Fog Carport - 2. Semester Eksamen

Gruppe D
02.01.2023

Aleks Buha
cph-ab279@cphbusiness.dk

Pelle Dahl
cph-pd94@cphbusiness.dk

Demovideo:
https://youtu.be/FQYbIq7JCNU

Droplet:
http://164.92.135.211:8080/carport/index.jsp

Login til hjemmesiden:
Bruger: Kunde, Brugernavn: Aleks, Password: 1234
Bruger: Medarbejder, Brugernavn: Admin, Password: 1234

Indholdsfortegnelse

1. Links

I dette afsnit finder man alle de links til de forskellige sider som er relevante for dette projekt.

Github:
https://github.com/Aleks1075/carport.git

Droplet:
http://164.92.135.211:8080/carport/index.jsp

Figma:
https://www.figma.com/file/qQCRWQkFnzAqMlWDVGPR8o/Carport?node-id=0%3A1&t=97MAIPi9pAXGP0cS-1

Demovideo:
https://youtu.be/FQYbIq7JCNU

Pitchvideo:
https://youtu.be/BiWScC1DgPY

Kanban:
Uge 48:
https://trello.com/invite/b/aFoXw5B3/ATTId4a3c92825fe8bbd81fa5add6e8de3cbB333636D/kanban-fog-projekt-uge-48

Uge 49:
https://trello.com/invite/b/HMHtOm4j/ATTIb28b195e309756365757ad23d7ff8867367E17D3/fog-projekt-uge-49

Uge 50:
https://trello.com/invite/b/HgOPaaAA/ATTIe6a9986ecd41564ad50fcf2dfa73607f834F2267/fog-projekt-uge-50

Uge 51:
https://trello.com/invite/b/SPwOfFz2/ATTIb1aeba89205e56c901a9c8f380e1c6dbCC338E0C/fog-projekt-uge-51

Klassediagram:
https://docs.google.com/document/d/1J78Ri6tonli-eU2FMVyElhojd8UPjr2-QN3oRid7wwU/edit?usp=sharing

Logbog:
https://docs.google.com/document/d/1_DUav6CR_X9e_U5br7ZeKf3H71xZTTPnoXLVLlHUsSo/edit?usp=sharing

2. Indledning

I dette projekt arbejdede vi på at udvikle en dynamisk webapplikation til virksomheden Johannesfog, som handler om salg af carport på specialmål. Vi skulle blandt andet udvikle en beregningsmotor som kan håndtere de forskellige dimensioner på carporten som kunden vælger. Projektet gik ud på at analysere og imødekomme kundens krav og arbejde efter Kanban principper og skabe userstories med estimater og acceptkriterier.
Løsningen var baseret på en MySQL database og afspejlede en flerlagsarkitektur, der kørte på en Java-server og blev bygget ved hjælp af Java-klasser, servlets og JSP-sider.
Det blev også implementeret i skyen, og kildekoden er tilgængelig i et GitHub-repository.
I løbet af den fire uger lange projektperiode, var der fire møder med lærerene, der fungerede som repræsentanter for kunden.
Vi tog notater og referater fra disse møder og arbejdede systematisk, når vi stødte på problemer.
Projektet var en vigtig læringsmulighed, med det formål at opnå ny viden og erfaring inden for kernefagene på andet semester.
Det endelige produkt for dette projekt inkluderer en fungerende webapplikation og en rapport, der dokumenterer processen.

3. Baggrund

Johannes Fog er en velfungerende og godt etableret virksomhed som blev grundlagt i 1920. Siden da har de haft mange succeser og har derfor udvidet flere gange igennem årene. I dag ligger der 9 trælaster & byggecentre fordelt over Sjælland og Bolig & Designhuset i Lyngby. De har også en fungerende IT system men den lever ikke længere op til nutidens krav og der kommer vi på banen.

I systemet som vi skal lave, skal man kunne registrere sig og logge ind enten som en kunde eller som en administrator.

Kunden skal kunne sende forespørgsel på en carport, hvor de vælger både bredde og længde. Udover det, skal kunden kunne se deres igangværende forespørgsler, gemmeførte ordrer, tegninger af carporten og ordrer som skal betales og dermed efterfølgende kunne se stykliste samt tegning af carporten. Og det skal være muligt for kunden at se og rette i sin profil sådan at den er altid opdateret med de rigtige informationer.

Administrator skal kunne se alle ordrer, se forespørgsler på de enkelte ordrer og kunne godkendte en ordre sådan at den er klar til kunden at betale. Samtidig skal der genereres en stykliste på carporten, som efter betaling kan ses på kundernes side. Administrator skal også kunne slette en ordre fra systemet. Udover det, skal man også kunne se alle kunder som er registreret i systemet, sådan at man kan komme i kontakt med dem hvis nødvendigt.
Til sidst, skal en administrator også kunne se alle materialer som er på lageret. Der skal man kunne tilføje ny materiale, rette i pris og fjerne materiale.

4. Virksomheden/Forretningsforståelse

SWOT

	Usually Positive	Usually Negative
	Strengths	**Weaknesses**
Internal	• Vi er ærlige og direkte. Integritet er alt for os. Følelsesmæssig manipulation, tankespil og trøstende løgne går alle imod vores præference for at håndtere virkeligheden i de situationer, vi møder med almindelig ærlighed. • Vi er viljestærke og samvittighedsfulde. Vi legemliggør også denne integritet i vores handlinger, arbejder hårdt og forbliver fokuseret på vores mål. Tålmodige og beslutsomme, vi opfylder vores forpligtelser, punktum. • Vi er meget ansvarlige. Vores ord er et løfte, og et løfte betyder alt. Vi vil hellere udmatte os selv med ekstra dage og mistet søvn end at undlade at levere de resultater, vi sagde, vi ville. Loyalitet er en stærk følelse for os, og vi opfylder vores pligter over for de mennesker og den gruppe, vi har forpligtet os til. • Vi er rolige og praktiske. Ingen af vores løfter ville betyde meget, hvis vi mistede besindelsen og brød sammen ved hvert tegn på modgang - vi holder fødderne på jorden og træffer klare, rationelle beslutninger. Vores gruppemedlemmers præferencer er en faktor, der skal tages i betragtning i denne proces, og vi arbejder på at få mest muligt ud af individuelle karakteristika, men disse beslutninger træffes med effektivitet i tankerne mere end empati. Det samme gælder kritik af andre og os selv. • Vi skaber og håndhæver orden. Det primære mål for os er at være effektive i det, vi har valgt at gøre, og det tror vi bedst opnås, når alle involverede ved præcis, hvad der foregår og hvorfor. Uklare retningslinjer og mennesker, der bryder etablerede regler, underminerer denne indsats og tolereres sjældent af os. Struktur og regler fremmer pålidelighed; kaos skaber uforudsete tilbageslag og overskredne deadlines.	• Vi kan være stædige. Fakta er fakta, og vi har en tendens til at modstå enhver ny idé, der ikke understøttes af dem. Denne faktuelle beslutningsproces gør det også svært for os at acceptere, at vi tog fejl i noget – men alle kan gå glip af en detalje, også os. • Til tider kan vi uretfærdigt dømme folk, der ikke kan matche vores strenge selvkontrol - i mistanke om, at nogen er doven eller uærlig, når vedkommende måske faktisk kæmper med andre udfordringer. • Vi er ufølsomme nogle gange. Selvom vi ikke er bevidst hårde, sårer vi ofte mere følsomme personers følelser med det simple mantra, at ærlighed er den bedste politik. Vi kan tage hensyn til følelser, men egentlig kun i det omfang, vi kan bestemme den mest effektive måde at sige det, der skal siges. • Vi foretrækker at gøre ting efter bogen. Vi tror på, at tingene fungerer bedst med klart definerede regler, men det gør os tilbageholdende med at bøje disse regler eller prøve nye ting, selv når ulempen er minimal. • Vi er af og til fordømmende. Meninger er meninger, og fakta er fakta, og det er usandsynligt, at vi respekterer folk, der er uenige i disse fakta, eller især dem, der bevidst forbliver uvidende om dem. • Vi bebrejder ofte os selv urimeligt. Vi tror ofte, at vi er de eneste, der pålideligt kan gennemskue projekter. Så når vi belaster os selv med ekstra arbejde og ansvar, afviser gode intentioner og brugbare ideer, rammer vi før eller siden et vendepunkt, hvor vi simpelthen ikke kan levere. Da vi har lagt hele ansvaret på os selv, mener vi, at ansvaret for fiasko er vores alene at bære.

	Opportunities	Threats
External	• Når det kommer til gruppearbejde, er vi begge nærmest en stereotype af den klassiske hårdtarbejdende, pligtopfyldende elev. Vi søger struktur og klart definerede regler og respekt. Ansvar er ikke en byrde for os, det er den tillid, der er blevet vist os, en mulighed for endnu en gang at bevise, at vi er den rette person til projektet. • Vi tørster begge efter ansvar. Vi kan kompetent tackle ethvert projekt, der kommer med en instruktion. • Udover vores stædighed, eller måske på grund af det, er vi muligvis nogle af de mest produktive elever. Punktlighed vil næppe nogensinde være et problem for os, hverken i forhold til at komme til tiden på skolen eller i forhold til at overholde projektdeadlines. • Ingen kan stoles mere på for at sikre, at projekter bliver afsluttet til tiden og efter bogen, end os. Stille og metodiske holder vi os kolde, når det bliver hårdt, men forventer, at hinanden deler vores tilgang. • Vi sætter pris på ro og pålidelighed, når vi arbejder, og den nemmeste måde at ske på er, at vi simpelthen arbejder hver for sig, men sammen. Innovationer, brainstorming, teorier og nye ideer forstyrrer alle denne tilstand. • Vi elsker ansvar og den magt, der følger med det. Vi presser os selv hårdt for at opfylde vores forpligtelser, vi går regelmæssigt ud over pligten og forventer, at hinanden handler med samme niveau af dedikation.	• På grund af vores store ansvars lyst gør det, at vi ikke er villige til at opgive ansvaret til den anden i gruppen, selv når vi er overbebyrdede, eller når den anden er bedre egnet til opgaven. • Vi tager vores skolearbejde, vores forpligtelser og virkelig alle aspekter af vores liv ganske alvorligt, og vi arbejder hårdt på at planlægge godt og træffe smarte beslutninger. Men når tingene ikke går som planlagt, bebrejder vi os selv. Vi har svært ved blot at acceptere, at livet nogle gange ikke går som planlagt, og i stedet vil vi føle en følelse af personlig fiasko og nederlag – hvilket bliver en kilde for stress. • Vores engagement kan også blive en svaghed, som andre mennesker udnytter. Med vores stærke arbejdsmoral og pligtfølelse har vi rutinemæssigt oplevet, at vi påtager os andre menneskers ansvar. Selvom vi ikke brokker os over situationen, ender vi med at være udmattede eller modløse, hvis vi konstant forventes – eller påtager os – at gøre arbejdet for de andre gruppemedlemmer.

Refleksion på SWOT:

At skrive en SWOT-analyse baseret på os selv, før vi startede vores projekt, gjorde det muligt for os at identificere og evaluere vores styrker, svagheder, muligheder og trusler.

Vores styrker omfattede ærlighed, stærk vilje og samvittighedsfuldhed, ansvarlighed, ro og praktisk gennemførlighed og evnen til at skabe og håndhæve orden. Disse egenskaber hjalp os til at arbejde hårdt, holde fokus på vores mål, opfylde vores forpligtelser, træffe klare og rationelle beslutninger og opretholde struktur og effektivitet i vores projekt.

