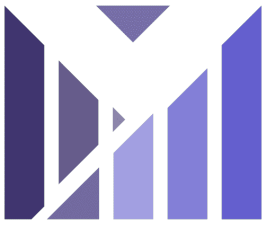


CoderGram

FOG

Datamatiker 2. semester
5. Januar 2021

Gruppe: CoderGram
Sigurd Arik Gaarde Nielsen – cph-at89@cphbusiness.dk – GitHub: ariktwena
Jacob Lange Nielsen – cph-jn352@cphbusiness.dk – GitHub: langeeee
Emil Elkjær Nielsen – cph-en93@cphbusiness.dk – GitHub: eelkjaer

Præsentation af produktet: https://youtu.be/BPg9Hajrepo
Hjemmeside: https://codergram.me/fog/
GitHub: https://github.com/eelkjaer/Fog
JavaDoc: https://eelkjaer.github.io/Fog/
Taiga: https://tree.taiga.io/project/eenielsen-fog/backlog

Test users (Case-sensitive)
Administrator
E-mail: admin@admin.dk
Kodeord: admin

Sælger
E-mail: sales@sales.dk
Kodeord: sales

Indholdsfortegnelse

Introduktion
Som en del af datamatiker studiets 2. semester på Cphbusiness, har vi fået til opgave at udvikle en hjemmeside for Johannes Fog A/S, som sælger carporte med brugerdefinerede mål.
Vi har som gruppe udviklet projektet samme – herunder kode og rapport - og vi tager alle ligeligt ansvar for udførelsen af dette.

Formål
Formålet med rapporten er at dokumentere og beskrive udviklingen af en hjemmeside for Johannes Fog A/S.

Mål
Målet med rapporten er:
- At give studerende på samme faglige niveau en forståelse for projektet.
- At beskrive hjemmesidens formål og krav.
- At dokumentere udviklingen og kravspecifikationerne på en sådan måde, at projektet kan videreføres til tredje mand på samme faglige niveau.
- At udvikle en rapport, som en potentiel kunde (Johannes Fog A/S) vil kunne forstå, og som lever op til de faglige krav der forventes af studerende på datamatikerstudiets 2. semester på Cphbusiness.

Baggrund
Virksomheden Johannes Fog A/S blev grundlagt i 1920 ved etableringen af Johannes Fogs Tømmerhandel på Rolighedsvej i Lyngby. Frem til grundlæggerens død i 1970 var det en enkeltmandsejet virksomhed.

Siden 1920 blev Johannes Fog A/S udvidet med to afdelinger. Den ene på Rolighedsvej i Lyngby, mens den anden befandt sig på Firskovvej i Nærum, som i løbet af 70'erne blev udvidet til to pladser. I 1988 blev de to tømmerpladser på Firskovvej samlet med en "privat" jernbaneoverskæring inde på virksomhedens grund og fylder nu i alt 31.000 m^2, hvoraf 9.200 m^2 er under tag.

I løbet af 1990´erne blev Ejby Trælast og Farum Tømmerhandel købt. I 1999 fusionerede Johannes Fog med Hørsholm Trælast og Kvistgård Trælast. I 2004 blev Anco i Herlev købt. Derefter købte man Asminderød Savværk & Trælast i 2005. Den næste i rækken var butikken i Vordingborg, der blev købt i marts 2008. I december 2008 købte man så Værebro Tømmerhandel, og senest i marts 2013 blev S.A Pedersen i Helsinge købt.

I dag fylder Johannes Fog A/S over 40.000 m^2, og er 100% ejet af Tømmerhandler Johannes Fogs Fond, som går meget op i CSR mht. miljø og samfundsudvikling. Visionen for Johannes Fog A/S er at give kunderne en kvalificeret rådgivning til den bedste løsning. Deres mål er at hjælper deres kunder med at få overblik og komme i mål med deres byggeprojekt. Denne rådgivning handler både om vejledning og viden indenfor byggeprojekter og valg af materialer. Fokusset hos Johannes Fog A/S, ligger i afdækning af kundens behov og servicering af deres behov.

Teknologier

Vi har til projektet benyttet os af følgende teknologier:

Værktøjer:
- IntelliJ Ultimate (Version 2020.3)
- Navicat Premium (Version 15.0.22)
- Navicat Data Modeller (Version 3.0.8)
- Bootstrap Studio (Version 5.4.2)
- MySQL Workbench (Version 8.0.18)
- Selenium IDE (3.17.0)

Teknologier:
- Ubuntu (Version 20.04.1 LTS)
- Tomcat (Version 9.0.22)
- Nginx (Version 1.18.0)
- Java 14 (Version 14.0.1)
- MySQL (Version 8.0.22)

Stack:
- HTML5
- CSS
- Bootstrap (Version 4.5.3)
- JavaScript
- JQuery
- DataTables[1]

3rd Party:
- Let's Encrypt (https://letsencrypt.org)
- Mailgun (https://www.mailgun.com)

Problem/Analyse

Johannes Fog A/S (Fog) oplever salgsmæssige udfordringer ved salg af deres carporte med brugerdefinerede mål.
Udfordringerne består i, at sælgerne manuelt skal justere i bestillingernes udregninger, før de kan generere et tilbud til kunderne.
Materialerne og listepriserne er fastlåste i deres nuværende database, og Fog mangler den fleksibilitet, som kan frigøre mere tid til sælgerne og deres salgsarbejde.

Krav

For at kunne levere et personligt og skræddersyet tilbud til kunder der køber carporte med brugerdefinerede mål, ønsker Johannes Fog A/S (Fog) at få udviklet et IT-system, som kan hjælpe sælgerne med:

- At øge kundekontakten, så kunderne får et mere personligt salg med vejledning.
- At give sælgerne et bedre overblik over kundernes behov og ønsker.
- At give sælgerne et bedre overblik over materielet som kunden skal bestille, så sælgeren har bedre mulighed for at give et personligt tilbud ud fra kundes behov og ønsker.
- At give ledelsen større fleksibilitet over materielet og listepriserne.

User Stories

Ud fra Johannes Fog A/S (Fog) krav, har vi i samarbejde med product owner og udarbejdet følgende User Stories som vi vil gennemføre for Fog, og som skal hjælpe Fog med at øge deres værdi. Ud fra hver User Story har vi udregnet velocity, basseret på produktet af den estimerede størrelse gange med prioriteten af opgaven. Efterfølgende har vi delt User Stories ud på de forskellige sprints, så vi kan nå i mål med projektet.

Nr.	User Story	E	P	V	S
1	Som sælger skal man kunne logge ind, så sælger kan få adgang til kundernes forespørgsler og kontaktoplysninger.	3	5	15	1
2	Som sælger skal man kunne se alle ordre, så sælger nemt kan få et overblik over, hvad status er på alle ordrerne.	5	5	25	1
3	Som sælger skal det være tydeligt når en ny forespørgsel er kommet, så sælgerne ikke overser nye forespørgsler.	3	3	9	1
4	Som sælger skal man kunne ændre dækningsgraden på et tilbud, så sælger måske alligevel kan få et salg igennem, når en kunde er tøvende.	3	5	15	2
5	Som sælger skal man kunne se styklisten for en ordre, så sælger kan dobbelttjekke om der er nok materialer til at lave en stabil konstruktion.	5	5	25	2
6	Som sælger skal man kunne afslutte en ordre når fakturaen er betalt, så sælger nemt kan holde overblik over åbne sager.	1	1	1	1
7	Som sælger skal det være muligt at redigere i salgsprisen, så sælgeren bedre kan matche kundens købspris og derved gennemføre et salg	5	3	15	3
8	Som sælger skal man kunne slette en forespørgsel, så sælgerne kan holde styr på eksisterende forespørgsler.	1	1	1	1
9	Som admin skal man kunne slette en sælger bruger, så programmet ikke bliver brugt af gamle sælgere som ikke arbejder der mere.	1	1	1	2
10	Som admin skal man kunne rette i eksisterende materialer, så materialelisten er ”up to date” med det eksisterende lager.	3	3	9	3
11	Som admin skal man kunne oprette en sælger i systemet, så ledelsen kan se forskel på hvilke sælgere der er tilknyttet til hvilke salg	1	5	5	1
12	Som kunde får man tilsendt en bekræftelse e-mail med tegningen på carporten, så kunden har en ide om hvilken konstruktion kunden skal til at bygge.	5	3	15	1
13	Som kunde skal man få tilsendt styklisten så snart tilbuddet er betalt, så kunden kan gå ud i tømmerhandlen og få udleveret materialer.	5	5	25	2
14	Som kunde skal man kunne vælge et specifikt mål på en carport, så kunden kan blive kontaktet af en sælger mht. op følgende salg.	5	5	25	1
15	Som kunde skal man kunne se en tegning af carporten inden forespørgslen bliver sendt, så kunden kan sikre sig at carporten opfylder kundens behov.	5	3	15	1
16	Som admin skal man kunne se hvilken sælger der er tilknyttet til en pågældende ordre, så admin får et bedre overblik over salgsarbejdet og sælgernes salgsopgaver.	5	3	15	2
17	Som admin skal man kunne ændre i den sælger der er tilknyttet en ordre, så andre sælgere har mulighed for at gennemføre et salg, hvis den tilknyttede sælger, ikke er til stede.	5	3	15	2

18	Som sælger og admin skal man kunne se at man er logget ind på sin personlige profil, så sælger eller admin ikke kommer til at rette i andres profiler, eller bryder sikkerheden, vis kollegaer glemmer at logge ud.	5	3	15	2
19	Som sælger og admin skal man nemt kunne se datoen for ordres oprettelse, så medarbejderne og ledelsen nemt kan danne sig et overblik over ordrenes ventetid.	5	3	15	2
20	Som kunde skal man kunne se sin ordre på hjemmesiden når den er betalt, så kunden nemt kan få et overblik over ordren, styklisten og bestillingsinfoen.	5	3	15	2

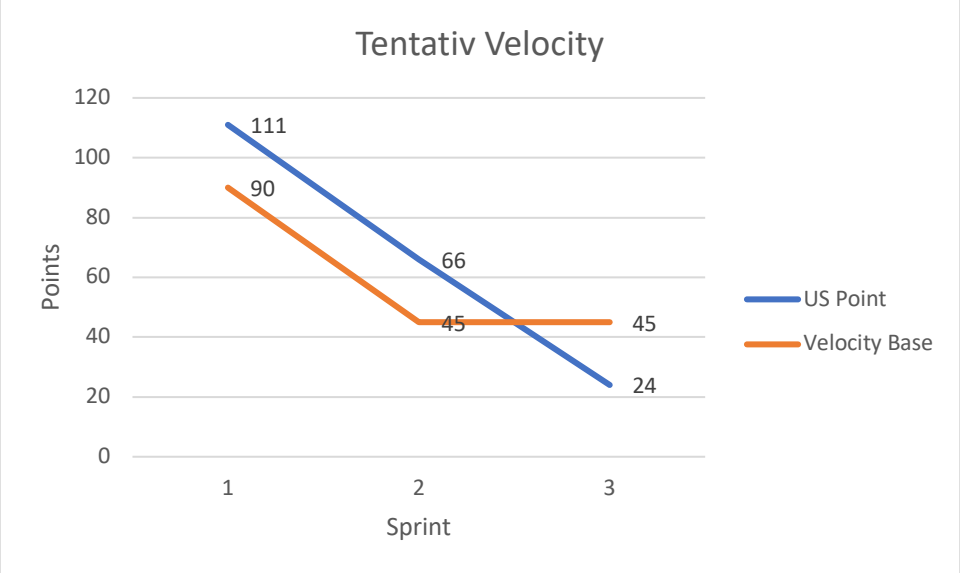
E: Estimeret størrelse (1, 3 eller 5) P: Prioritet (1, 3 eller 5) V: Velocity (E * P) S: Sprint nr.

Tentativ Velocity

For at kunne gennemføre projektet, har vi valgt at fastsætte en tentativ velocity værdi, som vi vil stræbe efter at opnå i hvert sprint.
Vores velocity i hvert sprint er udregnet på baggrund af prioriteringen og tidsfaktoren, som hver User Storie tager (US Point).

Vores base velocity er middelværdien af alle vores sprints.
Efter hvert sprint, vil vi evaluere vores opnåede velocity og justerer projektets efterfølgende sprints og User Stories på baggrund af denne evaluering.
I slutningen af rapporten vil vi evaluere vores samlede velocity med henblik på at måle vores reele effektivitet.

Vi har valgt at sprint 1 skal have en varighed på 2 uger, mens sprint 2 og 3 er på en uges varighed. Dette har vi valgt, da IT-systemets opsætning kræver tid, som vil hindre udviklingen af projektet. Den anden grund er, at statistisk set så er teams mest produktive mht. ”Completion” under 2 ugers sprints, når man arbejder med Scrum[1].



[1] https://advanz.dk/blog/scrum-sprint/

Afgrænsning

For at kunne give Johannes Fog A/S (Fog) størst mulig værdi i den tid vi har til rådighed til projekter, har vi valgt at afgrænse projektet.
Dette har vi valgt for at kunne levere et delvist gennemtestet og fuldt fungerende produkt. Derved prioriterer vi kvalitet frem for kvantitet.

Vi har valgt at afgrænse projektet med følgende punkter:

- Vi har valgt at afgrænse målene på carporten, så Fog kan starte med at sælge carporte hurtigst muligt.
  Længden er afgrænset fra 5,80m til 7,80m. Og bredden fra 3,00m til 6,00m.

- Vi har valgt at beholde en standardhøjde på carporte med hældning, som overholder minimumshældningen på 15 grader.

- Vi har valgt kun at have 2 typer skure til carporten.
  Et skur på fuld eller halv bredde, mens længden er fastlåst til carporttypen (længde 2,10m/2,25m).

