19. DECEMBER 2018

FOG CARPORT

Gruppe: J M R

Javadocs: https://mf226.github.io/fogCarport

Source code: https://github.com/mf226/fogCarport

Deployment: http://104.248.17.255/FogCarport-0.1-Test/FrontController

*Note: Lav eget login som customer og brug email: **admin** og password: **admin** for at bruge adminfunktioner.*

MADS FLØISTRUP	JONATAN MEJER HJELM	RASMUS PORSE
cph-mf266@cphbusiness.dk	cph-jh409@cphbusiness.dk	cph-rp127@cphbusiness.dk
mf266	JonatanHjelm95	RPorse

Indhold

Indledning

-Rasmus

Fog projektet er et eksamensprojekt for 2. semester. Produktet består af et website og en database som er udviklet efter kundens ønsker og behov. Designet består af en tre lags arkitektur, der kører på en java server.

Processen til udarbejdelse af projektet er gjort ved brug af Scrum med ugentlige møder med product owner. Denne rapport er lavet med udgangspunkt i projektet "fogCarport", der findes på github.com (link på forside).

Baggrund

- Fog Trælast og Byggecenter er en virksomhed, med hovedsæde i Nordsjælland, der specialiserer sig i at have et stort sortiment af træ og byggematerialer til alle slags opgaver. Specielt for Fog er deres mulighed for at kunne levere skræddersyede carporte efter deres kunders behov.
- Fogs kunder skal kunne vælge mål på en carport: højde, længde, bredde og fladt/skråt tag samt tilvalg af skur. Der skal ud fra de indtastede data, sammensættes en stykliste af materialer der skal bruges til at bygge carporten og der vises en skitse af carporten på skærmen. Ordren bliver gemt i databasen med en afventende status.
  Logget ind som admin, skal man kunne godkende ordrer og ændre status på dem. Herinde skal admin også kunne ændre prisen på den samlede ordre.

Teknologivalg

Nedenfor er en liste over de teknologier, som har været nødvendige for projektet:

- Netbeans IDE 8.2
  - Mysql Connector Java 6.0.5
  - Java EE Web API 7.0
  - jUnit 4.12
  - Java Hamcrest Core 1.3
  - JaCoCo Code Coverage 0.7.7
- MySQL Workbench 6.3
- Digital Ocean Cloud service

Krav

-Rasmus

I dette afsnit beskriver og analyserer vi Fogs håb/vision for udviklingen af systemet, samt hvilken værdi det vil tilføre deres virksomhed. Analysen er ud fra SCRUM userstories, SWOT- og interessentanalyse.

Overordnet beskrivelse af virksomheden

Rasmus

Nedenstående interessentanalyse belyser projektets forskellige interessenter, deres fordele/ulemper ved projektet, samt en vurdering af dem og hvordan de håndteres i projektforløbet.

Interessent	Fordele ved projektet	Ulemper ved projektet	Vurdering af interessent	Håndtering
Product owner (PO)	Ny struktur af platform med øget brugervenlighed.	Tilvænning af den nye platform.	Har stor interesse og bidrag til projektet og er med til at bestemme hvilke user stories der skal arbejdes med.	Ugentlige møder med udviklere afgør hvilke områder der skal behandles i kommende uge.
Udviklere	Erfaring med produktudvikling og samarbejde med virksomhed og at produktet potentielt bliver bedømt af mange.	Fremrykket deadline.	Denne interessent har stor interesse og om muligt den største indflydelse, da de står for selve udviklingen af platformen.	Er til de samme ugentlige møder med PO.
Medarbejdere	Øget brugervenlighed og arbejdsbyrden mindskes.	Tilvænning af den nye platform.	Interessentens indflydelse og bidrag vurderes lavt.	Efter end projektudvikling bør der laves vejledninger til interessenten i hvordan platformen betjenes.
Vejleder	At følge udviklingen af udviklerholdet.	Intet nævneværdigt	Har middel indflydelse og bidrag til projektet.	Har et ugentligt møde med udviklerholdet.

SWOT analysen af Fog er med udgangspunktet i det udleverede materiale, Fogs nuværende hjemmeside og videointerviewet med afdelingschef Martin Kristensen. Efterfølgende er der en SWOT analyse over projektets chance for at blive en succes set fra udvikler teamets perspektiv og efterfølgende er der en konklusion på begge analyser.

SWOT KUNDE	Helpful *…to achieving the objective*	Harmful *…to achieving the objective*
Internal Origin *(Attributes of the organization)*	***STRENGTHS***	***WEAKNESSES***
	Unikt produkt Veletableret organisation Dialog med deres kunder Detaljerede vejledninger	Forældet teknologi Begrænset tilpasning database Byg-selv platform • Skur placering
External Origin *(Attributes of the environment)*	***OPPORTUNITIES***	***THREATS***
	Konkurrents produkt ikke gennemskueligt Bedre visualisering	Konkurrent med samme ydelser

Strengths

Fog er en stor virksomhed bestående af ni trælaster og et bolig og designhus, som hovedsageligt er samlet i Nordsjælland med et enkelt i Sydsjælland. I trælasten, som vi fokuserer på, er Fog som mange andre byggemarkeder, men det som er vigtigt her er deres carporte. Martin beskriver deres produkt som noget unikt. Han fortæller at andre byggemarkeder ikke leverer samme produkt og de har brugt mange

ressourcer på at udvikle deres platform, så de ikke kun laver præfabrikerede carporte, men at kunderne selv kan vælge mål. Hvis deres kunder beslutter sig for at ville have andre dimensioner eller udseende af træet/delene til carporten, så er Fog fleksible og leverer det kunden ønsker.

Martin mener at det er vigtigt for dem at have en god personlig kontakt med deres kunder. Han beskriver at salgsmedarbejderen kontakter kunderne, hvis de mål der er sendt ind ikke er realistiske. Hvis Fogs kunder vælger byg-selv, kontaktes kunderne for at følge op på hvordan projektet skrider frem og skaber derfor en mere personlig kontakt.
Vejledningerne til byg-selv projekterne bliver beskrevet som yderst detaljeret, hvilket gør det praktisk for deres kunder.

Weaknesses

Fog har brugt mange ressourcer på udviklingen af en ny platform, som desværre ikke kan snakke sammen med deres nye IT-system. Dette er et problem da de bliver nødt til at bruge endnu ældre programmer, som ikke er opdaterede og det er derfor at de ønsker en ny platform.
Et dilemma der bliver omtalt er at adgangen til deres database er begrænset i sådan et omfang at det kun er muligt at ændre priserne på allerede eksisterende varer. Dette betyder at de heller ikke kan ændre navnene eller tilføje nye materialer til databasen.
Byg-selv platformen er et program salgsmedarbejderen bruger til at taste kundens mål ind, få styklisten af carporten og til sidst til at justere prisen. Byg-selv platformen er ikke integreret med lagerstyringen hvilket medfører at priser og materialer skal opdateres manuelt for at de vises korrekt på platformen.
Ydermere kan skurets placering ikke vælges, så det vises korrekt på skitsen, hvilket betyder at skuret altid vil have samme placering på skitsen.

Opportunities

Martin fortæller at konkurrentens produkt er ikke gennemskueligt i forhold til prisen. De udleverede materialer, som skitser og andre visuelle medier, bliver beskrevet som bedre end konkurrenternes.

Threats

Fogs nærmeste konkurrent leverer næsten det samme produkt. Martin beskriver at de bygger deres carporte ud fra grundmodeller og så kan kunderne have forskellige tilvalg.

SWOT TEAM	Helpful *…to achieving the objective*	Harmful *…to achieving the objective*
Internal Origin *(Attributes of the organization)*	***STRENGTHS***	***WEAKNESSES***
	Gode færdigheder indenfor programmering God gruppedynamik Overskuelig opgave Nuværende design af platform	Erfaring Geografi Lille udviklerhold
External Origin *(Attributes of the environment)*	***OPPORTUNITIES***	***THREATS***
	Stor kontrol over projektet Ugentlige møder med vejleder Sparring med klassekammerater	Deadline Misforståelser mellem gruppe, vejleder og PO Sygdom - Gruppe/Vejleder Konkurrentens gennemskuelighed

Strengths

Flere fra udviklerholdet har erfaringer med at kode fra tidligere studie/arbejde, der giver holdet et solidt fodfæste når der opstår nye opgaver. Dynamikken anses som

værende god i den forstand, at alle medlemmer giver plads til hinanden, når der er problemer der skal løses.
Opgaven som den ser ud nu, virker til at være overskuelig for holdet og med den baggrund holdet har, vurderes de til at kunne løse opgaven til et acceptabelt niveau.
Efter inspektion af Fogs hjemmeside vurderes de til at have et overordnet design der ser brugervenligt ud, som er et godt udgangspunkt fx ved design af den nye platform, så det er familiært for medarbejderne.

Weaknesses

Ikke alle har erfaring med at kode og dette kan blive et problem når opgavens kompleksitet eller arbejdsbyrden stiger. Holdets størrelse kan også blive en hindring, hvis holdet skal indhente arbejde, der på den ene eller anden måde er blevet forsømt.
Derudover bor holdet langt fra hinanden, der kan give problemer i forhold til at mødes, da transporttiden er uforudsigelig til tider.

Opportunities

Dette projekt er unikt for holdet, da det giver mulighed for at kunne forme projektet i retninger, der kunne være interessante for holdets egen udvikling. Dette kunne være at lægge vægt på områder der allerede er erfaringer med, eller udfordre holdet selv med opgaver der er nye, som kunne lede til at give projektet unikke features.
De ugentlige møder med vejleder ses som en mulighed til at få eventuelle fejl eller misforståelser rettet.
Da det er et studie, så giver det rig mulighed til at bruge klassekammerater til brug af sparring og diskutere valg af løsninger til problemer og opgaver.

Threats

En risiko der kan opstå er misforståelser. Ikke bare internt men også mellem udviklerholdet, vejleder og product owner. Det kan selvfølgelig give anledning til brud på de ugentlige deadlines, der kan medføre mangelfuld kode eller at andre ting kommer til at mangle til den endelige deadline.
Sygdom er også en stor risiko, som kan være problematisk, hvis spørgsmål omkring projektet står ubesvaret, fordi udviklerholdet, vejleder eller PO er syge.

I forrige SWOT analyse under *Opportunities* beskrives der at konkurrentens produkt ikke er gennemskueligt. Dette er ikke dybere forklaret og giver anledning til spørgsmål om hvorvidt dette er fakta og om det kunne være en potentiel trussel. En afklaring af dette kunne sørge for at udviklerholdet får gennemskueligheden med i projektet.

Vurdering

Overordnet set så har Fog overvejende flere styrker end svagheder. Fogs unikke design og imødekommenhed, gør at de skiller sig fra de almene byggemarkeder, hvilket giver dem en fordel over andre byggemarkeder. Desværre leverer en af deres konkurrenter et produkt der er familiært i forhold til Fogs. Forskellen er dog at Fogs kunder har mulighed for en komplet skræddersyet carport, hvor de kan vælge bestemte typer træ, tag mm. og samtidig selv vælge grundmålene på carporten, som er det der gør Fogs produkt unikt.
Deres svage side er dog den begrænsede adgang til databasen, som kan risikere at en ny database skal sættes op.
Udviklerholdet er forholdsvist nyt, men har meget at byde på med styrkerne kontra svaghederne og de ugentlige møder med vejleder og PO hjælper til at holdet forbliver målfaste.
Da Fogs styrker er mange, giver det en god basis for videre samarbejde og en ny platform vil betyde at Fog får program der kan snakke sammen med deres nye IT-system, løse deres svagheder, samt hjælpe med at holde deres produkt unikt.

