

# Fog Carports 2. semesterprojekt

**Marwa** - Hold B

cph-me319@cphbusiness.dk

**Allan** - Hold B

cph-ac326@cphbusiness.dk

**Marlene** - Hold B

cph-mb910@cphbusiness.dk

**Jantie** - Hold A

cph-jl398@cphbusiness.dk

Projektet startede 22/4/2024 og blev afleveret d. 24/05/2024

Links:

- Link til GitHub repository: <https://github.com/Marwamn/Fog-projektet>
- Link til demovideo: <https://youtu.be/WQyXpeNdI-4>
- Link til kørende version af deres website på Digital Ocean (en deployet version):  
<http://157.230.113.224:7070>

	<b>Admin</b>	<b>Kunde</b>
<b>Login navn:</b>	admin@admin.dk	kunde@kunde.dk
<b>Kodeord:</b>	1234	1234

# Indholdsfortegnelse

<b>Indledning.....</b>	<b>4</b>
<b>Baggrund .....</b>	<b>4</b>
<b>Interessentanalyse.....</b>	<b>4</b>
Indflydelse/medvirken-matrixen.....	7
Risikoanalyse: .....	8
<b>Teknologi valg.....</b>	<b>10</b>
Udviklingsværktøjer: .....	10
Webteknologier: .....	10
<b>Krav .....</b>	<b>12</b>
Aktivitetsdiagrammer .....	12
AS-IS: .....	12
TO-BE: .....	13
User stories: .....	14
<b>Domæne model og EER diagram .....</b>	<b>17</b>
Domænemodeller.....	17
EER diagrammer .....	19
Relationer .....	19
så relationerne er:.....	20
<b>Navigationsdiagram &amp; Mockups .....</b>	<b>22</b>
Mockups .....	22
Navigationsdiagram .....	23
<b>Valg af arkitektur .....</b>	<b>24</b>
Model Package: .....	24
View packages: .....	24
css: .....	24
html: .....	24

<b>Controller packages:</b> .....	<b>25</b>
<b>Singleton:</b> .....	<b>26</b>
<b><i>Særlige forhold</i></b> .....	<b>26</b>
<b><i>Udvalgte kodeeksempler</i></b> .....	<b>27</b>
<b>Material Mapper:</b> .....	<b>27</b>
<b>Calculator:</b> .....	<b>28</b>
<b>ShowOrderLinesCustomer:</b> .....	<b>29</b>
<b><i>Status på implementering</i></b> .....	<b>30</b>
<b><i>Kvalitetssikring (test)</i></b> .....	<b>30</b>
<b>Automatiserede Tests</b> .....	<b>30</b>
Unit- og Integrationstests: .....	30
Coverage test: .....	31
User Acceptance test .....	32
<b><i>Proces</i></b> .....	<b>34</b>
<b>Arbejdsprocessen faktisk</b> .....	<b>34</b>
<b>Arbejdsprocessen reflekteret</b> .....	<b>36</b>

# Indledning

Projektet består af en webapplikation, som er bygget i Java, HTML og CSS. Projektet er også tilkoblet en droplet, hvor vi har en delt database på en PostgreSQL-server. Opgaven lyder på at lave en carport-butik, hvor man kan bestille carporte med ens eget design. Vi startede med at lave en domænemodel, og ud fra domænemodellen prøvede vi at lave et ERD diagram for vores database. Ud fra vores ERD-diagram prøvede vi at få nogle fornuftige tables og columns, som gav mening at have til vores projekt.

I løbet af projektet har vi prøvet at have meget fokus på at planlægge, diskutere og opdatere hinanden på, hvad vi har fået lavet. En af de ting, som vi gjorde for at gøre dette, var at lave en kanban på vores projekt i GitHub, hvilket gjorde, at vi trinvist kunne få udarbejdet en god opgave. Efter vi havde klargjort vores diagrammer og database, begyndte vi at lave frontend for at få HTML-siderne og CSS i stand. Efter vi havde fået klaret frontend og havde fået et semifærdigt produkt, startede vi på at lave backend, altså Java-koden, hvor vi så skulle få de to til at interagere med hinanden for at få en funktionel webapplikation.

## Baggrund

Vi er blevet bedt om at udvikle et nyt it-system til virksomheden Fog, som er en førende leverandør af byggematerialer og boligforbedringsprodukter. Systemet skal imødekomme kundens behov ved at tillade brugerne at konfigurere og bestille carporte online. Kunderne skal kunne vælge specifikationer som højde og bredde for at designe en carport, der passer præcist til deres behov. Desuden skal kunden efter betaling modtage en stykliste.

## Interessentanalyse

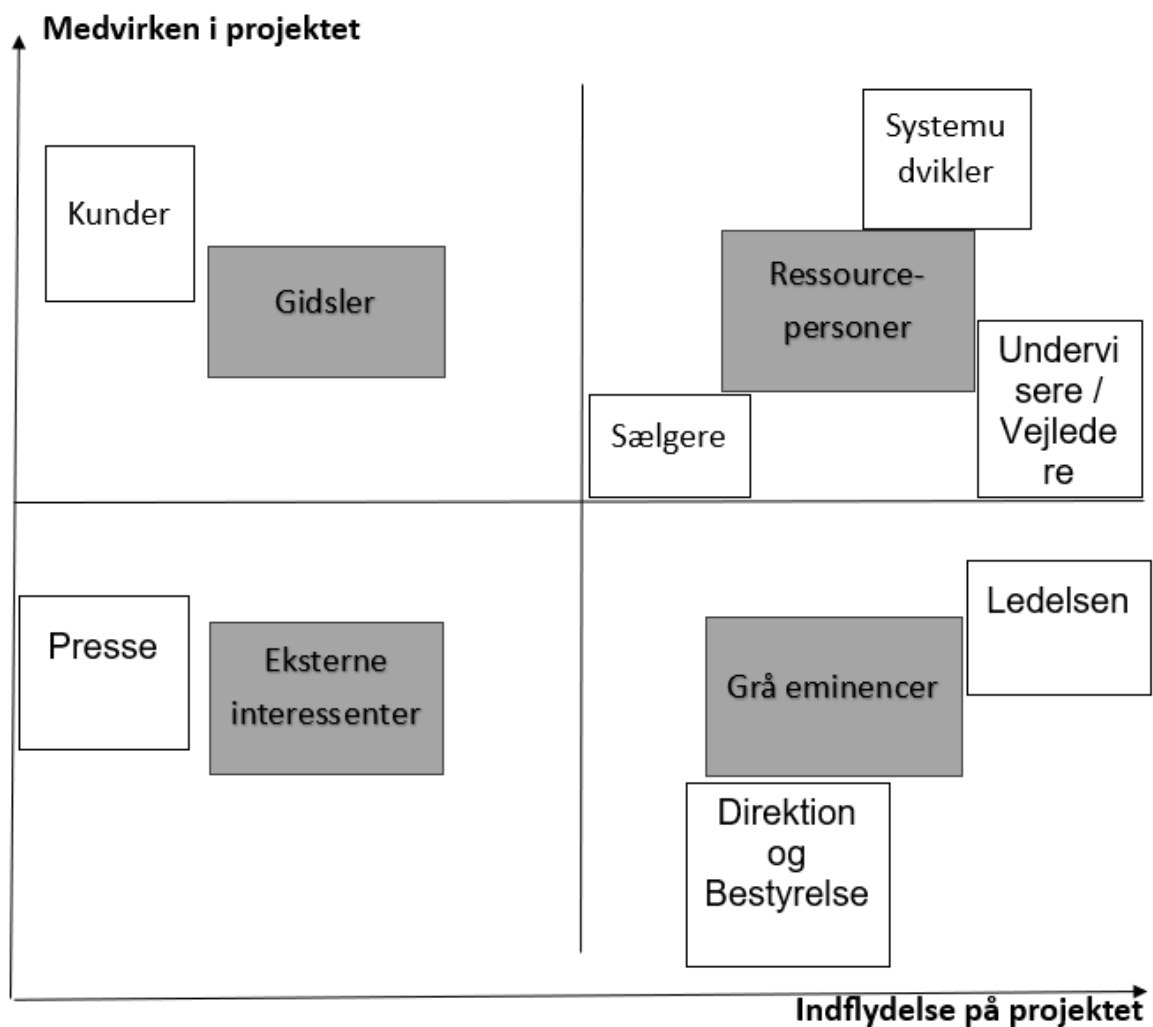
Interessentanalysen og risikoanalysen har ikke hjulpet os med at skrive koden hurtigere eller andet med koden, men den har hjulpet os med at forstå, hvordan det hænger sammen. Altså i forhold til at forstå, hvem de forskellige parter er, som for eksempel systemudviklerne, sælgerne, kunderne og lederne, som vil påvirke et projekt i et firma og få projektet til at ske. Derfor har det hjulpet lidt med at få et overblik over projektet og hvilken rolle de forskellige parter har.

