

Fog Carports

(I-dat-da 0222a) Datamatiker 2. semester E22

Projekt og rapport udarbejdet i perioden d. 28/11-2022 til d. 4/1-2023.

Afleveringsfrist: 4/1-2023 kl. 12:00

Antal anslag inkl. blanktegn og bilag: 86.343

Github link: https://github.com/thejamiegc/Fog_Carport.git

Demo video: <https://www.youtube.com/watch?v=BBHQcEKEkMM>

Pitch: <https://www.youtube.com/watch?v=2mXg2tpUMt8>

Digital Ocean (deployed version): http://164.90.177.110:8080/Fog_Carport-1.0-SNAPSHOT/

Demo admin bruger:

Login: Admin@admin.nu

Kodeord: 123

Demo user bruger:

Login: User@user.dk

Kodeord: 123

Udarbejdet af Fog-gruppe-C - deltagere:

Felicia Favrholt

cph-ff62@cphbusiness.dk

<https://github.com/FeliciaFavrholt>

Helena Botn Lykstoft

cph-hl237@cphbusiness.dk

<https://github.com/HelenaLykstoft>

Jamie Grønbæk Callan

cph-jc266@cphbusiness.dk

<https://github.com/thejamiegc>

Markus Isak Møgelvang

cph-mm712@cphbusiness.dk

<https://github.com/solskinIsak>

Andreas Grand

cph-ag198@cphbusiness.dk

<https://github.com/ElGrand>

1. Indledning	4
1.1. Baggrund	4
1.2. Afgrænsning	4
2. Firma/Forretningsforståelse	5
2.1 SWOT	5
2.2 VPC	6
2.3 Pitch	7
3. Teknologivalg	7
3.1 Programmer	7
3.3 Planlægning og rapportskrivning	8
3.4 Mockup	8
3.5 Video presentation	9
4. Krav til systemet	9
4.1 Firmaets vision og værdi til virksomheden	10
4.2 User stories (funktionelle krav)	10
4.3 Use-cases	13
5. Diagrammer og modeller	13
5.1. Konceptmodel	13
5.2. Domæne model	13
5.3 Aktivitetsdiagram	15
5.4 EER diagram	17
5.4.1. Overvejelser i udarbejdelsen af EER Diagrammet	20
5.5 Navigationsdiagram (state-diagram)	20
5.6 Sekvensdiagram og klassediagram	24

6. Valg af arkitektur	25
6.1. Model-View-Controller (MVC)	25
6.2 Særlige forhold	25
7. Udvalgte kodeeksempler	28
8. Status på implementering	29
8.1 Videreudvikling	30
9. Test	31
10. Proces	31
10.1. Arbejdsprocessen faktuelt	32
10.2. Arbejdsprocessen reflekteret	32
11. Bilag/Appendiks	36
11.1 User Stories	36
11.2. Model View Controller	41
11.3. Value Proposition Canvas (VPC)	42
11.4. Konceptmodellen	42
11.5. Use-cases	43
11.6. Trello	44
11.7. SWOT	46
11.8. Sekvensdiagram	47
12.0. GIT	48

1. Indledning

I vores 2. semester eksamens projekt fik vi til opgave at kode en digital løsning til virksomheden Johannes Fog A/S. Vores funktion som firma er at levere en digital løsning i form af et webbaseret system, der løser virksomhedens ønske om, at deres kunder kan bestille carporte på egne mål og materialer. Ud fra dette er hensigten, at systemet genererer et oplæg (inkl. stykliste med pris samt genererede tegninger af carporten) til en sælger hos Fog. Sælgeren kan herved kontakte kunden og rådgive nærmere omkring rabatter samt opsætning af carporten.

Målgruppen for dette er projekt er en fagfælle¹. Projektet er kodet med udgangspunkt i startkoden², som vi har fået udleveret af vores undervisere på 2. semester. Både backend³ og frontend⁴ er kodet på baggrund af udleveret materiale samt vores egne implementationer af user stories. Rapporten er udarbejdet med henblik på at dokumentere vores overvejelser og tekniske valg, som vi har truffet i tæt dialog med en virksomhedsrepræsentant.

1.1. Baggrund

Kunden er Johannes Fog A/S, et online byggemarked⁵, der sælger forskellige services/produkter og har en specifik afdeling, som håndterer bestillinger og salg af standard, samt skræddersyet carporte. Fog har som virksomhed positioneret sig som et byggemarked, der leverer carporte med en stor valgfrihed for kunden i forhold til valg af dimensioner samt materialer⁶. Deres vision for afdelingen med salg af carporte er, at være konkurrencedygtige på markedet, yde en direkte service i form af kompetent vejledning samt prisgennemsigtighed. Kundens krav til systemet er uddybet i punkt 4: Krav til systemet.

1.2. Afgrænsning

For at kunne sørge for størst mulig præcision og skarphed i vores beregninger, tegninger og projekt var vi nødsaget til at foretage os nogle afgrænsninger - med udgangspunkt i det udleveret materiale, heriblandt video interviewet med kunden.

¹ Fagfælle: En datamatiker studerende på 2. semester der er på samme niveau, men som ikke kender opgaven.

² Startkode 2. sem: https://github.com/jonbertelsen/startcode_2sem_2022

³ Backend: Den grundlæggende aktivitet - på serverniveau

⁴ Frontend: Brugergrænseflade orienteret aktivitet

⁵ Kunden: Johannes Fog A/S - hjemmeside: <https://www.johannesfog.dk/>

⁶ Internt materiale (moodle) via projektoplægget: Interview (video) med afdelingsleder Martin hos Fog byggemarked - 'Flow 5 - Fog byggemarked - kundemøde nr. 1'.

Vi har taget højde for, at kunden kun kan modtage faste længder på træ, og at samme kunde besidder egenskaben til at tilskære deres materialer til det givne projekt/produkt. Herudover tager vi højde for, at de priser (pris/cm + pris/stk) der er fastsat, er udregnet ud fra Fog's egne meter priser, der fremgår af deres hjemmeside. Højden på en carport er fastsat til en fast højde på 300 centimeter - uanset type. Vi har valgt, at antal stolper beregnes ud fra det kunden har udtalt sig om i video-interviewet; derfor beregnes de på carporten størrelse målt i kvadratmeter:

$$0 \geq 25 \text{ m}^2 = 4 \text{ stolper}$$

$$25 \geq 35 \text{ m}^2 = 6 \text{ stolper}$$

$$35 \geq 45 \text{ m}^2 = 8 \text{ stolper}$$

$$45 \geq 55 \text{ m}^2 = 10 \text{ stolper}$$

Herudover tager vi udgangspunkt i, at Fog ikke allerede har et IT system, der leverer samme løsning som os, og at den løsning vi har udviklet er et konkurrerende system, til det de opererer med på nuværende tidspunkt. Vores system har øget fokus på brugervenlighed for både kunde, sælger og det aspekt, at vi samler forskellige funktioner fra deres eksisterende programmer i en pakkeløsning. Vores IT løsning skal ligeledes ikke ses som en endelig version, og derfor er der plads til videreudvikling med input fra kundens side.

2. Firma/Forretningsforståelse

Vi som firma, har gjort os en del tanker og har taget en række beslutninger i forhold til vores arbejdsform. Ud fra dette, har vi analyseret os på vores arbejdskompetencer som gruppe ved brug af SWOT-analyse. Ydermere har vi opnået en forretningsforståelse af kunden ved at udarbejde en VPC. Vi har lavet en video pitch der har til formål at sælge vores værdi til kunden.

2.1 SWOT

SWOT-analysen (*bilag 11.7*) giver os et samlet overblik over vores styrker, svagheder, muligheder og trusler. Overordnet har vi bedømt, at vi er gode til gruppearbejde, på baggrund af vores forskellige erfaringer ift. kodning, GitHub og strukturerisering. Vi har en smule eksamensangst i forhold til, om vi når deadlines, men det retter vi op på ved at være strukturerede og mødestabile og afholde daglige scrum møder. Vi har ikke så meget erfaring i forhold til Git og merge konflikter, men det vil vi forbedre ved at lave branches og fastholde

enighed om coding standards. En anden styrke er, at vi har erfaring med startkoden - dette betyder, at vi ikke skal starte fra scratch.

En stor trussel i vores gruppe er, at vi desværre har en langtids sygdomsramt gruppemedlem som ikke kan være her fysisk. Dette løser vi dog ved at uddelegerere opgaver iblandt os, så vi sikrer både at arbejde online og fysisk så alle kan deltage i arbejdsprocessen. Vi ved, at hackerangreb er en mulig trussel ift. at have vores data på ekstern droplet, men vi opererer med et sikkerhedslag på bruger og password, så det ikke let kan tilgås. Vi har mulighed for, at opnå ny læring og iterere vores process undervejs ud fra den nye viden vi opnår i workshopsene, SVG og Digital Ocean. Det giver os mere motivation, at vide at vi har mere, vi kan lære og koble til vores projekt.

2.2 VPC

I vores VPC model⁷ har vi forsøgt at få et bredt overblik over, hvordan FOG tilfredsstiller deres kunder, og hvordan FOG skaber en kunderelation ud fra deres servicer. Der er derfor ikke taget udgangspunkt i en bestemt vinkel ved analysen, men mere et bredt spektrum, således vi kunne brainstorme frit om alt og ingenting. VPC modellen er udarbejdet ud fra tildelt materiale, der gennemgår et besøg hos kunden⁸ samt vores egen gennemgang/analyse af deres eksisterende hjemmeside og chat dialog med en Fog ansat.

Udarbejdelsen af vores VPC danner grundlag for, hvilken værdi vi som firma vil tildele kunden, ud fra den IT løsning vi levere til dem. Den er udarbejdet før implementeringen af koden og viser sammenhængen mellem kundesegmentets ønsker/forventninger til virksomheden (højre side), og hvad virksomheden kan tilbyde/levere af services for at tilfredsstille kundesegmentet (venstre side). Der er et ‘fit’ mellem de to sider, når der tydeligt er en sammenhæng herimellem.

Eksempler på dette fremgår af vores model (*bilag 11.3*). Nogle af de fokuspunkter vi har valgt at arbejde videre med, er den stærke brand identitet, som FOG har skabt igennem deres mange år på markedet. FOG appellerer til et smalt segment med et bredt sortiment og har primært fokus på erhvervkunder men sælger også til privatpersoner. Vi har især lagt vægt på

⁷ Value Proposition Canvas (VPC) kan ses via link:

<https://jamboard.google.com/d/1KRACqJxDloN94YRb0Zjb6bqIFdV56CDh0VzGH2u5hoE/viewer?f=1>

⁸ Internt materiale (moodle) via projektoplægget: Interview (video) med afdelingsleder Martin hos Fog byggemarked - ‘Flow 5 - Fog byggemarked - kundemøde nr. 1’.

ikke at afvige fra deres nuværende løsninger og har undersøgt nærmere, hvilke services de tilbyder kunden samt samarbejde med Fog Fonden og eksterne håndværkere.

Vi har tænkt os, at det skulle være nemt for en privatperson at navigere rundt på Fog's hjemmeside, kunne logge ind og bestille en carport - deraf fokus på den service kunden opnår og optimeringen af den professionelle rådgivning, sælgeren skal kunne videregive, og den rådgivning som kunden gerne vil modtage. Vi ville gerne tænke udover den fysiske butik, idet den typiske kunde her vil være erhvervkunden, der allerede har kendskab til sortimentet.

Ydermere vil Fog gerne holde deres omkostninger nede og have øget fokus på deres lagerstyring. Derfor har vi i vores IT løsning udarbejdet en materialeliste, der kan redigeres i. I forhold til vores pitch har der været et bredere fokus på fremtidsmulighederne for Fog som virksomhed, i forhold til de løsninger de kan leverer ud fra et bæredygtigt perspektiv men også med den teknologiske udvikling i mente.

2.3 Pitch

Som firma vil vi gerne kunne sælge vores IT løsning og hermed skabe en øget værdi for virksomheden Johannes Fog A/S, såvel som deres nye og eksisterende kunder. Vi startede med at udarbejde nogle tekstbaserede pitches til både kunden Fog og også deres kunder⁹. Vi har valgt at have fokus på at sælge værdi til Fog's afdeling der rådgiver om salg af carporte.

Link til pitch: <https://www.youtube.com/watch?v=2mXg2tpUMt8>

3. Teknologivalg

I forbindelse med udarbejdelsen af vores projekt har vi benyttet os af nogle forskellige programmer og teknologi valg. Disse er listet herunder:

3.1 Programmer

- Tomcat 9.0.67 (Tomcat Webcontainer / server)
- Digital Ocean: Droplet. IP: 164.90.177.110
- OS - Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-122-generic x86_64)
- Google Chrome version 108.0.5359.124 (64-bit)
- GIT - Git bash version 2.38.0

IntelliJ:

- IntelliJ IDEA 2021.2.4 (Ultimate Edition)

⁹ <https://docs.google.com/document/d/1tUGHkAcCcDuDzVlr0KDYGeGHVwzeyQQpgv3-0GafueQ/edit>

- Corretto-11 Amazon Corretto version 11.0.16. (Java SDK)
- PlantUML Integration version 5.19 (plugin)
- Sequence diagram version 2.2.6 (plugin)
- Maven version: 3.6.3

Database:

- Mysql-connector 8.0.30 (Java database connector)
- Mysql workbench 8.0 CE (database)
- Mysql server og biblioteker

3.2 Programmeringssprog

- HTML 5
- CSS
- Twitter Bootstrap v5.1.3
- Java Version 11.0.16
- HTTP Protocol
- JSTL javax.servlet v. 1.2

3.3 Planlægning og rapportskrivning

- Google Docs - Anvendt til at skrive dokumenter, tage noter mm.
- Trello - Projektstyringsværktøj der giver overblik over arbejdsprocessen - link:
<https://trello.com/b/mpKTWIp1/fog-carports-eksamensprojekt-gruppe-c> (version 1)
- Logbog - Anvendt til grundigere projektstyring på daglig basis samt noter.
- Draw.io - Anvendt til udarbejdelse af forskellige diagrammer.
- Github - projektet tilgås via link: https://github.com/thejamiegc/Fog_Carport.git

Dette er version 1 af main branch.

3.4 Mockup

Vi har anvendt Figma til at brainstorme en tidlig mockup af webshoppen version 1 (v1)¹⁰, og så har vi udarbejdet en mockup version 2 (v2)¹¹, der repræsenterer en prototype af den implementerede kode, der udgør vores produkt.

¹⁰ Figma Mockup - Version 1:

<https://www.figma.com/file/z4Au9OMd5WB13zD9zFEqcE/Fog-Carports?node-id=0%3A1>

¹¹ Figma Mockup - Version 2:

<https://www.figma.com/file/z4Au9OMd5WB13zD9zFEqcE/Fog-Carports?node-id=128%3A193&t=cvaj7D04AjCosRVM-0>

Vi har tiltænkt, at vores IT løsning skal anses som et udsnit af det komplette website. Forstået på den måde, at løsningen ville kunne inkorporeres i deres nuværende design. Dette betyder, at vores index side ville skulle kombineres med den eksisterende index side¹² således, at der er en login funktion mere i navigationsbaren.

Interaktionen imellem de forskellige knapper og sider kan ses her:

<https://www.figma.com/proto/z4Au9OMd5WBl3zD9zFEqcE/Fog-Carports?node-id=137%3A193&scaling=min-zoom&page-id=128%3A193&starting-point-node-id=137%3A193>

Målet med vores mockup var at udarbejde et design, der lå så tæt på Fog's eget UI design på hjemmesiden¹³. Derfor har vi i version 1 (v1) taget screenshots af deres egen hjemmeside, og anvendt det som et moodboard og en slags inspiration for vores layout. Navigationsbaren er den primære interaktion for brugeren på hjemmesiden til at navigere rundt på de forskellige jsp sider, og derfor har vi beholdt dette layout. Vi har vægtet det funktionelle aspekt rent kodemæssigt, men har dog forsøgt at bestræbe os på at levere et produkt, der også er æstetisk og brugervenligt ud fra et UI/UX perspektiv.

