Projektrapport: Fog Carporte

maj 2020

Klasse: l20dat2ef (e klassen)

Gruppe 407:

- **Navn:** Anne-Maj Andersen
  Mail: cph-al221@cphbusiness.dk
  GitHub: Anne-Maj

- **Navn:** Claes Lindholm
  Mail: cph-cl303 @cphbusiness.dk
  GitHub: por964

- **Navn:** Casper Thomassen
  Mail: cph-ct139@cphbusiness.dk
  GitHub: Goya90

- **Navn:** Jonas Brøchner-Nielsen
  Mail: cph-jb373@cphbusiness.dk
  GitHub: jbndk

Link til projektet på DigitalOcean droplet: http://167.71.73.119:8080/Fog407-1.0/

Link til projektet på GitHub: https://github.com/Goya90/Fog

Link til Javadoc på Github pages: https://goya90.github.io/Fog/

Link til demo video af system: https://youtu.be/HYHtdozrFyE

Link til mockup i Adobe XD: https://xd.adobe.com/view/1181f999-2891-4c19-5eba-62ddfd3f1ce1-67b0/

Indhold

Indledning

Baggrund

Johannes Fog er både et bolig- og designhus samt en trælast og et byggecenter. De sælger træ, byggematerialer og, ifølge dem selv, alt til hjemmet og haven.

Fog ønsker en ny webapplikation, som deres kunder kan bruge til at bestille byg selv-carporte med specialmål, da deres nuværende system er ved at være forældet. Det er meningen, at en kunde skal kunne angive højde, længde og bredde, til- eller fravælge redskabsskur samt vælge tagtype- og materiale. Ud fra kundens valg, skal der genereres en 2D-tegning af carporten, som kunden kan se, og en materialeliste til medarbejderen hos Fog. Kundens ordre samt personlige oplysninger lagres i databasen, så medarbejderen kan godkende bestillingen og kontakte kunden.

Teknologivalg

Til projektet er anvendt følgende teknologier:

- **Java 8** (herunder **Tomcat** og **JDBC**) til websitets back-end funktionalitet
- **HTML 5** til websitets front-end opbygning
- **CSS** (herunder **Bootstrap**) til websitets front-end styling
- **IntellJ IDEA Ultimate version 2020.1** som udviklingsværktøj
- **MySQL** til databasen
- **SVG** til tegning af carport og skur
- **DigitalOcean** som host af websitet og databasen
- **Adobe XD** til mockup

Krav

Virksomhedens vision

Underviserne på Cphbusiness har besøgt og filmet afdelingslederen hos Fog, som viser, hvordan deres nuværende system fungerer samtidig med, at han forklarer, hvad de ønsker fremover. De vil gerne bevare mange af de samme funktioner, men kunne godt tænke sig, at det nu skal være muligt at vælge højde på carporten. Derudover er der nu en salgsmedarbejder, som mellemled, når styklisten skal udarbejdes, hvor de godt kunne tænke sig, at programmet automatisk genererede den.

Arbejdsgange der skal IT-støttes

Systemet har til formål at understøtte en række arbejdsgange omkring salg af carporte hos Fog, og medføre at Fog kan øge deres fortjeneste ved hjælp af systemet i disse arbejdsgange:

- **Afgivelse af ordren** kan ske digitalt og kan gøres af kunden selv. Det betyder at Fog kan spare omkostninger til løn til de medarbejdere der ellers skulle tage i mod ordre. Alt andet lige kan det også øge omsætningen, da det gør det nemmere for kunderne at afgive en bestilling.
- **Udarbejdelse af materialeliste og tegning** sker automatisk og dermed skal Fogs medarbejdere blot sende materialet til kunden når betaling er modtaget. Det betyder at Fog kan spare lønomkostninger til de medarbejdere, der ellers skulle have beregnet materialelisten og tegnet tegningen manuelt.
- **Kunders kontaktoplysninger** er let tilgængeligt i systemet for medarbejderne, og de har dermed styr på kunderne og deres ordre. De kan også se hvilke ordrer der er behandlet. Det mindsker risikoen for fejl og øger effektiviteten.

User stories

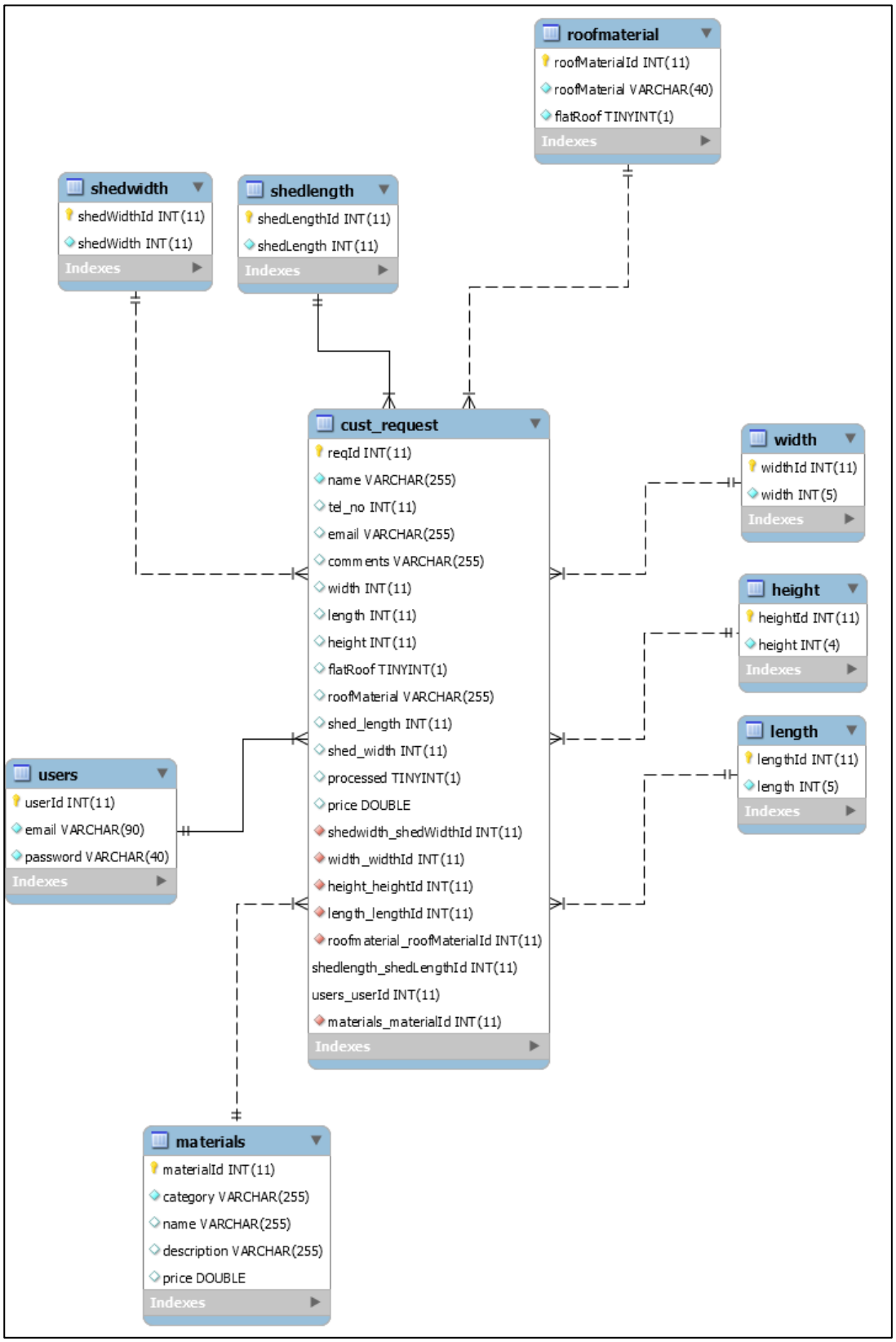
Ud fra videoen med Fog-medarbejderen, har vi i gruppen formuleret en række user stories:

- US-1: As a customer I want to be able to choose custom width, length, and height so that I can design a carport.
- US-2: As a customer I want to be able to choose custom width, length, and height so that I can get a bill of materials.
- US-3: As a customer I want to be able to choose between flat and slanted roof.
- US-4: As a customer I want the bill of materials to reflect my choice of roof type.
- US-5: As a customer I want to be able to choose material for the roof.
- US-6: As a customer I want the bill of materials to reflect the choice of roof material.
- US-7: As a customer I want to be able to add a tool shed to the carport.
- US-8: As a customer I want the bill of materials to reflect if a tool shed is added.
- US-9: As a customer I want to be able to see a 2D-drawing of the carport I have chosen.
- US-10: As a customer I want the drawing to include a tool shed if a tool shed is added to the carport.
- US-11: As an employee I want to be able to log in as “employee” to access non-public data.
- US-12: The material list should only be visible if you are logged in as an employee.
- US-13: As a customer I want to be able to enter my contact information so that Fog can contact me.
- US-14: As an employee I want to be able to get an overview of submitted forms so that I can contact the customers.
- US-15: As an employee I want to be able to see the price from the chosen dimensions

- US-16: As an employee I want to edit the price on an order
- US-17: As a user I want the web page to notify me of missing input in the form so that I am not able to submit blank values
- US-18: As an employee I want to be able to mark submitted forms as “done” in the overview so that I know which customers to contact

ER diagram

Nedenfor findes et ER diagram over domænet, samt vores overvejelser I forbindelse med opbygningen af indholdet.