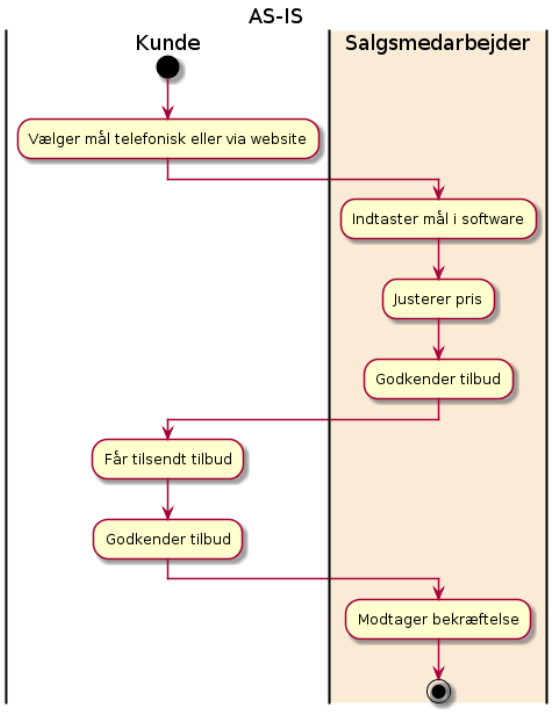
Arbejdsgange der skal IT-støttes

-Rasmus

AS-IS

Den nuværende arbejdsgang hos Fog er vist i nedenstående aktivitetsdiagram(AS-IS). Den er udarbejdet fra det udleverede videomaterialet.

Fogs kunder kontakter deres salgsmedarbejder ved enten at ringe direkte, eller indtaste ønskede mål inde på hjemmesiden, som så sender en email til salgsmedarbejderen.



Salgsmedarbejderen vurderer om målene er realistiske og taster dem ind i systemet, som beregner målene. Der bliver dannet en stykliste og medarbejderen kan nu justere prisen som godkendes og sendes videre til kunden.

Kunden modtager tilbuddet og kan vælge at godkende prisen, som fører til at medarbejderen får sendt en bekræftelse af ordren.

AS-IS vurdering

I den nuværende arbejdsgang er der meget manuel indtastning af data fx når kunden har sendt målene til salgsmedarbejderen, så skal de indtastes i systemet. Dette er et led der kan springes over ved at kundens valg kommer direkte i behandling i systemet og beregnes lige så snart det er tastet ind. Da der i dette led ikke foregår nogen medarbejder-kundekontakt, som er vigtigt for Fog, så er det et oplagt at fjerne dette led.

Alle data bør gemmes så det ikke går tabt pga. fx menneskelige fejl, så derfor skal databasen have alle informationer på ordren så snart den er godkendt af kunden. bliver så opdateret når salgsmedarbejderen har godkendt den.

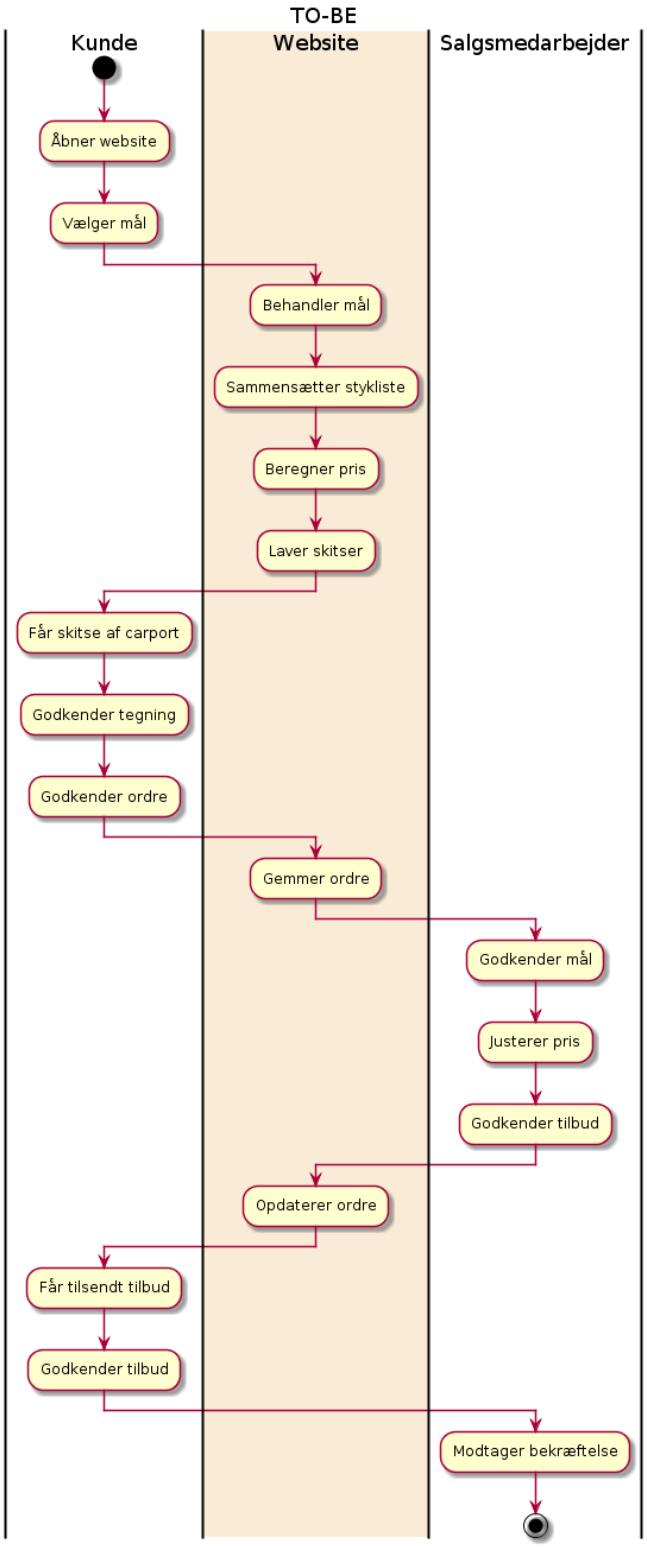
Med disse ændringer af arbejdsgangen, så vil den fremtidige arbejdsgang se ud som på “TO-BE” diagrammet.

TO-BE

Arbejdsgangen som ønskes er vist på skitsen “TO-BE”. Den er udarbejdet fra Fogs ønskede arbejdsgang.

Kunden vælger mål på hjemmesiden, som behandler dette og sammensætter stykliste, beregner prisen ud fra materialepris og laver skitser af carporten. Skitsen bliver vist på hjemmesiden for kunden. Kunden kan nu godkende tegningen og ordren på carporten.

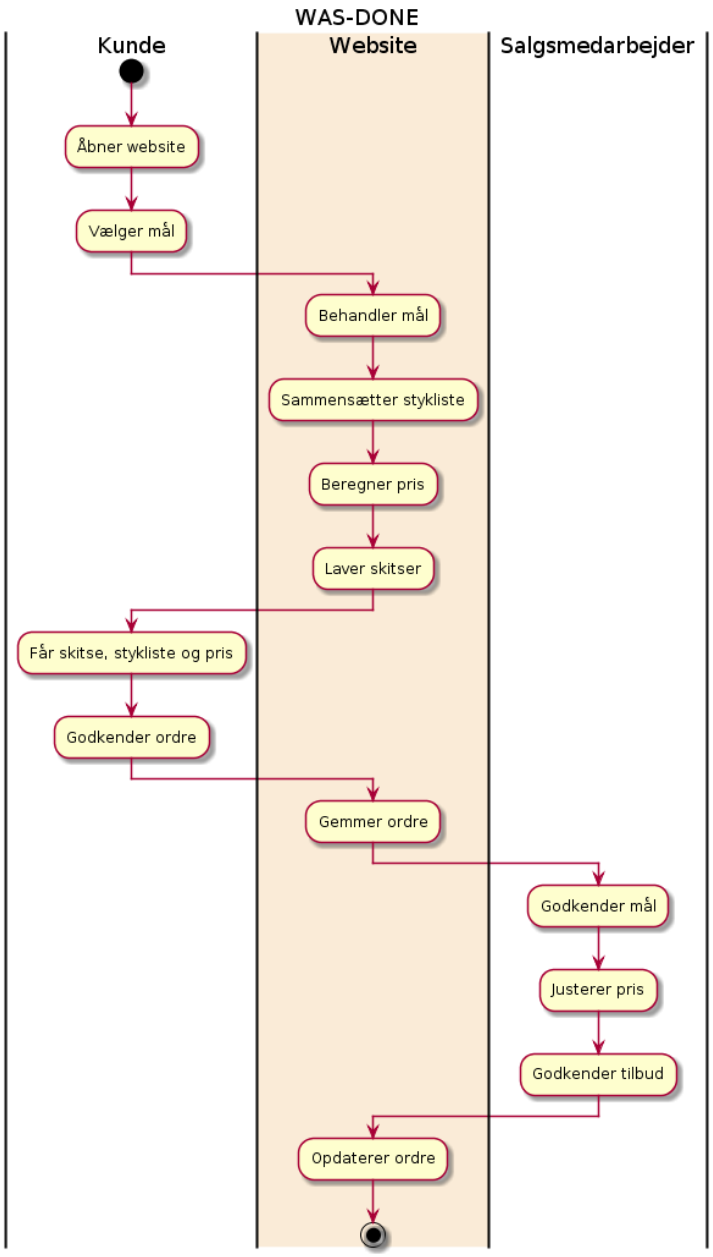
Ordren gemmes i databasen og bliver sendt videre til en salgsmedarbejder, som nu skal godkende om målene er korrekte. Medarbejderen justerer prisen på ordren efter behov og godkender tilbuddet. Ordren opdateres i systemet og sendes til kunden. Her godkendes tilbuddet og bekræftelse sendes til medarbejderen.



WAS-DONE

Efter endt projektperiode er det endelige produkt endt ud som på “WAS-DONE” diagrammet.

Kunden vælger mål på hjemmesiden, som behandles i systemet. Stykliste sammensættes og pris beregnet ud fra dette. Dette bliver sendt til kunden som nu kan godkende ordren. Den bliver så gemt i databasen og salgsmedarbejderen kan nu se den i systemet. Medarbejderen kan justere pris og godkende tilbuddet som opdateres i databasen.



Scrum userstories

-Mads

Til udarbejdelse af userstories er målet at følge INVEST princippet så meget som muligt. INVEST står for:

- Independent
- Negotiable
- Valuable
- Estimable
- Small
- Testable

Det vil sige at en userstory som udgangspunkt skal kunne løses uafhængigt af andre userstories og dermed kunne rykkes rundt fra sprint til sprint. Hvis en userstory af for afhængig af en anden bør det overvejes om de skal kombineres til én. En userstory er ikke en kontrakt, men skal i stedet kunne diskuteres så den bedste løsning findes. Derudover skal en userstory skabe værdi for productowner og arbejds størrelsen skal kunne estimeres. En opgave må ikke være så stor at den bliver umulig at planlægge og hver userstory skal kunne testes.

Da dette er et skoleprojekt, og udvikling af tests først har været et emne i løbet af semesteret har udviklingen af test haft user stories for sig selv. Det kan man argumentere for skulle være en mere løbende process, hvor en udvikling af en test er en del af at færdiggøre en userstory, så man løbende har fungerende og vel testet kode.

I SCRUM er det normalt PO der er ansvarlig for backlog og opdatering af userstories. I dette projekt er det klaret i samarbejde med product-owner, men med SCRUM-teamet som ansvarlig for opdatering af backlog mellem PO-møder, Fra PO-møderne blev en række userstories beskrevet som løbende blev opdateret som projektet tog form. Det følgende er en liste af alle userstories der blev aftalt med product-owner.

- Valg af mål
- Udregning af stykliste
- Beregning af spærmængde
- Beregning af antal stolper
- Beregning af pris

- Tilføjelse af materialer til stykliste, step 2 (skruer, beton, rem, beslag)
- Skitse af Carport fra siden (SVG)
- Skitse af carport top-down (SVG)
- Tilvalg af skur
- Valg af beklædningstyper
- Valg af tagtype
- Toggle mellem SVG tegninger
- Test af logicfacade
- Test af database
- Test af brugerinterface
- Kontroller kundens valg af mål (admin)
- Kontroller kundens valg af skur (admin)
- Mulighed for ændring af samlet pris
- Generer PDF
- Lagerbeholdning
- Mulighed for online betaling
- Mulighed for at tilføje nye varer til DB
- Mulighed for ændringer i DB

Det følgende er en komplet tabel der beskriver vores userstories for sprint 4.

Lignende tabeller kan findes for de resterende sprint i appendix.