## Analysen

Interessent	Interessenten kan opleve følgende FORDELE ved projektet	Interessenten kan opleve følgende ULEMPER ved projektet	Samlet vurdering af interressentens bidrag/position	Håndtering af interressenten
Systemudvikler	Ekspertise og vejledning i systemudvikling.	Tekniske udfordringer og modstand fra brugere.	Afgørende for at sikre systemets funktionalitet og brugervenlighed.	Tæt samarbejde og kommunikation.
Kunder	Kunderne kan opleve større overskuelighed og bedre brugeroplevelse ved at finde rundt på hjemmesiden ved indførelse af det nye IT-system	Det nye system vil måske ikke fungere optimalt og de finder ikke præcist hvad de vil have, så måske vil de ikke foretage et køb og gå over til en konkurrent	De har ikke stor medvirken til projektet. Men de har stor indflydelse så firmaet ikke går konkurs.	man har medarbejdere som holder styr på kunderne og håndtere dem hvis de fx. har brug for hjælp
Ledelsen	Ledelsen kan få succes, samt inddrage flere kunder og tilfredsstillende dem med et bedre system til firmaet og tjene flere penge.	Det kan være at projektet ikke bliver som forventet og får dårlige resultater som forårsager mindre kunder.	Ledelsen har ikke en stor medvirken da det ikke er dem som udvikler noget, men de har stadigvæk et stort bidrag da det er dem som uddelegere arbejde så det bliver mere effektivt.	De holder møder til at se hvor langt de er nået, samt de får deadlines til at blive færdige med at dele og rapportere, hvis der kommer problemer.
Sælgere	Da det er et nyt og forbedret system, vil det gøre det nemmere for sælgerne at finde rundt og bruge systemet, da de har lært det nye system.	Kan have problemer med at se ordrer hvis der er nogle systemfejl, da det også er et nyt system kan de have problemer med at finde rundt	De har ikke medvirken i projektet, men de har indflydelse da de modtager ordrer og ringer til kunder hvis nødvendigt og sender.	De skal læres på det nye system af systemudviklere

Direktion og Bestyrelse	Øget profitabilitet og styrket brand.	Skade på virksomhedens omdømme ved fiasko.	Afgørende for at definere strategi og ressourcer.	Regelmæssig kommunikation og rapportering.
Presse	Øget omtale og positiv PR.	Negativ omtale eller mangel på dækning.	Kan påvirke virksomhedens omdømme positivt eller negativt.	Tæt samarbejde med PR-afdelingen.
Undervisere/Vejledere	Ekspertise og vejledning i brugerundervisning.	Modstand eller mangel på forståelse fra brugere.	Afgørende for at sikre brugernes forståelse og adoption af systemet.	Tæt samarbejde og kommunikation.

## Indflydelse/medvirken-matrixen



**Kunder:** Har lille indflydelse, og lille medvirken til projektet.

**Systemudviklere:** Har stor indflydelse, og stor medvirken i projektet.

**Sælgere:** Har stor indflydelse, men stor medvirken i projektet.

**Undervisere/Vejledere:** har stor indflydelse, men lidt medvirken i projektet da deres ekspertise bliver brugt.

**Presse:** Kan have en stor indflydelse, men lille medvirken til projektet

**Ledelsen:** Har stor indflydelse, men lille medvirken til projektet.

**Direktion og bestyrelse:** Har stor indflydelse, men lille medvirken til projektet.

## Risikoanalyse:

### Forebyggelse

Risk ID	Risiko	Alvor	Sandsynlighed	Risikoniveau	Plan for forebyggelse/afværgning
1	Forandringsmodstand fra medarbejder omkring det nye IT-system	Tolereres	Muligt	Medium	Kommunikation, træning og involvering af medarbejder
2	Sygdom i udviklingsteamet	Tolereres	Muligt	Medium	Fleksibilitet og luft i arbejdsbyrden, prioritering af opgaver, fælles arbejdsplatform for at kolleger kan tage over
3	Tekniske udfordringer under udvikling af systemet	Uønsket	Muligt	Høj	Samarbejde i projektteamet omkring problemet, identifikation af problem og testing, søge ekstern ekspertise
4	Hvis Martin (Salgschef) er syg eller ikke kan være til stede.	Uønsket	Muligt	Medium	Kommunikere med virksomheden omkring en anden der kan tiltræde i dette tilfælde
5	Manglende integration med eksisterende systemer	Uønsket	Muligt	Høj	Grundig analyse af eksisterende systemer og behov, tidlig inddragelse af IT-afdeling og brugere, test af integration før implementering
6	Utilstrækkelig ressourcer til projektet	Uacceptabelt	Muligt	Høj	Klar ressourceplanlægning, prioritering af opgaver, løbende



					opfølgning og justering af ressourcer, overvejelse af ekstern ressource, anskaffelse hvis nødvendigt.
7	Uforudsete ændringer i lovgivning eller markedsforhold	Uønsket	Muligt	Medium	Kontinuerlig overvågning af ændringer i lovgivningen og markedsforholdene, proaktiv tilpasning af projektplanen, etablering af robuste kommunikationskanaler med interessenterne, tidlig inddragelse af relevante eksperter ved behov
8	Manglende accept fra slutbrugerne	Uønsket	Muligt	Medium	Tidlig inddragelse af slutbrugere, brugerinvolvering gennem hele udviklingsprocessen, klare kommunikation om fordelene ved det nye system, brugertilpasning og -træning
9	Sikkerhedsbrister i det nye system	Uacceptabelt	Muligt	Høj	Gennemførelse af sikkerhedsanalyse og -test, implementering af sikkerhedsforanstaltninger (f.eks. kryptering, adgangskontrol), løbende overvågning og opdatering af sikkerhedsprotokoller
10	Konflikter i teamet eller med vejledere	Tolereres	Muligt	Medium	Implementering af konfliktløsningsst

					rategier, opmærksomhed på teamdynamik, klare kommunikationsk analer.
--	--	--	--	--	-------------------------------------------------------------------------------------