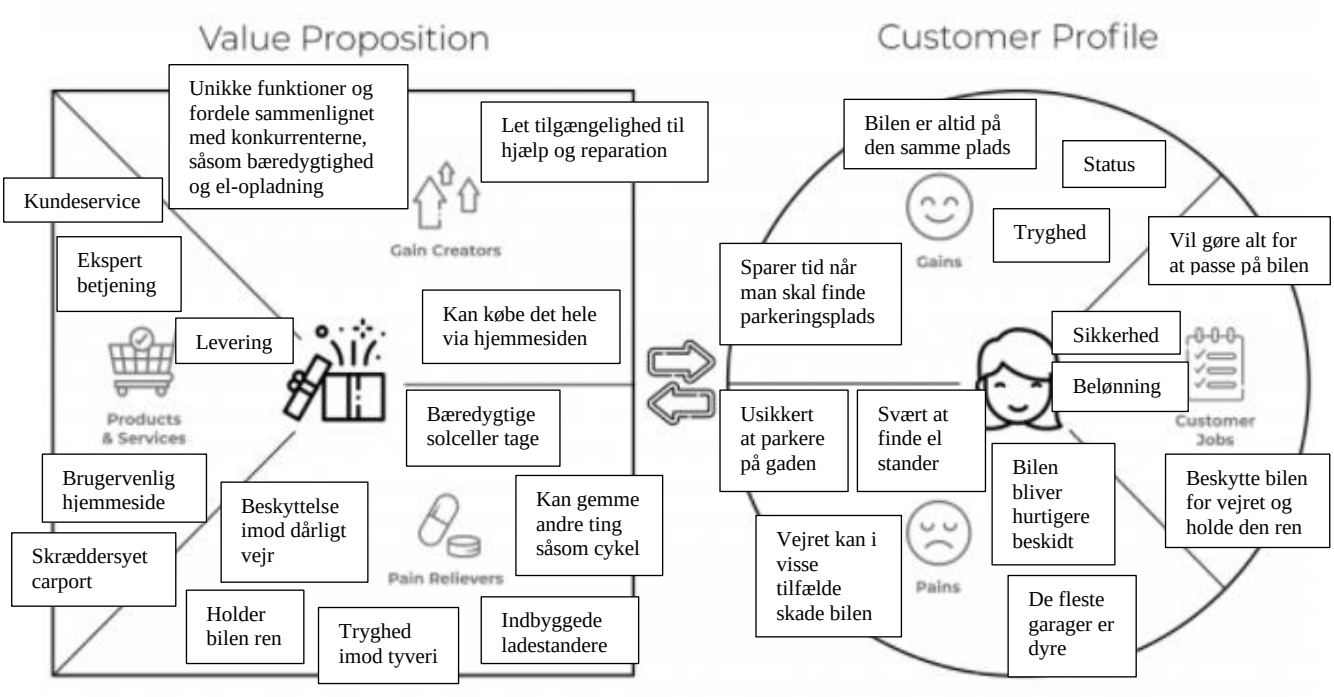
Men vi identificerede også adskillige svagheder, herunder stædighed og en tendens til at kunne være for hårde nogle gange overfor andre og dermed kommer der lidt ufølsomhed frem også. Vi erkendte, at disse svagheder potentielt kunne hindre vores fremskridt og påvirke vores gruppedynamik og var fast besluttet på at tale om ting sådan at vi undgår konflikter.

Med hensyn til muligheder så vi potentialet til at bevise os selv som ansvarlige og pålidelige teammedlemmer og til at tackle enhver udfordring, der kom vores vej, med kompetence og effektivitet. Vi så også muligheden for at forbedre vores svagheder og lære af vores fejl.

Endelig identificerede vi trusler, der potentielt kunne hindre vores fremskridt, herunder eksterne faktorer som manglende ressourcer eller en stram deadline, samt interne udfordringer som personlige konflikter eller manglende kommunikation i gruppen.

Samlet set gav det at skrive en SWOT-analyse os mulighed for at reflektere over vores styrker og svagheder og identificere muligheder og trusler, som vi kunne løse for at blive mere succesfulde i vores projekt.

VPC



Refleksion på VPC:

Vi startede ud med at tænke bredt og hvordan vi kan skabe endnu mere værdi for både Fog og kunderne. Vi kom frem til et forslag hvor udover at tilbyde de eksisterende ting, vil vi gerne komme med nye ting såsom at implementere indbyggede ladestationer og bæredygtige solceller tage i fremtiden.

Fog er en virksomhed, der har til formål at levere overkommelige løsninger af høj kvalitet til bilejere, der ønsker at beskytte og vedligeholde deres køretøjer.

Deres carporte tilbyder også opbevaringsmuligheder, og de indbyggede EV-ladestationer og bæredygtige solcelletage tilføjer ekstra værdi og fordele for kunderne.

Virksomhedens brugervenlige hjemmeside og fremragende kundeservice bidrager yderligere til en positiv kundeoplevelse.

Fogs produkter og tjenester adresserer smertepunkterne ved at finde bilopbevaringsløsninger af høj kvalitet til en god pris, såvel som vanskelighederne med blandt andet, at beskytte bilen mod snavs og ridser og oplade et elektrisk køretøj.
Ved at adressere disse smertepunkter fungerer Fogs produkter og tjenester som "pain relievers" og "gain creators" for deres kunder, hvilket i sidste ende resulterer i en sikrere og mere bekvem oplevelse af bilejerskab.

5. Teknologi valg

Teknologier som blev brugt i dette projekt:

- IntelliJ IDEA 2021.2.4 (Ultimate Edition)
- MySQL Workbench Version 8.0.28
- Apache Tomcat 9.0.67
- Java version 19
- Corretto 19
- JDBC MySQL connector Version 8.0.30
- JSTL Version 1.2
- Bootstrap 5.0
- Digital Ocean – Linux Ubuntu droplet server

Projektet tager udgangspunkt i denne startkode:
https://github.com/jonbertelsen/startcode_2sem_2022.git

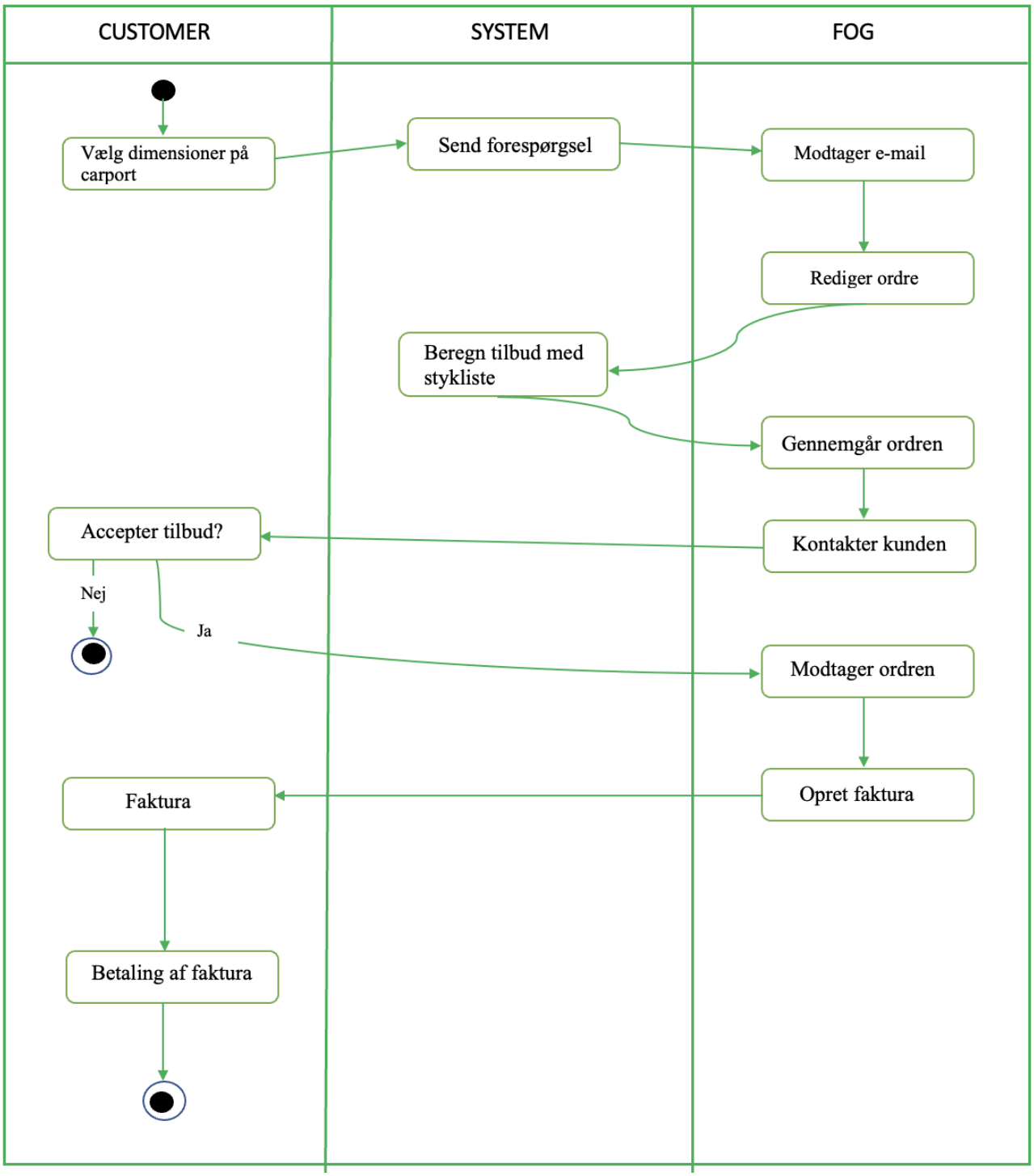
6. Krav

Krav afsnit er delt op i aktivitetsdiagrammer som hører under kapitel 7 "Arbejdsgange der skal IT-støttes" og op i user stories med acceptkriterier som hører under kapitel 8 "User stories".

