

Semesterprojekt - Fog

Deltagere (Navn – Mail – Github):

Alle deltagere er fra hold b.

- Kristofer Pedersen – cph-kp278@cphbusiness.dk – "Krisdanoda"
- Michael Frederiksen – cph-mf315@cphbusiness.dk – "Magical Glove"
- Sofia Mathilde – cph-sm436@cphbusiness.dk – "sofiadoingcode"
- Søren Merved – cph-sm428@cphbusiness.dk – "SMerved"

Projektet blev udarbejdet fra d. 02/05-2022 til d. 31/05-2022.

Indholdsfortegnelse

Semesterprojekt - Fog.....	0
Links	2
Indledning	3
Baggrund	3
Virksomheden/Forretningsforståelse	4
Teknologi valg	6
Krav	7
Arbejdsgange der skal IT-støttes.....	9
User-stories.....	11
Domæne model	14
EER-diagram	16
Navigationsdiagram	18
Mockups.....	20
Valg af arkitektur.....	20
Særlige forhold.....	21
Udvalgte kodeeksempler	23
Status på implementering.....	25
Test.....	26
Arbejdsprocessen faktuel	27
Arbejdsprocessen reflekteret	29
Bilag.....	31
Logbog.....	31

Links

- GitHub repository: <https://github.com/Sofiadoingcode/ProjektSemester.git>
- Demo video: <https://youtu.be/VSdwAcnGEWg>
- Pitch video: <https://youtu.be/hgK0moXXQdI>
- Link til deployet version: <http://165.227.155.35:8080/>
 - Login navn: admin
 - Kodeord: unek34unejkSKDqwe

Indledning

Til dette projekt har vi fået en opgave at bygge et system for JohannesFog, der kunne håndtere bestillinger af carporte og det tilhørende lager. Meningen med projektet er vi lader som om, at vi er en virksomhed, som vil prøve at sælge og implementere et nyt system for Fog. Lageret håndteres i en mySql database. Useren skal få adgang til programmet via en hjemmeside. Den er bygget med Java Servlet og jsp-sider til front-end. Rapporten er skrevet ud fra, at læseren har en lignende forståelse for emnet som os, dvs. en datamatiker studerende på 2. semester.

Baggrund

JohannesFog er en veletableret virksomhed med ekspertise indenfor byggematerialer og vejledning til byggeprojekter. De har 9 butikker fordelt i Nordsjælland og har i mange år haft en solid onlinehandel. Virksomheden er dog vokset hurtigt de sidste par årtier, og deres IT-systemer har derfor ikke kunne følge med.

I systemet skal man kunne logge ind med en administrator-konto eller en midlertidig kundekonto. Kunden skal i systemet kunne bestille en carport, hvor de kan vælge højde, bredde og længde på carporten, samt diverse tagspecifikationer. Admin skal kunne se en liste over samtlige bestillinger i systemet og ændrer på deres tilstand, f.eks. acceptere eller fjerne bestillinger fra systemet. Når en bestilling er blevet accepteret af admin, skal kunden kunne logge ind med et brugernavn og kodeord, som de er blevet tildelt og betale for deres bestilling. Admin skal have adgang til en liste over alle produkter på lageret. Her skal admin kunne ændre eller fjerne de enkelte produkter, samt tilføje nye produkter i systemet. Derudover skal admin kunne lave nye admin brugere og fjerne eksisterende brugere.

Virksomheden/Forretningsforståelse

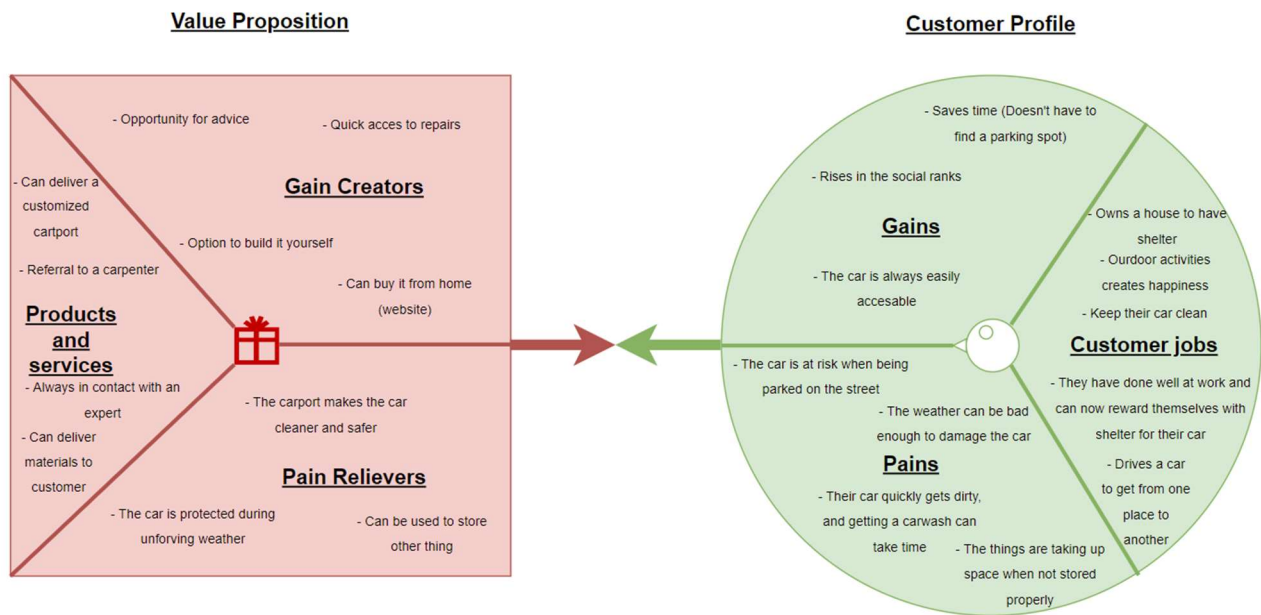
SWOT:

Strengths	Weaknesses
<ul style="list-style-type: none">• Good or strong relationship with customers• Large business with products in different markets.• Is an old and well established business.	<ul style="list-style-type: none">• Service runs on old systems. Systems lacks compatibility with each other.• Has only shops in north Zealand.
Opportunities	Threats
<ul style="list-style-type: none">• Johannes fog has seen increase sales through online shopping.	<ul style="list-style-type: none">• With the increase of the internet and automation, customers are less dependent on customer-employee relationship.

Figur 1: SWOT Diagram til Johansfog carport foratning

Fog's it-systemer er forældede og er bygget op på en måde der begrænser admins muligheder for at opdatere systemet. Det er derfor oplagt for os at bygge et nyt system fra bunden, der er mere fleksibelt og kan holdes ajour med lageret. Dette kan forhåbentlig også forbedre kompatibilitet med Fog's andre systemer og gøre admins arbejde mere effektivt. Da mere og mere handel foregår på nettet er det essentielt at have et velfungerende it-system, der kan gøre det så nemt som muligt for kunden så vel som administrator. Vi er dog opmærksomme på, hvor højt Fog vurderer det personlige forhold til kunden. Netop dette forhold mellem kunden og den individuelle medarbejder vil derfor stadig være en vigtig del af bestillingsprocessen, selvom den foregår på et mere effektivt system online.

Value Proposition Canvas:



Figur 2: VPC

Refleksion:

Vi startede med at stille os i kundens sted, og overvejede hvad de kunne få gavn af fra Fogs ydelser. Vi fandt frem til de forhold som kundens dagligdag skulle bestå af, og vi tænkte derefter over hvilke mangler en potentiel kunde kunne have, som ville udløse en nødvendighed til en ny carport. Efter dette overvejede vi, hvordan en carport ville kunne gavne kunden. Dette satte vi i perspektiv til Fogs produkter og ydelser og fandt derfra ud af, hvordan Fog kunne hjælpe kunden. Dette ledte også til ydelser som kunne gavne kunden mere end det de originalt havde forventet.