## Teknologi valg

### Udviklingsværktøjer:

#### IntelliJ

- IntelliJ 2023.2.2
- IntelliJ 2023.3.3

#### Java 17

- Corretto-17
- Java 17

#### JDK (Java Development Kit)

- Version 19.0.2

#### PostgreSQL

- PostgreSQL 42.7.2

#### JDBC (Java Database Connectivity)

- JDBC 4.2 API

#### Thymeleaf

- Version 3.1.2.RELEASE

#### Javalin

- Version 6.1.3

### Webteknologier:

#### HTML

- HTML5

#### CSS

- CSS3

**Figma**

- Version 116.17.12

**PgAdmin**

- PgAdmin 8.3

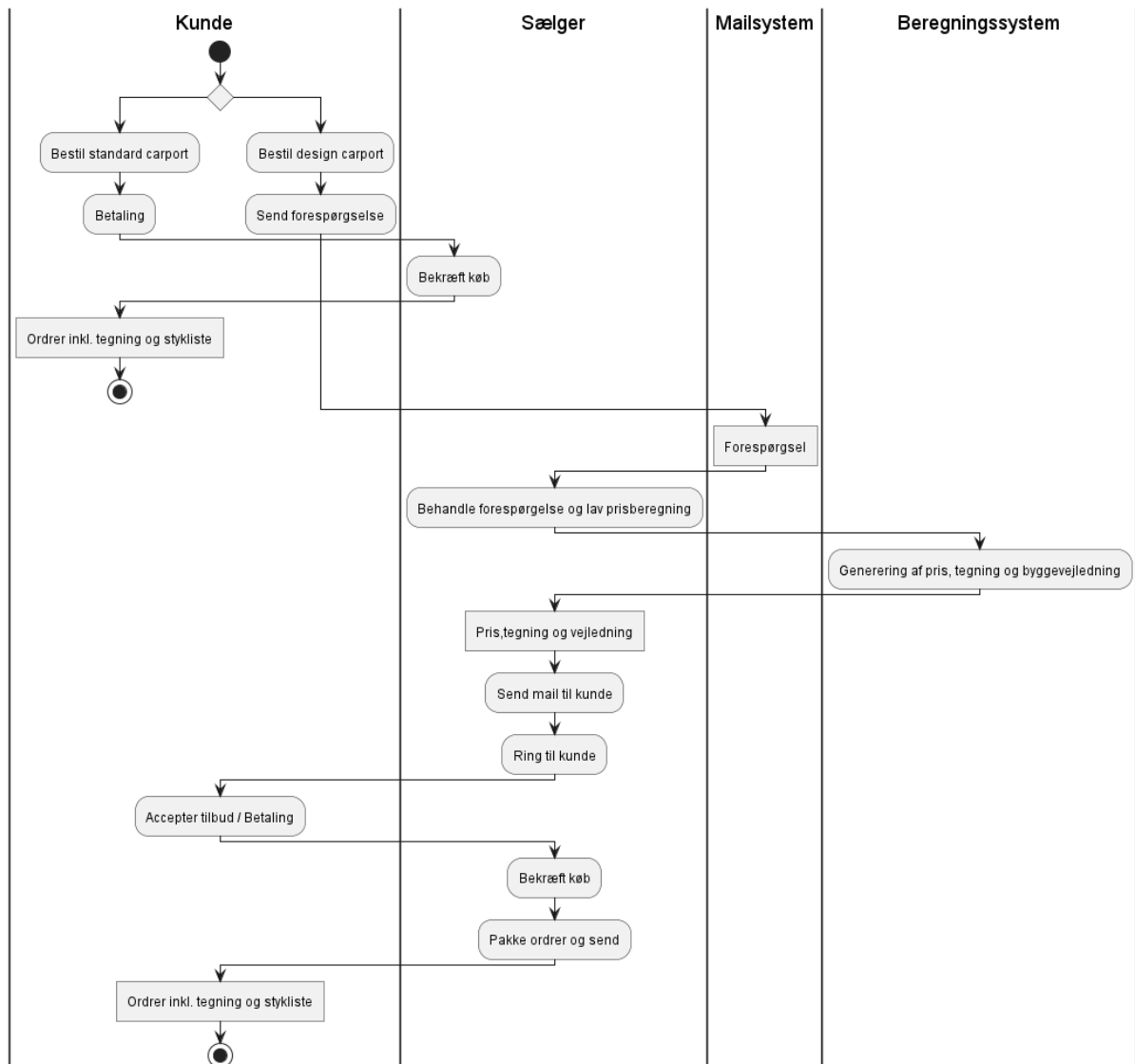
**Docker**

- Docker Engine v25.0.2

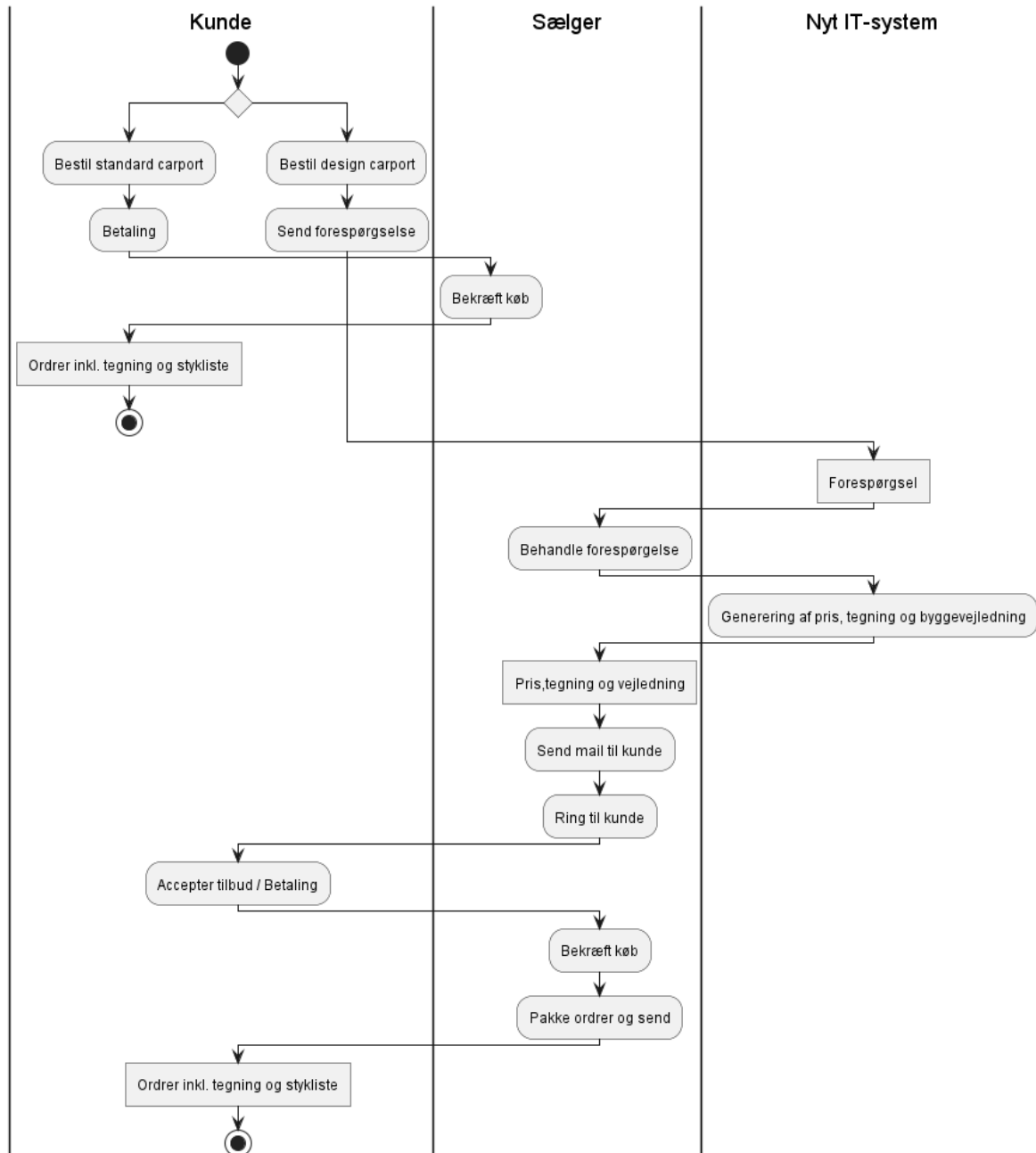
# Krav

## Aktivitetsdiagrammer

AS-IS:



TO-BE:



Forskellen på AS-IS og TO-BE diagrammerne er, at AS-IS består af flere komponenter i forhold til TO-BE. AS-IS består af kunder, sælger, mailsystem og beregningssystem. Den aktuelle arbejdsprocedure indebærer, at sælgeren anvender flere forskellige systemer i sin daglige arbejdsgang.

I modsætning hertil består TO-BE af kunder, sælger og nyt IT-system. Formålet med at implementere det nye IT-system er at samle funktionerne under et system og forenkle arbejdsprocessen. Derudover skal det nye IT-system være mere automatiseret, så det er selve systemet, som udfører de mere repetitive opgaver som beregning osv. Så sælger ikke behøver at indtaste det manuelt i systemet. Dette vil gøre arbejdsprocessen mere effektiv og produktiv, samt frigøre sælgerens tid.

## User stories:

**US-1:** Som kunde ønsker jeg at oprette mig som bruger for at jeg kan betale og gemme en eller flere ordrer.

**Acceptkriterie:** Givet at jeg indtaster mine personlige oplysninger i en formular når jeg klikker på “opret”, så opretter jeg en konto som jeg kan logge ind på.

**Estimat:** small

**US-2:** Som kunde ønsker jeg at kunne logge ind på min bruger for at jeg kan se status på mine bestilte carporte, samt at jeg har mulighed for at bestille en ny carport.

**Acceptkriterie:** Givet at jeg indtaster mine personlige oplysninger i en formular når jeg klikker på “login”, så bliver jeg logget ind på systemet og kan nu få lov at bestille en carport, og se status på mine købte carporte.

**Estimat:** small

**US-3:** Som kunde ønsker jeg at bestille en valgfri carport, for at jeg kan designe den efter mine behov.

**Acceptkriterie:** Givet at jeg vil indtaste mine ønskede mål for længde og bredde på skur og carport, når jeg klikker på “Design nu”, kan jeg udfylde en formular, og derefter klikke på “Send forespørgelse” og indsende min udfyldte formular.

**Estimat:** medium

**US-4:** Som kunde ønsker jeg at kunne se min status for mine ordrer, herunder se styklisten på mine ordrer, for at få de informationer jeg har behov for omkring mine ordrer.

**Acceptkriterie:** Givet at jeg vil se min status, stykliste og tegninger over min carport, når jeg klikker på "status", kan jeg trykke på nogle knapper, så jeg kan se min stykliste og tegninger over mine udvalgte ordrer.

**Estimat:** large

**US-5:** Som kunde ønsker jeg at kunne se en totalpris af min stykliste for at jeg ved hvor meget jeg skal betale for komponenterne af carporten.

**Acceptkriterie:** Givet at jeg vil se totalprisen, når jeg klikker på "status" knappen, så kan jeg se totalprisen for min ordrer.

**Estimat:** medium

**US-6:** Som kunde ønsker jeg at kunne bekræfte min ordre, efter jeg har set totalprisen, for at herefter kunne se min stykliste ud fra min bestemte ordre.

**Acceptkriterie:** Givet at jeg gerne vil bekræfte min ordre så den bliver behandlet når jeg klikker på "Bekræft" så kan jeg se min stykliste for ordren.

**Estimat:** medium

**US-7:** Som kunde ønsker jeg at se en stykliste efter betaling, for at jeg kan se hvad der skal bruges til bygning af min carport.

**Acceptkriterie:** Givet at jeg gerne vil se min stykliste af min købte carport når jeg klikker "Bekræft", så får jeg vist en stykliste af min udvalgte carport så jeg kan bygge carporten.

**Estimat:** large

**US-8:** Som administrator ønsker jeg at se alle ordrer i systemet, for at jeg kan se hvad der er blevet bestilt.

**Acceptkriterie:** Givet at jeg gerne vil se alle ordrer i systemet, når jeg klikker på "Order", så får jeg vist ordrelisten.

**Estimat:** small

**US-9:** Som administrator ønsker jeg at se alle ordrelinjer i systemet, for at jeg kan se hvad der er blevet bestilt ud fra hver ordre.

**Acceptkriterie:** Givet at jeg gerne vil se alle ordrelinjer i systemet, når jeg klikker på “Order”, så får jeg vist ordrelinje listen.

**Estimat:** small

**US-10:** Som administrator ønsker jeg at kunne beregne totalprisen for alle materialer som bliver brugt ud fra styklisten, for at kunne vise kunden hvor meget der skal betales.

**Acceptkriterie:** Givet at jeg gerne vil beregne totalprisen af styklisten, når jeg klikker på “Calculate” så bliver det vist for kunden hvor meget de skal betale for den specifikke ordre.

**Estimat:** medium

**US-11:** Som administrator ønsker jeg at kunne beregne styklisten, for at vise kunden alle materialerne som skal bruges til at bygge carporten.

**Acceptkriterie:** Givet at jeg beregner styklisten, når jeg klikker “Se styklisten”, så genere jeg styklisten så kunderne kan se den.