7. Arbejdsgange der skal IT-støttes

Aktivitetsdiagram

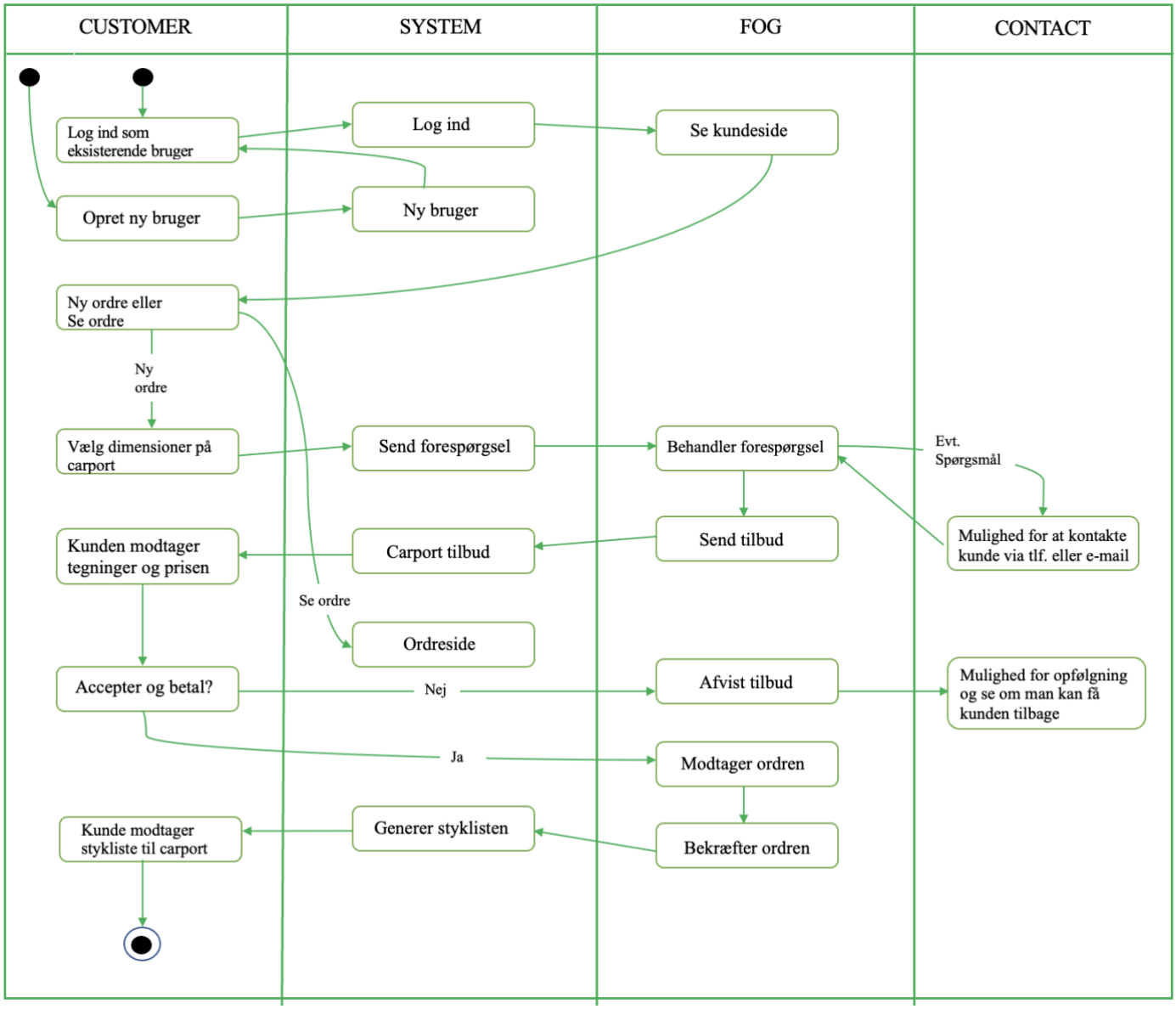
As-is



Ovenfor ses as-is diagram for den nuværende arbejdsgang hos Fog. Kunden sender en forespørgsel på en carport og den bliver modtaget via e-mail hos en administrator. Administrator kan derefter

kigge ordren igennem, kontakte kunden hvis nødvendigt og når alt er på plads, indsætter man alle informationer i et program som genererer tegning og stykliste. Til sidst, modtager kunden en faktura som skal betales og derefter modtager kunden byggevejledning med tegning og stykliste.

To-be



Ovenfor ses to-be diagram som repræsenterer den nye system som Fog skal implementere.
Kunden starter med at logge ind eller lave en bruger og bagefter sende en forespørgsel på de valgte dimensioner på carporten. Kunden kan nu se status og detaljer om sin ordre på ordresiden. Her kan de også vælge at se tegningen men selve styklisten bliver vist først når de har betalt og gennemført ordren.
Administrator skal kunne se alle de ordrer som er registreret i systemet, se detaljer om de enkelte

ordrer, tegninger af carporten og man skal også kunne godkendte en ordre, sådan at kunden kan betale og gennemføre den. Der skal også nævnes at som administrator skal man kunne have mulighed for at kontakte kunden i tilfælde af spørgsmål til deres ordre, opfølgning på en afvist ordre og feedback.

8. User stories

Nedenfor kan man se alle user stories samt acceptancekriterier som blev lavet og gennemført. De er opdelt i to tabeller for at skabe bedre overblik. Den første tabel indeholder alle user stories og acceptancekriterier fra kundens perspektiv imens anden tabel er fra administrators perspektiv.

Estimat på de forskellige user stories og acceptancekriterier er opdelt i tre størrelser.

S = 1 dag

M = 2-4 dage

L = 5-7 dage

Fra kunde perspektiv			
Str.	**Nr.**	**Funktion**	**User Story og Acceptance Criteria**
S	**1**	Lave en bruger	US: **Som** en kunde, **vil** jeg oprette en bruger profil, **så** jeg kan bestille skræddersyet carport
			AC: **Givet at** en bruger skal have en profil for at kunne bestille, **når** kunden prøver at oprette en profil, **så** bliver brugeren henvist til en opret bruger side
S	**2**	Log ind	US: **Som** en kunde, **vil** jeg gerne kunne logge ind, **så** jeg kan bestille, se og acceptere min ordre
			AC: **Givet at** en kunde er registreret, **når** man logger ind, **så** skal man videresendes til shopping siden
M	**3**	Bestille en carport	US: **Som** en kunde, **vil** jeg anmode om skræddersyet carport, **så** at jeg kan modtage et tilbud fra Fog
			AC: **Givet at** kunden har valgt carport dimensioner, **når** man sender en forespørgsel, **så** vil den forespørgsel lande hos Fog
M	**4**	Se ordren	US: **Som** en kunde, **vil** jeg gerne se min ordre, **så** at jeg kan følge op på status
			AC: **Givet at** kunden har lagt en ordre, **når** man gerne vil se detaljer, **så** bliver man henvist til "Se bestilling" siden
S	**5**	Betale ordren	US: **Som** en kunde, **vil** jeg acceptere og betale en bestilling, **så** jeg kan modtage min stykliste

			AC: **Givet at** en bestilling er aktiv, **når** kunden har betalt og gennemført en bestilling, **så** får kunden adgang til styklisten
L	**6**	Se tegning	US: **Som** en kunde, **vil** jeg se tegning af min bestilte carport, **så** jeg kan hvordan den ser ud samt bygge ud fra denne
			AC: **Givet at** kunden har lagt en ordre, **når** kunden ønsker at se tegning, **så** gå til "Se bestilling" siden og få tegning fremvist
L	**7**	Se styklisten	US: **Som** en kunde, **vil** jeg se stykliste til den valgte ordre, **så** jeg kan se hvad der skal bruges til at bygge min carport
			AC: **Givet at** kunden har betalt for ordren, **når** kunden ønsker at se styklisten, **så** gå til "Se bestilling" siden og få styklisten fremvist
S	**8**	Se "Om os" siden	US: **Som** kunde, **vil** jeg læse om Fog, **så** jeg kan få en bedre billede af virksomheden
			AC: **Givet at** man skal have mere informationer om Fog, **når** kunden prøver at undersøge nærmere, **så** har man adgang til "Om os" siden
S	**9**	Kontakt Fog kundeservice	US: **Som** en kunde, **vil** jeg komme i kontakt med Fog kundeservice, **så** jeg kan stille spørgsmål og få hjælp til at lægge en ordre
			AC: **Givet at** kunden har spørgsmål/brug for hjælp, **når** kunden prøver at komme i kontakt, **så** kan man kontakte kundeservice via telefon eller e-mail
M	**10**	Se din profil	US: **Som** en kunde, **vil** jeg gerne se min profil, **så** jeg kan tjekke eller rette i mine oplysninger
			AC: **Givet at** kunden har en profil og er logget ind, **når** kunden ønsker at se informationer om sig selv, **så** bliver man henvist til profil siden
S	**11**	Skift kodeord	US: **Som** en kunde, **vil** jeg gerne se skifte mit kodeord, **så** jeg kan tilknytte et nyt kodeord til min profil
			AC: **Givet at** kunden har en profil og er logget ind, **når** kunden ønsker at skifte kodeordet, **så** bliver man henvist til profil siden
S	**12**	Opdater dine oplysninger	US: **Som** en kunde, **vil** jeg gerne opdatere mine oplysninger, **så** jeg kan tilknytte korrekte informationer til min profil
			AC: **Givet at** kunden har en profil og er logget ind, **når** kunden ønsker at rette i sine oplysninger, **så** bliver man henvist til profil siden

Fra admin perspektiv			
Str.	**Nr.**	**Funktion**	**User Story og Acceptance Criteria**

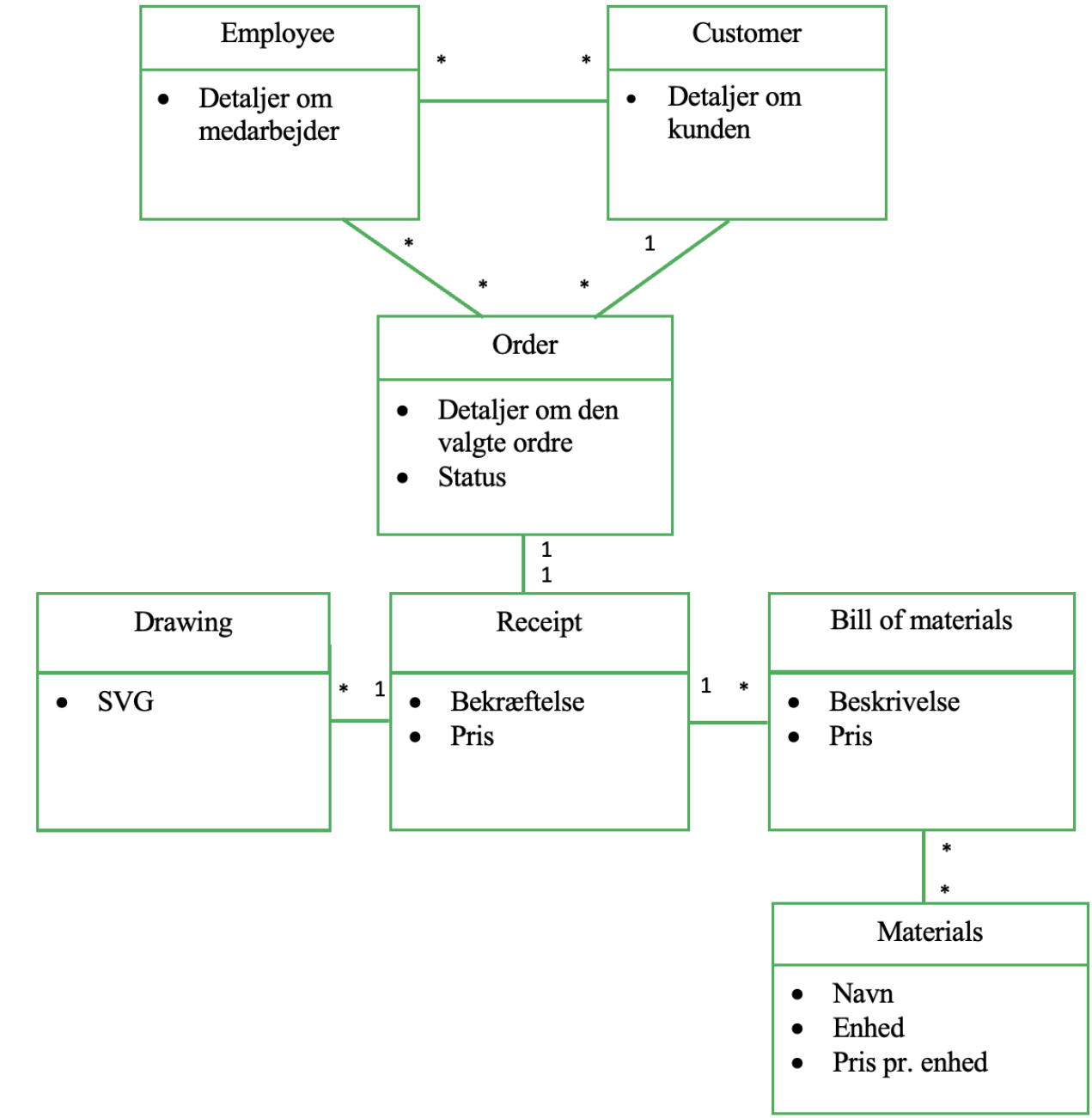
S	**13**	Log ind	US: **Som** en medarbejder, **vil** jeg gerne logge ind, **så** jeg kan se alle forespørgsler og ordrer
			AC: **Givet at** en medarbejder har en admin bruger, **når** man logger ind, **så** vil man lande på oversigt siden
S	**14**	Lav ny admin bruger	US: **Som** ny medarbejder, **vil** jeg oprette en ny admin bruger i systemet, **så** jeg kan lave mit arbejde
			AC: **Givet at** en eksisterende medarbejder opretter en ny admin bruger, **når** man skal lave oprette en ny admin bruger, **så** bliver den eksisterende medarbejder henvist til "Opret ny admin" siden
M	**15**	Se alle ordrer	US: **Som** en medarbejder, **vil** jeg gerne se alle ordrer ligegyldig ordrestatus, **så** jeg har overblik og kontrol over ordrer
			AC: **Givet at** en medarbejder er logget ind som admin, **når** man ønsker at se alle ordre, **så** bliver man henvist til ordreoversigt siden
S	**16**	Se materialeliste	US: **Som** en medarbejder, **vil** jeg gerne se materialeliste, **så** jeg har overblik over materialer
			AC: **Givet at** materialelisten eksisterer, **når** en medarbejder vil se materialeliste, **så** gør man det via "Materialer" siden
M	**17**	Opdater, slet og tilføj i materialeliste	US: **Som** en medarbejder, **vil** jeg gerne opdatere, slette og/eller tilføje materialer til materialeliste, **så** jeg altid har en opdateret liste
			AC: **Givet at** materialelisten eksisterer, **når** en medarbejder vil opdatere, slette og/eller tilføje materialer til materialeliste, **så** gør man det via "Materialer" siden
S	**18**	Se alle kunder	US: **Som** en medarbejder, **vil** jeg gerne se alle informationer på Fog kunder, **så** jeg har overblik og mulighed for kontakt til kunder
			AC: **Givet at** kundeliste eksisterer, **når** man ønsker at se kundeliste eller at komme i kontakt med kunder, **så** bliver man henvist til "Kunder" siden
M	**19**	Slet en kunde	US: **Som** en medarbejder, **vil** jeg gerne slette en kunde fra systemet, **så** jeg altid har en opdateret kundeliste
			AC: **Givet at** kundeliste eksisterer, **når** en medarbejder ønsker at slette en kunde, **så** gør man det via "Kunder" siden
S	**20**	Se ordren	US: **Som** en medarbejder, **vil** jeg gerne se kundens ordre, **så** at jeg kan se indholdet i kundens bestilling
			AC: **Givet at** der er lagt en ordre, **når** medarbejderen gerne vil se detaljer, **så** bliver man henvist til "Se bestilling" siden.

