

Fog Carporte - Eksamens projekt

**Deltagere:**

Rasmus Højholt

Lukas Kjellerup Simonsen

Cph-Mail

cph-rs255@cphbusiness.dk

cph-ls491@cphbusiness.dk

Github-brugernavn

RHoejholt

Montif16

Projekt start til slut:

Dato: 16-11-2024 - 19-12-2024

Indholdfortegnelse

Indholdfortegnelse	2
Links	3
Indledning	3
Baggrund	4
Interessentanalyse for Fog Carport-projektet	4
Risikovurdering	6
Forretningsforståelse bidragelse under projektet	6
Teknologi valg	7
Development Tools	7
Application Development	7
Database Management	7
Server Environment	7
Krav	8
User stories	13
Domæne model og ER-diagram	16
Database	17
Navigationsdiagram & Mockups	21
Valg af arkitektur	23
Oversigt over pakker og klasser i systemet	23
Brug af hjælpeklasser og designmønstre	25
Særlige forhold	25
Sessions, request og variabler	25
Håndtering af exceptions	26
Validering af brugerinput	26
Sikkerhed med login og opbevaring af sensitiv data	27
Dynamisk metode til hentning af data:getDimensionOptions	27
Status på implementering	29
Fremtidig forbedring: Dynamisk validering af databaseforespørgsler	29
Email	30
Kvalitetssikring (test)	30
User acceptance tests	31
Proces	34
Arbejdsprocessen faktisk	34
Arbejdsprocessen reflekteret	36
Tak til dig som læser	38
ER-Diagram v.1.0	42
ER-Diagram v-2.0	42
getDimensionOptions	43
User Stories	45

Links

Repository:

<https://github.com/RHoejholt/Fog-Projektet?tab=readme-ov-file>

Github projects:

<https://github.com/users/RHoejholt/projects/3>

Demo video:

<https://www.youtube.com/watch?v=vrsjNMsQNKw>

Deployed hjemmeside:

<http://164.90.237.79:3484/>

(Admin login:

Username: admin

Password: admin)

Indledning

Dette projekt omhandler udviklingen af et system til bestilling af skræddersyede carporte, som Fog har bestilt. Formålet er at give brugeren mulighed for at tilpasse deres carport efter specifikke behov og derefter generere en styklister over de materialer, som carporten skal bruge. Systemet henvender sig primært til sælgere, som skal kunne oprette og administrere ordrer baseret på kundens ønsker.

Projektets overordnede mål er at demonstrere evnen til at udvikle en fuld-stack applikation, der integrerer backend-funktionalitet med en brugervenlig frontend.

Rapporten beskriver udviklingsprocessen, teknologivalg, og løsninger på de udfordringer, der er opstået undervejs. Forudsat at læseren har en grundlæggende forståelse af programmering og softwareudvikling, vil rapporten give en klar forståelse af, hvordan systemet er designet og implementeret.

Baggrund

Fog er et dansk byggefirma, som bl.a. tilbyder skræddersyede carporte.

Krav fra Fog:

Kunden skal kunne tilgå hjemmesiden via internettet, oprette en bruger, logge ind, vælge specifikationer til sin skræddersyede carport (tagtype, tagmateriale, bredde og længde på carport, materiale for spær og rem, og kunne fravælge et skur).

De valg vil blive sendt til en repræsentant fra Fog, som vil kontakte kunden med et tilbud. Når dette tilbud er accepteret og betalt, vil kunden se en stykliste med oversigt over alle materialer til deres bestilte carport.

Interessentanalyse for Fog Carport-projektet

Her er et generelt Interessentdiagram. De højeste indflydelse interessenter er blevet analyseret nærmere under modellen. Modellen blev oprettet før projektstart

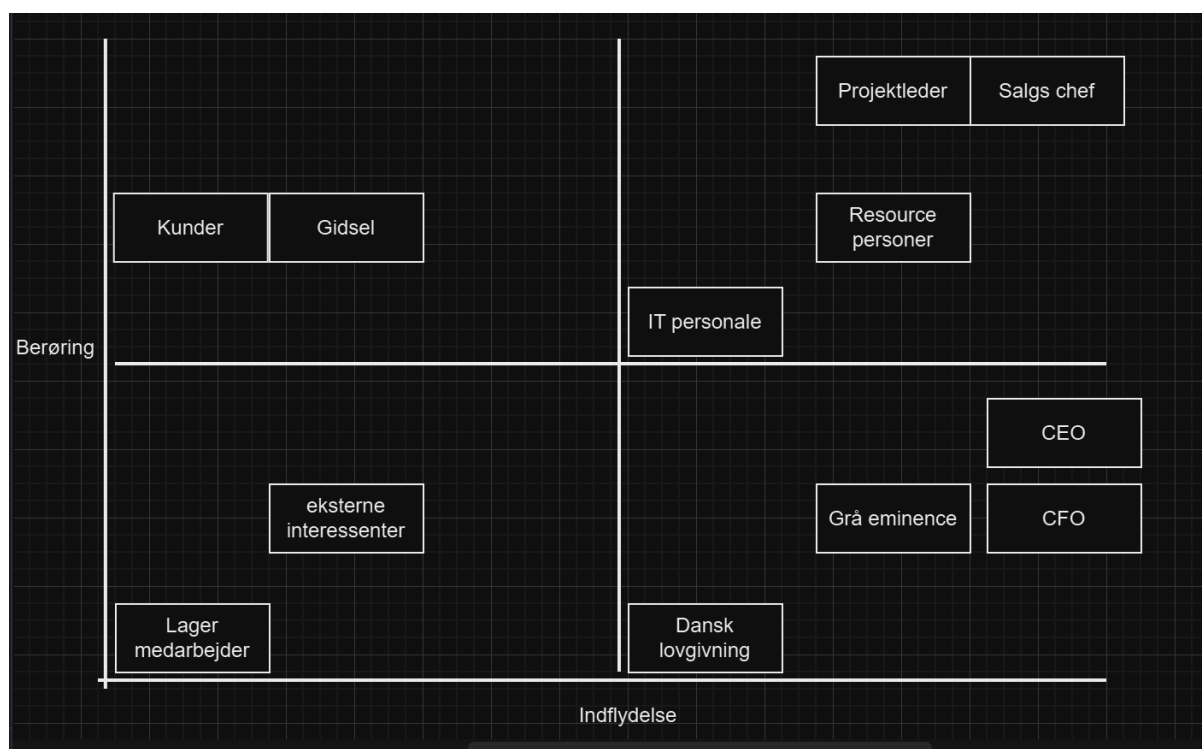


Fig 1. Interessantdiagram

Interessent	Rolle	Interesse	Indflydelse	Risiko	Håndtering af risiko
Kunder (Privatkunder)	Køber carporte, modtager produktet	Ønsker fleksible designmuligheder , høj kvalitet og hurtig levering	9/10	Utilfredshed med design eller leveringstid kan føre til tab af salg.	Klart definerede produktmuligheder og leveringstider samt en brugervenlig platform.
Sælgere (Fog medarbejdere)	Ansvarlige for at rådgive kunder og oprette ordrer i systemet	Ønsker et intuitivt værktøj til at oprette ordrer og generere tilbud	8/10	Komplicerede værktøjer kan forsinke salgsprocessen og skabe frustration.	Brugervenlig grænseflade og uddannelse for sælgere i brug af systemet.
Udviklingsteam	Ansvarlige for udvikling og vedligeholdelse af platformen	Ønsker klare krav og stabil infrastruktur	7/10	Manglende kravspecifikation er kan føre til forsinkelser eller fejl i systemet.	Løbende dialog og klare kravspecifikationer fra projektledelsen.

Ledelsen (Fog A/S)	Overordnet ansvarlige for budget og projektets succes	Ønsker en omkostningseffektiv løsning, der øger salget	10/10	Overskridelse af budgettet kan føre til projekt stop eller reducerede investeringer.	Løbende rapportering om økonomi og fremskridt samt tydelig prioritering af funktioner.
Leverandører	Leverer materialer til carports	Ønsker klare bestillinger og betalingsbetingelser	6/10	Forsinkelser kan påvirke leveringstid	Oprettelse af bufferlager

Risikovurdering

Risiko	Sandsynlighed	Alvor	Risiko Niveau	Plan for forebyggelse/afværgning
Systemet oplever tekniske fejl ved høj belastning. Virker ikke ordentligt.	7/10	9/10	63/100	Belastningstest systemet, før det tages i brug, og implementér integrations- og unit tests.
Kunder og sælgere føler, at siden ikke er brugervenlig.	8/10	4/10	32/100	Tilføj flere tilpasningsmuligheder og sørg for klar kommunikation på platformen.
Projektet overskrider budgettet på grund af uforudsete udviklingsomkostninger.	7/10	9/10	64/100	Stram budgetstyring og løbende opdatering af estimater.

Forretningsforståelse bidragelse under projektet

I udviklingen af Fog Carport-projektet fokuserede vi primært på at nå frem til et Minimum Viable Product, der kunne levere grundlæggende funktionalitet og tilfredsstille kernebehovene til projektet. Vi har holdt brugervenlighed og god teknisk opsættelse i baghovedet under processen, men vores fokus var primært på at lave en funktionel hjemmeside.

Teknologi valg

Development Tools

Interface Mockup Design Tool: Figma Desktop App Version 124.5.5

IDE: IntelliJ IDEA 2023.3.4

Version control & Code sharing: Git & github

Diagram/UML Tools: Draw.io, PlantUML Integration for IntelliJ 7.11.2-IJ2023.2

Containerization Tool: Docker Desktop 4.34.3

Application Development

Framework: Javalin 6.1.3

Template Engine: Thymeleaf 3.1.2 (Release), Thymeleaf-extras 3.0.4 (Release)

Frontend Technologies: CSS3, HTML5

Build Tool: Maven (compiler version 3.13.0)

Encoding Standard: UTF-8

JDK: Amazon Corretto 17 (version 17.0.12)

Unit Testing Framework: JUnit 5.10.2

Database Management

Database Management System: PostgreSQL 42.7.2

Database Administration Tool: PGAdmin 4 (version 8.3)

JDBC Driver: JDBC 4.2

Connection Pool Library: HikariCP 5.1.0

Server Environment

Operating System: Ubuntu 22.04 (Linux Kernel 5.15.0-125-generic)

Container Runtime: Docker 25.0.3

Database Version: PostgreSQL 17.2

Hosting Service: DigitalOcean Droplet (1 GB Memory / 25 GB Disk)

Web server: Caddy

Krav

Denne sektion beskriver de forskellige krav som Fog har sat for deres nye system. Denne sektion er baseret på en kundeanalyse, som blev skabt baseret på et møde med kunden. I mødet med Fog og gennem dialogen om deres nuværende arbejdsproces har vi identificeret en række vigtige krav, som vores program skal kunne imødekomme. Disse krav udspringer både af deres nuværende system og af deres ønsker til, hvordan deres arbejdsgange kan optimeres. Her er en gennemgang af de vigtigste punkter, der danner grundlag for vores design og udvikling af vores system.

Tegninger og specifikationer

Fog ønsker, at deres kunder skal kunne hente tegninger på både enkelte og dobbelte carporte. Tegningerne skal inkludere beskrivelser, specifikationer og leveringsdetaljer for alle standard carporte. Dette krav afspejler Fogs ønske om at give kunderne en nem og visuel måde at forstå produktet på.

Specialmål og emailhåndtering

Ved bestilling af carpote med specialmål ønsker Fog, at deres system automatisk skal sende en email til deres sælgere med information om ordren. I deres nuværende system indtaster de oplysningerne manuelt fra emailen i et andet program. Derfor har Fog ønsket et system, der kan modtage oplysningerne automatisk og indsætte dem i databasen uden manuel indtastning.

Redskabsrum og beklædning

Fog ønsker, at kunderne kan vælge, om de vil tilføje et redskabsrum med specifikke mål, eller om de vil fravælge det helt. Derudover skal kunderne kunne vælge blandt alle beklædningstyper, som Fog tilbyder. Fog har understreget, at deres nuværende system ikke indeholder alle valgmuligheder, hvilket skaber begrænsninger. Derfor ønsker de et system, der automatisk kan opdatere beklædningstyper, lagerstatus og poser for at minimere manuel vedligeholdelse.

Automatisk vedligeholdelse

Fog ønsker et system, der kan integreres med deres lagerdatabase og opdatere priser, lagerstatus og produktdata automatisk. Dette skal reducere manuel opdatering og sikre, at deres system altid er opdateret med de nyeste data. I deres nuværende system, kan de ikke nemt ændre i deres data omkring produkter.

Sælgerens rolle

Fog lægger vægt på, at sælgeren fortsat skal være en central del af processen mellem kunden og deres sælgere. Systemet skal derfor støtte sælgeren i at have fleksibilitet og overblik over alle detaljer i processen.

Begrænsninger i det nuværende system

I det eksisterende system kan sælgeren kun rette priser, men ikke tilføje nye varer, oprette nye produkter eller ændre eksisterende mål på deres tømmer. Fog ønsker et nyt system, der både kan hente data fra databasen og give sælgeren et brugervenligt interface til at administrere data direkte.

Kriterier for styklisten

Styklister og guides må først sende til kunden, når betalingen er gennemført. Dette krav sikrer, at værdifulde oplysninger ressourcer ikke deles uden økonomisk sikkerhed for købet.

Med udgangspunkt i disse krav er vores system designet til at imødekomme Fogs behov og løse nogle af de udfordringer, de har med deres nuværende løsning. Kravene er omdannet til konkrete funktioner, metoder og user stories, som understøtter både kunder og sælgerens behov. Ikke alle krav er blevet fuldt ud implementeret pga. tidspress, men fremtidige opdateringer til programmet vil inkludere disse krav til produktet.