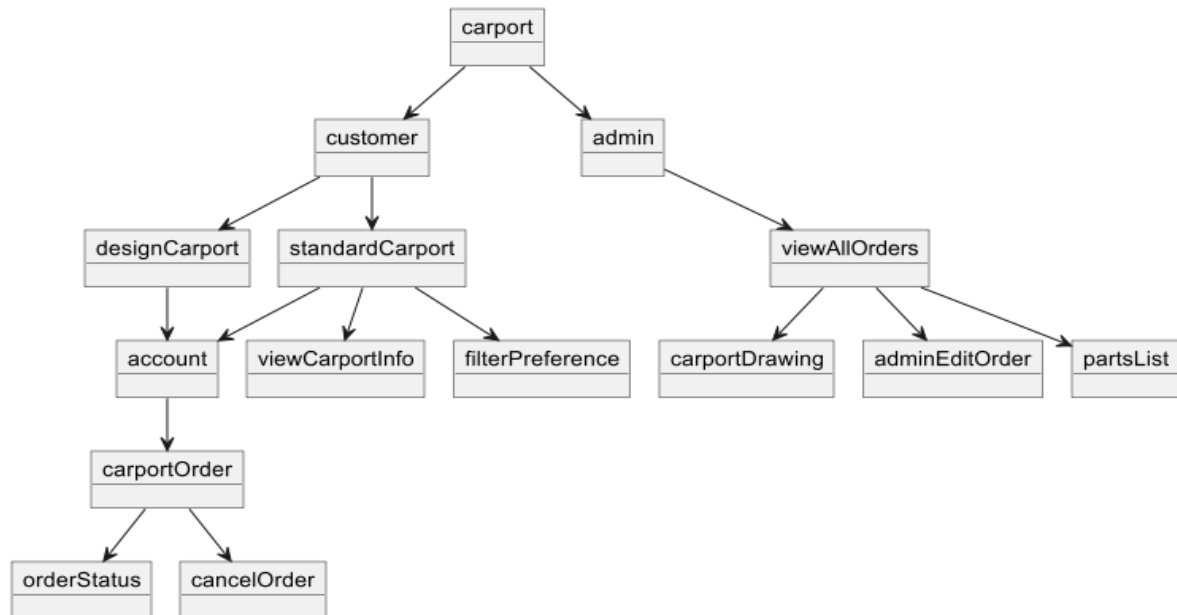
**Estimat:** large



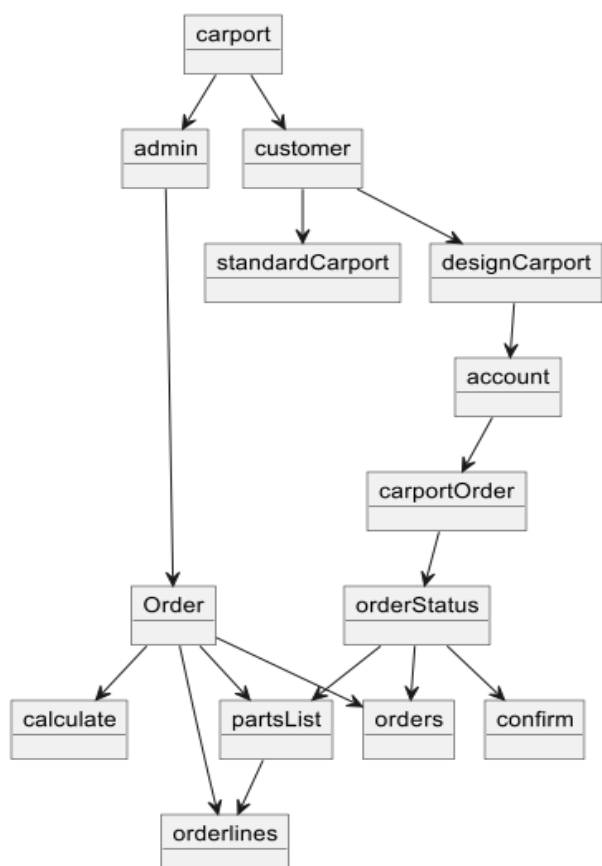
# Domæne model og EER diagram

## Domænenemodeller

Dette var vores domænenemodel, som vi udtænkte i starten af projektet:



Her er den nye domænenemodel, hvor vi foretog ændringer i forhold til den gamle, som vi afsluttede med:



Under projektet har vi foretaget ændringer for admin og kunden. Oprindeligt planlagde vi, at admin skulle have mulighed for at redigere en ordre, men vi lavede ikke funktionen alligevel. Formålet var at give administratorer mulighed for at rette eventuelle fejl, som kunder havde begået, men vi aftalte til sidst ikke at lave det alligevel. Vi droppede SVG-tegningerne, da vi ikke havde tid til at lave dem.

Vi besluttede at lave en totalprisberegner, som administrator kunne bruge til at beregne totalprisen for alle materialerne sammenlagt for ordrelinjer på en ordre. Vi kom også til den beslutning, at lave så administrator kan se alle ordrelinjerne og ordrer på en HTML-side.

Vi fik ikke lavet noget i forhold til standard carporte alligevel, da vi ikke kunne nå det. Derfor reroutede vi den i stedet til Fogs side. Vi ændrede også ordrestatusen, så kunderne først skulle bekræfte deres ordre, før parlisten kunne blive set. Kunden kunne også se selv ordren, dog ikke Status ID eller User ID, da det er lige meget for en kunde at kunne se det.

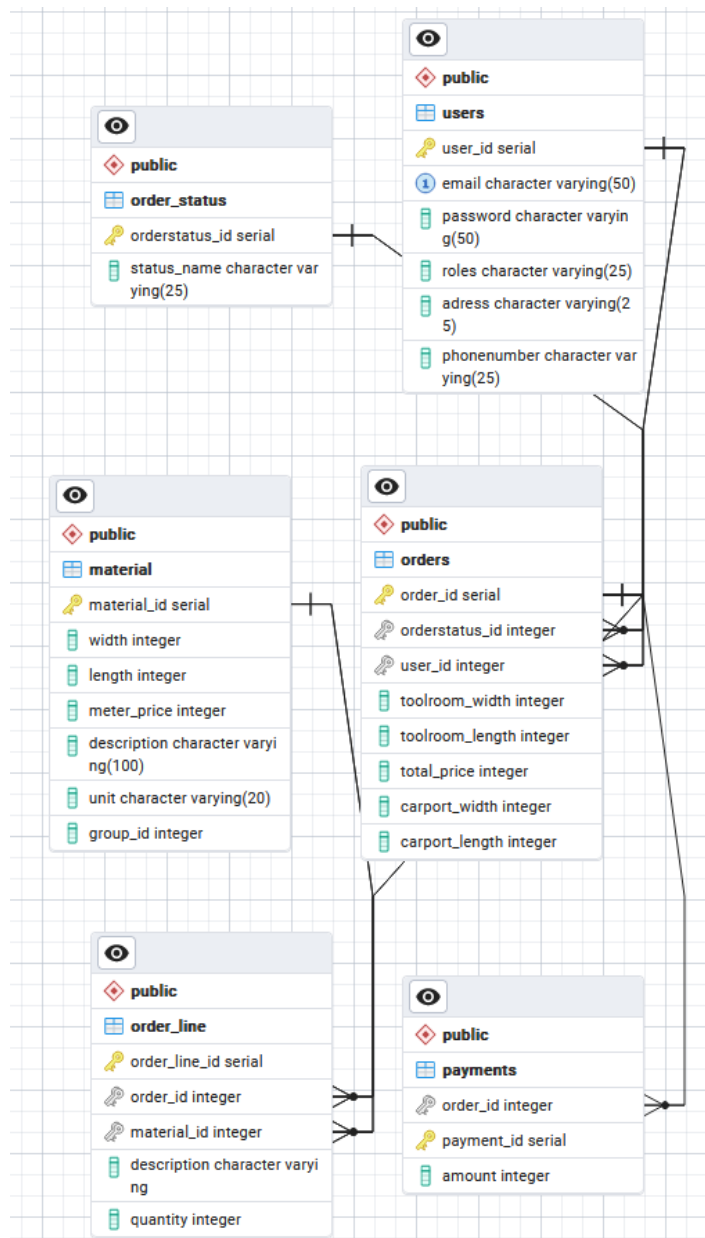
Vi ændrede også det med at kunne annullere en ordre, men vi opgav det, da vi tænkte at det var lidt upraktisk, da efter man har trykket bekræft, har man jo allerede betalt. Men grunden til at vi tænkte det i starten, var hvis der var eventuelle fejl i bestillingen, så kunne kunden annullere bestillingen, så det er også ok at have den funktion, men vi undlod.

## EER diagrammer

Public skema datastruktur, dette skema blev brugt til at lave projektet, altså den generelle kodning.

De fleste af tabellerne er 3. normalform (3NF), hvilket betyder, at de er struktureret for at undgå unødvendig gentagelse af data og sikre dataintegriteten.

I vores users tabel er adressen og telefonnummeret knyttet til useren og ikke til noget andet. Det betyder, at tabellen ikke er i overensstemmelse med 3. normalform. Årsagen var, at vi syntes, at det var lettere for os, da det gjorde det mere enkelt og mindre komplekst. Så vi gjorde det for at lave databasen simpel, hvilket vi synes var en fordel.



## Relationer

- en-til-mange-relation mellem users og orders: en bruger (user) kan placere flere ordrer (orders), men hver ordre tilhører kun en bruger.
- en-til-mange-relation mellem orders og order\_line: en ordre (order) kan have flere ordrelinjer (order\_line), men hver ordrelinje er knyttet til en enkelt ordre.