L	**21**	Se tegning	US: **Som** en medarbejder, **vil** jeg se tegning af en kundens bestilte carport, **så** medarbejderen kan sikrer at tegning er korrekt
			AC: **Givet at** der er lagt en ordre, **når** medarbejderen ønsker at se tegning, **så** gå til "Se tegning" siden og få tegning fremvist
S	**22**	Godkend ordren	US: **Som** en medarbejder, **vil** jeg gerne godkende en ordre, **så** kunden kan betale og gennemføre sin ordre
			AC: **Givet at** der er lagt en ordre, **når** medarbejderen ønsker at godkende en ordre, **så** gør man det via "Bekræft" knappen i "Alle ordrer" siden
S	**23**	Slet ordren	US: **Som** en medarbejder, **vil** jeg gerne slette en ordre, **så** jeg har en ryddelig ordreliste
			AC: **Givet at** der er lagt en ordre, **når** medarbejderen ønsker at slette en ordre, **så** gør man det via "Fjern" knappen i "Alle ordrer" siden
L	**24**	Prisberegning af carport	US: **Som** en medarbejder, **vil** jeg gerne beregne prisen på en ordre, **så** at jeg kan oplyse en pris til kunden
			AC: **Givet at** der er lagt en ordre, **når** medarbejderen ønsker at oplyse en pris til kunden, **så** beregn carports pris ud fra de valgte dimensioner
L	**25**	Generer stykliste til carport	US: **Som** en medarbejder, **vil** jeg gerne generere en stykliste, **så** kunden kan modtage den
			AC: **Givet at** kunden har gennemført en ordre, **når** medarbejderen ønsker at generere en stykliste, **så** beregn materialer som skal bruges til carporten efter de valgte dimensioner

9. Domæne model, EER diagram & Klassediagram

I denne afsnit har vi de diagrammer som skaber grundlag for vores system.

Domæne model

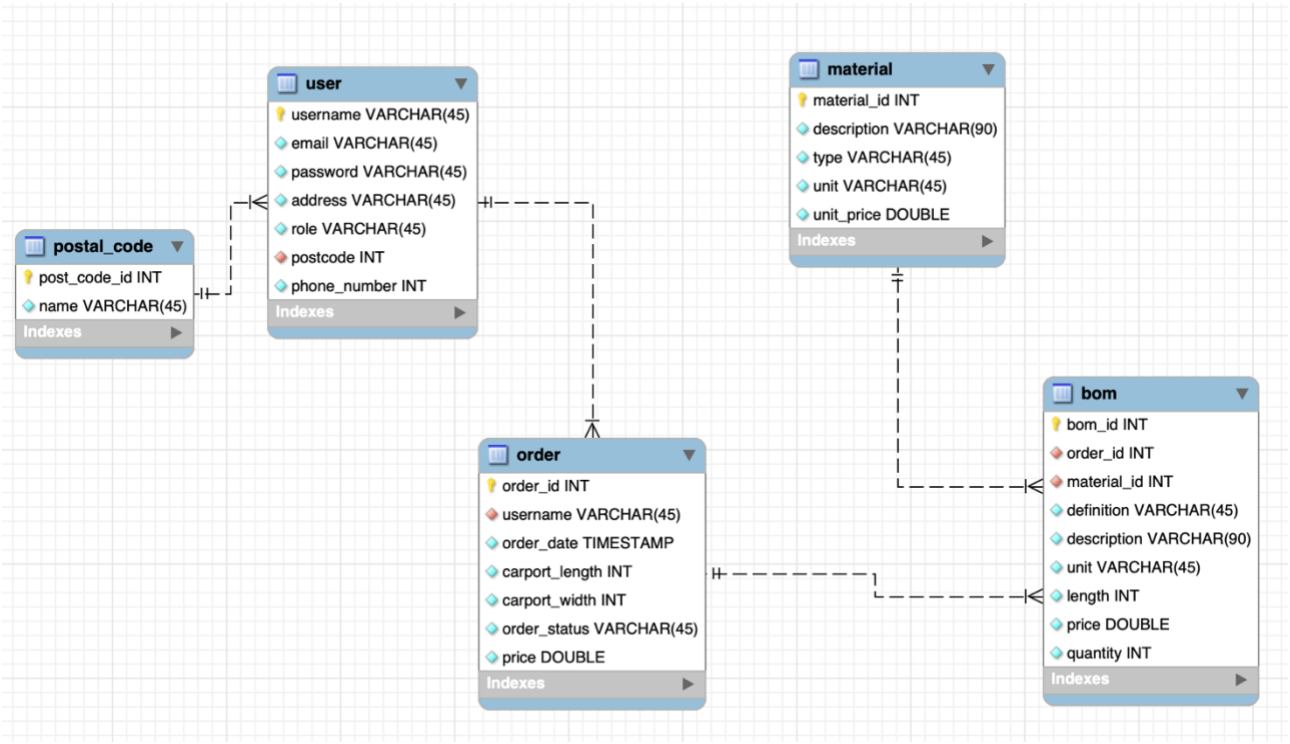


Vores domænemodel er lavet ud fra en vinkel, hvor kunden kommer ind i Fog butik og har mulighed for at købe alle de ting som Fog tilbyder, dvs. ikke kun carport på specialmål men at man også kan købe/bestille andre ting.

En kunde kan derfor have flere ordrer imens en ordre kan være tilknyttet til en kunde da den er unik. Men en ordre kan have flere medarbejder tilknyttet, såsom rådgiver, kassepersonale og transport. En medarbejder kan også have flere ordrer da den kan være ansvarlig for flere ordrer på en gang eller overtage andres ordrer.

En kvittering kan have en ordre men den kan have flere tegninger (Oppefra og fra siden). Tegninger hører til en kvittering da den ikke kan eksistere uden den specifik kvittering, hvor der står specifikt mål. Kvittering kan også have flere styklister, hvis kunden bestiller flere carporte og andre ting. Imens stykliste ligesom tegning, hører til en kvittering da den afhænger af de mål som er lagt på kvittering. Til sidst, stykliste kan have flere materialer tilknyttet da den skal bruge højt sandsynligt mange forskellige materialer for at generere en stykliste. Og materiale kan være tilknyttet til så mange styklister som der er brug for.
Samlet set fandt vi domænemodellen som en nyttig visuel repræsentation af et system. Det hjalp os med at forstå relationerne og interaktionerne mellem forskellige entiteter i vores givne domæne. Et meget nyttigt værktøj til at afklare design og implementering af vores system. Det hjalp med at forbedre effektiviteten af programmerings processen.

EER diagram



Ovenfor ser man vores EER diagram som viser opsætning af vores database. Tanken var at gøre det enkelt, let at arbejde med samt brug af 3. normalform.

I fremtiden kunne man helt sikkert lave lidt flere tabeller som opfylder 3. normalform, da på nuværende tidspunkt er det kun postal_code tabel som gør det. Man kunne for eksempel adskille

role fra user tabellen og lave en role tabel for sig selv med relation til user tabellen. Man kunne også adskille unit fra material tabellen og lave en unit tabel som relaterer til material tabellen.

De tabeller som ikke bruger automatisk genereret ID som nøgle er ”postal_code” og ”user”. Hos postal_code bruges postnummer på den enkelte by som nøgle, da der ikke kan være to byer med det samme postnummer. Hos user har vi brugernavn som nøgle, da de forskellige brugere skal have unikt brugernavn som ikke kan gentages af andre ny brugere.

Primære nøgler er markeret på de attributter med gul nøgle. De attributter som har en rød farve, er fremmede nøgler. Disse referer til et ID som tilhørende tabel har en relation til. For eksempel ”postcode” i user tabellen er forbundet med ID tilhørende postal_code tabellen.

Forklaring af hver tabel:

Postal_code:

- Postnummer som primærnøgle
- Bynavn som hører til det postnummer

User:

User tabellen er blevet brugt til både kunde og admin.

- Primær nøgle som repræsenterer brugernavn og som man bruger til at logge ind på hjemmesiden
- Kodeord som bruges til at logge ind på hjemmesiden
- Adresse, sådan at man ved hvor man befinder sig
- Role, sådan at man ved hvem bruger systemet (Kunde eller admin)
- Fremmednøgle som repræsenterer postcode, sådan at man har mulighed for at vælge byen man bor i
- Telefonnummer, sådan at man kan komme i kontakt med vedkommende

Order:

Order tabellen er brugt til at modtage forespørgslen fra kunden når de skal vælge dimensioner på carport.