3.5 Video presentation

Vi har anvendt Screencast-O-Matic til at optage en gennemgang af vores færdige projekt, der viser to løsninger.

- Kundens brugeroplevelse - bestilling af en carport.
- Sælgerens brugeroplevelse - håndtering af bestillinger samt oprettelse af ordre.

Videoen kan tilgås via link: <https://www.youtube.com/watch?v=BBHQcEKEkMM>

4. Krav til systemet

Kundens krav til systemet fremgår her:

- Systemet skal kunne generere en produkttegning samt en stykliste.
- Systemet skal kunne gemme data i en database.
- Systemet skal kunne beregne priser på materialer ud fra brugerens ønsker på mål.
- Systemet skal udvikles med henblik på optimering af den visuelle brugeroplevelse for kunden.

¹² <https://www.johannesfog.dk/>

¹³ <https://www.johannesfog.dk/>

Bruger:

- Som bruger¹⁴ skal man kunne oprette en bestilling på en carport ud fra brugerdefinerbare mål og trætype.
- Som bruger skal man kunne se en tegning af den carport man har lagt til bestilling.

Admin:

- Som admin¹⁵ skal man kunne se den bestilling, som kunden har lagt.
- Som admin skal man kunne genererer en ordre til kunden.
- Som admin skal man kunne annullere og redigere eksisterende kunde ordre.

4.1 Firmaets vision og værdi til virksomheden

Vores vision som firma er at løse Fog's ønske om en optimering af en del af deres hjemmeside, der opererer en afdeling, der sælger carporte. Her kan kunder bestille en skræddersyet carport. Herunder skal kunden føle, at det er nemt at logge ind, oprette en bruger samt bestille en carport på baggrund af deres ønsker. Herudover skal det være nemt for kunden at se sine nuværende og gamle ordrer inkl. ordrestatus.

Det skal desuden også være nemt for en admin at kunne se bestillinger, og overføre dem til en ordre. Derudover skal admin også kunne ændre kundens status og ændre i ordrerne, hvis nu kunden har lavet forkerte mål eller de sammen til et møde har fundet ændringer. Admin skal også have mulighed for at se alle informationerne i en ordre, og se den fulde materialeliste og kunne opdatere/slette informationer i denne.

Den værdi vi tilføjer til virksomheden, er en mere optimeret version af deres carport afdeling. Vi vil have fokus på funktionalitet frem for design.

4.2 User stories (funktionelle krav)

Efter et kundemøde med virksomheden har vi fået udarbejdet en række user stories, der kort beskriver hvilke brugere, som har hvilke behov, samt hvad de ønsker at opnå. Alle vores user stories er angivet i *bilag 11.1*. Hver user story har fået et estimat, der indikerer størrelsen på implementeringen af den dertilhørende kode. Disse estimerater har gjort, at vi har kunne

¹⁴ Bruger skal forstås som en besøgende på hjemmesiden.

¹⁵ Admin skal forstås som en medarbejder/sælger i Johannes Fog A/S

planlægge rækkefølge og tidsestimat på implementering af hver user story. De er baseret på, at man er logget ind, eller har oprettet sig som bruger, og at man enten har en ‘user’-rolle eller en ‘admin’-rolle. Et udpluk af disse beskrives nærmere her:

US-3: Lav forespørgsel for carport - Estimat M

Som <bruger> ønsker jeg at <kunne logge ind på en brugerside>, sådan at jeg <kan foretage en forespørgsel på et carport design>

En af de beslutningsprocesser, vi havde oppe til dialog, var, hvordan vi skulle oprette en kundes forespørgsel på en carport. Vi havde i starten tænkt, at en forespørgsel skulle indeholde en tom carport, og herefter kunne man så tildele data til carporten. Vi var alle enige om, at det så skulle være sælgeren, der kunne omdanne denne forespørgsel til en ordre og dermed, på en eller anden måde, kunne sætte en status på orden. For at brugeren kunne oprette en carport, tænkte vi, at indsætte en ‘opret bestilling’ knap ind i designet.

US-4: Se forespørgsel eller ordre - Estimat M

Som <bruger> ønsker jeg at <kunne logge ind på en brugerside>, sådan at jeg <kan se mine forespørgsler eller ordrer>

Vi havde tiltænkt, at måden man skulle kunne se sine forespørgsler eller ordrer, ville være på en ordre side, hvor al nyttig information står. Siden ville være anderledes i forhold til, om man er en bruger eller en admin, da det som admin er mere nyttigt at se navnet på den bruger, der har oprettet forespørgslen eller ordren i forhold til, at det som kunde ikke er nyttigt at se sit eget navn, men at se andre relevante informationer om forespørgslen/ordren.

US-4a: Se og følge status på ordre - Estimat S

Som <bruger> ønsker jeg, at <kunne logge ind på en brugerside> sådan, at jeg <kan se og følge status på mine ordre>

	statusID	statusname
1	1	Oprettet
2	2	I behandling
3	3	Afventer kundesvar
4	4	Ordre på vej
5	5	Afsluttet

Vi ville gerne skelne mellem om det er en forespørgsel eller en ordre, så derfor har vi valgt at implementere forskellige typer af status:

- *Oprettet*: Når en kunde opretter en bestilling, vil statussen være sat til status nummer 1.
- *I behandling*: Herefter vil en admin-bruger kunne gå ind og se bestillingerne, og så snart admin-brugeren trykker på 'vis detaljer' vil statussen blive sat til 'i behandling'.
- *Afventer kundesvar*: Derefter vil en admin-bruger kunne gå ind og trykke 'opret ordre', og her bliver statussen sat til 'afventer kundesvar'.
- *Ordre på vej*: Herefter kan en kunde gå ind og acceptere tilbuddet under 'mine ordre', og så vil ordren have statussen 'ordre på vej'.
- *Afsluttet*: Den sidste status har vi ikke implementeret, da vi ikke kan tjekke for, om kunden har modtaget ordren.

Vi har valgt at implementere en status, så det gør det nemmere for brugeren at se hvor langt han er i processen i at få en carport. Grunden til, at vi har udarbejdet de forskellige statusser, er at der nemt skiftes imellem statusserne på automatisk vis afhængigt af, om man som bruger eller admin har interageret med ordren. Ulempen ved at have et automatisk status-system er dog, at man ikke kan gå tilbage til en lavere ranket status. Vi tænker dog, at det ikke skal være muligt at lave en ordre tilbage til en bestilling, da man derfor ville skulle ringe til selve lageret og sige, at de ikke skal begynde at pakke ordren.

US-6c: Se tegning - Estimat M

Som <admin> ønsker jeg at <kunne logge ind på en admin side>, sådan at jeg <kan se tegninger over carportene>

For at kunne vise tegninger af en carport, besluttede vi, at implementere en side hvor man kan se ordre detaljerne. Dette sker, når man trykker på 'vis detaljer' knappen. Vi har udarbejdet to forskellige tegninger: Én snittegning og én plantegning - sådan så kunden både kan se en model fra siden og fra toppen.

4.3 Use-cases

Som supplement til vores user stories har vi udarbejdet nogle use-cases, der viser interaktion for hhv. bruger (kunde) og admin (sælger) på hjemmesiden. Vi har også lavet en specifik use-case, for hvordan brugeren bestiller en carport, og hvordan admin opretter en ordre. Se tegninger over dette via *bilag 11.5*.

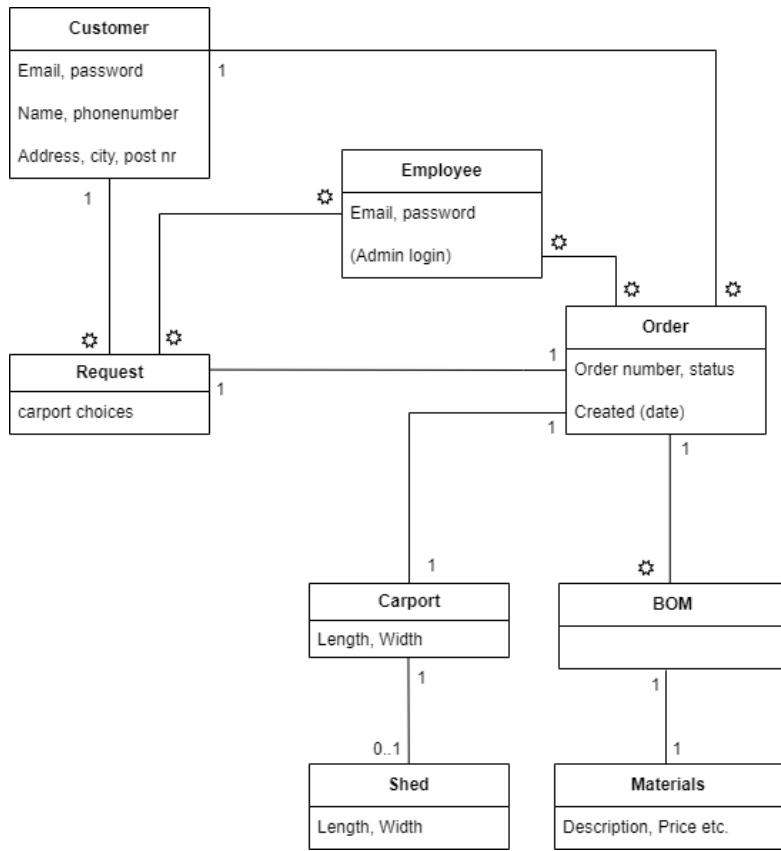
5. Diagrammer og modeller

5.1. Konceptmodel

Konceptmodellen (*bilag 11.4*) viser et overblik og en sammenhæng over, hvilke interaktioner der er imellem kunden og Fog. Modellen tager udgangspunkt i en kundes bestilling/køb af en carport og viser hele processen fra bestilling til salg og transport af materialer til, at kunden får leveret et produkt og kan tage det i brug. I modellen ses det tydeligt, at Fog opererer med en direkte kommunikationsmodel, hvor sælgeren er den primære aktør, der holder salgsprocessen i flow, og kunden inddrages på to måder; kunden skal ned i den fysiske butik for at aflægge en bestilling eller kan tilgå hjemmesiden og aflægge en bestilling. Sælgerens ekspertise er essentiel for et godt salg, og derfor inddrages kunden på sælgerens præmisser - der er ikke så meget "vælg selv" funktion over deres nuværende salgsflow, og derfor har vi valgt at have fokus på, at kunden skal kunne se sine ordre og følge status herpå.

5.2. Domæne model

Nedenstående model viser adfærd og data af et domæne på et konceptuelt niveau. Med dette, skal der forstås, at ikke-tekniske interesser skal kunne aflæse modellen og stadig kunne se en sammenhæng i, hvordan den systematiske data skal udformes. Modellen identifierer de konceptuelle klasser og skaber et overblik over kodens opbygning. Som det kan ses i modellen, har vi bestemt nogle attributter samt defineret nogle associationer og fastlagt nogle relationer mellem klasserne. Diagrammet læses fra venstre mod højre og nedad.



Ved udarbejdelsen af modellen gjorde vi os mange tanker om, hvilket data der skulle være tilgængeligt; *Hvordan gemmer vi en ordre? Hvad en ordre består af?*

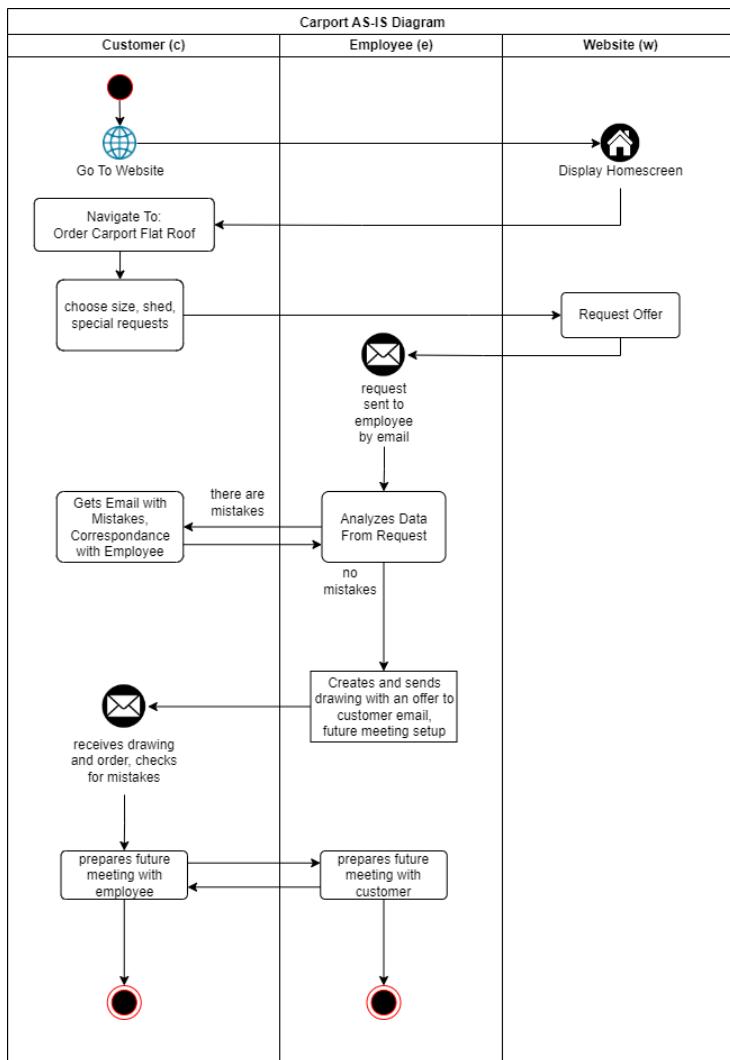
Vi har derfor opstillet relationerne således:

- En *customer* kan sende mange *requests*.
- En *request* er et ønske om en *carport*.
- En *request* bliver til en *order* - der indeholder data fra *request*.
- Mange *employees* kan tilgå mange *requests*.
- Mange *employees* opretter flere *orders* ud fra de *requests*, der er oprettet i systemet af en *customer*.
- En *order* består af en *carport*, et *shed* og *BOM* (Bill Of Materials).
- *Shed* er implicit i oprettelsen af en *carport* og er derfor tilknyttet en 0..1 relation.
- En *BOM* består af *materials* der har en titel, beskrivelse, pris mm.

5.3 Aktivitetsdiagram

Aktivitetsdiagrammet er en oversigt over, det overordnede workflow. Vi har udarbejdet to diagrammer; AS-IS¹⁶ og TO-BE¹⁷. Diagrammerne følger en kunde, der bestiller en carport via Fog's hjemmeside¹⁸.

