().getCompDesc().equalsIgnoreCase( anotherString: "rem")) {
            quantity = entry.getValue() / 2;
        }
    }

    remme.add("<rect x=\"" + pushRight + "\" y=\"" + 0 + "\" height=\"" + heigth + "\" width=\"" + width + "\" style=\"stroke:#000000; fill:#ffffff\" />");

    for (int i = 1; i < quantity; i++) {
        pushRight += width;
        remme.add("<rect x=\"" + pushRight + "\" y=\"" + 0 + "\" height=\"" + heigth + "\" width=\"" + width + "\" style=\"stroke:#000000; fill:#ffffff\" />");
    }

    pushRight = 0;
    remme.add("<rect x=\"" + pushRight + "\" y=\"" + pushDown + "\" height=\"" + heigth + "\" width=\"" + width + "\" style=\"stroke:#000000; fill:#ffffff\" />");

    for (int i = 1; i < quantity; i++) {
        pushRight += width;
        remme.add("<rect x=\"" + pushRight + "\" y=\"" + pushDown + "\" height=\"" + heigth + "\" width=\"" + width + "\" style=\"stroke:#000000; fill:#ffffff\" />");
    }

    return remme;
}

```

Figur 4.21: Kodens bag assignText

## 4.8 Komponering af SVG string

BlueprintSVG attributten er som de andre, tom fra starten, men har en metode som arrangerer og implementerer de nu fyldte tekst elementer.

De attributter som er lister, har dog behov for at blive pakket ud, så de har deres egne compose-metoder.

```

private String composeRem(){
    String collector = "";

    for (int i = 0; i < rem.size(); i++) {
        collector += rem.get(i) + "\n";
    }

    return collector;
}

```

Figur 4.22: composeRem metoden

## 4.9 Sammenkobling

```
private void generateBlueprint() {
    int canvasBorderSides = 50;
    int canvasBorderTopBot = 50;
    int canvasBackX = canvasBorderSides + carport.getConfLength() + canvasBorderSides;
    int canvasBackY = canvasBorderTopBot + carport.getConfWidth() + canvasBorderTopBot;

    assignCanvasBack(canvasBackX, canvasBackY);
    assignMarkerHead();
    assignMarkers(canvasBorderSides, canvasBorderTopBot);
    assignCanvasFront(canvasBorderSides, canvasBorderTopBot);
    assignCanvasFill();
    assignText(canvasBorderSides, canvasBorderTopBot);
    assignStolpe();
    assignRem();
    assignSper();
    assignStern();
    if (carport.getRoof().inclination != 0) {
        assignLegte();
    }
}

blueprint.setBlueprintSVG(blueprint.composeSVG());
} // generateBlueprint
```

Figur 4.23: Kodens bag generateBlueprint

For at få hele møllen til at køre, har vi en metode som kronologisk kalder de individuelle elementers metoder og til sidst sætter den færdige tekstværdi ind i blueprintSVG variabelen. Denne metode, se figur 5.23, sætter også de overordnede mål for billede kanvasset og forholdet til carport kanvasset.

## 4.10 Resultat

Her vises indholdet af blueprintSVG string værdien når programmet er kørt og har genereret alle målene og placeringerne på elementerne i 2 dele, se figur 5.24 og 5.25 (2 næste sider):

```

<?xml version="1.0" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<!-- CanvasBack -->
<svg version="1.1"
  xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  height="490" width="430" viewBox="0 0 430 490"
  preserveAspectRatio="xMinYMin">

  <!-- MarkerHead -->
  <defs>
    <marker
      id="beginArrow"
      markerWidth="12"
      markerHeight="12"
      refX="0"
      refY="6"
      orient="auto">
      <path d="M0,6 L12,0 L12,12 L0,6" style="fill: #000000;" />
    </marker>
    <marker
      id="endArrow"
      markerWidth="12"
      markerHeight="12"
      refX="12"
      refY="6"
      orient="auto">
      <path d="M0,0 L12,6 L0,12 L0,0" style="fill: #000000;" />
    </marker>
  </defs>

  <!-- MarkerX -->
  <line x1="30" y1="55" x2="30" y2="435"
    style="stroke: #000000;
    marker-start: url(#beginArrow);
    marker-end: url(#endArrow);"/>

  <!-- MarkerY -->
  <line x1="55" y1="455" x2="375" y2="455"
    style="stroke: #000000;
    marker-start: url(#beginArrow);
    marker-end: url(#endArrow);"/>

  <!-- CanvasText -->
  <text style="text-anchor: middle" transform="translate(20,245) rotate(-90)">390cm </text>
  <text style="text-anchor: middle" x="215" y="470">330cm </text>
  <text style="text-anchor: middle" transform="translate(37,245) rotate(90)">Carport front</text>

  <!-- CanvasFront -->
  <svg x="50" y="50" width="330" height="390">

  <!-- CanvasFill -->
  <rect x="0" y="0" height="390" width="330" style="stroke:#000000; fill:#797D7F" />

```

Figur 4.24: Resultat del 1

```

<!-- Stolpe -->
<rect x="95" y="0" height="10" width="10" style="stroke:#000000; fill:#ffffff"/>
<rect x="225" y="0" height="10" width="10" style="stroke:#000000; fill:#ffffff"/>
<rect x="95" y="380" height="10" width="10" style="stroke:#000000; fill:#ffffff"/>
<rect x="225" y="380" height="10" width="10" style="stroke:#000000; fill:#ffffff"/>

<!-- Rem -->
<rect x="0" y="0" height="5" width="330" style="stroke:#000000; fill:#ffffff" />
<rect x="0" y="385" height="5" width="330" style="stroke:#000000; fill:#ffffff" />

<!-- Sper -->
<rect x="0" y="0" height="390" width="5" style="stroke:#000000; fill:#ffffff" />
<rect x="45" y="0" height="390" width="5" style="stroke:#000000; fill:#ffffff" />
<rect x="92" y="0" height="390" width="5" style="stroke:#000000; fill:#ffffff" />
<rect x="139" y="0" height="390" width="5" style="stroke:#000000; fill:#ffffff" />
<rect x="186" y="0" height="390" width="5" style="stroke:#000000; fill:#ffffff" />
<rect x="233" y="0" height="390" width="5" style="stroke:#000000; fill:#ffffff" />
<rect x="280" y="0" height="390" width="5" style="stroke:#000000; fill:#ffffff" />
<rect x="327" y="0" height="390" width="5" style="stroke:#000000; fill:#ffffff" />

<!-- Stern -->

<!-- Legte -->
<rect x="0" y="0" height="7" width="330" style="stroke:#000000; fill:#ffffff fill-opacity="0.6" stroke-opacity="0.3"/>
<rect x="0" y="383" height="7" width="330" style="stroke:#000000; fill:#ffffff fill-opacity="0.6" stroke-opacity="0.3"/>
<rect x="0" y="49" height="7" width="330" style="stroke:#000000; fill:#ffffff fill-opacity="0.6" stroke-opacity="0.5"/>
<rect x="0" y="86" height="7" width="330" style="stroke:#000000; fill:#ffffff fill-opacity="0.6" stroke-opacity="0.5"/>
<rect x="0" y="123" height="7" width="330" style="stroke:#000000; fill:#ffffff fill-opacity="0.6" stroke-opacity="0.5"/>
<rect x="0" y="160" height="7" width="330" style="stroke:#000000; fill:#ffffff fill-opacity="0.6" stroke-opacity="0.5"/>
<rect x="0" y="341" height="7" width="330" style="stroke:#000000; fill:#ffffff fill-opacity="0.6" stroke-opacity="0.5"/>
<rect x="0" y="304" height="7" width="330" style="stroke:#000000; fill:#ffffff fill-opacity="0.6" stroke-opacity="0.5"/>
<rect x="0" y="267" height="7" width="330" style="stroke:#000000; fill:#ffffff fill-opacity="0.6" stroke-opacity="0.5"/>
<rect x="0" y="230" height="7" width="330" style="stroke:#000000; fill:#ffffff fill-opacity="0.6" stroke-opacity="0.5"/>