Designet tager udgangspunkt I en user (medarbejder), som logger ind og kan tilgå indkommende customer requests. Alle nyindkomne customer request er markerede som ubehandlede. Her kan medarbejderen se, hvad kunden, i en browser har valgt. Kunden tilgår en browser, hvor han først bliver præsenteret med en side, hvor han kan designe en carport med fladt tag. Han kan vælge at klikke på et link, som fører ham til en side, hvor en carport med rejsning kan designes. Dette er opnået gennem flatroof variablen i roofmaterial tabellen, som er en boolean. Kunden bliver bedt om at vælge mellem en række dimensioner og materiale-type. De ligger som rækker i de respektive tabeller i databasen. De valgte dimensioner og materialetyper bliver samlet i customer request, hvor også kundens indtastede kontaktoplysninger gemmes. På bagrund af dimensioner og materiale bliver en pris beregnet, som også ligger i request. Eftersom kunden ikke skal logge ind, giver det mening at samle al information i tabellen customer request.

Alle tabeller er opbygget med en primary key. 3. Normalform er overholdt i alt bortset fra customer request. Primary keys bruges, når materiallist og pris samt dimensioner skal udregnes.

Vi kunne have lavet en customer tabel, da dette ville normalisere customer request tabellen. Primary key i customer request er for nuværende afhængig af for mange forskellige data, til at 3. normalform overholdes.

Det skal også nævnes, at man kunne have valgt at sætte SQL til at generere et timestamp under customer request tabellen, da dette ville lette ekspeditionen for medarbejderen.

Vi har benytter hverken 1:1 eller mange:mange relationer blandt tabellerne. Grunden til dette er, at vi typisk ville slå tabellerne sammen, hvis vi opdagede en 1:1 relation, og typisk opdelt en mange:mange relation ved at indføre en tabel mellem denne relation (bridge-tabel).

Generelt er ideen bag designet, at gøre domænet så gennemsigtigt og pålideligt som muligt. Dette er blandt andet opnået ved at lave alle dimensioner og materiale-typer til seperate tabeller. Det er generelt en god ide, at holde tabellerne små, hvis det giver mening, da det gør det nemmere og langt mere overskueligt at opdatere og hente data.

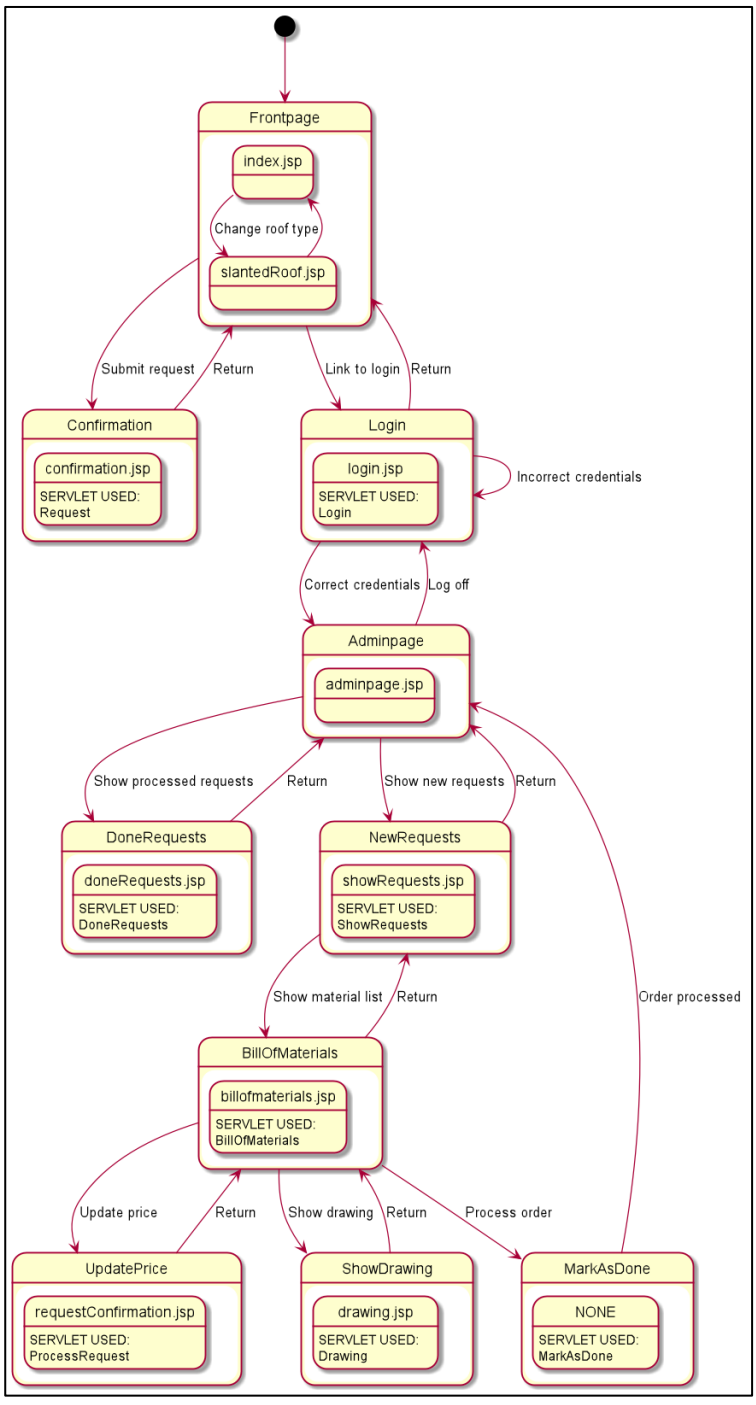
Alle tabeller har et ID som auto-incrementer. Dette sikrer at alle rækker er unikke, og derfor nemt kan hentes via queries.

Navigationsdiagram

Vi har valgt kun at inkludere det overordnede navigationsdiagram, da vi har kunne få alle relevante detaljer ind i diagrammet og derfor giver det ikke mening at lave separate diagrammer for hver enkel navigation.

Den sorte cirkel som diagrammet starter med er brugeren. I boksen Frontpage er det illustreret hvordan brugeren kan navigere mellem de to forsider ”index.jsp” og ”slantedRoof.jsp” og dermed skifte mellem de to tagtyper. Hvis brugeren indsender en forespørgsel på en carport bliver bruger sendt til siden ”confirmation.jsp”. Her bliver brugt en servlet ved navn ”Request” i systemet.

Samme koncept anvendes i resten af diagrammet. I bilag 2 er koden for diagrammet som er lavet i PlantUML.

