FOG CARPORTE

Gruppe: inSession

Jens Gelbek
E-mail: jensgelbek@gmail.com
Github: jensgelbek

Gustav Bidstrup
E-mail: guf@ejbybidstrup.dk
Github: GustavBidstrup

Peter Rambeck
E-mail: cph-pa127@cphbusiness.dk
Github: Peter-Rambeck

Droplet: http://134.209.229.51:8080/fog
Github repository: https://github.com/jensgelbek/Fog
Taiga: https://tree.taiga.io/project/gustavbidstrup-fogproject/backlog
Video:

Afleveringsdato: 06-06-2021

INDHOLDSFORTEGNELSE

INDLEDNING

Indledning 3

BAGGRUND

1. Virksomhed 4
2. Glossary 5
3. Problem
   - 3.1 Problem aktører 7
   - 3.2 SWOT 8
   - 3.3 Stakeholder 9
4. Systemet, og merværdi mod nuværende system. 11
   - 4.1 Workflow, activity diagrams. 13
5. Krav, User Stories. 16
6. Aktører og Use Case Diagram. 18
7. Use Cases 19

SYSTEM DESIGN

1. Overvejelser. 20
2. Design, arkitektur og valg. 21
3. Sikkerhed. 23
4. Domænemodel, og Domæne Diagram. 25

IMPLEMENTATION

1. Database, og ER Diagram 27
2. Tabel relationer 28
3. Sekvensdiagram 29
4. Navigationsdiagram. 31
5. Teknologier, Droplet, Tomcat 32
6. Status 34

PROCESS

1. Test. Unit & integration test 35
2. Evaluering. 37
3. Gruppe, og gruppearbejde. 38

AFSLUTNING

1. Afrunding
2. Referencer.

INDLEDNING

Indledning

Følgende rapport er en system-teknisk rapport af 'FOG Carporte opgaven' som beskriver og illustrerer gennem diagrammer, systemets:

Forretning, -design valg, -opbygning, -funktionalitet, -database og generelle overvejelser for valg truffet i systemets udviklingsfase.

Det er til hensigt at rapporten skal give et fuldt og dybdegående overblik, og sammen med de tekniske detaljer, give en udvikler med god forståelse en mulighed for, forholdsvis hurtigt at sætte sig ind i projektet og kunne arbejde videre på det.

BAGGRUND

1. Virksomhed

Johannes Fog er et byggemarked og trælast som tilbyder alt fra værktøj, til enkeltdele, til delløsninger, til komplette byggeløsninger.

Fog handler både med professionelle håndværkere som private kunder, hvilket gør deres sortiment bredt.

BAGGRUND

2. Glossary

Term	Forklaring
Bruger	End-user, Fog's kunde.
Sælger	Sælger, ansat hos Fog. Ansvarlig for salg og kontakt af bruger.
System	Software program som tilgås via browser
Tilbud	Uforpligtende indtil kunde og sælger accepterer
Ordre	Accepteret tilbud
Dynamik	Automatisk
DIY	Gør-det-selv. Ikke standardiseret.

BAGGRUND

3. Problem

Manglende dynamik.

Fog's trælast og byggemarked tilbyder gør-det-selv Carport pakkeløsninger, med og uden skur, efter egne mål og design.

Et forretningsområde som online kan forbedres med lidt ekstra funktionalitet samt dynamisk front-og bag-end.
Med nuværende løsning foregår tilbudsgivning i 3 trin.

1. Trin
   På Fog's webpage kan en bruger vælge mål på hhv. længde og bredde af Carporten ud fra en drop Down menu, hvorefter bruger da kan sende de valgte mål til Fog.

2. Trin
   En sælger vil da modtage en mail med brugers valg.
   Sælger skal da på den baggrund manuelt sammensætte en stykliste af materialer og udregne et tilbud.

3. Trin
   Sælger vil da sende tilbud til bruger via mail og kontakte kunden telefonisk for at både vejlede og lukke salget.

BAGGRUND

3.1 Problem aktører

Dynamiske problemer bruger.

I den nuværende 3 trins proces opstår følgende dynamiske problemer for bruger,

1. Bruger kan ikke lave en bestilling uden sælger.
2. Tid. Bruger skal vente på tilbud/prisoverslag.
   - Bruger bliver utålmodig og kontakter konkurrenter,
3. Manglende visuel oversigt kan give kunden mål-usikkerhed
   - Eks. Hvor meget plads til skur, bil og andet giver mine valgte mål mig?
4. Pris.
   - Hvad koster en given løsning, hvor er jeg i forhold til budget? med skur? uden skur?
5. Bruger kan vælge forkerte mål, - Eks. Skur er større end Carport.

Dynamiske problemer sælger.

I den nuværende 3 trins proces opstår følgende dynamiske problemer for sælger,

1. Sælger kan ikke få en ordre uden kontakt til bruger
2. Sælger skal manuelt rette fejl fra bruger.
3. Sælger skal manuelt udregne materialer til stykliste.
4. Sælger skal manuelt korrigere for 'gamle og ukorrekte' materiale mål.
5. Sælger skal sende tilbud via Email til bruger.

BAGGRUND

3.2 SWOT-analyse på teamet

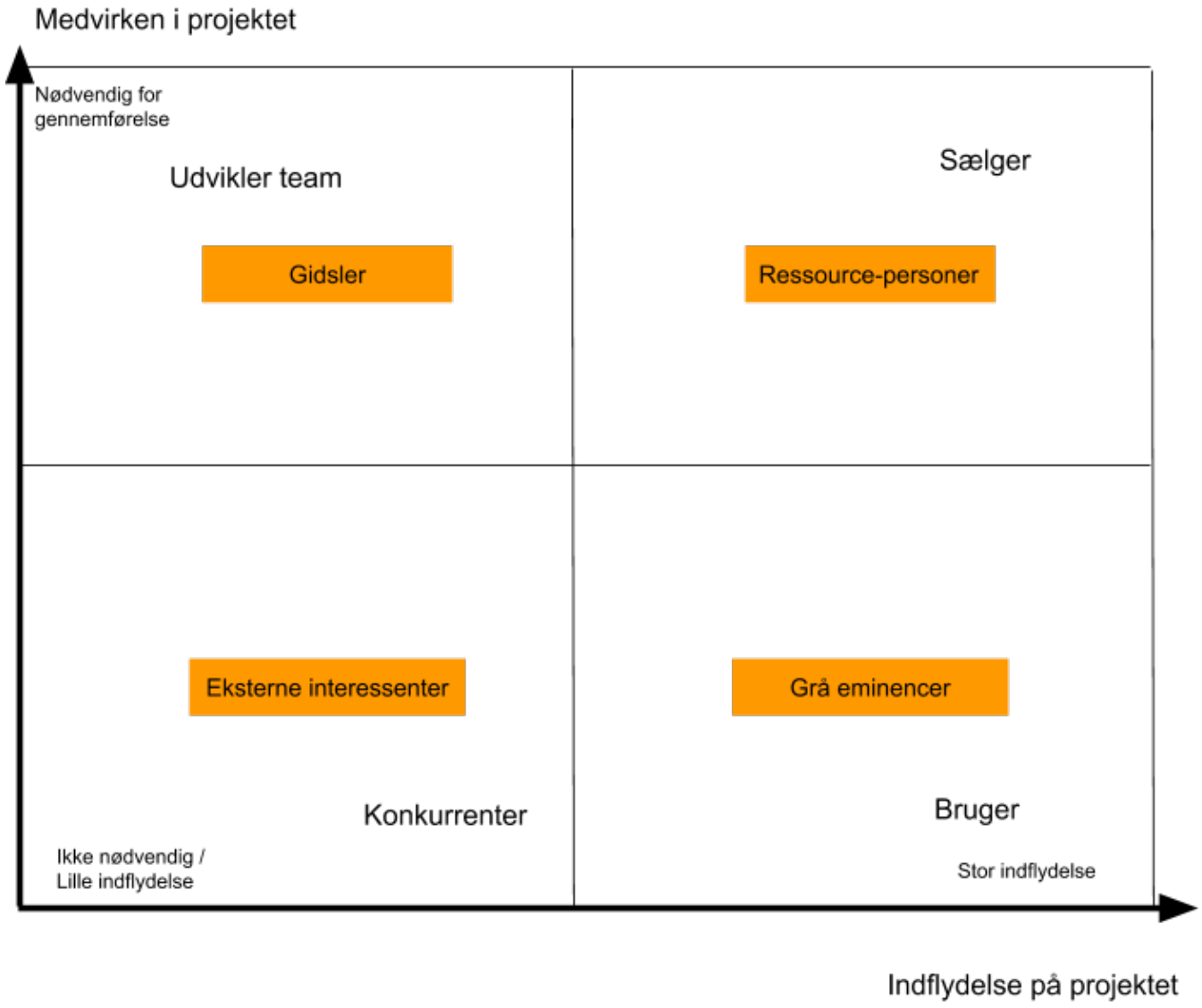
SWOT	Hjælp *…til at komme i mål*	Problemer *…for at komme i mål*
Internt *(organisation)*	***STRENGTHS***	***WEAKNESSES***
	Godt gruppesamarbejde. God gruppe og arbejdsdisciplin. Vi laver ting der virker.	Vi starter nogen gange på at kode for tidligt Vi laver ikke test nok.
Ekstern *(miljø)*	***OPPORTUNITIES***	***THREATS***
	Har altid mulighed for hjælp, fra kompetent lærer.	Tid.