- en-til-mange-relation mellem order\_status og orders: en status (order\_status) kan være knyttet til flere ordrer (orders), men hver ordre har kun en status.
- en-til-mange-relation mellem material og order\_line: et materiale (material) kan være en del af flere ordrelinjer (order\_line), men hver ordrelinje refererer kun til et materiale.
- en-til-en-relation mellem orders og payments: hver ordre (order) har nøjagtigt en betaling (payment), og hver betaling er knyttet til en enkelt ordre.

så relationerne er:

- en-til-mange-relation mellem users og orders.
- en-til-mange-relation mellem orders og order\_line.
- en-til-mange-relation mellem order\_status og orders.
- en-til-mange-relation mellem material og order\_line.
- en-til-en-relation mellem orders og payments.

En en-til-en-relation mellem orders og payments tabellerne adskiller ordredataen fra betalingsdataen, hvilket forbedrer fleksibiliteten ved at oprette separate tabeller. Dette kan resultere i en mere skalerbar database.

Vi synes det var en god ide ikke at have nogle mange-til-mange-relationer i vores database af tre årsager:

- Simpel vedligeholdelse: Enklere relationer gør det lettere at forstå og opdatere databasen.
- Dataintegritet: Ved at undgå komplekse relationer bevares dataens konsistens og pålidelighed.
- Skalerbarhed: Databasen er mere fleksibel og kan bedre tilpasses fremtidige behov uden at miste overblikket.

I vores database har vi klart definerede og enkle relationer, for at sikre konsistens. For eksempel:

- Ordre er knyttet til brugere via bruger-id.
- Ordre er knyttet til statusser via status-id.
- Ordre er knyttet til betalinger via ordre-id.
- Ordrelinjer er knyttet til ordre via ordre-id og også til materialer via materiale-id.

Disse præcise relationer gør det nemmere at opretholde databasens integritet uden unødvendig kompleksitet.

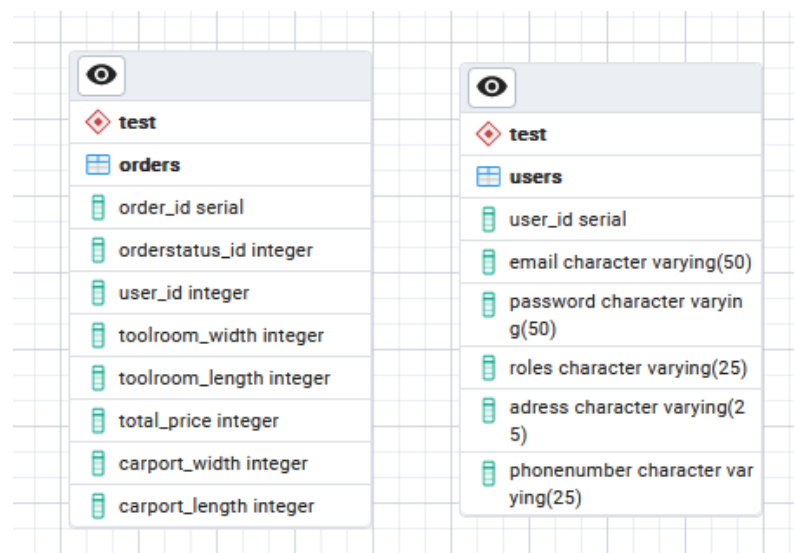
I databasen bruger vi primært automatisk genererede ID'er, kendt som datatypen serial, som nøgler. Dette gør det nemt at administrere og sikre unikke identifikatorer.

Fremmednøgler: Vi inkluderer fremmednøgler for at oprette forbindelser mellem tabellerne og sikre dataintegritet. Dette sikrer, at vores data forbliver forbundet og konsistent.

Constraints: Vi har indført constraints såsom primary key og foreign key constraints for at sikre dataens konsistens og nøjagtighed. Disse hjælper med at forhindre inkonsistens.

### Test skema datastruktur, dette skema blev brugt til integration testningen.

Vi brugte skemaet til integration testning for både OrderMapper og UserMapper. Vi oprettede de nødvendige tabeller korrekt og testede dem. Efter at dette var gjort, testede vi om der kunne tilføjes en ordre til orders-tabellen. Derefter testede vi om der kunne oprettes en bruger i users tabellen. Vi oprettede et separat skema for at undgå at rode med det primære public skema. Dette gjorde, at vi ikke risikerer at påvirke den eksisterende data i vores public skema.