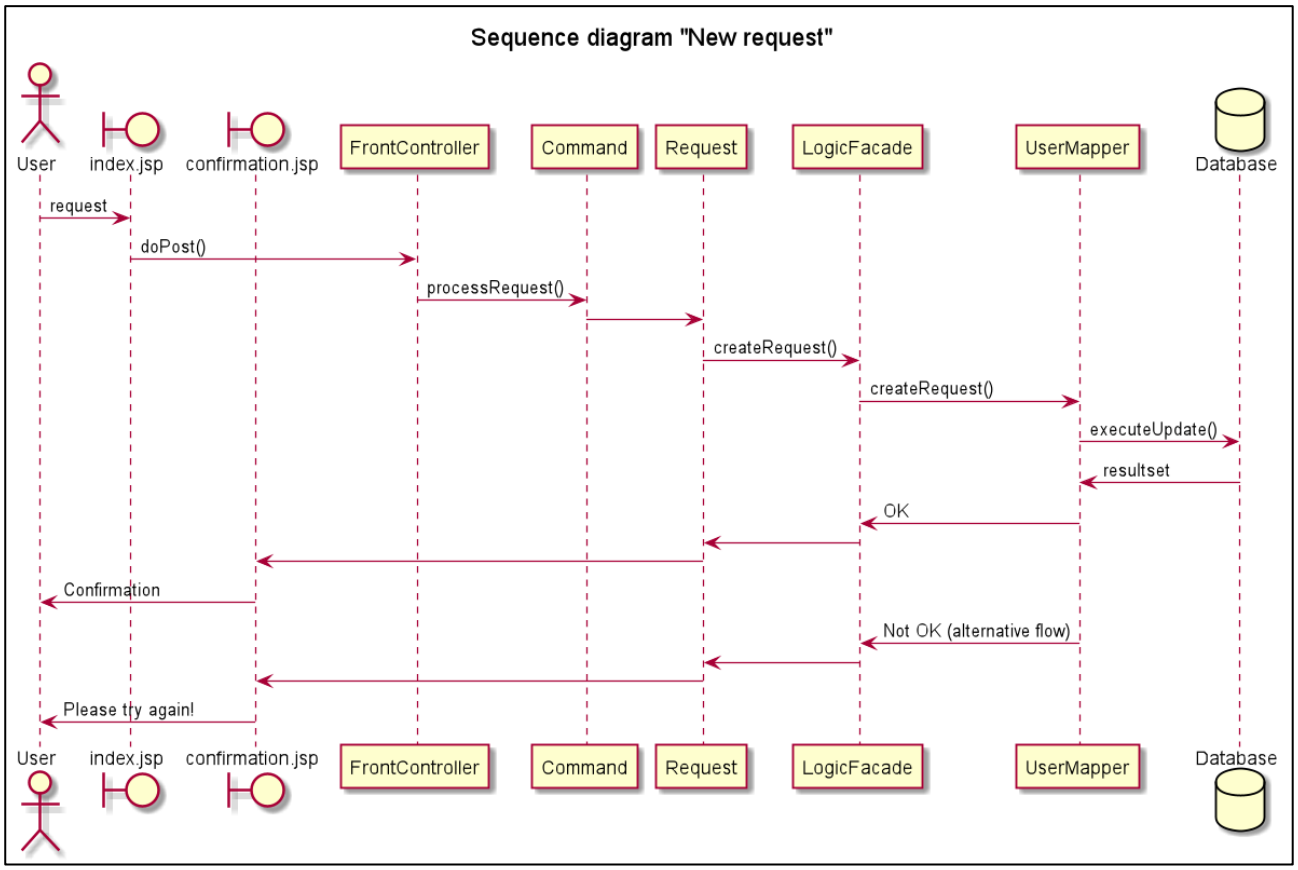


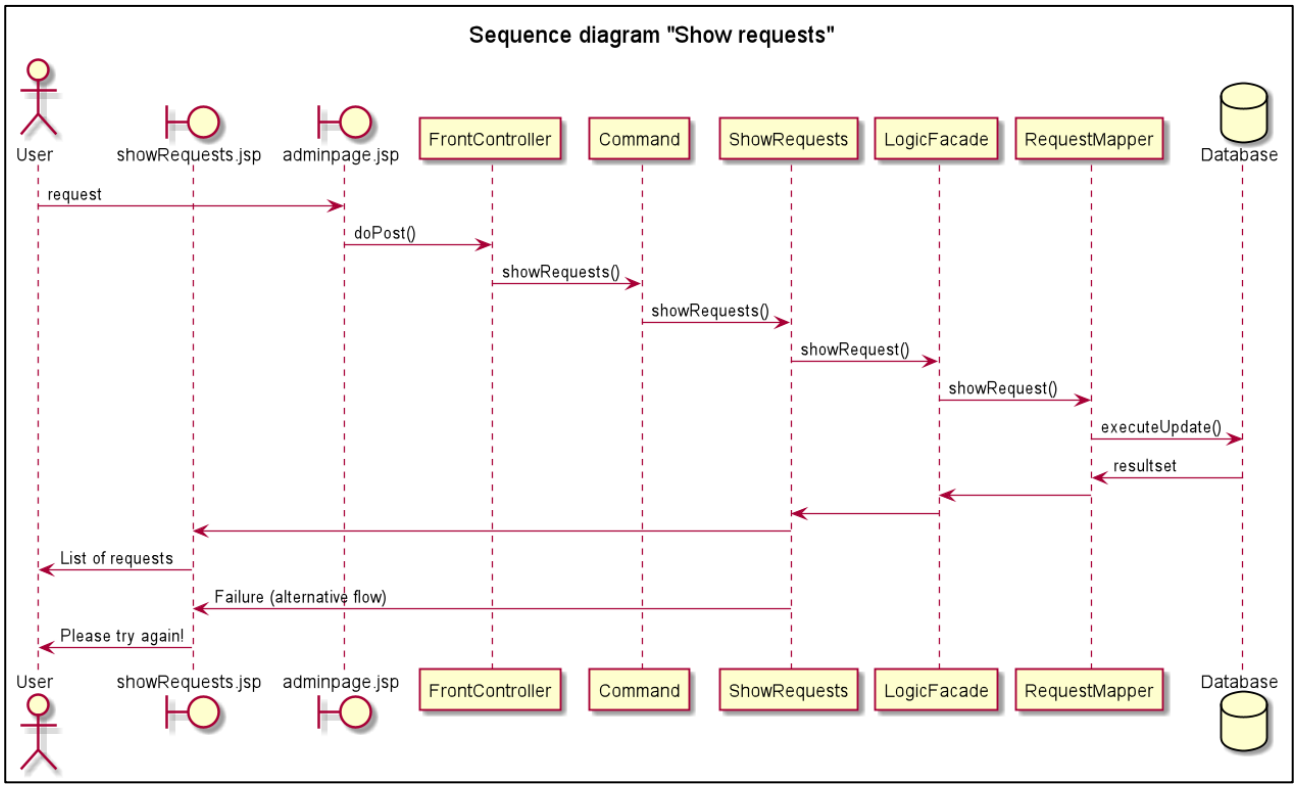
Sekvensdiagram

Vi har valgt at inkludere to centrale sekvensdiagrammer i denne rapport. Det har vi gjort, da vi synes at diagrammerne danner et godt overblik over programmet, og hvordan programmet bruger Command pattern med FrontController.

Diagrammet ”New request” viser sekvensen hvor brugeren indsender en ny forespørgsel indeholdende information om carport, skur og kontaktoplysninger. Diagrammet er lavet i PlantUML og koden kan findes i bilag 3.

Diagrammet ”Show requests” viser sekvensen hvor en bruger får vist alle ubehandlede forespørgsler i en tabel. Diagrammet er lavet i PlantUML og koden kan findes i bilag 4.





Særlige forhold

Informationer, der gemmes i sessions

I web-applikationen benyttes tre forskellige scopes:

- **Application:** Objekter af typen application er fælles for alle sider og alle brugere af webapplikationen. Når en kunde har valgt carportmål, lagres de som et objekt, hvis attributter kan tilgås indtil webapplikationen lukkes.
- **Session:** Når en medarbejder logger på Fogs web-applikation, tildeles et session-objekt, som følger brugeren. Attributterne bruger, email og brugerID sættes, og her bliver de gemt indtil brugeren logger ud. Materialelisten bliver også gemt i en session.
- **Request:** Kundeforespørgslerne og svg-tegningen er gemt i en request scopes, hvilket betyder, at de er tilgængelige i en enkelt http-request.

Exceptions

I vores web-applikation bruges følgende exceptions:

- **LoginSampleException:** Kastes, når en bruger, der forsøger at logge ind, ikke kan valideres. Dette er en custom exception.
- **ClassNotFoundException:** Kastes, når JVM forsøger at åbne en specifik klasse, og denne ikke findes i klassestien.

- **NumberFormatException:** Kastes, når en String ikke kan konverteres til en numerisk værdi
- **NullPointerException:** Kastes, når programmet forsøger at bruge et objekt med værdien 0. For eksempel, hvis en medarbejder indtaster id på en kundeforespørgsel, som ikke findes.
- **SQLException:** Kastes både, når der ikke er forbindelse til JDBC-driveren, og når der er fejl i databasen.

Brugertyper i databasen

Da det kun er medarbejderen, der kan logge ind i vores system, er der ikke flere forskellige typer af brugere.

Validering af brugerinput

På index.jsp har vi benyttet os af validering på to måder. Den første er i de dropdown-menuer, hvor en kunde vælger højde, længde, bredde og tagmateriale. Her har vi gjort sådan, at der allerede er sat en defaultværdi, som benyttes, hvis kunden ikke selv aktivt vælger én. Dette gør, at vi ikke får nogle blanke felter og dermed nulværdier.

Når det kommer til valg af redskabsskur, så er defaultværdien sat til ”0”, som for kunden vises som ”ønsker ikke redskabsrum”.

```
<div class="form-group">
    <label for="height">Vælg højde:</label>
    <select class="form-control" name="height" id="height" style="...">
        <c:forEach var="height" items="${applicationScope.height}">
            <option value="${height}">${height} mm.</option>
        </c:forEach>
    </select>
</div>

<div class="form-group">
    <label for="roofMaterial">Vælg materiale til tag:</label>
    <select class="form-control" name="roofMaterial" id="roofMaterial" style="...">
        <c:forEach var="roofMaterial" items="${applicationScope.roofMaterial}">
            <option value="${roofMaterial}">${roofMaterial}</option>
        </c:forEach>
    </select>
</div>

<div class="form-group">
    <label for="shedWidth">Vælg bredde til redskabsrum:</label>
    <select class="form-control" name="shedWidth" id="shedWidth" style="...">
        <option value="0" selected>Ønsker ikke redskabsrum </option>
        <c:forEach var="shedWidth" items="${applicationScope.shedWidth}">
            <option value="${shedWidth}">${shedWidth} mm.</option>
        </c:forEach>
    </select>
</div>
```

Når kunden skal indtaste sine kontaktoplysninger, er nogle af felterne sat til at være ”required” i html. Det betyder, at det ikke er muligt at trykke ”Send forespørgsel”, før der er indtastet noget. I email-feltet har vi sat input type til at være ”email”. Kunden kommer kun videre, hvis der er skrevet en email-adresse i feltet, og under telefonnummer er input typen sat til ”tel” med mønstret 8 cifre fra 0-9.