- Vi har valgt ikke at fokuserer på leveringen af carporten, da kunderne fx kan leje en trailer.

- Vi har valgt at genererer en estimeret stykliste, som ikke er 100% præcis.
  Dette har vi valgt, da vi ikke kender den præcise algoritme for den bærende konstruktion og materialeforbrug.

- Vi har valgt ikke at lade sælgerne rette i de genererede styklister, da vi ser mere værdi i at styklisterne bliver præcise, end at sælgerne ændrer i data og evt. laver fejl.

- Vi har valgt ikke at oprette kunder med tilhørende login i vores system.
  Dette har vi valgt, da vi bruger UUID til at generere et unik link til kunder der har bestilt en carport, så de kan se deres ordre.
  På den måde gør vi hjemmesiden mere sikker, da det kun er medarbejderne der har adgang til back-end delen af hjemmesiden og at der er en relativt lille sandsynlighed for at kunden ønsker at købe endnu en carport.

Ordliste

I løbet af rapporten vil vi benytte os af følgende termer og forklaringer.

Term	Forklaring
Fog	Johannes Fog A/S.
Sælger/salgsmedarbejder	Den medarbejder der har den primære kundekontakt og som har ansvaret for tilbudsgivningen.
Lagermedarbejder	Den medarbejder som hjælper kunderne med at plukke varerne fra Johannes Fog A/S lokale eller fjernlager.
Afdelingsleder/Daglige leder	Den daglige leder, som har ansat en stab af salgsmedarbejdere og har det ledelsesmæssige ansvar over salgsmedarbejderne.
Distriktschef	Har det regionale ansvar og ledelsesansvaret over afdelingslederne.
HR	Human Ressource afdelingen som tager sig af personalesager
Controller/Bogholderi	Regnskabsafdelingen som løbende kontrollerer Johannes Fog A/S finanser.
Revision/Revisor	Det firma der står for at kontrollere, kommentere og underskrive Johannes Fog A/S regnskaber.
Direktør/Øverste ledelse	Den øverste ledelse i Johannes Fog A/S, som har det overordnet ansvar for driften af virksomheden, samt er beslutningstageren.
Bestyrelsesmedlemmer	De medlemmer som bestyrelsen består af, og som Johannes Fog A/S direktør refererer til.
Fond	Den fond som ejer Johannes Fog A/S.
Samarbejdspartnere	De samarbejdspartnere som Johannes Fog A/S arbejder sammen med, og som deler fælles interessenter.
Leverandører	De leverandører som Johannes Fog A/S er afhængig af, for at drive virksomhed.
Kunde	Den kunde som ønsker at købe eller få et tilbud på en carport med brugerdefinerede mål.
Kommune	Den pågældende Kommune, som den lokale Johannes Fog A/S afdeling har adresse under.
Regering	Den sidende regering.
Admin/Administrator	Den person/leder som har fuldt adgang til IT-systemet

Interesseanalyse

I det følende afsnit vil vi lave en interesseanalyse over de interessenter som Fog har, samt hvilken rolle og indflydelse de har.
Dette er vigtigt, da der kan opstå interessekonflikter i løbet af projektet, når fx ny viden skal deles eller læres, samt når beslutningerne skal træffes eller implementeres.
Ved først at analysere Fog i forhold til Fog's omverden, og derefter analysere Fog i forhold til os, får vi en bedre forståelse for de problematikker og konflikter der kan opstå under projektet ift. interessenterne.
Men vi får også et overblik over de strategier og muligheder der ligger i vores interaktion med interessenterne.

For at gennemføre denne analyse, vil vi først starte med at se på de interessenter som Fog interagerer med. Når vi har fundet frem til dem, så vil vi i det næste skema præciserer de fordele og ulemper som vores projekt kan medføre den enkelte interessent.
Ud fra disse fordele og ulemper, vil vi forsøge at analysere os frem til hvilken rolle interessenten spiller og hvordan vi kan være proaktive mht. at "møde" og håndtere denne interessent.
Det er vigtigt at vi har en strategi på plads, som vi kan læne os op ad, da det er for sent at lave denne strategi, hvis konflikter eller justeringer først opstår.

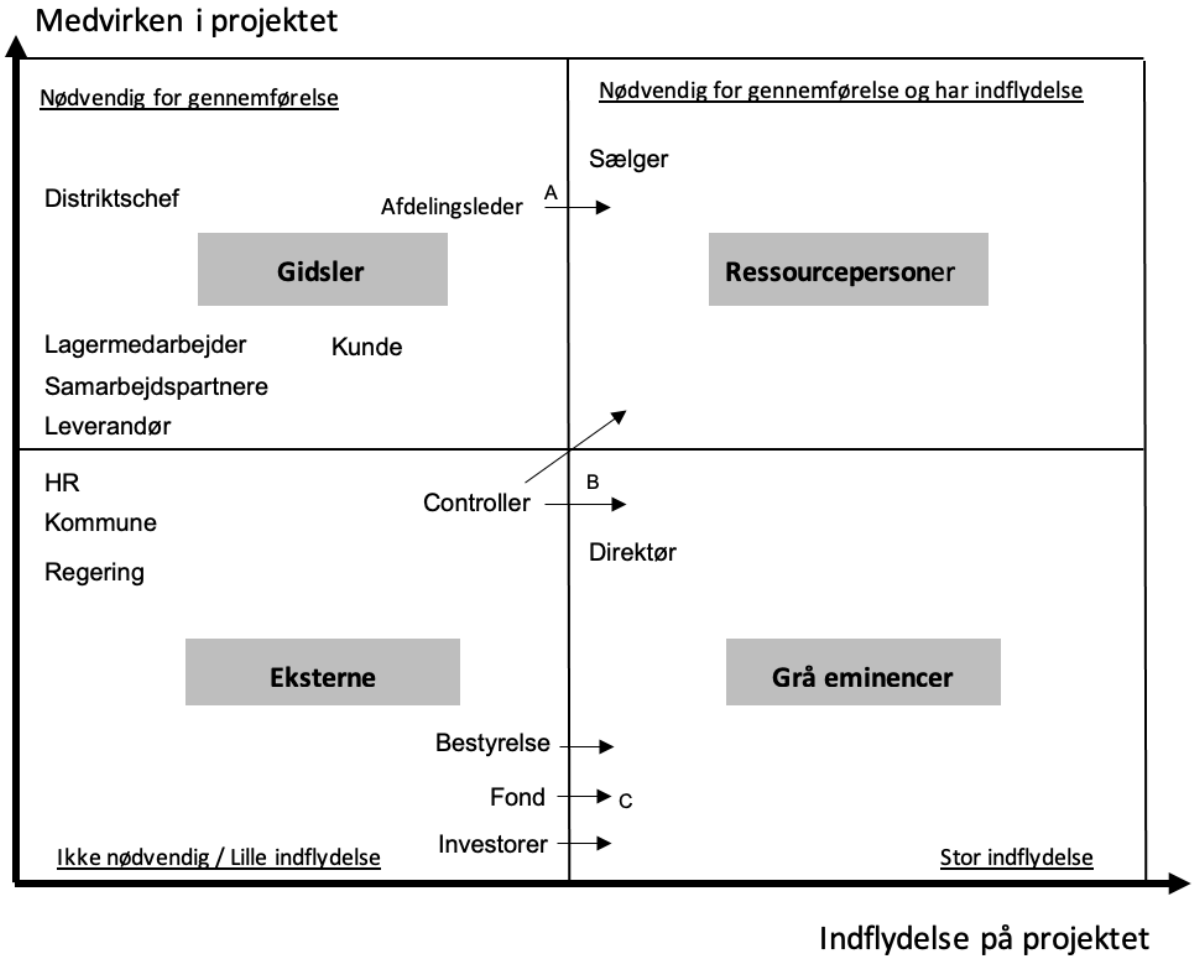
Når vi har vores strategi på plads for de enkelte interessenter, så visualiserer vi overfor os selv, hvilken indflydelse de forskellige interessenter har på projektet.
Dette er endnu et værktøj til at forstå, hvor indflydelsen mht. til viden ligger, samt hvem der ejer projektet mht. Kapitalen.

Vi skal finde den rette balance mellem at gøre kunden glad (Fog's topledelse), mens vi samtidige motiverer og får adgang til den tavse viden som medarbejderne har.
Ved at gøre dette, så anerkender vi at vores IT-system, nok kan være med til at ændre kulturen i afdeling via nye processer, og at vi derfor skal assistere ledelsen med deres forandringsledelse[2].
Det er især gidsler og ressourcepersoner som vil være blandt dem, som kommer til at mærke til de forandringer som projektet vil medføre.

[2] https://net2change.dk/hvad-er-forandringsledelse/

Projekt: FOG	Udfyldt af: Arik	Dato: 19. november 2020

Interessent	Fordel	Ulempe	Samlet vurdering af interessentens bidrag/position	Håndtering af interessenten
Sælger	Bedre materiale overblik Bedre behovsafdækning af kunden Øget salg og gennemsigtighed	Nye processer og krav Øget arbejdspres	Nøgleperson mht. nuværende og fremtidigt krav, viden og processer	Tæt kommunikation
Lagermedarbejder	Bedre materiale overblik Bedre bestillingsoversigt	Nye processer og krav Øget arbejdspres	Har en vigtig position mht. levering af salget	Skal inkl. i processen når styklisterne skal udarbejdes
Afdelingsleder	Øget materiale- og salgsforståelse Evt. frigørelse af tid/ressourcer Bedre overblik	Nye processer og krav Øget administration	Kan bidrage med de krav som systemet skal have mht. gennemsigtighed	Møder mht. back-end og lokal materiale- og regnskabsføring
Distriktschef	Bedre materiale- og salgsstrategi Evt. frigørelse af tid/ressourcer Bedre overblik	Nye processer og krav Øget administration	Kan bidrage med de krav som systemet skal have mht. gennemsigtighed	Møder mht. back-end og regional materiale- og regnskabsføring
HR	Bedre medarbejderforståelse	Øget personalesager grundet nye krav	Kan pt. ikke bidrage med info	Kan evt. være med til ledelsesmøderne
Controller	Bedre salgs- og økonomiforståelse Evt. frigørelse af tid/ressourcer	Nye processer	Nøgleperson mht. systemets regnskabsudregning	Løbende møder mht. bogføringskrav
Revisor	Bedre salgs- og økonomiforståelse	Evt. nye processer	Kontakt til revisor sker gennem controller.	Revisorgodkendelse sker gennem controller
Direktør	Videreudvikling af virksomheden Evt. frigørelse af tid/ressourcer Bedre overblik	Nye krav fra bestyrelsen og aktionærerne	Nøgleperson mht. finansiering af projektet	Løbende møder mht. processen, deadlines og projektets økonomi
Bestyrelsen	Videreudvikling af virksomheden Evt. frigørelse af tid/ressourcer Bedre overblik	Nye krav fra aktionærerne	Vigtigt personer mht. godkendelse af fremtidige implementeringer	Vores information skal være gennemsigtig for at få accept for bestyrelsen
Fond	Øget indtægt	Investeringen i systemet kan give et dårligt årsresultat	Bliver vigtige hvis projektets tidshorisont ikke bliver overholdt	Kommunikationen går gennem den øverste ledelse og bestyrelsen
Samarbejdspartnere	Øget salg	Evt. nye krav til samarbejde	Vigtig i forhold til lager/service	Orienteringsmøde
Leverandører	Øget salg	Nye krav til levering	Vigtig i forhold til lager	Evt. orienteringsmøde
Kunde	Øget kundetilfredshed Bedre behovsafdækning	Købsprocessen kan tage længere tid	Vigtig mht. købs flow og design	Indsamling af kundedata vha. sælgerne
Kommune	Øget indtægt via skatter Flere arbejdspladser i kommunen	Øget administration	Vi skal overholde gældende love	Undersøge gældende love
Regering	Stigning i moms. Øget indtægt via skatter	Øget administration	Vi skal overholde gældende love	Undersøge gældende love



A

Afdelingslederen vil være en god kandidat til en fremtidig nøgleperson. Dette skyldes, at afdelingslederen skal kende til sit kundesegment, samt kende til de arbejdsprocesser som hans sælgere følger. En anden vigtig ting er, at salgsstrategien bliver givet fra ledelsen ned til afdelingslederen. Derfor kender afdelingslederen de bagvedliggende mål og krav som vores IT-system skal følge. Ved at få vores information fra afdelingslederen, undgår vi medarbejder bias.

B

Controlleren kan alt efter situation bytte plads. Hvis projektet fx går over budget, så vil en controller få stor indflydelse på projektet, da den øverste ledelse bliver adviseret af regnskabsafdelingen. På den anden side, så kan en controller blive en ressourceperson, da IT-systemets regnskabsmæssige udregninger skal overholde Fog's controller krav.

C

Bestyrelsen, revisor og fond kan få betydelig indflydelse på projektet. Dette kan bl.a. ske, hvis den øverste ledelse ikke har været gennemsigtig med informationen, eller hvis vores information ikke er god nok. Budgetmæssige krav kan også spille ind, da bl.a. bestyrelse har et ansvar overfor fonden. Kvartals- og årsregnskaber kan også spille en rolle, da projektet er afhængigt af det samlede årsregnskab.

Delkonklusion

Ud fra vores interessentanalyse kan vi konkludere, at vi skal have et tæt samarbejde med sælgerne. Dog skal vi prøve at konvertere dette samarbejde, så det også inkl. afdelingslederne og controllerne. Dette vil bl.a. give os en bedre økonomisk forståelse samt et bedre udgangspunkt mht. standardiserede processer.