AS-IS Diagram

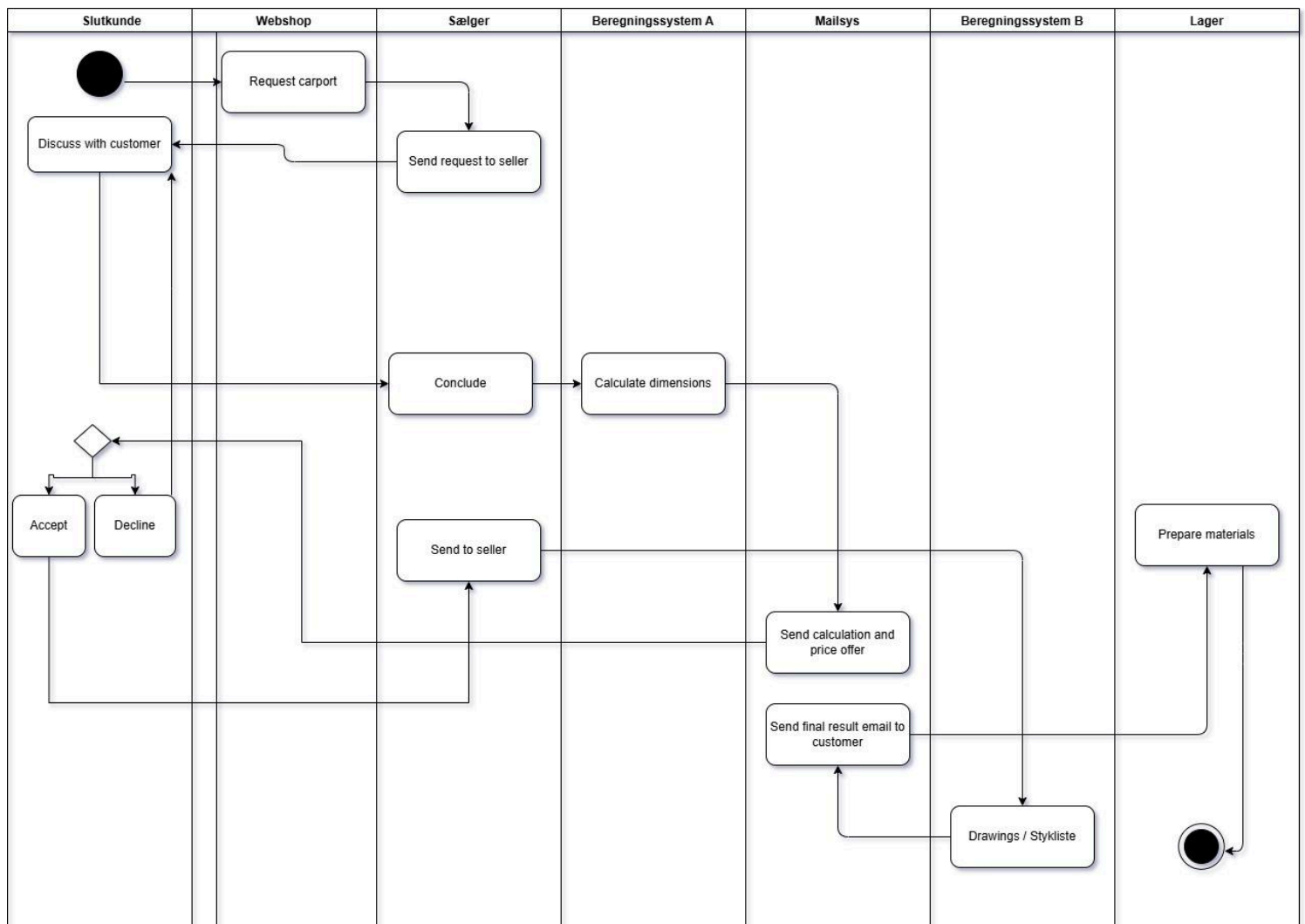


Fig 2. AS-IS diagram

AS-IS-diagrammet beskriver den nuværende proces for carport ordrer hos Fog, som den blev præsenteret under mødet med Martin. Den starter med, at kunden anmoder om en carport via webshoppen. Webshoppen sender herefter anmodningen videre til en sælger, som vurderer kundens ønsker og påbegynder processen. Sælgeren benytter et beregningssystem til at beregne de nødvendige beregninger for carportens dimensioner. Når beregningerne er afsluttet, konkluderer sælgeren og sender data videre til andre systemer, herunder mailsystemet, der sørger for, at kunden modtager det endelige tilbud, inklusive pris og eventuelle tegninger eller styklister. Styklister og materialelister skabes også i samarbejde med et andet beregningssystem, som sørger for, at lageret kan forberede de nødvendige materialer til carporten. Processen afsluttes, når materialerne er klar, og kunden har truffet en beslutning om at acceptere eller afvise tilbuddet.

TO-BE Diagram

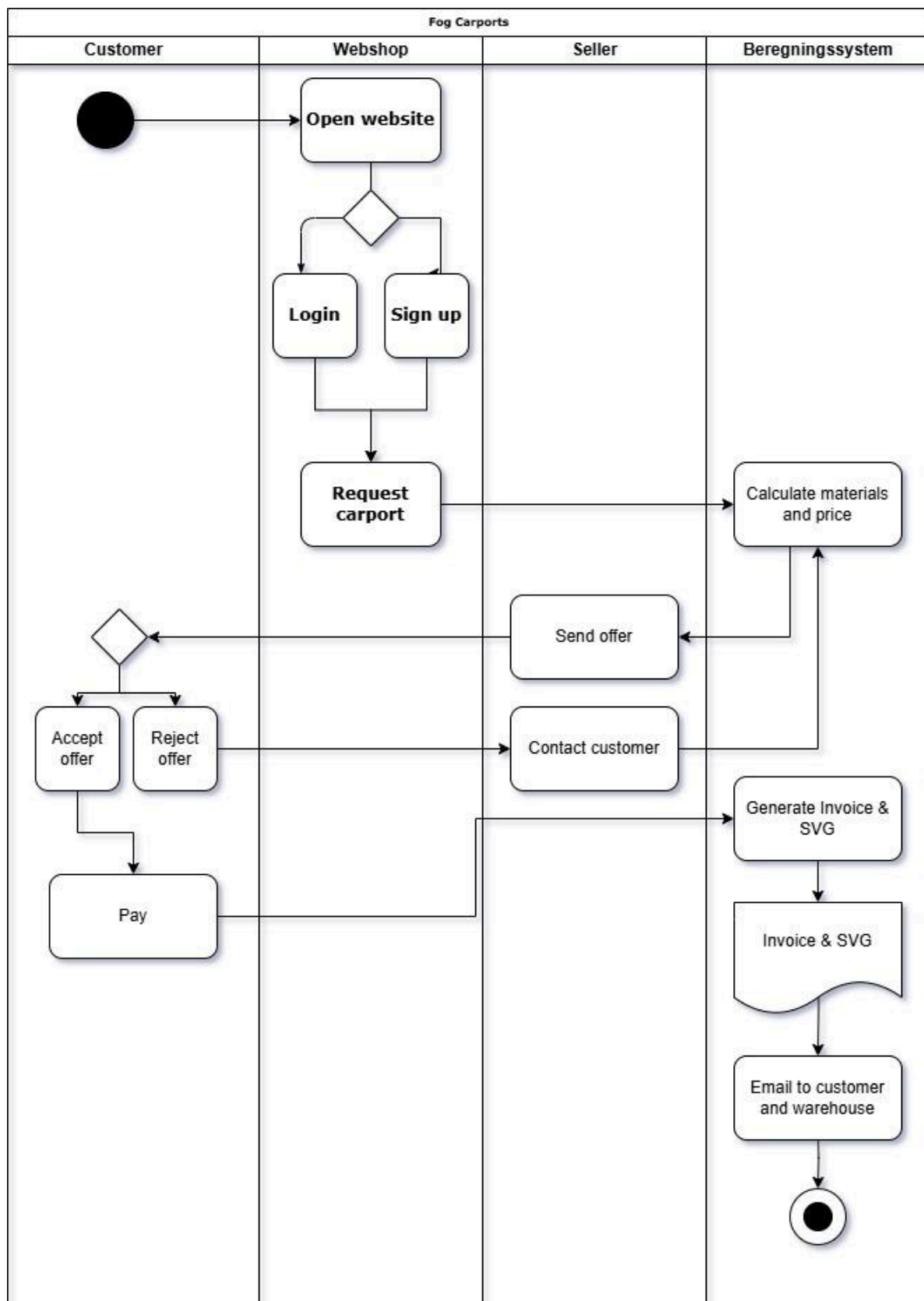


Fig 3. TO-BE diagram

Dette TO-BE-diagram viser, hvordan vi havde tænkt os at håndtere fremtidige carport ordrer via det nye system. Kunden starter med at åbne websitet, hvor de enten logger ind eller opretter en konto. Derefter kan de anmode om en carport, hvilket sender en forespørgsel videre til sælgeren. Sælgeren bruger beregningssystemet til at udregne materialer og pris, hvorefter et tilbud bliver sendt til kunden. Hvis kunden afslår tilbuddet, kan sælgeren kontakte kunden og sende et nyt tilbud. Når tilbuddet er accepteret, betaler kunden, og så vil en faktura og tegninger blive genereret og sendt til både kunden og lageret via e-mail.

Aktivitetsdiagram (af den leverede app/hjemmeside)

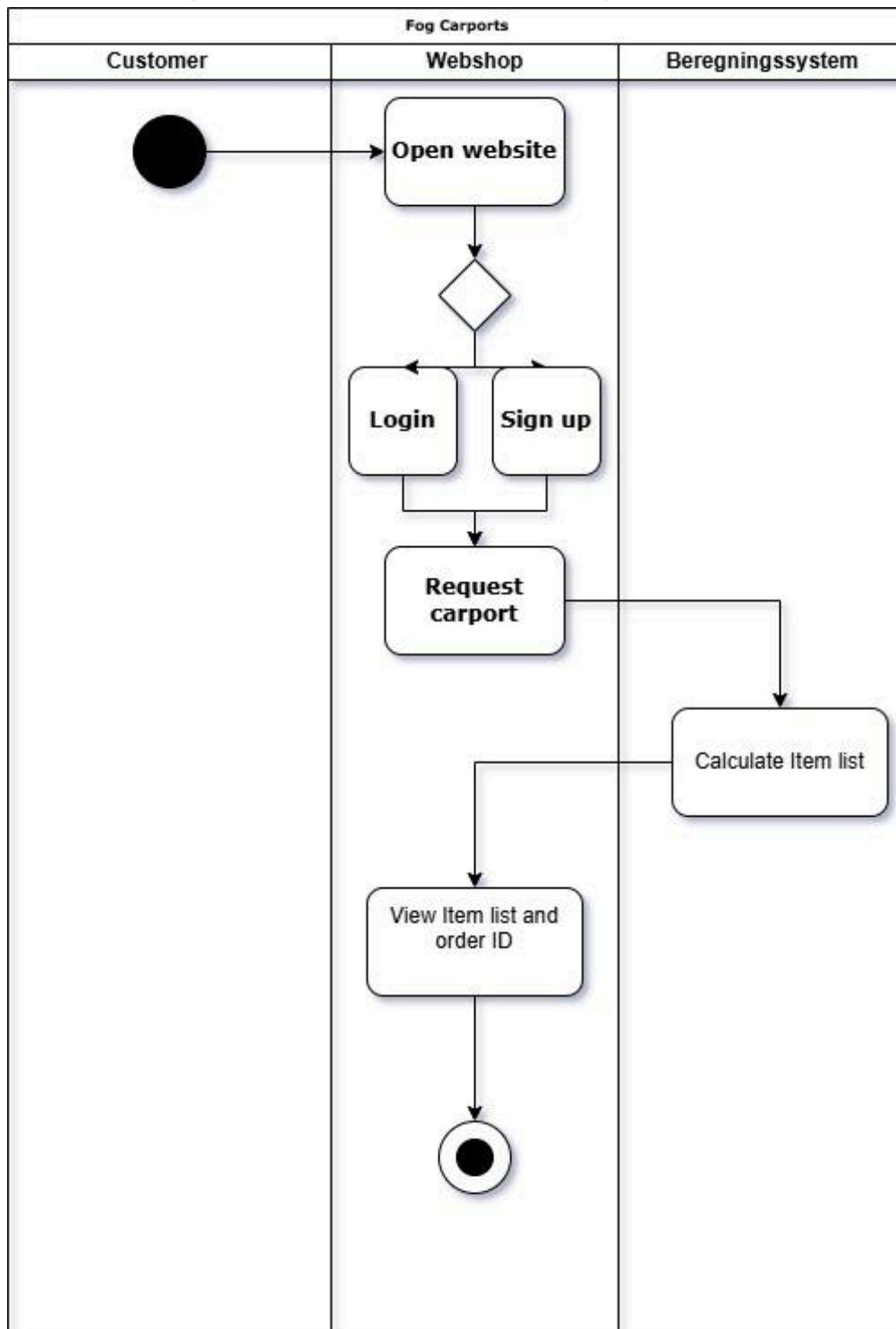


Fig. 4. Aktivitetsdiagram

Dette aktivitetsdiagram viser den løsning, vi faktisk leverede, efter at vi måtte descope projektet på grund af tidspres. Kunden kan stadig starte processen ved at åbne hjemmesiden og enten logge ind eller oprette en konto. Derefter kan de anmode om en carport. Systemet beregnede en stykliste baseret på kundens specifikationer, og kunden kan herefter se stykliste og ordrenummeret. Den mere komplekse funktionalitet som tilbudsgenerering, kontakt med sælger og betalingssystem blev ikke implementeret i denne version, men løsningen fungerer forhåbentlig som et tilfredsstillende minimum viable product for Fog.

User stories

[En fuld liste over user stories kan findes i vores bilag \(klik her\)](#)

Det er svært for os præcist at udpege enkelte user stories, da de alle er relaterede til de ovenstående aktivitetsdiagrammer, især da de alle indgår i Martin's ønsker om projektet. Her er der udvalgt et par user stories som repræsenterer kernen i projektet.