Kommentarfeltet er valgfrit, så kunden kommer videre uanset om det er udfyldt eller ej.

```
<br>
<h3>Indtast dine kontakt detaljer:</h3>
<div class="form-group">
    <label for="name">Navn</label>
    <input type="text" class="form-control" name="name" id="name" placeholder="Dit navn" required>
</div>
<div class="form-group">
    <label for="mail">Email addresse</label>
    <input type="email" class="form-control" name="mail" id="mail" placeholder="Din E-mail" required>
</div>
<div class="form-group">
    <label for="telno">Telefonnummer</label>
    <input type="tel" class="form-control" name="telno" id="telno" placeholder="Dit telefonnr" pattern="[0-9]{8}" required>
</div>
```

Når en medarbejder er logget ind, har denne adgang til at se kundeforespørgsler. I bunden af skemaet er en rubrik, hvor der skal tastes ordrenummer for den materialeliste, man ønsker at se. Hvis man ikke taster et heltal, smides en NumberFormatException med teksten ”Fejl: Du har ikke indtastet et heltal, ret din indtastning”. Tastes der et heltal, hvor der ikke findes en ordre med det nummer, kastes der en NullPointerException, som via en fejlmeddelelse gør medarbejderen opmærksom på, at der ikke findes en kundeforespørgsel med det indtastede nummer.

På siden, hvor en medarbejder kan ændre prisen på en garage valideres input på den måde, at hvis den nyindtastede pris er højere end 0, så sættes den til det, men hvis prisen ikke er højere end 0, kastes en NumberFormatException, og på requestet sættes attributten som ”error” med teksten ”Fejl: Den nye pris skal være et positivt tal, prøv igen”

Sikkerhed i forbindelse med login

På siden for medarbejderlogin, login.jsp, er feltet til brugernavn og kodeord ligeledes sat til som ”required”, så der ikke kan logges ind, hvis et af felterne er tomme. På sidstnævnte felt er inputtypen sat som ”password”, hvilket gør, at de tegn, der skrives vises som prikker.

Når en bruger taster brugernavn og kodeord, sættes gang i metoden login i UserMapper.java. Hvis kodeord og brugernavn ikke passer sammen, kastes en LoginSampleException med beskeden ”Could not validate user, please try again”.

JSP siderne, der ligger i mappen WEB-INF kan kun tilgås, når brugeren er logget ind.

Tegning

SVG er et billedformat, som genereres dynamisk ud fra de valgte carportdimensioner.

I klassen Drawing.java hentes attributterne længde og bredde for både carport og eventuelt skur. Viewboxen sættes, så størrelsen er afhængig af disse mål. Derefter oprettes et Svg-objekt med målene, og fra Svg.java tilføjes rektangel (carporten). Diverse variabler, der skal bruges, instantieres, og der tilføjes remme.

Stolper tilføjes, mængden afhænger af carportens dimensioner og om der er valgt skur. Herefter tegnes krydset, hvorefter antallet af spær tilføjes. Til slut sættes attributten ”svgdrawing”, som er toString-metoderne i Svg.java.

Styklisteberegner

Styklisteberegneren er vedlagt i bilag 1 og i afsnittet ”Udvalgte kodeeksempler” neden for, er det også beskrevet hvordan vi har brugt styklisteberegneren som grundlag for den algoritme der viser materialelisten.

Der er i projektet ikke afsat ressourcer til at lave en fuldstændig nøjagtig og korrekt beregning af alle materialer. Der er derfor gjort en række antagelser og simplificeringer. For en byggefaglig person vil mange information i materialelisten derfor være åbenlyst forkerte, men selve algoritmen og det kodemæssige er korrekt. Det er blot datagrundlaget som er ukorrekt. Hvis PO have ønsket det, og der dermed var afsat ressourcer i projektet til det, kunne man have gøre materialisten helt korrekt byggefagligt, med relativt få ændringer i algoritmen og datagrundlaget.

Udvalgte kodeeksempler

Java-klassen Calculator indeholder selve algoritmen, der udregner, hvilke materialer, der skal benyttes til en carport med specialmål.

I Calculator instantieres en statisk arrayliste med materialer-objekter. Den er statisk, fordi klassen ShedCalculator.java tilføjer materialer til samme liste.

Der er i alt 38 forskellige typer af materialer (se bilag 1), som alle er lagret i databasen, og har et unikt idnummer, et navn, en beskrivelse, en kategori og en pris. I Calculator-klassen er der 38 undermetoder, som hver især tilføjer ét materiale. De hedder ”addMaterial1”, ”addMaterial2” osv.

Materialer har blandt andet attributterne ”quantity” og ”price”. Ud fra matematiske formler udregnes og sættes disse to attributter i hver enkelt addMaterial-metode. Visse typer af materiale har også attributten ”length”, som også sættes her. Det er målene på carporten, der afgør antal, pris og længde på materialerne.

Hovedmetoden, som hedder bomCalculator (bom står for ”bill of materials”), tager imod de parametre, som kunden har indtastet på forsiden (højde, længde, bredde, tagtype, tagmateriale, længde og bredde på skur).

Allerførst i bomCalculator kaldes de addMaterial-metoder, som tilføjer materialer, der skal bruges i enhver form for carport.

Dernæst skal der tilføjes materialer til taget. Et tag kan enten være fladt eller have resjning. Ved hjælp af en if-else-sætning, tilføjes der materialer til den ønskede tagtype.

Herefter kommer der en switch case, som, ud fra den tagtype, kunden har valgt, tilføjer lige netop det materiale, der skal benyttes. Default er sat til addDummyMaterial(), som en metode, der bruges til at vise, at der ikke blevet tilføjet et materiale.

```
//Hvis taget er fladt tilføjes disse materialer:
if (flatRoof) {
    addMaterial2();
    addMaterial7();
    addMaterial18();
    addMaterial22();
    addMaterial25();
}

//Hvis taget har rejsning tilføjes disse materialer:
else {
    addMaterial10();
    addMaterial11();
    addMaterial12();
    addMaterial13();
    addMaterial16();
    addMaterial17();
    addMaterial19();
    addMaterial20();
    addMaterial21();
}
```

```
//Tilføjer tag materiale:
switch (roofMaterial) {
    case "Plastmo sort":
        addMaterial34();
        break;
    case "Plastmo gennemsigtig":
        addMaterial35();
        break;
    case "Plastmo hvid":
        addMaterial36();
        break;
    case "Tagsten sort":
        addMaterial15();
        break;
    case "Tagpap sort":
        addMaterial37();
        break;
    case "Trapez plast sort":
        addMaterial38();
        break;
    default:
        addDummyMaterial();
}
```

Hvis kunden har tilvalgt redskabsskur, skabes en instans af klassen shedCalculator, og på denne instans kaldes metoden shedBomCalculator, som modtager parametrene carportens højde samt længde og bredde på skuret.

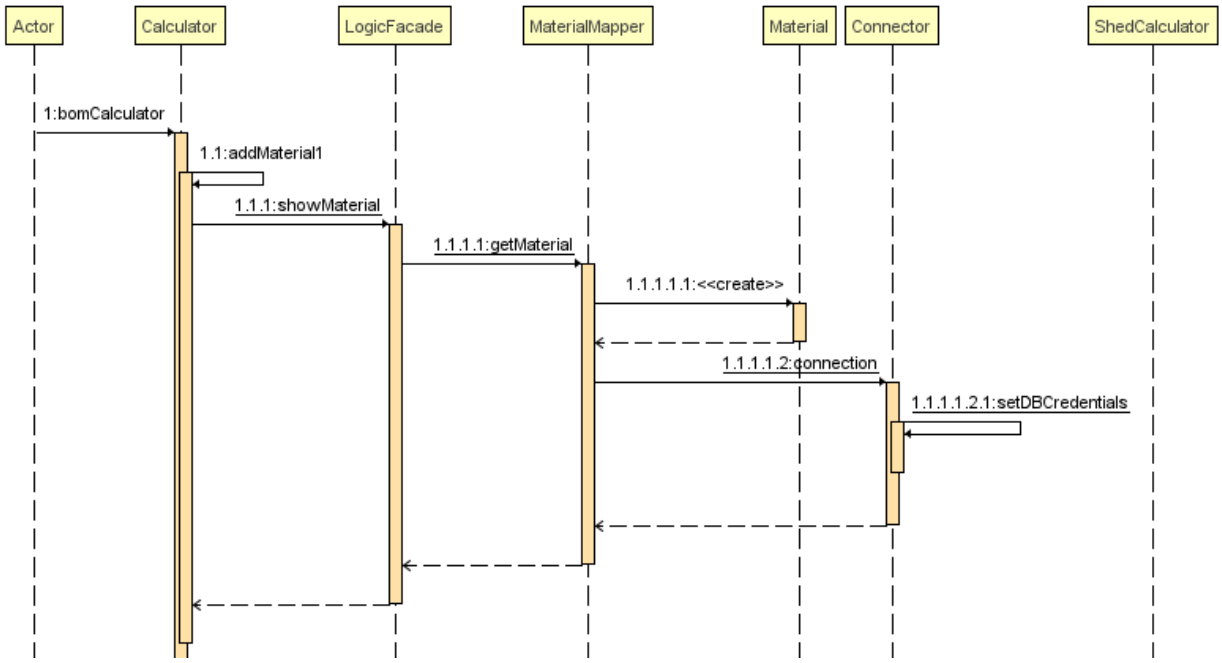
Til et skur er der fire forskellige materialer, og dermed fire metoder, der skal kaldes. De fire metoder udregner og sætter antal og længde og tilføjer materialerne til arraylisten ”bom”.