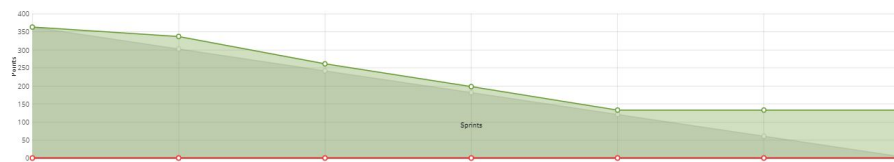
</svg>
</svg>

```

Figur 4.25: Resultat del 2

## Kapitel 5

# Status på implementation, Joakim/Lasse



Figur 5.1: Sprint forløb

Som det kan ses af figur 6.1 er vi som udgangspunkt nået i mål med vores plan for projektet, som sluttede efter vores 4. sprint forløb. PO kan modtage anmodninger fra en kunde, med specifikke dimensioner til en carport, samt valg af materialer. Konfigurationen er automatiseret, så den skaber en stykliste, plantegning og prisberegner til anmodningen ud fra de nødvendige materialer samt dele til disse dimensioner.

Anmodningen ligger i deres database, og kan derfra behandles af sælger, som også kan ændre status for hvor i behandlingsprocessen anmodningen er nået til.

### 5.1 Det opbyggede projekt

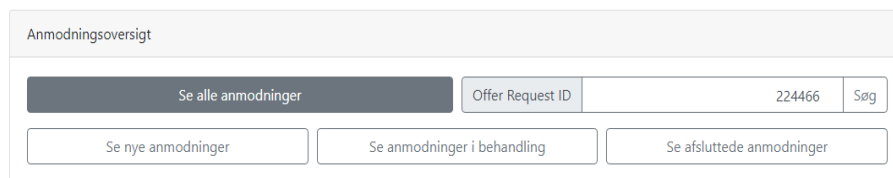
- **Nr. 1, Carport konfigurator:** Denne userstory er blevet implementeret og har skabt fundamentet for resten af konfiguratorens funktionaliteter, således at en kunde kan skræddersy en carport til dem selv. Beskrivelsen af den lyder som følger: “Som kunde vil jeg kunne sammensætte en carport med de måler jeg selv vælger, så sælger kan sende mig et personligt tilbud.”.

Arbejdet med denne userstory har bestået af udvikling på front-end. Formularen til at indtaste og vælge gyldige mål.

- **Nr. 58, Anmodninger: Oversigt:** Userstory er beskrevet således: “Som ansat vil jeg kunne se alle anmodninger der er forekommet i firmaet, så jeg kan finde en specifik anmodning eller holde overblik.”.

Dette er blevet implementeret med en udvidelse af carport konfiguratoren. Dette indebærer en dato for hvornår en anmodning er blevet oprettet og hvornår en anmodning sidst har set en aktivitet. Eksempelvis ved en ændring af status herpå.

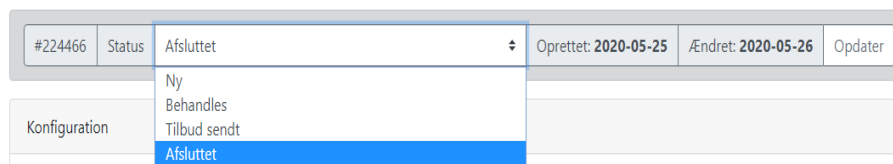
Udover en oversigt kan man som resultat af arbejdet med denne story, også tilgå en anmodning direkte, ved indtastning af et ID, som set på figur 6.2 nedenfor.



Figur 5.2: Anmodningsoversigt

- **Nr. 62, Anmodninger: Status:** Denne userstory beskrives med: “Som ansat vil jeg kunne opdatere status på anmodninger, så jeg kan redigere og holde overblik over hvilke anmodninger der er aktuelle.”.

Implementering af denne userstory er resulteret i en ny værktøjslinje som man kan benytte på en specifik anmodning, for at ændre status på den, som det ses på figur 6.3.



Figur 5.3: Ændring af anmodningens status

Her bliver en liste over alle mulige status, for en anmodning, hentet fra databasen. Se figur 6.4.

```

17 <form action="FrontController" method="POST">
18   <div class="alert alert-dark px-3">
19     <div class="input-group">
20
21       <div class="input-group-prepend">
22         <span class="input-group-text border-secondary">#{requestScope.offerRequest.confId}</span>
23         <span class="input-group-text border-secondary">Status</span>
24       </div>
25       <select class="form-control custom-select border-secondary" name="offerRequestStatus">
26         <c:forEach var="statusList" items="${requestScope.offerRequestStatusList}">
27           <option
28             <c:choose>
29               <c:when test="${requestScope.offerRequest.carport.confStatus.equals(statusList)}">
30                 selected
31               </c:when>
32             </c:choose>
33             >#{statusList}</option>
34         </c:forEach>
35       </select>
36       <div class="input-group-append bg-white rounded">
37         <span class="input-group-text border-secondary">
38           Oprettet: <b>#{requestScope.offerRequest.carport.CREATED_DATE}</b>
39         </span>
40         <span class="input-group-text border-secondary border-right-0">
41           Ændret: <b>#{requestScope.offerRequest.carport.changedDate}</b>
42         </span>
43       </div>
44       <input type="hidden" name="target" value="viewOfferRequest">
45       <input type="hidden" name="confId" value="${requestScope.offerRequest.confId}">
46       <input type="hidden" name="pageFunction" value="1">
47       <button type="submit" class="btn btn-outline-secondary" value="submit">Opdater</button>
48     </div>
49   </div>
50 </form>
51

```

Figur 5.4: Koden bag ændringen til statuslinje

- **Nr. 91, Bestil carport: Plantegning (Funktionalitet og logik):** Fokuset med denne userstory er arbejdet med back-end delen af hvordan en plantegning skal tegnes. Beskrivelsen lyder derfor: “En plantegning skal kunne genereres ud fra test data, således at den senere kan blive implementeret i produktet.”.

Som resultat af den logik der er blevet udviklet med denne userstory, er det en string der bliver genereret i form af SVG kode. Denne string er sammensat af mange forskellige elementer som en given carport består af. Denne string ses på figur 6.5 (næste side).