ID	Name	Est	Done	How to demo
#7	Kontroller kundens valg af mål	4	Mulighed for at kunne kontrollere valg af mål og godkende samme.	Åbn browser og log ind som admin. Adminsiden kan vælges fra drop down. På denne side vises kunders ordrer. Vælg “Review-knappen” ud for den ønskede ordre. Tryk på knappen “godkend”. Bemærk nu at status for ordren har ændret sig fra “pending” til “approved”.
#16	Kontroller kundens valg af skur	2	Mulighed for at kunne kontrollere valg af mål og godkende samme.	Åbn browser og log ind som admin. Adminsiden kan vælges fra drop down. På denne side vises kunders ordrer. Vælg “Review-knappen” ud for den ønskede ordre. Tryk på knappen “godkend”. Bemærk nu at status for ordren har ændret sig fra “pending” til “approved”.
#11	Mulighed for ændring af samlet pris	1	Kunden skal have mulighed for at kunne ændre den samlede pris.	Åbn browser og log ind som admin. Adminsiden kan vælges fra drop down. På denne side vises kunders ordrer. Vælg “Review-knappen” ud for den ønskede ordre. Skriv i feltet, den nye

				samlede pris for carporten og tryk på knappen “godkend”. Bemærk nu at status for ordren har ændret sig fra “pending” til “approved”.
#61	Generer PDF	4	Når brugeren kan få en PDF af den indtastede ordre.	Tryk på “Design carport” og vælg enten “Med fladt tag” eller “Med skråt tag”. Indtast de ønskede mål og vælg “Fortsæt”. Vælg her “Gem som PDF” og download starter.
#49	Lagerbeholdning	4	Når kunden kan se lagerbeholdning på adminsiden og bestille nye varer.	Log ind som admin. Åbn adminsiden og se listen af materialer på lager. Vælg bestil nye varer og kontroller at listen bliver opdateret med flere varer.

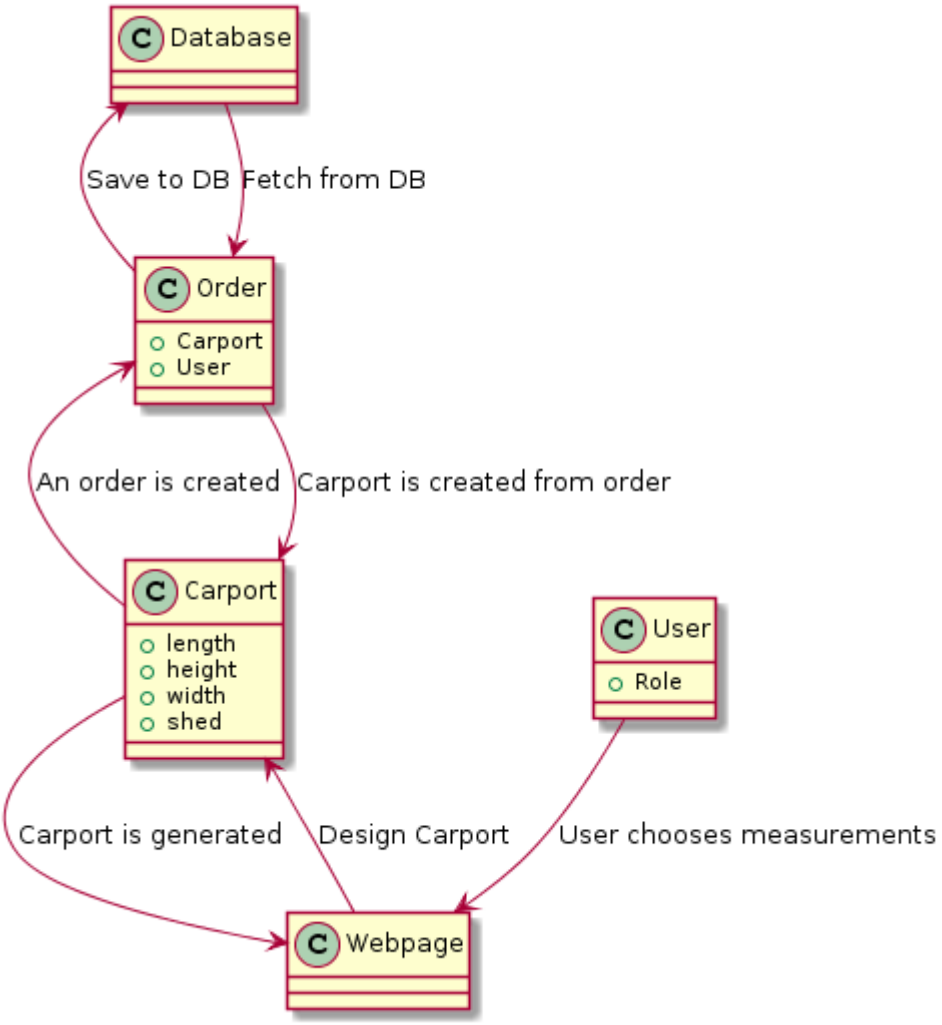
Efter godkendelse af PO begyndte sprint 4 ved at nedbryde hver userstory i følgende tasks:

ID	Name	Tasks
#7	Kontroller kundens valg af mål	#62 JSP admin side, #77 reviewpage til admin-kontrol, #80 Tilføj relevante kolonner til DB order, #74 addOrderToDB, #75 createOrder command, #63 getOrderFromDB
#16	Kontroller kundens valg af skur	#64 add to admin page
#11	Mulighed for ændring af samlet pris	#65 add to admin page
#61	Generer PDF	#76 add iText til pom.xml , #78 PDF generator #79 Style PDF
#49	Lagerbeholdning	#66 Tilføj kolonner til DB, #67 Udvid datamapper, #68 Ret DTO’er, #69 Udvid calculators

Domæne model og ER diagram

-Jonatan og Rasmus

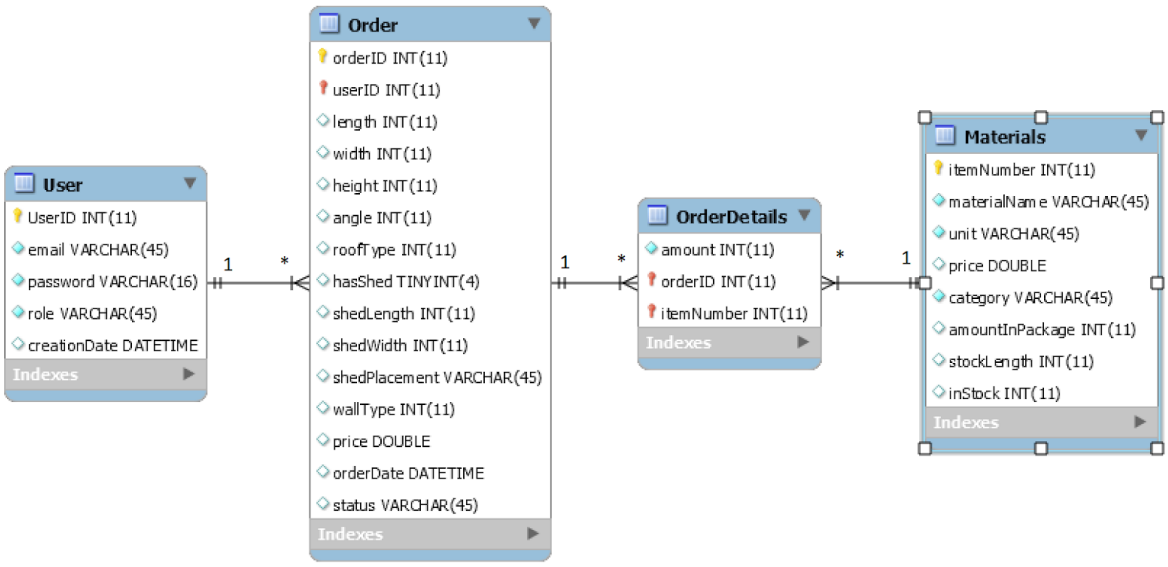
Domæne model



Domæne-modellen beskriver et konceptuelt overblik over produktet. En bruger har mulighed for at designe en carport via. hjemmesiden, hvorefter en carport genereres, og efterfølgende bliver lavet til en ordre. Den ordre bliver derefter gemt i databasen. Ligeledes kan en ordre hives ud af databasen og laves om til en carport, som kan visualiseres for brugeren.

ER diagram

I nedenstående ER diagram vises relationerne til de forskellige tabeller, der tilhører Fog databasen. NB: Dette er ikke Fogs nuværende database, men er en midlertidig database der er arbejdet med, da der ikke er givet adgang til Fogs egen database.



User

Der skal bruges en brugertabel til at kunne gemme data på systemets brugere, som eventuelt senere kunne bruges til markedsføring, kundestatistikker eller lignende.

Oprettelse af en ny bruger genererer automatisk (auto increment) et nyt 'UserID' til brugeren og dette er også primærnøglen for tabellen.

Order

userID i denne tabel er en fremmednøgle til tabellen 'User'. Den er her, da der skal kobles en forbindelse fra ordren til den kunde som har foretaget en bestilling.

Order Details

Denne tabel holder mængden af et given materiale der skal bruges til at sammensætte styklister. Der er skabt relationer mellem Order og Materials vha. fremmednøglerne 'orderID' og 'itemNumber'.

Refleksion

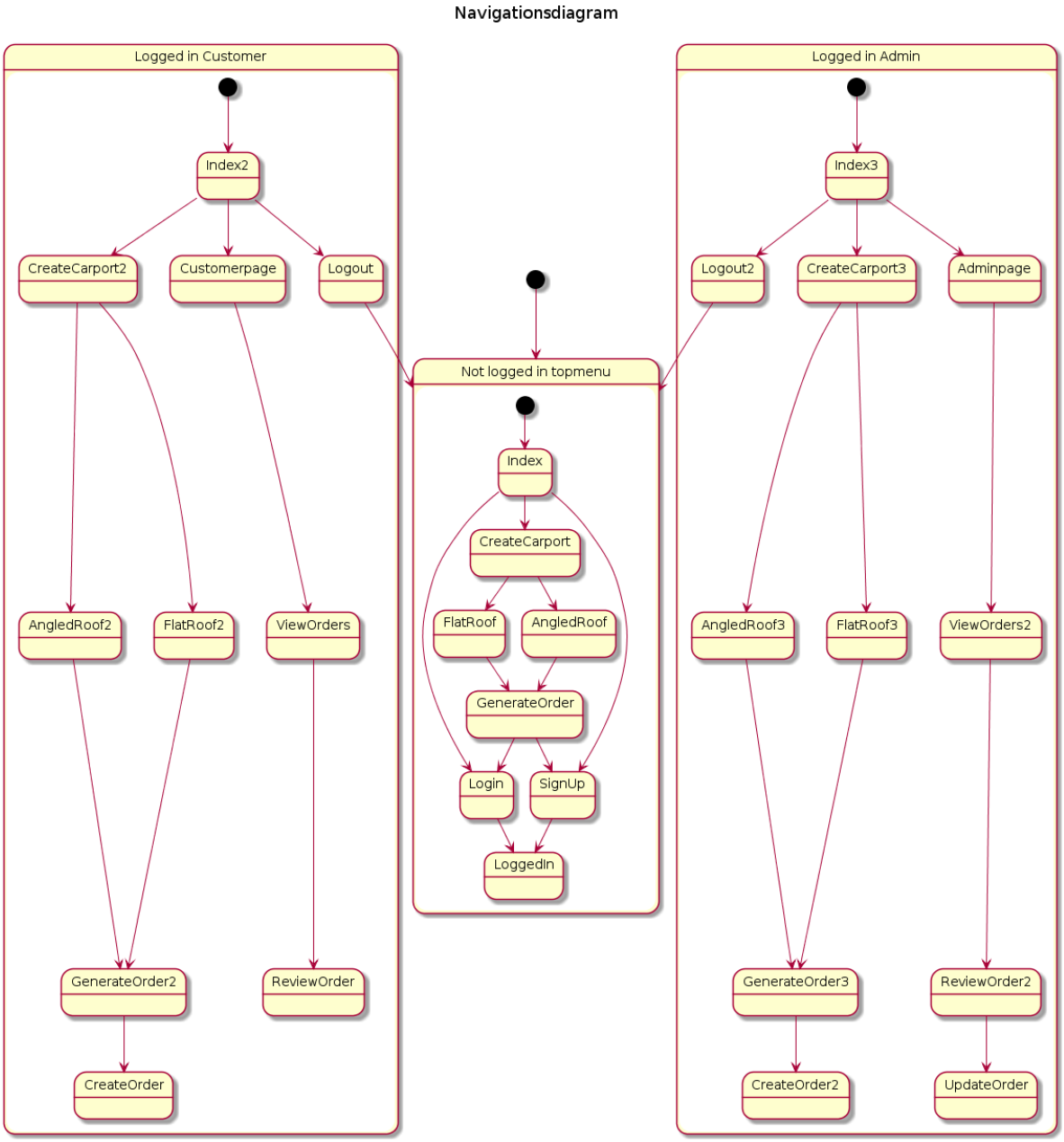
Navnet på ‘orderDetails’-tabellen ville være mere sigende, hvis den blev omdøbt til ‘materialDetails’, da det er tabellen indeholder detaljer om hvor mange enheder af et givent materiale der skal bruges.

