NTNU

Kunnskap for en bedre verden

MISC - en interaktiv prisliste for skipsbelysning

Olav Telseth

Simon Kalsnes Synnes

Kristin Hagen

Einar Weltan

Mai 2019

PROSJEKT / BACHELOROPPGAVE
Data
Norges Naturvitenskapelige Universitet

Veileder 1: Kjell Inge Tomren

Forord

Denne bacheloroppgaven er utarbeidet for Multilux AS, Våren 2019. Oppgaven er den avsluttende delen av vår dataingeniørutdannelse ved NTNU Ålesund.

Vi valgte denne oppgaven for å gi oss selv nye utfordringer. Gjennom denne prosessen har vi opparbeidet oss verdifull erfaring, og er derfor takknemlige for å kunne delta på et slikt prosjekt.

Vi vil takke vår oppdragsgiver Stian Nielsen hos Multilux for å gi oss mulighet til å utføre dette prosjektet og for det gode samarbeidet. Vi vil også takke vår veileder Kjell Inge Tomren hos NTNU for å ha gitt oss mange råd og anbefalinger til arbeidet vi har utført.

Sammendrag

Dette dokumentet beskriver gjennomføringen og resultatet av en bacheloroppgave, utgitt av NTNU Ålesund. Selve problemstillingen er utgitt av Multilux AS.

Multilux er en betydelig leverandør av belysning for samferdsel, infrastruktur, idrettsarenaer, samt en mengde andre utendørs bygg og anlegg. De har nå en veldig tidskrevende løsning for håndtering for salg av lys på båt. I dag blir kundene tilsendt en liste over produkter som de kan velge mellom. Selgerne må deretter skrive ned all nødvendig informasjon om de ønskede produktene i et eget regneark, som de så sender gjennom deres systemer før bestillingen er klar.

Følgende problemstilling er dermed satt for dette prosjektet: "Hvordan kan vi lage en brukervennlig og sikker løsning som holder seg oppdatert med dagens utvalg, spesifikasjoner, priser og valuta?". Multilux ønsket en applikasjon hvor brukeren skulle bli veiledet gjennom en oversikt over fartøy og dets produkter. Det skulle være mulighet for å legge ønskede produker i en handlekurv, og deretter lagre handlekurven som en fil, tilegnet deres systemer for håndtering av bestillinger. Ytterligere kravspesifikasjoner ble inkludert etterhvert som oppgaven pågikk.

Vi sto fritt til å velge hvilke materialer og metoder som skulle brukes, da oppdragsgiver ikke hadde noen spesifikke ønsker om dette. Vi vurderte Java, men endte opp med en tilstandsløs web løsning. Mye av grunnen bak dette var at de fleste i gruppen har hatt faget "Mobile og distribuerte applikasjoner" og har derfor erfaring med slike system. Systemet er derfor delt i 3 deler; frontend, backend og database.

Vi har kalt var løsning "MISC" som står for: "Multilux Interactive Shopping Cart". MISC er spesialisert for oppdragsgivers behov, og fungerer i praksis som en tradisjonell netthandel, uten mulighet for direkte kjøp i nettsiden. Her kan brukere velge skipsbelysning ut i fra hvilke type skip og rom i skipet de har valgt. Man kan deretter legge disse produktene i en handlekurv, som kan lagres i et format egnet for oppdragsgiveren. Nettsiden har også en administrasjonsdel, der de ansatte kan logge inn på nettsiden og redigere nettsidens innhold.

Vi kan med denne rapporten konkludere med at det utviklede systemet oppfyller problemstillingen og oppdragsgivers krav.

Innhold

Begreper

Angular Et rammeverk i HTML/CSS/JS brukt for utvikling av web applikasjoner.

Angular Material Rammeverk som gir et sett med UI-komponenter tilgjengelig for Angular utviklere.

API Et grensesnitt i en programvare som gjør at spesifikke deler av denne kan kjøres/aktiveres fra en annen programvare.

ARC Gjør det mulig å gjøre REST/HTTP kall for testing av endpoints.

Backend Kode som kjører direkte på serveren, ofte for å håndtere data, autentisering osv.

Cookie informasjonskapsel som lagrer data lokalt i nettleseren til brukeren.

CSV dataverdier separert med komma.

Database En organisk samling med data som kan hentes og endres elektronisk.

Frontend Koden som former det du ser på skjermen og interaksjoner mellom disse elementene.

HTML Element Består av to HTML noder. En start og slutt node: <test> et element </test>

JSON En enkel tekstbasert standard for å formatere dokumenter som brukes for datautveksling, for eksempel mellom en server og en web-applikasjon.

Product backlog De oppgavene som er definert i et Scrum prosjekt.

RESTful API En bestemt type API som gjør det mulig å ha kommunikasjon mellom separate programmer over nettet.

SCRUM Scrum er en arbeidsmetode for hvordan man skal strukturere lag og arbeid.

Server Programvare som gjør at en datamaskin kan tilby tjenester til andre maskiner.

Single Sign On En tjeneste som lar brukere bruke kun ett sett med brukernavn og passord for å logge inn på flere forskjellige applikasjoner og tjenester.