Teknologi valg

Til udviklingen af vores system, har vi brugt følgende teknologier:

- IntelliJ – 2021.2.2
- JDBC 4.3
- MySQL – 8.0.23
- Java – Version 17
- CSS – 4.15
- HTML – 5
- Bootstrap – 5
- TomCat – 9.0.60
- Maven – 3.6.3
- JUNIT – 5.8.2
- Digital Ocean

Krav

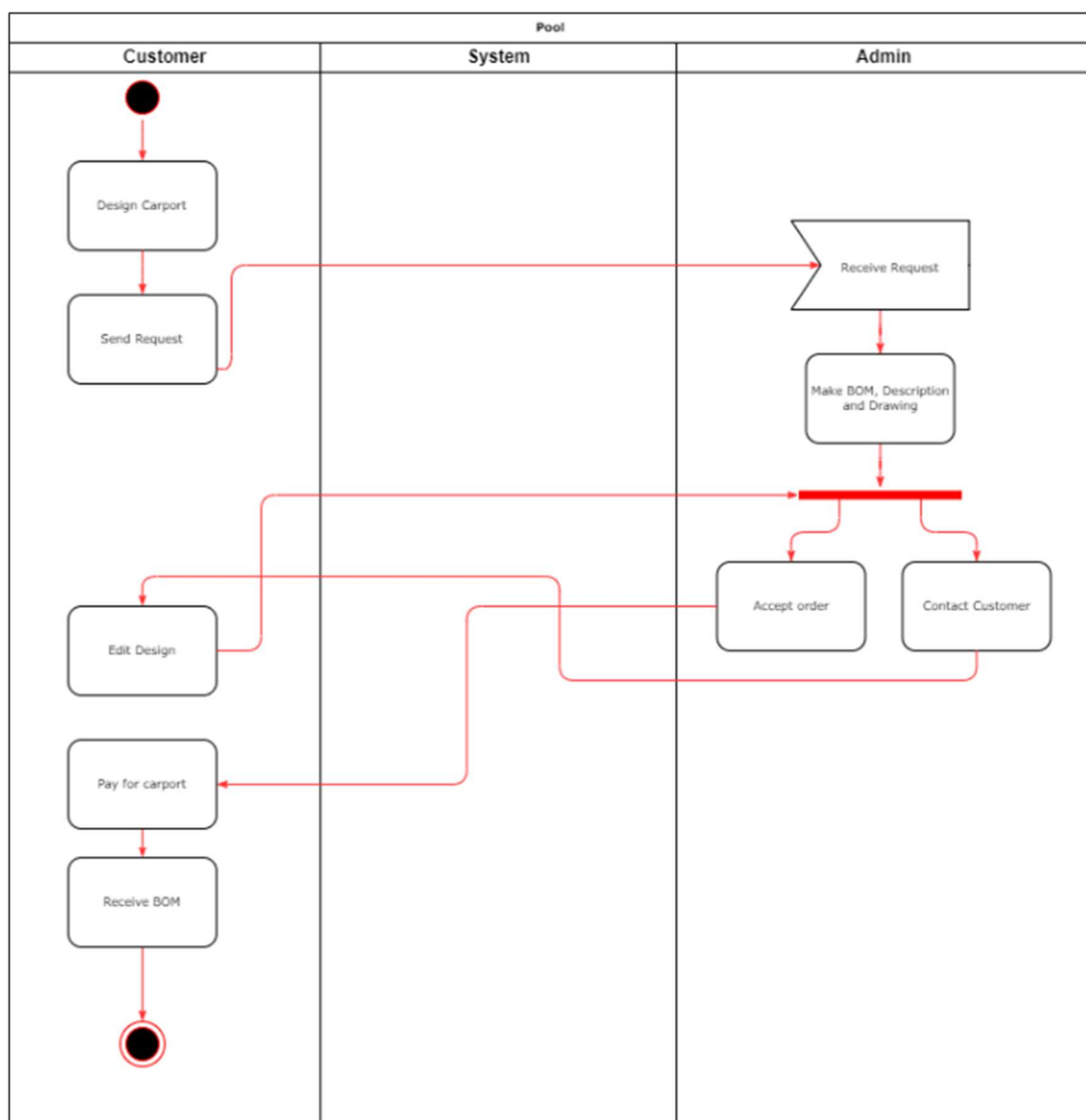
Afsnittet er sat op således:

- User Stories
 - Acceptkriterier
1. Som administrator skal jeg kunne logge ind og se forespørgsler, samt tidligere ordrer.
 - Hvis man ikke er logget ind og man har en administrator-konto i databasen, skal man kunne logge ind og se forespørgslerne, så længe man taster rigtigt på log in siden.
 2. Som kunde ønsker jeg at kunne gå til en side for at kunne indtaste specifikationerne min carport samt kontaktinformation, hvilket jeg herefter kan sende som en forespørgsel.
 - Hvis man ikke er logget ind, skal man kunne gå til en side, hvor målene, specifikationerne og kontaktinformationerne kan indtastes og herefter blive sendt.
 3. Som administrator skal jeg kunne se alle forespørgsler, og deres tilsvarende styklister.
 - Hvis man er logget ind som administrator, kan man i navigationsbaren trykke på "Anmodninger" og blive ført til en side, hvor man kan se alle forespørgsler og deres styklister.
 4. Programmet skal kunne generere en stykliste ud fra målene, så jeg (admin) kan se den og redigere prisen.
 - Når en forespørgsel bliver sendt, genererer programmet en stykliste ud fra de mål som er blevet indtastet, hvorefter en admin, via "Anmodninger" i navigationsbaren, kan se den og redigere prisen.
 5. Som kunde skal jeg kunne se min forespørgsels tilhørende tegning og beskrivelse.
 - Hvis man er logget ind på sin konto, kan man gå til en side, hvor tegningen og beskrivelsen til ens carport bliver vist.

6. Som admin skal jeg kunne acceptere en carport og sende en form for accept til kunden, samt være i stand til at kunne annullere en accept af en carport.
 - Hvis man er logget ind som admin, kan man via "Anmodninger" i navigationsbaren, acceptere en specifik forespørgsel og fortryde en accept af en forespørgsel.
7. Som kunde skal jeg kunne se at min forespørgsel er godkendt og herefter kunne betale.
 - Hvis man er logget ind som kunde, kan man via knappen "Din carport" i navigationsbaren, se en status på hvor langt ens anmodning er, altså om den afventer accept, afventer betaling eller om styklisten er til rådighed.
8. Som kunde, givet at jeg har betalt, skal jeg kunne se styklisten til min carport.
 - Hvis man er logget ind som kunde og har lavet en bestilling, som er blevet accepteret af en administrator og betalt, vil en knap fremgå i bunden af "Din carport", som vil føre kunden til sin stykliste.
9. Som administrator skal jeg kunne ændre priser på varer i databasen og tilføje nye elementer.
 - Hvis man er logget ind som admin, skal man via navigationsbaren kunne gå til en side, hvor man kan tilføje nye og redigere nuværende elementer i databasen.
10. Som administrator skal jeg kunne lave flere admin-konti og slette den konto jeg er logget ind som.
 - Hvis man er logget ind som admin, kan man via knappen "Konti" i navigationsbaren, indtaste et brugernavn og et kodeord, og så vælge om man vil slette den konto fra databasen eller oprette den.

Arbejdsgange der skal IT-støttes

Nedenfor ses et AS-IS diagram for Fog's nuværende bestillings proces. Kunden designer en carport og skriver sin personlig information, som laver forespørgslen. Forespørgslen bliver sendt til en administrator fra Fog som modtager den på e-mail. Her har en Fog ansat muligheden for at kontakte kunden om deres carport, kigge på forespørgslen, indsætte informationen ind i et program, hvor den så laver en stykliste og tegning. Den ansatte har så muligheden for at ringe til kunden om deres carport. Hvis alt er godt og godkendt kan kunden betale for carporten, hvorefter de modtager byggevejledning med styklisten.