Den information som vi giver ledelsen skal være så fyldestgørende og gennemsigtig, at det giver den øverste ledelse de nødvendige informationer. Vores tidshorisonter og budgetter skal også være realistiske for at minimere sandsynligheden for at eksterne aktører får indflydelse på projektet.

Vi kan også se, at der er gidsler blandt vores interessenter. Det er især vigtigt for projektet succes, at vi får snakket med lagermedarbejderne om deres krav til fx styklisterne, da de er gidsler i forandringen. Kunderne er også gidsler, da de bliver præsenteret for en ny måde at bestille på. Her er det også vigtigt at vi får undersøgt, om vores fremgangsmåde er den korrekte i forhold til kundens ønsker. Men vi kan også opleve gidsler blandt samarbejdspartnere og ledelsen. Derfor er det vigtigt at få kommunikeret de ændringer og processer som IT-systemet kommer til at påføre, så vi kan være proaktive mht. specielle tiltag og ønsker.

SWOT-analyse

I det følgende afsnit vil vi lave en SWOT-analyse, for bedre at kunne identificere vores styrker og svagheder, samt hvilken muligheder og trusler vi har i det følgende samarbejde med Fog.

Stryker	Svagheder
Vi bor tæt på kunden Vi er innovative Vi arbejder gratis Vi er ivrige efter at lære nyt Vi har stor individuel erfaring	Manglede kendskab til carporte Kort tid til opgaven Manglede erfaring Ny gruppe mht. samarbejde
Muligheder	**Trusler**
Vi kan evt. konvertere hjemmesiden til andre brancher Vi kan evt. sælge løsningen til andre virksomheder der sælger carporte COVID-19 kan sætte gang i hjemmeprojekter	COVID-19 kan medføre restriktioner i samfundet Lovgivning mht. miljøkrav og spild Konkurrenter på markedet med større erfaring

Ud fra SWOT-analysen kan vi konkludere, at vores erfaringsmæssige grundlag kan have stor betydning for vores succes. Derfor er det vigtigt at vi fokuserer på de opgaver som vi kan nå

indenfor den fastsatte tidgrænse. Det er også vigtigt at vi ikke bliver for ivrige, men holder os til Fog's ønsker. Vores manglende erfaring kommer også til at spille en rolle i forhold til vores finish. Vi bliver nødt til at prioriterer ekstra tid til denne del, da vi ikke har så stor erfaring med bl.a. testning og fejlfinding. Derfor er det ekstremt vigtig at vi giver os selv den nødvendige tid til denne del af projektet.

Samfundsmæssigt står vi også overfor en pandemi, som kan have stor betydning for vores projekt. COVID-19 har medført, at vi kommer til at arbejde online og ikke kommer til at have så mange fysiske møder. Dette kan have en negativ effekt på opgaveløsningen, da man kommer til at arbejde mere autonomt i små øer. Derfor er det vigtigt med flere daglige møder, hvor vi opdaterer hinanden mht. arbejdsstatus.

Domænemodel

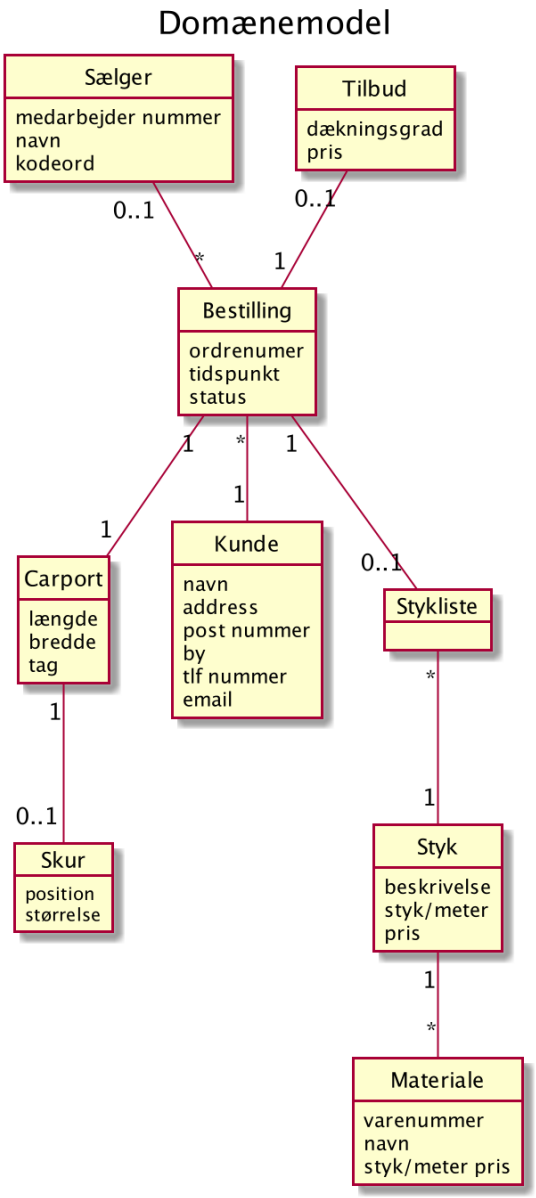
Følgende model er en konceptuel model over domænet. Her kan vi se hvilken datatyper og variabler vi forventer at bruge i de forskellige klasser, ud fra den information vi skal indhente. Modellen giver os også et overblik over, hvilket forhold klasserne har til hinanden. Det er vigtigt at modellen er forståelig for kunden, da Fog's sælgere og øverste ledelse skal kunne forstå relationerne i modellen.

I dette projekt er det primært Bestilling som er i fokus. Dette skyldes, at hele hjemmesiden beror på, at en Kunde gennemfører en bestilling. Derfor skal en Kunde kunne kobles sammen med en Bestilling. Da vi ikke ved om Fog ønsker et login system, er det vigtigt at vi identificerer den pågældende Kunde, så hver Bestilling kun har én Kunde. Men en Kunde kan godt gennemføre flere bestillinger.

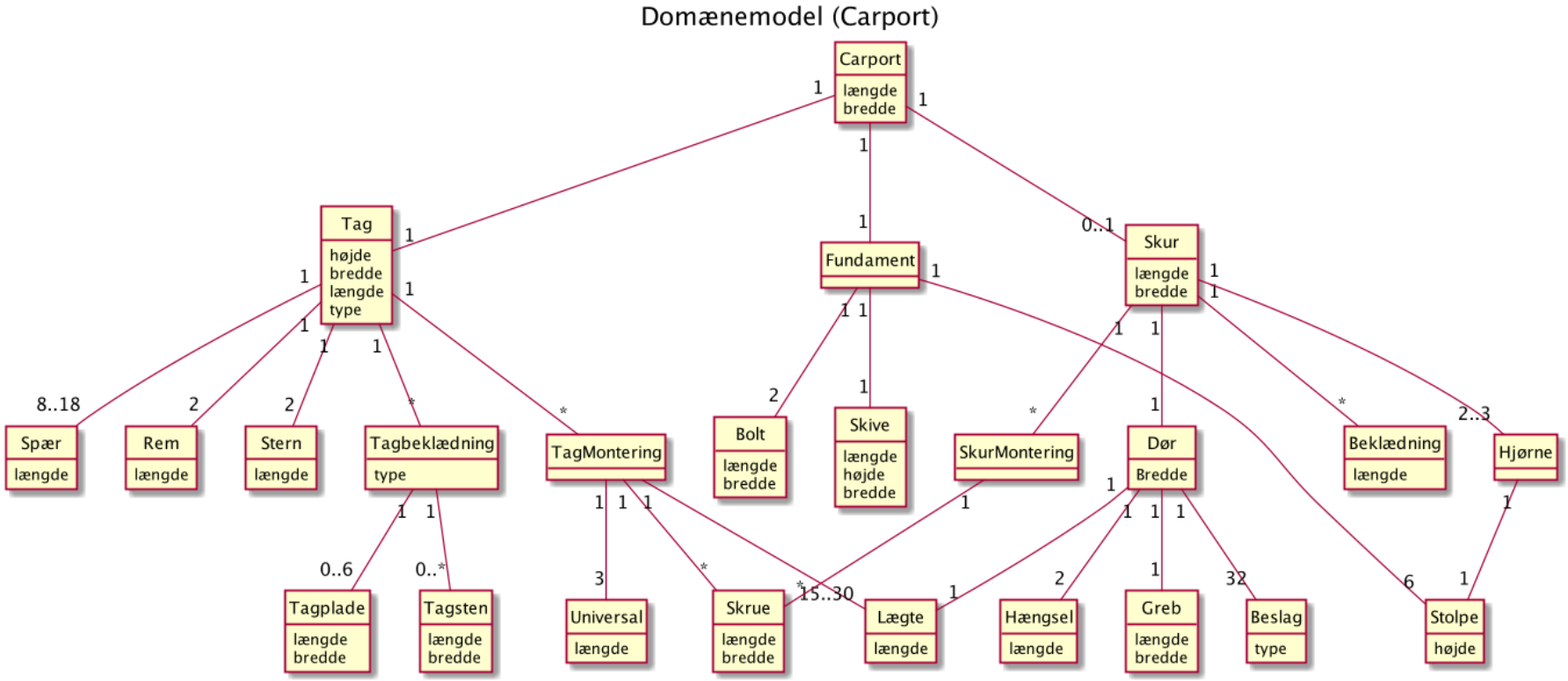
En Bestilling vil bestå af en kombination af Kunde, Carport, Stykliste, Sælger og Tilbud. Dette skyldes, at kunden skal indtaste sine behov på hjemmesiden, før bestillingen kan modtages af en sælger. Derfor skal en Bestilling have én Kunde, Carport og Stykliste. Carporten kan evt. have et Skur tilknyttet, hvis kunden ønsker det. Derfor har en Carport ét eller intet Skur. Styklisten er genereret ud fra kundens behov, og vil derfor inkl. materiel til et skur, hvis dette er blevet valgt til under bestillingen.

Hvis vi zoomer helt ind på Styklisten, så vil denne bestå af forskellige Styk. Disse Styk vil være unikke til den specifikke Stykliste, men Styklisten kan godt have mange Styk tilknyttet. Hvert Styk bliver skabt ved, at ét Materiale får tilknyttet en beskrivelse, pris og evt. længde. På den måde peger Styk på et specifikt Materiale, mens der godt kan indgå flere bestillingsdefinerede Materialer i vores Styk.

Grunden til at Stykliste, Sælger og Tilbud har en "én eller ingen" relation er, at vi ikke nødvendigvis behøver at tilknytte en Sælger til en Bestilling fra starten af. Dette kan godt ske længere henne i tilbudsprocessen. Derfor behøver vi heller ikke at tilføje et Tilbud eller en Stykliste. Det er først når en Sælger bliver tilknyttet, at disse domæner bliver aktiveret. Der er ingen grund til fx at gemme en stykliste i en database, hvis ikke en sælger har set på et tilbud. Men det betyder ikke, at vores system ikke kan generere en tentativ stykliste, som kan rettes til, men som først bliver gemt, når sælger sender et tilbud til kunden.



Den næste fase i vores domænemodel, er at forstå carporten. Det er vigtigt, at vi forstår de dele, som skaber en carport. Ved at forstå carporten, så kommer vi også til at forstå den algoritme, som vi skal bruge når vores Styk skal udregnes. Derfor vælger vi at dele en carport op i tre elementer, som vi kalder Tag, Fundament og Skur. Ved at dele carporten op i tre, så skaber vi et bedre overblik over de elementer vi skal bruge og hvor de hører til. Bl.a. kan vi se ud fra modellen, at vi ikke behøver et Skur, og derved behøver vi ikke Beklædning, Dør, SkurMontering og Hjørne. Så i vores algoritme og forståelse, kan vi nu afgrænse en carport med disse elementer, hvis carporten ikke skal have et skur tilknyttet.


Domænemodel (Carport)
Carport
længde
bredde
Tag
højde
bredde
længde
type
Fundament
Skur
længde
bredde
Spær
længde
Rem
længde
Stern
længde
Tagbeklædning
type
TagMontering
Bolt
længde
bredde
Skive
længde
højde
bredde
SkurMontering
Dør
Bredde
Beklædning
længde
Hjørne
Tagplade
længde
bredde
Tagsten
længde
bredde
Universal
længde
Skrue
længde
bredde
Lægte
længde
Hængsel
længde
Greb
længde
bredde
Beslag
type
Stolpe
højde

Ser vi på en carports Fundament, så bliver det også indlysende at stolpe mængden er afhængig af om en carport er med eller uden skur. Da vi har afgrænset vores carport mht. længden, så ved vi at en carport som minimum altid skal bruge 6 stolper. Men vi skal derefter skabe en algoritme som konstruerer 2 eller 3 stolper afhængig af skurets design. Carportens Tag spiller også en vigtig rolle, da tagets type er afgørende for tagets materiale. Vi har valgt at samle tagplader og tagsten under et domæne som vi kalder for Tagbeklædning, og som er afhængig af tagtypen.