SQL Et spørrespråk for databaser som benyttes til å formuler og kjøre operasjoner mot databaser (lese, skrive, endre og slette data).

SQL spørringer En forespørsel til en SQL-database.

TypeScript Et open source programmeringsspråk som baserer seg på JavaScript, og legger til funksjonalitet som static typing og kompilering til JavaScript.

User Interface Brukergrensesnittet til en programvare

Visual Studio Code En kodeeditor utviklet av Microsoft.

Wizard Type brukergrensesnitt som trinnvis fører brukeren gjennom en prosess.

Card En komponent ifra Angular Material bibloteket. Blir som oftest brukt når man har en side med flere mindre komponenter hvor man deler komponentene opp i blokker som ligner på kort.

Forkortelser

ARC Advanced Rest Client

API Application Programming Interface

CLI Command-Line Interface

CRUD Create, read, update and delete

CSV Comma-Separated Values

ECDSA Elliptic Curve Digital Signature Algorithm

FAB Floating Action Button

HMAC Hash-Based Message Authentication Code, også kjent som Keyed-Hashing for Message Authentication

HTTP HyperText Transfer Protocol

HTTPS HyperText Transfer Protocol Secure

JS JavaScript

JSON JavaScript Object Notation

JWT JSON Web Token

MISC Multilux Interactive Shopping Cart

MITM Man-in-the-middle

REST Representational State Transfer

ROI Return On Investment

SQL Structured Query Language

SSL Secure Sockets Layer

SW Software

TLS Transport Layer Security

UI User Interface

VS Code Visual Studio Code

Figurer

Tabeller

1 Innledning

1.1 Bakgrunn og problemstilling

"Multilux er en betydelig leverandør av belysning for samferdsel, infrastruktur, idrettsarenaer, samt en mengde andre utendørs bygg og anlegg". [36]

Multilux har hovedkontor i Bø, Telemark, og har i over 15 år vært en betydelig leverandør av utendørs belysning. Ved hjelp av et nettverk av både nasjonale og internasjonale underleverandører, er de blant de beste innen sitt fag når det gjelder teknologisk og praktisk utvikling. [36]

Multilux ønsker å utvide målgruppen sin for å få et fotfeste innen skipsindustrien, som er så og si urørt blant norske leverandører av belysning. I den sammenheng ønsker de å gjøre salgsprosessen enklere for både kunder og selgere.

Frem til nå har Multilux brukt Excel-dokumenter, som inneholder informasjon om de forskjellige varene som ligger til salgs. Dette er tungvint for både kunde og selger. Selgeren må manuelt skrive kundens orde inn i systemet, noe som kan være tidskrevende. Kunden må også skrive ned de ønskede produktene fra produktlisten, for så å sende den til Multilux.

Den interaktive prislisten vil effektivisere denne prosessen. Dette gjør at kunden enkelt og oversiktlig kan velge ønskede produkter og få tilbake en riktig formatert ordreliste. Kunden kan så sende denne ordrelisten til Multilux, som legger den inn i sine systemer.

Følgende problemstilling er dermed satt for dette prosjektet: "Hvordan kan vi lage en brukervennlig og sikker løsning som holder seg oppdatert med dagens utvalg, spesifikasjoner, priser og valuta?".

1.2 Oppgaven

Multilux ønsker en interaktiv prisliste, der kunden selv kan finne produkter og dens tilhørende spesifikasjoner og pris. Produktene er belysning for fartøy og prislisten skal dermed kunne vise produkter tilhørende de ulike fartøyene og dets rom. (Vedlegg B)

Prislisten skal settes opp som en tradisjonell nettside for netthandel, men uten mulighet for direkte kjøp i nettsiden. I stedet skal kunder kunne eksportere handlelisten til en CSV-fil, basert på Multilux sitt eksisterende oppsett i Excel (Vedlegg B), som kunden så kan sende til Multilux.

1.3 Kravspesifikasjon

For å løse oppgaven og problemstillingen, ble følgende kravspesifikasjon utarbeidet i samarbeid med Multilux.

Kundene skal ha tilgang til følgende funksjoner:

- Oversikt over alle fartøy.
- Veiviser for å finne underprodukt ved hjelp av fartøy, rom og produkt.
- Liste over alle tilgjengelige produkter.
- Se priser i forskjellige valutaer.
- Legge produkter i handlekurven.
- Handlekurv:
    - Se valgte produkter.
    - Fjerne uønskede produkter.
    - Endre antall ønskede produkter.
    - Laste ned en PDF av handlekurven, som passer til utskrift.
    - Laste ned en riktig formatert CSV-fil av handlekurven.

På grunn av mulige endringer, av blant annet vareutvalg og priser, har vi valgt å gå for en modulær løsning over en statisk løsning. Dette vil si at Multilux selv kan legge til og fjerne produkter etter behov.

Applikasjonen skal tillate selgere å logge inn, hvor de deretter får tilgang til systemets redigeringsmodus. Her skal de ha mulighet til å kunne gjøre følgende administrative funksjoner:

- Lage nye brukere.
- Endre sine egne brukernavn og passord.
- Legge til nye og slette valutaer, samt sette valutaens kurs.
- Lage nye, endre på og slette fartøy, rom og produkter.
- Legge eller fjerne produkter fra fartøyenes rom.