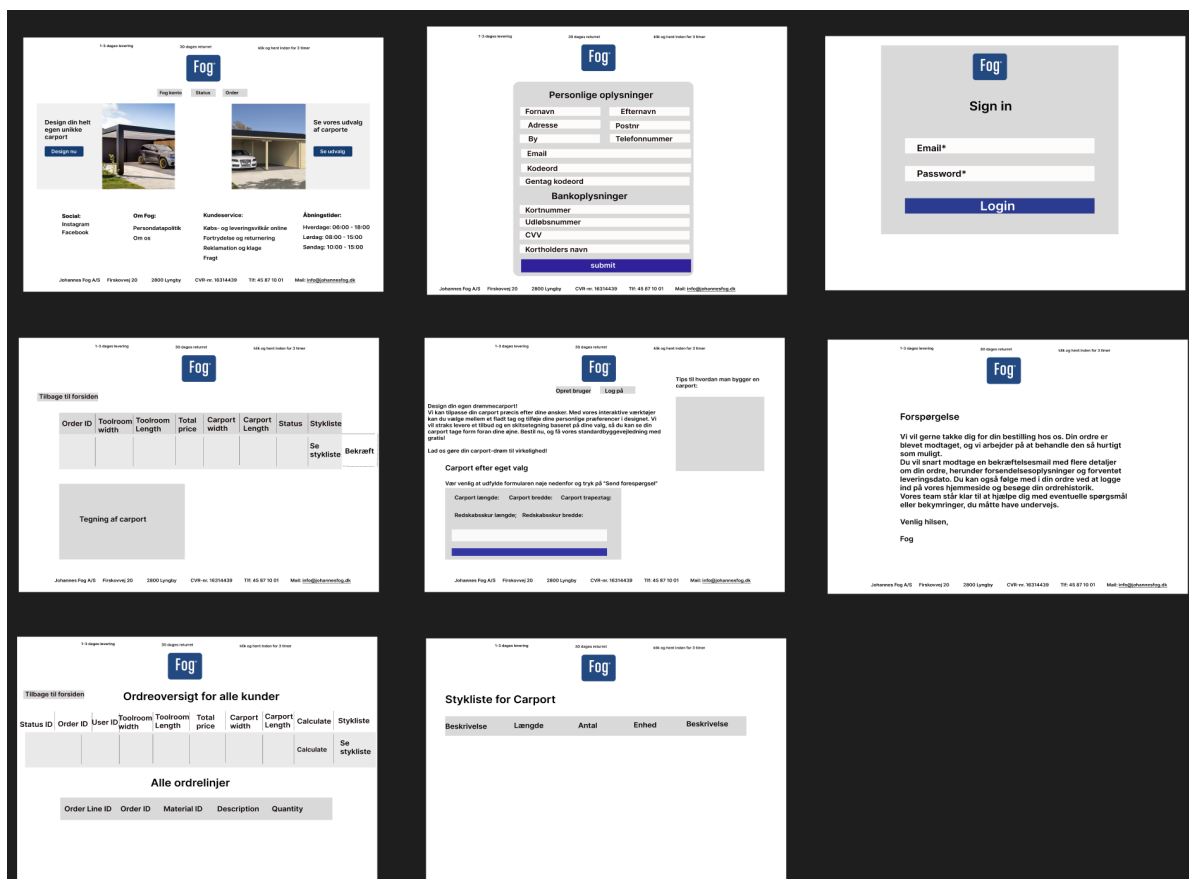
# Navigationsdiagram & Mockups

## Mockups

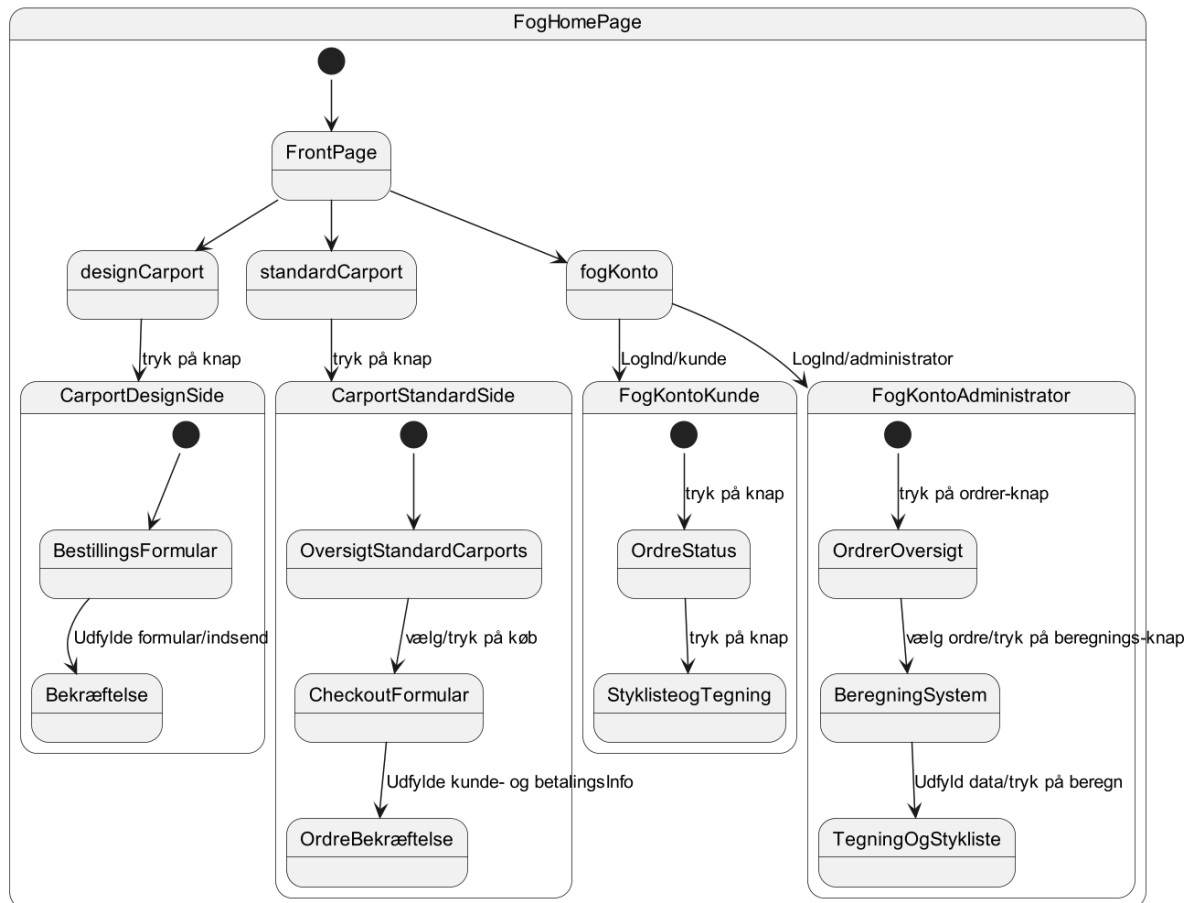
På forsiden kan man vælge mellem to muligheder: "Byg Selv Carport" og "Standard Carport". Når kunden vælger "Byg Selv Carport", har de mulighed for at logge ind eller oprette en ny brugerprofil. Ved oprettelse indtaster kunden sine kontaktoplysninger. Derefter kan kunden indtaste de ønskede mål for sin carport og sende en forespørgsel ved at klikke på "Send forespørgsel". Kunden modtager herefter en besked om, at forespørgslen er sendt.

Efter at have sendt en forespørgsel, kommer kunden ind på forsiden og vælger at klikke på "Fog konto" for at logge ind og se status på deres bestilling eller bestille en ny carport. På denne side kan kunden se deres bestilte ordrer, bekræfte valget af carport, og få et overblik over pris og tegning. Når kunden klikker på "Bekræft", får de adgang til en stykliste.

Når man klikker på "Fog konto", kan man også logge ind som administrator. Som admin kan man klikke på knappen "Ordre" for at se alle kunders ordrer og beregne priser ved at klikke på "Calculate". Admin kan også få adgang til styklister og materialelister for alle ordrer.



# Navigationsdiagram



# Valg af arkitektur

## Model Package:

Attributes som er baseret på vores EER diagram, samt getters og setters som bliver brugt fra attributes, dog PartList var en som vi lavede for at kunne lave partlisten ved at joine tabeller sammen.

- Materialer
- Order
- OrderLine
- OrderStatus
- PartList
- User

## View packages:

css:

Vi har valgt at bruge to css filer som er følgende:

- styles
- styleless

html:

- frontpage

Dette er hovedsiden på hjemmesiden, hvor brugeren starter.

### Undersiderne inde i frontpage

- aboutUS

Her kan brugerne læse om virksomhedens historie, mission og værdier.

- cancellationAndReturns

Information om, hvordan kunder kan afbestille ordrer og returnere produkter.

- personalData



Detaljer om, hvordan virksomheden håndterer kundernes personlige data.

- `personalInformation`

Information om, hvordan kunder kan opdatere deres personlige oplysninger.

- `shipping`

Information om forsendelsesmuligheder, leveringsmetoder og omkostninger.

- `termsAndConditions`

De juridiske vilkår og betingelser for at bruge hjemmesiden og købe produkter.

- `Warranty`

Information om produktgarantier og hvordan man kan gøre krav på dem.

### **Andre vigtige sider**

- `createUser`

En side, hvor nye brugere kan oprette en konto.

- `customerOrder`

Her kan kunder se og administrere deres ordrer.

- `designCarport`

En side, hvor brugerne kan designe deres egen carport.

- `login`

En side, hvor brugere kan logge ind på deres konto.

- `Orderinquiry`

Kunden modtager en ordre forspørgelse efter bestilling

- `orderOverview`

En side, der giver et overblik over alle kundens ordrer.

- `orderStatus`

Her kan kunden se stykliste efter bekræftelse

## **Controller packages:**

Controlleren bestemmer, hvilken HTML der skal routes til, samt opdaterer dataen, som bliver smidt ind i Main for at kunne køre det.

- `OrderController` : indsætter ordre og ordrelinjer og totalpris og router til html side.
- `OrderStatusController` : opdatere ordrestatus samt viser partlist og router til html side.

- UserController : logger ind, samt laver en ny bruger og router til html side.

## Singleton:

Der er kun en enkelt instans af en dataforbindelse for at sikre effektiv ressourceudnyttelse og ensartethed.

- ConnectionPool

Overordnet hjælper arkitekturen med at holde vores projekt organiseret og skalerbart. Ved at vi opdeler systemet i klare og definerede packages, samt anvender et designmønster som Singleton, kan vi nemt vedligeholde, og udvide og teste vores system. Hver af vores packages har en defineret rolle, hvilket gør vores program læsbar og mindre kompleks.

## Særlige forhold

Det som gemmes i sessionen er de valg brugerne taster ind i formularen når de bestiller en carport som, carport\_length, carport\_width, toolroom\_length, toolroom\_width, dette er vores form parameters for at bestille en carport som bliver gemt inde i sessionen. Derudover gemmes brugerne i sessionen når de er logget ind, så sessionen kan holde styr på brugernes tilstand f.eks. om de er logget ind eller ikke, og til at knytte brugerne til deres ordre. Vi har også gemt order\_id inde i sessionen, da order\_id er tilknyttet en specifik stykliste, og til at checke om det specifikke order\_id er betalt, og det beløb passer med det beløb som svarer til totalprisen.

Vi håndterer typisk exceptions ved brug af Database Exception og SQL Exceptions, da vi skal integrere med vores database så vi kan håndtere det og så fejlen bliver vist så vi ved hvor der kommer fejl.

Vi har gjort processen med at oprette en bruger mere sikker, ved at indføre en funktion hvor brugeren skal gentage kodeordet. Dette sikrer at brugeren indtaster det rigtige kodeord, og undgår stavfejl.

Vi har valgt at lave to roller, som er admin og kunder. De har forskellige html sider som, de bliver routet til, og forskellige knapper, som de kan se og tilgå.

## Udvalgte kodeeksempler

### Material Mapper:

```
public class MaterialMapper {  
    7 usages  ▲ AllanChandler +1  
    public static List<Material> getMaterialsByMinLengthAndGroupId(int minLength, int groupId, ConnectionPool connectionPool) throws DatabaseException {  
        List<Material> materials = new ArrayList<>();  
        String sql = "SELECT * FROM material WHERE group_id = ? AND length >= ?";  
        try (Connection connection = connectionPool.getConnection()) {  
            PreparedStatement ps = connection.prepareStatement(sql);  
            ps.setInt( parameterIndex: 1, groupId);  
            ps.setInt( parameterIndex: 2, minLength);  
            ResultSet resultSet = ps.executeQuery();  
            while (resultSet.next()) {  
                int material_id = resultSet.getInt( columnLabel: "material_id");  
                int width = resultSet.getInt( columnLabel: "width");  
                int length = resultSet.getInt( columnLabel: "length");  
                int meterPrice = resultSet.getInt( columnLabel: "meter_price");  
                String description = resultSet.getString( columnLabel: "description");  
                String unit = resultSet.getString( columnLabel: "unit");  
                int group = resultSet.getInt( columnLabel: "group_id");  
                Material material = new Material(material_id, width, length, meterPrice, description, unit, group);  
                materials.add(material);  
            }  
        } catch (SQLException e) {  
            throw new DatabaseException("Could not get materials from the database", e.getMessage());  
        }  
        return materials;  
    }  
}
```

Vi har lavet en SQLException samt DatabaseException i dette eksempel, det der sker når man laver de exceptions:

SQLExceptionen skal blive håndteret da der kan ske fejl under udførelsen af string sql forespørgsel, herefter bliver en DatabaseException også dannet for at give brugeren en fejlmeddelelse, så man kan se hvor fejlen ligger, i dette eksempel vises der en fejl hvis man ikke kan finde nogle materialer fra databasen.