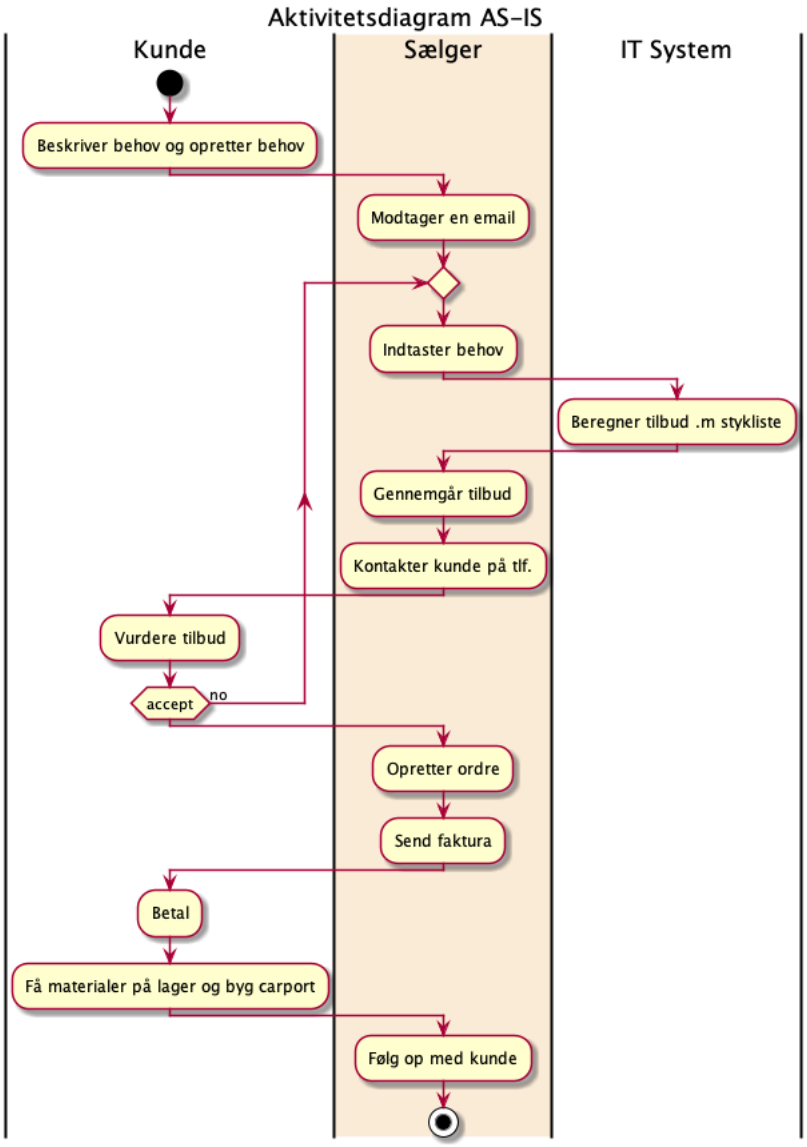
Når vi senere sammenligner vores domænemodel med vores User Cases, så hjælper domænemodellen os med at forstå, hvilken information de forskellige User Cases skal indhente. Det vil også give os en bedre forståelse for de datatyper som vi bliver nødt til at indhente. Bl.a. kan vi ud fra domænemodel få et hurtigt overblik over hvilken datatyper der er ”String”, ”Decimal” eller ”Integer”, som vi også komme til at benytte i vores ER diagram til vores MySQL database.

Aktivitetsdiagram

I dette diagram, vil vi fokusere på at fortælle kundens historie, når kunden ønsker at bestille en carport med brugerdefinerede mål fra Fog’s hjemmeside. Vi vil dele aktivitetsdiagrammet op i to. Først vil vi fortælle hvad der sker under den nuværende situation, og derefter vil vi fortælle hvordan vi ønsker den fremtidig situation skal se ud.

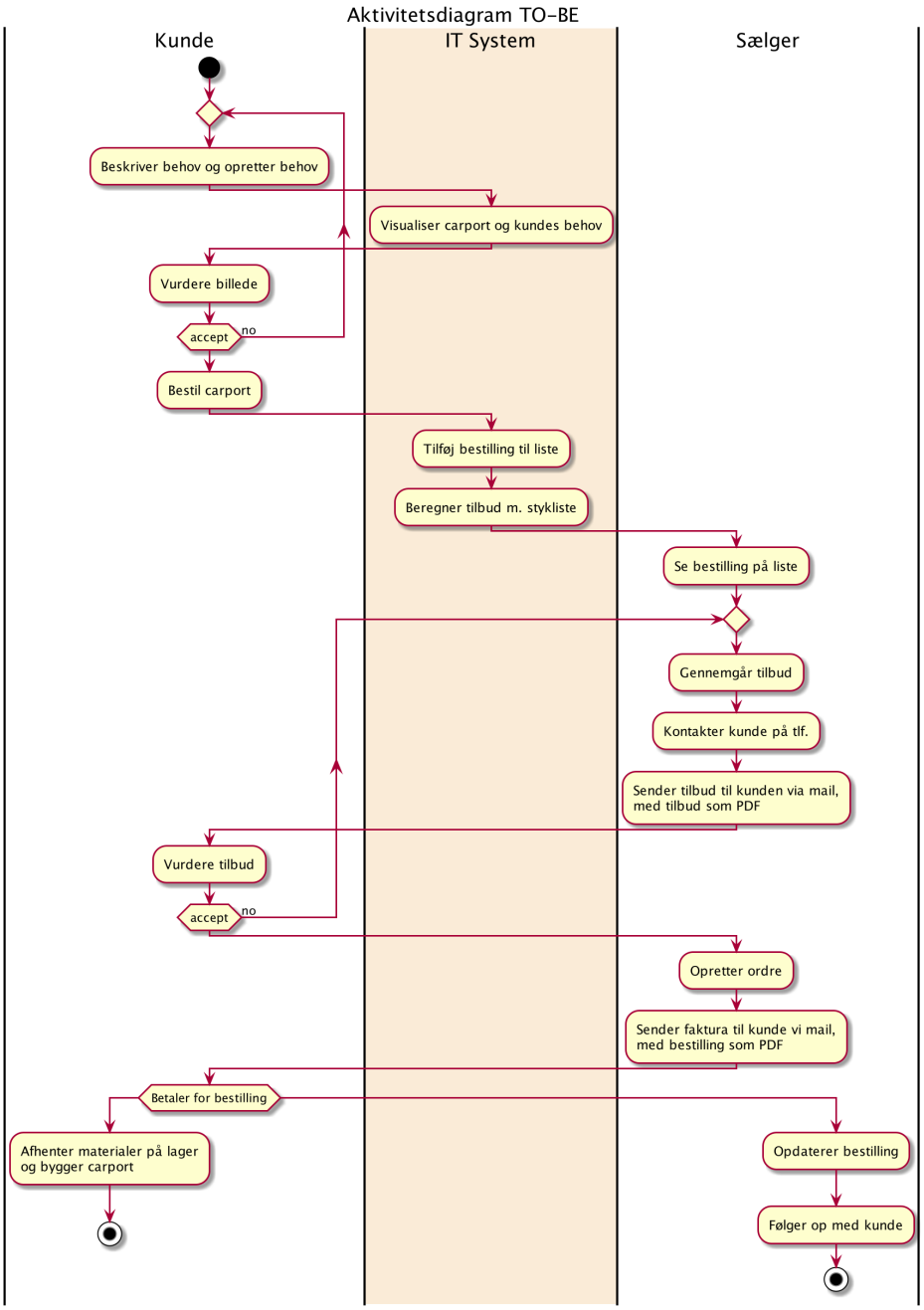
Ser vi på den nuværende situation, så får en sælger en mail med kundens behov. Sælgeren indtaster derefter kundens behov i deres nuværende IT-system, som udregner et tilbud. Sælgeren har nu begrænset mulighed for at ændre i tilbuddet, og kan blot gennemgå det for åbenlyste fejl. Bl.a. kan varebeskrivelsen eller navne ikke ændres.

Når sælgeren har gennemgået tilbuddet, kontakter sælger kunden via tlf. Sælgeren har ingen måde til at visualisere tilbuddet overfor kunden på via fx en plantegning. Accepterer kunden tilbuddet, så skal sælgeren oprette en ordre og sende en faktura. Derefter skal kunden betale og afhente materialet. Sælgeren vil efter et stykke tid opsøge kunden mht. kundetilfredshed eller spørgsmål.



Ser vi på den fremtidige situation, så kommer kunden til at få sine behov visualiseret via hjemmesiden. Dette skal hjælpe kunden med at forstå produktet bedre. Når kunden har vurderet visualiseringen, kan kunden vælge at bestille carporten. Hjemmesiden vil derefter udregne en tentativ stykliste baseret på kundens behov, som vil indgå i en liste over bestillinger. Sælgeren vil i det fremtidige system have adgang til bestillingslisten, som også giver sælgeren mulighed for at ændre i de enkelte bestillinger. Bl.a. vil en sælger kunne ændre beskrivelsen, pris eller navn på materiale. De rettelser som sælgeren ændrer i bestillingen, vil generere en mail med en UUID, som sendes til kunden. Nu kan kunden visuelt se sin bestilling og vurdere om bestillingen skal gennemføres.

Hvis bestillingen bliver gennemført, vil kunden igen få tilsendt en mail fra sælgeren med en PDF. Denne gang vil PDF'en indeholde visualiseringen af ordren, samt styklisten. Når kunden har betalt, vil kunden afhente materialet til carporten, mens sælgeren opdaterer status på ordren. Sælgeren vil efter et stykke tid opsøge kunden mht. kundetilfredshed eller spørgsmål.



Risikoanalyse

I denne analyse vil vi prøve at forudse de risici som vores projekt kan støde ind i. Det vigtige ved denne analyser er at tage hånd om de risici som vi vurderer som høje, så tidshorisonten og budgettet for projektet overholdes. Vi har valgt at fokuserer på alle de risici som er over 10. Her ser vi, at case 7 har en risiko på 15, som vi mener er bekymrende (gul), mens case 6 og 9 er alarmerende (rød) og er dem vi skal have primært fokus på.

#	Hvad kan gå galt?	K	S	K * S	Forebyggende handlinger	Afbødende handlinger
1	Salgsmedarbejder siger op	4	2	8	Vidensdeling og dokumentation	Aftale med kunden om at de finder en afløser hurtigt. Kunden skal have en højre hånd til salgsmedarbejderen i hele processen.
2	Sygdom pga. COVID-19	2	4	8	Vidensdeling og dokumentation	Aftale med gruppemedlemmerne hvem der overtager opgaver som anden mand.
3	Vi overskrider deadline	5	2	10	Vi skal allokere alt vores tid til at løse opgave hurtigst muligt	Forhandle med kunden.
4	Gruppemedlem stopper	2	2	4	Vidensdeling og dokumentation	Aftale med gruppemedlemmerne hvem der overtager opgaver som anden mand.
5	Manglende algoritme til stk. beregning	3	3	9	Have en alternativ udregningsmetode klar	Aftale med product owner at regnemetoden bliver implementeret anderledes.
6	Manglende erfaring med SVG	4	5	20	Læse op på SVG på nettet	Aftale med product owner vigtigheden af SVG-tegningerne, samt alternativer.
7	Droplet går ned	5	3	15	Have en backup droplet	Aftale i gruppen hvem der står for backup
8	Andet IT går ned	3	3	9	Kontinuerlig backup	Alle i gruppen tager kontinuerlig backup
9	Bugs	4	5	20	Kontinuerlig test af IT-systemet	Alle i gruppen udfører kontinuerlig test

Almen risiko - Bekymrende risiko - Alarmerende risiko

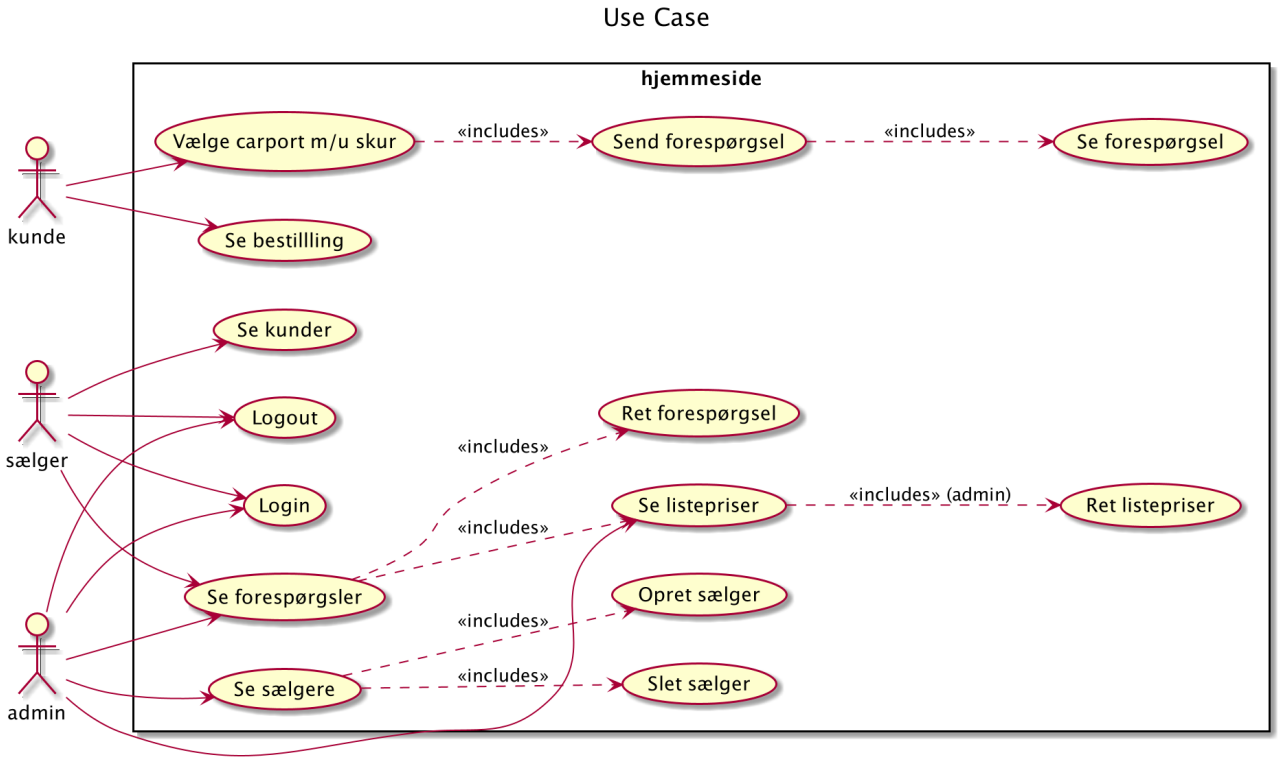
K: Konsekvens for projektet hvis problemet opstår fra 1-5, hvor 1 er lav og 5 er højest.
S: Sandsynlighed for at et problem opstår fra 1-5, hvor 1 er lav og 5 er højest.
K * S: Risikotal. Produktet af K og S. Det indikerer hvor stor en risiko er, og anvendes til at prioritere indsatser.