AS-IS:



Først går kunden ind på hjemmesiden, som viser forsiden (index.jsp). Derefter nавигerer kunden til 'order carport' siden, og bliver sendt videre til en side, hvor der kan vælges mål og materialer på en carport. Ud fra dette bliver der dannet en forespørgsel, som bliver videresendt til hjemmesiden. Herefter modtager en Fog medarbejder en e-mail, der

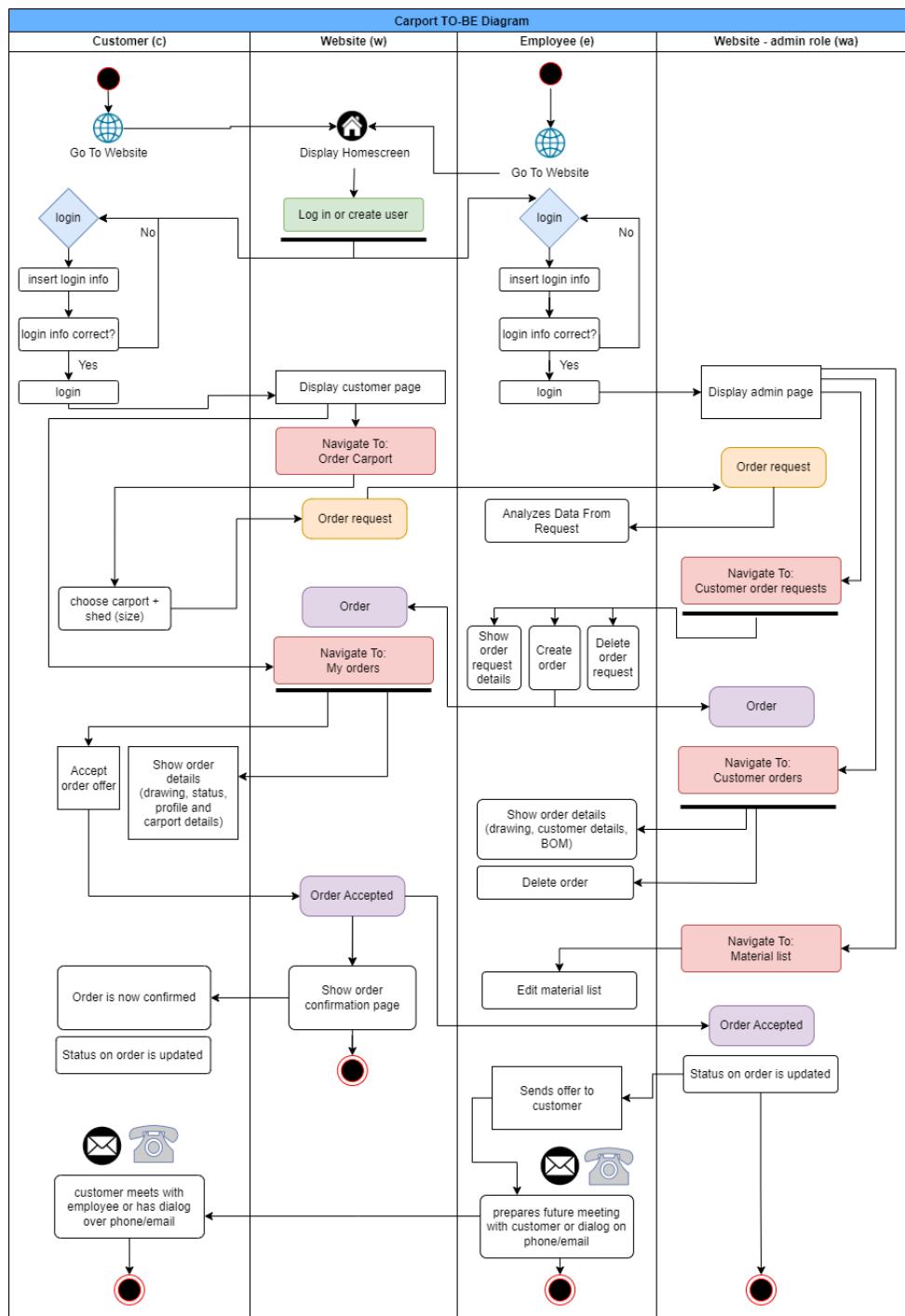
¹⁶ AS-IS diagram: Diagram over virksomhedens nuværende workflow

¹⁷ TO-BE diagram: Diagram over virksomhedens workflow efter vi har leveret vores system

¹⁸ <https://www.johannesfog.dk/>

indeholder data fra denne forespørgsel. Dvs. data omdannet til at udgøre informationer i en ordre. Hvis der er fejl i ordren, eller noget Fog medarbejderen skal diskutere med kunden, så foregår dette via e-mail korrespondance. Medarbejderen udarbejder ligeledes en tegning af carporten, og sender den så til kunden sammen med en anmodning om et kundemøde i fremtiden. Kunden modtager sin ordre og tjekker den for eventuelle fejl eller forkerte ændringer. Derefter forbereder begge parter sig til kundemødet.

TO BE:



Som kunde (customer):

Kunden starter med at navigere til hjemmesiden, som viser forsiden (index.jsp). Derefter trykker kunden på log ind knappen i navigationsbaren, og bliver videresendt til en login side. Her tjekkes, om der anvendes et korrekt login. Hvis login er ok, bliver kunden sendt til en kundeside, og hvis ikke, rammer man en fejl side. Givet at informationerne er korrekte, så kan kunden navigere til 'Byg en carport' siden eller 'Mine ordrer' siden. Ved klik på 'Byg en carport' i navigationsbaren, kan kunden selv vælge mål og materialer på sin carport. Derefter kan han se sine ordre ved at navigere til 'Mine ordre' siden.

Som sælger (employee):

Hvis en sælger vil gå ind og se bestillinger og ordre på carporte, der er lavet, skal sælger også logge ind, som beskrevet ovenfor. Derefter kan sælgeren navigere til 'kunde bestillinger', 'kunde ordrer' eller 'materialeliste'-siden. Hvis sælger vil ind og se en bestilling, skal der klikkes på 'kunde bestillinger' i navigations baren. Her kan sælger se alle kunders bestillinger, og herfra kan en bestilling omdannes til en ordre, ved at sælger klikker på 'lav ordre' knappen. Ordren vil nu kunne vises på 'kunde ordrer'-siden. Hvis kunden går ind og kigger på sin ordre og trykker på 'accepter'-knappen, så kan sælger se, at statussen har ændret sig fra at være 'afventer kundesvar' til at være 'ordre på vej'. Hvis en admin vil se materialelisten, kan han klikke på 'materialeliste' i navigationsbaren. Herinde kan sælger redigere i materialerne, hvor sælger så kan give kunden andre længder, beskrivelser, priser mm.

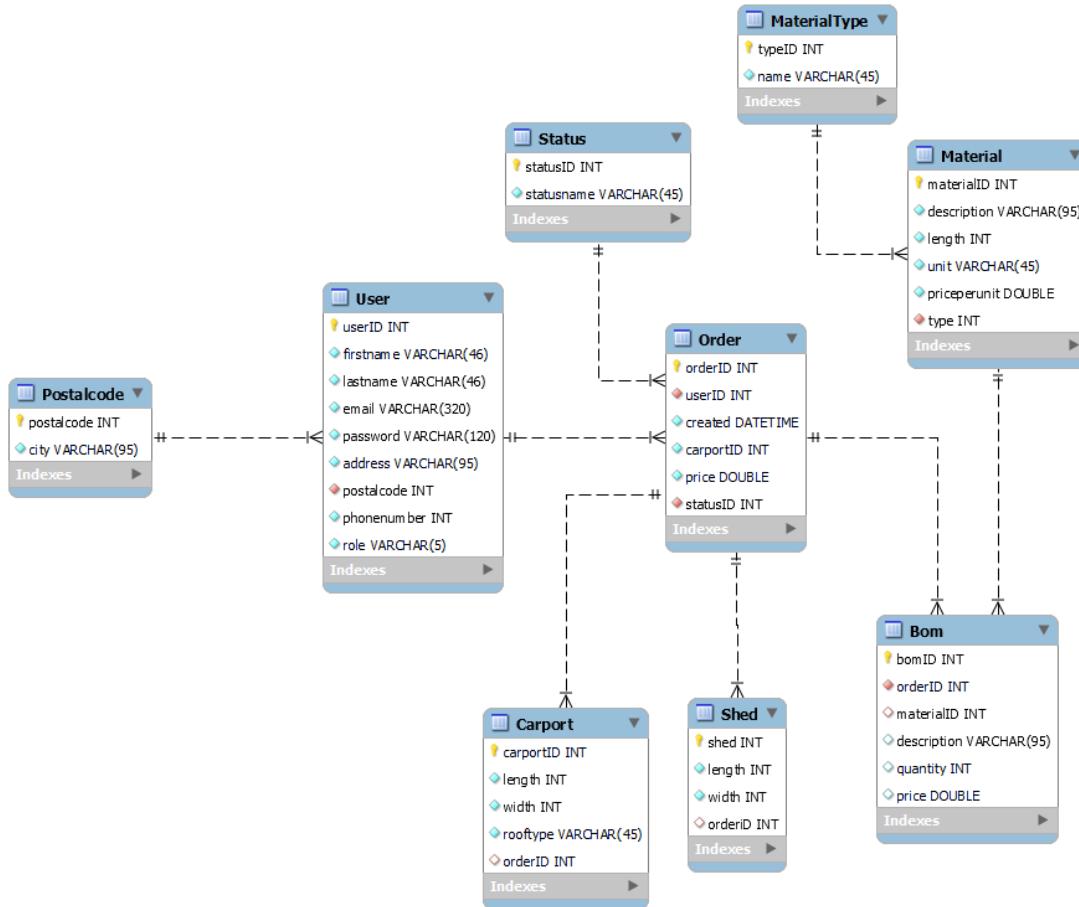
5.4 EER diagram

Vi har udarbejdet vores database med udgangspunkt i udleveret materiale og domænemodellen. EER diagrammet viser en oversigt over, hvordan vi har valgt at gemme data i en database. Denne data bliver behandlet igennem CRUD funktioner i vores mappers i java koden, når man kører programmet. Diagrammet består af en række tabeller, som vi har navngivet ud fra de konceptuelle klasser, der fremgår i domænemodellen. Tabellerne skal opfylde de forskellige normalformer ud fra en række kriterier:

- 1. normalform: Første normalform har ingen tomme celler, ingen gentagende felter eller kolonner, og de har alle har en tydelig angivet primære nøgle, som består af autogenereret heltal. Denne data er skjult fra brugeren.

- 2. normalform: Første normalform er opfyldt, og ingen attributter, der ikke selv tilhører nøglen, må afhænge af en del af nøglen.
- 3. normalform: Første samt anden normalform er opfyldt og ingen attributter må afhænge af andre attributter, der ikke selv er nøgler.

Tabellerne opfylder 1. normalform, idet de alle har en entydig nøgle. I praksis er dette sikret ved, at lade primærnøglen bestå af et autogenereret heltal, og da der ikke fremgår nogen delte primærnøgler, opfylder tabellerne ligeledes 2. normalform. Vi har indsat tabeller, der knytter et id som nøgle (INT) sammen med et variabelnavn (VARCHAR). Dette fremgår af tabellerne 'MaterialType', 'Postalcode' og 'Status', og sikrer at vi opfylder 3. normalform.



Model: Viser EER-diagram som er implementeret ved afleveringstidspunktet. Det er disse tabeller, som bliver behandlet data i, via mappers i java-koden, når programmet kører (v7).

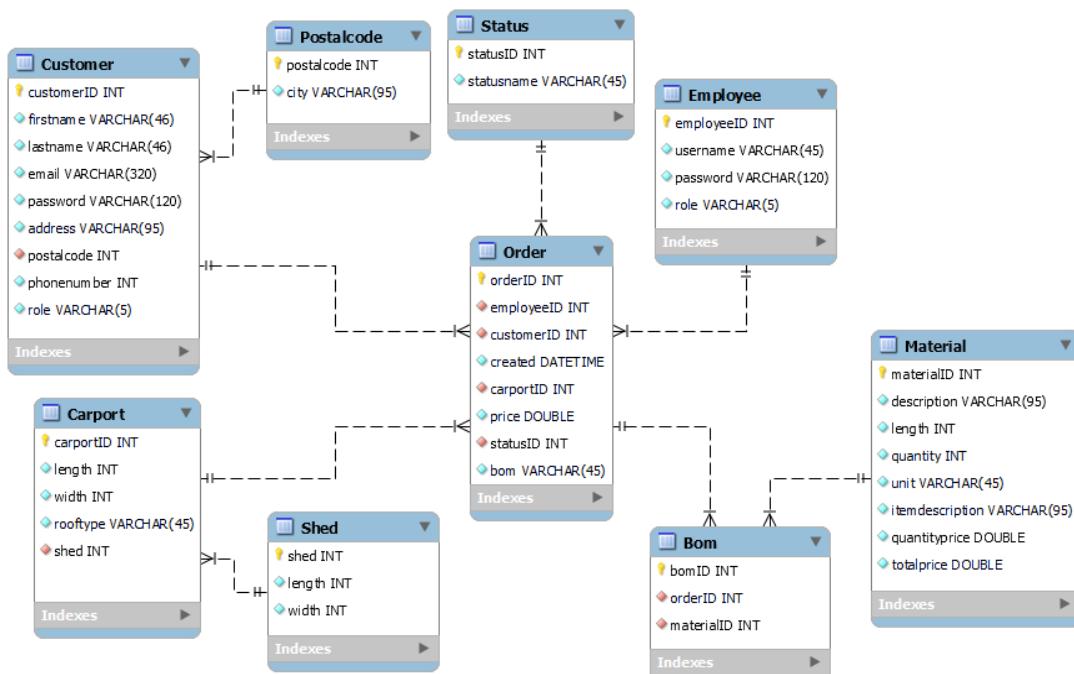
EER diagrammet viser 1-mange relationer mellem tabellerne, og især her er der fokus på sikring af dataintegritet, hvor tabellerne er knyttet sammen gennem deres primære nøgler, som herved anvendes som fremmednøgler i de tabeller, de er knyttet til. I praksis betyder

dette, at der ikke kan sættes en værdi ind på fremmed nøglenes plads, hvis ikke der allerede er oprettet den samme værdi i de tilknyttede tabeller, hvor primære nøglerne fremgår.

Et eksempel på dette ville være, hvis vi forsøgte at oprette en bruger i tabellen ‘User’ med datafeltet ‘postalcode’, så ville den give fejl, idet vi først skal angive data i tabellen ‘Postalcode’. Et andet eksempel på dette er, at i ‘Carport’ tabellen har vi et id ‘carportID’, som agerer som primærnøgle, og dette id er knyttet til ‘Order’ tabellen via ‘carportID’, der her er en fremmednøgle. Relationen mellem de to tabeller er 1-1, da der kun kan være 1 carport i 1 ordre.

Vi har ydermere valgt, at adressefeltet, ‘address’ både indeholder vejnavn samt tal, og benytter derfor dette felt, som et variabelnavn. Man ville dog nok argumentere, for at disse datafelte også ville være relevante at dele op til andre typer projekter, men vi har valgt at benytte os af denne løsning, da vi kun har haft brug for at kunne oprette et par brugerprofiler til at teste koden.

Tabellen ‘Carport’ indeholder den data, der bliver indtastet af brugeren, når brugeren skal oprette en bestilling. Dette betyder, at når man vælger sine mål/dimensioner, så bliver dataen gemt i tabellen under ‘length’, ‘width’, ‘rooftype’. Tabellen afspejler den konceptuelle klasse ‘request’, som fremgår i domænemodellen.



Model: Viser EER-diagram som er udarbejdet ved opgavens udlevering (v1).

5.4.1. Overvejelser i udarbejdelsen af EER Diagrammet

Vi har itereret i vores process af udarbejdelsen af EER diagrammet mange gange. Vores primære idé var, at vi ville have en ‘Customer’ tabel og en ‘Employee’ tabel, da en Employee ikke nødvendigvis skulle have et fornavn, efternavn, adresse m.m. Det kom vi dog hurtigt fra, da vi kunne se, at det var nemmere at skelne mellem en kunde og en medarbejder på baggrund af rollen. Og vi tænkte også, at man skulle have mulighed for, at se alt information om en medarbejder inde i databasen.

Den største forskel mellem EER diagram v1 (version 1) og v7 (version 7) er, at i vores v1 har en ‘Order’ tabel som har både employeeID og customerID, men da vi slettede vores employee tabel, har vi måtte ændre tabellen således, at en en ordre har et userID.