US-02 Login

User story

Som kunde eller administrator

Kan jeg logge på systemet med email og kodeord.

Når jeg er logget på, så skal jeg kunne se de muligheder jeg har udfra min rolle som kunde eller administrator

Acceptance criteria

Givet at jeg ikke er logget ind, så kan jeg logge ind på en oprettet bruger.

Når jeg indtaster en eksisterende bruger, som er oprettet i systemet.

Så kan jeg blive logget ind på hjemmesiden

Tasks

Der skal laves en html-side med en loginform, som sender et post-request til /login

Controlleren skal bruge dataen fra post-requested og sende det videre til login metoden i UserMapper.

Der skal returneres et user objekt hvis login er succesfuldt og det user objekt skal gemmes til en session variable

ctx.sessionAttribute("currentuser",user) så skal man rediregeres til front pages hvor man nu kan se sine muligheder ud fra sin rolle som kunde eller administrator.

Der skal vises en fejlmeddelelse via ctx.attribute("message", "Fejlmeddelelse") hvis ens login fejler.

US-03 - US-06 Carportens tilpasningsmuligheder

Denne user story består af US 3 til 6, som er kogt ned til én user story for denne del af rapporten, da de samlet udgør en central del af projektet. De faktiske user stories kan ses i bilagene.

User Story

Som bruger eller admin, vil jeg kunne vælge hele carporten tilpasningsmuligheder, så kunden kan få deres ønskede carport

Acceptance criteria

Når man som bruger eller admin

når til punktet hvor man kan vælge skræddersyede muligheder for carporten

skal man kunne klart vælge ens ønskede mål, materialer og opsætning.

Tasks

En html side skal findes, med alle justeringsmuligheder for carporten. Mulighederne skal hentes fra en database.

Valgene skal gemmes så de kan tilgås senere, så bl.a. styklisten kan bruge dem.

US-08 Stykliste

User story

Som admin

vil jeg kunne downloade eller få vist en detaljeret stykliste

så kunden har alle nødvendige komponenter til at bygge carporten.

Acceptance criteria

Givet at alle specifikationer er indtastet korrekt og kunden har betalt

og jeg går til sidste trin i processen ved at trykke f.eks. "beregne stykliste"

bliver der vist en stykliste, som evt. kan downloades

Tasks

Der skal laves en HTML fil som kan vise en liste over alle nødvendige deles navne, antal og pris.

Den skal generere det ud fra en ctx der er givet med, så den kan finde de korresponderende data fra databasen og vise dem fra listen, gennem en mapper klasse der kan oversætte databasen til java objekter, som siden hen derefter kan bearbejde

Brugernavn, dato og totalpris kunne også indgå på siden

Domæne model og ER-diagram

Domænemodellens relationer og indhold

Kunde og Order

- En kunde kan afgive én eller flere ordrer.
- Hver ordre indeholder detaljer om carportens dimensioner og tilpasningsmuligheder, såsom tagmateriale og skur.

Order og Sælger

- En sælger modtager ordrer fra kunder.
- Dette sikrer, at hver ordre kan behandles og videreformidles til den ansvarlige medarbejder.

Order og Stykliste

- En ordre genererer én eller flere styklister, som indeholder en oversigt over de nødvendige materialer og komponenter til carporten.
- Styklisterne inkluderer også generelle informationer som totalpris og en visuel tegning af carporten (SVG).

Stykliste og Produkt

- En stykliste består af flere produkter, såsom bjælker, skruer eller tagmaterialer.
- Produkterne indeholder informationer om navn, måleenhed og pris, som danner grundlaget for stykkens totalpris.

Fog Carports Domænemodel

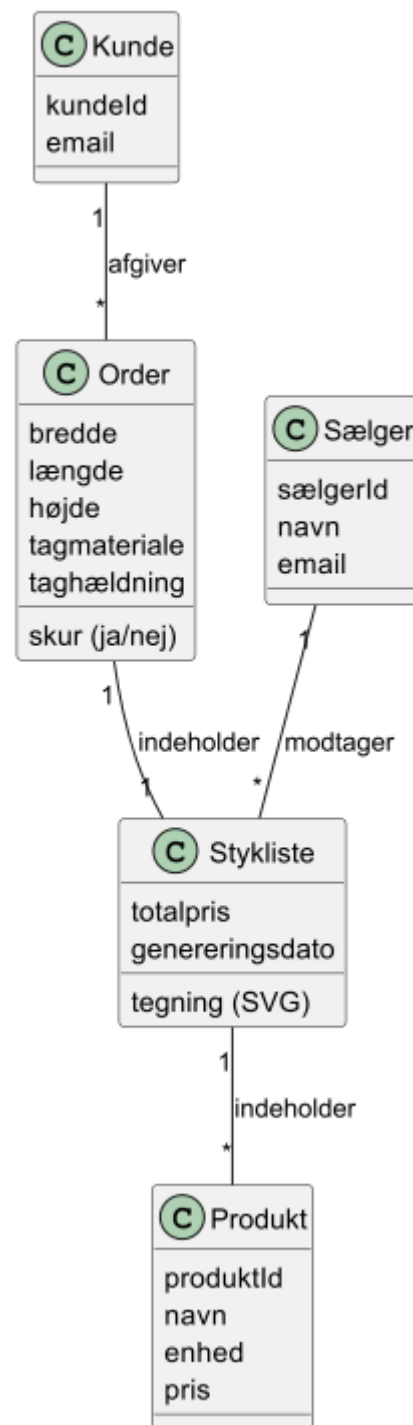


Fig 5. Domænemodel

Database

Intro

I denne sektion af rapporten, skriver vi lidt om databasen og de valg vi har gjort os i forhold til dens struktur. Der vil blive skrevet om ER-Diagrammer og datastrukturer, normalformer osv.

Læs denne sektion hvis du gerne ville vide mere om vores database i projektet.

Alle nøgleord som er kode relevante er highlighted med [cornflower blue](#) for at øge denne sektions læsbarhed. Dette kunne f.eks. være [user_id](#) eller [order_id](#).

ER-Diagram

Den første version af vores ER-diagram så således ud [ER-Diagram v.1.0](#).

Vi havde allerede en idé om, hvordan vores database skulle se ud fra starten af. Selvom vi ikke fuldt ud vidste, hvor meget der skulle i vores database, så var der 2 ting som i hvert fald skulle med. Vi ville have users og orders. Vi var sikre på, at vi gerne ville have disse 2 fra starten af. Der blev nemlig besluttet tidligt i vores proces, at man skulle være en bruger på platformen, før man kunne bestille en carport. Dette er gjort fordi der var et ønske fra Fog om, at de gerne ville være i løbende kontakt med deres kunder, som var interesserede i carporte. Derfor har vi valgt at kunder skal have en bruger på platformen før et køb kan ske. En bruger på platformen skal også angive nogle kontaklinformationer som Fog kan bruge til at kontakte deres kunder. Når en bruger opretter sig på hjemmesiden, er de påtvunget at gøre det med en email adresse. Dette valg giver Fog en mulighed for altid at have en kontaktmulighed for dem som bestiller en carport.

ER-Diagram v2.0.

I version 2 af vores ER-Diagram er der selvfølgelig en del ændringer. v.1.0 blev lavet i starten af projektet, inden vi havde et fast overblik over, hvordan vi gerne ville strukturere vores projekt. Her i v.2.0 er der lavet en del ændringer og databasen ser derfor anderledes ud. Her er vores [ER-Diagram v2.0](#).

I denne version af vores database har vi inkluderet 8 nye tabeller. Diagrammet viser nu en lidt mere avanceret opbygning, hvor tabellerne hænger godt sammen. De er koblet via primær- og fremmednøgler, som hjælper med at holde styr på dataene og gøre det nemt at hente den rigtige information.

Med de her ændringer passer databasen bedre til projektets behov, lige nu, og kan også nemt tilpasses, hvis vi skal lave noget om eller tilføje noget senere. Version 2.0 er derfor et vigtigt skridt mod en løsning, der fungerer godt både nu og i fremtiden.

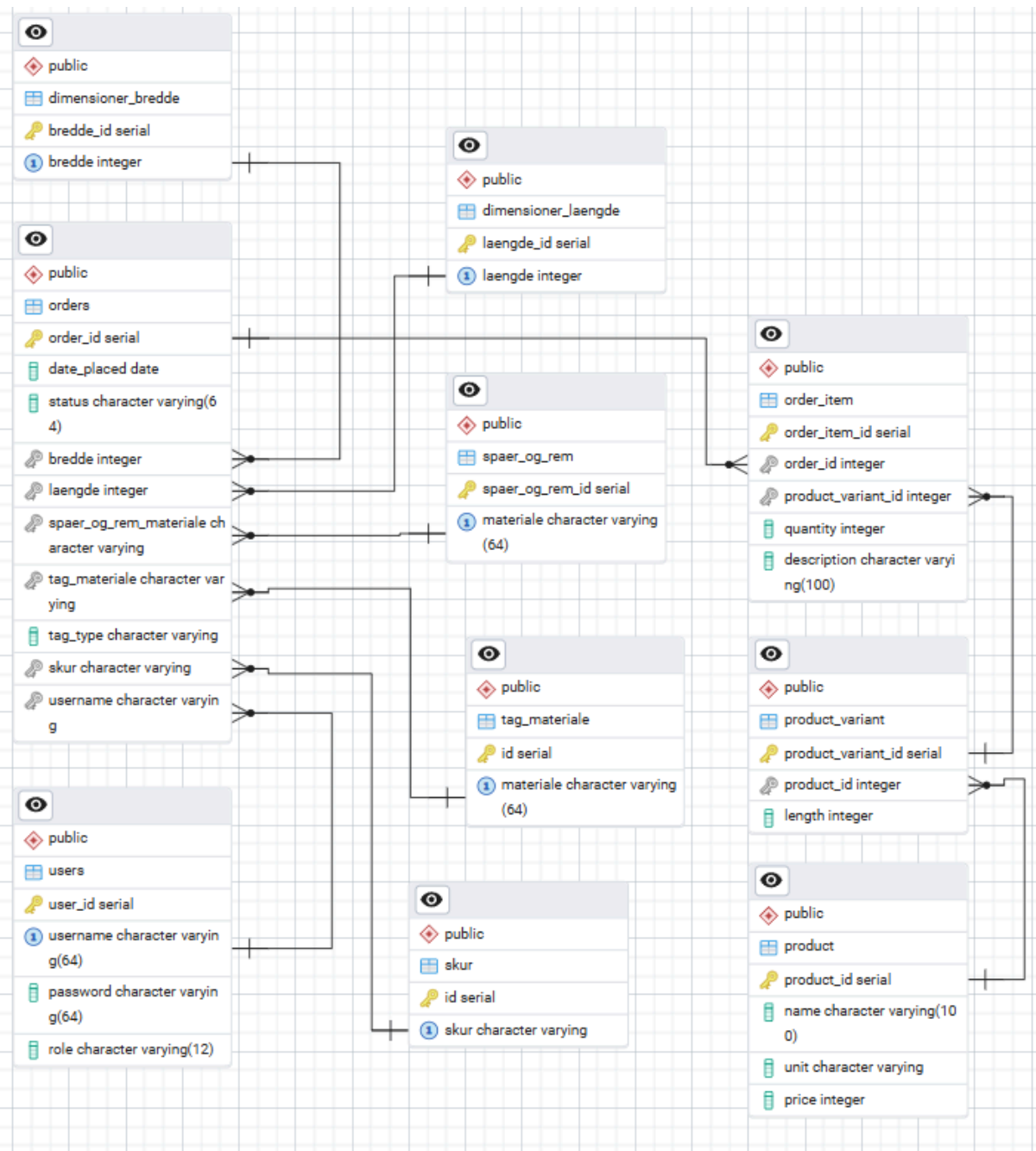


Fig 6. ERD v.2.0

Normalformer og databasestruktur

Databasen til dette projekt er bygget for at være enkel, effektiv og nem at arbejde med. Herunder beskrives, hvordan databasen er opbygget, og hvorfor vi har valgt at gøre det på denne måde.

Databasestruktur

Vi har ti tabeller i databasen:

1. Users

Denne tabel gemmer oplysninger om brugere, som for eksempel deres brugernavn, adgangskode og hvilken rolle de har (f.eks. bruger eller admin). Hver bruger har et [user_id](#), som gør det nemt at identificere dem.

2. Orders

Denne tabel bruges til at holde styr på ordrer. Den gemmer blandt andet et [ordre-ID](#), [date_placed](#) for, hvornår ordren blev lagt, [status](#) på ordren og information om, hvilken bruger der har lagt den i form af en fremmed nøgle som afhænger af [user_id](#) fra [user](#)-tabellen. Derudover indeholder tabellen oplysninger om dimensioner, tagmateriale, skur og andre produktdetaljer, som kommer af foreign keys fra de andre tabeller.

3. Order_item

Denne tabel bruges til at lagre information om produkter, der er knyttet til en ordre. Hver række har et unikt [order_item_id](#) og refererer til både [order_id](#) og [product_variant_id](#).

4. Dimensioner_bredde og Dimensioner_laengde

Disse tabeller indeholder de mulige bredder og længder for carporte. Kolonnerne [bredde](#) og [laengde](#) er unikke og fungerer som valgmuligheder for brugeren.

5. Tag_materiale

Tabellen indeholder tilgængelige materialer til carportens tag. Hver række har et unikt ID og en [materiale](#)-værdi, f.eks. *Benders sort* eller *Sunlux 1300K*.

6. Skur

Denne tabel angiver, om et skur skal inkluderes i carporten. Den har værdierne [Ja](#) og [Nej](#). Den har kun disse 2 værdier. Selvom den kun har disse 2 værdier, så har vi valgt at lave en hel tabel for det pga. vores databasestruktur.

7. Spear_og_rem

Denne tabel specificerer mulige materialer til spær og rem. Den indeholder en unik ID og et [materiale](#)-felt med værdier som *Benders brun* og *Eternit grå B6*.