I tabellen ‘order’ burde al data tilhørende carport flyttes over i en ‘carport’-tabel og have en ‘carport’-fremmednøgle i ‘order’-tabellen. Det samme gælder for data omhandlende skuret.

Andre overvejelser var at dele ‘Materials’ op flere tabeller, som indeholdte hver deres materialetype, som træ, metal osv.

Navigationsdiagram

-Jonatan



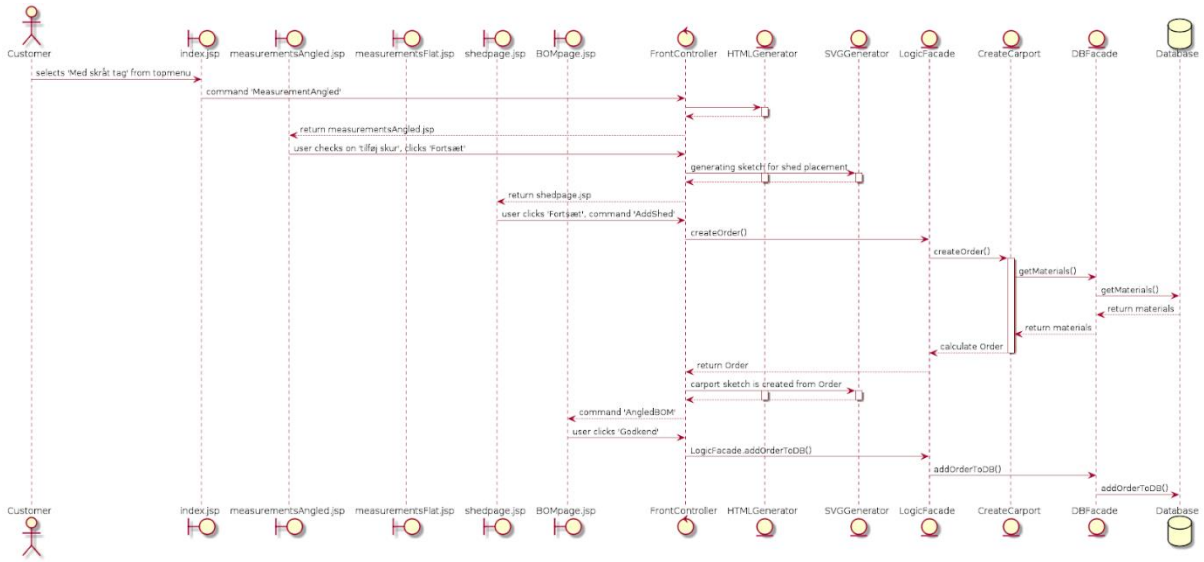
Hele siden er bygget op omkring en fælles topmenu. Her har man mulighed for at logge ind/registrere og designe en carport med fladt eller skråt tag. Er man ikke logget ind, kan man ikke foretage en ordre, men blot se en skitse, stykliste samt en overslagspris.

Hvis man er logget ind, enten som Admin eller Customer, kan man foretage sin ordre, og den vil derefter bliver gemt i databasen.

Admin og Customer har også adgang til hhv. Adminpage.jsp og Customerpage.jsp. Admin har der mulighed for at gennemse alle de ordrer der er gemt i databasen, og samtidig justére prisen og godkende/afvise ordren. Customer har kun mulighed for at gennemse sine egne ordrer.

Sekvens diagrammer

Use-case 1



I ovenstående sekvensdiagram, ser vi en Customer/bruger der der foretager en ordre på en carport med skråt tag og et skur.

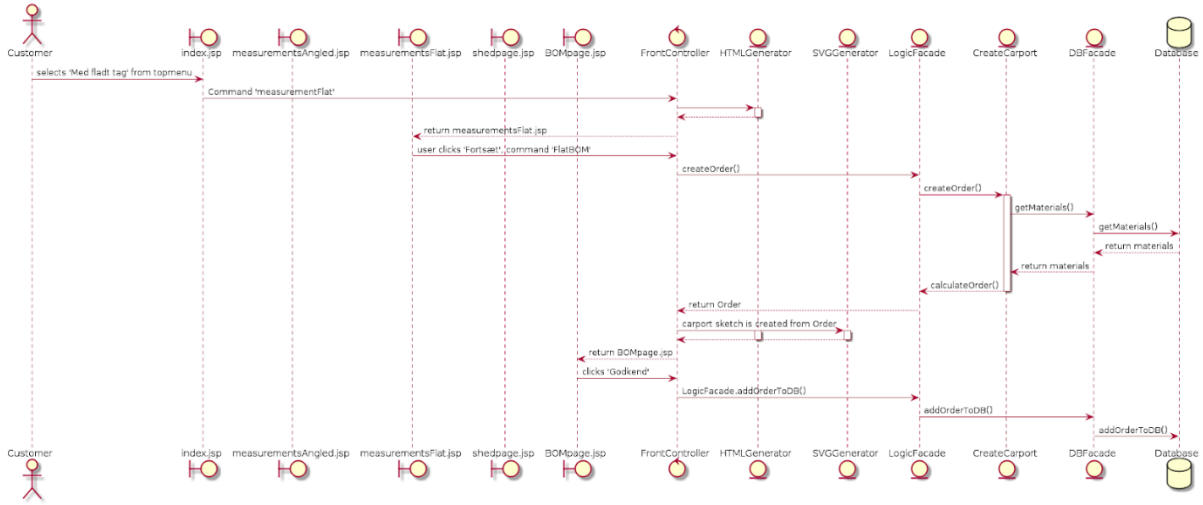
I første trin vælger brugeren ‘Med skråt tag’ fra Design Carport-dropdown i topmenuen. Brugeren bliver derefter ført til measurementsAngled.jsp, hvor der er mulighed for at vælge højde, længde, bredde på carporten og en vinkel på taget. Brugeren har også mulighed for at tilvælge et skur til carporten, via en checkbox. Brugeren checker ‘skuret’ af og trykker ‘Fortsæt’. Checkboxens værdi bliver hevet ud, som en parameter, både i FlatBOM og AngledBOM. Er checkboxen tom, vil dens værdi være null. Når der bliver trykket ‘Fortsæt’ bliver der lavet en if-sætning, der kontrollerer om parameteren er lig med null eller ej. Hvis parameter er null, bliver man sendt til BOMpage.jsp (Bill of Materials). Hvis ikke, bliver man sendt til shedpage.jsp. Her bliver brugeren

præsenteret for en lille skitse af carporten med de korrekte dimensioner, hvor der er mulighed for at vælge mål til skuret, samt dets placering.

Efter brugeren har valgt mål og placering af skur, bliver brugeren sendt videre til BOMpage.jsp. Her bliver får brugeren præsenteret en stykliste med overslagspris, samt en skitse fra siden og en skitse fra fugleperspektiv.

Når brugeren trykker 'Godkend', bliver ordren gemt i databasen.

Use-case 2



Ovenstående diagram, viser Customer/brugeren der vælger at designe en carport med fladt tag. Brugeren vælger derfor 'Med fladt tag' fra Design Carport-dropdown. Brugeren bliver sendt til measurementsFlat.jsp, hvor der er mulighed for at vælge mål til carporten.

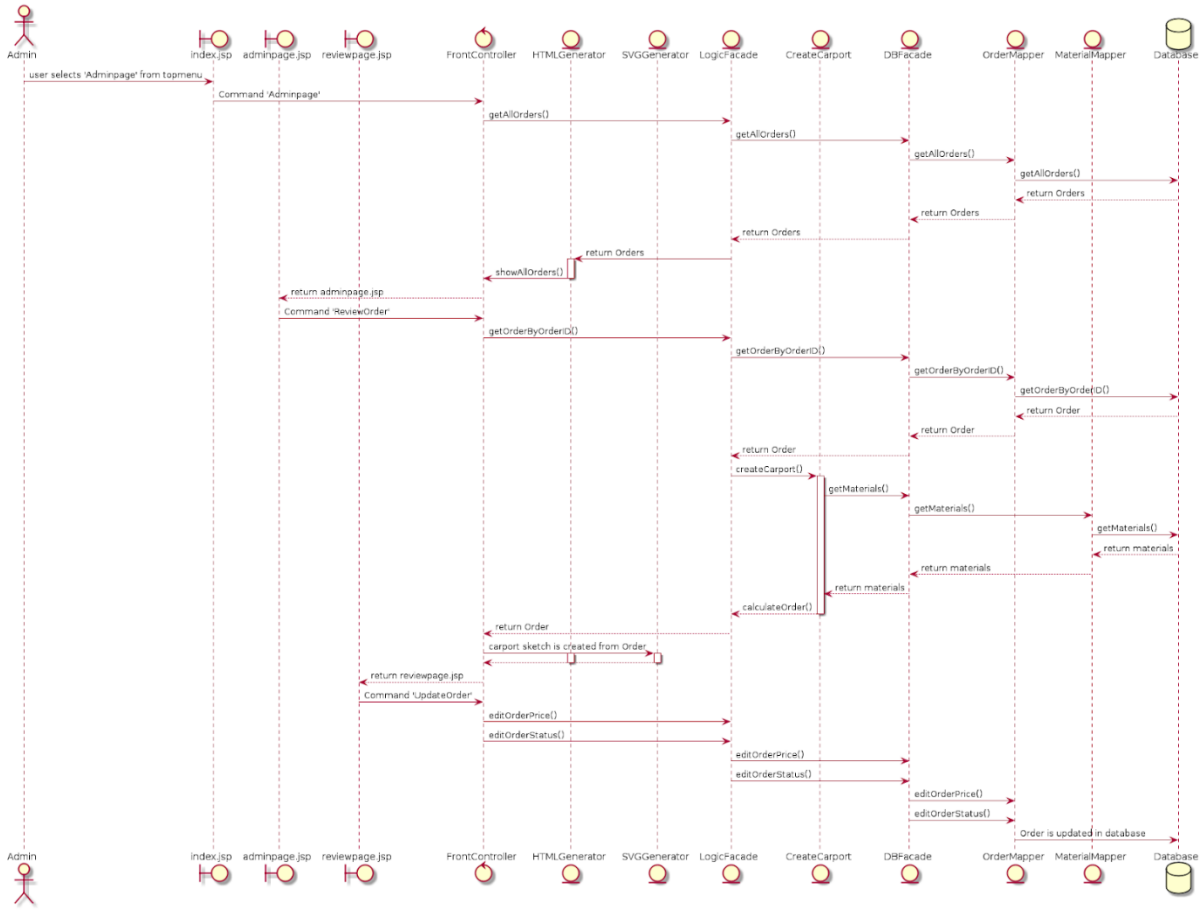
Når brugeren trykker 'Fortsæt' bliver BOMpage.jsp præsenteret med stykliste, overslagspris og skitser af carporten.

Derefter trykker brugeren 'Godkend' og ordren bliver gemt i databasen.

Use-case 3

Det tredje og sidste diagram viser use-casen: 'UpdateOrder' som er eksklusiv for administratoren. I første trin klikker Admin på 'adminpage', fra user-dropdown i topmenuen. Herefter bliver alle ordre, der findes i databasen, hevet ud og vist i en autogenereret tabel, med oversigt over de væsentlige informationer ved ordren. Hver række i tabellen har sin egen 'Review-knap'.