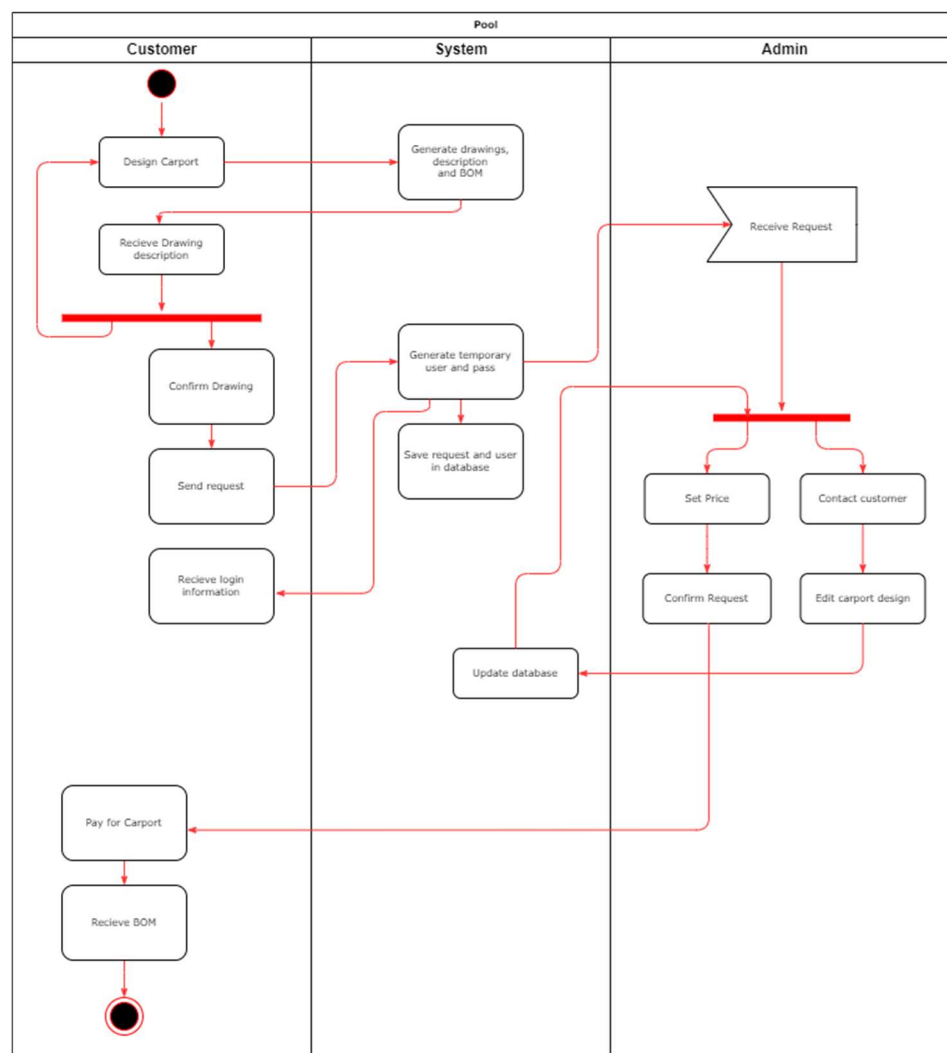


Figur 3 As-is Aktivitets Diagram

Nedenfor ses TO-BE diagrammet som beskriver forløbet af bestilling på vores hjemmeside. Kunden starter med at besvare formen, ved at skrive dimensionerne til deres carport og deres personlige information. Hvis de har lyst, kan de se tegningen til den carport, som de har designet ved at trykke på ”se tegning”, hvorefter programmet genererer både tegningen og styklisten, men kun viser tegningen på hjemmesiden. Hvis kunden er tilfreds, kan de godkende forespørgslen. Carporten og kundens personlig information bliver så gemt i databasen, og de får tildelt en midlertidig bruger. Kunden kan så logge ind og se deres carport og status på deres forespørgsel.

En administrator til systemet kan logge ind og se en liste af alle anmodninger. De kan så kigge på kundens carport, sætte prisen til den og godkende den.

Herefter kan kunden logge ind på deres konto for at se deres carport, og om den kan betales. Når kunden har betalt, modtager de styklisten.



Figur 4 to-be Aktivitets diagram

User-stories

User story 1

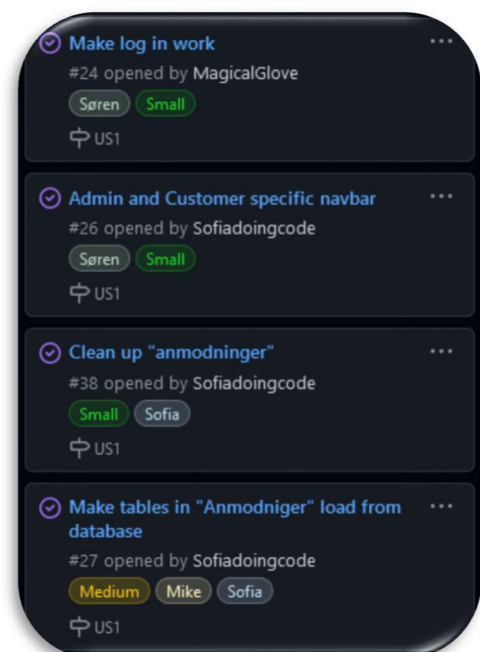
Som admin skal jeg kunne logge ind og se forespørgsler, samt tidligere ordrer.

1. How to demo

- Tryk på "Log Ind" i navigationsbaren
- Udfyld felterne "Brugernavn" og "Kodeord" med et log ind, som tilhører en admin konto (Eksempel: brugernavn: "fog" - kodeord: "123")
- Tryk på "Log ind"
- Tryk på "Anmodninger" i navigationsbaren
 - Tryk eventuelt på "Betalte Forespørgsler" for at se forespørgsler som er gennemført

2. Tasks

User story 1 fik i alt tildelt fire issues, men det er i virkeligheden kun det sidste, som udgør essensen af denne user story. Det sidste issue handler nemlig om at modtage data fra databasen, mens de andre tre issues, enten var oprydning, eller ting der lå forud for at fuldende denne user story.



Figur 5 Userstory 1 tasks

3. Accept-kriterier

- Hvis man ikke er logget ind og man har en admin konto i databasen, skal man kunne logge ind og se forespørgsler, så længe man taster rigtigt på log in siden.

User story 2

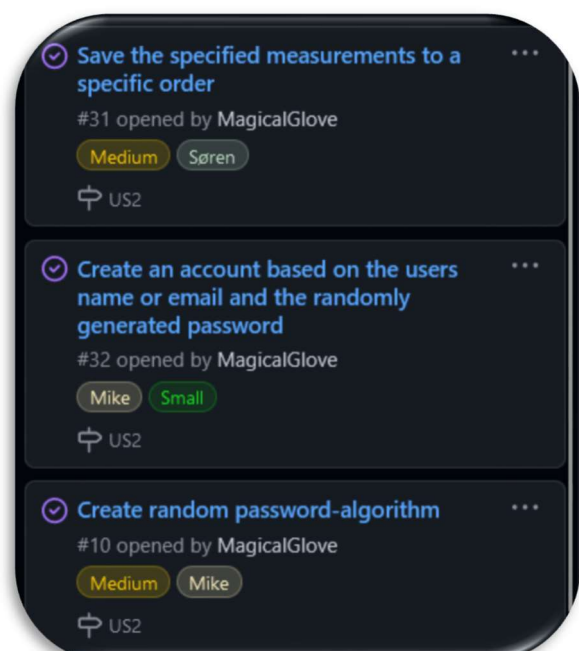
Som kunde ønsker jeg at kunne gå til en side for at kunne indtaste specifikationerne min carport samt kontaktinformation, hvilket jeg herefter kan sende som en forespørgsel.

1. How-to-demo

- Tryk på "Bestil Carport" i navigationsbaren
- Rul ned til afsnittet "Design din egen carport"
- Udfyld de nødvendige felter (hvis et nødvendigt felt ikke er udfyldt, bliver man stoppet, når man prøver at gå videre)
- Tryk på knappen i bunden kaldet "Send Forespørgsel"

2. Tasks

User story 2 blev delt op i tre task. Den ene del går ud på, at de mål som bliver skrevet ind på siden, bliver gemt på en specifik bestilling. Denne bestilling skal så knyttes til en konto, hvilket er det de to næste issues går ud på. En af dem værende genereringen af kontoen, og den anden (selv algoritmen) som generer den tilfældige kode, der bliver tildelt brugerens konto.



Figur 6 User Story 2 tasks

3. Accept-kriterier

- Hvis man ikke er logget ind, skal man kunne gå til en side, hvor målene, specifikationerne og kontaktinformationerne kan indtastes og herefter blive sendt.