8. Product og product_variant

[Product](#) indeholder oplysninger om produkternes grundlæggende egenskaber som navn og pris, mens [Product_variant](#) repræsenterer variationer af produkterne, f.eks. forskellige længder.

Hvorfor vi har valgt denne opbygning

Primærnøgler

Hver tabel har en primærnøgle (et unikt ID), som gør det nemt at holde styr på, hvem der er hvem, og hvad der er hvad. Det sikrer, at der ikke opstår forvirring, hvis vi skal finde bestemte data eller lave ændringer i tabellerne.

Fremmednøgler

For at skabe sammenhæng mellem tabellerne har vi brugt fremmednøgler. For eksempel refererer `username` i `Orders` til `Users`, mens `bredde` og `laengde` i `Orders` refererer til henholdsvis `Dimensioner_bredde` og `Dimensioner_laengde`. Disse relationer sikrer integritet i databasen og gør det nemt at oprette præcise forespørgsler.

Unique constraints

Vi har sikret, at felter som `bredde`, `laengde` og `materiale` er unikke, hvilket forhindrer dubletter og opretholder dataintegriteten.

Normalisering af databasen

Vores database er bygget, så den overholder principperne for normalisering, og vi har især haft fokus på at nå 3. normalform. Det betyder, at vi har fjernet unødvendige gentagelser og sørget for, at data kun findes ét sted, hvor det giver mening.

For eksempel gemmer vi ikke information om brugere flere steder – i stedet har vi én tabel, der holder styr på brugernes data, og vi refererer til den med fremmednøgler i andre tabeller. På den måde undgår vi problemer som modsatrettede eller forældede oplysninger.

3. Normalform betyder også, at hver tabel kun indeholder data, der hører sammen. For eksempel har vi en separat tabel til `bredde` og `længde` i stedet for at blande det med andre dimensioner. Det gør databasen simpel og nemmere at vedligeholde.

Eksempler på, hvordan vi overholder 3. normalform

Vores database er designet til at overholde 3. normalform, og her er nogle konkrete eksempler fra vores tabeller, der viser det:

1. Adskillelse af gentagende data:

Vi har separate tabeller til `dimensioner_bredde` og `dimensioner_laengde`, som indeholder unikke værdier for `bredde` og `længde`. I stedet for at gemme disse dimensioner direkte i andre tabeller gentagne gange, bruger vi fremmednøgler til at referere til de respektive tabeller. Dette forhindrer redundans og sikrer konsistens, da en bestemt `bredde` eller `længde` kun findes ét sted.

2. Ingen afhængighed mellem ikke-nøglekolonner:

I tabellen `orders` har vi en kolonne `username`, der er en fremmednøgle til tabellen `users`. Hvis vi skulle gemme brugerens kontaktoplysninger direkte i `orders`, ville det bryde 3. normalform, fordi brugerens data ikke afhænger direkte af `ordre-ID`'et. Ved

at opdele oplysningerne i separate tabeller undgår vi unødvendig duplikering og afhængighed mellem ikke-nøglekolonner.

3. Unikke værdier sikret med constraints:

For at undgå gentagelse har vi sat unikke constraints på kolonner som [bredde](#) i [dimensioner_bredde](#) og [materiale](#) i [tag_materiale](#). Dette sikrer, at hver værdi kun kan optræde én gang, hvilket opfylder 3. normalform.

4. Fremmednøgler sikrer relationer:

Vi bruger fremmednøgler til at forbinde tabeller på en måde, der opretholder dataintegritet. For eksempel:

- Kolonnen [bredde](#) i [orders](#) refererer til [dimensioner_bredde](#).
- Kolonnen [tag_materiale](#) i [orders](#) refererer til [tag_materiale](#).

Disse relationer sikrer, at hver tabel kun indeholder data, der direkte afhænger af dens primærnøgle, og at vi undgår unødvendige duplikationer.

Ved at bruge de her principper har vi fået en database, der er effektiv, overskuelig og klar til at vokse med projektet.

Fordele ved denne løsning

- **Nem at vokse med:** Opbygningen gør det muligt at tilføje flere data, f.eks. flere brugere eller ordrer, uden at det bliver uoverskueligt.
- **God orden:** Det er nemt at holde styr på, hvordan tabellerne hænger sammen, så vi kan lave præcise forespørgsler og undgå fejl.
- **Sikkerhed:** Ved at opbevare oplysningerne, som f.eks. brugernavne og adgangskoder, kan vi nemmere sikre, at følsomme data bliver behandlet korrekt. Ved at opbevare en brugers kodeord i form af password hashing, giver det brugeren tryghed ved at ingen har adgang til deres adgangskode.
- **Fleksibilitet:** Hvis vi vil tilføje nye funktioner eller ændre noget i fremtiden, er det nemmere at gøre det, når databasen er bygget op på denne måde.
- **Effektivitet:** Relationerne og de unikke constraints sikrer hurtige og præcise forespørgsler.

Denne struktur gør, at databasen er tilpasset projektets behov nu og i fremtiden og er nem at arbejde videre med.

Navigationsdiagram & Mockups

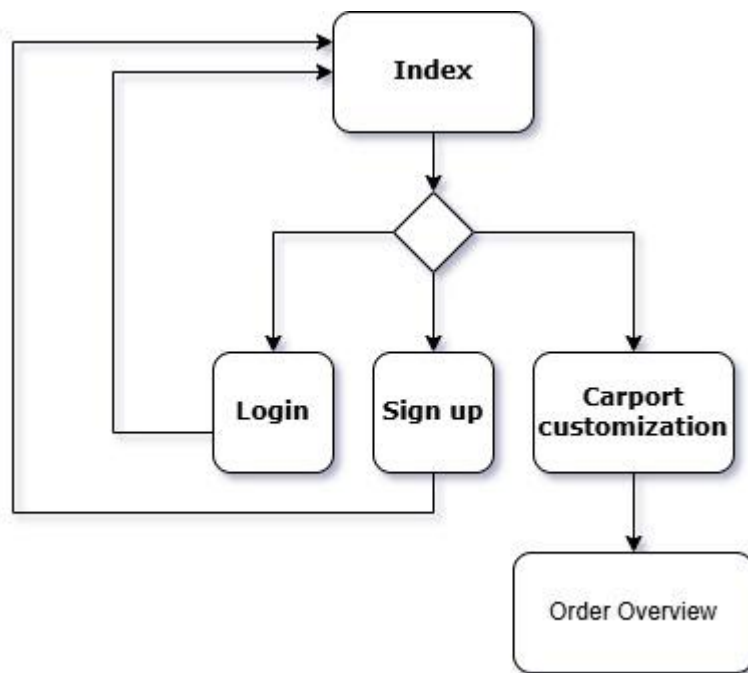


Fig 7. Navigationsdiagram

I toppen af alle sider findes der en navigationsbar (fragment) med Logo man kan klikke for at komme til forsiden, login/logout, og admin muligheder hvis man er af brugertype admin (ikke implementeret andet end placeholder proof of concept)

Order overview kan kun tilgås af brugere som er logget ind. Man kan stadig gå ind og kigge igennem mulighederne hvis man ikke er logget ind, men for at oprette en ordre skal man være logget ind.

index.html

- Forside, hvor brugeren kan vælge mellem login, opret bruger eller høj/lav rejsning af tag, som tager dem videre til dimensions.html
- **Route:** /

login.html

- Login-side, hvor brugeren indtaster deres e-mail og kodeord.
- **Route:** /login

createuser.html

- Formular til oprettelse af en ny bruger, med email og password
- **Route:** /createuser

dimensions.html

- Valg af dimensioner og materialer for carport.
- **Route:** /dimension

order.html

- Viser styklister og ordre-ID.
- **Route:** /order?id={ordrelid}

Valg af arkitektur

Vi bruger løst en MVC inspireret arkitektur, da det giver et godt overblik og sammenhæng i programmet, at vi har en frontend (view), database (model) og en bro i mellem dem i form af controllers og mappers (controller).

Vores view er seperet ind i **/resources** pakken, hvor der findes alt vores html, css og billeder. Vores model er en postgresql server som vi tilgår via pgadmin og har sql script til, i **/resources/sql**.

Oversigt over pakker og klasser i systemet

/config

Indeholder konfigurationsklasser for systemet.

- SessionConfig: Håndterer sessioner og konfiguration.
- ThymeleafConfig: Konfigurerer Thymeleaf som templatemotor til at rendere views.

/controller

Indeholder controllers, som håndterer brugerinput og routing af HTTP-requests.

Disse controllers interagerer med persistence-laget for at hente og manipulere data.

- CarportController: Håndterer anmodninger vedrørende carporte. Og viser dynamiske formularer med dimensioner, tagmaterialer, skur og spær/rem.
- OrderController: Håndterer ordre-relaterede anmodninger. Bruger en Calculator-service til at udregne materialer og opdatere ordren.

- UserController: Administrerer brugerrelaterede operationer, og implementerer grundlæggende validering af brugerinput. Bruger UserMapper til at interagere med databasen.

/entities

Indeholder dataklasser (modeller), der repræsenterer objekter i systemet, og er næsten udelukkende bestående af boilerplates da de bare bruges af andre klasser til at putte data fra databasen ind i instanser af.

/exception

Indeholder brugerdefinerede undtagelser (exceptions).

/persistence

Indeholder mapper-klasser (DAO'er), der interagerer med databasen.

- ConnectionPool: Singleton, der håndterer forbindelser til databasen.
- CarportMapper, OrderMapper, ProductMapper, UserMapper: Mapper klasser, der håndterer CRUD-operationer for specifikke datamodeller.

/services

Indeholder hjælpeklasser og serviceklasser med forretningslogik.

- Calculator: Indeholder metoder til at udføre beregninger af længder og antal til styklisten

/resources

Indeholder statiske filer og views til applikationen.

- /public: Indeholder statiske ressourcer som CSS og JavaScript.
- /templates: Indeholder Thymeleaf HTML-skabeloner.
- /sql: Indeholder SQL-scriptet til at opsætte og administrere databasen.

Brug af hjælpeklasser og designmønstre

Singleton:

ConnectionPool bruger Singleton-mønsteret til at sikre en enkelt database-forbindelses-instans, der kan genbruges på tværs af applikationen, ved at oprette den i Main og passe den med ned i metodekald.

Serviceklasser:

Calculator er en hjælpeklasse, der indeholder beregning logik for carporte.

Dette holder controllere rene og fokuserede på routing og input håndtering.

Særlige forhold

Dette afsnit beskriver de særlige tekniske forhold og valg, der er gjort i udviklingen af vores program. Herunder gennemgår vi blandt andet sessionhåndtering, brugerinputvalidering, sikkerhed i forbindelse med login, håndtering af roller og andre nøgleelementer i systemet.

Ligesom tidligere i rapporten, så vil alle nøgleord, som er kode relevante, blive highlighted med `cornflower blue` og skrevet med `Roboto mono` for at øge denne sektioners læsbarhed. Dette kunne f.eks. være:

`SessionAttribute`

Sessions, request og variabler

I vores program benyttes **session variabler** til at holde styr på brugerens login-status og gemme midlertidig information, der er unik for den enkelte bruger. Eksempelvis gemmer vi det aktuelle brugernavn og rolle for en bruger i sessionen, når de logger ind:

```
try {  
    User user = UserMapper.login(name, password, dbConnection);  
    ctx.sessionAttribute("currentUser", user);  
}
```

`sessionAttribute("currentUser")` bruges til at gemme den aktuelle bruger i sessionen. Dette gør det muligt at holde styr på, hvilken bruger der er logget ind, og hvilke handlinger de har tilladelse til at udføre.

Vi bruger også attributes til at sende beskeder til brugeren. Dette kunne være enden beskeder pga. fejl eller succes:

```
if (!password.equals(confirmPassword)) {  
    ctx.attribute("message", "Kodeord matcher ikke. Prøv igen");  
}
```

Håndtering af exceptions

For at sikre stabilitet i systemet håndterer vi exceptions med **try-catch** blokke. Hvis en fejl opstår, opfanges den og håndteres ved at vise en fejlmeddelelse til brugeren. Her gør vi også brug af attributes.

Eksempel på håndtering af fejl:

```
} catch (DatabaseException e) {  
    ctx.attribute("message", "Error fetching dimensions: " + e.getMessage());  
}
```

Validering af brugerinput

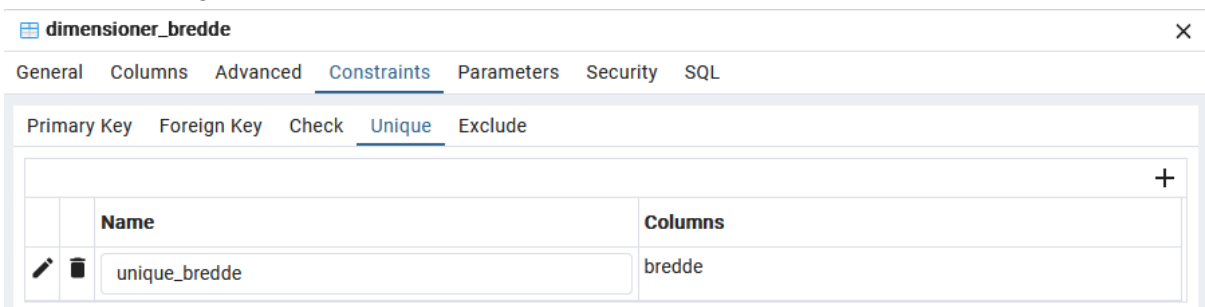
For at sikre, at brugerinput er korrekt, benytter vi en kombination af **server-side validering** og database definitioner som **NOT NULL** og unikke constraints i vores database.