I tillegg skal både fartøy og produkter kunne skjules fra den vanlige visningen som kundene har tilgang til. På denne måten kan man legge til nye fartøy og produkter og bestemme synligheten for kunder. Når man oppretter et nytt fartøy eller produkt er disse automatisk satt som skjult.

1.4 Rapportinnhold

Resten av rapporten er strukturert som følger.

Kapittel 2 - Teoretisk grunnlag: Kapittelet forklarer teorien bak gruppens arbeidsmetode og designvalg. Teorien bak de verktøy og teknologier som blir benyttet i prosjektet vil også bli forklart her.

Kapittel 3 - Materialer og metode: Dette kapittelet beskriver prosjektets organisering og planlegging, samt verktøyene som er tatt i bruk for å løse oppgaven.

Kapittel 4 - Resultater: Kapittel 4 tar for seg det endelige produktet i frontend og backend. Her ligger en oversikt over applikasjonens struktur og konseptmodell. Her finnes også en gjennomgang av applikasjonens brukergrensesnitt, samt en forklaring på hvordan distribusjon og administrasjon av applikasjonen håndteres.

Kapittel 5 - Drøfting: I dette kapittelet drøfter vi prosjektet i sin helhet.

Kapittel 6 - Konklusjon: Vår konklusjon for prosjektet presenteres i dette kapittelet.

2 Teoretisk grunnlag

2.1 Smidig arbeidsmetode og Scrum

2.1.1 Smidig metode

Smidig metode er et rammeverk for prosjektstyring. Dette rammeverket innebærer i hovedsak at oppgaver utføres trinnvis og gjennom iterasjoner. Fordelen med dette er at man kan være bedre forberedt på avvik og endringer. [50]

Smidige metoder krever tett dialog med kunden, noe som resulterer i hyppige tilbakemeldinger. Dette gjør at kunden kontinuerlig blir med å forme sluttresultatet. Utviklingsteamet får også på denne måten bedre innsikt i hva de skal ende opp med. [50]

2.1.2 Scrum

Scrum er et smidig rammeverk for teamarbeid som er ofte brukt innenfor systemutvikling. Scrum har blitt mer vanlig i nyere tid på grunn av sin evne for problemløsning, tilpasning og samarbeidmuligheter. [50]

I scrum jobber man i et team på 5 til 9 personer. Teamet er tverrfaglig, og er lagt opp til å kunne håndtere de fleste utfordringer. Sammen har de den kunnskapen som trengs for å jobbe raskt og effektivt. [50]

I Scrum deles det endelige produktet opp i mindre deler for å lettere kartlegge sluttresultatet. Disse delene kalles til sammen "product backlog" og hver del er et "backlog item". Et backlog item inneholder en detaljert beskrivelse av hva som skal gjøres i denne delen. Det er også definert krav til fullførelse av hvert backlog item. [50]

Teamet jobber med en liten del av product backloggen i en gitt periode, kalt en sprint. En sprint varer vanligvis en til to uker. Etter hver sprint har teamet et møte hvor oppdragsgiver er til stede. Her diskuteres det også hvordan teamets prosesser og produksjon kan forbedres. [50]

Det holdes også daglige møter, kalt "Daily Scrum meeting". Møtene holdes stående og varer i maksimalt 15 minutter. Her snakker teammedlemmene sammen og diskuterer kun de viktigste sakene. Vanligvis skal hvert medlem si noe om hva de gjorde i går, hva de planlegger å gjøre i dag og hva som kan hindre dem i arbeidet fremover. [50]

2.1.3 Lagstruktur

I scrum er det vanligvis tre roller. Disse er: Scrum-master, produkteier og teammedlem. [50]

Scrum master Scrum eksperten som vet om alle scrum rutiner og regler. En scrum master har som hovedoppgave å hjelpe og forbedre laget sin ytelse innad i scrum rammerverket.

Produkteier Har hovedansvar for maksimering av produktets ROI. Denne personen fokuserer laget sin innsats på det som skaper mest verdi. Dette gjør produkteier ved å sette forskjellig prioriteringer på visse problem.

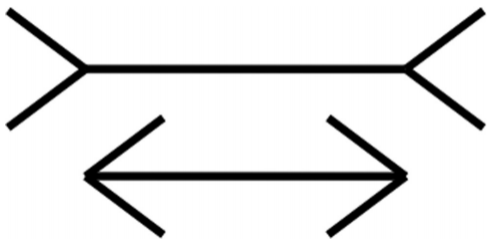
Lagmedlem Har høyt ansvar for sitt eget produkt. Hovedoppgaven er å skape profesjonelle produkt og resultat i hver sprint.