BAGGRUND

3.2 Stakeholder

Fog's carport systems interessenter:
Bruger, Her defineret som End-user/kunde af Fog Trælast og Byggecenter.
Sælger(ere), De personer/ansatte i sælger position hos Fog som i daglig drift bruger og håndterer systemet.

Interessent	Fordele	Ulemper	Vurdering af interessentens bidrag/position	Håndtering af interessenten
Bruger	Mulighed for eget online design, med dynamisk pristilbud. Kan ikke fejlberegne skur størrelse. Har visuel oversigt af valgte mål med tilhørende 'indre mål' = faktisk plads. Kan lave en online bestilling på baggrund af tegning og pris. Egen konto: 1 Mulighed for flere tilbud som kan sammenlignes. 2. Efter betaling, adgang til online Stykliste.	Mindre menneske-kontakt, som giver bruger mulighed for nemt at sammenligne tilbud med Fog's konkurrenter.	Bruger er bruger af systemet, men er ikke direkte med i udviklingsprocessen Brugerens oplevelse af systemet med fordele, er vital for Fog.	Eventuelt inddrage bruger og lave tests med brugerinput, i udviklingsfasen
Sælger(ere)	Sælgere skal ikke manuelt overføre data mail til beregningsprogram	Automatiseret med mindre kundekontakt.	Sælger giver input til krav og funktionalitet, og dermed vigtig for udvikling af systemet.	Taler med sælger ved de ugentlige sprint Reviews.


Medvirken i projektet
Nødvendig for gennemførelse
Udvikler team
Gidsler
Sælger
Ressource-personer
Eksterne interessenter
Konkurrenter
Grå eminencer
Bruger
Ikke nødvendig / Lille indflydelse
Stor indflydelse
Indflydelse på projektet

BAGGRUND

4. Systemet, og merværdi mod nuværende system.

Carport:
Som en del af Fog's sortiment tilbyder de Carport løsninger i 2 varianter,

1. enten som en standardløsning hvor kunden vælger en bestemt pakke af standard mål, som da kommer leveret komplet med tilhørende delelementer,
2. eller som en DIY løsning hvor bruger selv vælger carportens og skurs mål

Afgrænsning:

Vores opgave og system er designet til at afhjælpe Fog's kunder og Fog til en bedre online løsning af #2. varianten:
DIY Carporte og skur.

System:

DIY carporte kan designes i tre variationer efter brugers valg:

1. Carport med fladt tag
    1. Tagmateriale findes kun i 'Plasttrapetsplader'
2. Med -eller uden skur.

Merværdi

Som beskrevet i 'BAGGRUND: 3.1 Problem'

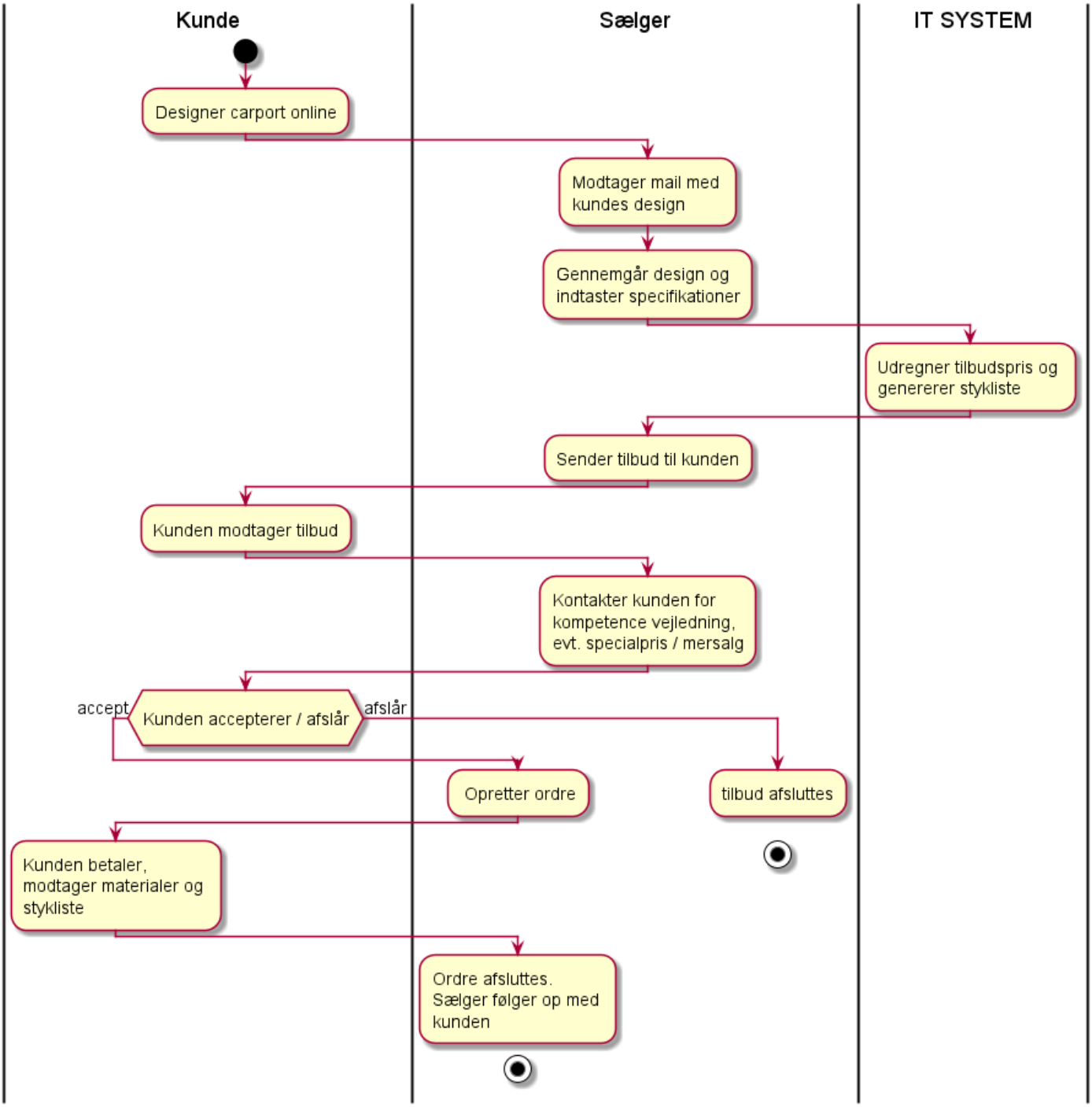
<u>Nuværende system online carport:</u>

Bruger kan individuelt vælge hhv bredde og længdemål samt tagmateriale på en ønsket Carport og skur størrelse ud fra et sæt af pre definerede mål størrelser.
Herefter at indtaste personlige oplysninger og sende en forespørgsel på tilbud.

Processen mellem Bruger, sælger og IT systemet er illustreret i 'Aktivitetsdiagram ASIS',

Processen er langsommelig og IT systemet har ingen dynamik til hverken kunde eller sælgers værdi.
De mange skridt og manuelle delprocesser øger sandsynligheden for fejl, misforståelser og utilfredshed hos begge aktører.

Aktivitetsdiagram: (ASIS - Nuværende)



BAGGRUND

4.1 Workflow, Activity diagrams

inSession forbedret system online carport:

Carport tegning:

Bruger kan individuelt vælge hhv bredde og længdemål på en ønsket Carport størrelse ud fra et sæt af pre definerede mål størrelser.
Carporten vil herefter blive dynamisk tegnet på siden efter valgte mål, sammen med de indre mål.