Eksempel på simpel validering af input:

```
// Add simple validation for dimensions  
if (bredde == null || laengde == null || tagMateriale == null || skur == null || spaerOgRem == null) {  
    ctx.attribute("message", "Venligst udfyld alle felter.");  
    showAllForms(ctx, dbConnection);  
    return;  
}
```

Derudover valideres input i databasen ved hjælp af constraints, som f.eks. unikke værdier for **dimensioner_bredde** og **dimensioner_laengde**. Dette sikrer, at der ikke kan indsættes ugyldige eller dublerede værdier i databasen.

Eksempel fra pgadmin:



Sikkerhed med login og opbevaring af sensitiv data

Sikkerheden omkring login er implementeret ved hjælp af en kombination af:

1. Password hashing og saltning:

Brugernes adgangskoder bliver ikke gemt som almindelig tekst, men i stedet som en hashed+saltet værdi. Vi bruger **BCrypt** til at hash og validere adgangskoder:

```
String hashedPassword = BCrypt.hashpw(password, BCrypt.gensalt());
```

Og når en bruger skal logge ind:

```
if (BCrypt.checkpw(password, storedHashedPassword)) {
```

2. Session-styring:

Efter et vellykket login gemmes brugerens data i sessionen, som sikrer, at de kun har adgang til deres egne ressourcer.

3. Rollebaseret adgangskontrol:

Brugere har forskellige roller i systemet, som gemmes i databasen i kolonnen `role` i `users` tabellen. Rollen hentes fra sessionen og bruges til at begrænse adgangen. Rollen kan opfanges af Thymeleaf og kan bruges til f.eks. At vise nogle knapper til brugere med admin rollen:

```
<!-- Right Section: Admin-only buttons -->
<div class="navbar-right">
  <ul class="navbar-nav">
    <!-- Admin-only buttons -->
    <li class="nav-item" th:if="${session.currentUser != null and session.currentUser.role == 'admin'}">
      <a class="nav-link">Admin Placeholder 1</a>
    </li>
    <li class="nav-item" th:if="${session.currentUser != null and session.currentUser.role == 'admin'}">
      <a class="nav-link">Admin Placeholder 2</a>
    </li>
  </ul>
</div>
```

Dynamisk metode til hentning af data: `getDimensionOptions`

En vigtig del af vores program er funktionen `getDimensionOptions` (Bilag). Denne metode gør det muligt at hente data fra forskellige tabeller i databasen ved at bruge variabler til at angive kolonnen og tabellen.

Hvorfor brug af variabler er smart

I stedet for at skrive specifik kode til hver enkelt tabel og kolonne, har vi lavet én generisk metode, som kan genbruges. Metoden tager to parametre:

1. `column`: Navnet på den kolonne, vi ønsker at hente data fra.

2. `table`: Navnet på den tabel, hvor dataene skal hentes.

Ved at bruge disse variabler har vi undgået at skrive gentaget kode for hver tabel. For eksempel:

- I stedet for at lave én metode til `dimensioner_bredde` og en anden til `tag_materiale`, bruger vi den samme metode til begge tabeller ved blot at ændre værdien af variablerne.
- Dette gør koden nemmere at vedligeholde, da ændringer kun skal laves ét sted.

Derudover giver denne tilgang en fleksibilitet, som gør det muligt at tilføje nye tabeller eller kolonner senere uden at skulle skrive ny kode.

Beskyttelse mod SQL-injektion

Når man arbejder med dynamiske forespørgsler, så er der altid en risiko for **SQL-injektion**, hvor en bruger kan indsætte skadelige SQL-injections i inputtet. Vi har imidlertid sikret os mod dette på flere måder:

1. Whitelist af gyldige tabeller og kolonner:

Først tjekker vi, om værdierne i variablerne `table` og `column` er en del af vores accepterede lister over gyldige tabeller og kolonner:

```
List<String> validTables = List.of("dimensioner_bredde", "dimensioner_laengde", "tag_materiale", "skur", "spaer_og_rem");
List<String> validColumns = List.of("bredde", "laengde", "materiale", "tag_type", "skur");

// Check if the inputs are valid
if (!validTables.contains(table) || !validColumns.contains(column)) {
    throw new DatabaseException("Invalid table or column name");
}
```

- Dette gør, at kun tilladte tabeller og kolonner kan bruges i SQL-forespørgslen.
- Forsøg på at indsætte uautoriserede værdier vil blive afvist og resultere i en exception.

2. Prepared Statements:

Vi bruger PreparedStatement til at lave SQL-forespørgslen. Dette sikrer, at input behandles som værdier og ikke som en del af selve SQL-kommandoen, hvilket gør SQL-injektion umulig:

```
String query = "SELECT " + column + " FROM " + table;
List<String> options = new ArrayList<>();
try (Connection connection = dbConnection.getConnection();
    PreparedStatement stmt = connection.prepareStatement(query);
    ResultSet rs = stmt.executeQuery()) {
```

Fremtidssikring og genbrug

Metoden er designet med tanke på **fremtidig udvidelse**. Hvis vi senere ønsker at tilføje flere tabeller eller kolonner, skal vi blot tilføje dem til de eksisterende lister over gyldige værdier:

```
List<String> validTables = List.of("dimensioner_bredde", "dimensioner_laengde", "tag_materiale", "skur", "spaer_og_rem");  
List<String> validColumns = List.of("bredde", "laengde", "materiale", "tag_type", "skur");
```

Derudover sparer vi tid ved at genbruge denne generiske metode flere steder i programmet i stedet for at skrive en ny metode for hver forespørgsel. Dette gør koden kortere, mere overskuelig og nemmere at vedligeholde.

Status på implementering

Fremtidig forbedring: Dynamisk validering af databaseforespørgsler

I den nuværende løsning bruger vi whitelisting til at validere de accepterede tabeller og kolonner i vores metode `getDimensionOptions`. Denne metode fungerer ved at sammenligne input med en hardcoded liste over gyldige tabeller og kolonner.

Selvom denne tilgang sikrer høj sikkerhed og forhindrer uønskede SQL-injektioner, har den også nogle begrænsninger. Fog har udtrykt et ønske om at få mere kontrol over deres database. Den nuværende hardcoded løsning kræver ændringer i backend-koden, hver gang der skal foretages justeringer.

Forslag til en fremtidig løsning

En mulig løsning, som vi kunne implementere, hvis vi havde mere tid til projektet, ville være at give **admin** mulighed for dynamisk at opdatere de accepterede tabeller og kolonner via en brugergrænseflade. Ideen er, at de gyldige værdier kan gemmes i en database i stedet for at være hardcoded i koden.

Admin-kontrolpanel:

Vi kunne lave et simpelt admin-panel, hvor administratorer kan:

- Tilføje nye tabeller og kolonner til listen.
- Fjerne eksisterende værdier, hvis de ikke længere skal være tilladt.

Dynamisk hentning af gyldige værdier:

Metoden `getDimensionOptions` kunne opdateres til at hente de gyldige værdier fra databasen i stedet for en hardcoded liste

På den måde kan listen af gyldige tabeller og kolonner opdateres uden at ændre backend-koden.

Selvom vi i denne version af projektet valgte en simpel whitelisting-metode for at prioritere sikkerhed og tidsbesparelse, er dette en oplagt forbedring, som vi ville implementere, hvis vi havde mere tid. Det ville give Fog større fleksibilitet og gøre systemet endnu mere dynamisk og fremtidssikret.

Email

På vores nuværende hjemmeside har vi to placeholder-knapper, hvor det var intentionen, at den ene skulle fungere som et mailsystem. På grund af tidsmangel og kompleksiteten ved at udvikle et fuldt fungerende mailsystem, valgte vi ikke at påbegynde denne del af projektet.

Fog ønsker at holde en løbende korrespondance mellem deres kunder og sælgere. I deres nuværende arbejdsgang bruger de en ekstern mailtjeneste til at kommunikere med deres kunder. Dette indebærer, at de manuelt skal hente kundeinformation og ordredata fra ét system og derefter skrive mails i et andet.

Hvis vi havde haft mere tid, ville vi have udviklet et integreret mailsystem direkte i platformen. Dette ville gøre det muligt for Fog at:

- Sendte mails direkte fra systemet, hvor information om kunden og ordren allerede er tilgængelig.
- Automatisk indhente data fra ordrer, som f.eks. brugerens navn, ordrestatus og dimensioner på carporten, for at gøre kommunikationen hurtigere og mere præcis.
- Effektivisere arbejdsprocessen ved at samle flere funktioner i én platform og dermed mindske afhængigheden af eksterne programmer.

Dette ville gøre det nemmere for Fog at håndtere ordrer og holde kontakt med kunderne, samtidig med at det ville spare tid. Selvom funktionen ikke blev implementeret i denne version af projektet, ser vi det som en vigtig fremtidig forbedring, der vil skabe stor værdi for både Fog og deres sælgere.

Kvalitetssikring (test)

På vores java app har vi udviklet unit- og integrations test, for at afprøve og kunne demonstrere, at dele af vores program virker som forventet.

Unit tests giver kun mening at køre på Calculator klassen, da det er de eneste metoder som isoleret kan testes. Resten af programmets metoder afhænger af database og andet kontekst. Der var ikke tid til at lave god code coverage, men succesfulde integrationstest er sat op i ProductMapperTest, som systematisk renser og opsætter en testdatabase, tjekker at den findes, og tester connections før den kører metode tests.

User acceptance tests

I tabellen her vises et overblik over vores udviklede user stories, acceptance kriterier, og status på implementation og funktionalitet efter vores selvtestede user stories.

<u>User story</u>	<u>Acceptance kriterier</u>	<u>Status</u>
US-1 Som kunde eller admin kan jeg skabe en bruger For at senere kunne logge ind eller betale for en carport	Givet at jeg ikke er logget ind og trykker på opret bruger menupunktet. når jeg har udfyldt opret bruger formularen og trykket på opret bruger knappen Så bliver der oprettet en bruger, som jeg kan logge ind med senere	Godkendt ✓ Fuld implementation
US-2 Som kunde eller administrator Kan jeg logge på systemet med email og kodeord. Når jeg er logget på, så skal jeg kunne se de muligheder jeg har ud fra min rolle som kunde eller administrator	Givet at jeg ikke er logget ind, så kan jeg logge ind på en oprettet bruger. Når jeg indtaster en eksisterende bruger, som er oprettet i systemet. Så kan jeg blive logget ind på hjemmesiden	Godkendt ✓ Fuld implementation
US-3 Som bruger eller admin vil jeg kunne indtaste ønskede dimensioner (bredde, længde, højde) på carporten, så kunden kan få deres ønskede carport med skræddersyede dimensioner.	Når man som bruger eller admin når til punktet hvor man kan vælge sine egne dimensioner på carporten skal man kunne indtaste sine egne mål, indenfor visse grænser.	Godkendt ✓ Fuld implementation
US-4 Som bruger eller admin vil jeg kunne vælge om hele carporten skal være et redskabsrum/skur, eller om skuret ikke skal optage alt pladsen i carporten, så kunden kan få deres ønskede carport eller skur	Når man som bruger eller admin når til punktet hvor man kan vælge om man vil have skur eller carport skal man kunne klart vælge hvilken udgave man ønsker	Godkendt ✓ Fuld implementation
US-5 Som bruger eller admin vil	Når man som bruger eller admin	Godkendt ✓ Fuld implementation

jeg kunne indtaste ønskede tagmateriale på carportens tag, så kunden kan få deres ønskede tagtype.	når til punktet hvor man kan vælge sine tagmateriale skal man kunne fra en dropdown vælge tagmateriale	
US-6 Som bruger eller admin vil jeg kunne indtaste ønskede trætype for spær og rem, så kunden kan få deres ønskede trætype.	Når man som bruger eller admin når til punktet hvor man kan vælge sine trætype skal man kunne fra en dropdown vælge sine ønskede trætyper for spær og rem	Godkendt ✓ Fuld implementation
US-7 Som bruger eller admin vil jeg kunne tilføje et redskabsrum med justerbare bredde og dybde samt vælge beklædning og gulvtype Så kunden kan få din ønskede skræddersyede carport	Når man som bruger eller admin når til punktet hvor man foretager valg omkring redskabsrum skal man kunne indtaste sine egne mål, inden for visse grænser, og vælge beklædning og gulvtype.	Godkendt ✓ Fuld implementation
US-8 Som admin vil jeg kunne downloade eller få vist en detaljeret stykke liste så kunden har alle nødvendige komponenter til at bygge carporten.	Givet at alle specifikationer er indtastet korrekt og kunden har betalt og jeg går til sidste trin i processen ved at trykke f.eks. "beregne stykke liste" bliver der vist en stykke liste, som evt. kan downloades	Acceptable + Ikke fuld implementering. På nuværende tidspunkt skal en bruger ikke betale før de får deres stykke liste. Dette ville vi selvfølgelig rette op på i fremtiden
US-9 Som administrator kan jeg tilgå mine administrator redskaber Når jeg er logget på skal jeg kunne se mine redskaber, som jeg skal bruge til beregning af carporte eller lign.	Givet at jeg er logget ind som administrator når jeg kigger på index siden så kan jeg se alle mine redskaber og klikke på dem for at åbne siderne	Acceptable + Ikke fuld implementering. Knapperne eksistere, men på nuværende tidspunkt har de ingen funktion
US-10 Som kunde Kan jeg bestille en carport med special mål	Givet at jeg bestiller en carport med special mål Når jeg indtaster mine ønskede mål	Ikke implementeret. ✗ Vi sprang hen over dette trin, fordi vi ikke følte det var nødvendigt. Alle mål som