User story 10

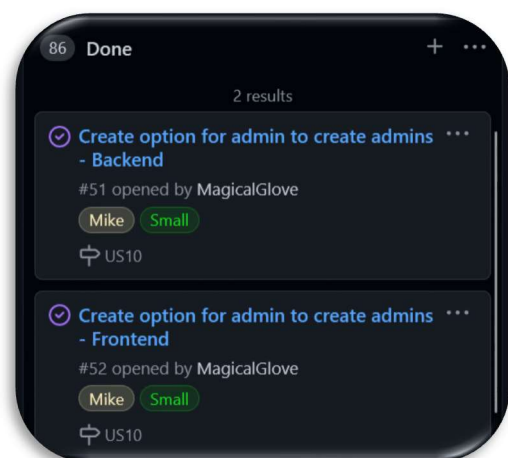
Som administrator skal jeg kunne lave flere admin-konti og slette den konto jeg er logget ind som.

1. How to demo

- Tryk på "Log Ind" i navigationsbaren
- Udfyld felterne "Brugernavn" og "Kodeord" med et log ind som tilhører en admin-konto (Eksempel: brugernavn: "fog" - kodeord: "123")
- Tryk på "Log ind"
- Tryk på "Konti" i navigationsbaren
- Indtast et brugernavn og et kodeord på en konto, som findes i databasen
 - Hvis det er en konto som findes og man ønsker den slettet, tryk "Slet konto"
 - Hvis det er en konto man ønsker at lave, tryk "Lav konto"

2. Tasks

Denne user story er delt op i to tasks. Den ene task gik ud på at lave jsp-delen af siden, mens den anden task handlede om at lave koden bag det. Disse to tasks tilsammen, er hvad der udgør admin-siden "Konti" på hjemmesiden.



Figur 7 User Story 10 Tasks

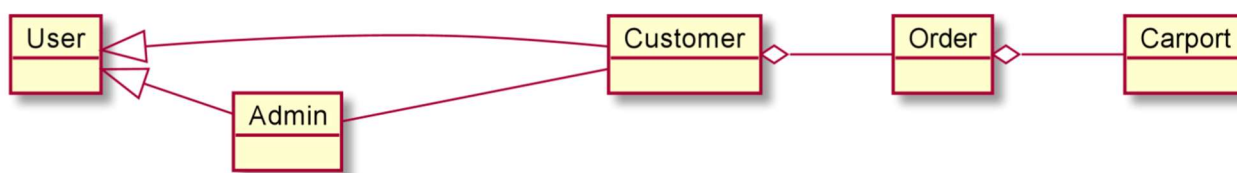
3. Accept-kriterier

- Hvis man er logget ind som admin, kan man via knappen "Konti" i navigationsbaren, indtaste et brugernavn og et kodeord, og så vælge om man vil slette eller tilføje den konto til/fra databasen.

Domæne model

Domænemodellen (sammen med EER Diagrammet) ligger som grundlag for hele vores system. Den skal vise den grundlæggende data- og systemopbygning med de vigtigste konceptuelle klasser, som vi har tænkt det inden og under konstruktionen af systemet.

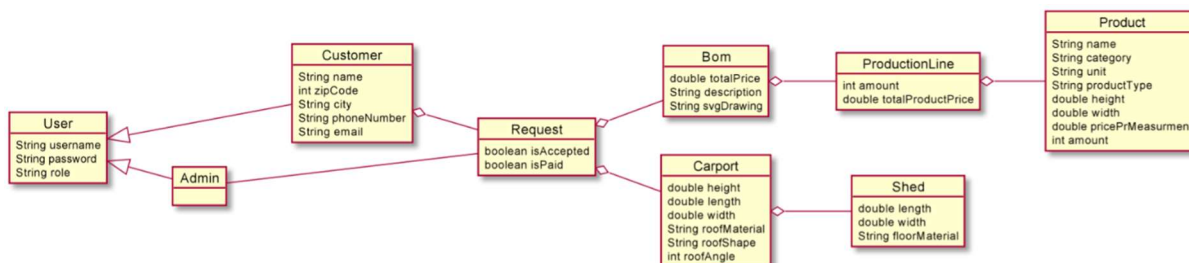
Her ses den **første version** af vores Domænemodel. Denne version er lavet på dag 2 af hele projektprocessen (Vi er ikke begyndt at kode):



Figur 8 Domænemodel. Første Version.

Vi har lavet en simpel domænemodel med klasser, der er lavet ud fra de vigtigste navneord, der skal fremgå i systemet. Vi mener at både kunden (Customer) og administratoren (Admin), er en slags bruger (User), hvilket er grunden til, at vi har sat dem til at have en "generalization" relation. Dog skal administratoren kunne se alle kunderne, hvilket derfor giver "Admin" og "Customer" en "association" relation. Vi mener herefter, at ordren/forespørgslen (Order) er en del af en kunde (Customer), grundet at den enkelte ordre ikke kan findes, uden den kunde der har bestilt den. Ligeledes er en carport (Carport) en del af en ordre (Order) på samme grundlag. Dette giver begge disse relationer en "aggregation" relation.

Her ses den **sidste version** af vores Domænemodel. Denne version lavede vi på dag 23 af hele projektprocessen (Vi er færdige med at kode):



Figur 9 Domænemodel. Opdateret version.

Med tiden er domænemodellen vokset i takt med systemets udvikling. De vigtigste attributter er sat på hver klasse, og relationerne er blevet opdateret. Det ses, at det der tidligere hed "Order" er blevet ændret til "Request", hvilket betyder "forespørgsel". Dette ændrer vi, da vi

mener at "en ordre" refererer til en bestilling, som ikke kan ændres. Altså kunden har lavet en ordre, og forventer derfor præcis det de bestilte. I vores system skal en administrator/en medarbejder hos Fog kunne se, foreslå ændringer og godkende kundens bestilling, før at kunden kan få mulighed for at vælge og betale for den. Derfor giver det mere mening at kalde det en forespørgsel i stedet for en ordre.

Det er også blevet ændret til, at administratoren (Admin) ikke længere kan se en kunde (Customer), men en forespørgsel (Request). Dette giver langt bedre mening, da det er vigtigere for administratoren at se, hvad kunden har forespurgt og alt der hører til, i stedet for hvem kunden er.

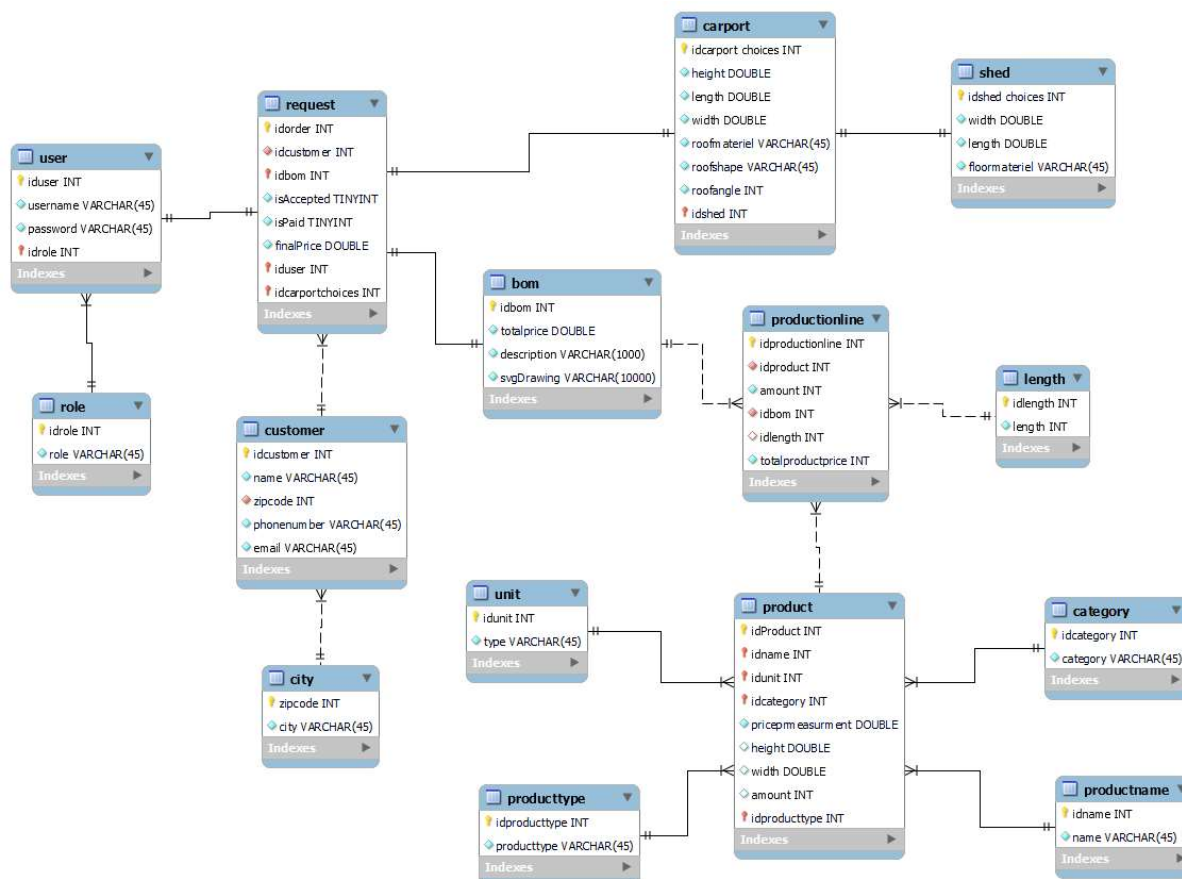
Udover disse ændringer er der blevet tilføjet en hel række klasser: "Bom" (Carportens stykliste), "Productionline" (Den enkelte linje på styklisten) og "Product" (Det enkelte produkt i Fogs varehus). Vi mener, at et produkt (Product) er en del af en produktlinje (ProductionLine), ligeledes er en produktlinje (ProductionLine) en del af en stykliste (Bom), hvilket gør relationerne mellem disse klasser til "aggregation" relationer. På samme måde er en stykliste (Bom), en del af en forespørgsel (Request), så dette er også en "aggregation" relation.