2.2 Design og Layout

2.2.1 Forutinntatt Oppfatning

Dette underkapitlet baserer seg på kapittel 1 i "Designing with the Mind in Mind" [29]. Boken forteller at menneskets oppfatning er forutinntatt. Dette betyr at måten individer tolker ting på nødvendigvis ikke er sant. Menneskets oppfatning er sterkt påvirket av tre faktorer:

1. **Fortid — Våre erfaringer**
   Vi lager koblinger mot tidligere hendelser og erfaringer for å bedre forstå nåværende situasjoner og problemstillinger. Se for deg at du blir presentert et todimensjonalt bilde med fem forskjellige figurer. Dersom en eiendomsmegler viser deg bildet, vil du naturlig tenke at dette har noe med bygg å gjøre. På den andre siden, hvis en kunstner hadde vist deg bildet ville du mest sannsynligvis ikke tenkt at det hadde noe med bygg å gjøre.
2. **Nåtid — Vår nåværende situasjon**
   En persons oppfatning representerer ikke alltid sannheten. Dette har med hvordan vi mennesker innhenter informasjon. For eksempel er de horisontale linjene i figuren nedenfor (Figur 2.1) like lange, men for oss ser den øverste linjen litt lengre ut.



Figur 2.1: "Müller–Lyer illusion: equal-length horizontal lines appear to have different lengths" [29]

3. **Fremtid — Våre mål og ønsker**
   Vi filtrerer ut det som ikke er en del av vårt mål. For eksempel hører vi mange forskjellige lyder til enhver tid, men vi filtrerer vekk de lydene vi ikke trenger å høre på.

For å få disse tre faktorene under kontroll kan vi gjøre følgende:

1. **Unngå tvetydighet**
   Unngå tvetydig informasjon, og test design og layout for at alle tolker det på lik måte. Når dette er vanskelig, bør man bruke etablerte standarder innen design og layout. Dersom brukeren ofte tar i bruk en spesifikk nettside, kan man selv benytte samme standarder for design og layout. På denne måten vil brukeren forstå den nye nettsiden mye raskere. Mennesker liker generelt godt det som er kjent.
2. **Være konsistent**
   Plasser informasjon og funksjoner på samme plass gjennom hele applikasjonen. Eksempler på dette kan være at knappene for navigering alltid bør være på samme sted, eller at "Legg til" knappen alltid ligger på samme sted, uavhengig av innholdet ellers på siden.
3. **Forstå målet til brukeren**
   Forstå hva målet til brukeren er når de bruker applikasjonen. Brukerens mål kan variere basert på hvilken type applikasjon det er. Det er viktig å fokusere på at det brukeren trenger mest skal være lett å finne.

2.2.2 Hånd-Øye Koordinasjon

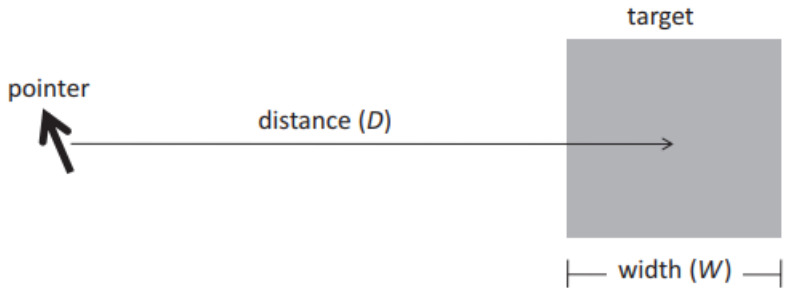
Fitts lov

"Closer and bigger target = faster reach" [29]

$$T = A + B \cdot \log_2\left(1 + \frac{D}{W}\right)$$

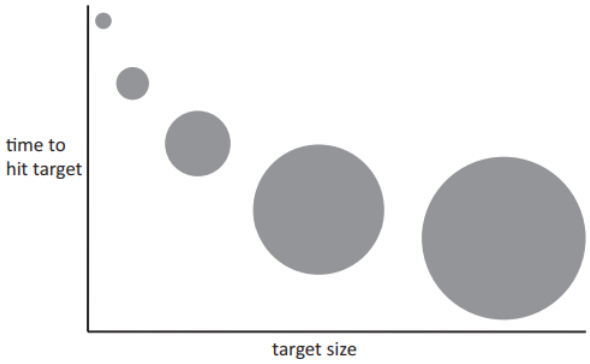
T = Tid for å nå målet.
A = Hvor enkelt det er å starte å avslutte bevegelsen.
B = Hvor enkel bevegelsen er.
D = Distansen til målet.
W = Bredden til mål elementet.