Use Case Diagram

Ud fra de User Stories som er blevet udarbejdet, kan vi nu konstruere en Use Case for hvad en kunde, sælger og en administrator (admin) skal kunne tilgå på hjemmesiden. Denne Use Case giver os et bedre overblik over de elementer vi skal have med i vores tanker, arkitektur og kodning. Det er her vigtigt at bemærke, at hver aktør skal have adgang til forskellige elementer på hjemmesiden, samt at det kun er få elementer, som kunden skal tilgå.

Ser vi på modellen, så fremgår det, at sælger og admin begge skal kunne se en specifik forespørgsel og listepris. Ser vi mere isoleret på kunden, så er kundens Use Case kun koncentreret på hjemmesidens front-end, hvor sælger og admin primært er koncentreret på hjemmesidens back-end. Denne information kan vi bl.a. bruge til at strukturere vores kald til databasen, da vi nu ved, at kunderne primært skal bruge GET kald, og sælger og admin POST kald. Dette skyldes, at kunderne ikke skal ændre noget på hjemmesiden.

Use Case giver os også en godt overblik over, hvilken jsp sider der skal udvikles. Bl.a. kan vi se, at en admin skal have lov til at tilgå databasen mht. rette listepriserne, mens en sælger kun må se listepriser. Dette har vi valgt for at skabe gennemsigtighed i systemet, så afdelingslederne kan gennemgå systemet for snyd eller fejl.



Implementering

I de næste par afsnit vil vi kigge på implementeringen af projektet. Vi vil gennemgå de valgt vi har truffet og hvilken muligheder og konsekvenser de har haft. Vi vil prøve at give et helhedsbillede af de beslutninger vi har taget, samt hvilke mulige løsninger der kan traffes i fremtiden.

ER Diagram

ER diagrammet er blevet udarbejdet på baggrund af domænemodellen. I denne model kan vi se, hvilke datatyper vi kommer til at benytte os af. Den viser også forholdet mellem de forskellige klasser, der i MySQL databasen bliver konverteret til tabeller, der indeholder data til objekternes variabler. ER diagrammet er således en arkitektonisk tegning over vores database, men også over vores hjemmeside.

Kigger vi på vores domænemodel og ER diagram, så svarer vores Sælger klasse i domænemodellen til users tabellen i vores ER diagrammet.
Som i domænemodellen, kan en users godt have flere ordre, men en specifik ordre kan kun have én user. På samme måde som Sælger passer til users, passer Kunde, Bestilling og Carport til orders, customers og carports.
En Bestilling (orders) kan kun bestå af én sælger, kunde eller carport. Dog behøver vores orders ikke have users tilknyttet fra startet af, da det er noget vores hjemmeside tilknytter vores orders senere i processen. Derfor kan orders have én eller ingen users til at starte med.

På samme måde som vores orders ikke behøver at have users fra starten af, så behøver vores carports ikke at have sheds (skur) tilknyttet, da dette er noget Fog's kunder kan vælge til eller fra via bestillingsprocessen. Selve partslists er noget som carports skal have tilknyttet én af. Da carports kan bestå af mange parts, men parts også kan indgå i mange carports, vælger vi at lave en partslist tabel, for at undgå mange til mange relation. Partslist tabellen gør at vi nu har mange til én relation mellem carport og partslist og partslist og parts.

Fokuserer vi ind på parts, så har vi her en mange til én relation mellem parts og usage. Dette skyldes, at usage godt kan indgå i mange parts, men at hvert af vores parts kun har én usage. Det samme gælder for usage og de to tabeller type og materiale. De kan indgå i mange usage, men hver usage kan kun have én type og materiale. Det smarte ved at dele databasen op på denne måde er, at vi kan have materiale liggende i vores database, som først får et reelt formål mht. carporten, når materiale får tilknyttet én type og usage.

Tilknytningen mellem tabellerne sker via primære- og fremmednøgler. En primær nøgle er et indekseret ID, som hver række i en tabel får, og som er unik. I alle vores tabeller har vi valgt at bruge datatype "Integer" til at generere rækkernes ID'er. Vi har også valgt at lade ID'en stige automatisk, når vi indsætter flere rækker ind i vores tabeller. På denne måde sikre vi os, at to primærnøgler aldrig er ens, samt at den enkelte nøgle aldrig er tom (null)[3].

[3] Vores automatiske id-stigning (AI) starter fra 202001, så ordrenumre bliver mere realistiske og passer til oprettelsesåret.

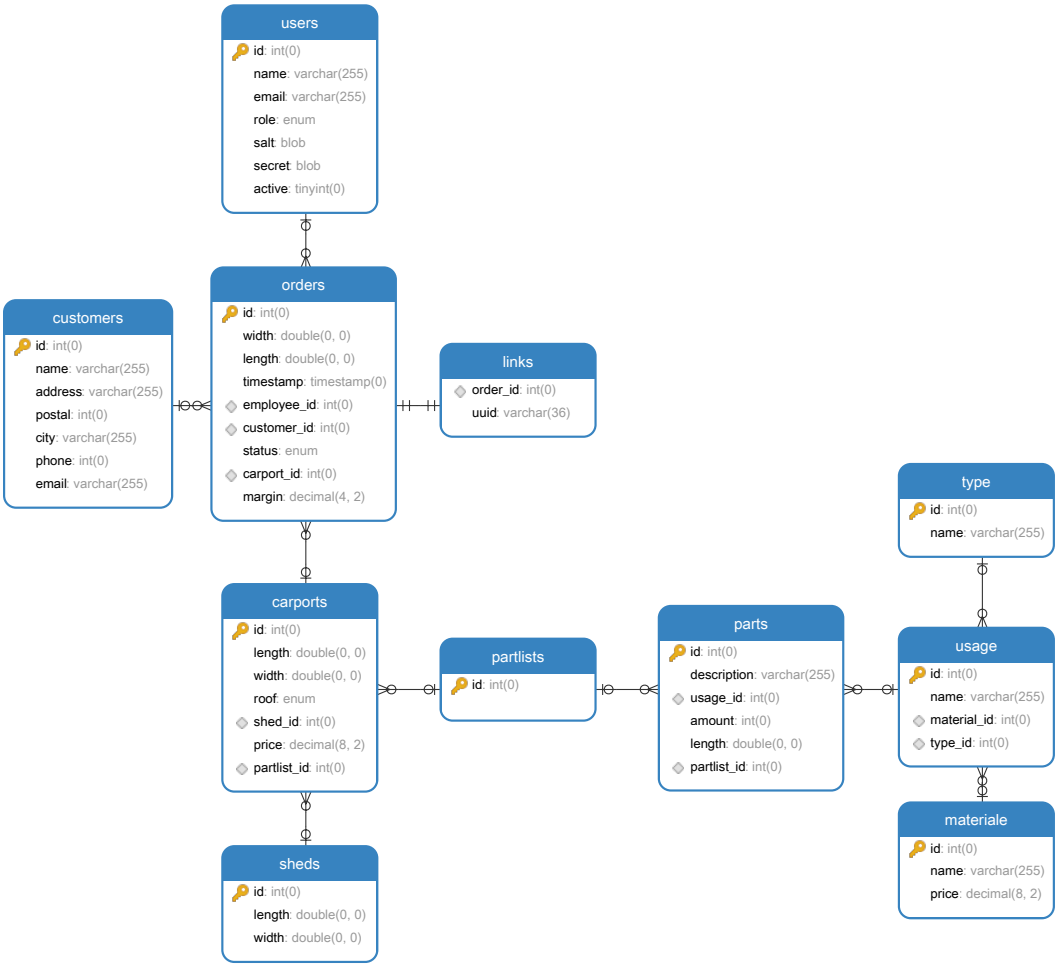
Fremmednøglen placeres i en anden tabel end hvor primærnøglen befinder sig, og er således en reference til primærnøglen. I tabellen users er ”id” vores primærnøgle, og i tabellen orders indgår den som en fremmednøgle via ”employee_id”. På den måde kan en række i orders tabellen kun have én users via ”employee_id” som peger på users ”id”. Dog har vi valgt, at ”employee_id” godt kan være null når orders oprettes. Dette skyldes, at en sælger (users) først bliver tilknyttet en ordre senere i bestillingsprocessen, end når selve orders bliver oprettet i vores database.

For at gøre databasen så nem at vedligeholde som muligt, er det vigtigt, at databasen opfylder 3. normalform. Det betyder bl.a. at der ikke må være gentagende data i vores tabeller, samt at man har mange til mange relationer. Vores database opfylder disse krav, dog er der steder hvor vi har valgt at afvige fra 3. normalform, da det enten øger MySQL databasens ydeevne eller data ikke er retvisende når vi laver ”JOIN” operationer.

I ”customers” tabellen har vi valgt ikke at opdele postal i sin egen tabel og afviger fra 3. normalform, da vi mener at det er kundens ansvar at indsætte det rette postnummer, da vi ellers har ansvaret for løbende at opdatere alle postnumre i vores system. Et anden sted vi afviger fra normalformen, er i ”carports” tabellen. Her gemmer vi prisen på carporten, selv om vi kan fremkalde data via ” Join”. Grunden til at vi vælger at gemme den eksisterende pris er, at det for det første er ydelsesmæssigt krævende for databasen at udføre ” Join” operationer, og for det andet, så får vi ikke et retvisende billede af prisen, hvis ”materiale” priserne bliver ændret i fremtiden, da carportprisen vil reflektere den nye pris ved ændringer. Det sidste kan vi undgå ved at have en tabel som bliver opdateret med de gamle udgående priser vha. af ”Trigger”. Men denne tabel kan sænke databasens ydeevne, da vi nu skal have mere komplekse ”Join” operationer. Vi ser mere ”Trigger” som en god måde at videreudvikle databasen på, så vi kan inkl. struktureret logning, som kan bruges til senere audit af hjemmesiden.

Som en ekstra bemærkning kan nævnes, at vi har valgt at gøre users email unik (constraints). Dette har vi valgt, da vi ikke ønsker at to sælgere har samme e-mail. På den måde, kan hver sælger med tiden få tilknyttet en mailkonto, hvor virksomhedsbeskeder kan sendes til. Vi benytter os også af Enum som datatype. Dette gør vi for at validere den data som bliver indsat i vores tabel, da Enum’s også kan bruges til validering. Vores links tabel benytter vi som en reference til orders. Denne benytter vi, når vi sender personlige links ud til kunderne. Når en kunde klikker på sit tilsendte link, så vil kunden kunne se ordren som er knyttet til linket. Da linket kun kan referere én ordre, som er forholdet én til én.

Hvis vi skulle videreudvikle vores database, så kunne vi med fordel inkl. Trigger’s, som kan bruges til logning af materiale prisændringer, men de kan også bruges til fx logning af onlineaktivitet, ændring af users og orders. Implementering af View’s og Stored Procedures kan sikre, at databasens ydeevne og sikkerhed forøges.



Navigationsdiagram

Når man som bruger besøger Fog's hjemmesiden, er der 4 veje man kan gå. I dette afsnit vil vi gennemgå brugernavigationen (sælger, admin og kunde) på hjemmesiden, samt hvad de forskellige trin betyder. Navigationsdiagrammet er således en mere detaljeret plan af vores aktivitetsdiagram, som skal sikre, at kunden, sælger og admin kan tilgå alle de cases som vi kom frem til i vores User Case. Vi vil derfor starte med at forklare navigationen ud fra en ny kunde som besøger hjemmesiden.

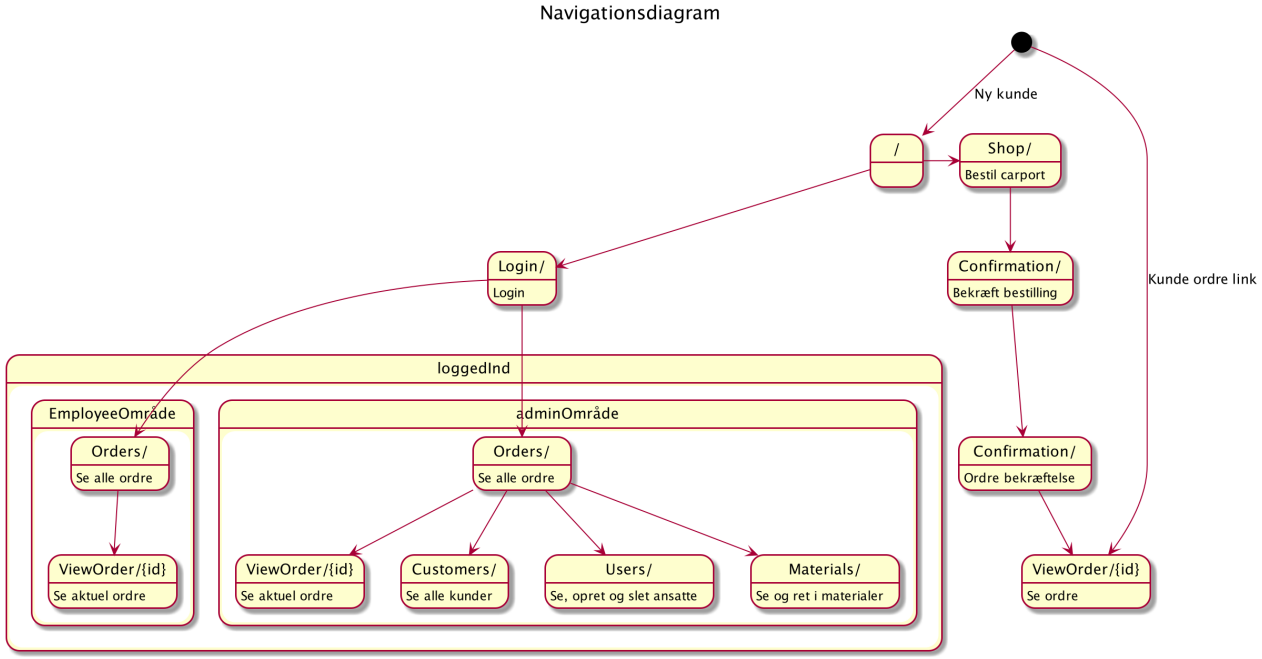
Som ny kunde bliver man præsenteret med følgende muligheder fra index.jsp ("/"). Man kan logge ind, eller udforske hjemmesidens produkter. Da man som kunde ikke har et login, vil en ny kunde udforske produkterne, der i dette tilfælde er en side der hjælper kunden med at konstruerer en carport. Når kunden har valgt de specifikke mål og tilvalg, vil kunden blive præsenteret med en tegning af carporten, samt mulighed for at indtaste sine kontaktoplysninger. Når dette er udfyldt og forespørgslen er send, vil kunden få en bekræftelse af afsendingen af forespørgslen, samt få en mail. Når kunden kommer til dette trin, så er navigationen for den nye kunde slut, og kunden kan forlade hjemmesiden.