Til sidst nævnes at Carporten (Carport) har fået en "aggregation" relation til en ny klasse "Shed" (Carportens skur), da et skur er en del af en carport.

EER-diagram

EER-diagrammet viser de forskellige tabeller der opstår i systemets database og relationerne mellem dem.

Her ses vores endelige EER-diagram:



Figur 10 EER Diagram fra MYSQL Workbench

Det ses, at de samme klasser som i domænemodellen går igen som tabeller i EER-diagrammet, sammen med en række nye som udvider de originale klasser.

Vores EER-diagram følger 3. normalform på alle tabeller undtagen to: "Carport" og "Shed". I "Shed" er der en kolonne ved navn "floorMaterial" (gulvmateriale) og ligeledes i "Carport" er der en kolonne ved navn "roofMaterial" (tagmateriale) og en ved navn "roofShape" (tagform). Disse kolonner opfylder ikke 3. normalform, da flere kunder kan have valgt den samme tagform, tagmateriale og/eller gulvmateriale. Derfor opstår de samme værdier i flere af tabellens rækker, hvilket betyder, at det ville være smartere at sætte fremmednøgler til tabeller, der indeholder værdierne. Vi har med vilje valgt ikke at gøre dette, da vores program kun understøtter at en enkel værdi kan være i de kolonner, der snakkes om. F.eks. kan man

kun vælge at have et fladt tag og ikke andre former. Derfor var det ikke relevant i vores projekt at sætte dem på 3. normalform. Udvider man programmet senere hen, ville det være mere nødvendigt at sætte disse kolonner ud på deres egne tabeller.

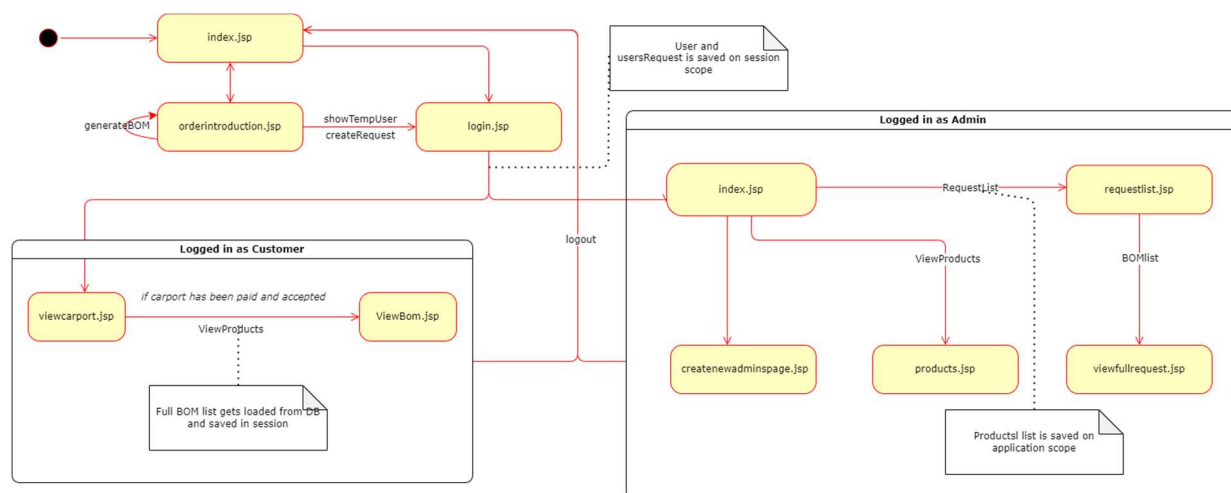
Kigger vi på relationerne, ses det at igennem meget af vores EER-diagram er der benyttet én til mange relationer. F.eks. kan mange forskellige produktionslinjer (productionline) have en fremmednøgle til det samme produkt (product), eller en by (city) kan bebos af mange forskellige kunder (customer). Vi benytter os dog også af én til én relationer en del især i forbindelse med "request" tabellen. Vi har i den forbindelse ikke valgt at sætte "user", "bom" og "carport"+"shed", som en del af "request" tabellen. Det har vi valgt ikke at gøre af følgende årsager:

1. Tabellen "request" ville blive alt for lang og uoverskuelig
2. Det giver logisk mere mening at lave tabeller for de forskellige entiteter, der opstår fra den ægte verden.
3. Vi ønsker i vores system at kunne tilgå de enkelte tabeller, som de er, med de kolonner som hører til, uden at skulle filtrere igennem en masse andre kolonner.
4. Vi vil i vores system gerne kunne gemme til de enkelte tabeller uden de er afhængige af en forespørgsel (request).

Den eneste tabel der ikke bruger automatisk genereret ID som nøgle er "city". Her bruges postnummeret (zipcode) på den enkelte by. Det gør vi, da der ikke kan være to byer med det samme postnummer, og det giver derfor mere mening at benytte sig af det.

"Product"-tabellen kan forstås som varehuset, altså de materialer Fog har til at bygge en carport. "Unit" beskriver hvilken slags enhed, som produktet angives i, f.eks. om det er en pakke eller et sæt. "Category" bestemmer hvilken af de to kategorier den tilhører, "Træ og Tagplader" eller "Beslag og Skruer". "Producttype" er en variabel, som vi har tilføjet, som afgør hvilken slags produkt det er, f.eks. en skrue eller en stolpe. Det gjorde det nemmere for os at kode styklisteberegneren. Vi har valgt at lægge "Length" på "ProductLine", da det ikke var klart hvilke produkter, som havde hvilke længder, og vi gik derfor ud fra at alle produkter kunne have alle længder.