Når jeg indtaster de ønskede mål, så kunne jeg godt tænke mig at systemet genkendte gyldige mål og gav beskeder om, hvad der er muligt	Så får jeg en fejlbesked hvis de ikke kan lade sig gøre, men også forslag til hvad der kan lade sig gøre. Hvis en mulighed ikke findes, så kan ønsket tvinges igennem alligevel og en dialog mellem sælger og kunde kan starte.	kan vælges fra vores drop downs er godkendte
US-11 Som administrator Kan jeg klikke på en mail knap og komme til et mailsystem Når jeg er logget på som administrator og klikket på mail, så kan jeg se de ordre og mails der er sendt fra kunder.	Givet at jeg er logget på som administrator. Når jeg klikker på mail knappen Så kommer jeg ind på en side med mails fra kunder	Ikke implementeret. ✗ Manglende tid i projektet. Bedømte at det var uden for scope på nuværende tidspunkt
US-12 Som administrator Kan jeg sende email til kunder. Når jeg er logget på som administrator og er inde på mail systemet, så kan jeg sende emails tilbage til de ordre der er placeret.	Givet at jeg er administrator og logget ind Når jeg trykker på en send email knap Så kan jeg skrive emails og sende dem til den email som har lavet ordren	Ikke implementeret. ✗ Manglende tid i projektet. Bedømte at det var uden for scope på nuværende tidspunkt
US-13 Som administrator Kan jeg hente data fra de email jeg modtager. Når jeg er inde i "Quick byg" systemet, så kan jeg hente data fra email systemet Sådan at jeg ikke manuelt skal bruge tid på de forskellige systemer US-15 Ændre data i databasen	Givet at jeg er inde på "Quick byg" systemet Når jeg trykker på den knap, så kan jeg vælge en email fra sit mailsystem Så indsætter den data ind i mit "Quick byg" system	Ikke implementeret. ✗ Manglende tid i projektet. Bedømte at det var uden for scope på nuværende tidspunkt
US-14 Som admin vil jeg kunne downloade eller vi vist en detaljeret SVG-tegning af deres skræddersyede carport så jeg kan præsentere designet for kunden	Givet at alle specifikationer er indtastet korrekt og kunden har betalt og jeg går til sidste trin i processen ved at trykke f.eks. "beregnet styklister" bliver der vist en SVG-tegning af carporten	Ikke implementeret. ✗ Manglende tid i projektet. Bedømte at det var uden for scope på nuværende tidspunkt

--	--	--

Proces

Arbejdsprocessen faktuel

Vi har i projektet anvendt **GitHub Projects** og et **Kanban Board** til at organisere vores opgaver og user stories. Kanban Boardet blev inddelt i de klassiske kolonner:

- **To Do:** Her blev alle opgaver og user stories placeret, når de blev identificeret.
- **In Progress:** Når en opgave blev påbegyndt, flyttede vi den til denne kolonne for at indikere, at der arbejdes på den.
- **Review:** Her blev de færdige opgaver lagt og et pull-request associeret med kortet.
- **Done:** Når en opgave var afsluttet, og pull-requestet accepteret, flyttede vi den til denne kolonne.

Dette system gjorde det nemt at holde styr på, hvilke opgaver der var i gang, og hvilke der var afsluttet. På vejledningsmøder brugte vi boardet til at gennemgå status for opgaverne og sikre, at vi begge, i teamet, var på samme side.

Vi kommunikerede løbende imellem os ved at kommentere direkte på opgaver i GitHub Issues og pull-requests, ved at tildele dem til det relevante teammedlem. Det gav os et klart overblik over, hvem der arbejdede på hvad.

Til vejledningstimerne gik det godt. Vi brugte mest disse møde til at rette i vores scope for projektet. Vi startede projektet ud med at absolut lave ingen kode i den første uge. Vi havde ikke engang oprettet et projekt i IntelliJ før uge 2. I stedet brugte vi tiden på: risikoanalyse, interessemøder, figma mockup, user stories, acceptance criteria, tasks, opsætning af hele vores kanban board, strukturering af projektet, kundeanalyse, generelt organisering af hele projektet og mange flere som vi måske ikke kan komme i tanke om.

Arbejde med User Stories og GitHub Projects

I projektet har vi arbejdet meget struktureret med vores **user stories** ved at bruge **GitHub Issues** og **GitHub Projects** som værktøj. For hver opgave, oprettede vi et *issue*, som indeholdt:

1. **User Story:**

En kort beskrivelse af, hvad brugeren skulle kunne gøre. Dette hjalp os med at holde fokus på den konkrete funktionalitet. F.eks.:

"Som bruger eller admin vil jeg kunne indtaste ønskede dimensioner (bredde, længde, højde) på carporten, så kunden kan få deres ønskede carport med skræddersyede dimensioner."

2. Acceptance Criteria:

Her beskrev vi tydeligt, hvornår opgaven kunne anses som færdig. F.eks.:

- Når man som bruger eller admin
- når til punktet hvor man kan vælge sine egne dimensioner på carporten
- Så skal man kunne indtaste sine egne mål inden for visse grænser.

3. Tasks:

Opgaverne blev nedbrudt i mindre dele for at gøre arbejdet mere overskueligt. Her listede vi f.eks.:

En html side skal findes, med alle justeringsmuligheder for carporten. Carportens dimensioner skal være en af mulighederne, gerne hvor der er min og max grænser for dimensionerne, som evt. kunne hentes fra en database, så det ikke er hardcoded ind i systemet.

Valget skal gemmes så dataet kan tilgås senere, når styklisten og SVG genereres.

Denne nedbrydning gjorde det lettere for teamet at tildele og arbejde på de forskellige dele.

Integration med Development og Kanban Board

Vi brugte Development-fanen i GitHub Issues til at forbinde hvert *issue* med en branch i GitHub. Når vi begyndte at arbejde på en opgave, oprettede vi en ny branch med et navn, der matchede *issue*et, f.eks.:

[feature/us-4-carport-dimensioner](#).

- Når vi oprettede et **Pull Request** (PR) til denne branch, blev PR'et automatisk forbundet med det tilhørende *issue*.
- Når koden blev gennemgået og godkendt, blev PR'et merged ind i dev-branchen (*dev*).
- Som en ekstra smart funktion flyttede **GitHub Projects** automatisk kortet fra *In Progress* til *Done*, når pull-requestet blev merged.

Denne automatisering gjorde arbejdsprocessen meget strømlinet og reducerede behovet for manuelt at opdatere boardet.

Eksempel:

På billedet af [US-4 carport dimensioner](#) (Bilag) kan man se, hvordan vi beskrev opgaven, nedbrød den i Tasks og brugte Development-fanen til at forbinde opgaven med den relevante branch. Når opgaven var færdiggjort, blev kortet automatisk flyttet til Done, hvilket gjorde det nemt at se, at opgaven var løst.

Denne måde at arbejde på, sikrede at vi kunne holde styr på projektets mange dele, samarbejde effektivt som team og levere funktionalitet på en struktureret måde.

Fordele ved denne tilgang

1. **Klar struktur:** User Stories og Tasks gjorde det nemt at forstå opgaverne og sikre, at vi kun arbejdede på det, der var nødvendigt.
2. **Automatisering:** Ved at bruge branches og PR's sammen med Kanban Boardet blev opgaver automatisk opdateret og flyttet. Dette sparede os tid og gjorde boardet mere præcist.
3. **Tæt kobling mellem kode og opgaver:** Hvert issue var direkte knyttet til en branch og en funktion i koden, hvilket gjorde det nemt at holde styr på, hvad der blev udviklet og hvorfor.
4. **Overblik:** Ved at bruge GitHub's værktøjer fik vi et visuelt overblik over projektet og kunne nemt følge med i, hvor langt vi var.
5. **Nemt at følge med:** Ved at linke alle issues med branches og kanban kort, kunne man nemt følge med i arbejdsprocessen for de forskellige user stories.

Arbejdsprocessen reflekteret

Ved brug af Kanban-metoden i GitHub Projects oplevede vi flere fordele, men også nogle udfordringer:

Det der fungerede godt:

- GitHub Projects hjalp os med at bevare overblikket over projektet, da alle opgaver var samlet ét sted.
- Muligheden for at flytte opgaver mellem kolonner gjorde det nemt at prioritere og fokusere på de vigtigste opgaver først. F.eks. blev de mindre vigtige opgaver lagt i backlog kolonnen.
- Vi kunne hurtigt identificere blokeringer eller opgaver, der blev hængende i In Progress, og tage handling på dem.
- Integration med development på kanban-boardet var utrolig hjælpsom og gjorde den manuelle proces med de forskellige kort en del nemmere. Det sparede os for en masse management arbejde.

Udfordringer:

- Vi havde i starten problemer med at nedbryde user stories i tilstrækkeligt små opgaver. Det betød, at nogle opgaver tog længere tid end forventet. Her havde vi f.eks. estimeret stykliste til 3-7 dage, men det kom til at tage 3+ uger.
- Det fungerede godt i starten, med at nedbryde projektet i user stories. Sammen med kanban boardet, gjorde det at vi ikke lavede forgæves arbejde. Det skabte til gengæld store huller, da nogle user stories byggede ovenpå dele af programmet, som ikke eksisterede endnu. Dette var grundet at nogle user stories, var baseret på at andre user stories skulle laves først.
- Ud over at kigge på user stories, var det svært at skabe et overblik over projektets opgaver. Der blev lagt lidt for meget energi i gætteværk til forskellige diagrammer, da vi burde først have fundet ud af hvad der skal på dem. Ting som deployment, tests

mm. Kom på bordet meget sent, da vi knap nok havde overblik over de første trin af projektet.

- Som team skulle vi vænne os til at opdatere boardet løbende, så det hele tiden var en nøjagtig afspejling af virkeligheden. Det gik ikke altid så let, men vi var gode til det 95% af tiden ca.
- Overskud og overblik var mindsket, da vi blot er en 2-mands gruppe. Vi sigtede højt, men var nødt til at de-scope projektet halvvejs i forløbet. Denne mangel på overskud, og følelse af at have mistet overblikket, gjorde at det med nogle dele af koden gik langsomt, og skulle kræve flere iterationer, før det passede ind i resten af projektet.
- Ca. halvdel i projektet begyndte at programmere mere individuelt. Der blev kommunikationen imellem medlemmerne af teamet mindsket. "Kommunikationen" var overladt til at kigge på vores kanban kort og følge med i hinandens opgaver.

Spot on med estimerer?

Vi erfarede, at vores estimeringer i starten ikke altid var præcise. Dette skyldes primært manglende erfaring med projektets kompleksitet. Over tid blev vi dog bedre til at vurdere opgavens omfang og fordele dem mere realistisk. Det gjorde også at vi, efter vejledningsmøder, var i stand til at tilpasse estimererne til opgaverne. Vi kunne derfor tildele nogle af opgaverne til backlog, da disse opgaver ville gøre vores scope for projektet for stort.

Refleksion over Integration af Development-fanen

Integration af Development-fanen i GitHub Issues har gjort vores arbejdsproces mere struktureret og effektiv. Ved at koble branches og pull requests direkte til vores opgaver blev arbejdsflowet automatiseret, så kort automatisk flyttede sig på Kanban-boardet, når Pull-requests blev merged.

Dette gav en tydelig sammenhæng mellem planlægning og udvikling og gjorde det nemt at følge op på, hvor langt vi var. Samtidig skabte det gennemsigtighed i teamet og sikrede, at koden blev gennemgået via Pull-requests, før den blev merged.

Samlet set har integrationen sparet tid, forbedret samarbejde og sikret høj kvalitet i vores kodebase. **Dette er en metode, vi vil tage med videre til fremtidige projekter. Dette gør sig også gældende i at bruge github projects generelt.**

Log-bog

Processen med logbogen gik faktisk rigtig godt i begyndelsen. De var meget fyldig gørende og dokumenterede vores proces fra dag til dag. Desværre da vi nåede ca. Halvvejs i projektet, begyndte vi at forglemme dem. De endte derfor op med ikke at være utrolig brugbare i sidste ende.

Vi er ikke sikker på om dette er en justering vi skal lave til fremtidige projekter. Det kunne måske evt. Være nemmere at skrive denne rapport hvis vi havde holdt dem opdateret.

Figma mock-up, diagrammer, modeller og faktisk design

Vi endte med ikke at følge vores mock-up til prikken. Da det rigtige kode skulle skabes, så var der bare løsninger og andre designs som var bedre, nemmere at lave eller mere intuitive end vores tidligere designs. Det samme gør sig gældende for vores tidlige diagrammer og modeller.

Afsluttende ord for processen

Hele processen i projektet er faktisk gået overraskende godt. Selvfølgelig havde vi vores små problemer med at scope vores projekt og opgaver som tog meget længere tid end forventet. Vi var dog i stand til at rette dette til og få et meget bedre projekt efter hvert vejledningsmøde.

Den overvældende generelle succes for vores projekt, vil jeg nok tildele til vores første uge af projektet. Processen i at nægte påbegyndelsen af vores kode, i den første uge, gjorde at vi havde rigtig meget tid til opsætning og strukturering. Vi er selvfølgelig ikke i stand til at se de alternative universer, hvor vi ikke startede projektet med så meget strukturering, men vi er stadig overbevist om, at dette er nøglen til vores succes i projektet.

Dette er helt klart en arbejdsproces, som vi vil tage videre til fremtidige projekter.

Til sidst vil jeg godt have lov til at sige, at det faktisk virker lidt mærkeligt med så meget succes. Måske det var fordi at vi kun var 2 i dette projekt, at ting var så overskuelige og at vi virkelig lagde noget arbejde i at gøre det godt fra starten.

Det kan næsten virke som om, at vi lægger skjul på de negative ting. Det er lidt surrealistisk ikke at kunne sige mange dårlige ting her i projektet, men igen, det har nok noget at gøre med vores proces i starten af projektet.

Tak til dig som læser

Vi vil gerne give nogle afsluttende ord her til sidst i rapporten.

Tak fordi du læste vores rapport. Vi håber den har givet en tilfredsstillende indsigt i vores projekt. Til og med håber vi også at det ikke har været alt for kedeligt at læse.

Vi håber du som læser får en god jul og godt nytår

Venlige hilsner fra

- Lukas Kjellerup Simonsen
- Rasmus Højholt

BILAG:

Website mockups v1:

Fog

Carport - skræddersyet

Lad os tegne din drømme carport - bestil din skræddersyede carport her!