Bruger kan som tilvalg vælge skur efter egne mål, som ligeledes vil blive tegnet ind i Carporten og give et komplet visuelt design.
Som 'sikkerhed' er det ikke muligt at vælge et skur som er større end selve carporten.

Konto:

Bruger skal oprette en konto.
Ved at klikke på 'Få tilbud' knap bliver pristilbud på Carport og evt. skur efter tegning sendt til brugers konto.
Flere tilbud kan laves dynamisk, eks. med eller uden skur.

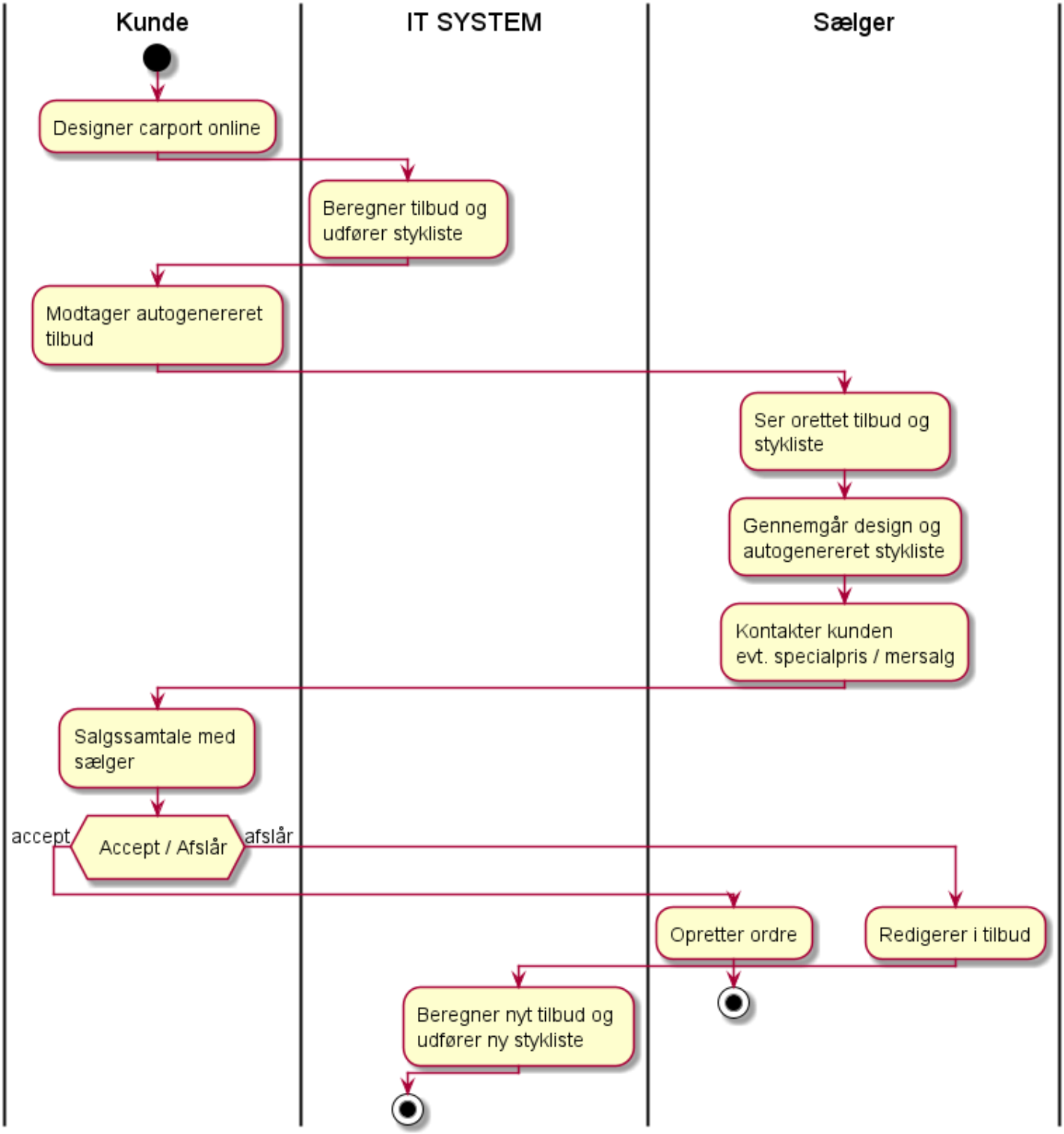
Bestilling/Kontakt:

Bruger kan direkte bestille, altså lave en ordre, eller sende en besked til sælger om at blive kontaktet.

Stykliste bliver dynamisk samlet under ordreId på en Admin konto.
Sælger kan derefter ændre i styklisten og/eller den samlede pris.
Når bruger færdiggør ordre ved betaling, vil styklisten blive tilgængelig på brugers konto.

Processen er illustreret i 'Aktivitetsdiagram TOBE', efterfølgende side.

Aktivitetsdiagram: (TOBE - Nyt system)



Tabelform oversigt
Merværdi og forbedrede kundeoplevelse, samt mere effektiv behandling af bruger:

Nuværende online kundeløsning:		
	Carport med fladt tag	Carport med rejsning
Med skur	Ja	Ja
Uden skur	Ja	Ja
Tegning	Nej	Nej
Pris	Nej	Nej
Egen konto med pris og Stykliste	Nej	Nej
Admin. Sælger opdatere i DB	Nej	Nej
Sælger dynamisk generer nyt tilbud, + opdaterer kundens konto	Nej	Nej

inSession online løsning med merværdi		
	Carport med fladt tag	Carport med rejsning
Med skur	Ja	Nej
Uden skur	Ja	Nej
Tegning	Ja	Nej
Online dynamisk pris	Ja	Nej
Egen konto med pris og Stykliste	Ja	Nej
Admin. Sælger opdatere i DB	Ja	Nej
Sælger dynamisk generer nyt tilbud, + opdaterer kundens konto	Ja	Nej

BAGGRUND

5. Krav, User Stories.

Kravspecifikation:

Systemet skal opfylde følgende krav:

Online Design af Fladt tag Carport og skur ud fra et sæt pre-definerede størrelser.
Dynamisk og online visuel redigering af eget design.,
Online bestilling af samme Carport med eller uden skur.
Opret brugerkonto.
Online tilbud på samme, evt. flere design.
Online afslå tilbud / bestil sælger kontakt.
Online stykliste efter eget design,
Admin konto
Admin konto - vis alle ordrer med status
Admin, rediger i ordre.
Admin, rediger i tilbudspris
Admin, rediger i materialepris

Sprint & user stories:

Sprint 1 gik med at få sat projektet op, og der er derfor ikke nogle userstories til Sprint 1 i tabellen.

Sprint	User Stories
2	Som kunde vil jeg gerne kunne oprette en carport med fladt tag
2	Som kunde vil jeg gerne kunne se den carport jeg har bestilt
2	Som kunde vil jeg gerne kunne bestille en carport med tagrejsning
2	Som bruger vil jeg gerne kunne oprette mig i systemet som kunde
2	Som kunde vil jeg gerne kunne logge ind på hjemmesiden
2	Som sælger vil jeg gerne kunne se alle ordrer
2	Som sælger vil jeg gerne kunne se individuelle ordrer
2	Som kunde vil jeg gerne kunne se mine ordrer
3	Som sælger vil jeg gerne kunne se og redigere ordrer
3	Som sælger vil jeg gerne kunne se og redigere styklister
3	Som kunde vil jeg gerne kunne se stykliste efter betaling
3	Som sælger vil jeg gerne kunne logge ind med sælgers rettigheder
3	Som sælger vil jeg gerne kunne ændre prisen på materialer

BAGGRUND

6. Aktører og Use Case diagram

Systemet har 2 aktører:

1. Bruger
2. Sælger(re)

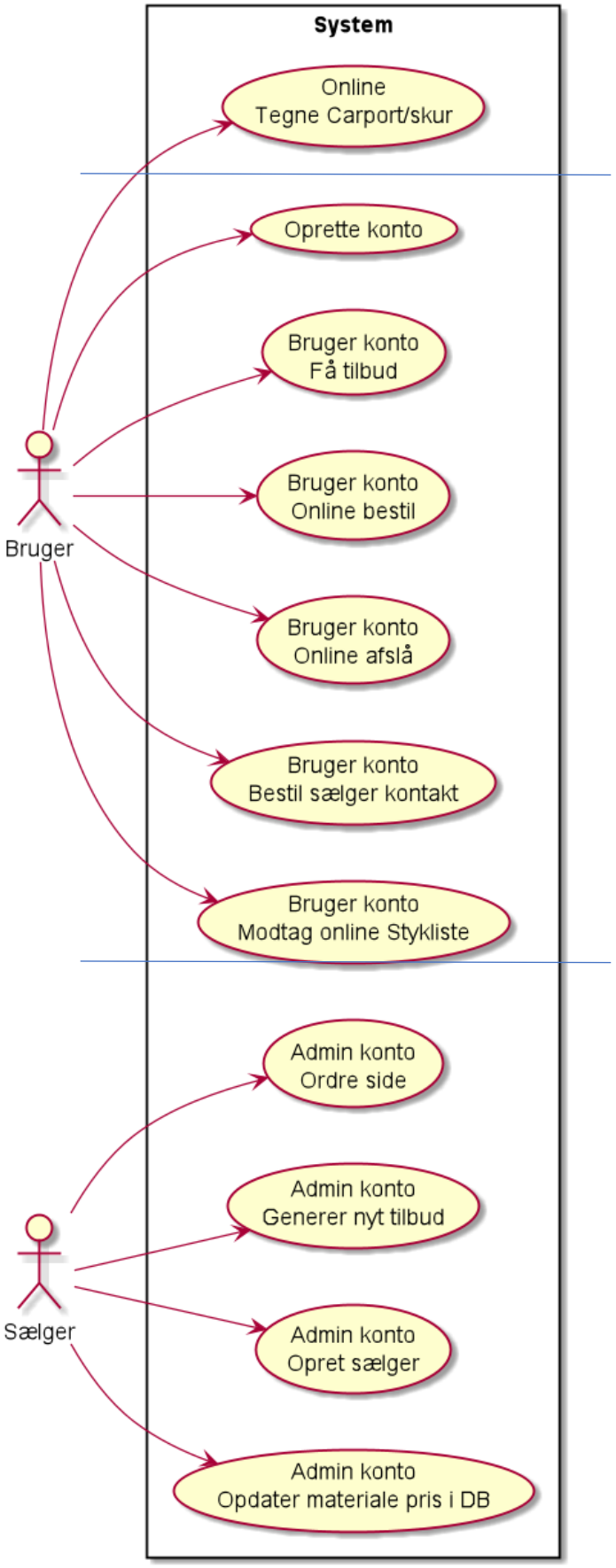
Systemet er delt op i 3 funktionelle dele som har:

1.
Et brugerinterface som både tegner Carport og skur samt beregner en dynamisk pris på baggrund af brugerinput.

2.
En personlig brugerkonto med individuelle tilbud, ordre og tilhørende stykliste.

3.
En administrationsdel hvor sælger kan administrere alle ordrer, udarbejde nye tilbud og opdatere nye priser direkte i databasen.

Pris opdateringer vil dynamisk blive brugt til nye beregninger af dynamiske tilbud på Del. 1.



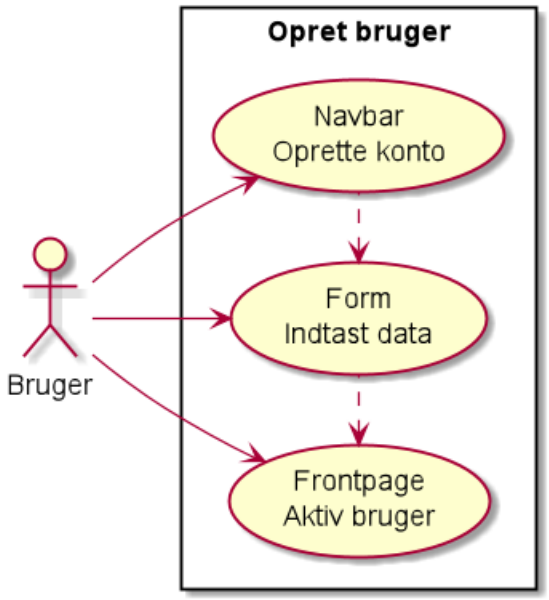
BAGGRUND

7. Use Cases

Vi har valgt her at beskrive en Use Case.
Samme som bruger i Sekvensdiagrammet. (ref. IMPLEMENTATION 3.Sekvensdiagram)

- Opret dig som bruger i systemet

User stories kan brydes ned i mange små Use Cases som hjælper udvikleren til mere direkte og specifikt at afgrænse en problemstilling.



Use Case 1:

Opret dig som bruger i systemet

Primary actor:

Bruger

Forudsætter:

Adgang til internet, på telefon eller computer.

Basic flow of events:

Kunden trykker på knappen 'Opret bruger' i navigationsbaren, hvorefter man bliver directed til en form-side for at taste input data.
Her skal indtastes navn, adresse, e-mail og password.
Bruger får *ikke* tildelt et id, men identificeret med e-mail.
At bruger er logget ind og aktiv fremgår ved at email ses i nav-baren.

SYSTEM DESIGN

1. Overvejelser

Agile:
I valg af systemdesign var hovedvægten på, så vidt som muligt, at adskille de forskellige dele, og på den måde også gøre dem så uafhængige af hinanden som muligt.
Således at kan man udvikle på en separat del og holde systemstrukturen uden komplikationer.

I store træk består det samlede system af 3 dele, -web delen, -database delen, og -java delen som binder web og database sammen.
Ved det er også være muligt at f eks flytte databasen, eller skifte databasen ud ved at skrive til filer uden for mange indvirkninger på den fundamentale struktur.

Java-delen kan så igen deles op i fire sub-dele: en kerne, en del med business-logik, en del der håndterer websiderne og en del der tager sig af databasen.

SYSTEM DESIGN

2. Design, arkitektur og valg

Vi har baseret vores projekt på en Onion arkitektur.
En arkitektur som bygger på flere data lag der holdes adskilt ved at lagene kun kommunikerer indad i strukturen.

2.1 Onion arkitektur

Onion arkitektur er objekt klasser som systematiskt samles lag-på-lag (Onion), efter et princip af samhørighed. Strukturen arbejder indad, forstået sådan at de indre lag ikke kan kalde de ydre lag 'ringe'.

Kald går således indad i modellen, lag-for-lag.
Man beskytter således lagene/strukturen mod ændringer eller modelleringer.

Adgang til systemet skal kun ske via det 'yderste lag' som implementerer og kontroller systemets infrastruktur samt Web.

I Kernen af modellen har vi domæne klasserne og Interfaces (repositories) til de metoder som gemmer og henter disse klasser "eksternt".

Interfaces implementeres af database repositories i 'Infrastructure' laget.

I Infrastructure kan vi nemt skifte vores datamedie ud til en anden type DB, filer eller noget helt tredje ved at bruge en anden form for implementering af interfaces, uden at det ændrer noget for resten af projektet.

Vi har valgt at database Repositories instantieres med den valgte database.
Dette udnytter vi i tests, da vi her bare instantierer med en test database og dermed ikke ændrer på den rigtige database.

2.2 Servlets

Vores webdel er baseret på servlets.
Alle servlets extender Baseservlet, som indeholder en render-funktion der renderer den ønskede jsp side.
Baseservletten har en statisk instans af api, instantieret med vores DB.
Inden jsp sider vises tjekkes de på en liste over hvilke sider alle må se holdt op med hvem som er logget-ind:
En bruger som er logget-ind er lagt i en attribut i sessionen.

Servlets har metoderne doGet og doPost, hvor doGet bruges når brugerens browser beder om at se en side. doPost bruges når brugerens browser, poster ting den har fået som input fra inputfelter, dropdown menuer osv.

2.3 Jsp sider

Alle vores jsp sider er lavet ved at Baseservletten, indsætter den relevante jsp side ind i en base.jsp side.
Base.jsp indeholder header, footer og includes af de relevante ting som bootstrap,css osv.

Servletten overfører data til jsp siden ved at stemple dem ind i HttpServletRequest som atributte.,
Eneste gange det ikke bliver gjort på denne måde er når man logger ind, så bliver brugernavn og om man er ansat/Admin gemt i HttpSession samt når man er ved at bestille en carport bliver målene gemt samme sted.