Navigationsdiagram

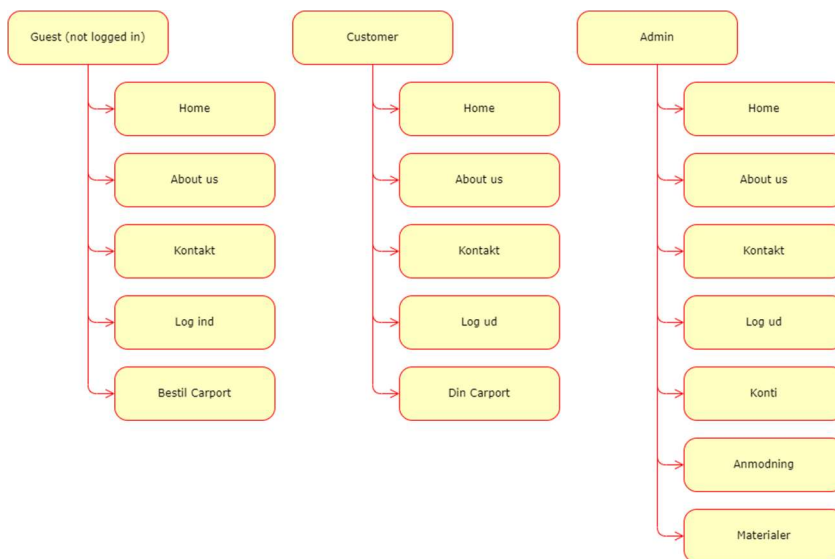


Figur 11 Navigations Diagram. Teksten på pilene er de forskellige servlets som redirecter siderne

Navigationsdiagrammet viser det generelle navigationsflow for hjemmesiden. Index-siden fungerer som forsiden til hjemmesiden. En bruger som ikke er logget ind, kalder vi en "guest". En guest kan trykke på "Om os" og "Kontakt os". På index-siden og i navigationsbaren, kan en guest-bruger komme til carport-design-siden, hvor de kan designe og bestille en carport. De skal udfylde formen om deres carport design og deres personlige information, før de kan gå videre med bestillingen. Herefter kan kunden logge ind med deres midlertidige bruger og se deres carport-beskrivelse, design og tegning. Styklisten kan kun ses af kunden, efter carport-forespørgslen er blevet accepteret af en administrator, og betalt af kunden.

En bruger kan også logge ind som en administrator, hvilke giver dem flere forskellige funktioner. Deres primære funktion er på anmodningssiden. Her kan en administrator se alle kunders carport-anmodninger, sætte prisen på dem og acceptere dem, således den efterfølgende kan blive betalt af kunden.

Nedenfor ses navigationsdiagrammet til navigationsbaren.

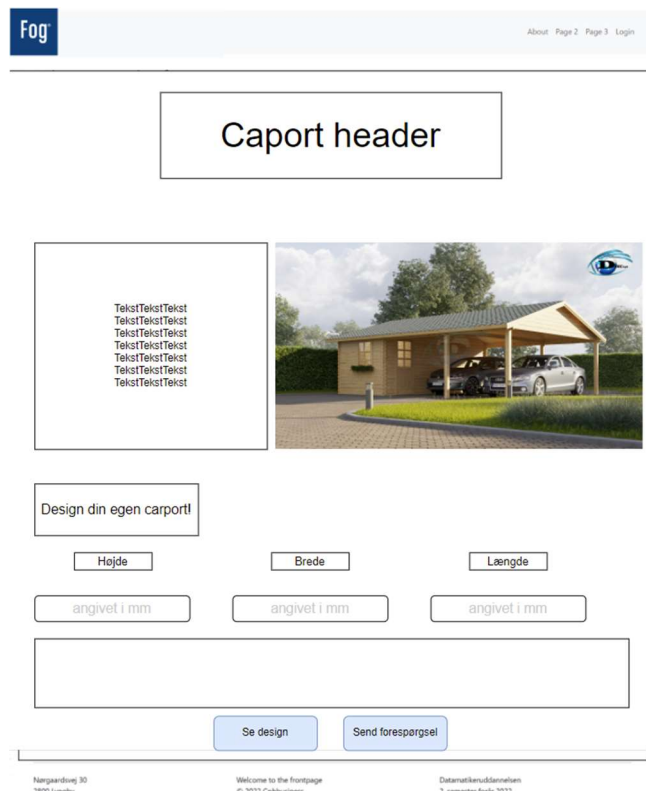


Figur 12 Forenklet navigations Diagram for Navigationsbaren

På hver side er der vist en navigationsbar, som ligger øverst på siden. De muligheder som er fælles for alle brugere, er Fog-logoet (som fører til forsiden), “Kontakt Os”, “Om Os” og “Log Ind”. “Log Ind” bliver erstattet med “Log Ud”, når en bruger logger ind. En guest har et link til “Bestil Carport”, mens en kunde har et link til deres egen carport (“Din Carport”). I navigationsbaren har en administrator også “Konti”, “Anmodninger” og “Materialer”. Konti-siden giver administratoren mulighed for at lave nye- og slette admin-konti. ”Materialer” er, hvor de kan kontrollere og tilføje nye produkter til databasen. ”Anmodninger” viser en liste over alle kundernes forespørgsler i databasen.

Mockups

Vi startede projektet ud med at lave mockups til de forskellige sider på hjemmesiden. Det gjorde vi for at danne et overblik over, hvordan siderne skulle se ud, og hvilke ting vi fandt nødvendige, så vi kunne skabe den bedst mulige oplevelse for kunderne. Billedet til højre er vores mockup over den side, hvor man som kunde bestiller en carport. Vi tænkte at denne side ville være god af flere årsager. Den ville nemlig starte ud med at have et tekstafsnit og et billede, som ville give brugeren en tryk følelse og klarhed omkring, hvad de ville være i gang med at bestille. Herefter ville brugeren få de felter, hvor de ville skulle indtaste de mål, som de ønskede, at deres carport skulle have. Disse felter ville have overskrifter og enheder angivet, så brugeren ikke kunne blive i tvivl. Idéen var så, at der efter man havde trykket på ”Send forespørgsel” ville dukke et popup-vindue frem, hvor man kunne se kontooplysningerne til ens autogenereret konto, samt et lille afsnits tekst, som ville informere brugeren om, hvad det næste skridt i deres bestilling vil være.



Figur 13 Mockup for carport bestilling side



Valg af arkitektur

Vi har valgt at lave vores projekt med brugen af Front Controller. Det vil sige at vi benytter en Command-klasse, som er bindeleddet mellem vores kode og jsp-sider. Når vi ønsker at en knap på vores hjemmeside skal transportere data, uanset om en bruger forsøger at logge ind, lave en konto, genere en stykliste eller andet, så bliver det sendt gennem vores Command-klasse, og videre til den relevante klasse i sammenhængen.

Vi valgte en struktur, opbygget af Front Controller, da det var nyt for os, og vi tænkte at det ville give os mulighed for at lære en ny måde at få jsp-sider og klasser til at spille sammen på.

Særlige forhold

I vores program gemmer vi "User" på session i forbindelse med login, og hvis det er en kunde der logger ind med en midlertidig konto, bliver deres tilhørende "Request" og "Carport/Shed Choices" også gemt på session. Derudover gemmer vi to lister af "ProductLines" der tilsammen dækker alle produkter i den gældende stykliste.

Håndtering af exceptions er ikke noget vi i denne opgave har valgt at bruge meget tid på. Vi catcher dem, hvor det er nødvendigt og har i det meste af programmet en out print, der kan hjælpe med at identificere, hvor problemet er.

Vi har lavet input validering på de fleste form-elementer i vores program, i form af at brugeren ikke kan efterlade felter tomme, der skal udfyldes, og der kommer en lille besked til brugeren om, hvad det er de mangler at udfylde. Derudover er der nogle begrænsninger på de enkelte former, som f.eks. et telefonnummer skal have otte tal mellem 0 og 9 og email skal indeholde et @. Brugeren kan heller ikke indtaste specifikationer på carporten, som programmet/Fog ikke kan håndtere, f.eks. en carport med en længde på 20 meter.

Vi har i vores program valgt ikke at gå særligt meget op i sikkerhed, når det kommer til at logge ind på hjemmesiden. Kodeord bliver for eksempel ikke krypteret eller lignende. Vi følte nemlig ikke at det var en del af pensum, og ville derfor hellere gå op i andre aspekter af koden, som vi fandt mere lærerige og aktuelle. Kunders kodeord er dog en tilfældigt genereret kode på 6 cifre, med mulighed for både store bogstaver, små bogstaver og tal, hvilket gør at koderne er relativt sikre. Ved dannelse af en admin-konto er der dog ingen krav eller guidelines og en admin-kontos kodeord kan derfor være alt fra "123" til en 45-cifre lang kode.

Vores program genererer en svg-tegning ud fra de mål brugeren har angivet i deres forespørgsel. Denne tegning viser carporten set oppefra, og angiver længde og bredde, samt afstand mellem spærrene.

Vores database indeholder kun fire postnumre, hvilket skyldes at vi ikke har fundet det relevant at load alle postnumre i Danmark. Hvis dette program skulle implementeres i virkeligheden, skulle man load alle postnumre, men da dette kun er en prototype, har vi ikke prioriteret det, og brugere skal derfor holde sig til de postnumre som findes (F.eks.: 3400).