På Index.jsp og slantedRoof.jsp, vælger kunden carportens mål mm. ved hjælp af dropdown-menuer. Disse befolkes fra databasen. Det er java-klassen Initializer, der sørger for, at befolke dropdown-menuerne med mål til carporten og materialer til taget. Det gøres via LogicFacade, som kalder metoder fra DimensionMapper og returnerer lister med de mål og tagmaterialer, kun kan vælge imellem.

Når en kunde har sendt en forespørgsel, sendes de valgte mål til FontController, som videresender dem til Request.java, der skaber et CustomerRequest-objekt. Dataen sendes via LogicFacades metode createRequest videre til RequestMapper, hvis metode, som også hedder createRequest, sender dataen til MySQL, hvor den gemmes i tabellen ”cust_request”.

Kunden sendes videre til jsp-siden ”confirmation”, som bekræfter ordren og viser, hvad der er valgt samt oplyser om, at en medarbejder hos Fog vil henvende sig snarest.

Når en medarbejder er logget ind, er der mulighed for at se både nye og behandlede kundeforespørgsler.



På jsp-siden billofmaterials kan en medarbejder hos Fog se hele materialelisten. Den vises i skemaer, som er opdelt efter materialekategori. Medarbejderen har mulighed for at ændre prisen, og give kunden rabat.

Fog®

Carport nr: 12 har følgende mål:

Højde: 2000 cm, bredde: 4400 cm, længde: 2800 cm
Skur længde: 0 cm, bredde: 0
Tagmateriale: Plastmo sort

Materialeliste

Kategori	Navn	Beskrivelse	Antal	Længde	Pris
Træ	25x200 mm. trykimp.bræt	Stern vandbræt sider	2	3400	100,00
Træ	97x97 mm. trykimp. stolpe	Stolpe	4	2900	239,80
Træ	45x195 mm. spærtræ ubeh.	Remme i sider	2	2800	500,00
Træ	25x200 mm. trykimp.bræt	Stern vandbræt forende	1	4400	50,00
Træ	45x195 mm. spærtræ ubeh.	Spær	5	4400	1.250,00

Kategori	Navn	Beskrivelse	Antal	Pris
Beslag & skruer	Universal 190 mm højre	Beslag til spær på rem	5	850,00
Beslag & skruer	Universal 190 mm venstre	Beslag til spær på rem	5	875,00
Beslag & skruer	Bræddebolt 10x120 mm.	Montering rem på stolper	8	120,00
Beslag & skruer	Firkantskiver 40x40x11 mm.	Montering rem på stolper	8	80,00
Beslag & skruer	4,5x60 mm. skruer 200 stk.	Skruer til montering af stern, vindskeder og vandbræt	1	420,00
Beslag & skruer	Hulbånd 1x20 mm. 10 mtr.	Vindkryds på spær	1	160,00
Beslag & skruer	4,0 x 50 mm. beslagskruer 250 stk.	Skruer til universalbeslag + hulbånd	1	550,00

Kategori	Navn	Beskrivelse	Antal	Pris
Tagpakken	Plastmo bundskruer 200 stk.	Skruer til tagplader	1	380,00
Tagpakken	Plastmo sort 1200x1200 mm.	Tagplade	17	5.100,00

Listepris: 10.674,80 kr inkl. moms
Opdateret pris: 9.500,00 kr inkl. moms

Ny pris: Opdater pris

Marker ordre som færdigbehandlet

Status på implementering

I dette afsnit præsenteres, hvor langt vi er nået med implementering af projektet. Vi har leveret den funktionalitet som product owner har krævet, så de formelle funktionskrav er opfyldt. Her følger en oversigt over funktionalitet som ville være en naturlig tilføjelse hvis vi havde fortsat dialogen med product owner.

Sikkerhed:

Sikkerheden i systemet kan udbygges. Pt består sikkerheden af brug af WEB-INF samt bruger login. Hvis man kender stien til Tomcat WEB-INF mapperne, vil man som udefrakommende kunne tilgå en medarbejder-side uden at være logget ind. Hvis vi skulle fortsætte med opgaven, ville vi foreslå en udbygning af sikkerheden. Derudover ville højere krav til brugeres password samt eventuelt kryptering være et af vores forslag til forbedring af sikkerheden.

Styling:

Vi har valgt et simpelt design, hvor vi har brugt Bootstrap og CSS. Dette giver en pæn brugergrænseflade, som samtidigt er overskuelig og funktionel. Vi ville forvente at product owner havde større og mere detaljerede krav til styling og design af deres websider. Vi har valgt denne enkle løsning, da Bootstrap og CSS var fokus på semestret. Hvis vi skulle have gået videre med stylingen, kunne vi for eksempel have reverse engineered Fog's eksisterende styling, og matchet den til hvad Fog i forvejen bruger. Det gør sig gældende for layout, farver, font og logoer.

Dynamik:

Der blev ikke stillet krav om dynamiske websider hvilket ville være en naturlig forbedring. Eksempelvis kunne det have været en fin løsning, hvis medarbejderen med et klik kunne fjerne en ordre fra nye ordrer, så den ikke længere fremgik af listen og I stedet fremgik af behandlede ordrer.

Hvis vi havde gået videre med dynamikken, havde vi benyttet Javascript til at lave mouse-overs, dynamisk opdatering af nye mål, materialer og typer. Vi ville også have benyttet Javascript til at gøre tegningen dynamisk. Dette ville højne brugeroplevelsen.

Det skal nævnes at Javascript ikke er en del af pensum, og at vi er blevet påbudt ikke at bruge det.

Datoer:

Dato og tidsstempling af data. Der blev ikke stillet krav om mulighed for sortering af data baseret på tidspunkt. Efter implementering ville mængden af data meget hurtigt blive meget stor. Dette ville sandsynligvis skabe krav om mulighed for at sortere data afhængigt af tidspunkt. Skulle vi fortsætte arbejdet for product owner ville vi foreslå at tidsstemple data og lave metoder til at sortere dette.

Test

I dette projekt har vi benyttet os af både unit- og integrationstests. Førstnævnte tester komponenter af koden, såsom metoder, der enten udregner eller udfærdiger en 2D-tegning efter kundens ønskede mål på carporten.

Integrationstestsene inddrager flere lag, og undersøger, om vores kode fungerer i samarbejde med databasen.

Som testværktøj, har vi benyttet JUnit og Hamcrest. Sidstnævnte er et framework, som giver lidt flere muligheder i forhold til at teste med assertions og matchers

Inden test af Mapper-klasserne oprettes der en tom database i MySql ved navn ”fogtest”. I ”@BeforeClass” oprettes der tabeller magen til dem, der findes i de klasser, der testes. Disse tabeller er til for, at vi ikke manipulerer med de tabeller, som webapplikationen bruger.

Oversigt over hvilke metoder, der er testet

Pakke: DBAccess

DimensionMapper	MaterialMapper	RequestMapper	UserMapper
getHeightList	getMaterial	createRequest	login
getLengthList	getRoofMaterials	showNewRequests	
getWidthList		getRequestFromID	
getShedLengthList			
getShedWidthList			

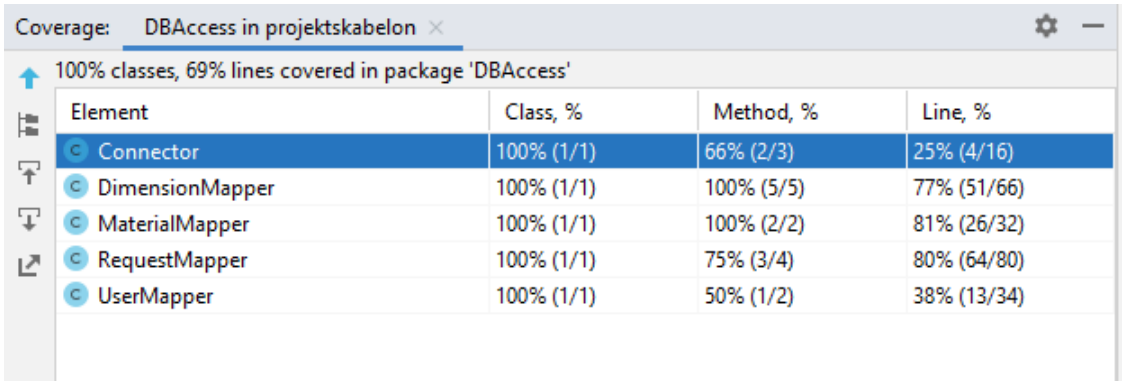
Der testes først og fremmest, om forbindelsen til databasen er ok.

Klassen DimensionMapper bruges til at hente de mål til carport og skur, som en kunde kan vælge i dropdownmenuen på index-siden. Der testes, at arraylisterne med målene hentes, og at indhold og størrelse er korrekte.

MaterialMapper er klassen, der henter materiale i databasen. Igen testes der, om arraylisterne får den rigtige størrelse, og om de indeholder de ting, de skal.