Vi fandt også meget tidligt ud af, at hvis vi prøvede at slette alt information fra en tabel, så kom der meget tit fejl, fordi vi ikke kunne slette fra en tabel, som havde en ‘foreign key’ fra en anden tabel. Relationerne mellem vores ‘Order’ tabel, ‘Carport’ tabel og ‘Bom’ tabel er derfor bygget op, sådan at de alle har et ‘order id’, hvorved vi kan knytte dataen fra de forskellige tabeller til en ordre.

Derudover gjorde vi det nemmere, for os selv ved, at når en bruger bliver oprettet, vil brugeren automatisk få tildelt rollen ‘user’, og kan derved komme ind på vores bruger-sider. Brugeren får også automatisk en carport og et skur, når brugeren opretter en ordre, hvilket gør, at dataen senere hen nemt kan blive opdateret ud fra de valg, som brugeren træffer.

Vores ‘Bom’ tabel har også ændret sig meget, da vi midt i projektet fandt ud af, at ‘Bom’ tabellen lidt skulle fungere ligesom ordre linjer, så der kommer til at ligge en masse linjer inde i tabellen. For at få vist styklisten for en specifik bruger og en specifik carport, forbinder vi ‘Bom’ tabellen på ‘Order’ tabellen med ‘orderId’ og ‘Material’ tabellen på ‘materialID’ . Ved at forbinde ‘Material’ tabellen kunne vi ligeledes vise en beskrivelse, antallet samt prisen for det specifikke materiale.

Alle vores overvejelser omkring vores database har været udarbejdet lidt efter lidt, så vi bedst muligt har kunnet lave projektet, så det giver mening.

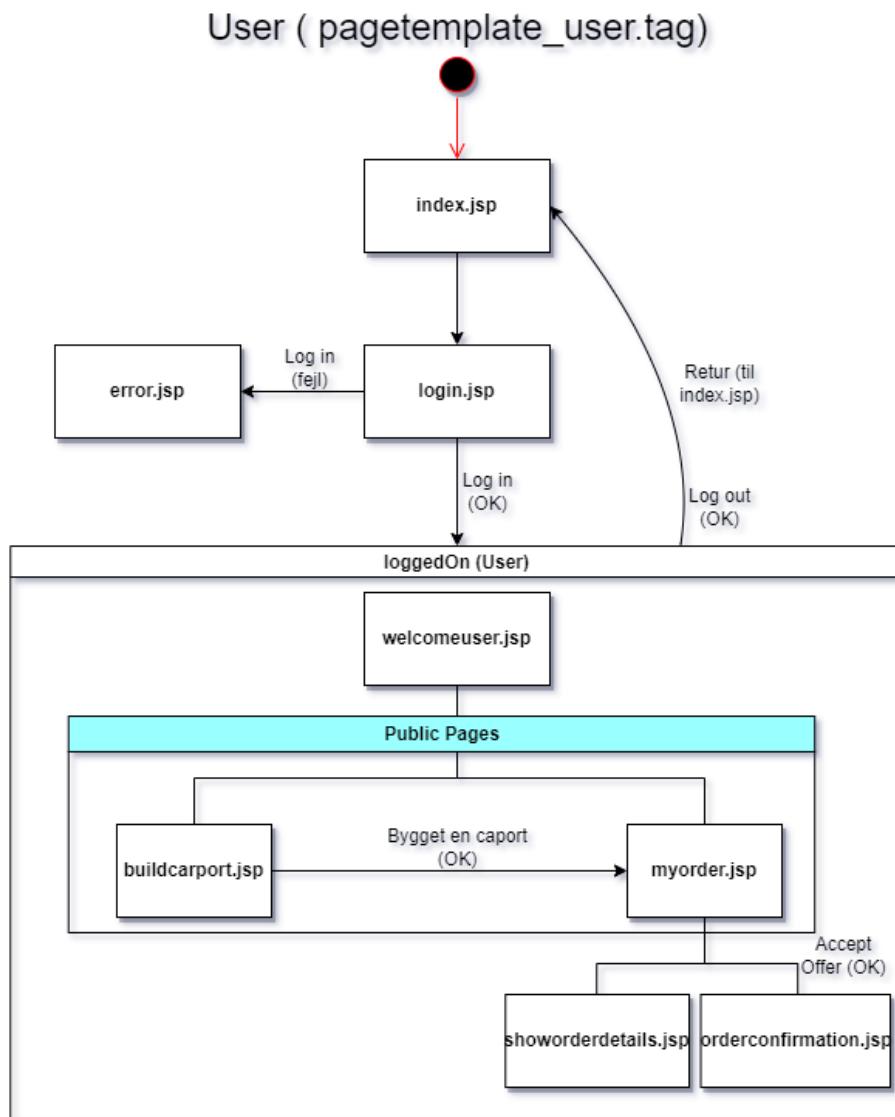
5.5 Navigationsdiagram (state-diagram)

Vores navigationsdiagram er delt op i to dele: admin og user. Diagrammerne starter med index siden, og derefter logger man enten ind eller laver en bruger. Både admin- og

user-diagrammet viser, at man kan trykke på forskellige sider i navigationsbaren, efter man er logget ind.

User navigation

Navigationsdiagram [2] - User Navigation (Opdateret)



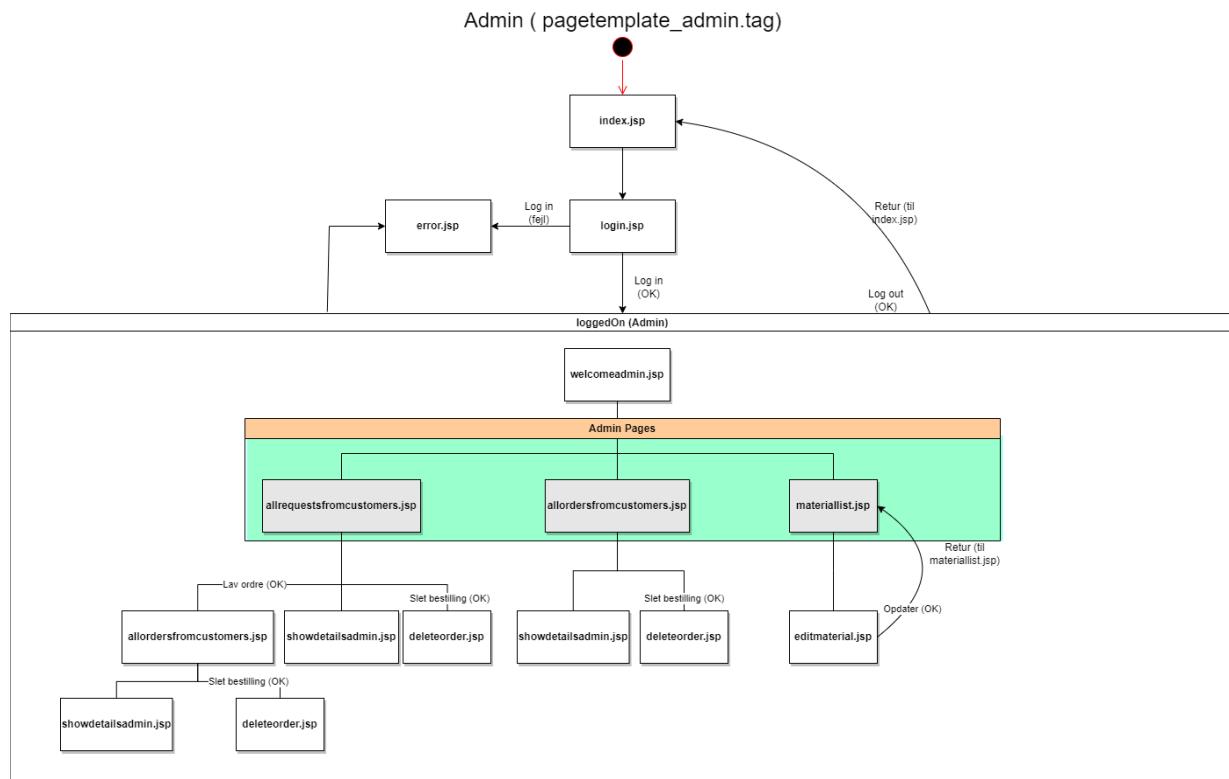
Brugeroplevelsen på hjemmesiden bliver her vist gennem rollen 'user'. Som user starter man på `index.jsp`. Herfra kan man logge ind på sin konto på betingelsen af, at man har en. Hvis man indtaster et forkert login (Log in: fejl), bliver man ført til `error.jsp`. Hvis login er indtastet korrekt (Log in: OK), bliver man ført til `welcomeuser.jsp`. Herfra kan user se to valgmuligheder i navigationsbaren - disse kategoriseret i containeren 'Public Pages'. User kan bygge sin carport ved at gå til `buildcarport.jsp`. Givet det kriterium, at der udfyldes oplysninger korrekt på `buildcarport.jsp` og user trykker 'bestil', videresendes user til

myorder.jsp. User kan se sine ordre ved at gå til *myorder.jsp*. Herunder fremstilles der to muligheder for hver individuelle ordre:

- Mulighed 1 (Vis detaljer): User klikker på ‘vis detaljer’ og bliver videresendt til *showorderdetails.jsp*, hvor oplysningerne samt en tegninger inkl. mål over carporten kan ses. Givet det kriterie, at user accepterer en ordre, der har statussen ‘afventer kunde svar’ eller ‘ordre på vej’, kan user også se sin stykliste for carporten i *showorderdetails.jsp*.
- Mulighed 2 (Accepter tilbud): User accepterer tilbuddet og bliver videresendt til *orderconfirmation.jsp*, givet det kriterie at status står som ‘afventer kunde svar’ er overholdt (Accept offer: OK). Hvis det givne kriterier ikke er overholdt, sker der ikke noget.

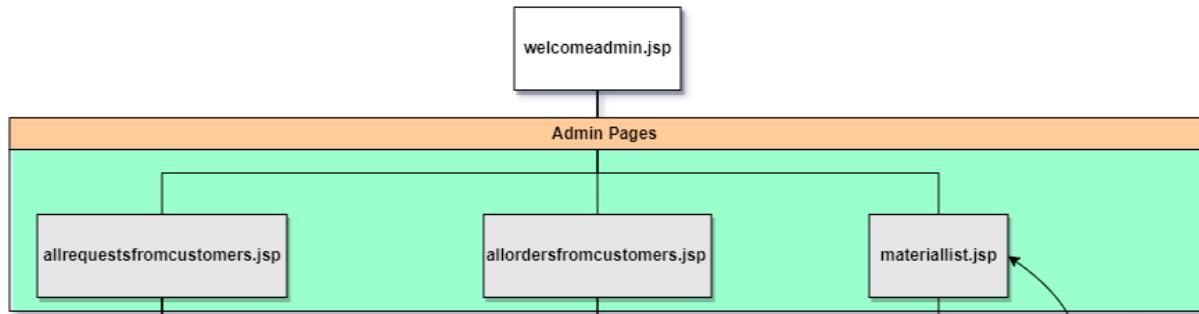
Admin navigation

Navigationsdiagram [2] - Admin Navigation (Opdateret)



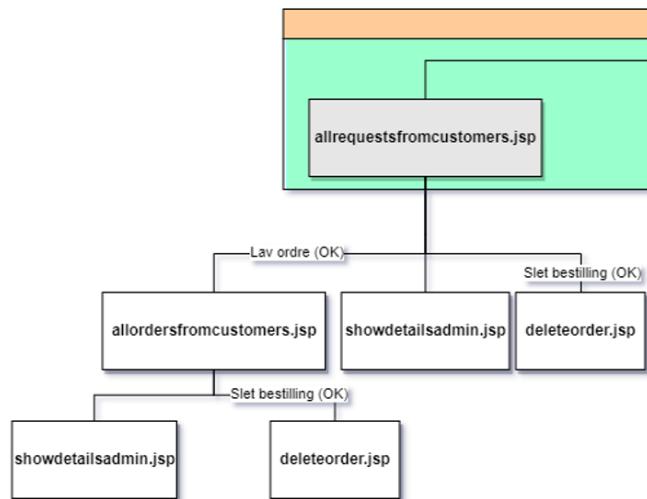
Brugeroplevelsen på hjemmesiden bliver her vist gennem rollen ‘admin’. Som admin starter man på *index.jsp*. Herfra kan admin logge ind på sin konto på betingelsen af, at det login man

logger ind med, har admin rollen. Hvis man indtaster et forkert login (Log in: fejl), bliver man ført til *error.jsp*. Hvis login er indtastet korrekt (Log in: OK), bliver man ført til *welcomeadmin.jsp*. I billedet under ligger *welcomeadmin.jsp* inde i “loggedOn” containeren, som indeholder admins følgende muligheder, som vist i billedet, herunder:



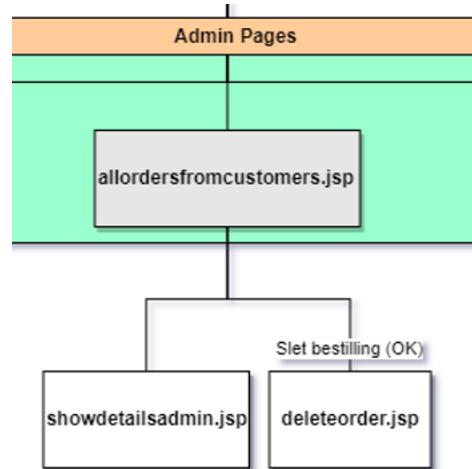
Model: Her ses tre valgmuligheder i navigationsbaren (Kunde bestillinger, Kunde ordre og Materialeliste) - disse kategoriserer i containeren ‘Admin Pages’.

Mulighed 1 (Kunde bestillinger): Admin bliver sendt til ‘Kunde bestillinger’ siden *allrequestsfromcustomers.jsp*. Her kan admin se alle kundens bestillinger samt ordre status.



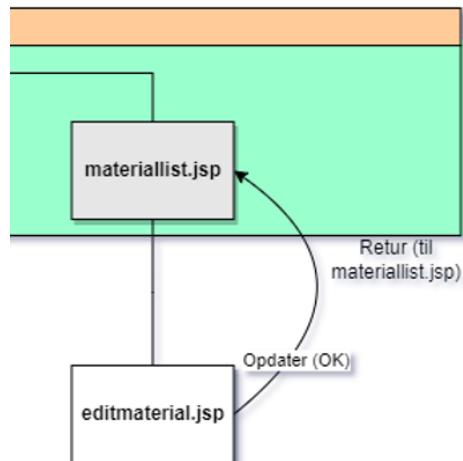
Model: Her ses de tre valgmuligheder (lav ordre, vis detaljer og slet bestilling) - disse tilgås gennem allrequestfromcustomers.jsp.

Mulighed 2 (Kunde ordre): Admin bliver sendt til ‘Kunde ordre’ siden *allordersfromcustomers.jsp*. Her kan admin se alle kundens ordrer samt ordre status.



Model: Her ses de to valgmuligheder (vis detaljer og slet bestilling) - disse tilgås gennem allordersfromcustomers.jsp.

Mulighed 3 (Materialeliste): Admin bliver sendt til ‘Materialeliste’ siden *materiallist.jsp*.
Her kan admin se alle materialerne og redigere i dem.



Model: Her ses valgmuligheden (rediger materialeliste) - disse tilgås gennem materiallist.jsp.

5.6 Sekvensdiagram og klassediagram

Klassediagram samt sekvens diagram kan ses i GitHub projektet under mappen ‘documentation’. Vi har valgt at vedlægge et udsnit af sekvensdiagrammet i *bilag 11.8* med udgangspunkt i funktionen *createOrder*. Klassediagrammet viser den overordnede kodenstruktur imellem de forskellige klasser, som vi har oprettet. Dette kan ses i dokumentations mappen i vores github projekt.