Admin kan klikke sig ind på hvilken som helst ordre, og få vist stykliste, overslagspris og skitse af ordren.



Når Admin klikker 'Review' til en ordre, bliver den hevet af databasen via sit 'OrderID'. Derefter bliver den sendt til CreateCarport-klassen, hvor de nødvendige materialer hives ud af databasen, og bliver gemt i ordren. Ordren bliver derefter returneret til FrontControlleren, som via HTML- og SVGGeneratoren, generer hhv. stykliste og skitser. Til slut bliver ordren vist på reviewpage.jsp.

Udover at få vist den eksakte ordre fra databasen på siden, har Admin også mulighed for at ændre status og pris på ordren. Admin kan vælge mellem 'Approved', 'Pending' og 'Denied' under status, og prisen kan justeres ned til 15 % under den originale pris, eller over uden begrænsinger.

Når Admin er tilfreds med eventuelle ændringer i ordren, trykkes der 'UpdateOrder', og ordren bliver derefter opdateret i databasen.

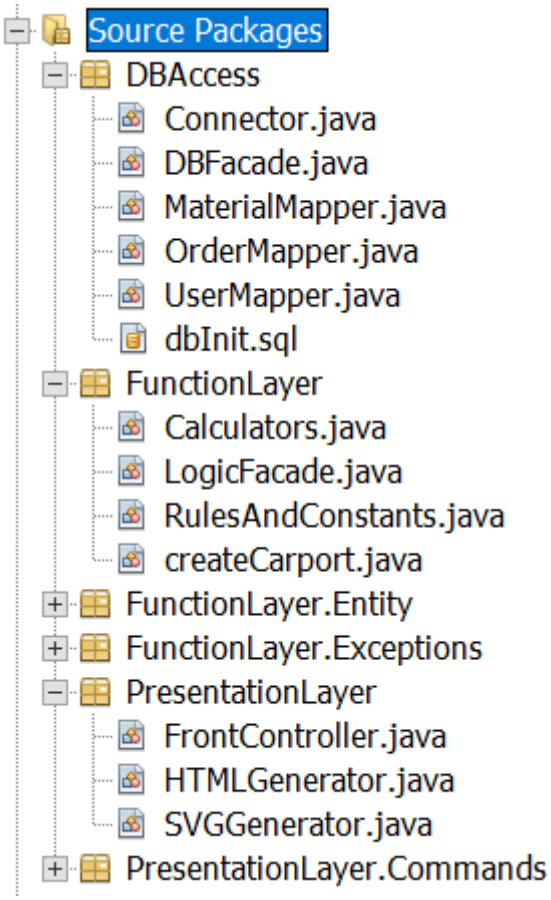
Særlige forhold

-Mads

Design mønstre

3-lags model:

Strukturen i vores projekt følger 3-lags modellen hvor ideen er at opdele brugergrænseflade, forretningslogik og database i hver sin del. Hvis dette gøres korrekt vil det opleves at koden opnår højere sammenhængskraft og lavere kobling. Når 3-lags modellen er benyttet korrekt skulle det således være muligt at udskifte en af de to ydre lag og stadig have et fungerende program. F.eks skift fra web præsentationslag til en app brugergrænseflade eller java UI. Ligeledes skulle database laget kunne skiftes til at benytte en anden databaseform. I dette projekt fremgår de 3 lag som PresentationLayer, FunctionLayer og DBAccess.



Facade:

Facade er et designmønster der benyttes i projektet til at reducere afhængigheder af andre klasser i projektet. Facaderne tager sig af alt instantiering der er nødvendig og holder dermed koblingen mellem lagene lav.

Da alle metoder i datamappers er package private tvinges funktionslaget til at benytte DBFacade til alle kald til databasen. Ligeledes giver LogicFacade en lavere kobling mellem præsentations laget og logikken og fungerer tilnærmelsesvis som et interface fra præsentationslagets perspektiv, det eneste præsentations laget har behov for kendskab til er LogicFacaden og entities.

Command og Front Controller:

Der benyttes en front controller der bearbejder alle requests og kører en command baseret på en parameter på det givne request. I projektet er består Command mønsteret af en klasse, Command, med en abstract metode execute() samt et map af mulige commands, der alle arver fra Command. Alle commands skal dermed implementere en execute() metode, og det er her forskellen på commands'ne skal findes. Det særlig smarte ved command designmønsteret er at den der kalder execute() metoden (executor) ikke behøver noget kendskab til indholdet eller nogen eventuelle afhængigheder. Dette gør kombinationen af command og front control mønster særligt smart, da frontcontrolleren slipper for alt ansvar af HVAD der sker når en bruger f.eks. trykker på en knap, dens eneste ansvars område er blot at viderestille requested til den rette kommando (hvilket er præcis den rolle en frontcontroller ønskes at have). Comand er særlig smart når der er et stykke kode der skal ske ved en særlig "trigger", for eksempel ved tryk på en knap som er det primære brugsscenarie i dette projekt.

Singleton:

Konceptet er at sørge for at en klasse max kan have netop én instantiering. Dette gøres ved at lægge ansvaret for instantieringen ind på klassen selv. Dette vil typisk fungere som et private felt i klassen som er null hvis den ikke er instantieret. Første gang klassen kaldes bliver dette felt så instantieret og alle fremtidige kald vil derfra returnere det oprette objekt i stedet for at oprette et nyt. Hvis det ønskes at udelukke objektet fra at bliver instantieret på nogle andre måder kan klassens konstruktør gøres private. I dette projekt bliver Singleton brugt i Connector til DB. Dette er utrolig vigtigt da databasen ikke kan håndtere for mange forbindelser af gangen, og på denne måde garanteres det at det maksimalt er én forbindelse til databasen fra dette software. Det er også benyttet i Command mønstret, hvor det sørger for at der maks er ét map af commands.

Exceptions og logging

I projektet har vi valgt at benytte et ekstra led i det normale arvehiereaki. Alle exceptions er throwable og derefter har vi en BaseException som arver fra

Exception. De to exceptions vi rent faktisk kaster op igennem lagene er LogicException og DBException. I den nuværende version af programmet gør BaseException ingen ting. Ideen med BaseException var at lade excepionen selv være ansvarlig for at logge, dvs, at alt logging logik skulle være i BaseException og logning dermed kun bliver gjort én gang, der hvor exceptionen bliver instantieret.

Session

På session gemmer vi:

- en user når brugeren er logget ind.
- den nuværende ordre som brugeren er ved at oprette.

Sikkerhed

For at logge ind kræves et matchende brugernavn og password som findes i databasen. Passworded er dog ikke hash'et, hvilket selvfølgelig giver en stor sikkerhedsbrist. Ved en reel implementation af sådan et projekt er det et must at have større sikkerhed ved brugerinformationer.

Udvalgte kodeeksempler

-Jonatan og Mads

Præsentationslaget

Eksempel 1

```java
    public static String generateBOM(Order order) {
        StringBuilder sb = new StringBuilder();
        HashMap<String, WoodDetails> materialsWood = order.getCarportWoodMaterials();
        sb.append("<table id=\"BillOfMaterials\">\n");
        sb.append("            <thead>\n");
        sb.append("            <tr>\n");
        sb.append("                <th>Vare</th>\n");
        sb.append("                <th>Varenummer</th>\n");
        sb.append("                <th>Beskrivelse</th>\n");
        sb.append("                <th>Længde i cm</th>\n");
        sb.append("                <th>Enhed</th>\n");
        sb.append("                <th>Pris pr. enhed</th>\n");
        sb.append("                <th>Antal</th>\n");
        sb.append("                <th>Lagerbeholdning</th>\n");
        sb.append("                <th>Samlet pris</th>\n");
        sb.append("            </tr>\n");
        sb.append("            </thead>\n");

        sb.append(getWoodMaterialsList(materialsWood));
        HashMap<String, MetalDetails> materialsMetal = order.getCarportMetalMaterials();

        sb.append(getMetalMaterialsList(materialsMetal));
        if (order.isShedExists()) {
            HashMap<String, WoodDetails> shedMaterialsWood = order.getShedWoodMaterials();

            sb.append(getWoodMaterialsList(shedMaterialsWood));
            HashMap<String, MetalDetails> shedMaterialsMetal = order.getShedMetalMaterials();

            sb.append(getMetalMaterialsList(shedMaterialsMetal));
        }

        sb.append("<tr><td>Ialt</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>").append(order.getTotalOrderPrice()).append(" kr </td></tr>");
        sb.append("</table>");
        return sb.toString();
    }
```

I ovenstående kodeeksempel, vises en af de mange metoder, som HTMLGeneratoren holder. I dette eksempel ser vi hvordan styklisten bliver genereret til en tabel, ud fra en ordre.

Bemærk at der bruges StringBuilder til at konstruere HTML-koden. Koden kunne også konstrueres ved koble flere string-objekter sammen, men eftersom styklisterne kan blive forholdsvis lange, vil det koste en del performance; Bruger man string-concatenation i et for-loop, vil java automatisk oprette et nyt StringBuilder-objekt per iteration.

Derfor kan man med fordel oprette sit eget StringBuilder-objekt inden man påbegynder sit for-loop, for at spare ram.

Eksempel 2

```
256     private static String getMetalMaterialsList(HashMap<String, MetalDetails> materialsMetal) {
257         StringBuilder sb = new StringBuilder();
            for (Map.Entry<String, MetalDetails> mapShedMetal : materialsMetal.entrySet()) {
259             sb.append("<tr>");
260             sb.append("<td>").append(mapShedMetal.getKey()).append("</td>");
261             sb.append("<td>").append(mapShedMetal.getValue().getMaterial().getItemNumber()).append("</td>");
262             sb.append("<td>").append(mapShedMetal.getValue().getMaterial().getName()).append("</td>");
263             sb.append("<td></td>");
264             sb.append("<td>").append(mapShedMetal.getValue().getMaterial().getUnit()).append("</td>");
265             sb.append("<td>").append(mapShedMetal.getValue().getMaterial().getPricePerUnit()).append("  kr </td>");
266             sb.append("<td>").append(mapShedMetal.getValue().getAmount()).append("</td>");
267             sb.append("<td>").append(mapShedMetal.getValue().getMaterial().getAmountInStock()).append("</td>");
268
269             sb.append("<td>").append(mapShedMetal.getValue().getTotalItemPrice()).append("  kr </td>");
270             sb.append("</tr>");
271
272         }
273         return sb.toString();
274     }
275     private static String getWoodMaterialsList(HashMap<String, WoodDetails> materialsWood) {
276         StringBuilder sb = new StringBuilder();
            for (Map.Entry<String, WoodDetails> mapWood : materialsWood.entrySet()) {
278             sb.append("<tr>");
279             sb.append("<td>").append(mapWood.getKey()).append("</td>");
280             sb.append("<td>").append(mapWood.getValue().getMaterial().getItemNumber()).append("</td>");
281             sb.append("<td>").append(mapWood.getValue().getMaterial().getName()).append("</td>");
282             sb.append("<td>").append(mapWood.getValue().getCmLengthEach()).append("</td>");
283             sb.append("<td>").append(mapWood.getValue().getMaterial().getUnit()).append("</td>");
284             sb.append("<td>").append(mapWood.getValue().getMaterial().getPricePerUnit()).append("  kr </td>");
285             sb.append("<td>").append(mapWood.getValue().getAmount()).append("</td>");
286             sb.append("<td>").append(mapWood.getValue().getMaterial().getAmountInStock()).append("</td>");
287             sb.append("<td>").append(mapWood.getValue().getTotalItemPrice()).append("  kr </td>");
288             sb.append("</tr>");
289
290         }
291         return sb.toString();
292     }
```

Eksempel 2 er ét af de to metodekald, som kaldes i ovenstående eksempel (linje 243 og 248). Igen opretter vi StringBuilder-objektet inden vi påbegynder for-loopet, hvor der til sidst i metoden, returneres toString.