- Primær nøgle order_id er automatisk genereret
- Fremmenøgle username som henter brugernavn fra user tabellen, så vi ved hvem det er som laver ordren

- Dato, for at vide tidspunktet på hvornår ordren er lagt
- Længden på carporten, så man kender til den valgte længde
- Bredden på carporten, så man kender til den valgte bredde
- Status på ordren, som gør at man kan styre hvorhenne i processen, er ordren henne, afventer eller godkendt.
- Prisen, for at vise den totale pris på den valgte carport

Material:

Material tabellen har det formål, for at repræsentere alle de materialer som Fog har på sin lager.

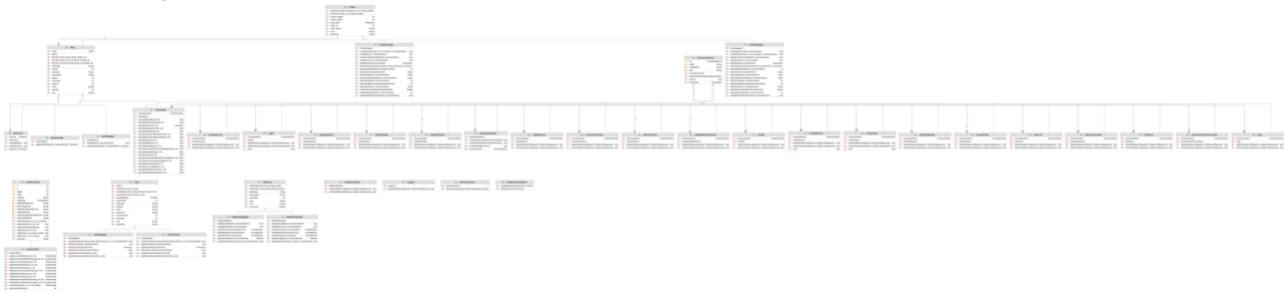
- Primær nøgle material_id som er automatisk genreret
- Description, som indeholder navn og beskrivelse på materiale
- Type fortæller hvilken slags materiale det er (Træ & tagplader eller beslag & skruer)
- Unit viser hvilken enhed materiale bliver leveret i, for eksempel ”Stk”
- Unit_price er lavet for at vise hvor meget de enkelte materialer koster, såsom en pakke af bundskruer. Når det kommer til træ, så viser den prisen per meter.

BOM:

BOM tabellen repræsenterer alle de linjer som findes på en stykliste.

- Primær nøgle bom_id bliver automatisk genereret
- Fremmede nøgle som henter order_id fra order tabellen, sådan at man ved hvilket ordre hører til styklisten
- Fremmede nøgle som henter material_id fra material tabellen, sådan at man ved hvilket materiale skal bruges til styklisten
- Definition, som beskriver hvordan materialer skal bruges
- Description, som indeholder navn og beskrivelse på materiale
- Unit viser hvilken enhed materiale bliver leveret i, for eksempel ”Stk”
- Length, som viser længden af materiale som skal bruges
- Price, som viser priser af de enkelte materialer afhængig af de valgte dimensioner på carporten
- Quantity, som viser antal af de materialer som man skal bruge for at bygge carporten

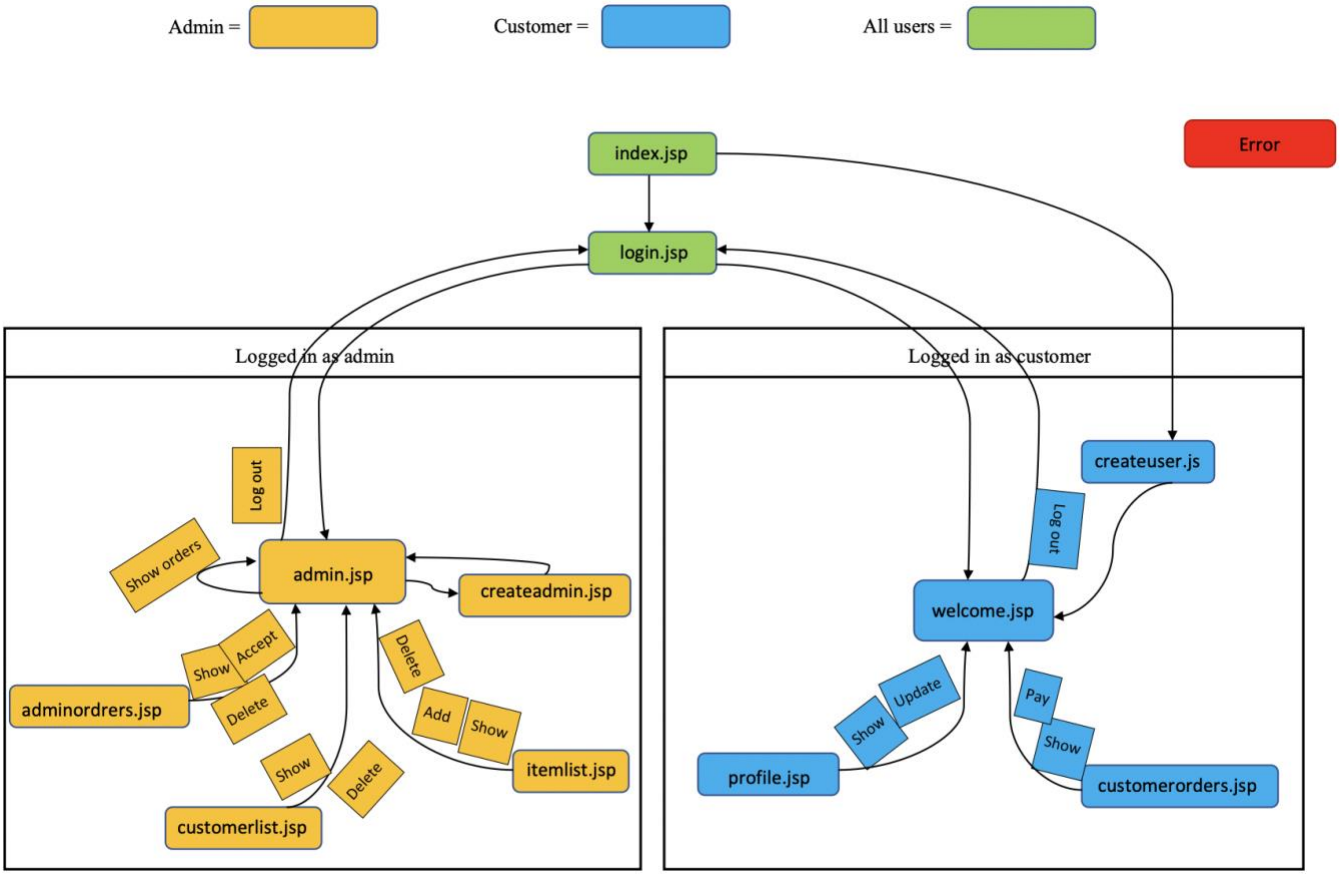
Klassediagram



Klassediagram set ovenfor viser hvordan hele er bygget op efter at vi er blevet færdige med vores projekt. Hvis man har lyst at kigge nærmere på, så har vi lavet en google docs side hvor vi har zoomet ind i klassediagrammet og lavet et par screenshots. Den link finder man under ”Klassediagram”, under Links afsnittet.

10. Navigationsdiagram & Figma

Navigationsdiagram



Navigationsdiagram set ovenfor viser den navigations flow vi har på vores hjemmeside og alt efter om man er logget ind som kunde eller som medarbejder, vises navigationsbar med funktioner tilhørende den bruger role.
Index.jsp og login.jsp sider tilhører både kunden og medarbejder. Når de går ind på hjemmesiden, lander de på index.jsp hvorfra de kan styre selv processen videre. For at kunne få adgang til de specifikke funktioner som kunden og medarbejder har adgang til, bliver de først nødt til at gøre det via login.jsp siden for at kunne logge ind.

En kunde som er logget ind, kaldes ”Customer” og har følgende muligheder:
Bestillingsside (welcome.jsp): Dette er den første side kunden lander på efter at have logget ind. Her kan de vælge de ønskede dimensioner på carport og sende forespørgsel afsted.
Profil (profile.jsp): Her kan en kunde se alle de brugeroplysninger som Fog har på den kunde. Kunden kan her vælge at opdatere sine brugeroplysninger eller blot lave en ny kodeord.
Ordrer (customerorders.jsp): Her kan kunde altid holde sig opdateret på de ordrer som er registreret på den specifik kunde. Udover at se detaljer om den valgte bestilling, når en ordre er godkendt fra admin, kan kunde gå videre ved at trykke på betal og gennemfør ordren.
Ny bruger (createuser.jsp): På denne side kan man oprette sig som en kunde hos Fog.

En medarbejder som er logget ind, kaldes ”Admin” og har følgende muligheder:
Forside (admin.jsp): Dette er en velkomst side for medarbejderen. Herfra kan de via navigationsbaren styre hvorhenne de vil videre på hjemmesiden.
Oversigt over alle ordrer (adminorders.jsp): Her kan man som admin holde styr på alle ordrer som er registreret i systemet. Man kan se en specifik ordre, slette en ordre og godkende en ordre.
Oversigt over alle kunder (customerlist.jsp): Her kan man finde alle de kunder som er registreret hos Fog, sådan at man kan komme i kontakt med dem. Udover det, har man også mulighed for at slette en kunde.
Oversigt over alle materialer (itemlist.jsp): Her kan man ændre pris, slette og tilføje ny materiale.
Ny admin bruger (createadmin.jsp): På denne side kan en eksisterende administrator oprette en ny admin til en ny medarbejder.

Navigationsdiagram hjalp os med at forstå flowet og sikre, at alle nødvendige sider var inkluderet på hjemmesiden.

Endelig hjalp navigationsdiagrammet os med at planlægge, hvordan vi ville strukturere vores kode. Det gav os et overblik over, hvordan de forskellige sider på hjemmesiden ville interagere og hjalp os med at planlægge, hvordan vi ville organisere vores servlets og JSP-sider.
Overordnet set var navigationsdiagrammet et nyttigt værktøj, der hjalp os med at strukturere vores projekt og sikre et klart overblik over de forskellige sider, der er tilgængelige for hver rolle, og hvordan de er forbundet.

Figma

For at se vores Figma løsning, kan man gå til afsnit ”Links”, hvor man kan finde link til vores Figma design under ”Figma”.
I vores projekt har vi brugt Figma til at designe og visualisere brugergrænseflader til vores hjemmeside. Figma er et cloud-baseret designværktøj, der giver os mulighed for at samarbejde om at skabe wireframes, prototyper og endelige designs til vores hjemmeside.
Vi fandt Figma særligt nyttigt, da det gjorde det nemt for os at dele og samarbejde om vores designs med vores lærer og med hinanden. Dette var især vigtigt, da vi arbejdede på at opfylde kundens krav og sikre, at vores hjemmesiden er brugervenlig og intuitiv.
Figma har også været nyttig for os, da den har en række værktøjer, der gør det nemt for os at designe og prototype vores brugergrænseflader. For eksempel brugte vi Figma træk-og-slip-funktion til at arrangere elementer på vores skærme og brugte deres forud designede elementer og komponenter for at spare tid og sikre, at vores design var professionelt og konsistent.
Alt i alt har Figma været en nyttig ressource for os i vores projekt, da det har gjort os i stand til at designe og visualisere vores brugergrænseflader på en effektiv og professionel måde. Det gjorde det også nemt for os at samarbejde om vores design og sikre, at vi opfyldte kundens behov og krav til vores webapplikation.