Figur 2.2: Fitts lov [29]



Figur 2.3: "Fitts' law: pointing time depends on distance (D) and the width (W) of the target" [29]

Loven for å peke på et mål (Figur 2.2 og 2.3) er oppkalt etter mannen som først oppdaget det, Paul Fitts. Den forklarer sammenhengen mellom hastighet og treffsikkerhet når det kommer til menneskets muskelbevegelser. I hovedsak beviser den at et større mål og mindre avstand til målet fører til at man kan peke på målet raskere og med større treffsikkerhet. Figur 2.4 under er et fint grafisk eksempel på sammenhengen mellom distanse til objekt og bredde på objekt. [20]



Figur 2.4: "Fitts' law: pointing time depends on distance (D) and the width (W) of the target." [29]

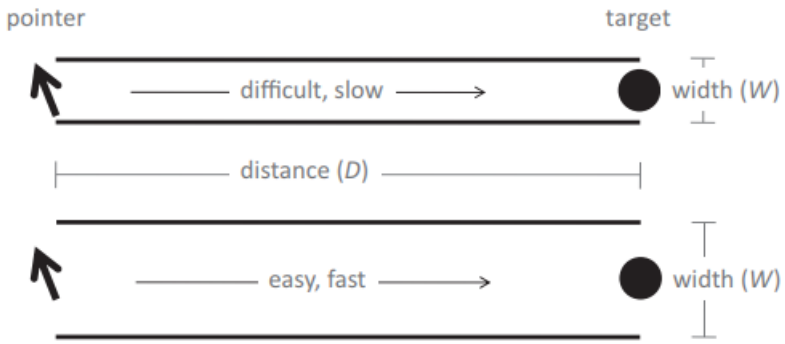
Steering loven

"If you must keep a pointer within a certain confined path while moving it to a target, then the wider the path, the faster you can move the pointer to the target" [29]

$$T = a + b\left(\frac{D}{W}\right)$$

T = Tid for å nå målet.
a = Hvor enkelt det er å starte å avslutte bevegelsen.
b = Hvor enkel bevegelsen er.
D = Distansen til målet.
W = Bredden på veien til målet.

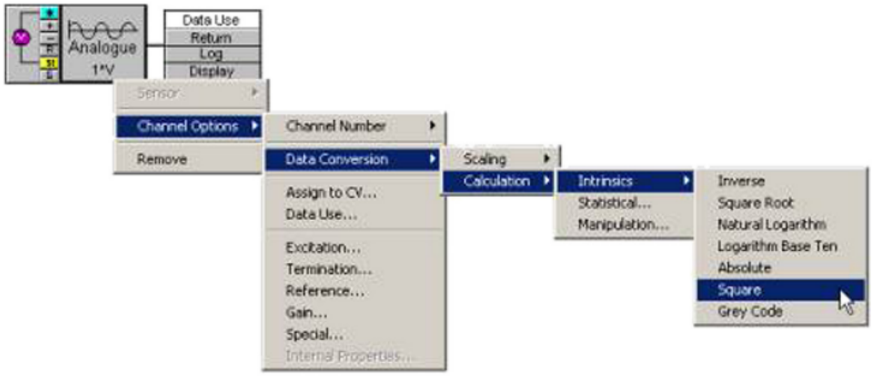
Figur 2.5: Steering loven [29]



Figur 2.6: "Steering law: pointing time depends on distance (D) and width (W) of path." [29]

Figurene ovenfor (Figur 2.5 og 2.6) beskriver steering loven. Loven forklarer tiden det tar å bevege pekeren gjennom en begrenset vei eller tunnel. Dette er relevant når vi må holde pekeren innen et visst område for å holde en meny oppe. [29]

Steering loven kan brukes i sammenheng med undermenyer, hvor menyen lukkes dersom pekeren havner utenfor undermenyene. Figuren under (Figur 2.7) er et eksempel på dårlig bruk av steering loven. Dette kommer av at brukeren må navigere gjennom en lang og avansert sti for å nå målet sitt. Et bedre eksempel hadde vært å delt opp menyen og bare hatt en undermeny på hver meny.



Figur 2.7: Eksempel på dårlig bruk av steering loven

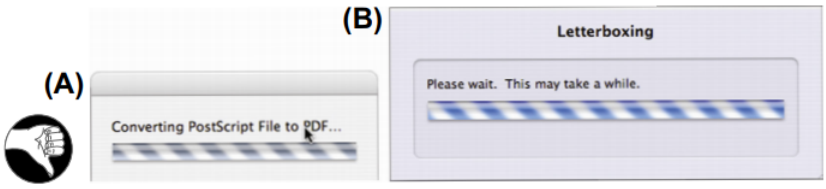
2.2.3 Respons

Respons er relatert til ytelse, men er ikke det samme. Ytelse er målt basert på beregninger per tidsenhet. Respons er målt i henhold til menneskelige tidskrav og brukertilfredshet. Dette betyr at interaktive systemer kan være responsive, selv om systemet har lav ytelse. [29]

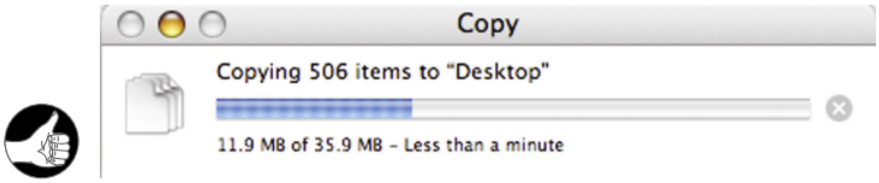
Trinn for bra respons [29]:

- La brukeren få vite med en gang at input har blitt mottatt.
- Gi en slags indikasjon på hvor lang tid operasjonen kommer til å ta.
- Frigjør brukeren til å gjøre andre ting mens de venter.
- Behandle operasjoner i kø på en intelligent og logisk måte.
- Gjør vedlikehold og lav prioritets oppgaver i bakgrunnen.
- Forutse de mest vanlige forespørslene.