```

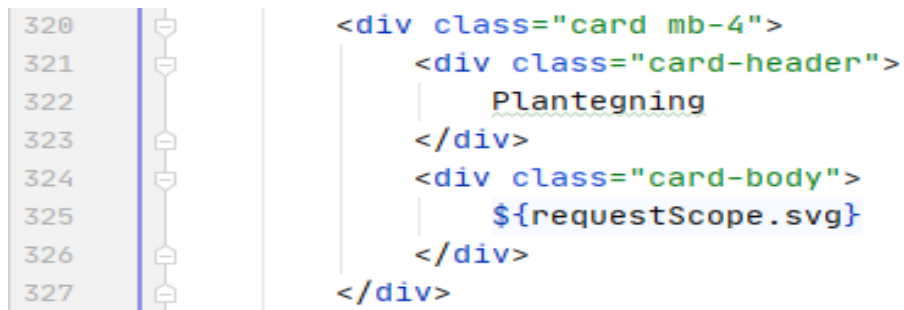
25 public String composeSVG() {
26
27     String SVG = "<?xml version=\"1.0\" ?>\n" +
28         "<!DOCTYPE svg PUBLIC \"-//W3C//DTD SVG 1.1//EN\""+
29         "\"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd\">\n" +
30         "\n" +
31         "<!-- CanvasBack -->" + "\n" +
32         canvasBack + "\n\n" +
33         "<!-- MarkerHead -->" + "\n" +
34         markerHead + "\n\n" +
35         "<!-- MarkerX -->" + "\n" +
36         markerX + "\n\n" +
37         "<!-- MarkerY -->" + "\n" +
38         markerY + "\n\n" +
39         "<!-- CanvasText -->" + "\n" +
40         canvasText + "\n\n" +
41         "<!-- CanvasFront -->" + "\n" +
42         canvasFront + "\n\n" +
43         "<!-- CanvasFill -->" + "\n" +
44         canvasFill + "\n\n" +
45         "<!-- Stolpe -->" + "\n" +
46         composeStolpe() + "\n" +
47         "<!-- Rem -->" + "\n" +
48         composeRem() + "\n" +
49         "<!-- Sper -->" + "\n" +
50         composeSper() + "\n" +
51         "<!-- Stern -->" + "\n" +
52         composeStern() + "\n" +
53         "<!-- Legte -->" + "\n" +
54         composeLegte() + "\n" +
55         "</svg>\n" +
56         "</svg>";
57
58     return SVG;
59 } //composeSVG
60
61 private String composeRem(){

```

Figur 5.5: Koden bag plantegnings-sammensætningen

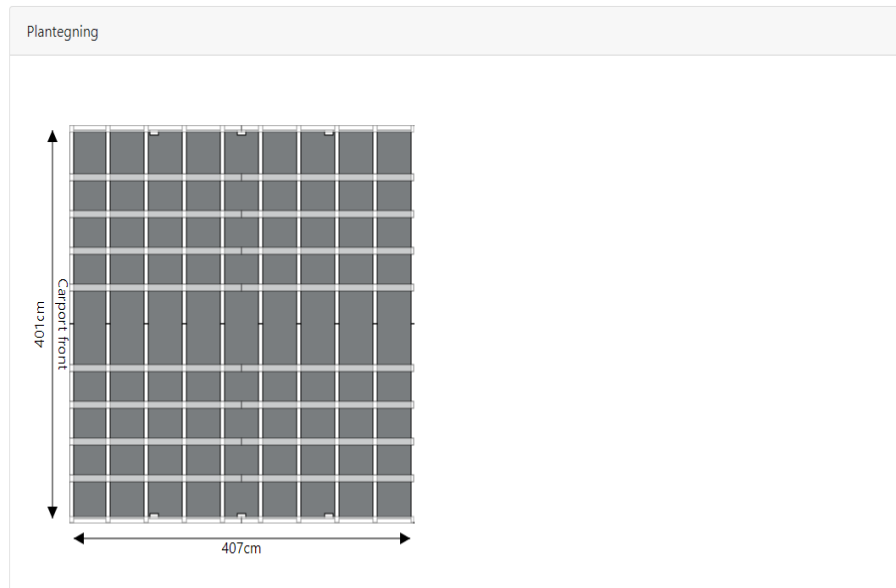
- **Nr. 143, Bestil carport: Plantegning (Front-end implementation):** I denne userstory er fokus på at implementere koden, fra userstory nr. 91, på front-end. Beskrivelsen lyder “Den genererede plantegning skal være implementeret i produktet, således at den kan blive vist for sælger, og ultimativt for kunden.”.

Den string som tidligere er blevet genereret, bliver indsat som en attribut i HTTP request. Derfra benyttes JSTL-kode til at vise denne attribut på siden. Dette fremgår af koden på figur 6.6.



Figur 5.6: Indhentning af plantegningen

Resultatet heraf ser således ud: Se figur 6.7.



Figur 5.7: Plantegningen visuelt

- **Nr. 2, Carport konfigurator: Vægge og tagrejsning:** Denne user-story omfatter mange ændringer til koden da både eksisterende front-end og back-end skulle tilpasses og udvides, således at yderligere funktioner kunne blive realiseret. Beskrivelsen af denne story lyder: “Når jeg har defineret målene på min carport, kan jeg indsætte væg sektioner. Jeg kan vælge om mit tag skal være fladt eller have tagrejsning, inklusiv vinkel på tag.”.

Udførelsen af denne story resulterede i en mindre ændring til front-end, som det ses på figur 6.8 (næste side).

Tagrejsning ☐ grader °

Materiale

Væg beklædning ☐ Højre side ☐ Venstre side ☐ Bagside

Figur 5.8: Mulighed for tagspecifikationer

- **Nr. 49: Ansat funktionalitet: Ændring af materialer:** Formålet med denne userstory er at man som ansat kan ændre i produktsortimentet. Således at kunder kan vælge nye materialer til deres carport. Denne userstory er beskrevet som følger: “Som ansat vil jeg kunne ændre på hvilke materialer som er tilgængelige, så virksomheden kan tilføje eller fjerne fra sortimentet.”.

Dette er blevet implementeret på ‘værktøj’-siden. Dette ses på figur 6.9.

Komponent værktøj

Komponent  id  Redigér

Komponent

Mål  cm  cm  cm

Salg  kr.  kr.

Figur 5.9: Ny komponent tilføjes

- **Nr. 53, Ansat funktionalitet: Ændring af priser:** Ligesom userstory nr. 49. Omhandler denne story, funktionalitet der gør det muligt, for en medarbejder hos Johannes Fog, at ændre salgsprisen på en eksisterende komponent. Se figur 6.10

Komponent værktøj

Komponent

2000

Redigér

Ny komponent

Komponent

Type

Stolpe

Materiale

Trykimprægneret

Salg

Pris

475 kr.

Ny pris

Opdater

Figur 5.10: Redigering af nuværende komponent

Dette gøres ved at sende en ‘UPDATE’-query til MySQL databasen, med den nye pris. Hver komponent har deres eget ID, som i dette tilfælde bliver benyttet til at specificere hvilken komponent, den nye pris tilhører og skal opdateres i databasen. Dette fremgår af koden på figur 6.11.