Udvalgte kodeeksempler

Nedenfor ses metoden ”generateBOMProductLines” som er en metode i klassen ”BOMAlgorithm”.

```
private List<ProductLine> generateBOMProductLines(CarportChoices carportChoice) {

    List<ProductLine> fullBom = new ArrayList<>();

    List<String> carportNeededItems = returnNeededListCarport();

    List<String> shedNeededItems = returnNeededListShed();

    for (int i = 0; i < carportNeededItems.size() + shedNeededItems.size(); i++) {

        String neededItem = carportNeededItems.get(i);

        if (i > carportNeededItems.size() - 1) {

            neededItem = shedNeededItems.get(i);
        }

        List<ProductLine> itemProductLines = generateItemProductlines(neededItem, carportChoice);

        for (ProductLine pl : itemProductLines) {
            fullBom.add(pl);
        }
    }
    return fullBom;
}
```

I BomAlgorithm klassen ligger en del metoder, der til sammen generer carportstyklisen (BOM), tegningen og beskrivelsen ud fra de mål, som kunden skriver ind.

Vi valgte at designe koden sådan, at man i andre dele af programmet kun behøver at kalde til én metode fra denne klasse for at få fat i hele styklisen. Denne metode kalder derfor en del

andre private metoder, som udfører de enkelte udregninger og hentninger fra databasen. Dette gør at koden er nemmere at læse og følge.

Denne metode returnerer en Arrayliste af entiteten ProductLine, som repræsenterer en enkelt linje af den enkelte stykliste. Så denne metode viser altså hele opbygningen af algoritmen.

Vi valgte at opbygge algoritmen således, at den looper igennem en liste af alle de dele, som en carport består af (carportNeededItems). Den enkelte del bliver gemt i en variabel

(neededItem), som så bliver sendt som parameter i en metode ved navn

”generateItemProductlines”. I denne metode er der en switch/case, som tager ”neededItem”

ind, og kører så den udregning, der passer til den specifikke del af carporten. Fra den metode

returneres så en liste af alle de “ProductLine’s”, som har med den del af carporten at gøre. Alt

i den liste bliver så lagt til den liste, der bliver returneret fra metoden, altså den liste der

indeholder hele styklisten. Den liste repræsenterer så alle de produkter (+ mængden og

prisen), som skal bruges, for at kunden kan bygge sin carport.

Status på implementering

Vores program opfylder kundens krav med undtagelse af, at admin ikke kan se tegningen til carporten. Derudover mangler der en jsp-side til “Kontakt os” knappen i navigationsbaren, og indhold til jsp-siden “Om os” gennem knappen i navigationsbaren.

Vi har lavet alle nødvendige CRUD-metoder til systemet, men vi har ikke fået lavet samtlige til alle tabellerne, specielt update mangler på mange af tabellerne, da de ikke var essentielle for programmet.

Vi havde fokus på jsp-sider og styling i et par dage i begyndelsen af projektet, men derefter blev især styling lidt en eftertanke, derfor mangler der noget styling nogle af de jsp-sider, vi har lavet.

Vi fik aldrig fuldt implementeret, at man kan bestille en carport med tilhørende skur, selvom det fremgår på bestillingssiden. For at implementere det, mangler vi at tilføje det til svg-tegningen af carporten og gøre det til en del af algoritmen, der laver styklisten.

Det ses på To-Be aktivitets diagram (figur 4), at der skulle være mulighed for, at administratoren kan ændre på carporten. Meningen var, at en administrator kan kigge på carportens design, og hvis de mener, at der er et problem med den, så kan de ringe til kunden og hører om et re-design. Administratoren vil kunne omdøbe designet hvor databasen vil blevet omdøbte. Denne feature er ikke implementeret endnu i vores program.

Test

Integrationstest er lavet ved brug af JUnit test. Da der ikke er lavet CRUD funktioner til alle mappers, er det ikke alle mappers, som er blevet testede. RequestAndBOMMapperTest simulerer den proces, som laver en forespørgsel og gemmer en ordre.

Algoritmen som generer styklisten, var meget begrænset, når det kom til, hvad man kunne teste med JUnit. Det var fordi algoritmen varierede meget med indholdet af databasen.

Algoritmen kigger i "product"-tabellen og søger efter hvad den synes er optimalt som materiale. For at kunne komme uden om dette problem, har vi lavet en main metode, hvor vi kan kalde BOMAlgorithm. Ved at teste med forskellige parametre skulle vi så tjekke om styklisten og tegningen var det vi forventede.

Test Classes	Classes Tested	Functions being tested
ProductMapperTest	ProductMapper Production Facade	createProduct getAllProducts getProductType delete
UserMapperTest	UserMapper	login, createUser
RequestAndBOMMapperTest	RequestMapper BOMMapper BOMAlgorithm	getAllAcceptedRequests getAllNonAcceptedRequests getAllPaidRequests createBOMInDB unacceptRequest acceptRequest DeleteRequest

Arbejdsprocessen faktuel

Fase 1: 02. maj – 6. maj

Første fase begyndte, da vi startede projektet, mandag d. 2. maj. Vi begyndte her at lave mockups og diagrammer, så vi kunne få et solidt udgangspunkt for vores projekt. Om onsdagen og i løbet af torsdagen lavede vi de første jsp-sider. Det var sider, såsom forsiden og login-siden, da de udgjorde det første skridt mod at få vores hjemmeside op at køre. Denne fase blev afrundet af vores vejledningsmøde om fredagen. Her talte vi med vores vejleder, som var imponeret over at vi ikke bare sad og viste mockups, men allerede havde en visuelt funktionel hjemmeside oppe at køre.

Fase 2: 09. maj – 11 maj

Anden fase startede mandagen efter vejledningsmødet, og det var her vi så småt begyndte på back-end, altså at skrive koden og lave funktionaliteten bag det pæne ydre. Det var hovedsageligt i denne periode vi implementerede CRUD-operationer til vores hjemmeside. Denne fases afrunding fandt sted onsdag med endnu et vejledningsmøde.

Fase 3: 11. maj – 18 maj

I tredje fase fortsatte vi med at kode af programmet, med henblik på at tilføje de forslag, som vi havde fået onsdag inden. Det var også her vi begyndte på stykliste-algoritmen.

Fase 4: 18 maj – 23 maj

Fjerde fase blev dedikeret til at færdiggøre vores stykliste-algoritme og svg-tegninger, med noget general finpudsning af koden på sidelinjen.

Fase 5: 23 maj – 31 maj

Det var i femte fase, vi for alvor stoppede med at kode, og i stedet begyndte at skrive rapporten, lave videoerne og deploy vores hjemmeside. Denne kodefases slutdato er den samme, som projektets afleveringsdato, nemlig tirsdag d. 31. maj.

Vi besluttede os i starten af forløbet for, at gruppen hele tiden skulle have et medlem, med titlen KanBan-Mester. Det vil sige én, som holdt styr på hvem der var i gang med hvilke user stories hvornår, således der ikke opstod overlapninger, da det selvfølgelig ville være spil af tid, hvis dele af koden, blev kodet to gange. Vi tænkte at alle burde få lov til at prøve rollen ad, og vi valgte derfor at lade rollen som KanBan-Mester gå på skift. Projektperioden strakte sig over ca. fire uger, og da vores gruppe bestod af fire medlemmer, tænkte vi det var oplagt at tage cirka en uge hver, med undtagelse af den sidste tur, som var omkring halvanden uge.