6. Valg af arkitektur

6.1. Model-View-Controller (MVC)

Den udleverede startkode benytter MVC pattern, som danner grundlag for arkitekturen i opgaven.

- *Model* indeholder al data og logiske regler; Dette lag fungerer uafhængigt af brugergrænsefladen.
- *View* præsenterer modellens data på en brugervenlig grænseflade, i dette tilfælde HTML/JSP-siderne.
- *Controller* håndterer input (request) samt output (response). Controlleren håndterer blandt andet requestScopes.

Koden anvender også Singleton pattern og Facade pattern:

- *Singleton* er anvendt til, at sikre at et objekt og/eller en klasse kun bliver dannet én gang. Vi anvender det til at adskille brugers data, når der er flere brugere logget på. Singleton findes ydermere i vores database connection, og sikrer, at der kun oprettes én forbindelse ad gangen.
- *Facade* pattern anvendes til kommunikation med databasen. Vi kan sikre en gennemsigtighed samt gøre programmet mere letlæseligt idet vi adskiller selve databasen samt SQL statements fra resten af koden - denne data ligger i 'mapperne'. I praksis betyder dette at de metoder der ligger her kan benyttes til at hente data ned fra forskellige steder - eksempelvis en database eller indlæse data fra en fil.

6.2 Særlige forhold

- **Applications scope:** Vi bruger application scope til at gemme data over hele applikationen. I dette tilfælde en Connection Pool.
- **Session scope:** Vi bruger session scopes til at gemme data på en session, når en user eller admin logger ind. Det gør, at vi kan gemme dataen til den specifikke bruger. I vores projekt er det username (e-mailen) og password.
- **Request scope:** Request scope anvendes til at gemme data på en JSP side. Denne data kan videreføres til den servlet, der refereres til i JSP siden. Derudover bruger vi

request scope til at gemme data, som vi gerne vil have fat i og vist på andre JSP sider.

- **Exception handling:** Vi har valgt at håndtere fejl mellem databasen og klienten i vores kode med exception handling. Dette gør vi, for at nemmere kunne forstå, hvor, hvornår og hvilke fejl der opstår i koden. På denne måde kan vi nemt tilgå specifikke fejlmeldelser og derved nemt løse opståede problemer.

Et eksempel på dette kan ses her:

```
public static int createShed(Shed shed, ConnectionPool connectionPool) throws DatabaseException {
    Logger.getLogger("web").log(Level.INFO, msg: "");
    String sql = "INSERT INTO Shed (length, width) VALUES (?,?)";
    try (Connection connection = connectionPool.getConnection()) {
        try (PreparedStatement ps = connection.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS)) {
            ps.setInt(parameterIndex: 1, shed.getShedLength());
            ps.setInt(parameterIndex: 2, shed.getShedWidth());
            ps.executeUpdate();
            ResultSet rs = ps.getGeneratedKeys();
            rs.next();
            return rs.getInt(columnIndex: 1);
        }
    } catch (SQLException ex) {
        throw new DatabaseException(ex, "Could not insert shed into database");
    }
}
```

Metoden smider en database exception, hvilket gør, at vi kan se outputtet i terminalen, når fejlen opstår. Connection til databasen, samt den kode, der skal udføres for at indsætte data i tabellen i databasen, er pakket ind i en try-catch. Ved dette kan vi sikre os, at dataen ikke manipuleres, ved at programmet ikke går helt i stå, selvom koden kører, og den fejler fra databasens side. Output teksten ved at have en SQLException i catch delen lyder derfor således “*Could not insert shed into database*”.

- **Login (roller):** Vi har i vores database valgt, at en user skal have en rolle: admin eller user. Når man logger ind, så ved systemet, om man er en user eller en admin, og ud fra dette, navigere koden brugeren hen til de .jsp sider, brugeren skal kunne tilgå.

```

HttpSession session = request.getSession();
session.setAttribute( name: "user", value: null); // invalidating user object in session scope
String email = request.getParameter( name: "email");
String password = request.getParameter( name: "password");

try {
    User user = UserFacade.login(email, password, connectionPool);
    session = request.getSession();
    session.setAttribute( name: "user", user); // adding user object to session scope

    if (user.getRole().equalsIgnoreCase( anotherString: "admin")) {
        request.getRequestDispatcher( path: "/WEB-INF/admin/welcomeadmin.jsp").forward(request, response);
    }
    if (user.getRole().equalsIgnoreCase( anotherString: "user")) {
        request.getRequestDispatcher( path: "/WEB-INF/user/welcomeuser.jsp").forward(request, response);
    }
    request.getRequestDispatcher( path: "index.jsp").forward(request, response);
} catch (DatabaseException e) {
    request.setAttribute( name: "errormessage", e.getMessage());
    request.getRequestDispatcher( path: "error.jsp").forward(request, response);
}
}

```

Vi tjekker om et login er en admin eller en user, i en try-catch så brugeren kommer ind på den rigtige welcome side - Login servlet.

I linjen `User user = UserFacade.login(email, password, connectionpool);` validerer vi gennem et SQL statement, om der eksisterer en oprettet bruger i databasen, ved at tjekke for en email og et password. Hvis dataen kan hentes ned, tjekker vi derefter, om brugeren er korrekt oprettet, ved at tjekke om brugeren har tildelt en rolle. Hvis ikke dette er tilfældet, sendes vi ned i catch, der sender brugeren videre til en fejlsidé og giver os en fejlmeddeelse.

- **Stykliste beregner:** Vi har valgt at lave en Calculator klasse, der indeholder alle vores beregninger. Grunden til, at vi har gjort det på den måde, er, at vi nemt kan holde et overblik over de forskellige metoder, og hvis der skal tilføjes flere materialer, så kan vi nemt tilføje beregningsmetoderne i Calculator klassen. Vores calculator bliver også brugt til at lave vores tegning.
- **Tegning:** Vores plantegninger er autogenereret via SVG og ved brug af vores Calculator klasse i vores persistens lag, kan vi beregne de forskellige mål på tegningerne. Derudover gør vi brug af de indbyggede SVG metoder, som vi har fået udleveret, til at kunne tegne en carport, som så senere bliver vist på vores jsp-sider.

7. Udvalgte kodeeksempler

SVG - Her ser vi et udsnit af hvordan vi opretter et SVG object, som viser en tegning. Vi har valgt at lave to forskellige; en carportDrawTop og en carportDrawSide. Vi kan herved nemt tilgå dem på jsp-siderne og få dem vist via showdetailsadmin.jsp og showdetailsuser.jsp.

```
SVG carportDrawTop = CarportSVG.createNewSVG( x: 0, y: 0, height: 100, width: 100, viewBox: "0 0 1280 720");
carportDrawTop = CarportSVG.makeSVGTop(order, carportDrawTop);
session.setAttribute( name: "carportDrawTop", carportDrawTop);

SVG carportDrawSide = CarportSVG.createNewSVG( x: 0, y: 0, height: 100, width: 100, viewBox: "0 0 1280 720");
carportDrawSide = CarportSVG.makeSVGSide(order, carportDrawSide);
session.setAttribute( name: "carportDrawSide", carportDrawSide);
```

Screenshot af showdetailsadmin servlet

Update status - Her viser vi et udsnit af vores showdetailsadmin servlet. Som man kan se, bliver statussen på en ordre her sat til 2, hvis statussen er 1. Det sker når vi går fra “Kunde bestillinger” og gerne vil ind og se detaljerne. Vores status 2 er ‘i behandling’, så her giver det mening at ordenen er i behandling, når en admin har læst detaljerne for orden.

```
Order order = OrderFacade.readDataFromAnOrder(orderID, connectionPool);
if (order.getStatusID() == 1) {
    order.setStatusID(2);
    OrderFacade.updateStatus(order, connectionPool);
}
session.setAttribute( name: "order", order);
```

Screenshot af showdetailsadmin servlet

Status check - Her tjekker vi for statussen, inde i createorder servletten. Forskellen på en bestilling og en ordre er, at en ordre har været ‘i behandling’ af en admin, så næste step er for admin at lave det til en ordre, som er det denne servlet, bliver brugt til. Hvis ordenen er sat til status 1 eller 2 sættes orden nu til status 3 ‘afventer kundesvar’.

```

HttpSession session = request.getSession();
int orderId = Integer.parseInt(request.getParameter("name: "orderID"));

try {
    Order order = OrderFacade.readDataFromAnOrder(orderId, connectionPool);
    if (order.getStatusID() < 3) {
        order.setStatusID(3);
        OrderFacade.updateStatus(order, connectionPool);
    }
    session.setAttribute("name: "order", order);
    request.getRequestDispatcher("path: "/navToCustomerOrders").forward(request, response);
}

```

Screenshot af createOrder servlet

Status check 2 - Her tjekker vi også for statussen, og brugeren bliver sendt til forskellige sider alt afhængig af, om kunden har 'betalt' for sin ordre eller ej. Forskellen på de to jsp-sider er, om man kan se en stykliste eller ej.

```

if (order.getStatusID() <= 2) {
    request.getRequestDispatcher("path: "WEB-INF/user/showorderdetailsuser.jsp").forward(request, response);
}
request.getRequestDispatcher("path: "WEB-INF/user/orderdetailswithbom.jsp").forward(request, response);

```

Screenshot af showOrderDetailsUser servlet

8. Status på implementering

Vi fik implementeret alle de user stories, som vi havde planlagt, undtagen at kunne redigere en ordre og at manuelt kunne sætte en status på ordren. Undervejs i processen besluttede vi os for, at ordre statussen skulle være noget, der blev opdateret automatisk, som også ses i kode eksemplerne i tidligere afsnit.

Vi har valgt ikke at implementere en email funktion, hvor kunde og sælger typisk ville have dialog i form af et kundemøde. Denne process har vi sprunget over, idet det ikke var en primær nødvendighed for projektets udførelse. I stedet for har vi valgt at selve mødet mellem kunde og sælger skal ligge imellem de to ordre statusser '*i behandling*' og '*afventer kunde svar*'. Når samtalen er afholdt, vil kunden så gå ind og manuelt trykke '*accepter ordre*'.

I vores kode har vi ikke valgt at implementere alle CRUD metoder for alle tabeller i databasen, men udelukkende fokuseret på dem vi har, skulle bruge for at løse user stories.

Vi har stylet .jsp siderne med HTML 5 samt BOOTSTRAP, så de er funktionelle, men vi har ikke lagt vægt på UI/UX og CSS. Dog har vi fulgt vores figma layout og itereret i dette undervejs.

8.1 Videreudvikling

Nogle af de muligheder, vi havde dialog om undervejs, var muligheden for at tilføje ekstra funktioner til vores nuværende kode.

- **Rediger en ordre:** Vi havde forestillet os, at man som sælger have mulighed for at tilgå kundebestillinger og -ordre og redigere i dataen. Dette kunne være en adresse der skulle ændres, mål på materialer eller tilføjelse af et leveringssted. Hvis der nu opstod taste-fejl eller miskommunikation med kunden, så ville det også være smart for sælgeren at kunne ændre fejl og mangler i ordren, før leveringen bliver afsendt.
- **Rabat på en ordre:** Det kunne være en god ide at have mulighed for at tilføje en rabat på totalprisen på ordren, men dette har vi ikke valgt at implementere.
- **Chatfunktion:** En implementation af en chat funktion, så en bruger nemt og overskueligt ville kunne søge information samt få hjælp online i forbindelse med bestillingen af en carport. Denne chat funktion skulle være en direkte forbindelse til sælgeren og ikke en chat bot.
- **Oprettelse af bestillinger fra sælgers side:** Vi havde også overvejet idéen om, at sælgeren skulle have mulighed for at bygge en carport for kunden.
- **Videreudvikling af eksisterende apps:** Idet Fog har deres egen app, tænkte vi, at man kunne videreudvikle denne således, at der kunne tilføjes nye funktioner, således, at bestilling af carporte var en integreret del heraf.
- **AR Teknologi:** En smart funktion ville være, at de genererede tegninger af carporten kunne ses i et 3D perspektiv, således at kunden kunne få en dybere forståelse for sit produkt, tjekke for kosmetiske fejl og holde den op imod deres ejendom og se om de er tilfredse med det genererede eksempel af deres produkt, de er i færd med at designe.
- **Samarbejde med vognmandsfirmaer:** Fog kunne også branche ud i partnerskab med vognmandsfirmaer. Der kunne være en tilkøbs service for kunden, således at der er mulighed for afhentning af overskydende materialer fra deres tidligere og måske nedrevne carport (givet at kunden er i gang med at erstatte deres gamle skur eller carport med et produkt fra Fog). Leveringsservicen, som Fog ville kunne tilbyde

(igenom deres nye partnere) sikrer, både at kundeoplevelsen er nemmere og mere brugervenlig, men sikrer også fokus på øget salg og kundetilfredshed.

9. Test

Generelt har vi testet koden via Google Chrome browseren og derefter kontrolleret vores database, om dataen er blevet ændret. Vi har valgt at lave unit test på nogle af funktionerne i klassen ‘Calculator’ for at tjekke om vores beregninger til styklisten er udregnet på korrekt vis. Herudover har vi arbejdet med integrationstest af connection til databasen.

Metode	Testtype	Testet og virker	Implementeret i kode
calculatePoles	Unit	Ja	Ja
calculateRafters	Unit	Ja	Ja
calculateRoof	Unit	Ja	Ja
login	Integration	Ja	Ja
createUser	Integration	Ja	Ja

Tabeloversigt over hvilke metoder der er testet i den implementeret kode.

10. Proces

For at nå i mål med implementeringen af vores user stories har vi som gruppe arbejdet på skift med håndtering samt opsætning af koden. Herudover har vi anvendt Git til at inddele vores kode i mindre dele, således at man kunne branche ud fra henholdsvis main og developer branchen, og arbejde særskilt på tværs af computere uden at koden ville blive overskredet eller slettet undervejs. Vi fik løst opgaven med godt samarbejde, hvor alle havde mulighed for at komme med inputs og ideer til den endelige løsning. Vi har haft gavn af at planlægge vores process og iterere i den før, under og efter hver user story og deres implementationer.

Vi har dagligt mødtes via Discord og/eller Google Meet kl. 9.00, hvor der er blevet afholdt en daglig scrum meeting dvs. en agenda for dagens delmål i projektet og til samarbejde/sparring. Disse møder har typisk taget 30 minutter, men i forbindelse med vejledning har det også kunne tage længere tid. Ved at afholde disse møder sikrede vi os, at alle gruppemedlemmer var up to date med, hvor langt vi var i processen på projektet. Vi har fulgt Figma mockup ift.

udarbejdelse af layout, og vi har anvendt Trello til opsætning af en kanban board, som vi løbende har itereret i. Link hertil:

<https://trello.com/b/yvGCwFpp/fog-carport-eksamensprojekt-kanban-board>

Note: Klik på hvert link for ugen for at tilgå den uges kanban

Vi har ikke haft en Scrum master i gruppen, da vi gerne ville sikre, at alle gruppemedlemmer kunne have indflydelse på de valg, der blev truffet. Dog har Helena ageret tovholder på Trello og dagligt ført logbog, så vi har kunnet lægge en plan for dagen. En af ulempene ved ikke at have denne Scrum master, har gjort, at gruppens beslutningsprocesser, har taget en del tid; Vi kunne nok have afkortet diskussioner og effektiviseret vores arbejde, men det har også været en omvæltning for gruppen kun at arbejde online. I andre projekter har gruppen mødtes fysisk, men idet vi har været ramt af meget sygdom i december måned, har vi valgt at blive hjemme.