```

104 public static boolean updateComponentSalesPrice(int compId, int newSalesPrice) {
105     try {
106         Connection con = Connector.connection();
107         String SQL = "UPDATE components SET salesPrice = ? WHERE compId = ?";
108         PreparedStatement ps = con.prepareStatement(SQL);
109         ps.setInt( parameterIndex: 1, newSalesPrice);
110         ps.setInt( parameterIndex: 2, compId);
111         ps.executeUpdate();
112         return true;
113     } catch (SQLException | ClassNotFoundException ex) {
114         System.out.println(ex);
115         return false;
116     }
117 }

```

Figur 5.11: Koden bag opdateringen

## 5.2 Projektets fremtid

Projektet har flere funktionaliteter planlagt for fremtiden. Efter aftale med PO, gav det bedst mening for projektet, at vi kunne få en simpel carport-anmodning sendt hele vejen igennem system, inden vi tilføjede flere funktionaliteter. Havde forløbet været forlænget ville PO have haft mulighed for at vælge i mellem flere forskellige userstories.

På figur 6.12 (a) (næste side), kan disse muligheder ses og herefter bliver de uddybet.

Votes		User Stories	Status	Points
<input type="checkbox"/>	▲ 0	Important #65 Carport konfigurator: Skur	New	34
<input type="checkbox"/>	▲ 0	Important #4 Carport konfigurator: Materiale valg	New	5
<input type="checkbox"/>	▲ 0	secondary #69 Bestil carport: Leveringsomkostninger	New	5
<input type="checkbox"/>	▲ 0	extra #43 Carport konfigurator: Advarsler	New	21
<input type="checkbox"/>	▲ 0	extra #67 Ansæt funktionalitet: Tilbud	New	21
<input type="checkbox"/>	▲ 0	extra #68 Ansæt funktionalitet: Planlagt tilbud	New	13
<input type="checkbox"/>	▲ 0	extra #146 JavaScript til visualisering af plantegning for kunde i anmodning	New	34

(a) Userstories
(b) Estimerede point

Figur 5.12: Figurer for ekstra userstories

Figur 6.12 viser fremtidsplanerne for projektet. Disse er ikke færdigt arbejdet, og ville som selvfølge undergå en revurdering inden disse skulle fremlægges for PO i et sprintmøde.

- **Nr 65, Carport konfigurator: Skur:** Denne userstory skal give kunden mulighed for at tilvælge et skur til carporten. Efter en revurdering af selve userstoryen, skal der skrives endnu en userstory, som beskriver kundens mulighed for at vælge skurets dimensioner, og en beregner som kan fortælle kunden hvorvidt skuret bliver for stort.

Dette ville kræve tilføjelser og ændringer i både Databasen, funktionslaget og Front-end. Derfor ville denne userstory blive en del større end estimeret.

- **Nr 4, Carport konfigurator: Materiale valg:** Ideen bag denne userstory bunder ud i, at vi havde ideen om at lave system, så man kunne vælge at konfigurere sin egen carport eller vælge færdiglavede konfigurationer.

Funktionaliteten til at vælge materialer og træsort er implementeret, så kunden kan vælge fra et sortiment, når han konfigurerer en specifik carport. Derfor kan denne userstory kan ses, som et levn fra fortiden.

- **Nr 69, Bestil Carport: Leveringsomkostninger:** Som en del af omkostningerne for kunden, er transport til hele landet en del af prisen. Derfor kunne dette tilvælges som en funktionalitet til hjemmesiden, der ville kunne udregne leveringsomkostningerne ud fra kundens indtastede postnummer. Den skulle også implementeres på ordrebeskrivelsen. Denne pris ville indgå i den totale pris for carporten.

- **Carport konfigurator: Advarsler:** Hvis kunden valgte funktionaliteter som kunne være i strid mod lovgivningen af konstruktionen, skulle han modtage en advarsel om dette.

Ligeledes hvis kunden gav et modstridigt input til skurets og carportens dimensioner, skulle han modtage en advarsel. Sidst skulle det noteres i anmodningen, hvis den var blevet gennemført trods advarsler.

- **Nr 67, Ansat funktionalitet: Tilbud:** Hjemmesiden skal vise tilbud på diverse carporte i perioder. Dette kræver at siden er udstyret med færdiglavede carporte, som er en userstory for en anden gang.

Den ansatte kan sætte en carport i 'tilbud' status. Prisen kan reguleres med en procentvis på den totale pris. Hjemmesiden skal genkende varen som værende på tilbud. Selve tilbuddet kan ikke overstige 95 procent eller være negativ.

- **Nr 68, Ansat funktionalitet: Planlagt tilbud:** Det ville indebære at den ansatte kunne sætte en carport som tilbud på et specifikt tidspunkt, at han kunne vælge perioden tilbuddet skal være aktivt og at siden vil advare ham hvis der allerede er et planlagt tilbud for den valgte carport.

- **Nr 146, JavaScript til visualisering af plantegning for kunde i anmodning:** Her er tanken, at kunden kan se en visualisering af de mål han indtaster til en carport. Tegningen skal ændre sig samtidigt med at han aktivt ændrer dimensionerne på i konfiguratoren.

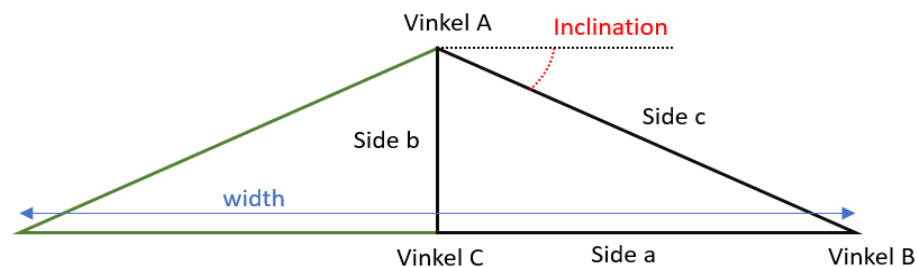
## Kapitel 6

# Test, Alex

Vi har ikke lavet mange tests i IntelliJ. Vi har i stedet benyttet os af at udvikle metoderne i et separat program, for så at implementere dem direkte i klasserne når metoderne er klar, tilpasse vores variabel-navne til vores projekt og så teste om de virker ved at køre programmet. Dette var let og fantastisk hurtigt, men ikke 'best practice' og kan give problemer i større, mere sofistikerede projekter. Så vi ønsker at implementere unit testing mere i fremtiden.

Vi har dog valgt at køre et par unit tests på vores Roof klasse, da denne indeholder nogle ret ligetil, matematiske metoder.

### 6.1 Forklaring



Figur 6.1

Som det ses på figur 7.1 ovenfor, så indeholder et tag mange mål. Når en kunde bygger en carport med tagrejsning, så vælger de kun hældningsgraden, hvilket vi kalder inclination. Denne grad, sammen med carportens bredde, som vi her kalder width, indføres i visse metoder som giver os de resterende mål som vi har brug for til vores stykliste.

### 6.1.1 Siden c (calcSideC())

Siden c eller hypotenusen i vores trekant er det første vi udregner. Udover at give et godt udgangspunkt for senere at finde siden b, tagets højde, så er det en vigtig dimension, både for vores komponenter som bruges til taget eller ligger langs taget, men også direkte for vores tagbelægning.

Der er foretaget test af maksimum og minimum værdierne for inclinationen, samt en tilfældig inclination's værdi indenfor disse.

### 6.1.2 Siden b (calcRoofHeight())

Siden b fungerer som tagets højde, som går fra carportens top til tagets top. Denne dimension er vigtig for vores lodrette midterspær og for beklædningen af tagets lodrette sider.

Der er foretaget test af maksimum og minimum værdierne for side c, samt en tilfældig side c værdi indenfor disse.

### 6.1.3 Konvertering mellem datatyper (toInt())

Vores datatyper for tal i vores projekt er altid integers, men mange af de Math.-funktioner som java har, kræver double værdier (da der er behov for decimaler). Dette betyder at vi i vores to forrige metoder har været nød til at konvertere vores integers til double værdier, hvilket er let nok, men når disse værdier skal genintroduces til programmet løber man ind i nogle problemer. Derfor har vi lavet en metode som tager imod en double værdi og returner den som nærmeste integer.

Der er foretaget tests af værdier som rundes ned og af værdier som rundes op.



## Kapitel 7

# Arbejdsproces, Lasse

### 7.1 Arbejdsprocesen faktuel

Gruppe 1, er en gruppe bestående af 3 medlemmer. Vi mistede vores fjerde medlem dagen efter projektets start, og en af de resterende tre, manglede sin arbejdsstation i 2 uger. Men vi har formået er nå vores målsætninger.

#### 7.1.1 Mødestruktur

##### Dagligt SCRUM møde

Der afholdes et dagligt SCRUM-møde internt i gruppen, hvor weekender er fritaget medmindre andet er aftalt. Mødetiden er klokken 10 om formiddagen, og denne tid kan også rykkes efter intern aftale. Disse møder færdiggøres først når en dagsorden er gennemarbejdet og debatteret.