## Calculator:

```
private void calcStrapsPort(Order order) throws DatabaseException {
    List<Material> materials = MaterialMapper.getMaterialsByMinLengthAndGroupId(carportlength, STRAPS, connectionPool);

    // If no materials of exact length are found, get the closest material with length greater than or equal to half of :
    if (materials.isEmpty()) {
        // Calculate half of the carport length
        int halfCarportLength = carportlength / 2;

        // Get materials with length greater than or equal to half of the carport length
        materials = MaterialMapper.getMaterialsByMinLengthAndGroupId(halfCarportLength, STRAPS, connectionPool);
    }

    // If materials are still not found, handle the case
    if (materials.isEmpty()) {
        // Handle case when no suitable materials are found
        System.out.println("No suitable materials found for straps.");
        return;
    }
}
```

Metoden begynder med at hente materialer fra databasen enten med samme længde som carportens længde fra formparameteren eller en længde, der er 1 højere, hvis den første ikke findes i databasen. Hvis der ikke findes nogen materialer med den ønskede længde eller en længde, der er 1 større, vil metoden fortsætte til næste linje.

Her opdateres længden, der søges efter, hvis den første søgning er mislykket. Længden for carporten findes ved at dividere dens længde med 2. For eksempel, hvis den maksimale længde i databasen er 600 og carportens længde er 780, vil den opdaterede længde være 390. Hvis materialet med denne længde ikke findes i databasen, vil den nærmeste større længde, f.eks. 420 i vores database, blive valgt. Hvis listen stadig er tom, fortsætter metoden til næste linje.

Her gives der en besked til brugeren, hvis der ikke findes noget materiale, så man kan se, at der kommer en fejl nede i konsollen, da der ikke er nogle materialer som bliver hentet. Dog hvis der bliver hentet et materiale fra databasen, bliver det brugt til at bestemme mængden i

**ordrelinjen ved brug af quantity metoden som er vist her:**

```
public int calcStrapsPortQuantity(int strapLength) {
    // Ensure there are at least 2 straps (one for each side)
    return (int) Math.ceil(2.0 * carportlength / strapLength);
}
```

**og det er så denne metode som bliver brugt til at beregne quantity som bliver indsat ind**

i ordrelinjen som vist her:

```
int quantity = calcStrapsPortQuantity(materials.get(0).getLength());
OrderLine orderLine = new OrderLine( orderLineId: 0, order.getOrderid(), materials.get(0).getMaterialId(), description: "Remme i sider, sadles ned i stolper", quantity);
orderLines.add(orderLine);
```

## ShowOrderLinesCustomer:

Koden tjekker om kunden har betalt den specifikke ordre før de kan se styklister for ordren.

```
public static void showOrderLinesCustomer(Context ctx, ConnectionPool connectionPool) {

    int orderId = Integer.parseInt(ctx.formParam( key: "orderId"));

    try {
        boolean payment = OrderMapper.checkPayment(orderId, connectionPool);

        if (payment) {

            try {

                List<PartList> partList = OrderLineMapper.PartList(connectionPool, orderId);

                List<String> orderStatusList = OrderStatusMapper.getOrderStatus(connectionPool, orderId);

                ctx.attribute("partList", partList);
                ctx.attribute("orderStatusList", orderStatusList);
                ctx.render( filePath: "orderStatus.html");
            } catch (Exception e) {
                ctx.status(500).result( resultString: "Fejl ved hentning og visning af ordrelinjer: " + e.getMessage());
            }
        } else {
            ctx.status(401).result( resultString: "Uautoriseret: Ingen betaling fundet for ordre ID " + orderId);
        }
    } catch (Exception e) {
        ctx.status(500).result( resultString: "Fejl ved behandling af anmodning: " + e.getMessage());
    }
}
```

## Status på implementering

Vi har hovedsageligt nået alle de websider vi havde planlagt i navigations diagrammet. Den eneste, vi ikke har nået, er siden til standard carporte. I stedet bliver man navigeret direkte til Fogs side for standard carporte, vi har heller ikke nået at lave en funktion for at filtrere og sortere standard carporte, da vi ikke fik lavet noget om standard carporte.

Vi har ikke nået at lave alle CRUD metoder til alle tabellerne. Vi har ikke fået lavet nogle delete-knapper, men vi har bevidst valgt ikke at gøre det.

Vi har ikke fået nået at lave SVG tegninger, og har heller ikke nået at give administratoren en mulighed for at rette og oprette nye varer i systemet.

## Kvalitetssikring (test)

### Automatiserede Tests

Unit- og Integrationstests:

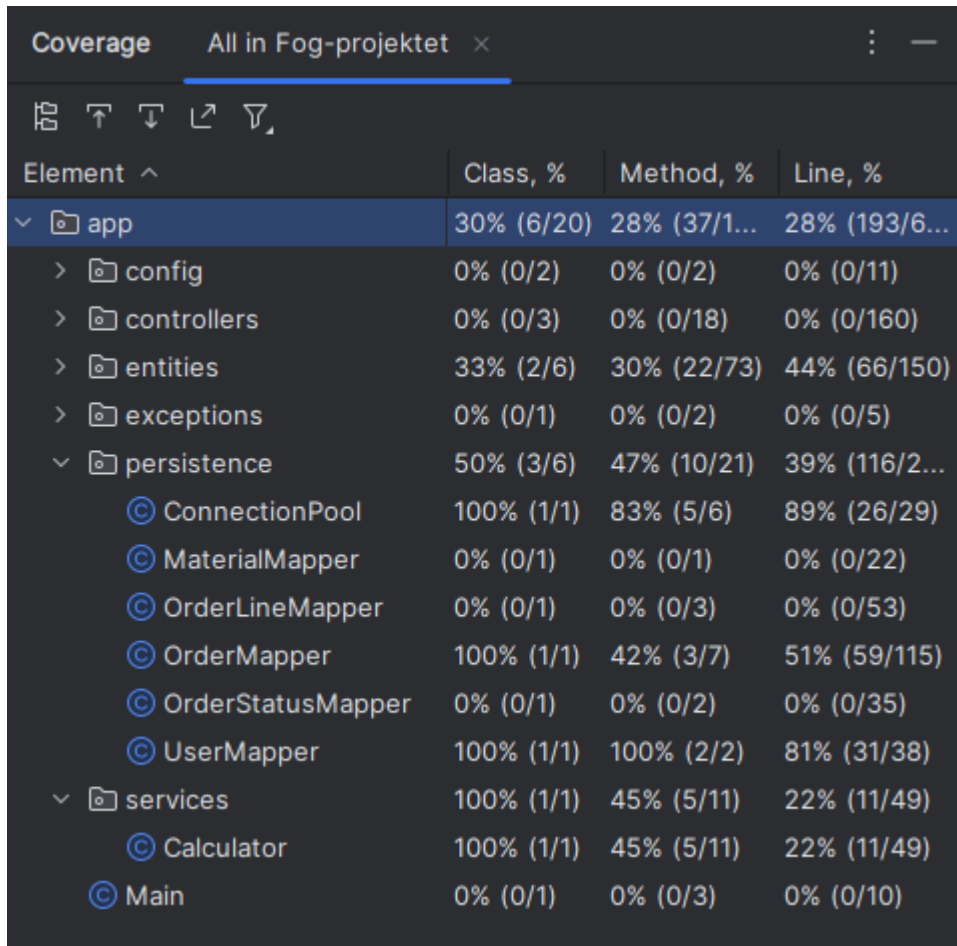
Klasse	Metode	Dækningsgrad
OrderMapperTest	getAllOrders	14%
OrderMapperTest	getAllOrdersUser	14%
OrderMapperTest	insertOrder	14%
UserMapperTest	login	50%
UserMapperTest	createUser	50%
CalculatorPost	calcPostQuantity	9%
CalculatorRafter	calcRafterQuantity	9%
CalculatorStrapPort	calcStrapsPortQuantity	9%
CalculatorStrapShed	calcStrapsShedQuantity	9%

Klasse: Hvilke klasser som bliver testet.

Metode: Hvilke metoder ud fra klasserne som bliver testet.

Dækningsgrad: Viser hvor meget af koden som er testet, så hvor godt koden er blevet gennemgået.

Coverage test:



The screenshot shows a coverage report for a project named 'Fog-projektet'. The report is organized into a tree view with columns for 'Element', 'Class, %', 'Method, %', and 'Line, %'. The 'app' directory is expanded, showing subdirectories like 'config', 'controllers', 'entities', 'exceptions', 'persistence', and 'services'. The 'persistence' directory is further expanded, showing classes like 'ConnectionPool', 'MaterialMapper', 'OrderLineMapper', 'OrderMapper', 'OrderStatusMapper', and 'UserMapper'. The 'services' directory is also expanded, showing 'Calculator' and 'Main'. The coverage percentages are shown for each element, with some elements having a percentage and a count in parentheses.

Element ^	Class, %	Method, %	Line, %
✓ app	30% (6/20)	28% (37/132)	28% (193/688)
> config	0% (0/2)	0% (0/2)	0% (0/11)
> controllers	0% (0/3)	0% (0/18)	0% (0/160)
> entities	33% (2/6)	30% (22/73)	44% (66/150)
> exceptions	0% (0/1)	0% (0/2)	0% (0/5)
✓ persistence	50% (3/6)	47% (10/21)	39% (116/295)
Ⓢ ConnectionPool	100% (1/1)	83% (5/6)	89% (26/29)
Ⓢ MaterialMapper	0% (0/1)	0% (0/1)	0% (0/22)
Ⓢ OrderLineMapper	0% (0/1)	0% (0/3)	0% (0/53)
Ⓢ OrderMapper	100% (1/1)	42% (3/7)	51% (59/115)
Ⓢ OrderStatusMapper	0% (0/1)	0% (0/2)	0% (0/35)
Ⓢ UserMapper	100% (1/1)	100% (2/2)	81% (31/38)
✓ services	100% (1/1)	45% (5/11)	22% (11/49)
Ⓢ Calculator	100% (1/1)	45% (5/11)	22% (11/49)
Ⓢ Main	0% (0/1)	0% (0/3)	0% (0/10)

Vi arbejdede med unit testing før, at vi fik den ind i main branch. Det er smart at teste, om man fik den rette quantity og om metoden virkede, før den blev merged ind i main branch. Det samme gjorde vi for integrationstestene; vi tjekkede på dataen og så, om det blev oprettet, som det skulle, før vi satte det ind på main branchen.