Herudover har vi i gruppen haft udarbejdet en plan for, hvornår de forskellige gruppemedlemmer har måtte gå fra til at passe deres deltidsjobs ved siden af studiet. Samt noteret, hvornår vi kunne holde fri undervejs i processen ift. lægebesøg og hen over jul/nytår. Dette har givet en øget frihed og ro til at kunne samarbejde bedre som et team, og har styrket vores indbyrdes relationer, da det har været et alternativ til at mødes fysisk og fastholde en social aktivitet.

10.1. Arbejdsprocessen faktuelt

- Trello - Vi har anvendt Trello (*bilag 10.12*) som et projektstyringsværktøj.
- Figma mockup - Anvendt til udarbejdelse af layout og brugt som en skabelon undervejs til at sikre, at der ikke har været afvigelser. Dette har især været behjælpeligt, mht. at ikke kan kunne mødes fysisk.
- Git og GitHub - til upload af den implementerede kode på tværs af branches.
- Discord/Google Meet - til videosamtaler og deling af skærm.

10.2. Arbejdsprocessen reflekteret

Det første vi gjorde, var at sætte os ned og lave en brainstorm (*bilag 11.9*), der viste alle JSP siderne, og hvordan deres indbyrdes relation skulle være, herunder enighed om navngivning heraf. Vi blev også enige om at lave tre page templates (normal, user, admin).

Herefter forsøgte vi, at notere de udleveret user stories ned, og hvordan deres tilhørerforhold til siderne var. Dette gjorde vi for at imødekomme en guideline, for hvordan vi skulle komme i gang med at kode.

Nedenstående tabel viser en kort oversigt over de forskellige ugers forløb, ift. hvad de forskellige gruppemedlemmer har arbejdet på. Der har været overlap og samarbejde på tværs, men vi har forsøgt at arbejde i mindre teams for at effektivisere arbejdsprocessen.

	FELICIA	JAMIE	ISAK	HELENA	ANDREAS
UGE	Modeller Opsætning Figma Analyse på udleveret materiale User Stories Rapport	Prisliste Opsætning GitHub Fokus på Git branches	Modeller Opsætning af Database EER diagram	Modeller Planlægning Trello User Stories Rapport	Modeller Rapport Use-cases
UGE	Layout / Kode SVG Workshop Rapport Kode - US 4, 4a, 5	Ændringer til startkode SVG Workshop Kode - US 1, 2, 3, 4, 4a, 5	Database op på Digital Ocean SVG Workshop Kode - US 1, 2, 3, 4, 4a, 5	Kode SVG Workshop Trello Rapport Kode - US 1, 2, 3, 4, 4a, 5	Opdatering af prisliste SVG Workshop Rapport Kode - US 4, 4a, 5
UGE	SVG Tegninger Opdatering Figma v1 Kode - US 7, 7a, 8, 8a, 6, 6b	SVG Tegninger Kode - US 8, 8a, 6, 6b	SVG Tegninger Kode - US 7, 8, 8a, 6, 6b Opdatering af Database EER diagram	SVG Tegninger Kode - US 8, 8a, 6, 6b Trello	SVG Tegninger Kode - US 8, 8a, 6, 6b
UGE	Kode stykliste Opdatering af modeller Kode - US 4b, 4c, 8a	Kode stykliste Rettelser admin Kode - US 6a, 6c, 7b	Kode stykliste Rettelser bruger Kode - US 6a, 6c, 7b	Kode stykliste Trello Kode - US 4b, 4c, 8a	Kode stykliste Opdatering af modeller Kode - US 4b, 4c, 7b, 8a

UGE 52	Rapport Figma mockup v2 Layout rettelser Unit Test	Rapport, update SVG tegning Opdater beregninger	Rapport, Upload projekt Digital Ocean Unit Test	Rapport Unit Test	Rapport Rettelser til SVG tegning
-------------------	---	--	--	----------------------	---

Uge 48 (uge 1):

I denne uge startede vi med at få oprettet en logbog og opsat de tekniske grundskabeloner, som at få sat et GitHub projekt op, hente startkoden ned samt oprette forskellige dokumenter til modeller og rapporten. Mandag var intro dag hvor opgaven blev præsenteret, og vi fik startet på vores modeller: SWOT, VPC, konceptmodel, domænemodel (v1), navigationsdiagram (v1) samt Figma mockup.

Uge 49 (uge 2):

I denne uge begyndte vi at afholde ugentlig update på forrige uge - *hvor langt var vi nået og hvad skulle gøres i denne uge?* Vi anvendte Trello til opsætning over, hvilke user stories vi ville fokusere på denne uge, og fik implementeret funktionerne log ind og opret bruger. Vi satte skabeloner op til JSP siderne og fik opsat vores database så den ligger på Digital Ocean. Ydermere oprettede vi et EER-diagram (v1) som en oversigt over, hvordan kodens struktur skulle opbygges, ift. hvordan data skulle gemmes for hhv. brugeren og sælgeren. I slutningen af ugen besluttede vi os for at lave de SVG workshops, der var blevet tildelt os fra undervisernes side, således at alle gruppemedlemmer kunne udarbejde tegninger til projektet.

Uge 50 (uge 3):

I denne uge startede vi med en ugentlig update, og snakkede lidt om SVG - *hvor kunne vi bruge det til og forstod alle det udleveret materiale?* Vi begyndte på styklisten i denne uge, og udarbejdelse af beregninger til denne. Derudover havde vi fokus at kunne oprette en ordre og vi fik implementeret nogle metoder der automatisk sætter status på en ordre. Ligeledes fik vi det visuelle layout på plads.

Uge 51 (uge 4):

I denne uge begyndte vi på tegningen til carporten. Vi fik implementeret den SVG workshop vi havde arbejdet med i uge 2, så vi var klar til at lave vores calculator-klasse, der har alle materialers beregninger i sig. Derudover fik vi også ændret i vores database/EER-diagram, så

vi kunne begynde at lave en BOM (Bill of Materials/stykliste) og få den vist ordentligt på siden. Ud over det, fik vi rettet i koden så man som admin kan redigere i de materialer der ligger på siden, hvis nu målene eller priserne skal ændres.

Uge 52 (uge 5):

I femte uge fokuserede vi udelukkende på rapporten, og fik sat alle vores modeller og diagrammer ind og skrevet tekst til det. I slutningen af ugen tog vi tid til at få udarbejdet vores unit testing, så vi kunne teste nogle af de overordnede beregninger mm. Disse tests skulle vi have lavet undervejs, men det kom vi fra. Derfor sikrede vi os i denne uge at dataen i databasen stemte overens, med det vi ville aflevere.

11. Bilag/Appendiks

Nedenstående finder du modeller, tabeller og diverse andre bilag der danner belæg for rapporten og vores projekt. Vi har valgt ikke at implementere disse direkte i rapporten, idet de fylder for meget ift. rapportens krav om antal anslag. Materialer er dog beskrevet i rapporten, og der er henvist til de forskellige modeller med hhv. bilag tal og sidetal.

11.1 User Stories

User stories er delt op således, at der er nogle tildelt brugeren (den besøgende på hjemmesiden) samt en admin (en sælger hos Fog A/S).

Bruger:

- **US-1: Opret bruger**

Som <bruger> ønsker jeg at <kunne oprette en profil>, sådan at jeg <kan logge ind>

Givet at: der tilgås en side hvor der kan oprettes en profil

Når: brugeren indtaster en email samt vælger et password og verificerer dette password

Så: bliver brugeren oprettet i systemet og dataen gemt i en database

Estimat: S

- **US-2: Log ind som bruger**

Som <bruger> ønsker jeg at <kunne gå til en brugerside> sådan at jeg <kan logge ind som brugeren>

Givet at: der tilgås en side hvor der kan logges ind

Når: brugeren indtaster sin email samt sit password (verificerer korrekte login oplysninger)

Så: bliver brugeren dirigeret hen til en brugerside hvor brugeren herfra kan navigere til en ‘Bestil carport’-side eller en ‘mine ordrer’-side

Estimat: S

- **US-3: Lav forespørgsel for carport**

Som <bruger> ønsker jeg at <kunne logge ind på en brugerside> sådan at jeg <kan foretage en forespørgsel på et carport design>

Givet at: brugeren er logget ind med sine oplysninger

Når: brugeren klikker på ‘Byg en carport’ i navigationsmenuen

Så: kan brugeren vælge sine mål og materialer til sin carport

Estimat: M

- **US-4: Se forespørgsel eller ordre**

Som <bruger> ønsker jeg at <kunne logge ind på en brugerside> sådan at jeg <kan se mine forespørgsler eller ordrer>

Givet at: brugeren er logget ind med sine oplysninger

Når: brugeren klikker på ‘Se mine ordrer’ i navigationsmenuen

Så: bliver brugeren dirigeret hen til en ordre side hvor kunden kan se sine forespørgsler og ordrer alt efter hvilken status den har

Estimat: M

- **US-4a: Se og følge status på ordre**

Som <bruger> ønsker jeg at <kunne logge ind på en brugerside> sådan at jeg <kan se og følge status på mine ordre>

Givet at: brugeren er logget ind

Når: brugeren tilgår ‘se mine ordrer’ siden

Så: kan brugeren se status på ordrerne - Oprettet, i behandling, afventer kunde svar, ordre på vej, afsluttet

Estimat: S

- **US-4b: Se ordre detaljer**

Som <bruger> ønsker jeg at <kunne se ordre detaljer> sådan at jeg kan <se tegning og stykliste hvis orden er oprettet>

Givet at: brugeren er logget ind og har lagt en bestilling på et tilbud af en carport

Når: brugeren trykker på ‘vis detaljer’ på ‘mine ordrer’ siden

Så: kan brugeren se sine ordre detaljer inklusiv tegning (og stykliste hvis orden er betalt)

Estimat: L

- **US-4c: Accepter tilbud**

Som <bruger> ønsker jeg at <kunne acceptere et tilbud på en ordre> sådan at jeg

kan <se tegning og stykliste af en carport>

Givet at: brugeren er logget ind og har lagt en bestilling på et tilbud af en carport og sælgeren har lavet en ordre herpå

Når: brugeren trykker på ‘accepter tilbud’ på ‘mine ordrer’ siden

Så: bliver brugeren sendt til en ordrebekræftelse-side

Estimat: S

Admin:

- **US-5: Log ind som admin:**

Som <admin> ønsker jeg at <kunne logge ind> sådan at jeg kan <tilgå en admin side>

Givet at: der tilgås en side hvor der kan logges ind

Når: admin indtaster sin email samt sit password (verificerer korrekte login oplysninger og tjekker om det er et admin-login)

Så: bliver admin dirigeret hen til en admin side hvor admin herfra kan navigere til andre admin sider

Estimat: S

- **US-6: Se kundebestillinger**

Som <admin> ønsker jeg at <kunne logge ind på en admin side> sådan at jeg <tilgå kundebestillinger>

Givet at: admin er logget ind

Når: admin klikker på ‘kundebestillinger’ siden

Så: kan admin se alle kunders bestillinger/forespørgsler

Estimat: S

- **US-6a: Slet kundebestillinger**

Som <admin> ønsker jeg at <kunne logge ind på en admin side> sådan at jeg <kan slette kundebestillinger>

Givet at: admin er logget ind

Når: admin klikker på ‘kunde bestillinger’ siden

Så: kan admin se alle kunders bestillinger og slette en bestilling

Estimat: S

- **US-6b: Se bestillings detaljer**

Som <admin> ønsker jeg at <kunne logge ind på en admin side> sådan at jeg <kan se detaljer for de tilbud kunderne har bestilt>

Givet at: admin er logget ind og står på ‘kunde bestillinger’ siden

Når: admin klikker på ‘vis detaljer’ knappen

Så: kan admin se detaljer på ordren (kundeoplysninger, carport oplysninger, stykliste)

Estimat: L

- **US-6c: Se tegning**

Som <admin> ønsker jeg at <kunne logge ind på en admin side> sådan at jeg <kan se tegninger over carportene>

Givet at: admin står på ‘kunde bestillinger’ eller ‘kunde ordrer’-siderne

Når: admin klikker på ‘vis detaljer’ siden

Så: kan admin se snittegning samt plantegning over carporten

Estimat: M

- **US-7: Oprette en ordre:**

Som <admin> ønsker jeg at <kunne logge ind> sådan at jeg <kan oprette en ordre>

Givet at: admin er logget ind og er på ‘kunde bestillinger’ siden

Når: admin trykker på ‘lav ordre’

Så: bliver statussen på ordren opdateret og der navigeres videre til ‘kunde ordrer’ siden

Estimat: M

- **US-7a: Angiv status på kundeordrer**

Som <admin> ønsker jeg at <kunne logge ind på en admin side> sådan at jeg <tilgå kunde ordre og angive status på dem>

Givet at: admin er logget ind

Når: admin klikker på ‘kundeordrer’ siden

Så: kan admin se alle kunders ordre og angive status på dem alt efter hvor langt ordren er i processen

Estimat: M

- **US-7b: Se ordre detaljer**

Som <admin> ønsker jeg at <kunne logge ind på en admin side> sådan at jeg <kan se ordre detaljer fra de enkelte ordrer>

Givet at: admin er logget ind og står på ‘kunde ordrer’ siden

Når: admin klikker på ‘vis detaljer’ knappen

Så: bliver admin sendt til ‘ordrer detaljer’ siden og kan se alle detaljerne omkring kunden og ordren (stykliste og tegning inkluderet)

Estimat: M

- **US-7c: Rediger ordre**

Som <admin> ønsker jeg at <redigere en ordre> sådan at jeg <kan ændre orden efter kundens ønsker efter kundemøde>

Givet at: der er oprettet en ordre og kunden er kontaktet

Når: admin står på ‘kunde ordre’ og klikker på ‘ordre detaljer’

Så: kan admin ændre i orden baseret på kundens ønsker

Estimat: M

- **US-8 Se materialeliste:**

Som <admin> ønsker jeg at <kunne logge ind på en admin side> sådan at jeg <kan se en materialeliste og give god vejledning>

Givet at: admin er logget ind

Når: admin klikker på ‘materialelisten’ i navigationsmenuen

Så: kan admin få et fuldendt overblik over materialelisten.