Som sælger og admin er navigationen lidt anderledes. De starter begge på index.jsp, hvor de vil vælge at klikke på login, som fører dem til login.jsp. Derfra sker der en validering af den logininformation som bliver indtastet. Bliver valideringen godkendt, så kommer både sælger og admin ind i vores back-end. Her vil sælger og admin have hver deres separate muligheder via navigationen, som starter på orders.jsp (”/Orders”).

Som sælger får man mulighed for at se nye ordre, tilføje sig som sælger på nye ordre, se listepriser på ordre, samt opdatere ordre. Som admin har man lidt flere muligheder. Her kan man oprette nye materialer, ændre og oprette sælgere, ændre i ordre, samt få et overblik over alle ordre. Vi har opdelt det på denne måde, da vi mener at det giver mening, at en admin har karakter af en leder på arbejdspladsen, og at sælgernes aktivitet ikke skal inkl. alt, men kun det mest nødvendige. Derfor kan en sælger bl.a. ikke se andre sælgers ordre, men kun de ordrer som ikke har en sælger tilknyttet.

Da vi har valgt ikke at oprette kunderne med tilhørende login, så kan tilbagevendende kunder der har bestilt en carport, besøge hjemmesiden via et unikt link. Dette link sikrer, at kundes ordre kun bliver vist til den kunde der har det rette link. Når en kunde besøger hjemmesiden via dette link, så kommer kunden direkte ind på ordren og skal derfor ikke forbi index.jsp.

Navigationsdiagrammet viser os, at Fog’s hjemmeside kommer til at have 3 stadier ”states”. Man kan som kunder være en ny kunde der skal bestille en carport, eller som eksisterende kunde der tilgår hjemmesiden for at se sin eksisterende ordre via sit personlig link (1. state). Eller man kan man være en sælger eller admin og være logget ind, men have hvert sit isolerede område på hjemmeside (2 og 3 state).



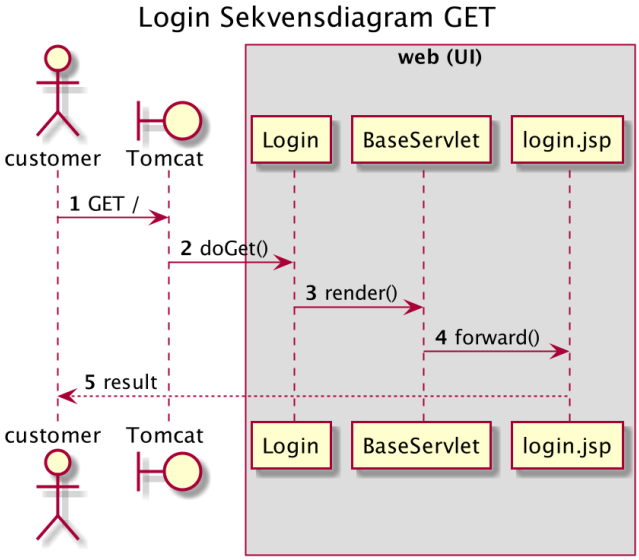
Sekvensdiagram

Vi vil i de næste to sekvensdiagrammer komme mere detaljeret ind på hjemmesidens login sekvens. Vi har tilføjet arkitekturen til diagrammerne, for bedre at illustrere hvor i systemet de forskellige handlinger sker. En gennemgang af arkitekturen vil komme i afsnittet om klassediagrammet.

Ud fra navigationsdiagrammet konkluderede vi, at der er 3 stadier på Fog's hjemmeside. Vi vil nu gennemgå loginsekvensen, som en sælger eller admin skal igennem for at logge ind på hjemmesiden.

Som sælger eller admin vil man lave et GET kald til vores Tomcat server på "/", som derefter vil lave et doGet() via vores Login servlet. Denne vil så lave en render(), som er en exstension af vores BaseServlet. Det er render() opgave, at vi bliver forward til den jsp side, som indgår i render() argumentet. Resultatet vil derefter blive sendt tilbage til vores kunde, som nu vil kunne se vores login.jsp side.

Under hele GET-sekvensen, vil sælger eller admin kun blive præsenteret for vores "web" lag i vores arkitektur. Dette lag er også vores User Interface, som er hvad sælger eller admin bliver præsenteret for, når sælger eller admin besøger hjemmesiden.



Næste skridt i loginfasen er, at sælger eller admin sender et POST til /Login på vores Tomcat server, som derefter sender et doPost() til vores Login servlet. I denne POST vil de informationer indgå, som sælger eller admin har indtastet i login.jsp formen <form>. Grunden til at vi sender et POST og ikke GET er, at vi nu ikke blot skal modtage data, men vi skal også håndtere denne data i vores database.

Vores servlet Login kalder så vores login(email, password) metode i vores API. API'et er der hvor alt vores business logik bliver styret. Grunden til at den kender til vores domæne er, at den bliver

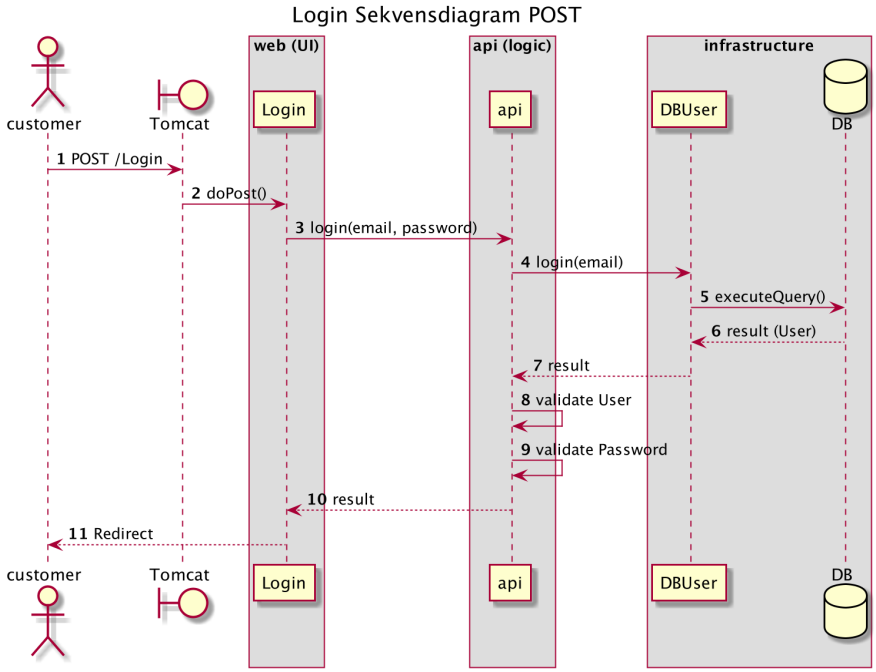
injiceret via vores BaseServlet, som Login extender. Dette vil fremgå mere tydligt i afsnittet omhandlende klassediagrammet og Onion arkitektur.

Fra vores API, bliver metoden login(email) kaldt i vores DBUser. Herfra bliver et database query konstrueret og kaldt via executeQuery(). Disse sidste kald sker i vores infrastrukturlag, som er det lag som står for datakilden. I dette tilfælde er det en database.

Resultatet vi får tilbage fra databasen, er et User objekt, som bliver sendt retur til vores API. Vores API gennemfører så en validering af den data den modtager. Valideringen har to faser. Først undersøger API'et, om det er et korrekt User objekt den har fået retur. Det gør den ved at sammenligne User objektet med vores domæne User, da API'et kender til vores domæne via vores injicering. Dernæst undersøger den, om sælgers eller admins kodeord er validt i forhold til krypteringen.

Hvis valideringen fejler i vores API, så vil API'et kaste en exception, som så bliver sendt retur. Bliver valideringen gennemført, vil User objektet blive sendt retur. Punkt nummer 10 afhænger derfor af denne validering. Hvis valideringen gennemføres, bliver sælgers eller admins User objekt gemt som en session i vores system, og sælger eller admin bliver omdirigeret til sælger eller admin området ("/Orders"). Fejler valideringen, vil punkt nummer 11 være en omdirigering til login.jsp ("/Login"). På denne side vil sælger eller admin se en advarsel, som svarer til den exception som blev kastet. Begge omdirigeringerne i begge senarier sker via doGet().

Vores exceptions bliver en måde for os at forstår, hvad det er der er gået galt i systemet, samt forudse hvad vi forventer der kan gå galt i vores system. Som udgangspunkt behøver man ikke at sende en besked retur til brugeren ved fejlvalidering, men vi synes det giver god mening, samt gør systemet mere brugervenligt, at man informerer brugeren, at noget er gået galt, samt hvad det er der er gået galt.



Klassediagram og Onion arkitektur

I det her afsnit vil vi kigger på den arkitektur som er blevet valgt til projektet. Vi vil gøre dette vha. klassediagrammet, hvor vi har udvalgt en bestemt klasse, for at vise hvordan dens implementering og arv forplanter sig i arkitekturen.

Hvis vi starter med at kigge på vores domæne, så har vi der User klassen, som via sin konstruktør skaber et User objekt som indkapsler nogle data og noget funktionalitet (adfærd)[4]. UserFactory er en grænseflade (interface), som stilles til rådighed for objektet, og som skal "beskytte" objektet i at blive ændret af andre objekter. Dvs. at UserFactory er afhængig af User klassen. Andre objekter skal på denne måde tilgå grænsefladen for at få User objektet til at udføre en bestemt opgave. Det smarte med denne opbygning er, at vi i domænet kan bestemme hvilken opgave andre objekter får adgang til, samt hvilket objekt eller datatype som disse objekter får retur. På den måde behøver andre dele af programmet ikke at vide hvordan opgaven udføres, eller hvordan data repræsenteres, men blot kende til de opgaver objektet kan udføre[5].

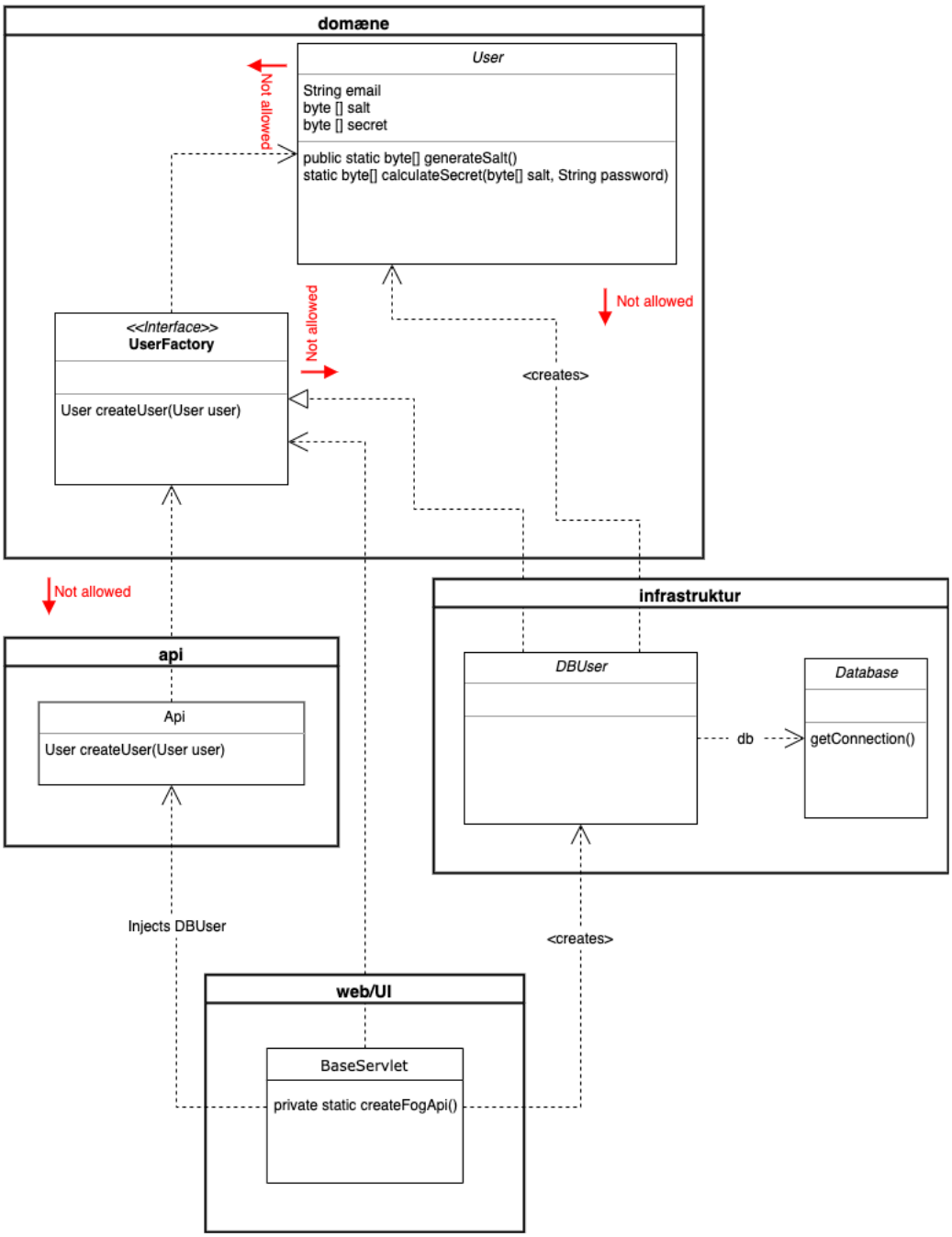
Et godt eksempel på overstående kan være UserFactory User createUser(User user). Når API'et kalder denne metode, behøver API'et ikke at kende til User objektets data og funktionalitet. Den ved blot, at der via UserFactory kan dannes et User objekt. Selve udførelsen af metoderne generateSalt() og calculateSecret() behøver API'et ikke at kende til. Den forventer bare at dette interface via User objektet kan skabe en ny User. Kan den ikke det, så vil API'et kaste en exception, hvis den ikke får det objekt retur, som den forventede at få.