Eksempel 3

```
        <div class ="pageContent">
57          <div class="tableContainer">
58              <%=request.getAttribute("table")%>
59              <form action="FrontController" method="POST">
60                  <input type="submit" value="Create Order">
61                  <input type="hidden" name="command" value="createOrder">
62              </form>
63
64
65          </div>
```

I 3. eksempel viser vi hvordan vi har opbygget JSP-siderne. I stedet for at holde StringBuilder i JSP'erne, bliver alt HTML-kode oprettet i HTMLGeneratoren, hvorefter det bliver gemt, som attributter på requestet. Fordelen ved dette, er at JSP-siderne bliver mere overskuelige at læse og skrive.

Funktionslaget

De følgende kodeeksempler viser den overordnede struktur i controlleren CreateCarport.

Eksempel 1

```
21      public static void createCarport(Order order) throws LoginException {
22          if (order.getAngle() == 0) {
23              createFlatRoofRafters(order);
24          } else {
25              createAngledRoofRafters(order);
26          }
27
28          addAllWoodMaterialsToList(order);
29          addAllMetalMaterialsToList(order);
30      }
```

I eksempel 1 ses det at metoden modtager en order som argument. Denne indeholder alle bruger indtastede mål såsom længde, højde, bredde og tagvinkel. Tagvinklen checkes, hvis den er 0 er der tale om et fladt ellers er det med hældning. Herefter kaldes metoder til at tilføje de resterende materialer til ordren. Selvom metoden ikke har en returværdi, bliver order opdateret grundet java's opbygning af med "call by reference" i forhold til objekter generelt.

Eksempel 2

```
56      private static void addAllWoodMaterialsToList(Order order) throws LoginException {
57          //Calculate lengths
58          double cmLengthEach = Calculators.postsLengthCalc(order.getHeight());
59          double remLength = Calculators.remLengthCalc(order.getLength());
60          //Get materials from DB
61          WoodMaterial rem = LogicFacade.getWoodMaterial(RulesAndConstants.PREFERRED_MATERIAL_REM);
62          WoodMaterial post = LogicFacade.getWoodMaterial(RulesAndConstants.PREFERRED_MATERIAL_POSTS);
63          //Add to materialsList
64          putIntoList(order.getCarportWoodMaterials(), new WoodDetails(post, order.getPostsAmount(), cmLengthEach, RulesAndConstants.CARPORT_POSTS_DESCRIPTION));
65          putIntoList(order.getCarportWoodMaterials(), new WoodDetails(rem, 2, remLength, RulesAndConstants.CARPORT_REM_DESCRIPTION));
66
67      }
```

```
100     private static void putIntoList(HashMap map, MaterialDetails mat) {
101         map.put(mat.getDescription(), mat);
102     }
```

Eksempel 2 viser indholdet af metoderne der bliver kaldt i createCarport();. Længder udregnes via calculators klassen, materialer hentes fra DB, og slutteligt bliver de tilføjet til den relevante liste. En order indeholder et map både til metalMaterialDetails og til woodMaterialDetails.

Databaselaget

Eksempel 1

```
12  public class Connector {
13
14      private static final String URL = "jdbc:mysql://104.248.17.255:3306/FogDB";
15      private static final String USERNAME = "transformer";
16      private static final String PASSWORD = "transformerpass";
17
18      private static Connection singleton;
19
20      public static void setConnection( Connection con ) {
21          singleton = con;
22      }
23
24      public static Connection connection() throws ClassNotFoundException, SQLException {
25          if ( singleton == null ) {
26              Class.forName( "com.mysql.cj.jdbc.Driver" );
27              singleton = DriverManager.getConnection( URL, USERNAME, PASSWORD );
28              System.out.println("abekat");
29          }
30          return singleton;
31      }
32
33  }
34
```

I eksempel 1 ses den benyttede connector til databasen. Som tidligere beskrevet bruges et singleton-pattern til at håndtere selve forbindelsen, således at der kun er én aktiv forbindelse.

Eksempel 2

```
229     static void editOrderPrice(Order order, double newPrice) throws LoginException {
230         try {
231             Connection con = Connector.connection();
232             con.setAutoCommit(false);
233             String SQL = "UPDATE `FogDB`.`Order` SET `price` = ? WHERE (`orderID` = ?) and (`userID` = ?);";
234             PreparedStatement ps = con.prepareStatement(SQL);
235             ps.setDouble(1, newPrice);
236             ps.setInt(2, order.getOrderID());
237             ps.setInt(3, order.getUserID());
238             ps.executeUpdate();
239             con.commit();
240             con.setAutoCommit(true);
241         } catch (ClassNotFoundException | SQLException EX) {
242             throw new LoginException(EX.getMessage());
243
244         }
245
246     }
```

I eksempel 2 ses at metoden er package private, og dermed er den eneste måde at kalde den fra andre lag via DBFacaden. Da denne metodes ansvar er at opdatere en kolonne i databasen vælges det at sætte autoCommit til false, da vi kun vil have ændringer i databasen hvis alle eventuelle ændringer sker uden fejl. Der benyttes preparedstatement da det for det første er nemmere når SQL-query er afhængig af variable input, og særligt for at undgå SQL-injections.

Status på implementation

-Jonatan

Mangel på PDF

I sprint 4 var det en user story at skrive stykliste og skitse til PDF. Der blev igangsat eksperimentation med iText-PDF, men eftersom der var vigtigere ting i koden der skulle fixes, måtte dette nedprioriteres. Resultatet endte med en tom pdf-fil, og kort tid inden PO-møde, blev det besluttet at opgive den userstory.

Brug af flere Connectors

Der var i starten af forløbet, gjort nogle tanker om brug af flere Connectors. Ideen var at, var man logget ind som Admin, ville man bruge en connector med alle privilegier i SQL-databasen. Var man logget ind som Customer, ville man blot have de nødvendige privilegier til at foretage en ordre og registrere sig som bruger.

Styling

Layoutet på websiden, har "drillet" lidt, når man skifter skærmopløsning. Især på shedpage.jsp, har der været tydelige mangler i CSS. Det var tanken at radio-knapperne til skurets placering, skulle være knyttet til hjørnerne af carportens skitse, men efter at have testet layoutet på kollegaernes pc'er, samt andre browsers, måtte det accepteres at det ikke var perfekt.

Flere detaljer til carporten

Det var begrænset, hvor mange detaljer der blev valgt at inkludere til at bygge carporten. Der er valgt at nøjes med træ, træbeklædning, tagbeklædning, beslag og skruer. Erfaringer fra en tømrer ville have gavnet projektudvilkingen, for at kunne producere et realistisk bud på en carport.

Logning

Logger-klasse blev ikke ikke implementeret til skrivning af logfiler. Det vil være obligatorisk, når programmet er deployet på en server, for at gøre fejlfinding mulig.

Test

-Jonatan og Rasmus

Manuel - Korrekt brugerregistrering

Benyt topmenuen til at navigere til registerings-siden, ved at klikke ‘Register’ oppe i højre hjørne. Her skal der udfyldes email, password og bekræftelse af password. Indtast “test123” i email og “asd” i password og confirm password.

Tryk på ‘Submit’. Hvis felterne udfyldes korrekt, bliver man returneret til index.jsp, og topmenuen viser nu hvem man er logget ind som, oppe i højre hjørne.

Manuel - Forkert brugerregistrering

Benyt topmenuen til at navigere til registrerings-siden, ved at klikke ‘Register’ oppe i højre hjørne.

Indtast “test123” og “asd” i password. I confirm password indtastes “dsa”. Tryk derefter ‘Submit’. Brugeren vil bliver sendt til error-siden med fejlbeskeden: The two passwords did not match.

Manuel - Design Carport med fladt tag og skur

Hold cursoren over Design Carport i topmenuen, for at få vist dropdown-menuen. Klik derefter på ‘Med fladt tag’, og Siden ‘Fladt tag’ vil blive vist. Her er der mulighed for at bestemme højde, bredde og længde af carporten, ud fra select-elementerne i html-koden. Lad målene være, som de er, men check derimod ‘Tilføj skur’. Klik ‘Fortsæt’ og ‘Tilvalg af Skur’ vil blive vist. Her kan man vælge længde og bredde af skuret; samme koncept som forrige jsp-side. Højden vil altid være den samme, som man har valgt til carporten. Bemærk at længde og bredde på skuret ikke kan overstige carportens længde og bredde. Som standard vil skurets længde og bredde være 1x1 meter.

Under select-elementerne er der 4 radio-knapper, placeret over carportens areal. Disse knapper, bestemmer hvilket hjørne af carporten, som skuret skal stå op ad. Som standard vil øvre højre hjørne altid være valgt. Lad skuret holde standard-værdierne og klik ‘Fortsæt’.

Herefter bliver man ført til Styklisten, hvor man får præsenteret stykliste, skitse og en overslagspris på 12555 kr. Klikker man 'Godkend', bliver man returneret til Error-siden, med fejlbeskeden: "not logged in".

Manuel - Design Carport med skråt tag som bruger

Klik 'Register' fra topmenuen, og registrér en ny bruger. En succesfuld registrering, returnerer brugeren til Forsiden. Man vil kunne se på topmenuen, i højre side, hvem man er logget ind som.

Klik derefter på 'Med skråt tag' fra Design Carport-dropdown. Her vil Skråt tag-siden blive præsenteret. Siden ligner Fladt tag, dog kan man vælge andre tagtyper, samt vinkel på taget.

Lad carporten mål, være som de er, men sæt vinkel til 20 grader i stedet for 10. Klik 'Fortsæt'. Da 'Tilføj skur' ikke blev hakket af, bliver man sendt direkte til Stykliste-siden. Her skulle man meget gerne ende med en overslagspris på 12180 kr. Klik derefter 'Godkend' og ordren vil blive gemt i databasen, og man vil samtidig blive sendt til Forsiden.

Brugeren vil have mulighed for at se sin ordre, ved at holde musen over 'Logged in as ...' i topmenuen og klikke 'Customerpage'. Klikker brugeren 'Review', vil man få vist stykliste, overslagspris og skitse.

Automatiserede manuelle tests

I tabellen står de klasser og metoder som der er lavet automatiserede manuelle tests på. De køres inden og efter der merges med 'master'-branch. Dette mindsker risikoen for oversete fejl i koden og sikrer at den nye kode ikke har ødelagt den eksisterende kode.

NOTE: JaCoCo Code Coverage plugin bruges til at bestemme dækningsgraden af koden, men af uforklarlige årsager vises der en fejlagtig dækning, da graden på calculators er 0%.

Klasser testet	Metoder testet	Dækningsgrad
Connector	• setConnection	50%
UserMapper	• login • createUser	50%
Calculators	• postAmountCalc • postsLengthCalc • flatRoofRafterAmountCalc • rafterBottomLengthCalc • angledRoofRafterBottomAmountCalc • angledRoofRaftersSidesAmountCalc • angledRoofRafterSidesLengthCalc • concreteAmountCalc • mountPerPost • mountPerRafter • remLengthCalc	0%

Process

-Mads

Arbejdsprocessen faktuelt

Projektet blev opdelt i fire sprints (se appendix for fuld beskrivelse af indhold i disse) der hver havde en scrum master. Det følgende vil have små detaljer om hvert sprint og derudover bliver der gået særligt i dybden med ét sprint (sprint 4). Vi benyttede taiga.io til håndtereing af backlog, sprints og tasks.

Sprint 1

Til PO-møde inden sprint start blev det tydeligt at de forberedte userstories og how-to-demo's ikke var præcist nok formuleret. Scrum master valgte derfor at første standup møde i sprint 1 havde extra længde, således at vi først kunne rette userstories til, og derefter sætte gang i udviklingen. Der blev ikke holdt standup møde hver dag i sprint 1. Alle userstories blev løst og var klar til demo til det følgende PO-møde på trods af dårlig vurdering af estimated time til hver userstory. Userstories blev ikke opdelt i tasks. Restrospective blev ikke holdt.