Til mødet skal der, som mindste krav, være gennemgået en dagsorden som indeholder hvad der var aftalt op til mødets begyndelse samt hvad der arbejdes videre på til mødet dagen efter.

Der er udarbejdet en skabelon over en fast dagsorden, se figur 8.1. De vigtigste pointer bliver der lavet referat på i form af en "LOG"<sup>1</sup>. Der skal være skrevet LOG til alle arbejdsdage, og disse skrives ind i rapportens bilag under LOG.

Dagligt møde:  
(forventet tid 5-15 minutter):

Dirigent:  
Referent:

Punkt 1 – Opsummering

Hvert medlem gennemgår:

- Jeg har arbejdet på userstory/task...
- Jeg har nået/ikke nået det jeg ville...
- Jeg har haft succes/problemer med...

Punkt 2 – Opgaven fortsat

Hvert medlem gennemgår:

- Jeg skal arbejde på userstory/task...
- Jeg har brug for...
- Jeg forventer at nå...

Punkt 3 – Næste møde

- Vores næste møde er...

Figur 7.1: Daglige mødeskabelon

---

<sup>1</sup>LOG - Bilag: Afsnit LOG.

Hver dag skal der vælges en referent og en dirigent. Vi valgte fra start, at uddele disse roller som en fast titel.

Referenten har været Lasse, som har ført LOG under hele forløbet og til alle møder. Dirigenten har været Alex, som med struktur og en konstruktiv tilgang har ledet møderne dagligt og ugentligt. Joakim blev tildelt rollen som agerende SCRUM-Master under hele projektet, og sørgede for vi holdte overblikket.

På denne måde kunne vi overholde vores tidsplan og kunne nemt revurdere arbejdsvilkårene ved at alle havde et individuelt ansvar for det fælles mål.

### Ugentligt SPRINT retrospective møde

Der afholdes et ugentligt SPRINT-møde internt i gruppen efter Review- og Sprintmøder med PO<sup>2</sup>, hvor tidslinjen og arbejdsbyrden debatteres. Under mødet sidder alle medlemmer med Backloggen<sup>3</sup> åben i [taiga.io](https://tree.taiga.io/).

Her fremlægges problemstillinger, og der revurderes på baggrund af PO's anvisninger, om der er user-stories som er estimeret højere end forventet og derfor skal omlægges.

Det er ved brug af vores ugentligt Sprintmøde-skabelon, at vi får planlagt hensigtsmæssigt efter Sprint- og Planningmøderne, se figur 8.2 (næste side).

Processen af arbejdet er tilrettelagt efter Backloggen. Ideen er, at vi får løst fremgangsmetoden for vores arbejdsproces ved hjælp af Scrum- og Sprint-metoderne. Dernæst bygger vi projektet op gennem de modelværktøjer vi kender som: ER-diagram, Navigationsdiagram, Domænemodel<sup>4</sup>, klassediagram<sup>5</sup> og aktivitetsdiagrammer<sup>6</sup>, inden vi påbegynder det konkrete kodningsarbejde.

ER- og Navigationsdiagrammet er beskrevet i deres respektive kapitler, resten af modellerne kan findes i bilag.

---

<sup>2</sup>PO: Product Owner

<sup>3</sup>Backlog: <https://tree.taiga.io/project/joakimkss-fog-carport/timeline>

<sup>4</sup>Domænemodel: Bilag: Side X

<sup>5</sup>Klassediagram: Bilag: Side X

<sup>6</sup>Aktivitetsdiagram: Bilag: Side X

Sprint møde:  
(forventet tid, 45-90 minutter)

Dirigent:  
Referent:

#### Punkt 1 – Opsummering

Er alle usecases klaret?  
Hvis en usecase ikke er klaret, hvorfor og hvad nu?  
Er der opstået nogen issues i forrige sprint?  
Er disse issues blevet løst? Eventuel uddybning.  
Er der opstået nye tasks i løbet af sprintet?  
Er disse tasks blevet løst? Eventuel uddybning.

#### Punkt 2 – Aktivitet

Har vi været aktive nok?  
Har arbejdspresset været for stort, for småt eller tilpas?  
Skal nogen opgaver omfordeles?

#### Punkt 3 – Samarbejde

Hvordan går samarbejdet?  
Bruger vi for meget, for lidt eller tilpas tid på vores møder og generelle samarbejde?  
Er vi gode nok til at kommunikere med hinanden?

#### Punkt 4 – Ny sprint

Hurtig gennemgang af planlagte userstories og tasks.  
Redefinering af uopklarede userstories eller tasks.  
Introduktion af identificerede userstories eller tasks.  
Gennemgang af arbejdskronologi og uddeling af opgaver/ansvar.

#### Punkt 5 – Diverse

Yderligere beretninger eller bemærkninger.  
Har nogen særlige planer eller aktiviteter i løbet af sprintet?

Figur 7.2: Ugentligt Sprintmøde-skabelon

Som det kan ses af figur 8.2 kommer vi ind på og gennemgår ugens forløb på fornuftig og struktureret vis. Med opsummering og revurdering kan vi overholde tidsplanen i Backloggen efter aftale med PO. Der tages også højde for privatlivet på sidelinjen i skabelonens sidste punkt: Diverse.

Disse møder blev afholdt enten lige efter et møde med PO eller fredag morgen klokken 10, før det daglige SCRUM-møde.

Userstory	Estimerede point	Total point	Sprint forløb
96: Hjemmeside	13		1
1: carport konfigurator, front-end	13	26	1
36: Bestil carport konfigureret	21		2
92: Bestil carport: Stykliste	55	76	2
58: Anmodninger: Oversigt	21		3
62: Anmodninger: Status	8		3
91: Bestil carport: Plantegning (funktionalitet og logik)	34	63	3
143: Bestil carport: Plantegning (Front-end)	21		4
2: carport konfigurator: Vægge og tegrejsning	34		4
49: Ansat funktionalitet: Ændring af materialer	5		4
53: Ansat funktionalitet: Ændring af materialer	5	65	4

Tabel 7.1: Sprint oversigt

## 7.2 SPRINT kronologisk

Opgaven som SCRUM-master var tildelt Joakim, som håndterede rollen ved altid at sørge for, at der blev skrevet relevante subtasks til de userstories vi arbejdede på hver især. Vi sad altid i plenum og debatterede hvorledes en user-story, samt subtask til disse, bedst kunne løses, så det stemte overens med vores tidsplan. Sprint oversigten kan ses via tabel 8.1.

### 7.2.1 Scrumreview- og Sprintmøde, Torsdag d. 23/04 og Fredag d. 24/04

Vi blev bekræftet i hvordan vi skulle forholde os til værktøjet og arbejdsprocessen. Vi forstod hvordan det vil sige at planlægge på baggrund af de valg, som bliver taget af PO for projektets retning, og hvordan vi spiller en rolle, når der skal argumenteres for hvilke valg der har en effekt på resultatet.

Vi fremlagde til mødet hvordan vi havde opbygget backloggen. Der blev fremvist et udkast på en domænemodel og et navigationsdiagram, og i samarbejde med PO fik vi sat en retning på disse tanker. Som det fremgår af LOG til disse to møder<sup>7</sup>, fremlagde vi en brugerflade med placeholder data, som fungerer visuelt og kunne give PO en ide om hvordan der blev navigeret i programmet, samt hvad han kunne beslutte sig til af funktionalitet. PO var tilfreds med de tanker vi havde gjort os, og bad os prioritere 2 userstories: Nr. 36, Bestil carport: Konfigureret og Nr. 92, Bestil carport: Stykliste.

Efter mødet fredag, var der på PO's anvisninger udarbejdet en prioritering af de forskellige userstories, som vi havde fremlagt sammen med visualiseringen af programmet. Derved kunne vi planlægge den konkrete kodning, som havde været skubbet indtil bekræftelsen af projektets retning. Her kunne vi opdatere backloggen og revurdere point-estimationen af userstoriesne.

<sup>7</sup>Bilag: LOG: Scrum review, Torsdag d. 23/04 og Fredag d. 24/04

### 7.2.2 Scrumreview- og Sprintmøde, Torsdag d. 30/04 og Fredag d. 1/05

Til forskel fra sidste møde med PO, havde vi lært hvordan vi skulle forholde os til SCRUM værktøjet. Derfor var der til dette møde en tilfredsstillende fremlæggelse af de userstories vi havde arbejdet på, med en kort demo af den funktionalitet vi havde færdiggjort. PO så sammenkoblingen mellem logikken i koden til styklisten og hvordan det blev hentet fra databasen og vist på front-end, som det fremgår af LOG<sup>8</sup>.

Vi nåede i mål med de opgaver der var aftalt med PO, dog måtte vi revurdere arbejdsbyrden, da vi havde bidt over lidt for meget at løse på en uges arbejde. Derved lærte vi at vurdere point-estimeringen af userstories korrekt, og kunne herfra argumentere for hvad der var vigtigst at få løst på fornuftig vis.

Endnu engang blev backloggen omstruktureret ud fra vores erfaringer i samarbejde med PO, og vi fandt den rigtige mængde arbejde for en uge i de kommende userstories: Nr. 58, Anmodninger: Oversigt, Nr. 62, Anmodninger: Status og Nr. 91, Bestil carport: Plantegning (funktionalitet og logik).