Med dette kan interaktive systemer være unresponsive, selv om de logiske handlingene som skjer i bakgrunnen er effektive. Dersom programmet ikke gir noen indikasjon på ventetider, eller det ikke er noen indikasjon på at handlingene brukeren tar oppfattes av systemet, kan programmet oppfattes som tregt. [29]



Figur 2.8: “Mac OS X file transfer: good progress indicator, useful time estimate, and cancel button (circled X).” [29]



Figur 2.9: “No progress bar (just a busy bar) and no cancel for (A) Mac OS X and (B) iMovie.” [29]

Figur 2.8 og 2.9 er et godt eksempel på forskjellen mellom bra og dårlig design for god responsivitet. På figur 2.8 får brukeren nesten ingen informasjon om hva som foregår. Brukeren får heller ikke vite hvor lang tid det vil ta, hvor mye som gjenstår og har ingen måte å avbryte prosessen. På figur 2.9 er alle de behovene dekt. Brukeren kan da anta hvor lang tid dette vil ta og hvis brukeren angrer eller gjorde en feil, er det mulighet for å avbryte prosessen.

Viktige ting å tenke på for "loading" indikator [29]

- Animasjon bedre enn statisk tekst.
- Stopp animasjonen og informere brukeren om noe har gått galt.
- Alt som blokkerer brukerinteraksjon skal ha en indikator på at det laster.
- Indikasjon på hvor mye tid som gjenstår.
- Mulighet for å avbryte slik at brukeren ikke blir låst ute av systemet.

Andre viktige designprinsipper for god respons [49]

- Hold brukeren opptatt, ikke la brukeren sitte å vente.
- Ingen forsinkelse mellom oppgavene (OK mellom oppgaver, ikke OK i en oppgave).
- Vis det viktige først, detaljer kommer senere.
- Forutse hva brukeren kommer til å gjøre for bedre ytelse .

2.2.4 Veiviser (Wizard)

Ifølge Wikipedia [61] er en veiviser en brukergrensesnittstype som presenterer brukeren med et sett av dialogbokser som fører brukeren gjennom definerte trinn for å oppnå et sluttresultat eller for å finne frem til noe. En veisviser kan være hjelpsomt om det er komplekse eller avanserte oppgaver som skal utføres.

Et eksempel på en veiviser er installasjonsveiviseren til et program. Her tar den deg gjennom noen standardiserte trinn for å hjelpe deg å installere og sette opp programmet på datamaskinen din.

2.3 Programvaredesign

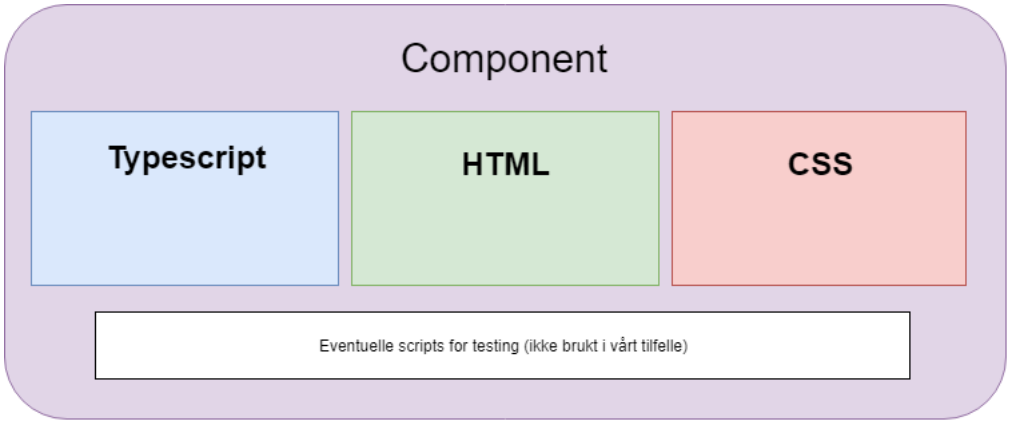
2.3.1 Coupling og cohesion

Coupling og cohesion refererer til to designprinsipper innen objektorientert programmering [42]. Et program som har høy coupling vil gjerne ha mange moduler eller komponenter som er avhengige av hverandre. Dersom man endrer en av disse modulene kan det ha uønskede effekter på andre komponenter. Det er derfor anbefalt å ha lav coupling mellom komponenter. [58]

Cohesion er i hvor stor grad delene av en modul eller et komponent henger sammen. En modul med lav cohesion vil for eksempel ha metoder som ikke har noe med hverandre å gjøre. Dette fører til at det blir vanskeligere å gjenbruke modulen senere, og det gjør det vanskeligere å vedlikeholde systemet siden man kan bli nødt til å endre på flere moduler enn nødvendig. På grunn av dette er det anbefalt å ha høy cohesion. [57]

2.3.2 Komponentbasert struktur

Komponentbasert struktur vil si at hvert element du ser på siden er en komponent. En komponent er enkelt forklart en gruppering av JavaScript, HTML [60] og CSS [59]. Dette fører til at komponentene i applikasjonen er uavhengig av hverandre, noe som leder til god cohesion og lav coupling. [21][18]



Figur 2.10: Visning av hvilke filer en komponent består av

Figur 2.10 viser komponenten som en helhet, men selve klassen til komponenten ligger i TypeScript [34]. Når vi oppretter en komponent setter vi først: selector, template og style.