I RequestMapper oprettes der et nyt customer request, som lagres i databasen. Herefter testes det, at arraylisten med customer requests har den rigtige størrelse, samt at et request kan hentes ud fra id.

Som det ses i skemaet nedenfor, er der i DBAccess-mappen en test coverage på 100% af metoderne i Mapperklasserne. Der er ingen testklasse for Connector, men den testes indirekte via de andre klasser.

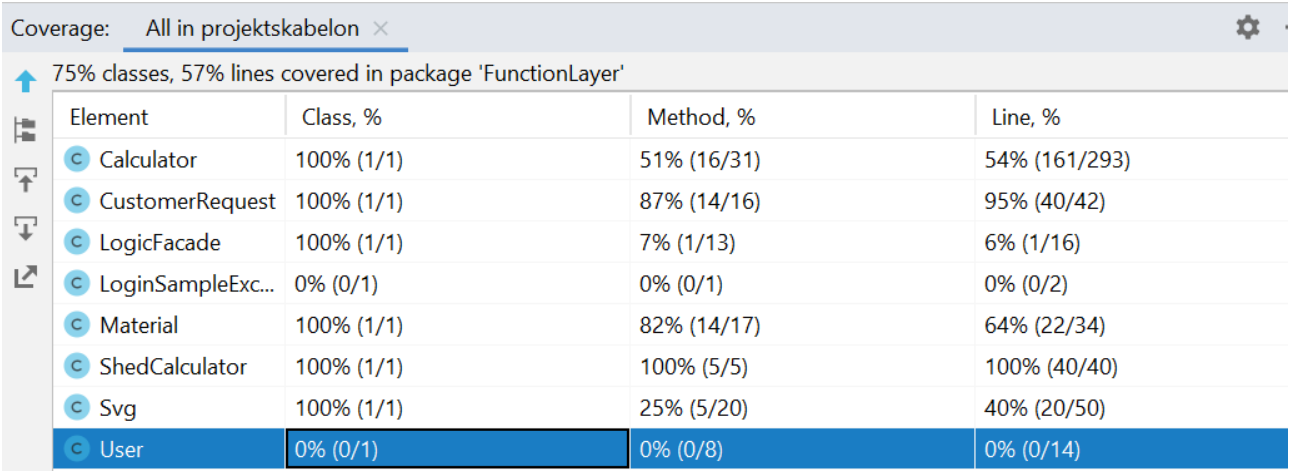


Pakke: FunctionLayer

Calculator	Svg
bomCalculator, herunder:	addRect
- Fladt tag med skur	addStrap
- Fladt tag uden skur	addCross
- Tag m. rejsning uden skur	
- Tag m. rejsning med skur	

I denne pakke, har vi testet:

- Svg.java: De metoder, der tilføjer henholdsvis rektangel, kryds og stropper til 2D-tegningen. I denne klasse er der en del get- og setmetoder, som ikke er testet, hvilket giver en lav overordnet test coverage (25%).
- Calculator: Metoden bomCalculator er testet med forskellige typer af carport. Dens undermetoder, som alle hedder noget med addMaterial(), er ikke testet, da de ikke returnerer noget. Der testes på to ting i Calculator: Først, om antallet af materialer, der skal bruges til en given carport er korrekt, og dernæst, om den samlede pris er rigtig.
- ShedCalculator: Testes indirekte gennem Calculator, da Calculator opretter en instans af ShedCalculator såfremt der skal tilføjes et redskabsskur.



Element	Class, %	Method, %	Line, %
Calculator	100% (1/1)	51% (16/31)	54% (161/293)
CustomerRequest	100% (1/1)	87% (14/16)	95% (40/42)
LogicFacade	100% (1/1)	7% (1/13)	6% (1/16)
LoginSampleExc...	0% (0/1)	0% (0/1)	0% (0/2)
Material	100% (1/1)	82% (14/17)	64% (22/34)
ShedCalculator	100% (1/1)	100% (5/5)	100% (40/40)
Svg	100% (1/1)	25% (5/20)	40% (20/50)
User	0% (0/1)	0% (0/8)	0% (0/14)

Proces

Arbejdsprocessen faktuelt

I dette projekt har vi arbejdet med agil udvikling efter scrum-modellen.

Vi har haft fire sprints af én uges varighed. Hvert sprint er begyndt med et scrum planning-møde med Product Owner (herefter omtalt som PO), hvor han har udvalgt nogle user stories, som vi skulle arbejde med. Til scrum review har vi for PO fremvist de dele af programmet, som relaterer sig til de enkelte user stories for den indeværende sprint. PO har så enten godkendt eller afvist løsningerne.

På denne måde har vi kunnet levere noget kode og præsenteret et program, der har virket til hver review, og vores webapplikation er gradvist blevet bygget op med tilføjelser af nye funktioner.

I begyndelsen udvalgte vi Jonas som scrum-master, da vi fra et tidligere projekt ved, at han har et godt overblik. Han har primært fungeret som den, der har delt skærm og administreret Taiga (som er et stykke online værktøj til at holde styr på scrum projekter) og præsenteret projektet for PO.

Pga. Covid-19 har det ikke været muligt for os at mødes fysisk. Koordination af arbejdsgangen er sket via Zoom-møder og samtaler på Slack. Udveksling af kode er foregået via GitHub.

Efter hvert møde med PO, har vi holdt et møde i gruppen, hvor vi har evalueret den seneste sprint og fordelt arbejdsopgaverne for den næste sprint mellem os. Under sprinten har vi afholdt møder efter behov, så uden faste intervaller.

Sprint 1 (17/4-23/4)

US-1 og US-2.

Subtasks til US-1: *As a customer I want to be able to choose custom width, length, and height so that I can design a carport*

- Lav en mock up i Adobe XD

- Opret index.jsp
- Opret github-projekt med MVC
- MySql database med højde, længde og bredde
- Lav drop down menuer, som henter mål fra databasen
- Styling af index.jsp
- Skab submit funktion

Subtasks til US-2: *As a customer I want to be able to choose custom width, length, and height so that I can get a bill of materials*

- Lav datagrundlag for algoritme
- Undersøg hvordan data flyttes fra index-siden til java (beregner)
- Skab en arrayliste af materiale-objekter, som kan vises på materialPage.jsp
- Lav en beregner, der kan oprette objekter i en materiale-arrayliste

Sprint 2 (24/4-30/4)

US3-US9

Subtasks til US-3: *As a customer I want to be able to choose between flat and slanted roof.*

- Tilføj dropdown-menu til index.jsp med tagtyper
- Opdater MySql og pricePage.jsp, så prisen afhænger af tagtype

Subtasks til US-4: *As a customer I want the bill of materials to reflect my choice of roof type.*

- Opdater databasen med tagtyper

Subtasks til US-5: *As a customer I want to be able to choose material for the roof.*

- Opdater databasen, så den indeholder tagmateriale

Subtasks til US-6: *As a customer I want the bill of materials to reflect the choice of roof material.*

- Tilføj en dropdown-menu til index.jsp til valg af tagmateriale
- Opdater MySql og pricePage.jsp, så prisen tager højde for tagmaterialet

Subtasks til US-7: US-7: *As a customer I want to be able to add a tool shed to the carport.*

- Tilføj en dropdown-menu med valg af mål til redskabsskur

Subtasks til US-8: US-8: *As a customer I want the bill of materials to reflect if a tool shed is added.*

- Opdater databasen med mål til redskabsskur

Subtasks til US-9: *As a customer I want to be able to see a 2D-drawing of the carport I have chosen.*

- Opret en java-klasse til tegning
- Opret en jsp-side til tegning

Sprint 3 (1/5-7/5)

US-10 - US-14.

Subtasks til US-10: *As a customer I want the drawing to include a tool shed if a tool shed is added to the* carport.

- Tilføj redskabsskur til Svg-tegningen

Subtasks til US-11: *As an employee I want to be able to log in as "employee" to access non-public data*

- Lav login.jsp
- Lav felter til brugernavn og kodeord
- Login-funktion med redirect til en side, som ikke er offentligt tilgængelig

Subtasks til US-12: *The material list should only be visible if you are logged in as an employee.*

- Tilføj fejlbesked til ugyldige indtastninger
- Lav employeePage.jsp
- Styling af login.jsp
- Opdater databsen med employee login information

Subtasks til US-13: *As a customer I want to be able to enter my contact information so that Fog can contact me*

- Tiføj felter til kontaktinformation til pricePage.jsp
- Lav en submit-funktion, der tilføjer data til MySql
- Opdater databasen med kontaktinformation og en ordretabel

Subtasks til US-14: *As an employee I want to be able to get an overview of submitted forms so that I can contact the customers.*

- Tilføj en tabel på employeePage.jsp indeholdende data fra "Orders" i MySql

Sprint 4 (7/5-18/5)

US-15 – US-18

Subtasks til US-15: As an employee I want to be able to see the price from the chosen dimensions

- Lav pricePage.jsp
- Opdater MySQL-database med kombinerede tabeller indeholdende længde, bredde og pris for alle forskellige kombinationer
- Tilføj totalpris til pricePage.js
- Styling af pricePage.jsp

Subtasks til US-17: As a user I want the web page to notify me of missing input in the form so that I am not able to submit blank values

- Sørg for, at der ikke kan submittes blanke værdier
- Generer fejlmeddelelser

Subtasks til US-18: As an employee I want to be able to mark submitted forms as “done” in the overview so that I know which customers to contact

- Tilføj checkbox til ordreoversigt på employeePage.jsp, så en ordre kan markeres som ”done”
- Tilføj funktion til submit/opdater ændringer til databasen

Arbejdsprocessen reflekteret

Vi valgte at have en fast scrum master (Jonas) hvilket fungerede ret godt. Efter hvert sprint review holdt vi gruppemøde på sprint hvor de nye user stories blev nedbrudt i tasks og fordelt. Taiga var en god hjælp til at holde styr på nye, igangværende og færdiggjorte user stories. Det gjorde arbejdet meget overskueligt og alle var klar over hvilke opgaver de selv og andre havde.