Men hvordan kender API'et så til User objektet og domænet? Det gør den via web/UI-laget. I web/UI laget benytter vi os af et BaseServlet som web/UI. I vores BaseServlet skaber vi et statisk API vha. singleton, som alle servlets benytter når de extender BaseServlet. Denne statiske metode er afhængig af DBUser, som er afhængig af en database (db), for at få adgang via getConnection(). Og fordi DBUser arver metoderne fra UserFactory, så kan DBUser også skabe User objekter. På den måde bliver DBUser injiceret ind i vores API, hvor alt vores business logik befinder sig, uden at API'et behøver at kende til vores domæne.

Det smarte ved Onion arkitekturen er, at vores infrastruktur ikke behøver at bestå af en database. Vi benytter vi os bl.a. af LocalPartslist og LocalSVG, for at vise at disse objekter findes lokalt. Man kunne også tænke sig, at man i fremtiden ville udskifte databasen med en læselig fil eller et eksternt API, hvor data bliver modtaget via JSON. Så skulle vores DBUser måske hedde JSONUser, hvor dets opgaven er at konvertere JSON til et objekt og sortere i den, frem for at sende en query til en database.

[4] https://da.wikipedia.org/wiki/Objektorienteret_programmering
[5] https://en.wikipedia.org/wiki/Class_diagram

Mini Klassediagram - Onion Arkitektur



Særlige forhold

I dette afsnit vil vi gennemgå de særlige forhold, som hjemmesiden benytter sig af. Der er forhold som er med til at give en bedre brugeroplevelse, og så er der forhold der er med til at hjælpe os som programmør, til at finde fejl og mangler.

Session

Hjemmesiden benytter sig af sessions, for at give brugerne en bedre oplevelse. Disse sessions gemmes i cookies på kundens computer. Minusset ved brugen af sessions er, at hvis brugeren har deaktiveret cookie på computeren, så vil brugeren kun få en begrænset oplevelse af hjemmesiden, samt der vil være funktioner som ikke vil være tilgængelige.

På hjemmesiden benytter vi os bl.a. af sessions til at indeholde vores User objekt, til både backend og når en kunde opretter en ny bestilling. Grunden til vi gemmer sælger eller admin data i en session er, at så behøver vi ikke kalde databasen hver gang en sælger eller admin klikker på et link. Ved at sælger eller admin data er gemt i en session, så kan systemet finde ud af den givne brugeres rettigheder på siden. Det benytter vi os bl.a. af når vi skal logge en sælger eller admin ind på hjemmesiden. Sælger eller admin data fortæller derved hjemmesiden om sælger eller admin rolle er en employee eller en admin, samt hvilket id sælger eller admin har.

Alle sessions bliver automatisk slettet efter 30 minutters inaktivitet eller hvis en bruger benytter sig af log ud funktionen.

Exception

Vi benytter exceptions som en måde at skabe et bede overblik over fx if/else sætninger. Ved at benytte os af exceptions, tvinger vi os selv til at gennemtænke hvilken fejl der kan opstå på hjemmesiden. Bl.a. benytter vi os af klasserne InvalidPassword og UserNotFound som alle extender exceptions.

Ved at bruge exceptions på denne måde, så kan vi bedre forudse hvilken fejl der kan opstå, samt kontrollere hvilken fejlmeddelelse vi eller brugerne af hjemmesiden får, når en exception bliver kastet. Især i loginprocessen benytter vi os af disse exceptions, for at afgøre hvilken besked brugerne skal have via UI.

Kryptering

På hjemmesiden benytter vi os af PBKDF2[6] som anvender en pseudorandom-funktion, såsom hashbaseret meddelelsesgodkendelseskode (HMAC), til adgangskoden sammen med en saltværdi. Hashstandarten som vi benytter os af, er SHA256[7]. Sammen skaber de en hemmelig sætning, som bruges til at kryptere adgangskoden i databasen, samt til validering ved login.

[6] https://en.wikipedia.org/wiki/PBKDF2
[7] https://en.wikipedia.org/wiki/SHA-2

Validering og sikkerhed
Ud over krypteringen, benytter vi os af "role" validering. Når en bruger besøger hjemmesiden, så kan den besøgende ikke komme ind på sælger eller admin området, medmindre denne person opfylder "role" kravet.

Database
For at sikre os at alle udviklerne på projektet benytter sig af samme databaseopbygning, er der blevet skabt en Migrate klasse med en statisk main metode. Via sql scripts vil en database blive opdateret til nyeste version, så alle udviklerne har ens fungerende versioner. For at teste hjemmesiden, er der også blevet lavet en testdatabase med tilhørende testbruger, som alle benytter sig af.

UUID
Når en kunde skal tilgå sin ordre via hjemmeside, bliver den pågældende ordre fremkaldt via en 128 bit unik identifikationsnøgle, som hedder UUID. Det er en metode vi har valgt, for at sikre back-end delen. Vi mener ikke, at kunden nødvendigvis behøver at få et unikt login, for at give kunden værdi eller information om ordren[8].

E-mail
For at give Fog's kunder en god brugeroplevelse, har vi valgt at sende en e-mail ud til kunderne med deres ordrenummer og bestillingsinformationer. Der bliver sendt en e-mail når kunden sender er forespørgsel, samt når ordren er gennemført og betalt.

PDF
Kundens faktura ved betaling bliver sendt via en PDF i en mail. Denne PDF indeholder ordreinformationerne, samt tegning over carporten.

PNG
Da der kan opstå problemer ved at lagre SVG tegninger i PDF filer, har vi valgt at lave en SVG til PNG konvertering, således at vi er sikre på tegningerne altid vises korrekt.

Google Java Style Guide
For at få en så ensartet kode, har vi valgt at benytte os af Googles code style guide[9].
Dette sikre at al kode er ens formateret.

Synkronisering
Da vi kører alle vores tråde kald gennem en databases query, behøver vi ikke at bruge synkronisering. Grunden til dette er, at databasens query ikke accepterer at flere tråde kan tilgå samme query, og da vi ikke har querys der kommer sideløbende med hinanden, vil der ikke opstå deadlock eller datarace på vores hjemmeside.

[8] https://docs.oracle.com/javase/7/docs/API/java/util/UUID.html
[9] https://google.github.io/styleguide/javaguide.html

Singleton
Singleton er et designmønster hvor man begrænser instantiering af en klasse til en "enkelt" forekomst som man kan betragte som en ”global variable”. Dette er nyttigt, når præcis ét objekt er nødvendigt for at koordinere handlinger på tværs af systemet. Vores API er en singleton som bliver instantieret og delt på tværs af alle servlets. På den måde ved vi, at alle bruger det samme API[10].

Facade Pattern
Facade Pattern er et designmønster, til at maskere kompleks underliggende eller strukturel kode. Facaden består at et objekt, der fungerer som en grænseflade til resten af it systemets kode. Vores API fungerer som denne facade, da den holder styr på vores businesslogik og er ”indgangen” til vores komplekse kode[11].

Factory Pattern
Factory Pattern er et designmønster, som består af en grænseflade som kan oprette objekter. Det smarte med denne opbygning er, at vi via grænsefladen kan bestemme hvilken opgave andre objekter får adgang til, samt hvilket objekt eller datatype som disse objekter får retur. På den måde behøver andre dele af programmet ikke at vide hvordan opgaven udføres, eller hvordan data repræsenteres, men blot kende til de opgaver objektet kan udføre. Factory Pattern beskytter derved vores klasser i domænet. Et godt eksempel er vores User klasse, som har grænsefladen UserFacoty, der opretter User’s, men som bliver tilgået af DBUser, uden at DBUser kender til den underliggende logik i User klassen.

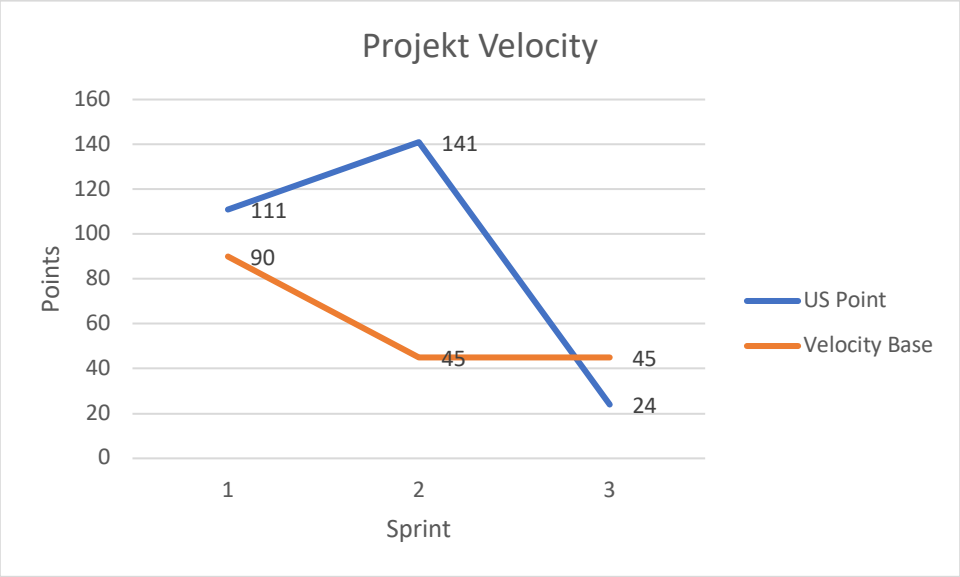
Evaluering
I de næste par afsnit vil vi evaluere vores projekt. Vi vil gennemgå projektet og resultatet af vores implementering. Vi vil se på testningen og logningen af hjemmesiden, samt evaluerer vores egen indsats og arbejdsbyrde i forhold til hinanden og Scrum. Dette gør vi for at evaluerer os selv, så vi kan blive bedre i forbindelse med fremtidige projekter.

Status på implementering
I dette afsnit vil vi komme ind på de udfordringer vi har haft med projektet, samt hvad status er på de enkelte User Stories.

Vi har haft mange udfordringer med vores servlets. Selv om to servlets er ens, så kan de agere på to forskellige måde, når hjemmesiden bliver genereret. Vi havde især problemer med redirects, som nogle gange ikke ville udføre det redirect som vi ønskede. Vi havde også problemer med vores SVG-tegning. Vi fandt ud af, at SVG ikke overholder de css begrænsninger vi satte for vores html div’r, og ende med at sætte begrænsningerne i selve SVG-koden.

[10] https://en.wikipedia.org/wiki/Singleton_pattern
[11] https://en.wikipedia.org/wiki/Facade_pattern

På User Story fronten, fik vi 5 nye User Stories midt i vores 2. sprint. Dette betød, at vi måtte ændre vores strategi mht. det ønskede færdigprodukt. Der var ikke tid til at implementerer alt den funktionalitet som vi ønskede, men som ikke nødvendigvis var ønsket af kunden. Derfor valgte vi kun at prioriterer kundens User Stories. Ændringen i opgaverne betød, at vi fik en markant stigning i vores velocity under sprint 2, som også viste os, at vi leverede over forventning, samt at vores velocity skal justeres mht. fremtidige opgaver, da vi når mere end forventet.



I det følgende skema kan man se status på alle de User Stories som opgaven har taget udgangspunk i. Farven indikere status på implementeringen.