For at tilgå vores bestillingsproces
skal du være logget ind på en bruger



Log ind



Ny bruger

Fog



← Carport - skræddersyet

Lad os tegne din drømme carport - bestil din skræddersyede carport her!

Vælg nedenfor, om du ønsker tilbud på carport
med fladt tag eller carport med høj rejsning.



Carport med
fladt tag



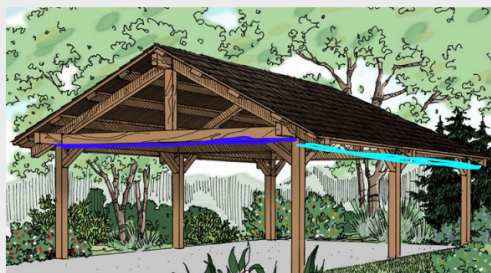
Carport med
høj rejsning

← Carport - skræddersyet

Lad os tegne din drømme carport - bestil din skræddersyede carport her!

Carport bredde

Carport længde



← Carport - skræddersyet

Lad os tegne din drømme carport - bestil din skræddersyede carport her!

Din valgte carport:

Bredde: 400 cm

Længde: 600 cm

Højde: 250 cm

Type: Halv shed

Shed bredde: 150 cm

Shed dybde: 200 cm

....

Bekræft og bestil

Fig 8. Mockups

ER-Diagram v.1.0

[Gå tilbake](#)

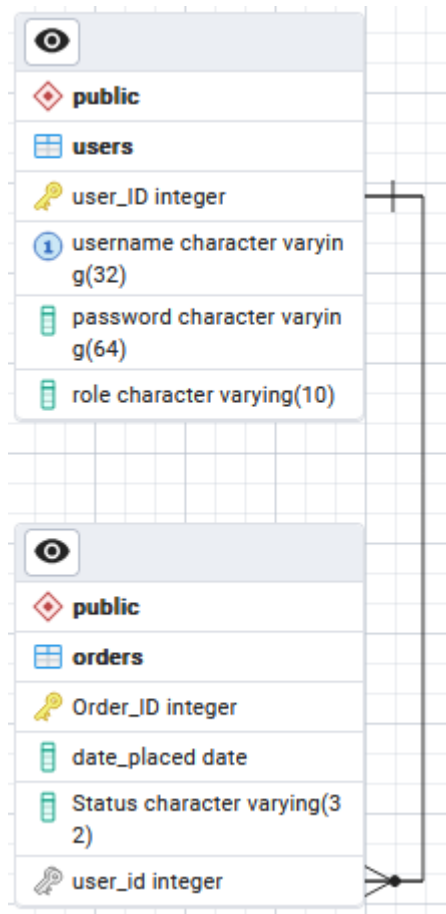


Fig 9. ERD v1

ER-Diagram v-2.0

[Gå tilbake](#)


```

public static List<String> getDimensionOptions(String column, String table, ConnectionPool dbConnection) throws DatabaseException {
    // Define valid table and column names
    // This is a pretty limiting way of doing it. It's just a whitelist.
    // We should probably make a new method in the future that gives the admins an option to add/remove/create tables
    List<String> validTables = List.of("dimensioner_bredde", "dimensioner_laengde", "tag_materiale", "skur", "spær_og_rem");
    List<String> validColumns = List.of("bredde", "laengde", "materiale", "tag_type", "skur");

    // Check if the inputs are valid
    if (!validTables.contains(table) || !validColumns.contains(column)) {
        throw new DatabaseException("Invalid table or column name");
    }

    String query = "SELECT " + column + " FROM " + table;
    List<String> options = new ArrayList<>();
    try (Connection connection = dbConnection.getConnection();
        PreparedStatement stmt = connection.prepareStatement(query);
        ResultSet rs = stmt.executeQuery()) {
        while (rs.next()) {
            options.add(rs.getString(column));
        }
    } catch (SQLException e) {
        throw new DatabaseException(e.getMessage());
    }
    return options;
}

```

Fig 11. getDimensionOptions()

Fog Carports - Domænemodel

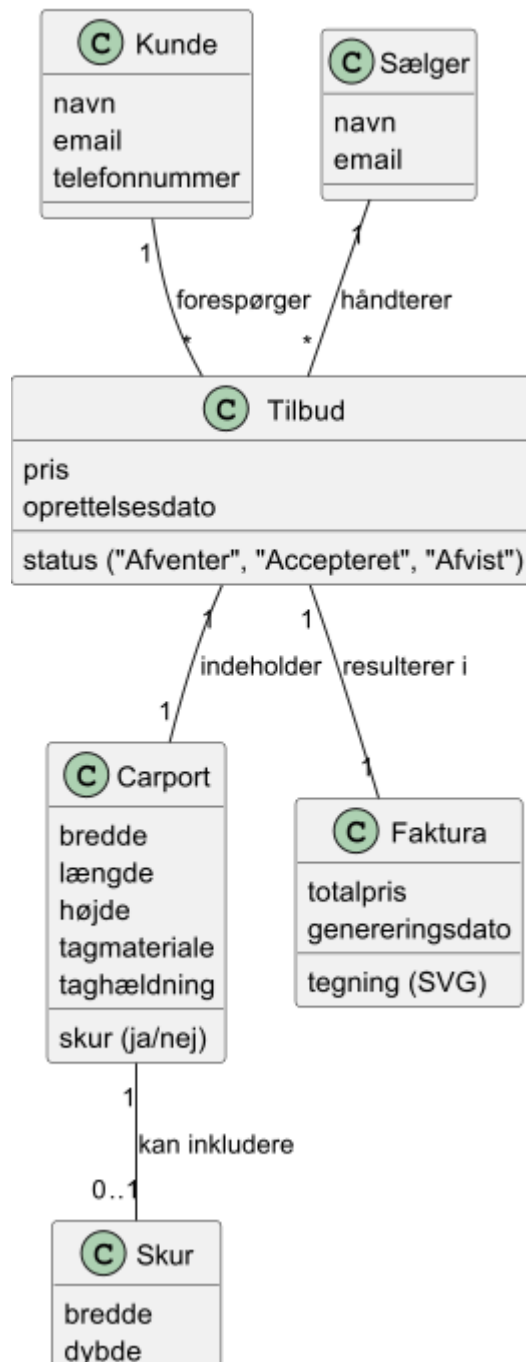


Fig 12. Domænemodel v1

User Stories

[Gå tilbage](#)

US-1 Index

[Edit title](#)

Draft

montif16 opened last month

montif16 3 weeks ago (edited)

[Edit](#)

User story

Som kunde eller admintrator

Kan jeg skrive en URL og komme ind på frontpagen af vores hjemmeside

Sådan at jeg har mulighed for at se carporte eller evt. logge ind

Acceptance criteria

Givet at jeg som kunde er kommet ind på den rigtige url så kan jeg se en hjemmeside.

når jeg er inde på frontpagen af hjemmesiden

så kan jeg se de forskellige muligheder som jeg har adgang til. Det kan være f.eks. login, opret bruger eller carporte.

Tasks

Der skal laves en HTML index side i vores kode, som kan vise både en login knap men også en en carport knap.

Der skal tilføjes link tag både for login knappen og carporte som peger på nogle HTML pages som f.eks. /Carporte eller /login

Assignees



RHoejholt

Status

Done


Priority

P0

US-2 Create user #6

Edit

Closed#7RHoejholt/Fog-ProjektetPublic



montif16 opened 3 weeks ago

User story


Som kunde eller admin kan jeg skabe en bruger
For at senere kunne logge ind eller betale for en carport



Acceptance criteria


Givet at jeg ikke er logget ind og trykker på opret bruger menupunktet, når jeg har udfyldt opret bruger formularen og trykket på opret bruger knappen Så bliver der oprettet en bruger, som jeg kan logge ind med senere


Tasks


Der skal laves en HTML fil med en opret bruger formular
Formularen, når man trykker på opret bruger knappen, skal sende et post-request til /createuser endpoint.
Dataen fra fra post-request skal opsamles i controlleren og sendes til createuser metoden i UserMapper
Hvis der kastes in exception så skal der vises en fejl besked ved hjælp af ctx.attribute("message","Besked til brugeren");






RHoejholt added this to  **Fog Projektet** last month

montif16 self-assigned this 3 weeks ago


montif16 converted this from a draft issue 3 weeks ago

montif16 added enhancement 3 weeks ago

montif16 linked a pull request that will close this issue [US-2 create user #7](#) 3 weeks ago

RHoejholt closed this as **completed** by moving to Done in  **Fog Projektet** 3 weeks ago


Assignees

montif16

Labels

enhancement

Projects

 **Fog Projektet**

Status Done


Priority P1

Time estimate Small - 1-3 hrs

Milestone

No milestone

Development



US-2 create user
RHoejholt/Fog-Projektet

Notifications


Unsubscribe

You're receiving notifications because you're subscribed to this thread.

Participants



→ Transfer issue

 Lock conversation

US-4 carport dimensioner

[Gå tilbage](#)

US-4 carport dimensioner #12

Closed

RHoejholt/Fog-ProjektetPublic

montif16

opened 3 weeks ago

...

User Story

Som bruger eller admin vil jeg kunne indtaste ønskede dimensioner (bredde, længde, højde) på carporten, så kunden kan få deres ønskede carport med skræddersyede dimensioner.

Acceptance criteria

Når man som bruger eller admin når til punktet hvor man kan vælge sine egne dimensioner på carporten skal man kunne indtaste sine egne mål, inden for visse grænser.

Tasks

En html side skal findes, med alle justeringsmuligheder for carporten. Carportens dimensioner skal være en af mulighederne, gerne hvor der er min og max grænser for dimensionerne, som evt. kunne hentes fra en database, så det ikke er hardcoded ind i systemet.

Valget skal gemmes så dataet kan tilgås senere, når styklisten og SVG genereres.

RHoejholt added this to Fog Projektet last month

montif16 self-assigned this 3 weeks ago

montif16 converted this from a draft issue 3 weeks ago

montif16 moved this from In Progress to Review in Fog Projektet 3 weeks ago

montif16 closed this as completed 3 weeks ago

github-project-automation moved this from Review to Done in Fog Projektet 3 weeks ago

Assignees

montif16

Labels

No labels

Projects

Fog Projektet

StatusDone

PriorityP1

Time estimateMedium - 3-6 hrs

Milestone

No milestone

Development

12-us-4-carport-dimensioner

RHoejholt/Fog-Projektet

Notifications

Customize

Unsubscribe

You're receiving notifications because you're subscribed to this thread.

Participants

→ Transfer issue

🔒 Lock conversation

📌 Pin issue

US-5 halv- eller fuldt skur #18

Edit

Closed

#21

RHoejholt/Fog-Projektet Public



montif16 opened 2 weeks ago

US-2 Som sælger vil jeg kunne vælge mellem et halv- eller fuldt skur, så kunden får den ønskede opbevaringsplads

User Story

Som bruger eller admin vil jeg kunne vælge om hele carporten skal være et redskabsrum/skur, eller om skuret ikke skal optage alt pladsen i carporten, så kunden kan få deres ønskede carport eller skur

Acceptance criteria

Når man som bruger eller admin når til punktet hvor man kan vælge om man vil have skur eller carport skal man kunne klart vælge hvilken udgave man ønsker

Tasks

En html side skal findes, med alle justeringsmuligheder for carporten. Der skal være et binært valg, om det skal være et skur eller ej. Det kan godt hardcodes, da det ikke er en ting der forventes at blive ændret i fremtiden.

Valget skal gemmes så dataet kan tilgås senere, når styklisten og SVG genereres.



RHoejholt added this to Fog Projektet last month

montif16 converted this from a draft issue 2 weeks ago

montif16 self-assigned this 2 weeks ago

montif16 moved this from Todo to In Progress in Fog Projektet 2 weeks ago

montif16 moved this from In Progress to Review in Fog Projektet 2 weeks ago

montif16 linked a pull request that will close this issue US-5 skur #21 2 weeks ago

montif16 closed this as completed 2 weeks ago

github-project-automation moved this from Review to Done in Fog Projektet 2 weeks ago

Assignees

montif16

Labels

No labels

Projects

Fog Projektet

Status Done

Priority P1

Time estimate Small - 1-3 hrs

Milestone

No milestone

Development

US-5 skur
RHoejholt/Fog-Projektet

Notifications

Customize

Unsubscribe

You're receiving notifications because you're subscribed to this thread.

Participants



Transfer issue

Lock conversation

Pin issue

US-6 tagmateriale #15

[Edit](#)[Closed](#)[RHoejholt/Fog-Projektet](#) [Public](#)

montif16 opened 3 weeks ago

...

User Story

Som bruger eller admin vil jeg kunne indtaste ønskede tagmateriale på carportens tag, så kunden kan få deres ønskede tagtype.

Acceptance criteria

Når man som bruger eller admin når til punktet hvor man kan vælge sine tagmateriale skal man kunne fra en dropdown vælge tagmateriale

Tasks

En html side skal findes, med alle justeringsmuligheder for carporten, tagmateriale skal hentes fra en database

Valget skal gemmes så dataet kan tilgås senere, når styklisten og SVG genereres.



- [RHoejholt](#) added this to [Fog Projektet](#) last month
- [montif16](#) converted this from a draft issue 3 weeks ago
- [montif16](#) self-assigned this 3 weeks ago
- [montif16](#) moved this from In Progress to Review in [Fog Projektet](#) 3 weeks ago
- [RHoejholt](#) moved this from Review to Done in [Fog Projektet](#) 3 weeks ago
- [RHoejholt](#) closed this as [completed](#) by moving to Done in [Fog Projektet](#) 3 weeks ago

Assignees



[montif16](#)

Labels



No labels

Projects



Fog Projektet

Status [Done](#)

Priority [P1](#)

Time estimate [Small - 1-3 hrs](#)

Milestone



No milestone

Development



15-us-6-tagmateriale
[RHoejholt/Fog-Projektet](#)

Notifications

[Customize](#)[Unsubscribe](#)

You're receiving notifications because you're subscribed to this thread.