Vores retrospektive møder var ikke lange, oftest checkede vi at listen med tasks stemte med det vi havde lavet. Da vi stort set ikke kom bagud med de planlagte tasks i forhold til reviews, var der ikke andre emner på møderne. Et par gange viste en opgave sig at være større end vi havde regnet med (f.eks. SVG-tegning). Vi nævnte dette til review og inkluderede det nødvendige ekstra arbejde i det følgende sprint.

Processen med at omsætte user stories til tasks var uden problemer. I første sprint var opgaverne indlysende, vi skulle have en struktur i projektet og var enige om hvordan det skulle gøres. Da vi derefter begyndte at lave klasser og metoder, brugte vi tasks som basis hvilket fungerede uden problemer.

Estimeringer har nok været et af vore svagere punkter. Vi fordelte opgaverne til hvert enkelt gruppemedlem så de burde kræve nogenlunde lige meget arbejde, men uden at sætte et præcist tidsestimat på hver enkelt opgave. Vore erfaringer har været følgende:

- Nye kodeområder som SVG var svære at tidsestimere
- At rette/tilpasse kode så den fungerer uden fejl og præcist som ønsket, tager nogle gange væsentligt længere tid end at skrive den originale kode
- Hver enkelt af os er stærkere/svagere på enkelte områder hvilket betyder at et estimat til dels vil afhænge af hvem der skal lave opgaven

Projektet har givet hver af os en bedre basis for at estimere indsats/tidsforbrug men det vil kræve mere erfaring og undervisning for at blive præcise.

Planning og review møderne gik ret hurtigt og vi var aldrig i tvivl om den afleverede og ny funktionalitet.

Det ville have været en stor fordel i højere grad at kunne få koden til de afleverede user stories samt nye gennemgået. Undervejs var vi ofte i tvivl om vores løsninger var teknisk gode/rigtige og havde brug for

feedback til dette. Denne udfordring blev større da den fysiske undervisning blev indstillet og adgang til tutorer og medstuderende blev mindre.

Vores rytme var meget konstant undervejs, de enkelte sprint indeholdt stort set samme arbejdsmængde og vi ”synkroniserede” vores arbejde med review/planning møderne.

Vores erfaring med scrum har været positiv, det har givet fuld klarhed over egen og andres opgaver og deadlines. At have en fast scrummaster har været den bedste løsning for os da opgaverne er blevet defineret og fordelt ”demokratisk”.

Vores største udfordring har været den meget formindskede mulighed for at få undervisning, råd/vejledning af lærere og tutorer samt ikke at kunne sparre med vores medstuderende i de andre grupper. Videoundervisningen har delvist fungeret men der har været meget lidt af den og spørgsmål til et specifikt kodeproblem i en enkelt gruppes projekt fungerer ikke i en videoundervisning for +50 elever.

Vores konklusion er at scrum har været en stor hjælp i arbejdet. Det har hjulpet med at bryde projektet ned i mindre og håndterbare opgaver uden at miste overblikket. Vores største udfordring har været at have meget begrænset mulighed for vejledning fra lærere og ingen tutorer. Vi har haft meget lidt kontakt med vores medstuderende hvilket normalt er en mulighed for at udveksle viden og erfaringer samt få feedback på kodetekniske detaljer.

Bilag 1 – Datagrundlag for materialeliste

Excel kan findes på GitHub her:

https://github.com/Goya90/Fog/blob/master/documents/Materialeliste%20beregninger.xlsx

Nedenfor er også indsat et billede, hvor indholdet dog er meget småt.

Alle tal er i mm hvis ikke andet er angivet.
l = længde, b = bredde og h = højde.