#### Sprint forløb efterfølgende

Som det fremgår af de to ovenstående eksempler, blev sprintprocessen langt mere triviel at arbejde med efter vi havde fået bekræftet hvordan vi håndterede userstories og hvordan vi estimerede dem korrekt. I afsnittet "Arbejdsprocessen reflekteret" er det beskrevet hvilke overvejelser det medbragte i vores arbejdsproces.

Vi havde nu styr på hvordan vi skulle forholde os til værktøjet, og kunne effektivt fremlægge vores argumentation for PO i de kommende møder. Dette resulterede i en flydende arbejdsproces og god forståelse efter det andet sprint møde og til projektets afslutning.

---

<sup>8</sup>Bilag: LOG: Scrumreview, Torsdag d. 30/04 og Fredag d. 1/05

## 7.3 Arbejdsprocessen reflekteret, Lasse

### 7.3.1 SCRUM som værktøj

Vi håndterede fra start opgaven ved at tage ejerskab for hver vores ansvarområde (vores titler: Dirigent, referent og SCRUM-master), samt at have indblik i hvad vi arbejdede på hver især.

Joakim sørgede for altid at have backloggen opdateret, og førte ordet i de møder vi havde med PO, samt de forskellige sprintmøder internt efterfølgende. Joakims rolle bestod af at opdatere backloggen og de forskellige sprints i forhold til de aftaler der var indgået med PO.

På denne måde havde vi alle en aktiv rolle i beslutningsprocessen, hvilket resulterede i, at alle følte de kunne komme til orde, og at arbejdsflowet var vel-fungerende. Den største 'udfordring' vi så i at en enkelt mand stod med hele ansvaret som scrum-master var, at retningen for opgaven kunne blive farvet af dette. Derfor blev det hurtigt indforstået, at dette ansvar, var en beslutning som skulle tages i plenum - så beslutningerne var fælles.

#### Det at skrive en god userstory

Det var på forhånd indforstået i gruppen, at der var et fælles ønske om at få planlagt vores arbejdsproces grundigt inden det egentlige kodningsarbejde. Der-ved kunne alle være med fra start og fornemme en retning for projektet. Det betød at vi tog godt imod SCRUM-værktøjet, som er en god måde at tage en ide (ordre) og inddele den i delmål og arbejdsprocesser - altså en planlægning af arbejdet.

Vi var usikre på hvordan vi skulle gribe SCRUM an. Derfor blev vi enige om, at bruge første sprint på at skrive en masse userstories, samt lave en visuel præsentation til PO i form af en demo og forskellige modeller. Visualiseringen skulle være en model som kunne tage imod de userstories PO vælger at priorisere, og dette var den rigtige måde at gribe processen an på. Dog var vi end ikke sikre på hvordan vi skulle estimere point til de enkelte userstories.

Efter første sprintmøde med PO, havde vi også lært hvad det vil sige at skrive en god userstory. Vi havde ikke helt forstået at skrive nogle optimale userstories. De var for store - én episk - og greb om flere opgaver på en gang. Vi måtte genskrive et par af dem for at kunne få valgt de rigtige til den kommende uges sprint.

Først efter andet sprintmøde, hvor vi havde prøvet at håndtere arbejdsbyrden aftalt med PO, kunne vi estimere pointgivningen korrekt. Vi havde gæbt over for meget arbejde for en enkelt uge med kun tre gruppemedlemmer. Vi havde arbejdet fra tidlig morgen til sen aften for at løse de valgte userstories, og dette viste sig ikke at være en optimal arbejdsmodel. Derved havde vi fået en fornemmelse for hvor meget en estimering af enkelte opgaver betyder.

Herfra var det ikke svært at håndtere backloggen, og SCRUM som værktøj. Vi havde fået bekræftet hvad det vil sige at skrive en god userstory og hvordan vi skal estimere point hensigtsmæssigt dertil. Arbejdsprocessen var noget nemmere efter at have forstået og debatteret effekten af disse situationer i gruppen, og vi fandt det meget ligetil at være produktive/effektive herfra.

### **Generel planlægning**

Som gruppe har vi fra dag et været struktureret. Første dag gik på at oprette projektet i de forskellige værktøjer: GitHub, taiga.io og Overleaf. Det handlede om at identificere hvad der skulle bruges til at holde overblikket i projektet, og skabe en daglig struktur for hvordan vi skulle håndtere de forskellige værktøjer.

Som det fremgår af LOG<sup>9</sup>, kan det ses hvordan vi bruger de første par dage på at oprette strukturer for vores arbejdsprocess, og først på tredje dagen igangsætter vi SCRUM som værktøj. Ideen er at de strukturer vi beslutter fra start er gennemgående og behandler bl.a. brugen af scrum og nedskrivning af daglig LOG. Dette viste sig at være en god og konstruktiv måde at håndtere nye værktøjer på, da disse dagligt bliver revurderet til scrum mødet.

### **Kort om Corona-pandemien**

Corona satte sit pres på hele verdenen lige før projektets start. Vi blev hurtigt enige i gruppen om, at det daglige fremmøde var det absolut vigtigste for vores projekts resultat. Derfor aftalte vi en fast mødetid klokken 10 hver morgen til scrum-mødet. Dette sammen med tilgængelighed og en stærk kommunikation gennem programmet Discord, gjorde at vi kunne fastholde en stærk arbejdsmoral og godt teamwork. Sidst men ikke mindst var det vigtigt at få skrevet en daglig LOG af arbejdet, så vi kunne dokumentere vores beslutningstagning.

---

<sup>9</sup>Bilag, LOG: Scrum møde: 1,2,3.

## Kapitel 8

# Konklusion, alle

### Arbejdsprocessen

Brugen af scrum som værktøj for første gang, kan være forvirrende til start, men bliver hurtigt en triviel proces når man i forvejen har en struktureret tilgang til projektet i form af daglig LOG-skrivning. Som gruppe erfarede vi, at det tager et par sprints at sætte sig ind i processen, og at det derefter er et meget effektivt værktøj at planlægge sin arbejdsproces nøjagtigt efter.

Det tog os 2 sprints at forstå hvordan man skriver den specifikke userstory samt estimerer den, så den er tilpasset i både arbejdsflow og mængde af arbejde. Vi lærte at processen handler om at kunne argumentere for sin struktur og planlægning overfor PO, så man i samarbejde med PO's ønsker kan strukturere processen efter hans ønsker. Derved føler PO at han betaler for det produkt han ønsker, samt at gruppen formår at nå de delmål der planlægges.

### Produktet

Vi har i samarbejde med PO identificeret et produkt, i form af en carport generator, som PO gerne vil have implementeret over flere sprint. Der er skabt en hjemmeside med en konfigurator til kunden samt flere værktøjer til den ansatte. Kunden kan specificere de ønskede dimensioner og vælge materiale ud fra FOG's sortiment. Den ansatte har mulighed for: at se alle anmodninger i en oversigt eller søge på enkelte anmodninger, redigere status og redigere tilbudspris på disse anmodninger, tilføje eller redigere komponenter til deres database, samt se en visualisering af kundens ønskede carport og en stykliste på de samlede komponenter og dele, som skal bruges.



Vi har oprettet en database, som indeholder data for FOG's sortiment samt de konfigurationer, som kunderne laver på siden. Den ansatte kan redigere i begge dele, og derved holde overblik på deres kommende opgaver, samt opdatere FOG's fremvisning af materialer på hjemmesiden.

Vi har opnået en funktionalitet, så en kunde kan lave en konfiguration med de ønskede dimensioner og materiale valg, sende den til FOG's database og gemme den i deres system. Her fra ligger den klar til behandling af en ansat.

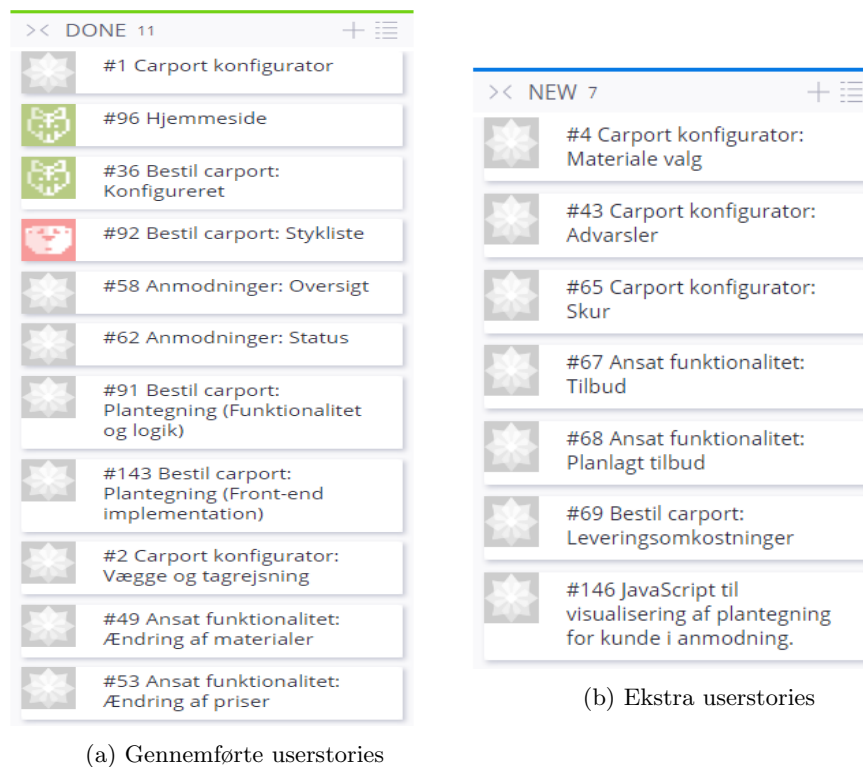
Vi har opnået en funktionalitet, hvor den ansatte i FOG kan åbne en konfiguration og se en prisberegning, en stykliste og en plantegning af denne. Disse bliver genereret og beregnet ud fra kundes ønskede mål, og sælger kan derved lave et tilbud til den enkelte anmodning.

# Kapitel 9

## Appendix

### 9.1 Product Backlog

Alle definerede user-stories fremgår forneden i to kategorier; gennemførte og ekstra.