Estimat: M

- **US-8a Rediger materialeliste:**

Som <admin> ønsker jeg at <kunne se en materialeliste> sådan at jeg <kan opdatere beskrivelse, pris mm i materialelisten>

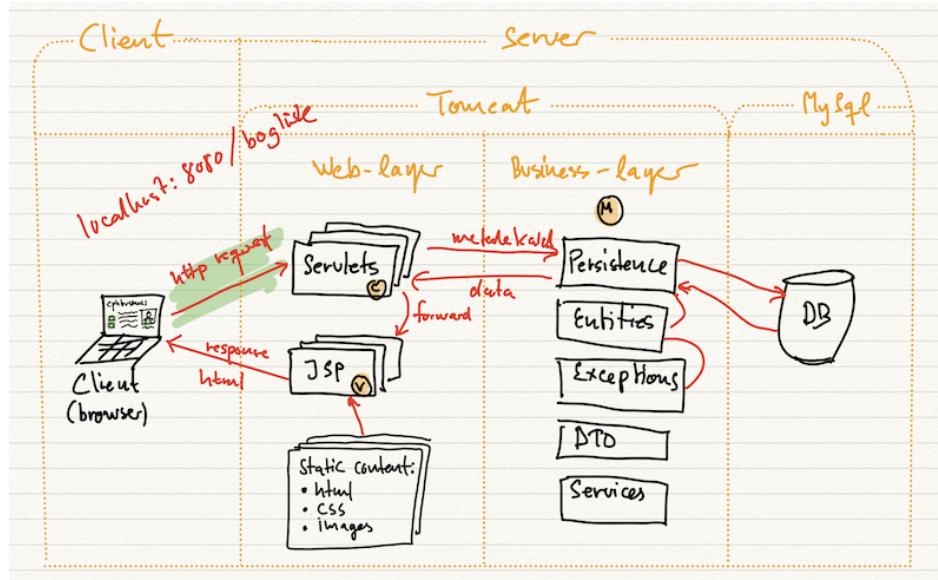
Givet at: admin står på ‘materialelisten’ siden

Når: admin klikker på ‘rediger’ knap ud fra det materiale han gerne vil ændre i

Så: kan admin opdatere data på et materiale i listen

Estimat: M

11.2. Model View Controller



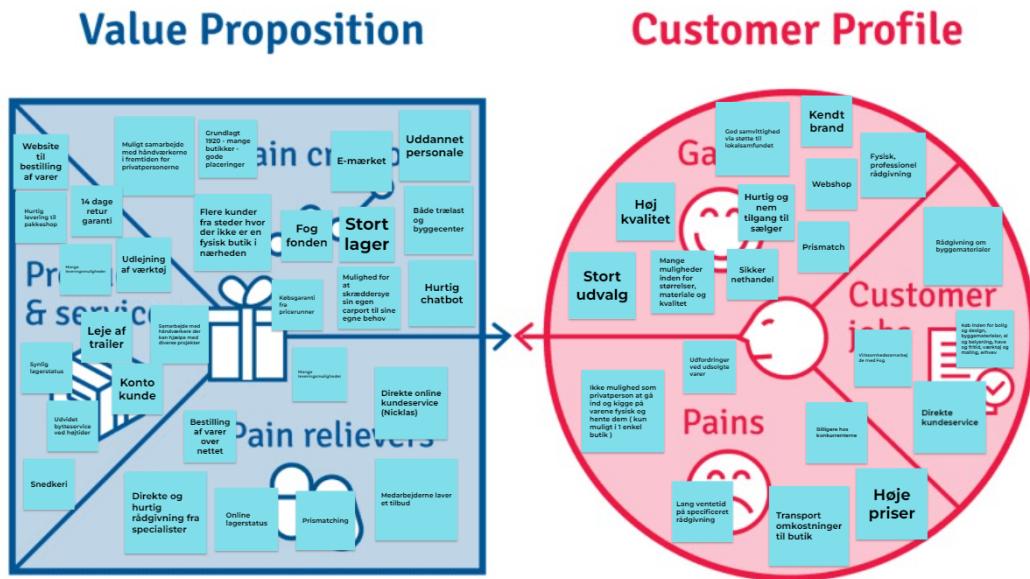
MVC modellen giver os et overblik over den anvendte arkitektur:

1. Vi starter fra *Client (browser)*, i vores “*localhost:8080/*”.
2. Herfra sender vi et *http request* ud til en korrekt servlet. Denne servlet fungerer som en *controller*, der håndterer modtagelsen af klienten samt metodekald og data fra vores modeller.
3. Dernæst opretter vi et *metodekald* til *persistence*, som går ned og henter data fra DB (databasen) i MySQL.
4. Data bliver ført tilbage og håndteret i vores persistence lag (mapper, facade).
5. Data bliver håndteret i de forskellige klassers metoder i *entities* og *exceptions* m.m.
6. Når dataen er hentet fra databasen, bliver dataen håndteret i den korrekte servlet (*Controller*) fra tidligere, som så bliver *forward requested* til den ønskede jsp-side man vil refererer til (*View*).
7. Yderligere kan en jsp-side håndtere selve de visuelle komponenter til hjemmesiden (billeder, layout, farver m.m.). Det henter vi så fra et *statisk sammenhæng*. F.eks. i en css fil: “file.css”.
8. Til sidst sender jsp-siden en *http response* med data tilbage til klienten.

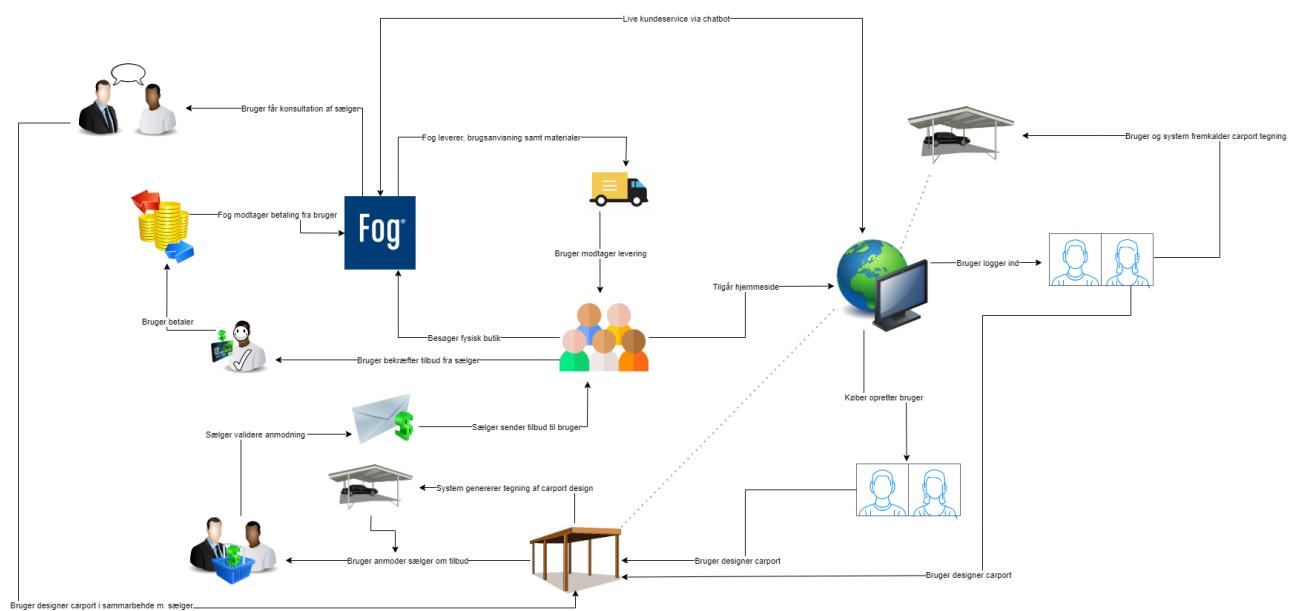
11.3. Value Proposition Canvas (VPC)

Link:

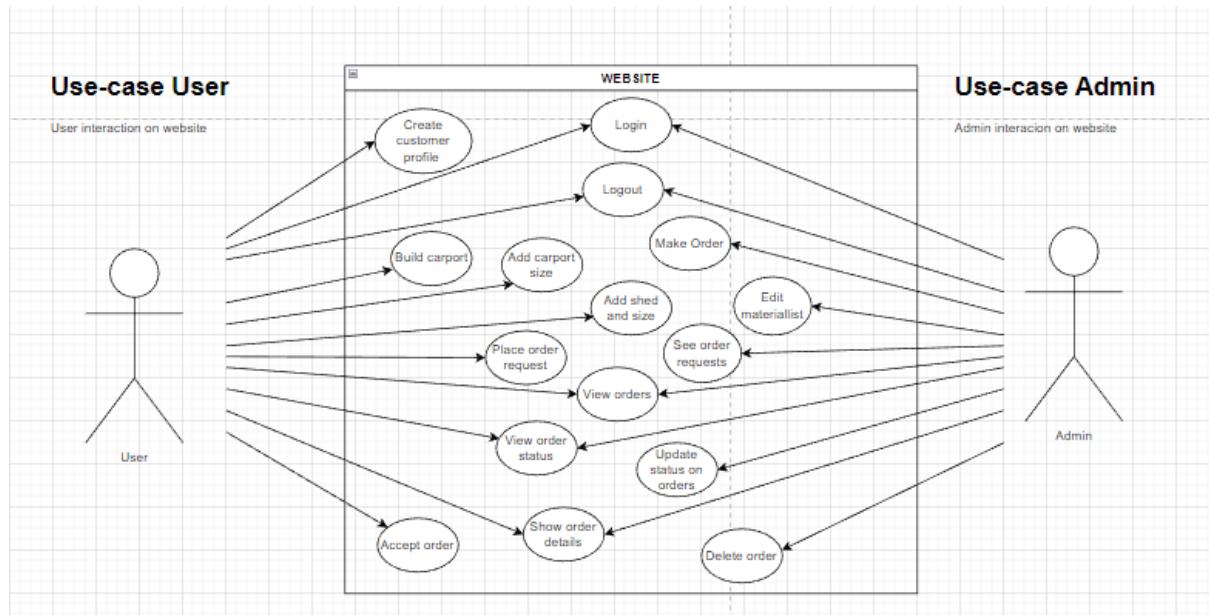
<https://jamboard.google.com/d/1KRAcQJxDloN94YRb0Zjb6bqIFdV56CDh0VzGH2u5hoE/viewer?f=1>