Selector Bestemmer hvordan du kan definere komponenten som et HTML-element. For eksempel vil "newsPage-Component" ha selector: "news-page". Da blir <news-page></news-page> HTML-navnet for den komponenten.

Template Bestemmer det visuelle til komponenten. Template inneholder HTML-kode og kan holde på enten HTML-elementer eller andre komponenter. Du kan enten skrive template direkte i typescript filen eller så kan du importere en egen HTML-fil som template.

Style Bestemmer hvilken stil som skal legges til på template. Style inneholder CSS-kode, men du kan også bruke filer som SCSS og lignende. Du kan enten skrive style direkte i typescript filen eller så kan du importere en egen CSS/SCSS-fil som style.

2.4 Sikkerhet

I følge Næringslivets Sikkerhetsråd sine Mørketallundersøkelser, opplever over en fjerdedel av norske virksomheter hvert år uønskede sikkerhetshendelser i sine datasystemer [46]. Beskyttelse mot dataangrep er derfor svært viktig for all systemutvikling, og sikkerhetstiltak bør planlegges tidlig i utviklingsstadiet.

2.4.1 Angrep

Brute force søkeangrep

Et brute force angrep, er når en angriper gjetter mange passord helt til det riktige passordet blir gjettet. Angriperen kontrollerer systematisk alle mulige passord til den har funnet frem til det korrekte passordet. Dette er veldig effektivt for å sjekke passord som er korte, men det sliter ved sjekking av lange passord. Dette er fordi den blir nødt til å sjekke for ekstra verdier og gjettingen tar derfor mye lenger tid. Man bruker heller ordbok angrep om dette er tilfellet. [31]

Ordbok angrep

Et ordbok angrep er en form for brute force angrep, hvor man tester sannsynlige passord istedet for å prøve alle mulige passord [4]. En stor andel brukere har passord på 8 tegn eller mindre. Mange av disse passordene er alfanumeriske, som vil si at de kun består av bokstaver og tall [43].

Et ordbok angrep har som formål å gå gjennom en stor mengde brukere på kort tid, og man leter da spesifikt etter slike brukere med lav passordsikkerhet. Dette gjøres ved å gå ut fra en liste med vanlige passord, eller ved å ta utgangspunkt i ordene fra en ordbok. Ved å benytte seg av denne metoden reduserer man antall forsøk fra flere millioner til hundretusener per bruker. Slik kan man angripe flere brukere mye raskere enn ved et normalt brute force angrep. [4]

Rainbow table angrep

Et rainbow table angrep består av en forhåndskalkulert oppslagstabell for å reversere kryptografiske hash funksjoner, vanligvis for cracking av passord hasher. Slike tabeller brukes vanligvis til å gjenopprette et passord opp til en viss lengde bestående av et begrenset sett med tegn. Dette angrepet bruker mindre tid på dataprosessering men bruker mer lagringsplass enn et brute force angrep som beregner en hash ved hvert forsøk. Angrepet har likevel mer prosesseringstid, og bruker mindre lagringsplass, enn en enkelt oppslagstabell med en oppføring per hash. [28]

Man-in-the-middle angrep

Et man-in-the-middle angrep er et angrep der angriperen i all hemmelighet plukker opp, videresender og muligens endrer på nettverkskommunikasjonen mellom to parter. [11]

Et eksempel på et MITM-angrep er tyvlytting. Angriperen oppretter uavhengige forbindelser med ofrene og sender meldinger mellom dem for å få dem til å tro at de snakker direkte til hverandre over en privat forbindelse, når hele samtalen faktisk styres av angriperen. Alle relevante meldinger som går mellom de to partene kan fanges opp, og nye meldinger kan bli injisert. Dette kan være en veldig lett operasjon, om man for eksempel er innen rekkevidde til et trådløst nettverk som er ubeskyttet. [11]

2.4.2 Beskyttelse

bcrypt

Bcrypt ble designet for hashing av passord og er derfor en langsom algoritme. Dette er bra for hashing av passord, siden det reduserer antall passord per sekund en angriper kan hashe når han lager et ordbok angrep. [37]

For å beskytte mot rainbow table angrep, kan man inkludere en salt. Salt er tilfeldig data som blir lagt til i en enveis funksjon for å hashe data, slik som passord [3]. Dermed vil ikke et rainbow table angrep klare å reversere hashen, da den har ekstra data som ikke kan reverses [28].