Disse to undtagelser skyldes at vi har bedømt at det giver god mening at have tilgang til disse på alle sider.

Headeren i base.jsp, indeholder en navigatons bar fra Bootstrap, som sikrer en nem hurtig måde til at navigere websiden.

De menupunkter der bliver vist i navbaren, afhænger af hvem som er logget ind eller om nogen er logget ind.
Dette gør at ingen får tilbudt at tilgå sider de ikke må/får lov til at se.

SYSTEM DESIGN

3. Sikkerhed

Bruger og Sælger har differentieret adgang til sider og da funktionalitet.
Det handles af 'BaseServlet klassen'.

Menupunkter,
nav-bar, if sætninger

Bruger adgang: af en methode 'api' der kalder *api = createApplication* som samtidig indeholder en liste af alle de sider der indgår.

Sælgers rettigheder: af en metode 'render' som identificerer sælger ved matche navn i Database. Og hvis objektet findes giver adgang til al 'content' i base.jsp.
Altså fuld rettigheder.

Det betyder at man som kunde ikke kan tilgå sælgerens sider.
Hvis ikke, -error handles '401'

Skur design

For at undgå uhensigtsmæssige designs af carport med skur, har vi indført betingelser for valg af skur mål afhængigt af valgte carport mål.

Dette handles fra 'Bestilling klassen' ved at carport's mål skal vælges fra drop-down menu, SVG tegnes på siden og gemmes i en Httpsession.
Drop-down menu for skur's længde og bredde valgmuligheder bliver reduceret med 'Carport - 600mm' i 'bestilling.jsp'

Yderligere, intuitivt er Skur dropdown ikke visuel før carport mål er stemplet i Session.

Altså, et skur kan ikke ved fejl hverken tegnes eller bestilles efter forkerte mål.

Få tilbud, og bestilling

For at foretage en bestilling efter eget design og senere en stykliste, skal en bruger oprette en konto først med tilhørende personlige oplysninger.
Dette handles fra 'Bestilling klassen' med *doPost Httpsession* som betinger Bruger er oprettet/logget ind.

Password til brugerkonto:

Hashing og salts:
Når bruger opretter en konto adgang, bliver han/hendes password hashet.
Dette er en form for énvejs-kryptering, som sikrer kundernes password ikke nemt kan efterlignes. Hvis password f.eks. er 'hej123', er det ikke 'hej123' der bliver gemt, men en hash-krypteret version af dette.
Dette vil typisk være en mange-cifret blanding af tal og bogstaver.

Får eventuelle hackere adgang til databasen med de hash-krypterede passwords, kan de ikke rigtig bruge det til noget, da der ikke er nogen mulighed for at dekryptere til den originale password-string, som er nødvendig for at logge på.
Der findes dog databaser med kendte password hashes, så vær kreativ med dit password!

For yderligere at forhøje sikkerheden for kunderne, er der også tilføjet *salt.*
Salt er nogle ekstra tegn, der tilføjes til ens password. Hvis det genererede salt f.eks. er 'ef53l2', og password er 'hej123', vil det saltede password blive 'hej123ef53l2'.
Det bliver derefter hashet, og gemt i databasen.
Dette er en god sikkerhed for folk, som har tendens til at lave nemme passwords. Passwordet 'hej123' ligger muligvis allerede i en hash decryption database et sted på nettet, men det gør 'hej123ef53l2' højst sandsynligt ikke.

Password fieldet er implementeret med en required attribut, der gør det umuligt at lave et tomt password. Krav til passwordet kan ændres efter behov. På nuværende tidspunkt er det eneste krav, at passwordet ikke må være null.

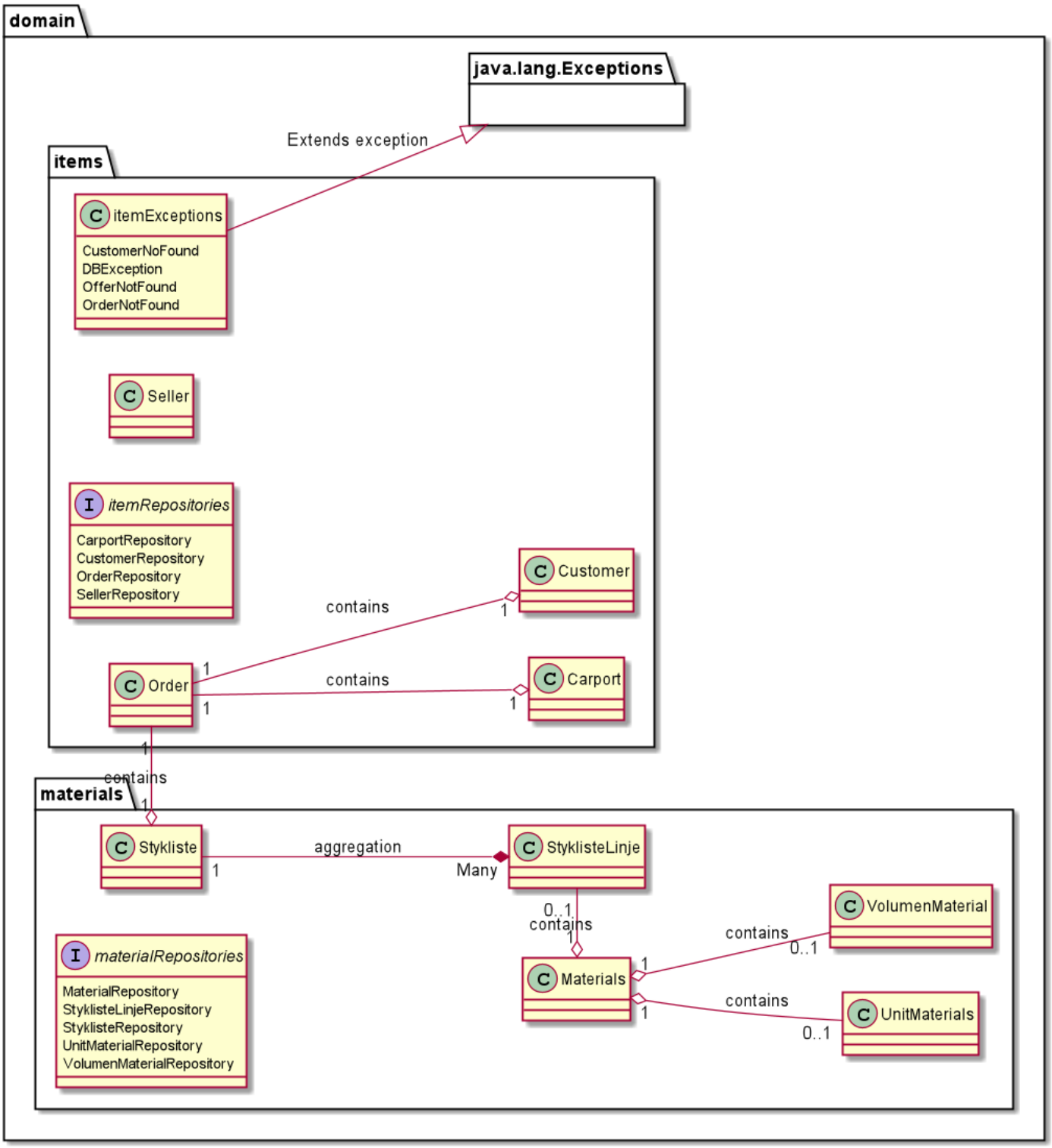
Password til sælgerkonto:

En ny sælger kan oprettes, når der logges ind med administratorrettigheder. Der indtastes sælgers fulde navn, og sælgers ønskede username. Ved at klikke på 'opret' knappen, bliver en ny konto med sælgerrettigheder oprettet, med det ønskede username, og password '1234' som standard. Første gang sælgeren logger ind på sin konto, bliver man promptet til at ændre sit password. Det er ikke muligt at have passwordet '1234', da man bliver promptet til passwordskift ud fra passwordet. Derudover er det ikke muligt, at lave et tomt password.

Der bliver ligeledes saltet og hashet på sælgerens password.

SYSTEM DESIGN

4. Domæne Diagram



4.1 Domænemodel

Modellen og systemets overordnet Domæne indeholder de elementer eller entities som systemet fundamentalt er bygget op omkring, eller den virkelighed som modelleres.