Figur 9.1: Samlede userstories - Backlog

Gennemførte userstories, figur 10.1, (a), er stories hvor alle tasks er blevet udviklet og gennemførte. Altså er disse userstories blevet implementeret i produktet.

Ekstra userstories, figur 10.1, (b), er stories der endnu ikke er blevet påbegyndt, men er blevet defineret som værende relevant for produktet.

## 9.2 LOG, Lasse

Der afholdes et daglig SCRUM-møde, hvor weekender er fritaget medmindre andet er aftalt. Mødetiden er klokken 10 om formiddagen og mødet færdiggøres når en dagsorden er gennemarbejdet og debatteret. Til mødet skal der være gennemgået - som minimum - hvad der var aftalt op til mødets begyndelse samt hvad der arbejdes på til næste møde. Dette føres der referat på, og der bliver skrevet en daglig log af møderne ind i rapporten.

Der afholdes et ugentligt SPRINT-møde, hvor tidslinjen og arbejdsbyrden debatteres. Under mødet sidder alle medlemmer med Backloggen åben i [taiga.io](https://taiga.io). Her fremstilles problemstillinger, og der revurderes om der er user-stories, som er større end forventet og derfor skal om-lægges.

Efter Sprint- og Planningmeetings gennemgås en ugentligt mødeskabelon, som skal gøre arbejdsprocessen let forståelig.

Processen af arbejdet er tilrettelagt efter Backloggen. Ideen er, at vi får løst fremgangsmetoden for vores arbejdsprocess ved hjælp af Scrum- og Sprint-metoderne. Dernæst bygger vi projektet op gennem de modelværktøjer vi kender som: Domænemodel, klassediagram og navigationsdiagram inden vi påbegynder det konkrete kodningsarbejde.

### 9.2.1 Scrum møde 1, Torsdag d. 16/04-20

Første Scrum møde var en start af hele processen. Vi oprettede Github repository på Lasses Github: <https://github.com/Lforlasse/FOG-Carport> med regler for commits samt branches til hver enkelt medlem i gruppen.

Der blev oprettet taiga.io på Joakims profil med start af backloggen til fremtidige Sprints- og Scrummøder:

<https://tree.taiga.io/project/joakimkss-fog-carport/timeline>

Sidst blev der skabt en skabelon til både rapportskrivningen i Overleaf samt en fremgangsmetode til de daglige Scrum- og ugentligt Sprintmøder.

### **9.2.2 Scrum møde 2, Fredag d. 17/04-20**

Andet Scrum møde var en procesdiskussion, hvor vi debatterede hvilken retning projektet skulle have. Vi har blandt andet sat estimation på vores user-stories, samt rangerede vigtigheden af hver enkelt user-story. Der er blevet aftalt opgaver til weekenden, og her afholder vi ikke Scrum møder.

### **9.2.3 Scrum møde 3, Mandag d. 20/04-20**

Tredje Scrum møde er første gang vi rigtigt bruger Scrum metoden. Database strukturen er igangsat, og der vurderes om der skal være et opdigtet lagersystem med indkøbspris samt lagerstatus.

Der er oprettet et mavenprojekt ud fra skabelonen. Der skal rettes til, men der er en færdig skabelon som er pushet til GitHub. Der er skrevet en ny User-story til mock-ups og hjemmeside layoutet, og der er skabt mock-ups i HTML.

Vi besluttede, at det ville være en god ide at komme igang med at få lavet nogle domænemodeller, så der kan skabes et overblik af de konkrete opgaver. Sidst er der rettet til i rapportskabelonen.

### **9.2.4 Scrum møde 4, Tirsdag d. 21/04-20**

Databasestrukturen er blevet færdigarbejdet. Der er lavet et udkast til et EER-diagram over databasen, som bliver diskuteret sammen med spørgsmålet om klasse diagram. Ligeledes er navigationen i HTML'en færdig, der er en smule problemer med noget af CSS'en, men det bliver der rettet op på i dag.

Domænemodellen bliver revurderet i dag, og der arbejdes videre på User-story-1 med disse modeller, så vi kan blive klar til at kode.

### **9.2.5 Scrum møde 5, Onsdag d. 22/04-20**

User-tasken med udfyldelse af databasen er 80 procent færdigt. Klasserne i projektet er halvt færdigt. Disse bliver færdige i dag. De sidste navigationelle funktionaliteter er blevet udarbejdet i HTML'en. Vi klargør til første Sprint-møde i morgen.

### 9.2.6 Scrum møde 6, Torsdag d. 23/04-20

Subtasks i den nye User story er blevet omformuleret en smule. Der vurderes at den skal have nogle færre point i backloggen, og at den er tæt ved færdig. Hjemmesiden er færdigopsat med JSTL, så den forbinder korrekt og henter data.

Logicfacaden i projektet er fyldt med placeholder data ind til vi når til de user-stories.

Klasserne: carport, komponent og order-class er påbegyndt, men viser sig også som en større opgave end forventet. Der er lidt problemer med indførelsen af visse metoder, da vi skal bruge databasemetoderne.

### Scrum Review møde 1, Torsdag d. 23/04-20

Første Sprint møde med produkt-owner var en god bekræftelse af, at vi har den rigtige retning med projektet.

Vi præsenterede en brugerflade med placeholder-data, som fungerer visuelt, og som snart er klar med den gældende funktionalitet i sammensplit med databasen. Det er forbindelsen mellem databasen og funktionaliteten som mangler, men vi er stort set i mål.

Grundet vores underbemanding, er den konkrete kodning skubbet en uge i backloggen. Vi ser ikke grund til bekymring for projektet, da vores for-arbejde er rigtig godt.

### 9.2.7 Scrum møde 7, Fredag d. 24/04-20

SCRUM blev i dag først afholdt efter planningmeeting, som var klokken 13:45.

Backloggen samt Userstories er nu faldet på plads efter vores første planningmeeting. Vi er alle stadig indforstået med at processen kan variere fremadrettet yderligere.

Der skal i dag ses på klasserne og den aktuelle kodning af disse. Vores planlægning heraf er veldokumenteret og vi er forberedt. Der bliver også bygget videre på databasen, så der er en tabel som kan modtage vores carport-konfigurationer, samt en tilknyttet status tabel.

### Sprint møde 1, Fredag d. 24/04-20

Der blev i samarbejde med product-owner udarbejdet en prioritering blandt de vigtigste userstories. Der blev sat fokus på 2 userstories: Nr. 36 Bestil carport: Konfigureret og Nr. 92 Bestil carport: Stykliste.

Efter vores genovervejelse mener vi, at vi fik nået de mål til Userstories som var relevante. Det blev accepteret at vi bruger en userstory på at dokumentere vores arbejdsproces i en log-fil til rapporten.

Samarbejdet internt i gruppen fungerer godt, kommunikationen og professionalisme spiller en vigtig rolle, som vi varetager. Vi er blevet enige om at bruge et medlem fra gruppen som "Key-holder" - altså en mand, som holder- og skaber overblik over en specifik userstory.

Vi har klarlagt en process for disse og fundet frem til de sub-tasks, som forventes færdige i slutningen af dette sprint 2. gennemgået de to vigtigste med productowner

### **9.2.8 Scrum møde 8, Lørdag d. 25/04-20**

Til de forrige møder har sidste punkt på dagsorden: ”andet”, været fyldt af, at vi manglede en arbejdsstation til et teammedlem. Dette er løst nu og vi har planlagt hvad det kommer til at betyde for det kommende sprint med start på mandag.

### **9.2.9 Scrum møde 9, Mandag d. 27/04-20**

Der er blevet tilføjet persondata til konfiguratoren, så en enkeltperson skal indtaste nogle basale oplysninger når han laver en anmodning om et tilbud, og sælger kan kontakte ham med et tilbud efterfølgende.

I denne uge starter sprint 2, og vi har planlagt at få arbejdet på de 2 vigtigste userstories: Bestil carport: Konfigureret, og Bestil carport: Stykliste, så de kan blive færdige til næste sprint møde med product owner.

### **9.2.10 Scrum møde 9, Tirsdag d. 28/04-20**

Efter mødet i går ramte vi ind i et problem med Github og hvordan den trackede vores filer. Det tog en halv arbejdsdag at få vores arbejdsprocess gennem GH klar igen. Så det kostede os noget tid, men var en nødvendighed for at kunne arbejde hver for sig i denne corona-tid.

Der blev oprettet klasser, samt påbegyndt den konkrete kodning til styklisten. Vi har en god start og sætter 2 mand på dette arbejde. Det skal bygges op korrekt, så vi kan holde farten i arbejdet.