Sprint 2

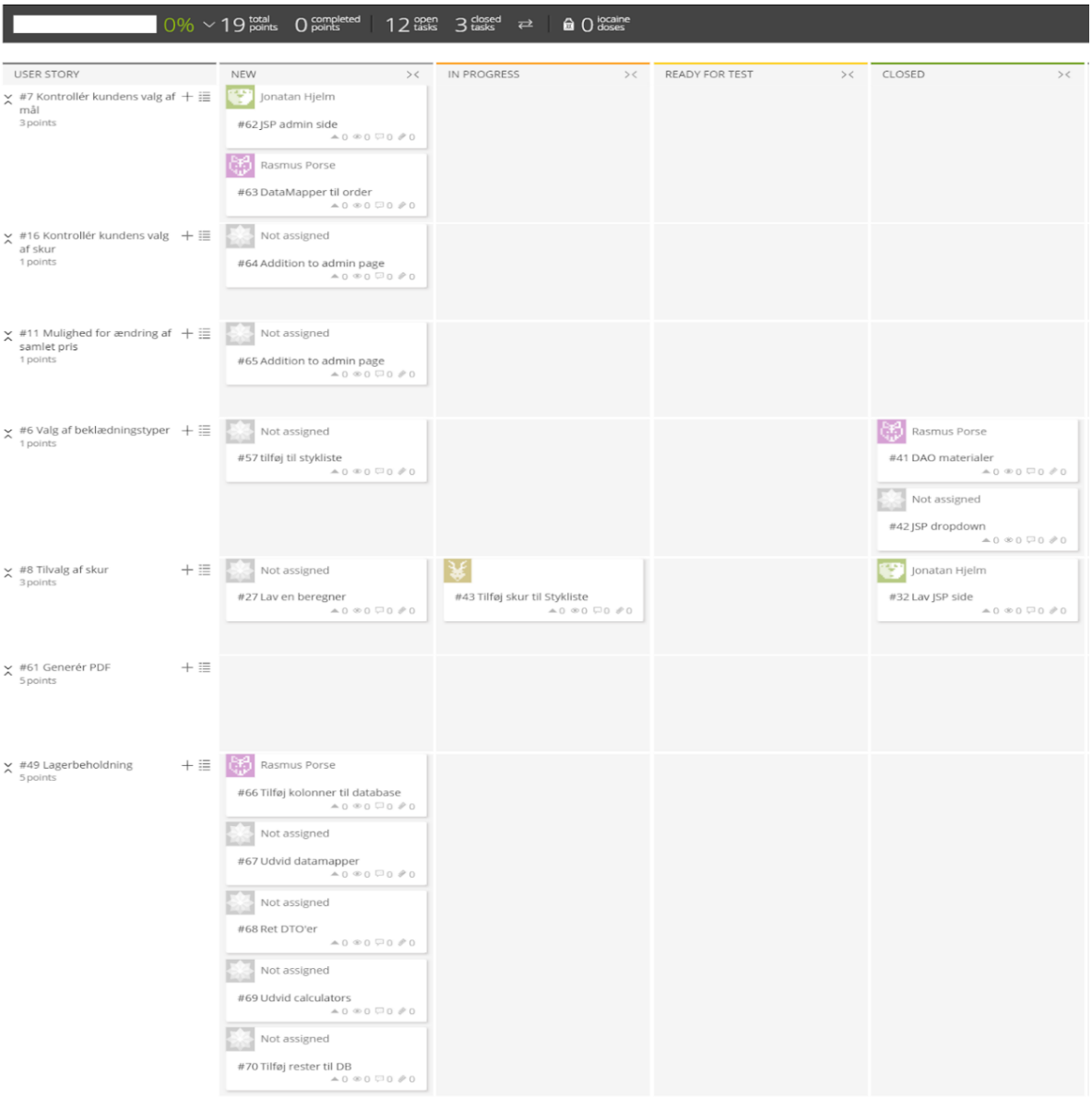
Sprint 2 fokus var på yderligere tilføjelser til stykliste beregner samt SVG tegninger. SCRUM-master afholdte første standup møde hvor der blev oprettet tasks til hver userstory. Der blev holdt to standup møder mere, et før vejleder møde og et før PO-møde, for at følge op. Restrospective blev holdt efter PO-møde.

Sprint 3

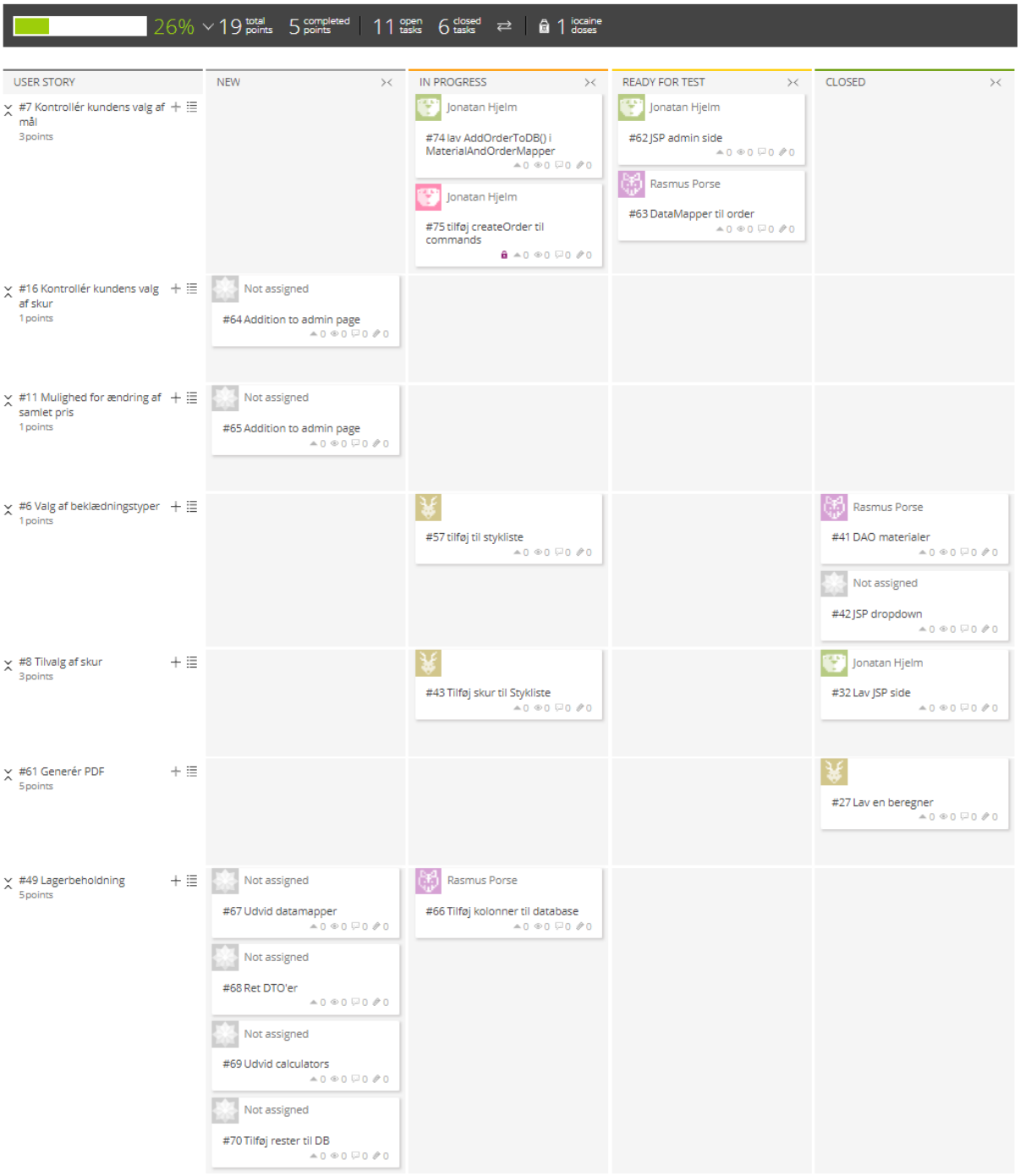
Gruppen var ramt af sygdom, daglige stand up møder blev klaret over nettet, ved hurtig opfølgning på tasks og fordeling af nye. Vejleder og PO-møde blev klaret uden fuld gruppe. Retrospective blev holdt i starten af sprint 4, grundet sygdom.

Sprint 4

Sprint 4 startede med standupmøde 1 hvor tasks blev oprettet til hver userstory, samt overførsel af ufærdige userstories fra sprint 3. Standupmøde 1 resulterede i følgende taskboard på taiga.io:

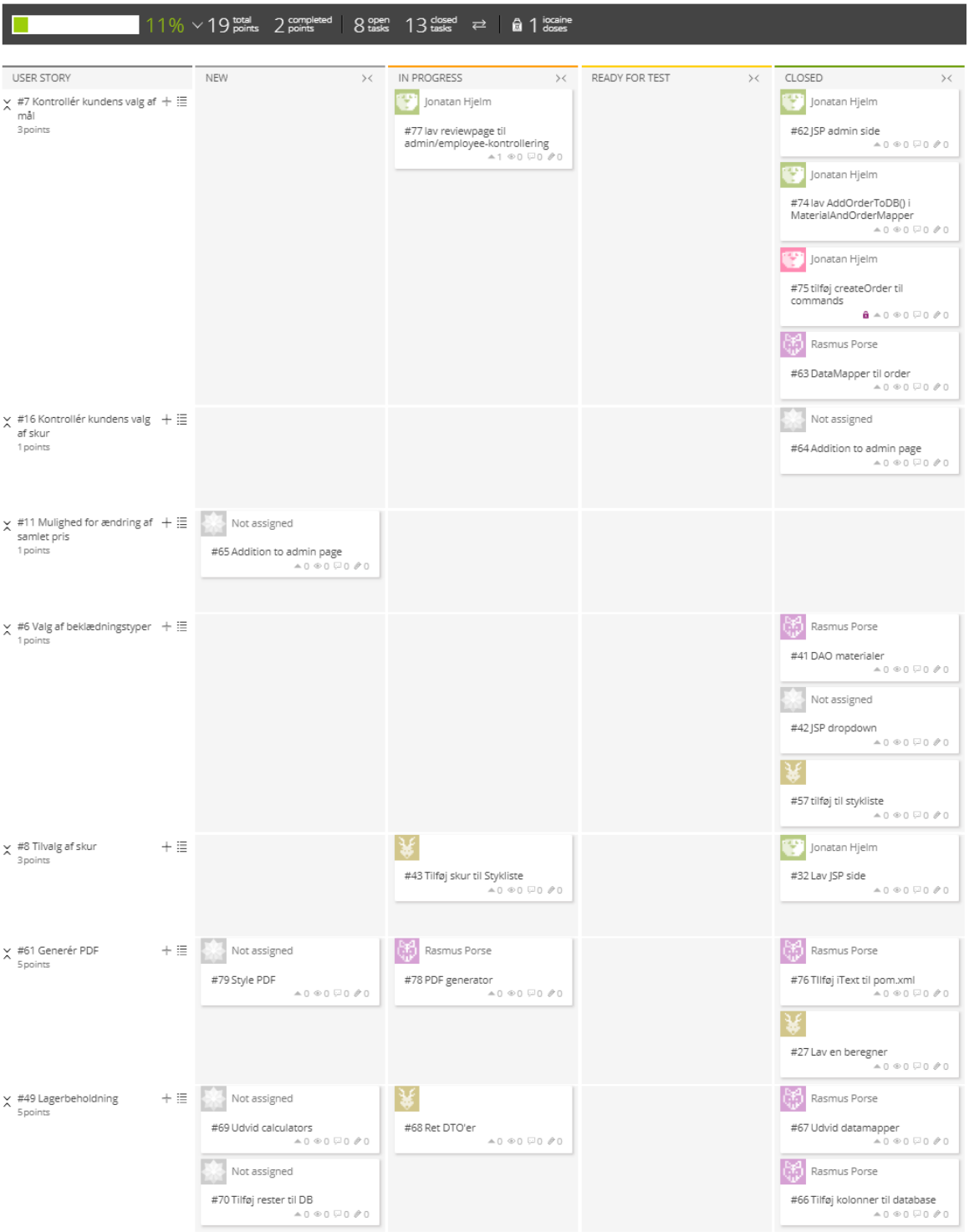


De tasks der allerede er gennemført, er fra userstories overført fra forrige sprint. Til standupmøde 2 blev der oprettet flere tasks i userstory #47 da enkelte tasks ikke var præcise nok. Det ses at alle nu er i gang med arbejdet:



Det ses i Userstory #7, task#75 at der er en lille lilla markør, dette betyder at udvikleren har behov for hjælp. Task'en blev diskuteret til standup møde hvor en løsning blev fundet.