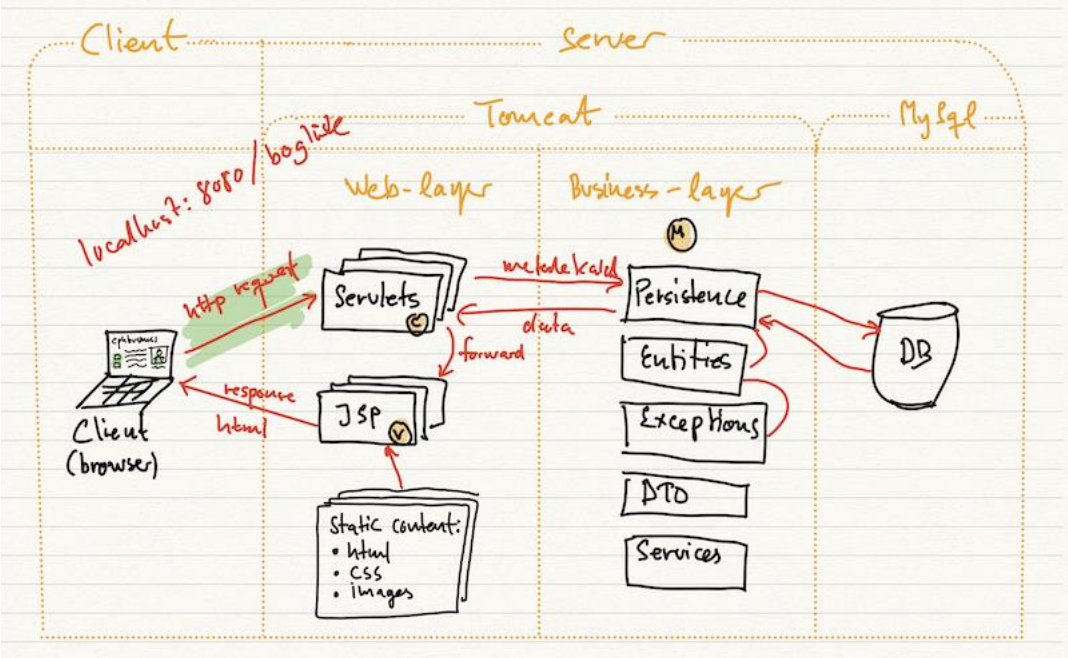
11. Valg af arkitektur

Vores kode tager udgangspunkt i starkoden som vi fik udleveret via denne link:
https://github.com/jonbertelsen/startcode_2sem_2022
Der er et par funktionaliteter lavet på forhånd i startkoden, såsom login og logout system, forsiden, genereret template, header og footer.

Vi gjorde brug af MVC arkitektur (Model-View-Controller), som adskiller forskellige lag i koden

sådan at hele kode oplevelse bliver mere overskueligt og lettere at finde rundt i. MVC er delt op i tre lag:

- **Controller,** som er ansvarlig for at behandle forespørgsel fra en klient og fortæller serveren hvad der skal ske efterfølgende. Her skal der ikke være særlig meget kode, den role er mere en bindeled mellem view og model. Der skal aldrig være direkte interaktion mellem view og model.
- **View** er det som bliver vist til klient. I denne projekts tilfælde vil det være selve hjemmesiden.
- **Model,** behandler alt logik i systemet. Den interagerer med logik og data, for eksempel alle de forskellige mappers som findes i vores projekt såsom UserMapper, OrderMapper, MaterialMapper osv.



Billedet ovenfor viser MVC arkitektur.

I vores startkode har vi opdeling i tre lag hvor man kan se brug af MVC arkitektur:

- **Control,** som behandler forespørgsler fra klient.
- **Model,** hvor alt vores logik kommer fra.
- **View,** inkluderer jsp sider, html, css. Alt det som kræves for at kunne vise hjemmesiden til klient.

Vi gjorde også brug forskellige patterns:

- **Singleton pattern,** som tillader kun en instans af en given klasse.

- **Facade pattern** gør koden meget mere simpelt at læse og den gemmer de store dele af koden bag en ”facade”. I vores projekt, mindsker facaden den overordnede kompleksitet og viser kun metoden og hvad den returnerer.

12. Særlige forhold

Sessions

Vi gjorde brug af RequestScope og SessionScope.
RequestScope brugte vi når vi skulle vise information eller data. Her behøves man ikke at gemme noget i en session, da vi kun skal bruge den enkelt gang. For eksempel, når man skal vise en materiale liste, kunde liste osv.
SessionScope brugte vi når for eksempel, en ”User” skal logge ind på systemet og bruge den. Vi er nødt til at bruge en session i denne tilfælde, da vi vil gerne have adgang til alle informationer på brugeren så længe sessionen kører.

Fejlhåndtering

Starkoden havde i forvejen håndtering af et par fejl såsom fejl ved log ind. Der bliver man sendt videre til en side hvor man får at vide at brugernavn eller kodeord er forkert. Vi har ikke brugt så meget tid på at håndtere exceptions. I de fleste tilfælde, har vi gjort sådan at man catcher fejl hvor vi får en besked om hvad der kan være for en fejl såsom SQLException.

Brugerinput validering

Vi har gjort sådan at en bruger skal udfylde alle felter eller i visse tilfælde skal man bruge tal i stedet for bogstaver, ellers bliver man sendt til en fejlside. Såsom hvis man glemmer at indtaste sin adresse eller hvis man kommer til at indtaste adresse i postnummer feltet.
En anden validering er når man skal vælge dimensioner på carporten, så skal begge to felter udfyldes.

Tegning og stykliste beregner

Vores system generer SVG tegninger ud fra de dimensioner som kunden har valgt på carporten. Der bliver genereret to tegninger, en set oppefra og en set fra siden.

Stykliste bliver beregnet og gereneret ud fra de dimensioner som kunden har valgt. Her bygger man styklisten op med de nødvendige materialer som skal bruges til at bygge carporten.

13. Udvalgte kodeeksempler

Beskrivelse af "calculateCarport"-metoden i vores Java-klasse "Calculator"

```
private static ConnectionPool connectionPool = ApplicationStart.getConnectionPool();

public List<Bom> calculateCarport(int width, int length) throws DatabaseException
{
    List<Bom> bomList = new ArrayList<>();

    bomList.add(calculatePosts(width, length));
    bomList.add(calculateStraps(width, length));
    bomList.add(calculateRafters(width, length));
    bomList.add(calculateUndersternBoardsFrontAndBack(width, length));
    bomList.add(calculateUndersternBoardsSides(width, length));
    bomList.add(calculateOversternBoardsFront(width, length));
    bomList.add(calculateOversternBoardsSides(width, length));
    bomList.add(calculateWaterBoardSides(width, length));
    bomList.add(calculateWaterBoardFront(width, length));
    bomList.add(calculateRoofPlates(width, length));
    bomList.add(calculateBottomScrews(width, length));
    bomList.add(calculateHulband(width, length));
    bomList.add(calculateUniversalRight(width, length));
    bomList.add(calculateUniversalLeft(width, length));
    bomList.add(calculateScrewsForSternAndWaterBoard(width, length));
    bomList.add(calculateHingeScrews(width, length));
    bomList.add(calculateBoardBolt(width, length));
    bomList.add(calculateSquareDiscs(width, length));

    return bomList;
}
```

Denne klasse giver en metode kaldet "calculateCarport", der tager to integers ind, bredde og længde, og returnerer en liste over BOM-objekter (Bill of Materials). Disse styklisteobjekter indeholder information om de materialer, der er nødvendige for at konstruere en carport med den angivne bredde og længde.

"calculateCarport"-metoden opretter først en tom liste over styklisteobjekter og tilføjer derefter resultaterne af forskellige beregningsmetoder, som hver returnerer et styklisteobjekt. Hver af disse beregningsmetoder er ansvarlige for at bestemme de nødvendige materialer til en specifik del af carporten og returnere et styklisteobjekt, der indeholder oplysninger om disse materialer.

For eksempel bruger "calculatePosts"-metoden MaterialFacade-klassen til at hente information om et materiale med et ID på 11 fra databasen og beregner derefter mængden, længden og prisen på dette materiale, der skal bruges til carportens stolper. Den opretter og returnerer derefter et styklisteobjekt med disse oplysninger.

Det er værd at bemærke, at flere af disse beregningsmetoder gør brug af klasserne "ConnectionPool" og "MaterialFacade", som importeres i begyndelsen af klassen "Calculator". Klassen "ConnectionPool" bruges til at administrere databaseforbindelser, mens klassen "MaterialFacade" giver metoder til at interagere med "Material"-entiteten i databasen.

Overordnet set tjener klassen "Calculator" som et nyttigt værktøj til at bestemme de nødvendige materialer til konstruktionen af en carport med en given bredde og længde. Dens forskellige beregningsmetoder giver en måde at opdele byggeprocessen i mindre, mere håndterbare stykker og bestemme de nødvendige materialer til hvert trin.

Beskrivelse af "calculatePosts"-metoden i vores Java-klasse "Calculator"

```
//Beregn stolper
public Bom calculatePosts(int width, int length) throws DatabaseException {
    Material material = MaterialFacade.getMaterialById( i: 11, connectionPool);
    int materialId = material.getMaterialId();
    String definition = "Stolpe nedgraves 90 cm. i jord";
    String description = material.getDescription();
    String unit = material.getUnit();
    int lengthOfPosts = 300;
    int quantity = 4;
    double price = (lengthOfPosts/100) * quantity * material.getUnit_price();

    Bom bom = new Bom(materialId, definition, description, unit, lengthOfPosts, price, quantity);
    return bom;
```

Denne metode er ansvarlig for at bestemme de nødvendige materialer til at konstruere stolperne i en carport.

Det første, metoden gør, er at hente information om et materiale med et ID på 11 ved hjælp af ”MaterialFacade”-klassens ”getMaterialById”-metode og ”ConnectionPool”-klassen. Dernæst sætter metoden flere lokale variabler ved hjælp af information fra ”material”-objektet. Disse variabler inkluderer ”materialId”, ”definition”, ”description” og ”unit”, som svarer til henholdsvis materialets ID, definition, beskrivelse og enhed. Metoden sætter også værdien af ”lengthOfPosts”-variablen til 300 og ”quantity”-variablen til 4.

Metoden beregner derefter prisen på det nødvendige materiale til stolperne ved at gange længden af stolperne (300), mængden (4) og enhedsprisen for materialet. Dette resultat gemmes i variablen ”pris”. Til sidst opretter metoden et nyt ”Bom”-objekt ved at bruge de lokale variabler ”materialId”, ”definition”, ”description”, ”unit”, ”lengthOfPosts”, ”price” og ”quantity” som argumenter og returnerer det.

Sammenfattende er ”calculatePosts”-metoden et nyttigt værktøj til at bestemme de nødvendige materialer til at konstruere stolperne i en carport. Den henter information om et bestemt materiale fra databasen og bruger den til at beregne mængden, længden og prisen på det materiale, der er nødvendigt for stolperne. Det returnerer derefter et ”Bom”-objekt, der indeholder denne information.