Participants



US-7 spær og rem #19

Closed

#23

RHoejholt/Fog-Projektet

Public

montif16 opened 2 weeks ago

User Story

Som bruger eller admin vil jeg kunne indtaste ønskede trætype for spær og rem, så kunden kan få deres ønskede trætype.

Acceptance criteria

Når man som bruger eller admin når til punktet hvor man kan vælge sine trætypee skal man kunne fra en dropdown vælge sine ønskede trætyper for spær og rem

Tasks

En html side skal findes, med alle justeringsmuligheder for carporten. Træmateriale-listen skal hentes fra en database

Valget skal gemmes så dataet kan tilgås senere, når styklisten og SVG genereres.

RHoejholt added this to Fug Projektet last month

montif16 converted this from a draft issue 2 weeks ago

montif16 self-assigned this 2 weeks ago

montif16 linked a pull request that will close this issue [US-7 spær og rem #22](#) 2 weeks ago

montif16 moved this from In Progress to Review in Fug Projektet 2 weeks ago

montif16 removed a link to a pull request [US-7 spær og rem #22](#) 2 weeks ago

montif16 linked a pull request that will close this issue [US-7 Spær og rem #23](#) 2 weeks ago

montif16 closed this as **completed** 2 weeks ago

github-project-automation moved this from Review to Done in Fug Projektet 2 weeks ago

Assignees

montif16

Labels

No labels

Projects

Fug Projektet

Status Done

Priority P1

Time estimate Small - 1-3 hrs

Milestone

No milestone

Development

US-7 Spær og rem

RHoejholt/Fog-Projektet

Notifications

Customize

Unsubscribe

You're receiving notifications because you're subscribed to this thread.

Participants

Transfer issue

Lock conversation

Pin issue

51

US-8 redskabsrum #20

Closed

RHoejholt/Fog-ProjektetPublic

montif16

opened 2 weeks ago

US-5 Som sælger vil jeg kunne tilføje et redskabsrum med justerbare bredde og dybde samt vælge beklædning og gulvtype

User Story

Som bruger eller admin vil jeg kunne tilføje et redskabsrum med justerbare bredde og dybde samt vælge beklædning og gulvtype Så kunden kan få din ønskede skræddersyede carport

Acceptance criteria

Når man som bruger eller admin når til punktet hvor man foretager valg omkring redskabsrum skal man kunne indtaste sine egne mål, inden for visse grænser, og vælge beklædning og gulvtype.

Tasks

En html side skal findes, med alle justeringsmuligheder for carporten. Redskabsrum skal være en af mulighederne, gerne hvor der er min og max grænser for dimensionerne, som evt. kunne hentes fra en database, så det ikke er hardcoded ind i systemet. Materialer for gulv og beklædning skal hentes fra en database

Valget skal gemmes så dataet kan tilgås senere, når styklisten og SVG genereres.

RHoejholt

added this to Fog Projektet last month

montif16

converted this from a draft issue 2 weeks ago

montif16

self-assigned this 2 weeks ago

montif16

moved this from In Progress to Backlog in Fog Projektet 2 weeks ago

montif16

closed this as completed 5 days ago

github-project-automation

moved this from Backlog to Done in Fog Projektet 5 days ago

Assignees

montif16

Labels

No labels

Projects

Fog Projektet

StatusDone

PriorityP2

Time estimateSmall - 1-3 hrs

Milestone

No milestone

Development

20-us-8-redskabsrum

RHoejholt/Fog-Projektet

Notifications

Unsubscribe

You're receiving notifications because you're subscribed to this thread.

Participants

Transfer issue

Lock conversation

Pin issue

52

US-9 styklister #13

[Edit](#)[Closed](#)[RHoejholt/Fog-Projektet](#) [Public](#)

RHoejholt opened 3 weeks ago · edited by RHoejholt

Edits · ⋮

User story

Som admin
vil jeg kunne downloade eller få vist en detaljeret styklister
så kunden har alle nødvendige komponenter til at bygge carporten.

Acceptance criteria

Givet at alle specifikationer er indtastet korrekt og kunden har betalt
og jeg går til sidste trin i processen ved at trykke f.eks. "beregnet styklister"
bliver der vist en styklister, som evt. kan downloades

Tasks

Der skal laves en HTML fil som kan vise en liste over alle nødvendige deles navne, antal og pris.
Den skal generere det ud fra en ctx der er givet med, så den kan finde de korresponderende data fra databasen og vise dem fra
listen, gennem en mapper klasse der kan oversætte databasen til java objekter, som siden hen derefter kan bearbejde
Brugernavn, dato og totalpris kunne også indgå på siden



- RHoejholt** added this to [Fog Projektet](#) last month
- RHoejholt** self-assigned this 3 weeks ago
- RHoejholt** converted this from a draft issue 3 weeks ago
- RHoejholt** moved this from In Progress to Review in [Fog Projektet](#) 4 days ago
- RHoejholt** moved this from Review to Done in [Fog Projektet](#) 3 days ago
- RHoejholt** closed this as [completed](#) by moving to Done in [Fog Projektet](#) 3 days ago

Assignees

RHoejholt

Labels

No labels

Projects

Fog Projektet

Status **Done**

Priority **P1**

Time estimate **Extra Large - 3-7 da...**

Milestone

No milestone

Development

[Create a branch](#) for this issue or link a pull request.

Notifications

[Customize](#)

[Subscribe](#)

You're not receiving notifications from this thread.

Participants



→ [Transfer issue](#)

[Lock conversation](#)

[Pin issue](#)

US-10 Admin-sider på index siden #26

[Edit](#)[Closed](#)

#25

RHoehholt/Fog-Projektet [Public](#)

montif16 opened 2 weeks ago

User story

Som administrator
kan jeg tilgå mine administrator redskaber
Når jeg er logget på skal jeg kunne se mine redskaber, som jeg skal bruge til beregning af carporte eller lign.

Acceptance criteria

Givet at jeg er logget ind som administrator
når jeg kigger på index siden
så kan jeg se alle mine redskaber og klikke på dem for at åbne siderne

Tasks

Det kræver først at US-3 Login og US-1 Index er lavet.
Der skal laves nogle tabs på index siden af vores hjemmeside
Når man klikker på en tab, så skal det linke til de redskab som man gerne tilgå som administrator.
Det skal kunne åbne en administrator page som f.eks. "Quick-byg" hvor administratoren kan bruge programmet.
Der skal sendes request til nogle metoder som tjekker om man er administrator, hvis man ikke er så skal man blive afvist for at se redskaberne.
Vi skal forhindre at en bruger bare kan indtaste en URL og få adgang til administrator redskaberne.



- RHoehholt added this to Fog Projektet last month
- montif16 converted this from a draft issue 2 weeks ago
- montif16 linked a pull request that will close this issue [Fragment + mere #25](#) 2 weeks ago
- montif16 self-assigned this 2 weeks ago
- montif16 moved this from Todo to Review in Fog Projektet 2 weeks ago
- montif16 closed this as completed 2 weeks ago
- github-project-automation moved this from Review to Done in Fog Projektet 2 weeks ago

Assignees

montif16

Labels

No labels

Projects

Fog Projektet

Status [Done](#)

Priority [P2](#)

Time estimate [Large - 1-2 days](#)

Milestone

No milestone

Development

[Fragment + mere](#)
RHoehholt/Fog-Projektet

Notifications

[Customize](#)[Unsubscribe](#)

You're receiving notifications because you're subscribed to this thread.

Participants

[Transfer issue](#)[Lock conversation](#)[Pin issue](#)

US-11 kompatibilitetstjek

Draft

RHoejholt opened last month

RHoejholt last week (edited)

Edit

US-9 Som sælger vil jeg blive advaret, hvis de indtastede dimensioner eller specifikationer ikke er kompatible, så jeg kan rette fejl, inden jeg præsenterer løsningen for kunden.

User Story

Som kunde
Kan jeg bestille en carport med special mål
Når jeg indtaster de ønskede mål,
så kunne jeg godt tænke mig at systemet kenkendte gyldige mål og gav beskeder om, hvad der er muligt

Acceptance criteria

Givet at jeg bestiller en carport med special mål
Når jeg indtaster mine ønskede mål
Så får jeg en fejlbesked hvis de ikke kan lade sig gøre, men også foreslag til hvad der kan lade sig gøre.
Hvis en mulighed ikke findes, så kan ønsket tvinges igennem alligevel og en dialog mellem sælger og kunde kan starte.

Tasks

Når en bruger bestiller en carport med special mål, så skal der tjekkes om det er en gyldig caport.

Hvis ikke det er en gyldig carport, så skal der sendes en fejlbesked tilbage til brugeren.
Man kunne godt tænke sig at nogle parametre ikke fungerer med andre, vi kan derfor hjælpe kunden med foreslag til en gyldig carport.

Foreslagene kan bare komme ved hjælp af cbx.attribute("message", "Fejlmeddelse") og så kan kunden selv rette i deres formular.
Der skal også sørges for at formularen ikke bliver sendt til sælgeren, hvis en fejlbesked bliver sendt.

Der skal også laves en knap hvor kunden kan gennemtvinge ordren alligevel. Der kan være en situation hvor kunden ikke længere har lyst til at rode i systemet og bare gerne vil i dialog med sælgeren. Hvis dette er tilfælde, så skal der også være kode som tilkendegiver overfor sælgeren, at denne ordre altså ikke er en gyldig carport.

Assignees

Add assignees...

Status

Backlog

Priority

P3

Time estimate

Large - 1-2 days

Convert to issue

Copy link in project

Archive

Delete from project

US-12 Mail

Draft

montif16 opened last month

montif16 last week (edited)

Edit

User story

Som administrator
Kan jeg klikke på en mail knap og komme til et mail system
Når jeg er logget på som administrator og klikket på mail, så kan jeg se de ordre og mails der er sendt fra kunder.

Acceptance criteria

Givet at jeg er logget på som administrator.
Når jeg klikker på mail knappen
Så kommer jeg ind på en side med mails fra kunder

Tasks

Der skal laves en HTML page, hvor en administrator skal kunne modtage mails fra kunder men også skrive en mail tilbage til en kunde.
Sælgeren kan have en dialog med kunden igennem dette system.

Hvis muligt kunne de være godt at have et sted hvor email om ordre bliver modtaget og et sted hvor dialogen imellem kunden kan foregå. Vi vil bruge emailen om odren til at sende til vores beregningssystem men også give et overblik for sælgeren om, hvad det er for en kunde/ordre han er igang med lige nu.

Der skal også tjekkes at når man går in på /mail så bliver der tjekket om man er en administrator, hvis man ikke er, så skal man ikke kunne få adgang til mailsystemet.

Opgaven inkludere både backend code og frontend HTML/CSS

Assignees

montif16

Status

Backlog

Priority

P1

Time estimate

Large - 1-2 days

Convert to issue

Copy link in project

Archive

Delete from project

55

US-13 Sende mail til kunde

Draft

montif16 opened last month

Edit title

montif16 last week (edited)

Edit

User story

Som admintrator
Kan jeg sende email til kunder.
Når jeg er logget på som administrator og er inde på mail systemet, så kan jeg sende emails tilbage til de ordre der er placeret.

Acceptance criteria

Givet at jeg er adminstrator og logget ind
Når jeg trykker på en send email knap
Så kan jeg skrive emails og sende dem til den email som har lavet ordren

Tasks

Dette er en del af vores mail funktion.

En adminstrator skal kunne trykke på en knap og dermed sende en email til de ordre han har i systemet.
Der skal laves en UI og en boks hvor en adminstrator kan skrive en besked til kundens email.

Når send email knappen bliver klikket, så skal der henvises til en side /sendEmail

Der skal også oprettets backend kode som tager den data der bliver sendt fra administratoren. En metode skal modtage den tekst som administratoren skriver og sende den tilbage som en email til kunden.

Det skal ikke være et full fletched system men mere et proof of concept. Mail systemet må godt være meget simpelt da vi skal begrænse hvad vi bruger tid på.

Assignees

montif16

Status

Backlog

Priority

P3

Time estimate

Medium - 3-6 hrs

Convert to issue

Copy link in project

Archive

Delete from project

US-14 Hente data fra mail

Draft

montif16 opened last month

Edit title

montif16 last week (edited)

Edit

User Story

Som adminstrator
Kan jeg hente data fra de email jeg motager.
Når jeg er inde i "Quick byg" systemet, så kan jeg hente data fra email systemet
Sådan at jeg ikke manuelt skal bruge tid på de forskellige systemer

Acceptance criteria

Givet at jeg er inde på "Quick byg" systemet
Når jeg trykker på den knap, så kan jeg vælge en email fra mit email system
Så indsætter den data ind i mit "Quick byg" system

Tasks

Dette kunne umiddelbart være en lidt større opgave
Løsningen har jeg ikke helt ved hånden da jeg skriver dette kort.

Idéen med dette er, at en sælger skulle kunne reducere hans arbejde.
Han skal kunne nemt ved hjælp af en knap, kunne populate fields inde i hans quick byg system sådan at han ikke manuelt skal gå imellem email og "Quick byg" systemet.
Det er en metode som skal kunne gennemlæse en email og derved automatisk sætte data ind. Den skal derfor kunne læse en email, relatere det til data fra databasen og bruge det data i quick byg systemet.

Assignees

montif16

Status

Backlog

Priority

P3

Time estimate

Extra Large - 3-7 days

Convert to issue

Copy link in project

Archive

Delete from project

56

US-15 Ændre data i databasen (WIP)

Draft

montif16 opened last month

montif16 2 days ago (edited)

Edit

User Story

Acceptance criteria

Tasks

Have en knap, hvor de forskellige muligheder i quick byg systemet kan indsættes, ændres eller slettes

Assignees

montif16

Status

Backlog

Priority

P3

Time estimate

Large ~ 1-2 days

○ Convert to issue

📄 Copy link in project

🗂 Archive

🗑 Delete from project

Fig 13. User Stories