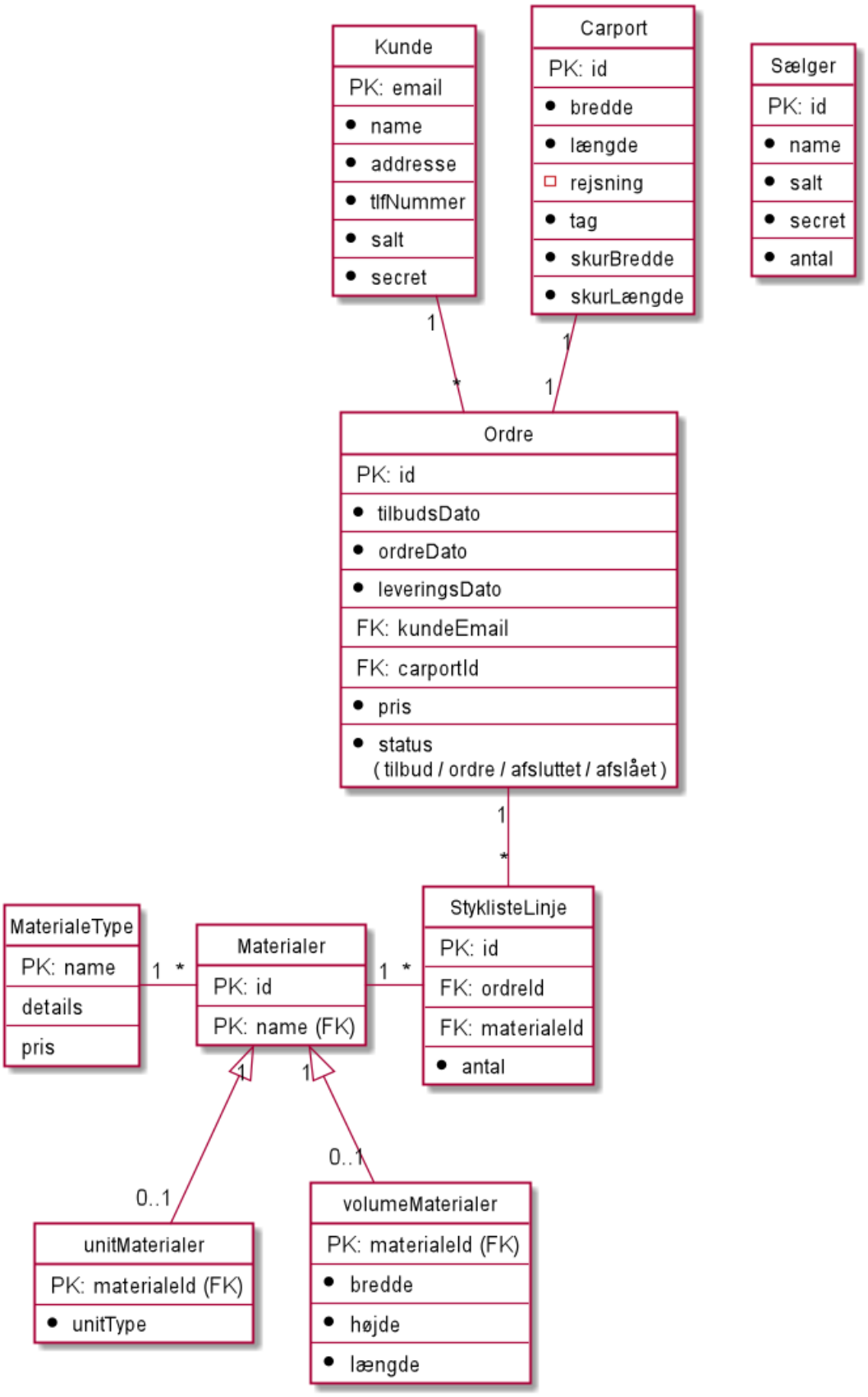
Da systemet har til opgave at dynamisk bestille Carporte, med tilhørende funktionalitet som kræves af kunden - (Fog's byggemarkedet) og endeligt de behov som bruger har i forbindelse med en bestilling, skal domænet indeholde de klasser/objekter som gør det muligt for systemet at udføre opgaven.
F eks Carport, materialer m.m.

Domænet er delt i to dele, hvor den ene del 'items' indeholder Carport og det som administrativt følger med, hvor den anden del 'materials' indeholder alle de specifikke materialer der skal benyttes, med pris.

Repositories er ikke objekter men Interfaces af domæne klasserne med metoder som da implementeres i lag: 'Infrastructure'.

IMPLEMENTATION

1. Database, og ER Diagram



IMPLEMENTATION

2. Tabel relationer

Fog databasen er bygget op af 9 tabeller.

Carport-tabellen indeholder 'Skur' i form af to felter: skurBredde & skurLængde, som kan være 'NULL' i tilfælde af skur ikke ønskes af bruger,

Ordre-tabellen - Carport-tabellen er 1:1 relation og har CarportId som Foreign-key. Status-felt tager som String parameter om en ordre er hhv.

1. et "Tilbud" = en Ordre som ikke er accepteret,
2. en "Ordre" = en ordre som er accepteret.
3. "Afsluttet" = sættes af sælger når bruger/kunde har betalt og modtaget stykliste.
4. "Afslået" = sætter af sælger når bruger/kunde ikke accepterer et tilbud.

Kunde-tabellen er beskrevet i IMPLEMENTATION 3. Sekvensdiagram, Process.

Styklisten er bygget op af flere stykliste linjer.

StylisteLinje-tabellen - Ordre-tabellen er mange:1 relation forstået at der kan være mange Stykliste Linjer med samme Foreign key: OrdreId, der tilsammen udgør Styklisten til en bestemt ordre.

Stykliste linjer er bygget op af flere materialer
StyklisteLinje-tabellen har Foreign Key: materialeId, fra Materiale-tabellen.

StylisteLinje-tabellen - Materiale-tabellen relation er mange:1, da der kan være mange Stykliste linjer med hver 1 materiale per linje.

Materialer er bygget op af enten, UnitMateriale eller volumenMaterialer.
Materiale-tabellen har Primary Key: Id, samt Primary & Foreign Key: name.

Alle materialer har altså et Foreign Key: navn, som hentes fra MaterialeType-tabellen der tilfører materialebeskrivelser, detaljer samt pris.

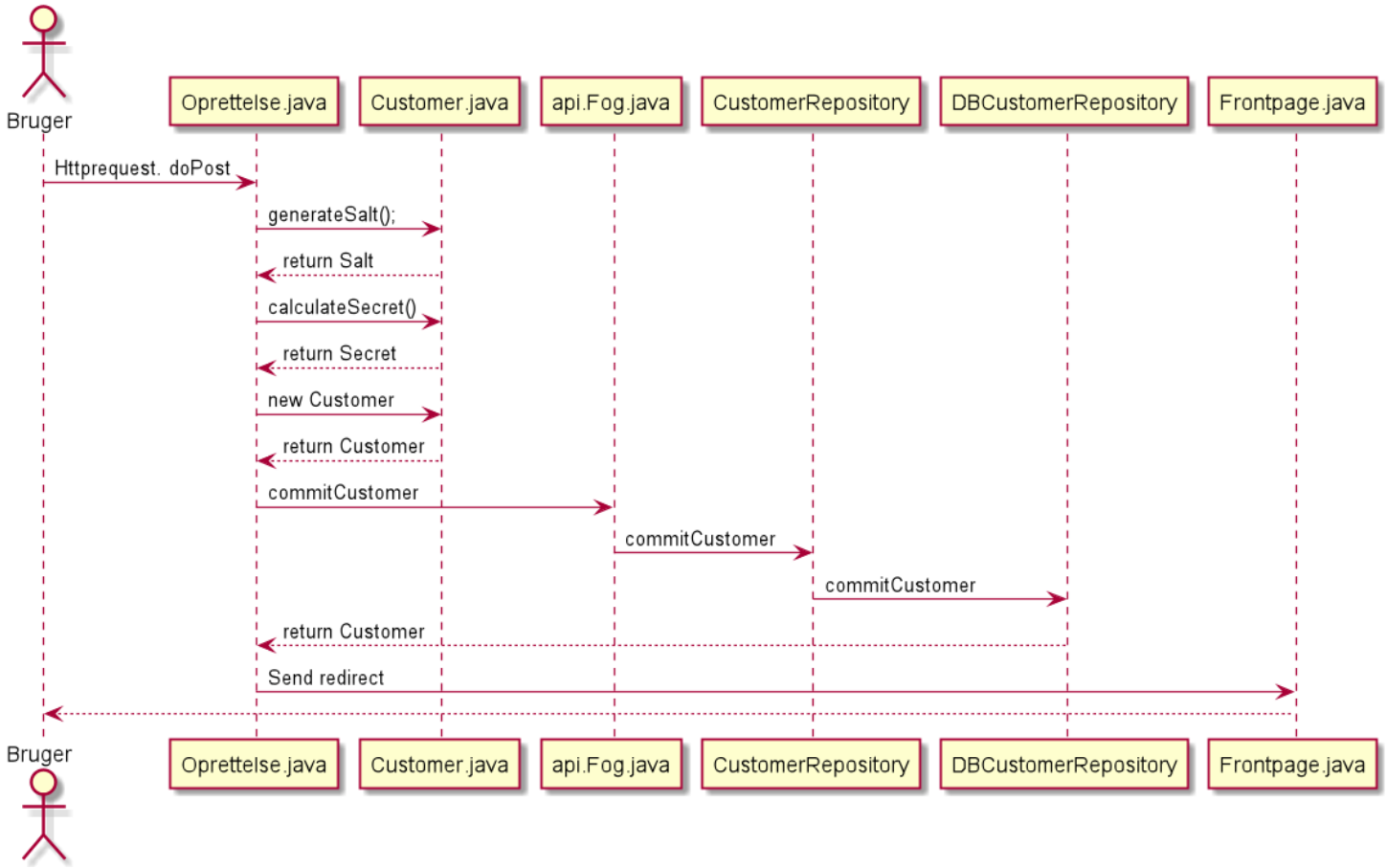
UnitMateriale-tabellen og volumenMateriale-tabellen har 0 eller 1:1 relation til Materiale-tabellen.
Et Materialenavn er da enten et Unit eller -Volumen materiale.