## User Acceptance test

User test over et familiemedlem, som ikke kender til programmering:

User Story ID	User Story Beskrivelse	Acceptkriterier	UAT (User Acceptance Test)	(Godkendt/Ikke Godkendt)	Kommentarer
US-1	Som kunde ønsker jeg at oprette mig som bruger	Givet at jeg indtaster personlige oplysninger og klikker "opret", så opretter jeg en konto og kan logge ind.	Forsøg at oprette en konto med gyldige oplysninger og log ind efterfølgende.	Godkendt	Fordi det er let overskueligt at gøre.
US-2	Som kunde ønsker jeg at kunne logge ind	Givet at jeg indtaster personlige oplysninger og klikker "login", så bliver jeg logget ind og kan se status på mine bestillinger og bestille en ny carport.	Forsøg at logge ind med gyldige oplysninger og se status på bestillinger samt bestille en ny carport.	Godkendt	Det er nemt at bestille.
US-3	Som kunde ønsker jeg at bestille en valgfri carport	Givet at jeg indtaster ønskede mål og udfylder en formular, når jeg klikker "Design nu" og "Send forespørgelse", så indsendes min formular.	Forsøg at udfylde en formular med ønskede mål og sende den.	Godkendt	Det er nemt at finde formularerne og sende dem.
US-4	Som kunde ønsker jeg at kunne se status på mine ordrer	Givet at jeg klikker på "status", så kan jeg se status, stykliste og tegninger af mine ordrer.	Forsøg at se status, og stykliste af en bestilt carport.	Godkendt	Man kan sagtens se bestillingerne for diverse carporte.



US-5	Som kunde ønsker jeg at kunne se totalpris for styklisten	Givet at jeg klikker på "status", så kan jeg se totalprisen for min ordre.	Forsøg at se totalprisen for en bestilt carport.	Godkendt	Kan sagtens se ud fra dens rubrik.
US-6	Som kunde ønsker jeg at kunne bekræfte min ordre	Givet at jeg klikker på "Bekræft", så kan jeg se min stykliste for ordren.	Forsøg at bekræfte en ordre og se styklisten.	Godkendt	Kan sagtens se stykliste og bekræfte ordre.
US-7	Som kunde ønsker jeg at se en stykliste efter betaling	Givet at jeg klikker på "Bekræft", så får jeg vist en stykliste af min udvalgte carport.	Forsøg at se en stykliste efter at have bekræftet en ordre.	Godkendt	Kan sagtens se styklisten og den er nemt at overskue.
US-8	Som administrator ønsker jeg at se alle ordrer i systemet	Givet at jeg klikker på "Order", så får jeg vist ordrelisten.	Forsøg at se alle ordrer i systemet.	Godkendt	Det er nemt at gøre og oversigten er nemt at overskue.
US-9	Som administrator ønsker jeg at se alle ordrelinjer i systemet	Givet at jeg klikker på "Order", så får jeg vist ordrelinjelisten.	Forsøg at se alle ordrelinjer i systemet.	Godkendt	Det er nemt at gøre og ordrelinjerne er nemt at overskue.
US-10	Som administrator ønsker jeg at kunne beregne totalprisen for alle materialer	Givet at jeg klikker på "Calculate", så bliver totalprisen vist for kunden.	Forsøg at beregne totalprisen for en ordre.	Godkendt	Knappen er nemt at finde
US-11	Som administrator ønsker jeg at kunne generere en stykliste	Givet at jeg klikker "Se styklisten", så genererer jeg styklisten.	Forsøg at generere en stykliste for en ordre.	Godkendt	Den er nem at finde og gøre.

# Proces

## Arbejdsprocessen faktuel

- **Faser og user stories**

### **Fase 1: Planlægning, forretningsforståelse og projekt setup:**

I den indledende fase af vores projekt arbejdede vi med forretningsforståelse og gik i gang med at lave interessentanalyse, risikovurdering, forberedte vores user stories og udarbejdede også aktivitetsdiagrammerne AS-IS og TO-BE.

Efter dette oprettede vi et projekt repository på Github og oprettede et kanban board på baggrund af vores user stories, som vi delte ned i mindre opgaver. Vi oprettede også et navigationsdiagram for at få overblik over navigationen på vores webside, og et klassediagram for at få overblik over de klasser som skulle oprettes i vores projekt.

Vi fik udarbejdet Figma over vores tænkte design, og derefter aftalte vi i gruppen at uddele de html-sider som vi skulle bruge.

Vores ERD diagram blev udarbejdet, og vores database blev også oprettet, og deployet til en remote server.

### **Fase 2: Grundlæggende funktionalitet:**

I anden fase færdiggjorde vi diagrammerne, og udarbejdede de resterende html-sider. Vi gik i gang med at få den grundlæggende funktionalitet på plads, såsom login og oprettelse af brugere. Derefter gik vi også i gang med funktionalitet for administratorens ordreoversigt.

### **Fase 3: Udvidet funktionalitet:**

I denne fase begyndte vi på de resterende user stories, som vi delte ud i gruppen. Vi startede fra de første user stories, og efterhånden som de blev lavet, så gik vi i gang med de næste. Vi arbejdede ud fra kanban board og udvalgte opgaverne derfra. Vi samlede løbende op på projektet ved at committe vores kode til github.

## **Fase 4: Afslutning af projekt og rapportskrivning**

I sidste fase af projektet gik vi i gang med de sidste user stories, og efterfølgende samlede vi op på resten af koden til projektet, og fik det hele samlet i vores main branch. Vi gik i gang med at oprette unit test klasser, for at sikre at vores kode fungerer som forventet. Til sidst i projektfasen gik vi i gang med at skrive på rapporten i et fælles dokument, og vi aftalte i gruppen hvad vi hver især skulle skrive om. Efterfølgende læste vi rapporten igennem i gruppen, og derpå afleverede vi det samlede projekt.

- **Kanban**

I begyndelsen af projektet gennemgik vi i fælles vores user stories, og delte dem ned i mindre opgaver, som vi satte ind i kanban boarded. Efterhånden som vi blev færdige med opgaver hver især, ville vi gå ind og opdateret kanban boarded.

- **Vejledningsmøde**

Vi brugte vejledningsmøderne til at få hjælp og få afklaret de spørgsmål som vi havde, derfor før vi mødte op til vejledningen, så havde vi aftalt hvad vi skulle bruge de 30 minutter som vi havde på vejledningen, nogle gange var det ting som vi var i tvivl om og andre gange var det programmerings spørgsmål.

- **Kommunikation i teamet**

I teamet har vi løbende kommunikeret ved at holde statusmøder et par gange om ugen over zoom. Der har vi snakket om hvor langt vi er nået med vores opgaver, og om hvordan det går med udførelsen af opgaverne. Vi har også snakket om eventuelle problemer der er opstået, eller hvis der var ting vi har været i tvivl om som vi gerne vil vende med hinanden i gruppen. Derudover har vi skrevet sammen på discord dagligt. Desuden har vi også mødtes personligt på skolen et par gange i løbet af projektet, især hvis der har været noget der har krævet lidt mere samarbejde.

## Arbejdsprocessen reflekteret

- Vi havde ikke en KanBan mester rolle, men gennemgik det sammen i gruppen.  
Generelt til vores fælles møder gennemgik vi også kanban boarded, for at få overblik over de opgaver der ikke var begyndt på, opgaverne der var i gang og færdige opgaver. Men da vi ikke har meget erfaring med brug af KanBan, så kunne der godt opstå manglende opdatering en gang imellem.
- De væsentligste emner på vores evalueringsmøder var, hvor langt vi var i processen med løsningen af vores opgaver, og hvis opgaven var færdiggjort, hvilken opgave vi ville gå i gang med. Hvis der opstod problemer eller hvis der var noget vi var i tvivl om, så snakket vi også sammen omkring det i gruppen for at finde en fælles løsning.
- I begyndelsen var vi lidt i tvivl om hvordan vi skulle nedbryde user stories ned i mindre tasks. Men vi talte om det sammen i gruppen og tænkte på hvilken funktionalitet hver task skulle bidrage med.
- Hovedsageligt var vi spot-on med vores estimeringer af opgaverne. Der var måske nogle opgaver, der tog lidt længere tid end forventet, da det var mere udfordrende end regnet med eller pga. af problemer der opstod undervejs.
- Vi fandt produktiv rytme allerede i starten af projektet. Vi har i gruppen et godt samarbejde hvor vi hjælper og støtter hinanden. Vi har god kommunikation i gruppen. Vi har haft fokus på klar og åben kommunikation, og sørget for at være i kontakt med hinanden ofte. Vi har også haft klar opgavefordeling i teamet, og hvert enkelt gruppemedlem har været klar over deres ansvar.
- Generelt gik det godt med at dele kode via Github. En gang imellem stødte vi på merge-konflikter. I samarbejde kiggede vi på pull requests og koden, og diskuterede ændringer før vi skubbet det ind i vores main branch.