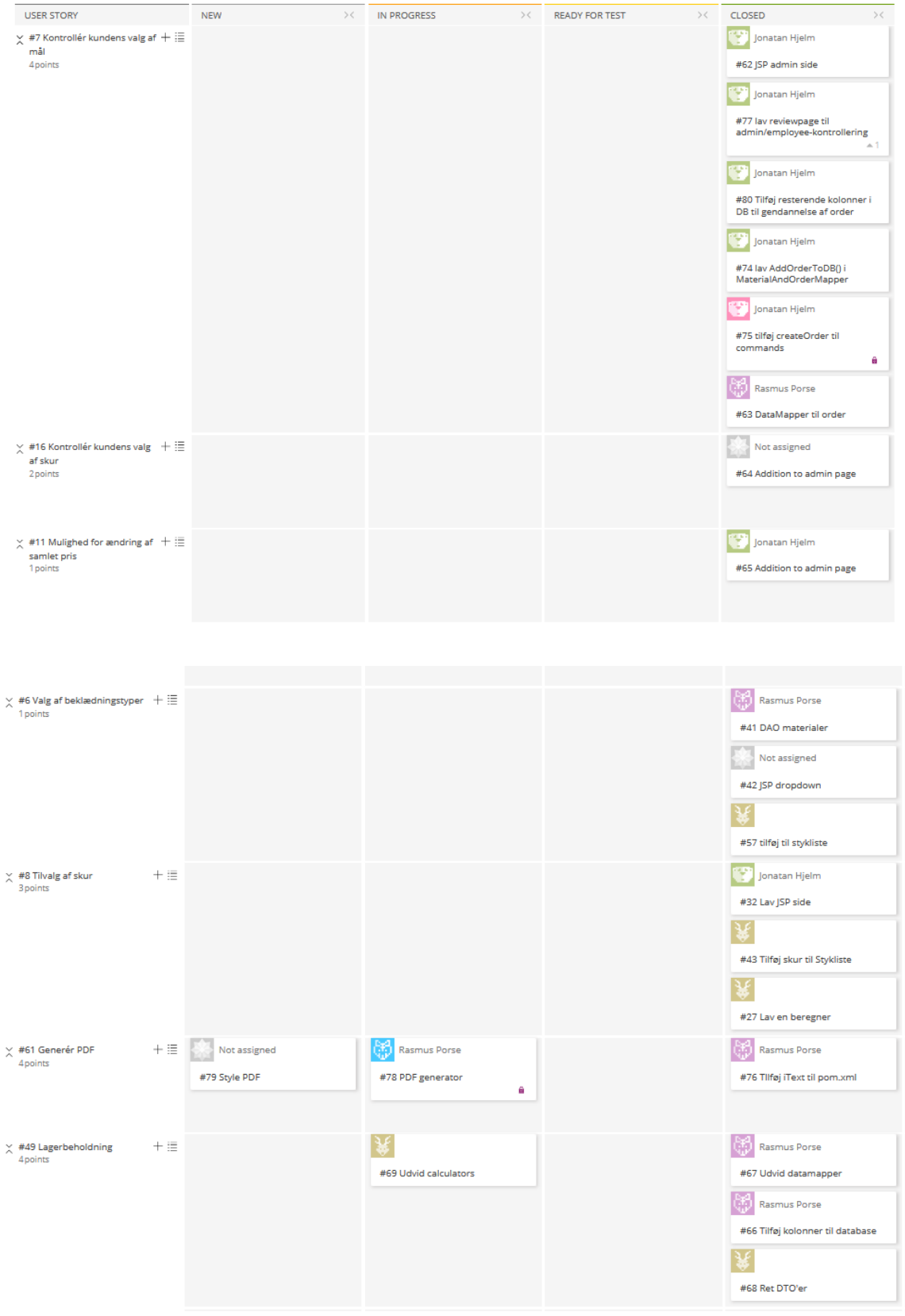
Standupmøde 3 resulterede i det følgende taskboard:



Sprintet skrider fremad og flere tasks er allerede løst. Alle team-medlemmer er tildelt minimum en task og det ses at task # 75 er blevet løst.

Standupmøde 4 resulterede ikke i et screenshot da de samme tasks stadig blev bearbejdet.

Standupmøde 5 før PO-møde viser at ikke alle task's blev færdige:



Retrospective blev afholdt efter PO-møde, som opfølgning på sprintet samt projektet i sin helhed.

Arbejdsprocessen reflekteret

Sprint 1

Der var en dårlig start på sprintet da vores Userstories ikke var præcise nok. Manglende erfaring og overblik over projektet gjorde vores tids estimeringer ret skæve, da vi ikke havde taget højde for meget af det ekstra arbejde der ligger i den første del af projektet i forhold til at opsætte struktur, oprette databasen, udfylde "dummy-data" osv. Vi benyttede os ikke af daglige standup-møder udover den første dag og userstories blev ikke splittet i tasks. Det lykkedes at blive færdig med det første sprint, men vi manglede generelt overblik og struktur i koden.

Sprint 2

Sprintet blev startet med standup møde hvor vi for første gang opdelte userstories i tasks. Dette gjorde hver userstory mere overskuelig, og særlig den estimerede arbejdstid var lettere at præcisere. Funktionsmæssigt blev sprintet ikke helt færdigt, da det ikke blev nået at lave sideview af carporten på SVG tegning. Til restrospective blev det diskuteret at omend opdeling i tasks havde været en hjælp, så kunne det gøres bedre, i flere tasks og flere detaljer. Det blev vedtaget at sørge for at holde daglige standup-møder, for at holde overblik over fremgangen i sprintet.

Sprint 3

Sprint 3 var teamet hårdt ramt af sygdom og dette resulterede i ufærdiggjort arbejde, og et kun halvfærdigt sprint. Det kunne dog konkluderes at grundet en god opdeling i tasks og taiga's funktion med fordeling af arbejdsopgaver gjorde at det trods alt var relativt nemt at arbejde videre og undgå konflikter eller dobbelt arbejde, selvom vi ikke sad sammen. Retrospective blev holdt i starten af uge 4, og det blev besluttet at fortsætte ambitionerne fra sidste restrospective om at benytte taiga grundigt, denne gang med alle mand friske.

Sprint 4

Sprint 4 var samarbejdsmæssigt det bedste sprint og det mener vi skyldes at SCRUM var blevet en mere naturlig del af udviklingsprocessen. Taskboardet styrede processen hele ugen, og det gjorde arbejdet nemt og ligetil for SCRUM master. Det var tydeligt at den mere præcise opdeling i tasks gjorde udvikling mere effektiv, da retningen for en story allerede var vist. Tidsestimeringer var også mest præcise i denne uge, og det mener vi skyldes en bedre opdeling i tasks.

Til retrospective var diskussionen primært om vores sidste udviklingstid skulle bruges på manglende features eller på struktur og oprydning i hele projektet. Det blev besluttet at fokusere på strukturen i koden.

Arbejdsprocess med github

Vi har valgt, fra starten af, at oprette en branch til hver af medlemmerne i gruppen, samt en Developer-branch til merging af vores kode, og til slut en Master-branch til deployering af færdiggjorte sprints. Master-branchen blev også brugt til PO-møderne, da den kun indeholder stabile versioner af programmet.
Denne arbejdsmetode viste sig at være meget effektiv og nem at adaptere til, da alle medlemmer i gruppen var meget disciplinerede med brug af de forskellige branches. Vi stødte kun på enkelte merge-konflikter, som i de fleste tilfælde, var hurtige at rette op på.
En anden mulighed kunne være at have en Developer-branch og Master-branch, og derfra oprette en nye branch, for hver feature der skal implementeres.

Appendix

Sprint 1

ID	Name	Est	Done	How to demo	Notes
#15	Valg af mål	2	Når kunden kan vælge længde, højde og bredde af carport.	Under “Design carport” vælges flat/skråt tag. Vælg mål af carport fra dropdowns(højde, længde, bredde, grader).	
#13	Beregning af spærmængde	2	Længde af spær beregnes ud fra bredden af carporten	Efter valg af mål for enten fladt eller skråt tag, vil den endelige stykliste vise det påkrævede antal spær, samt længde af samme.	
#12	Beregning af antal stolper	2	Antal af stolper beregnes ud fra længden af carporten.	Efter valg af mål, vil den endelige stykliste vise det påkrævede antal stolper, samt længde af samme.	Der medregnes at stolperne skal graves 90 cm ned i jorden.
#9	Udregning af stykliste	5	Systemet kan sammensætte en stykliste vha. databasen.	Åbn en browser, indtast mål og der vises en korrekt stykliste genereret ud fra valgte mål af carport. -Spærmængde -antal stolper	
#10	Beregning af pris	1	Færdig når systemet kan tage styklisten og beregne pris ud fra elementerne derpå.	Vælg dimensioner og tryk “submit”, for at få vist en stykliste med: -Pris pr. enhed -Samlet pris for den samlede vare -Total pris	Bør kun være et overslag.

Sprint 2

ID	Name	Est	Done	How to demo	Notes
#8	Tilvalg af skur	3	Valg af dimensioner.	På kundesiden kan man tilvælge skur med dimensioner og se dette afspejlet på styklisten: -længde -bredde	
#26	Tilføjelse af materialer til stykliste	4	Beregning og tilføjelse til stykliste af følgende materialer: remme, beslag, skruer, beton	På kundesiden skal jeg kunne vælge en carport i mine ønskede dimensioner og se en stykliste der viser de extra materialers mængde.	
#24	Skitse af carport top-down	3	Svg skitse af skuret med skal vises sammen med stykliste.	På kundesiden skal jeg kunne vælge en carport i mine ønskede dimensioner og derefter kunne se en SVG model af min carport der viser placering stolper og spær samt ydre og indre mål	
#24	Skitse af carport fra siden	3	Svg skitse af skuret fra siden.	På kundesiden skal jeg kunne vælge en carport i mine ønskede dimensioner og derefter kunne se en SVG model af min carport fra siden der viser placering stolper og spær samt ydre og indre mål	
#6	Valg af beklædningstyper	1	Kan vælge imellem forskellige materialetyper.	Hvis skur er tilvalgt på kundesiden, kommer en dropdown frem med valgmuligheder af beklædningstyper.	

				Valget fremgår på styklisten.	
#5	Valg af tagtype	1	Kan vælge imellem forskellige materialetyper samt hældning.	På kundesiden, kommer en dropdown frem med valgmuligheder af tagtype. Valget fremgår på styklisten.	

Sprint 3

ID	Name	Est	Done	How to demo	Notes
#8	Tilvalg af skur	3	Valg af dimensioner.	På kundesiden kan man tilvælge skur med dimensioner og se dette afspejlet på styklisten: -længde -bredde	
#16	Kontroller kundens valg af skur	1	Mulighed for at kunne kontrollere valg af mål og godkende samme.	På employee siden vises kunders ordrer og de kan redigeres/godkendes med en knap.	
#17	Mulighed for at tilføje nye varer til DB	3	At man kan indtaste nye varer, så de bliver en del af DB.	Log in og vælg admin siden fra drop down. Her kan indtastes varer. Vis i databasen .	
#18	Mulighed for ændringer i DB	3	Når man kan overskrive eller ændre på varer.	Log in og vælg admin siden fra drop down. Her kan indtastes vare ID. Ændringer til valgte vare kan indtastes i felter. Tryk “update” og varen opdateres i DB. Hvis valgte vare ønskes fjernet trykkes der på “delete” knappen. En “bekræft	

				valg” boks kommer op og skal godkendes eller afvises. Vis i DB at en vare er fjernet/ændret.	
#6	Valg af beklædningstyper	1	Kan vælge imellem forskellige materialetyper.	Hvis skur er tilvalgt på kundesiden, kommer en dropdown frem med valgmuligheder af beklædningstyper. Valget fremgår på styklisten.	
#5	Valg af tagtype	1.5	Kan vælge imellem forskellige materialetyper samt hældning.	På kundesiden, kommer en dropdown frem med valgmuligheder af tagtype. Valget fremgår på styklisten.	