Beskrivelse af "calculateRafters"-metoden i vores Java-klasse "Calculator"

```
//Beregn spær
public Bom calculateRafters(int width, int length) throws DatabaseException {
    Material material = MaterialFacade.getMaterialById( i: 10, connectionPool);
    int materialId = material.getMaterialId();
    String definition = "Spær, monteres på rem";
    String description = material.getDescription();
    String unit = material.getUnit();
    int lengthOfRafters = width;
    //Beregner antal spær
    double numberOfRafter = Math.ceil(length / 55.0);
    double endRafter = numberOfRafter + 1;
    int quantity = (int) endRafter;
    //Beregner pris for spær
    double price = (lengthOfRafters/100) * quantity * material.getUnit_price();

    Bom bom = new Bom(materialId, definition, description, unit, lengthOfRafters, price, quantity);
    return bom;
}
```

Beregning af antallet og prisen på spær til konstruktion af en carport er en vigtig opgave, og calculateRafters-metoden i vores Calculator-klasse i Java er designet til netop det.

Metoden begynder med at hente et Material-objekt ved hjælp af MaterialFacade-klassen og materiale-ID 10. Den gemmer derefter materialId, definition, description og unit i variabler og indstiller længden af spærene til at være den samme som bredden af byggeprojekt. Dernæst beregner metoden antallet af spær, der er nødvendige for projektet, ved at dividere projektets længde med 55 og runde op til nærmeste hele tal. Den tilføjer derefter 1 til dette tal for at tage højde for ende spærret, og gemmer resultatet i mængdevariablen som et heltal.

Metoden beregner derefter prisen på spærene ved at gange spærrernes længde, antallet af spær og materialets enhedspris. Dette resultat gemmes i prisvariablen. Til sidst opretter metoden et nyt Bom-objekt ved hjælp af de variabler, den har defineret, og returnerer det. Bom-objektet repræsenterer en liste over nødvendige materialer til byggeprojektet, herunder materiale-id, definition, beskrivelse, enhed, længde af spær, pris og mængde.

Overordnet set er calculateRafters-metoden et nyttigt værktøj til nøjagtigt at beregne antallet og omkostningerne for spær, der er nødvendige for at konstruere en carport, ved hjælp af den angivne bredde og længde af projektet samt information om det anvendte materiale.

Beskrivelse af "calculateStraps"-metoden i vores Java-klasse "Calculator"

```
//Beregn remme
public Bom calculateStraps(int width, int length) throws DatabaseException {
    Material material = MaterialFacade.getMaterialById( i: 9, connectionPool);
    int materialId = material.getMaterialId();
    String definition = "Remme i sider, sædles ned i stolper";
    String description = material.getDescription();
    String unit = material.getUnit();
    int lengthOfStraps = 600;
    int quantity = 2;
    if (lengthOfStraps > 600) {
        quantity += 1;
    }
    double price = (lengthOfStraps/100) * quantity * material.getUnit_price();

    Bom bom = new Bom(materialId, definition, description, unit, lengthOfStraps, price, quantity);
    return bom;
}
```

”calculateStraps” metoden i vores Calculator-klasse i Java er et nyttigt værktøj til at bestemme antallet og omkostningerne for remme, der er nødvendige for at bygge en carport.

Metoden begynder med at hente et Material-objekt ved hjælp af MaterialFacade-klassen og materiale-ID 9. Den gemmer derefter materialId, definition, description og unit i variabler og indstiller længden af remmene til at være 600. Metoden sætter så mængdevariablen til 2, da der skal bruges 2 remme til hver carport. Hvis længden af remmene er større end 600, øger metoden mængden med 1 for at tage højde for den nødvendige ekstra rem.

Dernæst beregner metoden prisen på remmene ved at gange længden af remmene, antallet af remme og materialets enhedspris. Dette resultat gemmes i prisvariablen. Til sidst opretter metoden et nyt Bom-objekt ved hjælp af de variabler, den har defineret, og returnerer det. Bom-objektet repræsenterer en liste over nødvendige materialer til carporten, inklusive materiale-id, definition, beskrivelse, enhed, længde af remmene, pris og mængde.

Overordnet set er calculateStraps-metoden et nyttigt værktøj til nøjagtigt at beregne antallet og omkostningerne for remme, der er nødvendige til en carport, ved hjælp af information om det anvendte materiale og længden af de nødvendige remme.

Beskrivelse af "addMaterial"-metoden i vores servlet "AddMaterial"

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
{
    response.setCharacterEncoding("UTF-8");
    request.setCharacterEncoding("UTF-8");
    String description = request.getParameter( s: "description");
    String type = request.getParameter( s: "type");
    String unit = request.getParameter( s: "unit");
    double unit_price = Double.parseDouble(request.getParameter( s: "unit_price"));

    MaterialFacade.addMaterial(new Material(description, type, unit, unit_price), connectionPool);

    List<Material> materialList = null;
    try {
        materialList = MaterialFacade.getAllMaterials(connectionPool);
    } catch (DatabaseException e) {
        e.printStackTrace();
    }
    request.setAttribute( s: "materialList", materialList);

    List<Material> woodList = null;
    try {
        woodList = MaterialFacade.getAllWood(connectionPool);
    } catch (DatabaseException e) {
        e.printStackTrace();
    }
    request.setAttribute( s: "woodList", woodList);

    List<Material> accessoryList = null;
    try {
        accessoryList = MaterialFacade.getAllAccessory(connectionPool);
    } catch (DatabaseException e) {
        e.printStackTrace();
    }
    request.setAttribute( s: "accessoryList", accessoryList);