11.4. Konceptmodellen

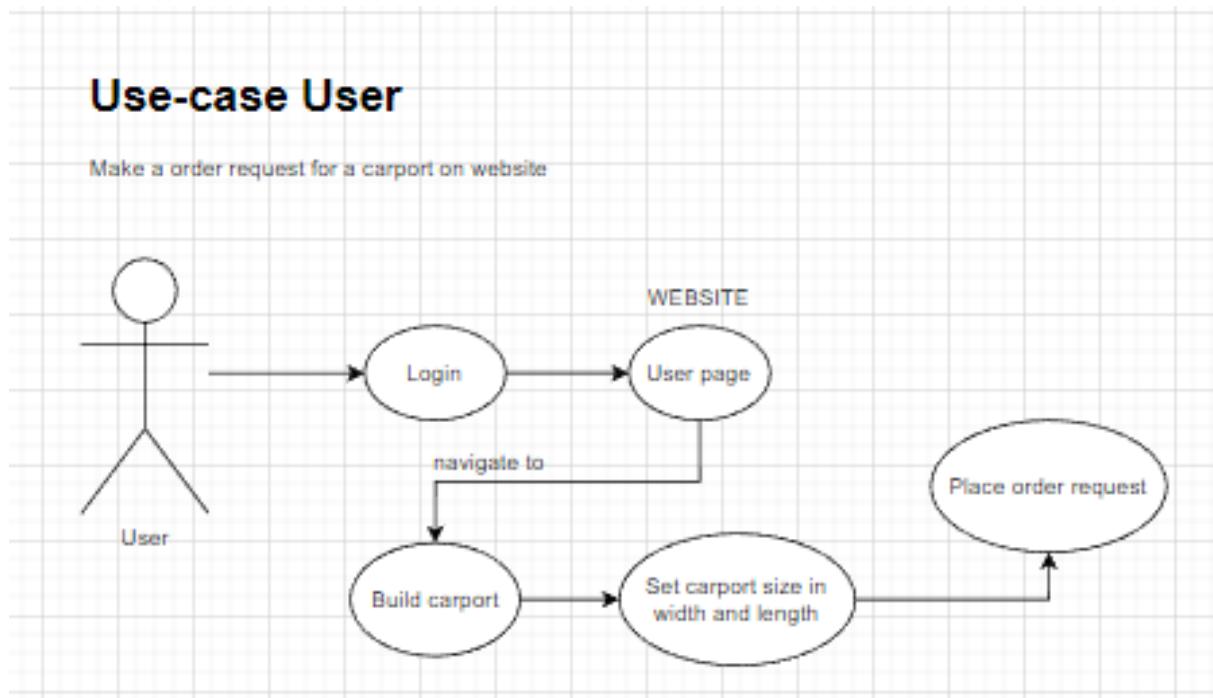


11.5. Use-cases



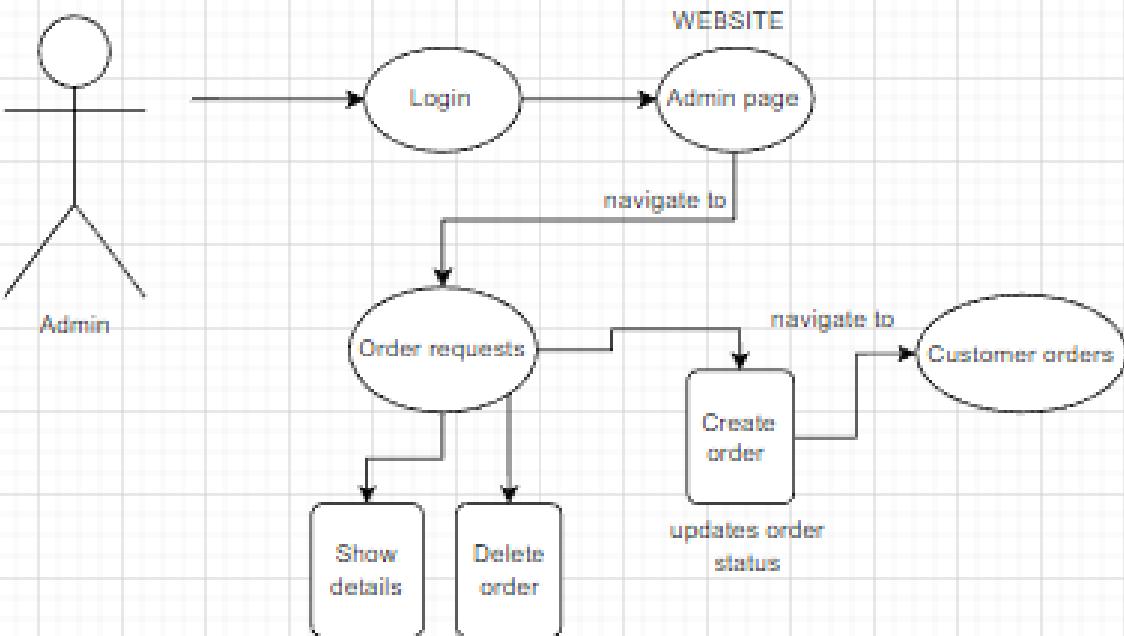
Use-case User

Make a order request for a carport on website

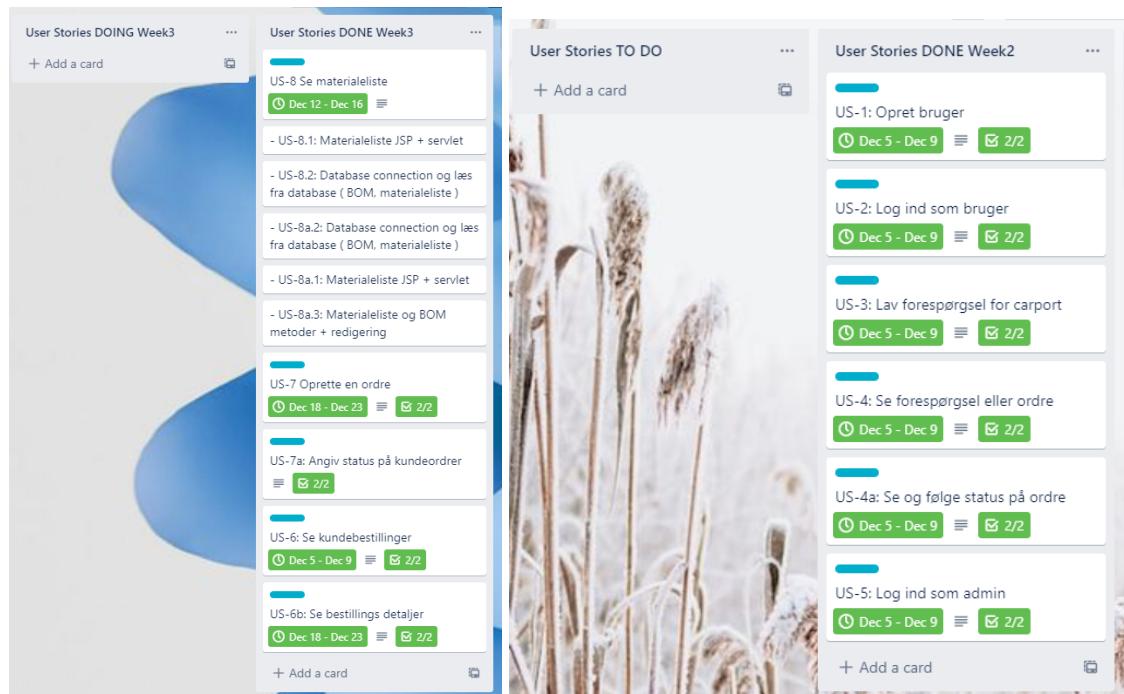


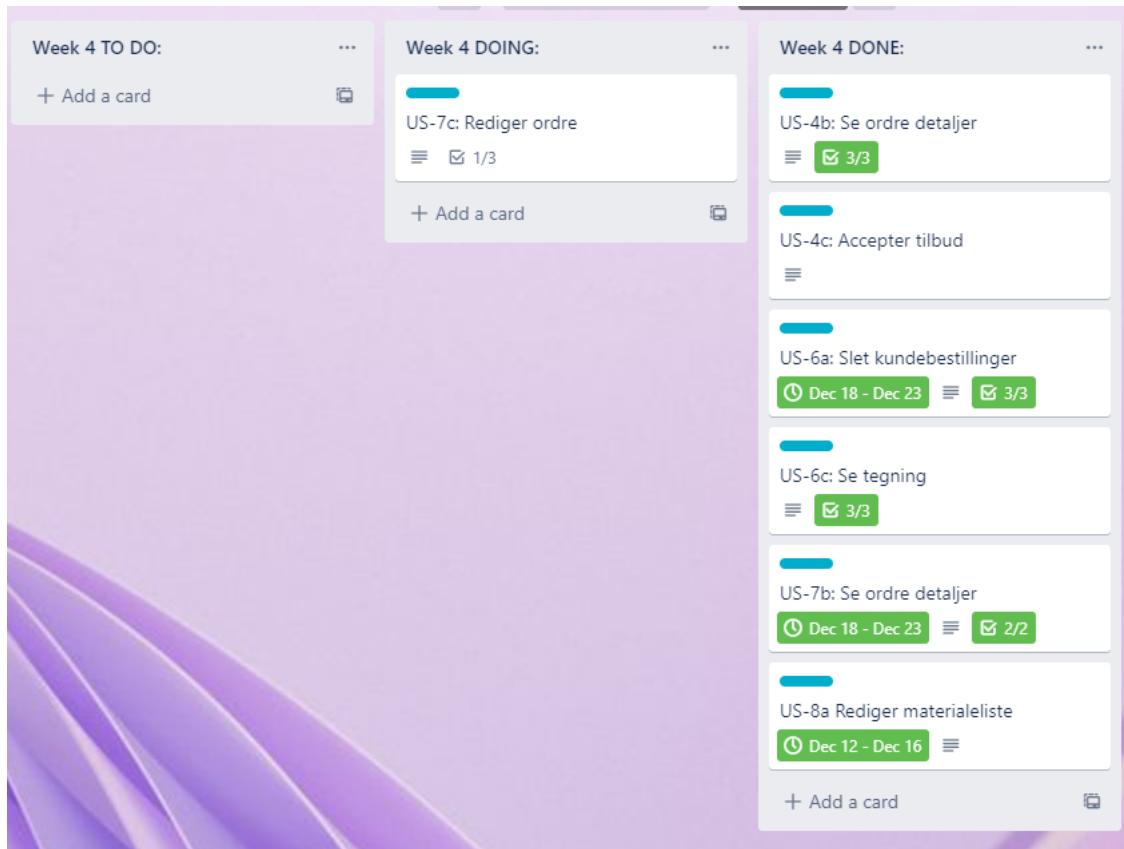
Use-case Admin

Create an order for a customer order request
and set order status



11.6. Trello

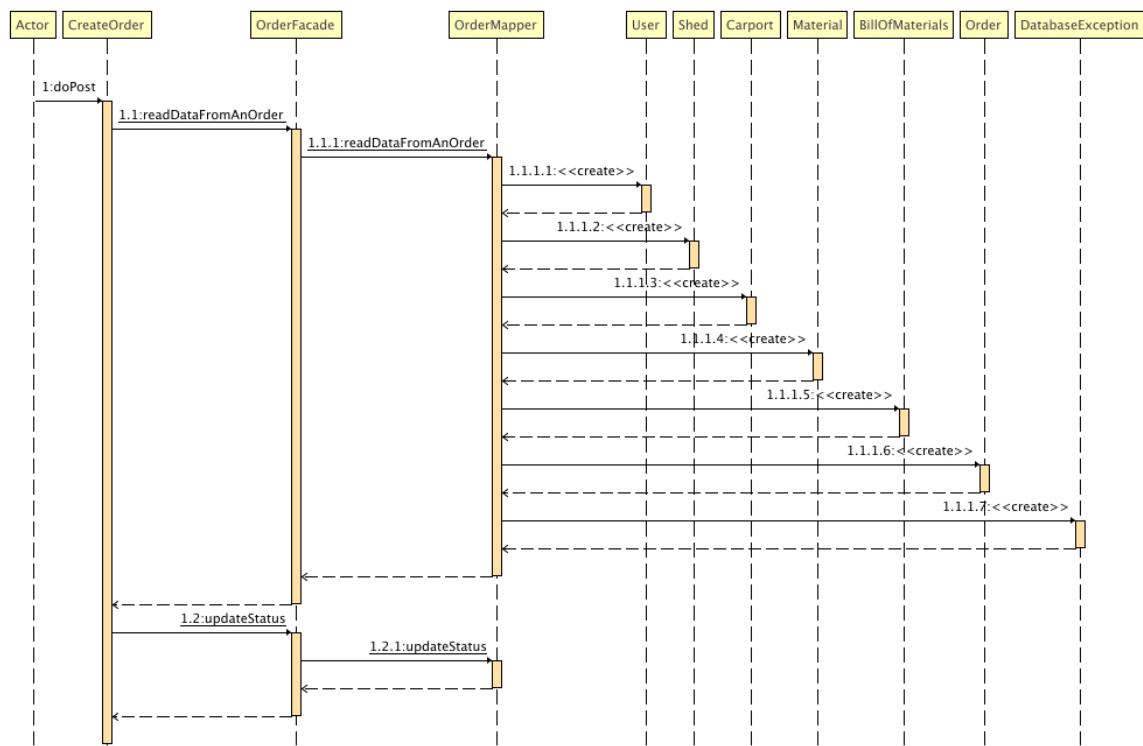




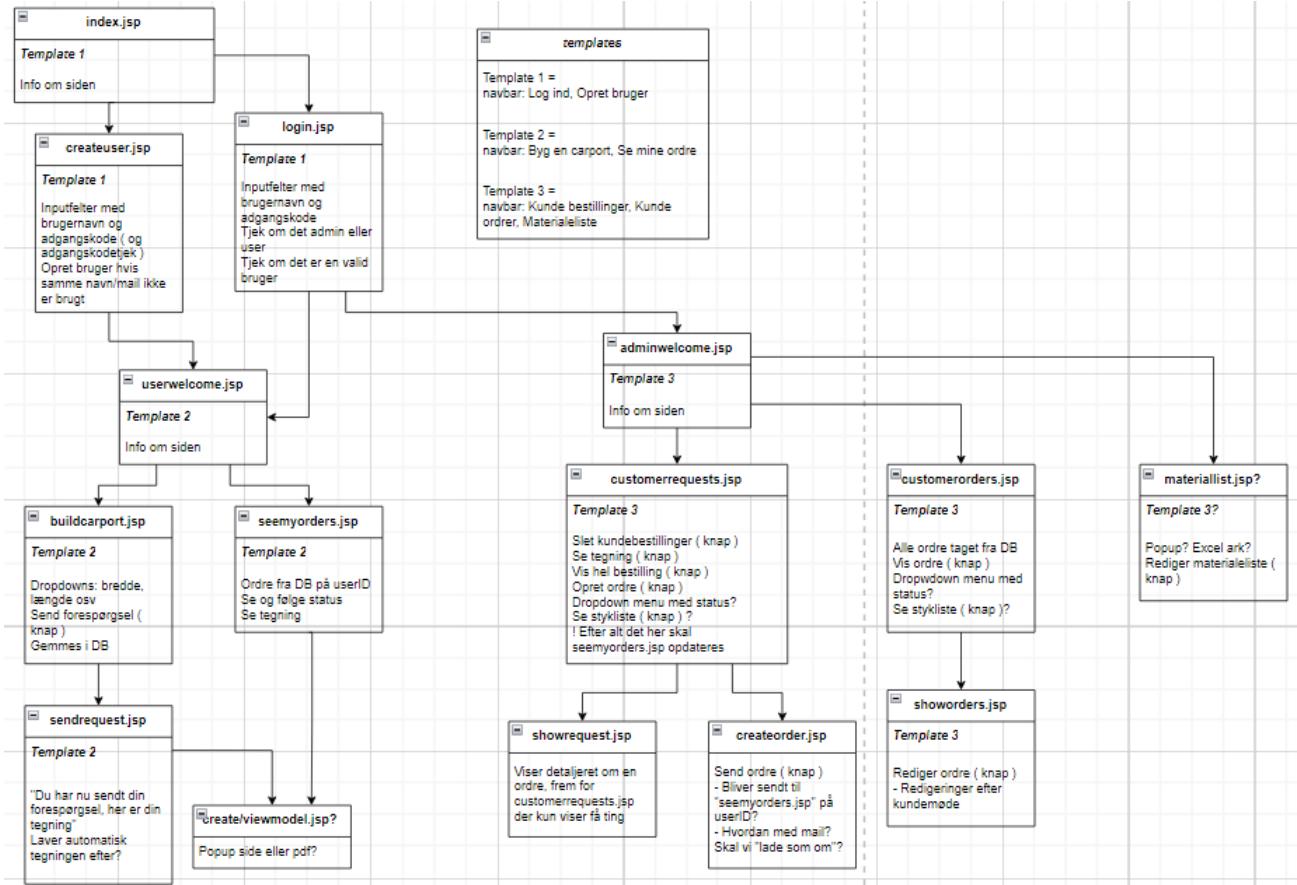
11.7. SWOT



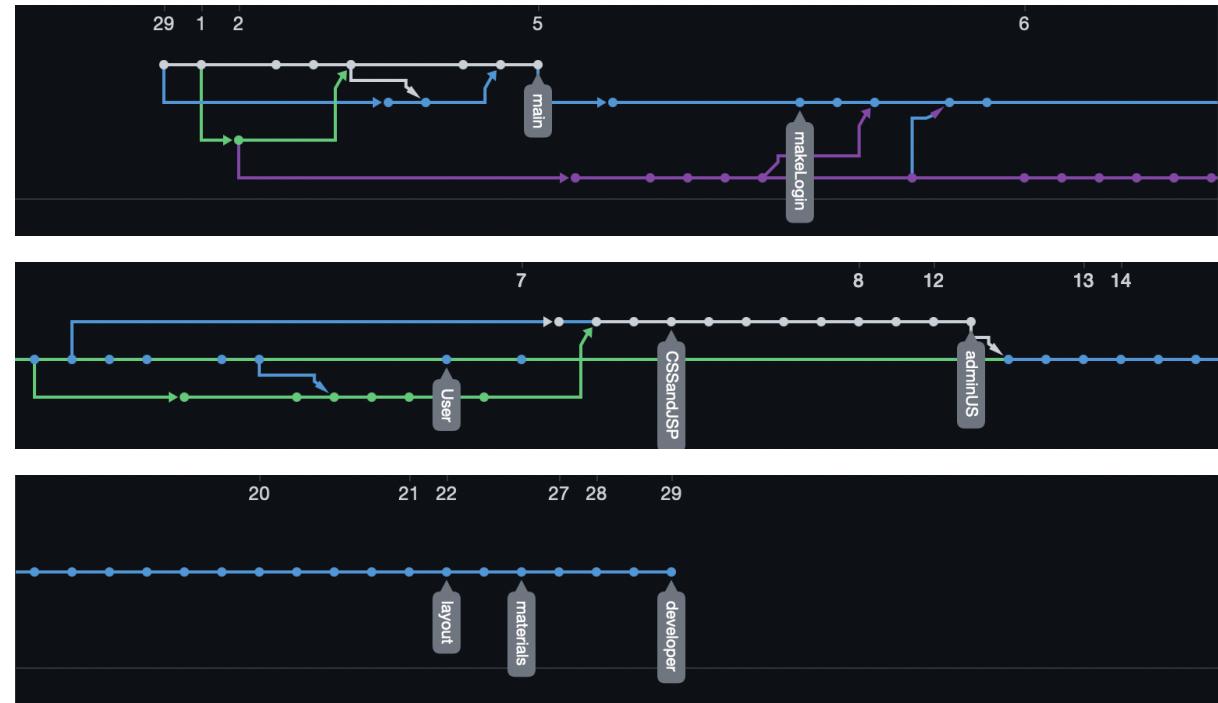
11.8. Sekvensdiagram



11.9. Brainstorm



12.0. GIT



```
* 9d1leaf (HEAD -> developer, origin/developer) adds sequence diagrams to documentation folder
* 8187a5e adds the last mats in calculator and in bom
* 6c11788 updates materiallist, adds more materials
* 16fdf30 Fixes update material and adds deleteorder
* faeec4a Showing materiallist in jsp and adds mappers accordingly
* f3f1e4f (origin/layout, layout) adds measurements to SVG arrow but bug with sideSVG
* 4b1fd7a adds full shed drawing
* c00cc4b adds shed
* 1d0b9b9 fixes problem with dashedline and arrowlines
* 590bf2c adds arrowheads to line template and placeholder text for lines(NEEDS FIXING)
* 1aa35f2 adds lines to svgTop, changes in buildcarport form
```

```
* 6cd4aea (origin/adminUS) Attempts fix for deletebutton crashing sql workbench, not deleting from carport schema
* b324c37 refactor show order and requests methods, adds delete function and sql methods
* 176b4f7 re-iterates sql statement in ordermapper with correct syntax
* fadd556 adds statusname to order object
* b6b938d adds methods to allrequestsfromcustomers.jsp and corresponding servlet
* 95204ff changes to the way we read data from carport and order DB
* 9ab8c02 tries to fix relation: servlet and jsp
* baad826 adds readCarport and readOrder methods in MyOrders
* ccc2283 (origin/CSandJSP) updates form in buildcarport.jsp to be responsive and changes labels to match database variables
* 2565d0a fixes spacing in bottom from button to footer on buildcarport.jsp
* 4e832a0 fix merge from adminUS
\ \
| * 5d55ca3 adds orderfacade and ordermapper
| * 9b45fae adds carportID to order entity
| * 959f94d adds Order Entity
| * a817bc9 Merge branch 'User' of https://github.com/thejamiegc/Fog_Carport into adminUS
| | \
| | * 1df8f87a adds bootstrap to allrequestsfromcustomers.jsp and adds navigation links
| | * 15ceea78 adds Navigation servlets for admin accessible pages
| | * cb05d0b up to date with developer branch
* || * b822c7c adds interface for materials and entities for beams, poles, rafter
* || * b83ae7c (origin/User) changes placement of labels in buildcarport.jsp
|| |
|| |
* || 634172e changes boolean on shed to an int in createCarport methods facade + mapper
* || df83ad6 creates classes Carport mapper + facade
* || 7f1d4b4 creates carport entity
* || 27c8868 creates empty servlets: BuildCarport + MyOrders
|| |
|| * d59c785 fixes path on link to servlet createUser
/ /
```