IMPLEMENTATION

3. Sekvensdiagram

User Story:
Som bruger vil jeg gerne kunne oprette mig i systemet som kunde.

Diagrammet illustrerer processen når en bruger skal oprette en konto og dermed tilgå funktionalitet som: få tilbud, bestilling, følge ordre og tilgå stykliste.



Process:

Bruger aktiverer 'Opret bruger' i nav-baren.

I *opretBruger.jsp* er en HTML post-form hvor bruger kan indtaste de forskellige parameterværdier som skal til for at oprette en bruger i Databasen.

De indtastede værdier bliver handlet fra Webservlet *Oprettelse.java:* `doPost` metode.

Password værdien bliver krypteret, ved at kalde de 2 metoder, 'Salt' & 'Secret', i *Customer* klassen som returnerer de krypterede byte-type: 'Salt' og 'Secret'.
Kryptering er detaljeret beskrevet i: 'SYSTEM DESIGN 6. Sikkerhed'

Parameterværdierne fra post-formen samt Salt og Secret bliver da brugt til at oprette et nyt Customer objekt, 'new Customer', i *Customer* klassen.

Customer bliver så ført videre som parameter til at kalde `commitCustomer` metoden i *api.Fog,* som endvidere kalder `commitCustomer` i Interface: *Customer.Repository.* `commitCustomer` er implementeret i *DBCustomerRepository.*

Fra *DBCustomerRepository* implementeret `commitCustomer` bliver alle parameterværdier indlæst til 'kunde' tabellens felter i Fog MySql database.

Metoden `commitCustomer` returnerer en Customer til den oprindelige *Oprettelse.java* klasse, metode `api.commitCustomer`.

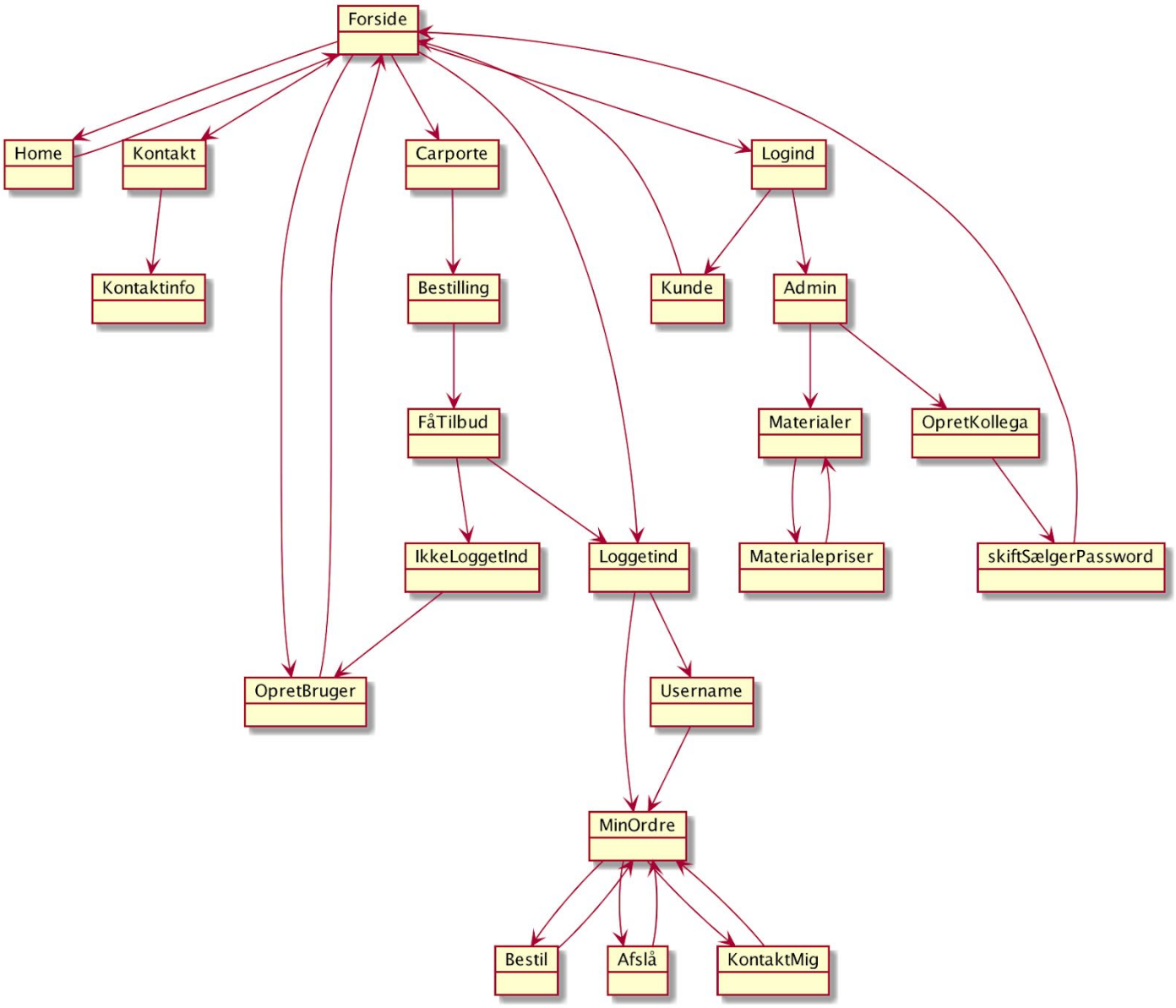
Oprettelse.java `doPost` metode laver da et Http response til Tomcat serveren og Webservlet *Frontpage.java* som med `doGet` metoden renderer tilbage til brugeren i browseren med *Frontpage.jsp.*

IMPLEMENTATION

4. Navigationsdiagram

Diagrammet beskriver aktørerne: bruger- og administrators, brug af systemet.
Der tages udgangspunkt på forsiden, hvor der er forskellige links og buttons - 'Home', 'Carporte', 'Kontakt', 'Opret Bruger' og en 'Log ind'-dropdown.

Er man allerede logget ind, vil ens brugernavn(e-mail) vises i navigationsbaren.
Dette fungerer også som et link til ens ordreside. Flowet igennem de forskellige sider samt valg der træffes, vises vha. pile.



IMPLEMENTATION

5. Teknologier, Tomcat, Droplet

Projektet er udført i IDE: IntelliJ, kodet i Java, jsp og Mysql.
Til test er der brugt Junit.
Det færdige projekt er lagt op på en Tomcat server på en droplet hos udbyder: Digital Ocean, der kører på Ubuntu Linux.
I det følgende vil jeg forklare de forskellige teknologier og hvorfor vi har valgt dem.