Liste over gruppemedlemmers KanBan-Mester-periode:

Michael: 02/05 – 08/05

Sofia: 09/05 – 15/05

Kristofer: 16/05 – 22/05

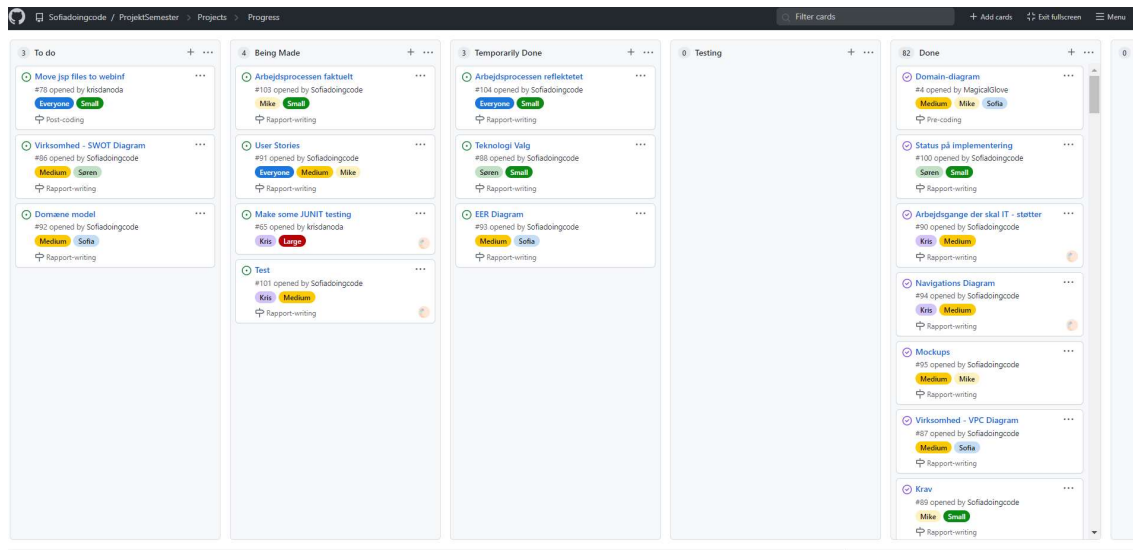
Søren: 23/05 – 31/05

Vores gruppe havde i alt fire vejledningsmøder, hvor vi fik mulighed for at få besvaret eventuelle spørgsmål og modtage konstruktiv kritik. Disse fire vejledningsmøder var spredt ud over de fire uger, med ét møde om ugen. Møderne havde alle den samme opbygning, hvor vi startede ud med at vise hjemmesiden og de implantationer vi havde fået tilføjet siden sidst. Vi fik herefter nogle forslag til hvad der kunne forbedres, og hvilke ting vi burde fokusere på.

Gruppen mødte dagligt på campus, hvor vi arbejdede det meste af dagen. Vores gruppes kommunikation uden for skoletiden, fandt udelukkende sted på platformen ”Discord”, hvilket fungerede enormt godt. Kommunikationen foregik næsten uden problemer og var generelt set en stor succes.

Arbejdsprocessen reflekteret

Nedenfor ses et billede af vores KanBan board.



KanBan Mester-rollen fungerede og KanBan-boardet er blevet holdt op to date, stort set gennem hele processen. Det var dog ikke udelukkende KanBan-mesteren som holdt styr på boardet, hvilket indikerer, at der skulle have været klarere enighed omkring, hvad rollen gik ud på, så alle forstod at det var KanBan mesterens job at have overblik over opgaverne, og hvem der var i gang med hvilke dele, men ikke nødvendigvis at lave issues og gøre det tekniske.

Vores evalueringsmøder gik hovedsageligt ud på at høre, hvordan det gik for gruppens medlemmer, og hvilke opgaver der ville give bedst mening at starte på efterfølgende.

Vores user-stories blev gjort overskuelige ved at bryde dem ned i mindre tasks, hvilket fungerede godt og gjorde processen nemmere.

Generelt set var vores estimeringer rimelig fornuftige. Da vi i ny og næ skød tidsmæssigt lidt ved siden af, havde det ikke så stor betydning, og vi tilpassede os bare efter behov.

Vi havde en vejleder, som var flink og hjalp så godt, han kunne. Der var dog et problem med hans forståelse af projektet. Hvis man f.eks. havde spørgsmål til, hvordan startkoden fungerede, var der en del usikkerhed. Det havde hjulpet en stor del, hvis lærerne havde haft

en slags fællesmøde, hvor de briefede hinanden og sikrede sig at alle var enige om kravene og forventningerne til projektet og koden.

Vi kunne godt have været mere forberedt til vejledningerne og været mere konkrete omkring præcis hvilke implantationer, vi ønskede at få feedback til. Grunden til vi dog ikke gjorde så meget ud af vores forberedelse til vejledningerne, skyldes til dels at vi følte at det ville være spildt tid, da feedbacken alligevel ikke ville være til meget gavn.

Vores gruppe fandt en god rytme og arbejdsfremgang allerede i det forrige projekt, altså Cupcakeprojektet. Denne gode rytme spillede vi videre på i dette projekt, og den gode rytme var derfor til stede, selv inden projektet gik i gang.

Vores gode rytme var baseret på medlemmernes fremragende evne til at kommunikere og vurdere om forslagene var realistiske og fornuftige. Ud fra dette, fik vi lagt en plan, som alle kunne stå ved, uden behov for at improvisere og ende ud i halvfærdige løsninger.

GitHub samarbejdet gik overordnet set enormt godt. Vores forskellige branches gav os mulighed for at arbejde parallelt, uden at påvirke andre og deres arbejde. Vi formåede også at merge vores branches sammen, på tidspunkter hvor det gav mening og fungerede. Vi havde nemlig enighed om, at vi ikke mergede vores branches ind i main, før vi var sikre på at det virkede, og alle var blevet gjort klar over det.

Bilag

Logbog

03/05-2022 –

Beslutning omkring kundens måde at genåbne og betale for bestilling

Muligheder: Unikt link sendt til email eller autogenereret, midlertidig log in.

Vi beslutter os for at systemet genererer en konto til en bruger, når deres bestilling bliver lavet. Grunden til at denne tilgang virker bedre, er fordi vi ikke har erfaring med autogenererede og sendte emails.

03/05-2022 –

Beslutning omkring modtagelse af stykliste.

Muligheder: Tilsending af email eller se stykliste ved checkout, inden kontoen slettes.

Konklusion: Brugeren kan se stykliste lige efter betaling, inden de trykker ”Videre” og kontoen slettes.

03/05-2022 –

Beslutning omkring kundens oplevelse og indsendelse af forespørgslen

Muligheder:

- Kundens request bliver med det samme sendt til admin, som så skal tage stilling til kundens carport. Kunden får her ikke mulighed for at se carport, før adminen har taget stilling til den.
- Kunden ser carporten inden den bliver sendt til admin. Her har kunden mulighed for at "accept" eller "reject" carporten, hvor den bliver sendt til en admin hvis man accepter, og man kan logge ind og lave en ny request hvis man rejcter.
- Kunden kan i bunden af sin request trykke noget alla "Generate carport", hvorefter de så kan se hvordan carporten vil se ud og eventuelle ændre den. Når de så er tilfredse med hvordan den ser ud, kan de sende den til admin.

Konklusion: Kunden kan i bunden af sin request trykke "Generate carport". Dette giver kunden mulighed for at se den og eventuelt ændre den. Når de så er tilfredse med den carport de får vist, bliver den sendt til en Admin.

04/05-2022

Beslutning omkring valg af hjemmesidens sprog

Muligheder: Engelsk eller dansk

Konklusion: Dansk, da fog er en dansk forretning, som kun befinder sig i Danmark.

10/05-2022

Beslutning omkring pop-up-vindue efter tryk af "Send forespørgselse".

Gamle løsning: Pop-up-vindue fremkommer så snart man trykker, hvorefter back-end danner konto, stykliste og tegning. Denne løsning giver ingen mening, da pop-up-vinduet skal dannes på baggrund af bacck-enden

Nye løsning: Når man trykker på "Send forespørgelse" danner back-end en konto, stykliste og tegning, hvorefter pop-up-vinduet viser brugerens konto.

10/05-2022

Beslutning omkring jsp-sidernes placering i koden igennem kodeperioden.

Muligheder:

- Flytte jsp-siderne til WEBINF nu, da det skal gøres før eller senere.
- Flytte jsp-siderne til WEBINF sidst i projektet.

Konklusion: Flytter jsp-siderne til WEBINF sidst i projektet, da det giver os mulighed for at hoppe rundt i koden via URL-en, mens vi tester og koder.

17/05-2022

Beslutning omkring algoritmes variable.

Problem: De forskellige calculate-funktioner er afhængige af andres calculate-funktioners mål (højde, længde osv.)

Konklusion: Nogle variable er public så de kan tilgås af andre funktioner:

18/05-2022

Beslutning omkring postnumre i databasen.

Problem: Bruger kan skrive forkert postnummer og dermed crashe databasen

Konklusion: Vi anerkender problemet og kender løsningen til det, men synes ikke det er tiden værd. Løsning ville være at loade alle postnumre i Danmark.