Il	Særlig beting	Tagtype	Skur (	Kategori	Type	Materiale navn	Længde pr. styk	Antal	Forklaring
1	N/A	Begge	Begge	Træ	Stern vandbræt sider	25x200 mm. trykimp. bræt	300 + carport l	2	300 mm udhæng i hver ende og et bræt på hver side.
2	N/A	Fladt	Begge	Træ	Stern vandbrædt forende	25x200 mm. trykimp. bræt	1 x carport b	1	Kun et bræt foran i samme bredde som skuret.
3	Carport l < 4800	Begge	Uden	Træ	Stolpe	97x97 mm. trykimp. stolpe	900 + carport h	4	4 stolper pr. skur. 900 mm nedgravet pr. stolpe.
3	Carport l > 4800	Begge	Uden	Træ	Stolpe	97x97 mm. trykimp. stolpe	900 + carport h	6	6 stolper pr. skur. 900 mm nedgravet pr. stolpe.
3	N/A	Begge	Med	Træ	Stolpe	97x97mm. trykimp. stolpe	900 + carport h	8	8 stolper pr. skur. 900 mm nedgravet pr. stolpe.
4	N/A	Begge	Med	Træ	Beklædningsplader skur	19x100 mm. trykimp. bræt	(1 x carport h) - 200	((skur l / 60) x 2) + ((skur b / 60) x 2))	Plader monteres 1 på 2 med 20 mm overlap i hver side.
5	N/A	Begge	Med	Træ	z til bagside af dør i skur	38x73 mm. lægte ubh.	420	1	1 stk. pr. skur uanset hvad.
6	N/A	Begge	Begge	Træ	Remme i sider	45x195 mm. spærtræ ubh.	1 x carport l	2	2 remme uanset hvad, og i samme længde som carporten.
7	N/A	Fladt	Begge	Træ	Spær	45x195 mm. spærtræ ubh.	1 x carport b	carport l / 550	Spær monteres med 550 mm mellemrum hvis taget er fladt
8	N/A	Begge	Med	Træ	Løsholter til skur sider	45x95 mm. reglar ubh.	1 x skur l	2	Løsholter er dem som belægningspladerne til skuret skrues ind i i bunden
9	N/A	Begge	Med	Træ	Løsholter til skur gavle	45x95 mm. reglar ubh.	1 x skur b	2	Løsholter er dem som belægningspladerne til skuret skrues ind i i bunden
10	N/A	Rejsning	Begge	Træ	Spær	Færdigsamlet spær	0	carport l / 1000	Spær monteres med 1000 mm mellemrum hvis tag har rejsning
11	N/A	Rejsning	Begge	Træ	Vindskeder på rejsning	25x150 mm. trykimp. bræt	1,4 x carport b	2	Taget antages samme højde/hældning uanset hvad. Så eneste faktor er bredden.
12	N/A	Rejsning	Begge	Træ	38x73 mm. Taglægte T1	Til montering på spær	540	carport l / 1000	5 stk. pr. spær. 1 spær pr. meter carport i længden.
13	N/A	Rejsning	Begge	Træ	38x73 mm. Taglægte T1	Til montering af rygsten	1 x carport l	1	Samme længde som carporten
14	N/A	Fladt	Begge	Tagpakken	Tagplade	Plastmo blåtonet 1200x1200 mm.	N/A	((carport l + 600) x (b+600)) / 1000000)	Plader overlapper med 200 mm. så forbrug = 1 plade per m2. Derudover er der 300 mm. udhæng på alle sider
15	N/A	Rejsning	Begge	Tagpakken	Tagsten	Tagsten dobbelt -s sort	N/A	((carport l x b) / 1000000) * 14	Carportens længde gange bredde i mm / 1000000 = carport m2. Der bruges 14 tagsten pr. m2.
16	N/A	Rejsning	Begge	Tagpakken	Rygsten	Rygsten sort	N/A	carport l / 400	Rygstenen er 420 mm og skal overlappe med 10 mm. Derfor skal der bruges 2,5 rygsten pr. 1000 mm carport i længden
17	N/A	Rejsning	Begge	Tagpakken	Rygstensbeslag	Rygstensbeslag	N/A	carport l / 400	Der skal bruges 1 beslag pr. rygsten
18	Tag = Plastmo	Fladt	Begge	Tagpakken	Skruer til tagplader	Plastmo bundskruer 200 stk.	N/A	(carport l x b) / 1000000 / 15	Der skal bruges 1 pakke pr. 15 m2 tag
19	N/A	Rejsning	Begge	Tagpakken	Monteres på toppen af spæret	Toplægte holder	N/A	carport l / 1000	1 holder pr. meter
20	N/A	Rejsning	Begge	Tagpakken	Montering af tagsten	1 pk. tagstens bindere og nakkekroge	N/A	(carport l x b) / 1000000 / 15	Leveres i pakker uden nærmere specificeret antal. Antaget forbrug er 1 pk pr. 15 m2 carport
21	N/A	Rejsning	Begge	Tagpakken	Til taglægter	5,0 x 100 mm. Skruer 100 stk.	N/A	(carport l x b) / 1000000 / 15	Antaget forbrug er 1 pk pr. 15 m2 carport
22	N/A	Fladt	Begge	Beslag & skruer	Vindkryds på spær	Hulbånd 1x20 mm. 10 mtr.	N/A	(carport l x b) / 1000000 / 25	Der skal bruges 1 stk. pr. 25 m2 carport
23	N/A	Begge	Begge	Beslag & skruer	Beslag til spær på rem	Universal 190 mm højre	N/A	carport l / 500	Forbrug er 2 pr. meter carport længde
24	N/A	Begge	Begge	Beslag & skruer	Beslag til spær på rem	Universal 190 mm venstre	N/A	carport l / 500	Forbrug er 2 pr. meter carport længde
25	N/A	Fladt	Begge	Beslag & skruer	Skruer til universalbeslag + hulbånd	4,0 x 50 mm. beslagskruer 250 stk.	N/A	((((carport l / 550) x 2) x 4) + (((carport l x b) / 1000000 / 25) x 100)	1 stk. spær pr. 55 mm l carport. 2 stk. beslag pr. spær. 4 skruer pr. beslag. Derudover 100 skruer pr. hulbånd.
26	Carport l < 4800	Begge	Uden	Beslag & skruer	Montering rem på stolper	Bræddebolt 10x120 mm.	N/A	8	Forbrug er 2 stk. pr. stolpe
26	Carport l > 4800	Begge	Uden	Beslag & skruer	Montering rem på stolper	Bræddebolt 10x120 mm.	N/A	12	Forbrug er 2 stk. pr. stolpe
26	N/A	Begge	Med	Beslag & skruer	Montering rem på stolper	Bræddebolt 10x120 mm.	N/A	16	Forbrug er 2 stk. pr. stolpe
27	Carport l < 4800	Begge	Uden	Beslag & skruer	Montering rem på stolper	Firkantskiver 40x40x11 mm.	N/A	8	Forbrug er 2 stk. pr. stolpe
27	Carport l > 4800	Begge	Uden	Beslag & skruer	Montering rem på stolper	Firkantskiver 40x40x11 mm.	N/A	12	Forbrug er 2 stk. pr. stolpe
27	N/A	Begge	Med	Beslag & skruer	Montering rem på stolper	Firkantskiver 40x40x11 mm.	N/A	16	Forbrug er 2 stk. pr. stolpe
28	N/A	Begge	Begge	Beslag & skruer	Montering af stern, vindskeder og vandbræt	4,5x60 mm. skruer 200 stk.	N/A	(carport l x b) / 1000000 / 25	1 pakke skruer pr. 25 m2 carport
29	N/A	Begge	Med	Beslag & skruer	Til lås på dør i skur	Stalddørsgreb 50x75 mm.	N/A	1	1 stk. pr. skur uanset hvad.
30	N/A	Begge	Med	Beslag & skruer	Til skur	T hængsel 390 mm.	N/A	1	1 stk. pr. skur uanset hvad.
31	N/A	Begge	Med	Beslag & skruer	Til montering af løsholter i skur	Vinkelbeslag 35	N/A	((skur l x b) / 1000000) + 4	Minimum 4 stk. og yderligere 1 pr. m2 skur
32	N/A	Begge	Med	Beslag & skruer	Til montering af yderste beklædning	4,5 x 70 mm. skruer 400 stk.	N/A	((skur l / 60) x 2) + ((skur b / 60) x 2)) / 2	Antal beklædningsbrædder / 2 da kun hver andet bræt er yderst.
33	N/A	Begge	Med	Beslag & skruer	Til montering af inderste beklædning	4,5 x 50 mm. skruer 300 stk.	N/A	(((skur l / 60) x 2) + ((skur b / 60) x 2)) / 2) x 2	Antal beklædningsbrædder / 2 da kun hver andet bræt er inderst men x 2, da der skal to skruer pr. inderste beklædningsbræt
34	N/A	Fladt	Begge	Tagpakken	Tagplade	Plastmo sort 1200x1200 mm.	N/A	((carport l + 600) x (b+600)) / 1000000)	Plader overlapper med 200 mm. så forbrug = 1 plade per m2. Derudover er der 300 mm. udhæng på alle sider
35	N/A	Fladt	Begge	Tagpakken	Tagplade	Plastmo gennemsigtig 1200x1200 mm.	N/A	((carport l + 600) x (b+600)) / 1000000)	Plader overlapper med 200 mm. så forbrug = 1 plade per m2. Derudover er der 300 mm. udhæng på alle sider
36	N/A	Fladt	Begge	Tagpakken	Tagplade	Plastmo hvid 1200x1200 mm.	N/A	((carport l + 600) x (b+600)) / 1000000)	Plader overlapper med 200 mm. så forbrug = 1 plade per m2. Derudover er der 300 mm. udhæng på alle sider
37	N/A	Rejsning	Begge	Tagpakken	Tagpap	Tagpap sort	N/A	((carport l x b) / 1000000) * 14	Carportens længde gange bredde i mm / 1000000 = carport m2. Der bruges 14 tagsten pr. m2.
38	N/A	Rejsning	Begge	Tagpakken	Tagplade	Trapez plast sort	N/A	((carport l x b) / 1000000) * 14	Carportens længde gange bredde i mm / 1000000 = carport m2. Der bruges 14 tagsten pr. m2.

Bilag 2 – PlantUML kode for navigationsdiagram

```
@startuml

[*] --> Frontpage

state Frontpage {
  index.jsp --> slantedRoof.jsp : Change roof type
  slantedRoof.jsp --> index.jsp
}

Frontpage --> Confirmation : Submit request

Confirmation --> Frontpage : Return
```

```
state Confirmation {
  confirmation.jsp : SERVLET USED: \nRequest
}

Frontpage --> Login : Link to login
Login --> Frontpage : Return

state Login {
  login.jsp : SERVLET USED: \nLogin
}

Login --> Adminpage : Correct credentials
Login --> Login : Incorrect credentials
Adminpage --> Login : Log off

state Adminpage {
  adminpage.jsp :
}

Adminpage --> DoneRequests : Show processed requests
DoneRequests --> Adminpage : Return

state DoneRequests {
  doneRequests.jsp : SERVLET USED: \nDoneRequests
}

Adminpage --> NewRequests : Show new requests
NewRequests --> Adminpage : Return

state NewRequests {
```

```
  showRequests.jsp : SERVLET USED: \nShowRequests
}

NewRequests --> BillOfMaterials : Show material list
BillOfMaterials --> NewRequests : Return

state BillOfMaterials {
  billofmaterials.jsp : SERVLET USED: \nBillOfMaterials
}

BillOfMaterials --> UpdatePrice : Update price
UpdatePrice --> BillOfMaterials : Return

state UpdatePrice {
  requestConfirmation.jsp : SERVLET USED: \nProcessRequest
}

BillOfMaterials --> ShowDrawing : Show drawing
ShowDrawing --> BillOfMaterials : Return

state ShowDrawing {
  drawing.jsp : SERVLET USED: \nDrawing
}

BillOfMaterials --> MarkAsDone : Process order
MarkAsDone --> Adminpage : Order processed

state MarkAsDone {
  NONE : SERVLET USED: \nMarkAsDone
}
```

```
}
@enduml
```

Bilag 3 – PlantUML kode for sekvensdiagram “New request”

```
@startuml

title Sequence diagram "New request"

actor User
boundary index.jsp
boundary confirmation.jsp
participant FrontController
participant Command
participant Request
participant LogicFacade
participant UserMapper
database Database

User -> index.jsp : request
index.jsp -> FrontController : doPost()
FrontController -> Command : processRequest()
Command -> Request :
Request -> LogicFacade : createRequest()
LogicFacade -> UserMapper : createRequest()
UserMapper -> Database : executeUpdate()
Database -> UserMapper : resultset
UserMapper -> LogicFacade : OK
LogicFacade -> Request
```

Request -> confirmation.jsp

confirmation.jsp -> User : Confirmation

UserMapper -> LogicFacade : Not OK (alternative flow)

LogicFacade -> Request

Request -> confirmation.jsp

confirmation.jsp -> User : Please try again!

@enduml

Bilag 4 – PlantUML kode for sekvensdiagram "Show requests"

@startuml

title Sequence diagram "Show requests"

actor User

boundary showRequests.jsp

boundary adminpage.jsp

participant FrontController

participant Command

participant ShowRequests

participant LogicFacade

participant RequestMapper

database Database

User -> adminpage.jsp : request

adminpage.jsp -> FrontController : doPost()

FrontController -> Command : showRequests()

Command -> ShowRequests : showRequests()

ShowRequests -> LogicFacade : showRequest()

LogicFacade -> RequestMapper : showRequest()

RequestMapper -> Database : executeUpdate()

Database -> RequestMapper : resultset

RequestMapper -> LogicFacade :

LogicFacade -> ShowRequests :

ShowRequests-> showRequests.jsp :

showRequests.jsp -> User : List of requests

ShowRequests -> showRequests.jsp : Failure (alternative flow)

showRequests.jsp -> User : Please try again!

@enduml