5.1 Java

Projektet er skrevet i Java, openjdk version "11.0.9.1".
Årsagen til at vi bruger Java er pga bedst kendskab, og fordi med Java er forholdsvis enkelt at kommunikere med databaser, samt god datadynamik ved brug af servlets som styrer hvilke websider og indhold.

5.2 Jsp

Websiderne er skrevet i Jsp. Vi har valgt jsp og ikke i ren html, da det i jsp er muligt at gøre indholdet dynamisk i modsætning til html som er statisk.
Fordelen ved jsp er at vi kan føre data fra servlet til jsp siden enten som parametre eller attributter. Det giver i udvikling flere dynamiske muligheder.

Vi har valgt at udvide JSP med JSTL 1.2 (JavaServer Pages Standard Tag Library) som er et bibliotek med mulighed for at bruge betingelser f.eks. (<c:if>) og loope lister igennem (<c:forEach >) samt andre muligheder, dog de nævnte tags er benyttet mest.

5.3 Bootstrap

Til at gøre opstillingen af websiderne mere brugervenligt har vi brugt Bootstrap framework 4.5.3. Bootstrap er en udvidelse af JSP som organisere sideindhold visuelt logisk, samt muligheden for relativt enkelt at styre input data og felter.
Det gør håndteringen af inddata end del lettere, eks. ved heltal input ikke er en tekststreng.

5.4 Tomcat

Vi har valgt Apache Tomcat 9.0.22 som vores webserver, som et naturligt valg til at håndtere java servlets i små projekter.
Begrænsningen er at kun et relativt mindre antal samtidige har tilgang.

5.5 MySql

Vi har valgt at vores database skal være en relationel database, valget af sprog er faldet på SQL, da det er standarden.

Databasen er MySQL 8.0.22, da MySQL er en god gratis SQL database som rummer mange muligheder. Den fungerer godt med Java, kan tilgås fra et terminalvindue samt få godt visuelt overblik over tabeller og data i MySQL-Workbench, der også kan bruges til redigering.

5.6 Digital Ocean

Vi har valgt at få vores projekt hosted hos Digital Ocean.
Fordelen er adgang til en tom ubuntu server mod mindre omkostning, hvor man kan installere de ønskede services (Java,MySQL,Tomcat...).
En enkel og fleksibel løsning.

5.7 Github

Github til versionsstyring, merge kode alle gruppemedlemmer.

Github gør det muligt at kode i hver sin “branch” for så at “merge” med en master branch, efter hvert medlem har lavet sin del som virker lokalt.

5.8 Intellij Ultimate

IDE Intellij Ultimate 2020.3, fungerer fint sammen med MySQL, GitHUb og Tomcat.

5.9 Junit

Til test af vores java har vi brugt JUnit, da det er nemt at bruge i IntelliJ.

IMPLEMENTATION

6. Status

Fog projektet er udviklet, implementeret, testet og funktionelt på en Tomcat server hostet på en droplet hos Digital Ocean.

Rapporten her er ligeledes fyldestgørende efter de krav til rapportskrivning vi er givet.

Vi føler vi har lavet et velfungerende og veldokumenteret projekt vi er yderst tilfredse med.

6. 1 Udeladelser.

Vi har ikke implementeret 'Carport med rejsning' i projektet.
Dvs man kan kun vælge en carport med fladt at.
Det beror på at vi har prioriteret at gøre de andre ting mere solidt og så vente med den option til en eventuel næste version.

Vi nåede ikke denne næste version.

Da specifik information om hvilke brædde længder som findes ikke foreligger, samt vi ikke har fået oplyst hvordan to brædder evt. kan sammensaættes, har vi forudsat at alle stykker træ findes i alle længder fra 1800 mm til 7200 mm.
Det har endvidere forenklet udregning af styklisten markant.

Sælger kan i vores implementaion rette i styklisten, dog kun i antal af elementer.

Det giver således da heller ej mening at kunne ændre på længden på brædderne når længden afhænger af størrelsen på carporten, som sælger kan ændre.
Sælger kan ikke tilføje til styklisten.

6.1 Ekstra

Vi valgt at når systemet bliver installeret oprettes en sælger med brugernavn: "admin" og password "1234".

Når en ny sælger oprettes i systemet af en admin. tildeles de således først også password "1234".
Vi har så implementeret det sådan at når en sælger logger ind første gang og password er "1234" bliver han bedt om at vælge et nyt pasword.

Dette mener vi er en god og sikker løsning.

PROCESS

1. Test. Unit og integrationstest

Vi har lavet både unit og integrations test vha junit5.

Unit testene er lavet løbende og blevet kørt hver gang programmet er startet.
På denne måde har vi hele tiden holdt styr på ændringer og evt tilfælde af fejl på units som tidligere har virket.

Integrationstestene har vi selv valgt hvornår de skulle køres, da det er en større proces en unit tests pga opsætning af testdatabase.

Vi har kun testet java delen vha Junit test og ikke webdelen.
webdelen er blevet testet ved hands-on test, der fungerer som en del af integrationstesten.
Hands-on ved at eks. at teste login, hvor vi online test taster korrekt og forkert, for at observere hvorledes begge scenarier handles af programmet.

1.1 Unittest.

Vi har Unit testet domain og dele af api.

Generelt har vi kun unit testet de ting som indeholder logik, dvs ikke gettere, settere, toString osv.
I api har vi ikke testet Fog, da den er indirekte testet ved kun at indeholde kald af ting som allerede er testet, samt kald af DB gennem repos, som igen bliver testet via integrationstest.

1.2 Integrationstest.

Integrations testene kræver en testdatabase.
Test Databasen oprettes før hver enkelt test med migration-script som starter med at slette den tidligere testdatabase, før den opretter en ny.
Det gør at testen altid svarer til en rigtige database hvor kunder, sælgere, carporte, ordrer og styklister læses ind.

På denne måde bliver alle tests kørt fra samme udgangspunkt hver gang, hvor tidligere tests slettes og ikke konflikter for nye.

<u>User Stories;</u>

Vi har integration testet alle User Stories således at vi opretter et test-Scope der inddrager, og kalder, alle de samme api kald som bliver kaldt i Servlets til den pågældende User Story. Vi kan da se om udkommet er mod forventet.

Efterfølgende har vi hands-on testet den pågældende User Story online.

1.3 Hands-on test

Hands-on test af User Stories:
F.eks ved User Story #25: 'Som bruger vil jeg gerne kunne oprette mig i systemet', foretages som følgende online på websiden.

1. Tryk på "Opret Bruger" i nav-baren's højre hjørne.
2. Udfyld de 5 felter
3. Tryk på knappen "Opret" i bunden.

Man kan nu se at brugeren er oprettet da brugerens e-mail bliver vist direkte i nav-baren, samt en "Log Ud" knap bliver tilgængelig i øverste højre hjørne.

PROCESS

2. Evaluering

Som nævnt i IMPLEMENTATION 6. Status, er vi tilfredse med projektet trods manglende 'Carport med rejsning' implementation.

Vi har fulgt Onion architecture modellens principper og logisk designet et projekt som fungerer godt efter formålet.

En del valg og overvejelser er foretaget i udviklingen hvor vi har diskuteret pro og con's samt konsekvenser.

Tidsdisponering kunne have været mere struktureret.

PROCESS

3. Gruppe, og gruppearbejde

Gruppearbejdet har fungeret med god disciplin fra alle gruppemedlemmer.
Alle gruppemedlemmer har bidraget.
I projektet er der øverst i filerne indikeret hvilke medlemmer som har har kodet pågældende fil eller afsnit.

Vi har fuldt scrum principperne, og som vi tidligere aftalte i kontrakten har vi holdt dagligt morgenmøde med gennemgang, -follow ups, -reviews, -problemstillinger, -hjælp og -status. Hverdage kl. 9,
Weekender kl. 10. (nogle)

Det har hjulpet på arbejdsfordeling og til at alle medlemmer har været i synch samt været en bidragende del af projektet med hjælp, råd og vejledning.

En ansvarsfordeling blev fordelt som følgende fra start:

- Scrummaster/Taiga: Gustav.
- Github logistik & Database struktur: Jens
- Databaseopsætning & rapportstruktur: Peter

AFSLUTNING

1. Afrunding

AFSLUTNING

1. Referencer

- Fog's hjemmeside
- Navbar: https://getbootstrap.com/docs/4.0/components/navbar/