NR	User Stories	Status
US-1	Som sælger skal man kunne logge ind, så sælger kan få adgang til kundernes forespørgsler og kontaktoplysninger.	
US-2	Som sælger skal man kunne se alle ordre, så sælger nemt kan få et overblik over, hvad status er på alle ordrerne.	
US-3	Som sælger skal det være tydeligt når en ny forespørgsel er kommet, så sælgerne ikke overser nye forespørgsler.	
US-4	Som sælger skal man kunne ændre dækningsgraden på et tilbud, så sælger måske alligevel kan få et salg igennem, når en kunde er tøvende.	
US-5	Som sælger skal man kunne se styklisten for en ordre, så sælger kan dobbelt tjekke om der er nok materialer til at lave en stabil konstruktion.	
US-6	Som sælger skal man kunne afslutte en ordre når fakturaen er betalt, så sælger nemt kan holde overblik over åbne sager.	
US-7	Som sælger skal det være muligt at redigere i salgsprisen, så sælgeren bedre kan matche kundens købspris og derved gennemføre et salg	
US-8	Som sælger skal man kunne slette en forespørgsel, så sælgerne kan holde styr på eksisterende forespørgsler.	
US-9	Som admin skal man kunne slette en sælger bruger, så programmet ikke bliver brugt af gamle sælgere som ikke arbejder der mere.	

US-10	Som admin skal man kunne rette i eksisterende materialer, så materialelisten er ”up to date” med det eksisterende lager.	
US-11	Som admin skal man kunne oprette en sælger i systemet, så ledelsen kan se forskel på hvilke sælgere der er tilknyttet til hvilke salg	
US-12	Som kunde får man tilsendt en bekræftelse e-mail med tegningen på carporten, så kunden har en ide om hvilken konstruktion kunden skal til at bygge.	
US-13	Som kunde skal man få tilsendt styklisten så snart tilbuddet er betalt, så kunden kan gå ud i tømmerhandlen og få udleveret materialer.	
US-14	Som kunde skal man kunne vælge et specifikt mål på en carport, så kunden kan blive kontaktet af en sælger mht. op følgende salg.	
US-15	Som kunde skal man kunne se en tegning af carporten inden forespørgslen bliver sendt, så kunden kan sikre sig at carporten opfylder kundens behov.	
US-16	Som admin skal man kunne se hvilken sælger der er tilknyttet til en pågældende ordre, så admin får et bedre overblik over salgsarbejdet og sælgernes salgsopgaver.	
US-17	Som admin skal man kunne ændre i den sælger der er tilknyttet en ordre, så andre sælgere har mulighed for at gennemføre et salg, hvis den tilknyttede sælger, ikke er til stede.	
US-18	Som sælger og admin skal man kunne se at man er logget ind på sin personlige profil, så sælger eller admin ikke kommer til at rette i andres profiler, eller bryder sikkerheden, vis kollegaer glemmer at logge ud.	
US-19	Som sælger og admin skal man nemt kunne se datoen for ordres oprettelse, så medarbejderne og ledelsen nemt kan danne sig et overblik over ordrenes ventetid.	
US-20	Som kunde skal man kunne se sin ordre på hjemmesiden når den er betalt, så kunden nemt kan få et overblik over ordren, styklisten og bestillingsinfoen.	

Testet og OK Virker med fejl Ikke udført

Test

Hjemmesiden er gennemtestet vha. følgende testformer. Unit test, integrations test, back-end test, usability test, ad hoc test og manuel test[12].

I Unit testen har vi testet vores klasser (domain), for at se om objekterne udfører de udregninger eller opgaver som de skal. Her har vi givet vores klasser fiktive data, så vi kan sikre os, at vi kender resultatet på forhånd. Vi har lagt ekstra fokus på at teste alle udregninger, da prisen på carporten afhænger af præcise udregninger. Da vi har forskellige materiale typer, som hver beregnes anderledes, har det været ekstra vigtigt at få disse udregninger præcise.

[12] https://www.softwaretestinghelp.com/types-of-software-testing/

Element	Class, %	Method, %	Line, %
domain.carport	50%	20%	26%
domain.material	77%	36%	64%
domain.order	50%	15%	26%
domain.partslist	33%	15%	26%
domain.svg	100%	100%	98%

I integrationens testen har vi testet API'et vha. af en test database. Alle tests er baseret på ikke fiktive data og kommer direkte fra test databasen. Vha. denne test, kan vi se om vores business logik passer, samt om de forskellige klasser og metoder udfører de opgaver som de skal. Da mange af vores metode kald i API'et omhandler send af e-mails, eller oprettelse af PDF'er, har vi valgt ikke at teste disse vha. integrationstesten, da disse bedst kan testes manuelt. Derfor ender vores samlede integrationstest på 54%.

Element	Class, %	Method, %	Line, %	Branch, %
API	20%	62%	54%	28%
domain	72%	67%	74%	21%
infrastructure	69%	56%	58%	37%

Back-end testen har vi brugt til at finde ud af, om vores data bliver gemt i vores database, når en kunde fx bestiller en ordre. Og usability test har vi brugt da vi skulle opbygge og teste vores Mock Up. Vi har forsøgt at have kundes ønsker for øje da vi byggede hjemmesiden, så brugervenligheden blev prioriteret.

Ad hoc testningen er de små tests vi har udført undervejs i projektet. Til slut har vi lavet en udførlig manuel test, som tester Fog's User Stories. Det er på baggrund af denne test, at vi kan konkludere, at projektet er løst og gennemtestet.

Logning

For at vedligeholde og fejlfinde bugs på hjemmesiden, har vi valgt at udføre ustruktureret logning. Vi logger "Info", "Debug", "Warning" og "Error" i vores Tomcat server log. Dette gør vi for at få et overblik over de handlinger som går galt på hjemmesiden. Det er vigtigt at vi hurtigt kan identificere fejl der sker og rette dem. Grunden til at vi deler dem op i kategorier er, at vi nemmere kan sortere i fejlene, samt løse de alvorlige fejl først.

Hver fejl der bliver logget, bliver gemt med information om klassen, tidspunkt, tråd, besked og kategori. Vi er gået efter "hvem, hvad og hvor" princippet, som vi har brugt flittigt under vores udvikling, når vi stødte på bugs.

JavaDoc

For at hjælpe fremtidige udviklere, har vi udgivet JavaDoc via gruppens GitHub profil. JavaDoc koncentrerer sig primært om business logikken i API'et. Pt. dækker den alle de offentlige metoderne i vores API. Vi har prøvet at leve op til markedsstandarderne inden for dokumentation. Dokumentationen kan findes via følgende link:
https://eelkjaer.github.io/Fog/

Videreudvikling

I dette afsnit vil vi komme ind på de fremtidige videreudviklingsmuligheder og ideer som vi har, og synes Fog skal overveje at få implementeret.

- Implementering af Trigger's i databasen for at implementerer struktureret logning. Via Trigger's kan vi gemme udgået data, som Fog kan bruge til deres audits.
- Mulighed for at admin kan tilføje materiale til materialelisten, og som bliver medregnet i en fremtidig carport.
- Videreudvikling af back-end, så den inkl. lagermedarbejderne og deres arbejdsgang.
- Implementering af lagerstyring, så lagermedarbejderne og sælgerne kan få et bedre overblik over lagerstatus. Her vil en mulighed for at genbestille et lager være en fordel.
- Fastsættelse af KPI'er af admin til medarbejderne, samt en kontrol og oversigt over KPI-målene fordelt ud på medarbejderne.
- Udbygning af statistikken så den inkl. enkeltsalg, KPI'er, afdelingssalg, mål, budgetter, prognoser, afleveringstider osv.
- Beskedsystem og opslagstavle som lederne og sælgerne kan bruge som internt kommunikationsværktøj.
- Logning af medarbejdernes brug af systemet, så ledelsen kan følge medarbejdernes arbejde mht. audit og LEAN udvikling.
- Implementering af et salgsakademi, som skal bruges til at oplære nye medarbejdere i systemet.

Gruppearbejde

I dette afsnit vil vi komme kort ind på vores gruppearbejde og de tanker vi har haft under projektet. Vi fik også fortaget en Belbin personlighedstest, som blev linket til gruppemedlemmerne, for at give os en bedre forståelse for vores styrker og svagheder internt i gruppen.

I starten af projektet gjorde vi det meget klart i gruppen, at vi ville være proaktive mht. projektet og alle ønskede et godt resultat. Vi havde alle i gruppen et eksperimenterende mind-set som gjorde, at vi blev enige om at holde os så meget som muligt til opgavens specifikationer. Dette blev også bekræftet i vores Belbin test, hvor det tydeligt kom frem, at vi var drevet af innovation og ideer.

Men på trods af vores innovative mind-set, så fortalte Belbin testen os også, at vores gruppe var i balance. Vi var gode til at få ideer, men også iværksætte disse ideer. Vi var dog opmærksomme på, at teams der arbejder i et højt gear, har fare for at brænde sammen med tiden. Derfor indførte vi teambuilding dage, som bestod af pizza, øl og socialt samvær. Vi var også gode til at uddelegere ansvar til gruppemedlemmerne, da vi både stolede på hinandens arbejdsindsats, samt ønskede at arbejde mere agilt med Scrum. Dette resulterede også i, at vi gav gruppemedlemmer frit spil til at udforske nye ideer, hvis tiden var til det. En beslutning som har givet os en mere kompleks og professionel hjemmeside, synes vi selv.

Konklusion

I vores konklusion vil vi komme ind på nogle af de tanker og udfordringer, som vi har haft og erfaret gennem projektet. Vi vil prøve at binde projektet sammen, så vi kan belyse projektet i et større perspektiv.

Til at starte med, så synes vi, at vi har udarbejdet og udviklet en flot og funktionel hjemmeside. Et af vores mål i gruppen var, ikke at lade os selv rive med mht. funktionalitet, så vi kunne gennemføre kvalitet frem for kvantitet. Derfor valgte vi at afgrænse projektet og holde os til specifikationerne og vores interne mål. Dog prøvede vi under vejs at udvikle projektet vha. andre teknologier som fx PDF og PNG. Den nye teknologi havde sine udfordringer, og derfor udviklede vi altid to mulige implantationer hvis teknologien viste sig ikke at virke.

Under projektet fik vi en bedre forståelse for bl.a. udarbejdelsen af domænemodellen og ER diagrammet. Vi så især domænemodellen som et overordnet syn på projektet, men fandt hurtigt ud af, at vi skulle meget mere ned i detaljerne når det kom til udarbejdelsen af domænet. Da vi først havde udarbejdet domænet, så vi styrken i at have detaljerne på plads når det kom til carporten og dens enkelte dele. Det gav os et bedre overblik, at man nu kunne have et domæne i domænet, når man skulle beskrive mere detaljerede sammensætninger.

ER diagrammet gav os nogle problemer, og vi følte os fastlåste i vores udvikling, så længe vi ikke havde et funktionelt diagram. Erfaringen her i ligger, at vi skal blive bedre til at udvikle ER diagrammet undervejs, og ikke fastlåse os selv til en bestemt arkitektur. Dette kunne have sparet os for ca. 3-4 dages udvikling, da vi gik i stå så længe databasen ikke kunne implementeres. Til gengæld fik vi et mere fleksibelt ER diagram i forhold til Fog's krav og ønsker. Da vi startede projektet, var vores plan at hardcode carportens opbygning, men vi endte med at udvikle en database med fleksible muligheder, hvor Fog kan ændre i priser og materialenavne. En fremtidig udvikling af databasen vil med fordel kunne inkl. Trigger's, View's og Stored Procedures. Trigger's kan bruges til at gemme ændret data og udføre struktureret logning, og View's og Stored Procedures kan bruges til at sikre databasen, samt øge dens performance mht. Join operationer.

Hvis vi ser på fremtidige implementeringer, så ville et lagersystem være nærliggende at få inkl. i projektet. Pt. udregner vi fx den totale længde på det træ som bruges til carporten. Men det jo ikke sikkert at disse stykker træ kommer i de længder vi udregner og som ligger på lager. Derfor vil et integreret lagersystem være en fordel at få implementeret, da sælgerne på den måde kan se lagerstatus og bestille ny varer i henhold til status. Med et lagersystem kan Fog også leve op til CSR[13], da vi nu kan udregne spild på baggrund af fx de længder som findes på lager. En anden implementering kunne være at give sælgerne mulighed for manuelt at ændre i styklisterne og antal. Ved at give sælgerne større mulighed for fleksibilitet, kan sælgerne matche kundernes behov bedre i specielle situationer, hvor systemets grundudregning ikke virker. Ulempen ved dette er dog, at sælgernes brug af systemet skal dokumenteres vha. struktureret logning, som Fog skal bruge til deres audit mht. at finde fejl, mangler og evt. snyd.

[13] https://da.wikipedia.org/wiki/Corporate_Social_Responsibility

En af de ting vi virkelig prøvede at efterleve under dette projekt, var brugen af Scrum. Vi var rigtige gode til at uddelegere opgaverne, samt holde flere daglige virtuelle møder mht. status. Dette gjorde vi da vores risikoanalyse viste os, at COVID-19 situationen kunne have betydning for vores projekt. Det var vigtigt for os at være ekstra opdateret på hinandens projektudvikling, så vi til enhver tid kunne overtage hinandens opgaver. I denne proces erfarede vi også, at vi under alle omstændigheder tit kom til at arbejde flere på enkelte opgaver. Dette skyldes, at de enkelte elementer der blev udviklet individuelt, skulle integreres i systemet. Tit skulle vi gennemgå variabler og argumenter, så vi fx ikke fik ”Nullpointerexception” når to udviklere skulle sammensætte deres kode. Dette er nok også grunden til at vores velocity blev så høj, de vi tit havde flere øjne på samme opgave og ikke nødvendigvis arbejde som selvstændige ”ø’er”.

Til slut vil vi gerne takke for en god vejledning i løbet af projektet, som var med til at løfte projektets niveau.
Det var rart at have tekniske reviews med en senior developer, da vi havde tendens til at fastlåse os selv i vores egne løsninger.
Skulle vi ønske os noget mere, så ville det være en mere teknisk gennemgang af en carport mht. bæreevne, materiale, muligheder og design.
Men på trods af dette, så synes vi selv, at vi har leveret det flot resultat, som vi håber i har nydt at se på og læse om.