    request.getRequestDispatcher( s: "WEB-INF/itemlist.jsp").forward(request, response);
```

Vores AddMaterial-klasse er en Java-servlet, der håndterer HTTP-requests og -responses. Dets formål er at tilføje et nyt Material-objekt til en database og derefter videresende en anmodning til en JSP-side kaldet itemlist.jsp.

AddMaterial-klassen extends HttpServlet, hvilket betyder, at den er en underklasse af HttpServlet og kan override doGet- og doPost-metoderne. I dette tilfælde overrides doPost-metoden og bruges til at håndtere HTTP POST-anmodninger. DoPost-metoden indstiller først tegnkodningen for anmodningen og svaret til UTF-8, som gør det muligt for servlet'en at håndtere og vise tegn fra forskellige sprog korrekt. Den henter derefter flere parametre fra objektet, inklusive en description, type, unit og unit_price, og bruger dem til at oprette et nyt Materialeobjekt.

MaterialFacade-klassen bruges derefter til at tilføje det nye Material-objekt til databasen ved hjælp af addMaterial-metoden. MaterialFacade-klassen har også metoder til at hente alle materialer, alle træmaterialer og alle tilbehørsmaterialer fra databasen. Disse metoder kaldes i doPost-metoden, og resultaterne gemmes i request-objektet som attributter.

Til sidst videresender doPost-metoden requesten til siden itemlist.jsp ved hjælp af RequestDispatcher-klassen. Itemlist.jsp-siden kan derefter bruge de attributter, der er angivet i request-objektet, til at vise en liste over materialer.

14. Status på implementering

Vi nåede alle de 25 user stories som vi har set os for at lave men vi kan også se at der er plads for forbedringer. Vi er tilfredse med resultatet efter at have arbejdet i et par uger på dette projekt da den nu står klar til brug og alle de essentielle ting er opnået. Noget som vi helt sikkert kan starte med at forbedre, er betaling funktion. Lige nu er det kun en knap med samme tekst som kommer altid frem når admin har godkendt ordren. For eksempel, man kan lave sådan at status bliver opdateret fra ”Godkendt” til ”Betalt”, sådan at kunden ikke skal hver gang se på knap med tekst ”Betal og gennemfør ordre”, når de skal til at se deres fulde ordre med tegning og stykliste. I disse tilfælde, efter at kunden har trykket første gang på ”Betal og gennemfør ordre”, så kan det fra næste gang af stå ”Se gennemførte ordre” når kunden skal kigge tilbage på den betalte ordre.
Vi har også brugt tid på at designe vores hjemmeside på en ren og elegant måde som ser indbydende ud i stedet for at efterlade den kedelig og uden farver. Vi fik for eksempel også lavet karrusel billeder på forsiden.
Hvis vi fik mere tid at udvide projektet, så vil vi helt sikkert implementere flere funktioner og også implementere mulighed for at tilkøbe skur når man vælger en carport og også vælge hvilken slags tag carporten skal have (fladtag eller skråtag).

15. Proces

Arbejdsprocessen faktuelt

For effektivt at styre vores projekt brugte vi et projektstyringsværktøj kaldet Kanban via Trello. Kanban giver brugerne mulighed for at oprette tavler, hvor de kan organisere og prioritere deres

opgaver, ideer og projekter. Hver tavle består af lister, og hver liste indeholder kort, der repræsenterer opgaver eller ideer.
Vi brugte Kanban til at oprette en tavle til vores projekt og opdelte den i flere lister, herunder en "To Do"-liste for opgaver, der skulle udføres, en "Doing"-liste for opgaver, der var i gang, og en "Done"-liste for udførte opgaver. Vi lavede også en liste over ideer og problemer, der skulle løses. Ved at bruge Kanban kunne vi visuelt se udviklingen af vores projekt og nemt flytte opgaver fra en liste til en anden, efterhånden som de blev afsluttet eller flyttet til en anden fase. Det gav os også mulighed for at samarbejde med hinanden ved at tildele opgaver til hinanden og efterlade kommentarer og feedback på individuelle kort.

Samlet set viste Kanban sig at være et værdifuldt værktøj til at styre vores projekt og hjalp os med at holde os organiseret og på det rigtige spor. Det gav os mulighed for at opdele vores projekt i mindre, mere overskuelige opgaver og spore vores fremskridt på en effektiv måde.

I den første vejleder møde med Kim blev vi introduceret til sprints. Vi endte med at bruge det og sprints var fordelt i et sprint per uge og det fungerede udmærket. Vi fik nået vores deadlines på denne måde og kunne lettere strukturere og se hvor meget vi har at lave i den efterfølgende uge. Udover det, lavede vi en discord gruppe ”Fog carport”, hvor alt vores kommunikation foregik når vi ikke var fysisk sammen. Her kunne vi oprette separate kanaler for hvert emne, vi arbejdede på, og sende beskeder og filer til hinanden, når vi skulle dele information eller bede om hjælp. Det gjorde det nemt for os at samarbejde og holde styr på projektets fremdrift, selvom vi ikke var fysisk tæt på hinanden.
En anden fordel ved Discord var, at det gjorde det nemt for os at holde styr på noter og referater fra vores vejledningsmøder. Vi kunne oprette en kanal for hvert møde og derefter dele noter og referater med hinanden efter mødet. Dette hjalp os med at huske de vigtigste informationer fra møderne og holde styr på projektets fremskridt.
Endelig var Discord også en nyttig platform til at holde kontakten med vores lærer, som fungerede som repræsentant for kunden. Vi kunne sende en besked til vores lærer og få et øjeblikkeligt svar, hvilket gjorde det nemt for os at få hjælp og svar på de spørgsmål, vi måtte have.

Alt i alt har brugen af Discord været et nyttigt og værdifuldt værktøj for os under vores projekt. Det har gjort det nemt for os at kommunikere og samarbejde med hinanden, holde styr på noter og

referater og holde kontakten med vores lærer. Det har været med til at gøre projektet så vellykket som muligt.

Til sidst, for at holde et slags referat af alle møderne med vores vejleder og opsummering på ugen, har vi lavet logbog i google docs. Den kan man kigge igennem under afsnit ”Links” under navnet ”Logbog”. Vi fik også gemt under links afsnit alle vores Kanban boards.

User stories samt acceptancekriterier som vi arbejdede på og blev færdige med for hver uge:

Sprint 1:

Der blev aftalt at vi ikke vil røre med kode delen i denne første uge fordi det er super vigtigt at gennemtænke opgaven og det var meget mere relevant at løse SYS del først. Derfor gik vores fokus på at få styr på SYS, som vil så hjælpe os når vi skal begynde på at kode i næste uge.

Sprint 2:

Lave en bruger

US-1 Som en kunde, vil jeg oprette en bruger profil, så jeg kan bestille skræddersyet carport
AC-1 Givet at en bruger skal have en profil for at kunne bestille, når kunden prøver at oprette en profil, så bliver brugeren henvist til en opret bruger side

Logge ind

US-2 Som en kunde, vil jeg gerne kunne logge ind, så jeg kan bestille, se og acceptere min ordre
AC-2 Givet at en kunde er registreret, når man logger ind, så skal man videresendes til shopping siden

Bestille en carport

US-3 Som en kunde, vil jeg anmode om skræddersyet carport, så at jeg kan modtage et tilbud fra Fog
AC-3 Givet at kunden har valgt carport dimensioner, når man sender en forespørgsel, så vil den forespørgsel lande hos Fog

Se ordren

US-4 Som en kunde, vil jeg gerne se min ordre, så at jeg kan følge op på status
AC-4 Givet at kunden har lagt en ordre, når man gerne vil se detaljer, så bliver man henvist til “Se bestilling” siden

Se om os siden

US-8 Som kunde, ønsker jeg at læse om Fog, så at jeg kan få en bedre billede af virksomheden
AC-8 Givet at man skal have mere informationer om Fog, når kunden prøver at undersøge nærmere, så har man adgang til “Om os" siden

Komme i kontakt med Fog

US-9 Som en kunde, vil jeg komme i kontakt med Fog kundeservice, så jeg kan stille spørgsmål og få hjælp til at lægge en ordre
AC-9 Givet at kunden har spørgsmål/brug for hjælp, når kunden prøver at komme i kontakt, så kan man kontakte kundeservice via telefon eller e-mail

Se sin profil

US-10 Som en kunde, vil jeg gerne se min profil, så jeg kan tjekke eller rette i mine oplysninger
AC-10 Givet at kunden har en profil og er logget ind, når kunden ønsker at se informationer om sig selv, så bliver man henvist til profil siden

Skifte kode

US-11 Som en kunde, vil jeg gerne se skifte min kode, så jeg kan tilknytte en ny kode til min profil
AC-11 Givet at kunden har en profil og er logget ind, når kunden ønsker at skifte koden, så bliver man henvist til profil siden

Opdatere sine oplysninger

US-12 Som en kunde, vil jeg gerne opdatere mine oplysninger, så jeg kan tilknytte korrekte informationer til min profil
AC-12 Givet at kunden har en profil og er logget ind, når kunden ønsker at rette i sine oplysninger, så bliver man henvist til profil siden

Logge ind

US-13 Som en medarbejder, vil jeg gerne logge ind, så jeg kan se alle forespørgsler og ordrer

AC-13 Givet at en medarbejder har en admin bruger, når man logger ind, så vil man lande på oversigt siden

Lave ny admin

US-14 Som ny medarbejder, ønsker jeg at få en admin bruger i systemet, så jeg kan lave mit arbejde
AC-14 Givet at en eksisterende medarbejder opretter en ny admin bruger, når man skal lave oprette en ny admin bruger, så bliver den eksisterende medarbejder henvist til "Opret ny admin" siden

Se alle ordrer

US-15 Som en medarbejder, vil jeg gerne se alle ordrer ligegyldig ordrestatus, så jeg har overblik og kontrol over ordrer
AC-15 Givet at en medarbejder er logget ind som admin, når man ønsker at se alle ordre, bliver man henvist til ordreoversigt siden

Se materialeliste

US-16 Som en medarbejder, vil jeg gerne se materialeliste, så jeg har overblik over materialer
AC-16 Givet at materialelisten eksisterer, når en medarbejder vil se materialeliste, så gør man det via "Materialer" siden

Opdater, slet og tilføj i materialeliste

US-17 Som en medarbejder, vil jeg gerne opdatere, slette og/eller tilføje materialer til materialeliste, så jeg altid har en opdateret liste
AC-17 Givet at materialelisten eksisterer, når en medarbejder vil opdatere, slette og/eller tilføje materialer til materialeliste, så gør man det via "Materialer" siden

Se alle kunder

US-18 Som en medarbejder, vil jeg gerne se alle informationer på Fog kunder, så jeg har overblik og mulighed for kontakt til kunder
AC-18 Givet at kundeliste eksisterer, når man ønsker at se kundeliste eller at komme i kontakt med kunder, bliver man henvist til "Kunder" siden

Slette en kunde

US-19 Som en medarbejder, vil jeg gerne slette en kunde fra systemet, så jeg altid har en opdateret kundeliste
AC-19 Givet at kundeliste eksisterer, når en medarbejder ønsker at slette en kunde, så gør man det via "Kunder" siden

Se ordren
US-20 Som en medarbejder, vil jeg gerne se kundens ordre, så at jeg kan se indholdet i kundens bestilling
AC-20 Givet at der er lagt en ordre, når medarbejderen gerne vil se detaljer, så bliver man henvist til "Se bestilling" siden

Godkend ordren
US-22 Som en medarbejder, vil jeg gerne godkende en ordre, sådan at kunden kan betale og gennemføre sin ordre
AC-22 Givet at der er lagt en ordre, når medarbejderen ønsker at godkende en ordre, så gør man det via "Bekræft" knappen i "Alle ordrer" siden

Slet ordren
US-23 Som en medarbejder, vil jeg gerne slette en ordre, sådan at jeg har en ryddelig ordreliste
AC-23 Givet at der er lagt en ordre, når medarbejderen ønsker at slette en ordre, så gør man det via "Fjern" knappen i "Alle ordrer" siden

Sprint 3:

Betale ordren
US-5 Som en kunde, vil jeg acceptere og betale en bestilling, så jeg kan modtage min stykliste
AC-5 Givet at en bestilling er aktiv, når kunden har betalt og gennemført en bestilling, så får kunden adgang til styklisten

Se styklisten
US-7 Som en kunde, vil jeg se stykliste til den valgte ordre, så at jeg kan se hvad der skal bruges til at bygge min carport
AC-7 Givet at kunden har betalt for ordren, når kunden ønsker at se styklisten, så gå til "Se bestilling" siden og få styklisten fremvist

Prisberegning af carport

US-24 Som en medarbejder, vil jeg gerne beregne prisen på en ordre, så at jeg kan oplyse en pris til kunden

AC-24 Givet at der er lagt en ordre, når medarbejderen ønsker at oplyse en pris til kunden, så beregn carports pris ud fra de valgte dimensioner

Generer stykliste til carport

US-25 Som en medarbejder, vil jeg gerne generere en stykliste, så at kunden kan modtage den

AC-25 Givet at kunden har gennemført en ordre, når medarbejderen ønsker at generere en stykliste, så beregn materialer som skal bruges til carporten efter de valgte dimensioner

Sprint 4:

Se tegning

- US-6 Som en kunde, vil jeg se tegning af min bestilte carport, sådan at jeg kan hvordan den ser ud samt bygge ud fra denne
- AC-6 Givet at kunden har lagt en ordre, når kunden ønsker at se tegning, så gå til "Se bestilling" siden og få tegning fremvist

Se tegning

- US-21 Som en medarbejder, vil jeg se tegning af en kundens bestilte carport, sådan at sikre mig at tegning er korrekt
- AC-21 Givet at der er lagt en ordre, når medarbejderen ønsker at se tegning, så gå til "Se tegning" siden og få tegning fremvist

Arbejdsprocessen reflekteret

Vi endte med at ikke have en Kanban master, da vi er kun to i gruppen og kunne nemt styre vores projekt pga. godt samarbejde og gennemtænkt struktur.

Vi fik en god rytme fra starten af fordi vi har lyttet til vores lærere og også lærte af den erfaring fra forrige projekt, "Olsker Cupcakes". Vi startede som nævnt tidligere, at kun fokusere på SYS delen først, som gjorde hele kodnings oplevelsen mere glat og struktureret. Vi vidste præcis hvad vi skulle gøre og kommunikation var i top hele vejen. Det som vi også særligt bemærkede ved dette projekt,

er hvor stor del SYS faktisk har. Det at vi fik styr på den del først, gjorde bare hele oplevelsen mere positivt og succesfuld når vi skulle til at kode. Det viser bare hvor vigtigt det er at fokusere lige så meget på begge to områder og ikke kun på kodnings delen.

Gennem hele projektet understregede vi vigtigheden af effektiv kommunikation og samarbejde, som var en essentiel del til vores succes. Vi holdt regelmæssige vejledningsmøder med vores lærer og tog grundige noter og referater for at spore vores fremskridt og løse eventuelle problemer, der opstod.
Det endelige produkt af vores projekt er en fungerende hjemmeside, der opfylder kundens krav og en omfattende rapport, der dokumenterer processen. Vi er stolte af det hårde arbejde og dedikation, vi har lagt i dette projekt, og de færdigheder og viden, vi har opnået som resultat.

Der er flere grunde til, at vi valgte at bruge separate GitHub-repositories til vores projekt:
Samarbejde: Ved at bruge separate repositories kunne vi hver især arbejde på vores egen kode uafhængigt uden at påvirke den anden persons arbejde. Dette gav os mulighed for at samarbejde mere effektivt, da vi hver især kunne pushe vores ændringer til vores egne repositories og derefter merge dem sammen på et senere tidspunkt.
Versionskontrol: Ved at bruge separate repositories kunne vi hver især vedligeholde vores egen versionshistorik, hvilket var nyttigt til at spore ændringer og fejlfinde problemer.
Kodegennemgang: Ved at bruge separate repositories kunne vi begge nemt gennemgå og godkende kodeændringerne for den anden person, da vi kan se alle ændringerne foretaget i et enkelt repository.
Organisation: Ved at bruge separate repositories kunne vi holde koden organiseret og adskilt, hvilket var særligt nyttigt, fordi projektet er stort og komplekst.
Sammenfattende gav brug af separate repositories os mulighed for at samarbejde mere effektivt, opretholde bedre versionskontrol, gennemgå kodeændringer lettere og holde koden organiseret. Derfor valgte vi at bruge separate repositories til vores projekt.
Selvom separate respositories var en nyttig måde at samarbejde om et projekt på, ved vi, at det også er muligt at bruge et enkelt respository og bruge branches til at opnå lignende - hvis ikke bedre - resultater.
Ved at bruge branches inden for et enkelt repository kunne vi have arbejdet på hver vores egen kode uafhængigt uden at påvirke den anden persons arbejde. Vi kunne så have merget vores ændringer sammen til en enkelt branch, når vi var klar til at kombinere vores arbejde. Dette ville have været en

mere effektiv metode til samarbejde, da vi ikke konstant skulle pushe og pulle ændringer mellem separate repositories.
Brug af branches ville også have givet os mulighed for at opretholde en enkelt versionshistorik og nemt gennemgå og godkende kodeændringer foretaget af den anden person. Det ville også have holdt koden organiseret og på et enkelt sted, hvilket ville have været nemmere at administrere og holde styr på.
Set i bakspejlet burde vi have brugt branches inden for et enkelt repository i stedet for separate repositories til vores projekt. Det ville have gjort samarbejde og kodegennemgang lettere og ville have holdt al koden på et enkelt, organiseret sted.