Sidste mand tager fat i arbejdet med DBAcces, og hvordan databasen kan tage imod et objekt. Dette skal klares snarest, så vi kan få et produkt stablet på benene.

### 9.2.11 Scrum møde 10, Onsdag d. 29/04-20

Funktionlayer og databasen er blevet koblet sammen, så der ligger en konfiguration i databasen samt at der bliver sendt et ID på denne tilbage. Der skal implementeres maks og minimumsværdier i databasen og der skal laves nogle DB-Acces metoder til materialevalgets sammenkobling også.

Strukturen på hvordan styklisten skal bygges op er blevet valgt og påbegyndt. Der skal refactors nogle ting, og når vi kører programmet får vi ikke de ting vi vil, da der mangler noget DB-acces formidling. Dette bliver afklaret snarest muligt. Der er blevet skrevet en del metoder med funktionaliteten, men mangler stadig de sidste.

Der skal i dag arbejdes på flere metoder og på forbindelsen mellem databasen og systemet(styklisten).

### Teknisk møde 1, Onsdag d. 29/04-20

Vi fik fremlagt vores systemkoncept tilfredsstillende for Arne, som ikke havde indvendinger. Vi fremlagde ydermere vores diagrammer, og fik afklaret vores spørgsmål om mock-ups.

### 9.2.12 Scrum møde 11, Torsdag d. 30/04-20

Frontend til anmodningsoversigt, og individuelle anmodningssider er klar til at få indført data på siden fra funktionaliteten i styklisten. Der er en smule refactoring-arbejde til disse sider, men der udover er det i mål.

Styklistens userstory er færdig i logikken og kodning. Der mangler test i form af indsættelse i front-end og databasen. Derefter kan vi afslutte ugens sprint. En ekstra ting er refactoring af navngivningen.

### Scrum Review møde 2, Torsdag d. 30/04-20

Vi fremlagde vores funktionalitet og arbejdsproces for productowner tilfredsstillende. Der var ikke de store pointers, og productowner ser frem til i morgen, så han kan se sammenkoblingen mellem logikken i koden til styklisten med databasen og front-end.

### 9.2.13 Scrum møde 12, Fredag d. 1/05-20

Userstories fra sprint 2, med styklisten og den byggede simple carport er færdige. Logikken og implementationen er i mål. Product owner var godt tilfreds med arbejdet og produktet. Vi fik svar på vores issues dertil, og er klar til at gribe næste sprints opgaver an.

Dagen i dag omhandler forberedelse af materiale til et planlagt møde klokken 12:00. Her afholdes sprint mødet. Efter forberedelsen står den på weekend.

## **Sprint møde 2, Fredag d. 1/05-20**

Ugens userstories er udført og gennemført. Vi er nået i mål med funktionaliteten og implementeringen for dette sprint. Der var kun det ekstra, at vi måtte refactor vores kode.

Vi fik lavet hvad vi havde planlagt, dog revurderer vi, at vi skal bide over mindre ting af gangen til fremtidige sprints. Derfor har vi omstruktureret i samarbejde med productowner, så vi for en mere flexibel og flydende arbejdsproces.

I det kommende sprint har vi planlagt processen, og prioriteret de vigtigste userstories samt subtask deri. Der bliver introduceret SVG i dette sprint. Derudover skal vi tage stilling til sortering i anmodningerne, hvorvidt det skal være på dato eller ordrenummer, men dette er en mindre problemstilling. Derfor kan vi konkludere, at vi er godt på vej med projektet.

## **9.2.14 Scrum møde 13, Mandag d. 4/05-20**

Opstart i dag står på at få en plantegning lavet, og derefter skabe logikken til implementeringen af det. Vi skal også have opdateret logikken og frontend til bestillingers status. Det kræver også en implementering af timestamp.

## **9.2.15 Scrum møde 14, Tirsdag d. 5/05-20**

Der er lavet metoder til at finde den del, som skal tegnes i plantegningen. I dag skal der laves logik til at placere dem korrekt.

Dernæst er der kommet status tekst på anmodninger, denne skal lige rettes lidt til i koden. Front-end er blevet opdateret til dette.

## **9.2.16 Scrum møde 15, Onsdag d. 6/05-20**

Metoderne til logikken om placeringen af de forskellige elementer i plantegning er næsten færdigt. Der mangler de sidste placeElement metoder, og derefter skal vi finpudse metoderne.

Der er blevet refactoret en del af koden. Userstory 62 er færdiggjort og implementeret. Der mangler noget sortering efter tid, og der er lavet et issue ift. change date cellen.



### 9.2.17 Scrum møde 16, Torsdag d. 7/05-20

De administrative værktøjer er så småt på plads i back- og frontend. Der skal ses på noget Javascript til opdateringen af specifikke felter i database-dataen, så man kan ændre dem som administrativ bruger.

Carportens visuelle implementering skal afklares i dag. Vi skal have lavet SVG-strengen til en fil og bruge den fil i et IMG tag i frontend.

### Sprint møde 3, Torsdag d. 7/05-20

Product owner var godt tilfreds efter at have set vores demo til carporten. Vi mangler at få implementeret det i front end, og dette er kriteriet for godkendelsen. Vi har aftalt med productowner, at efter implementationen af den visuelle carport i frontend, kan vi starte på sprint 4.

### 9.2.18 Scrum møde 17, Fredag d. 8/05-20

Vi har afholdt sprint møde: vi er alle godt tilfredse med samarbejdet i gruppen. Vores gruppedynamik er god og fungerer gnidningsfrit, og vi bliver færdige med de ting vi planlægger. Det næste sprint bliver planlagt og fordelt, og vi regner stærkt med at kunne blive færdige med disse userstories i denne sprint også.

Den mindre userstory med ændring af materialer kan godt blive større end 5 point. Så vi revurderer muligvis arbejdsbyrden på denne.

### 9.2.19 Scrum møde 18, Tirsdag d. 12/05-20

Vi regner med at være færdige med front-end til userstoriesne i dag. Der ud over bliver logikken til metoderne bag taget også udarbejdet. Implementeringen af Roof-klassen, giver nogle problemer i logikken, som er uforudset, men som vi finder en løsning på. Det giver muligvis en forsinkelse på vores sprint mål.

Vi debatterer en omstrukturering af hvordan kunden vælger mål for højden til tag/carport/total højde, som viser sig at give nogle udfordringer.

### 9.2.20 Scrum møde 19, Onsdag d. 13/05-20

I dag bliver der arbejdet på styklisten, som skal tage højde for skrå brædder. Der bliver også fundet en løsning til hvordan vi får fat i konfigurationerne i front-end. Der tilføjes også beklædningskomponenter i databasen.

Logikken til udregning af tagets C-linje samt alle de matematiske trekantudregninger er blevet færdig. Styklisten er også opdateret med disse metoder, samt komponent attributter. Front end er klar til disse attributter. Userstory 53 er i mål dog med småfejl klar, der bliver løst. Userstory 49 er delvist implementeret.

### 9.2.21 Scrum møde 20, Torsdag d. 14/05-20

User story 49 og 53, som handler om ændring af materialer og ændring af priser, er næsten færdig, men mangler blot nogle bugfixes.

I kodningen vedrørende taget, er der opdaget nogle små ting, som i fremtiden ville kunne forbedre projektet. Men vi anser ikke, at der vil være tid i vores sprint til at rette disse mindre fejl. Der bliver ikke taget højde for visse komponenters størrelse i forhold til hvordan de sammensættes enkelte steder. Ikke alle tagtyper passer til et fladt tag.

Der er skrevet metoder til logikken bag beklædningsbrædder og parts dertil. Der mangler rettelser til dette. Derudover skal der kobles lister til disse metoder, og sidst SVG-kodningen. Det skal indgå i rapporten, at de totale mål for karporten er uden beklædning, og at beklædning blot ligges derpå.

### 9.2.22 Scrum møde 21, Fredag d. 15/05-20

Der er arbejdet på klasserne med komponenterne og blueprint-delen angående taget, samt roof og parts. Der mangler noget bugfix til addPartBekledning metoden, som ikke giver det rigtige antal skruer på hjemmesiden. Dette bliver fikset i dag.

Der skal refactorers noget kode med lægterne, som ikke tager højde for sin egen bredde, når den måler op til næste lægte.