I tillegg til å inkludere en salt, er bcrypt en adaptiv funksjon, som vil si at man kan øke iterasjonstallet over tid for å gjøre generering av hashen langsommere. Slik forblir funksjonen motstandsdyktig mot brute force søkeangrep, selv om angriperen har økt beregningskraft. [37]

HTTPS

HTTPS er en sikrere versjon av HTTP, der trafikken mellom nettside og klient er kryptert ved bruk av TLS/SSL. HTTPS blir brukt i de fleste nettsider hvor sensitiv informasjon blir utvekslet. [45]

Noen av grunnene til å bruke HTTPS er for autentisering av brukere, beskyttelse av personvern og for å beskytte integriteten til data. Man beskytter seg dermed mot man-in-the-middle angrep, der man prøver å få tak i data som går fram og tilbake mellom klienten og nettsiden. Ved å bruke HTTPS kan man derfor føle seg sikker på at brukere kan kommunisere med din nettside, uten å risikere at informasjonen de sender blir fanget opp av andre. [45]

JSON Web Token

JWT er en åpen standard, RFC 7519 [15], som definerer en kompakt og selvstendig måte for sikker overføring av informasjon mellom parter som et JSON-objekt. Denne informasjonen kan verifiseres og stoles på da den er digitalt signert. [30]

Autorisering er det vanligste scenariet for bruk av JWT. Når brukeren er logget inn, vil hver etterfølgende forespørsel inkludere JWT, slik at brukeren får tilgang til ruter, tjenester og ressurser som er tillatt med den tokenen. Single Sign On er en funksjon som i stor grad bruker JWT i dag, på grunn av lite “overhead” og kan enkelt bli brukt på forskjellige domener. [30]

JWTer kan signeres ved hjelp av en “secret” med HMAC-algoritmen [24], eller et offentlig / privat nøkkelpar ved hjelp av RSA [16] eller ECDSA [17]. Signerte tokens kan verifisere integriteten til innholdet sitt, mens krypterte tokens gjemmer innholdet fra andre parter. Dette vil si at JSON Web Tokens er en god måte å sikre overføringen av informasjon mellom flere parter. Ved hjelp av offentlige / private nøkkelpar kan du være sikker på at identiteten til senderen er korrekt. [30]

Siden signaturen er beregnet ved hjelp av en header og en "payload", kan du også kontrollere at innholdet ikke har blitt manipulert. I sin kompakte form består JSON Web Tokens av tre deler separert av prikker, “.”, som er:

- Overskrift.
- Payload.
- Signatur.

Derfor ser en JWT vanligvis ut som følgende: xxxxx.yyyyy.zzzzz [30]

Tofaktorautentisering

Tofaktorautentisering er når et system krever to forskjellige autentiseringsfaktorer for at brukeren kan logge seg inn. Som oftest er en av disse to faktorene et vanlig passord. Den andre autentiseringsfaktoren kan for eksempel være en autentiseringsbrikke [40], en kode som man får tilsendt på SMS eller skanning av fingeravtrykk. [41]

Ved å bruke tofaktorautentisering får man dermed et ekstra lag med sikkerhet utover ordinære passord. Dette betyr at det blir vanskeligere å angripe brukere med brute force, ordbok og rainbow table angrep. [41]

2.5 RESTful API

Et API blir brukt for kommunikasjon mellom platformer over nettet [56]. REST er ikke en standard eller en protokoll, men heller en tilnærming til hvordan et API skal se ut. Om en backend server har REST API og man gjør forespørsler til dette APIet på klientsiden, så er klienten RESTful. REST baserer seg på HTTP, og alle operasjonene som blir gjort fra APIet bruker HTTP metoder. [39]

Bestepraksis til design av RESTful API kan bli brutt ned til fire operasjoner. Dette inkluderer:

- GET - hente data.
- POST - opprette data.
- PUT - oppdatere/modifisere data.
- DELETE - slette data.

Et REST API fungerer altså som det eneste grensesnittet for forespørsler fra klienter som de skal forholde seg til og det er veldig praktisk. [39]

2.6 Serveradministrasjon og sikkerhet

Når man skal kjøre Node.js og Angular applikasjoner i et produksjonsmiljø har man et par valg. Man kan enten velge å kjøre det rett fra kommandolinjen på en server eller å kjøre det som en service. Det er et populært valg å kjøre det som en service da applikasjonene kan settes opp til å starte på nytt automatisk dersom systemet krasjer eller serveren starter på nytt. Administrasjonsdelen av programmene blir ganske enkel dersom man har satt opp Node.js programmet som en service med en prosessovervåker. Det meste av administrering blir dermed overvåkning av prosessorbruk og lignende. [12]

Man kan ta forskjellige tiltak for å øke sikkerhet på server/applikasjonene: [5]

- Installere sikkerhetsoppdateringer.
- Overvåke systemet.
- Begrense tilgangen til systemet.
- Ta regelmessige sikkerhetskopier av databasen.
- Bruke sterke passord.

2.7 Modulær og Skalerbar Database

I en relasjonsdatabase er tabellen en relasjon fordi den lagrer relasjonen mellom data i et kolonne-rad format. Kolonnene er tabellens attributter, mens radene representerer registreringene. En enkelt rad kalles en "tuple". [54]

2.7.1 Relasjoner

Relasjoner gjør det mulig å beskrive forbindelsene mellom ulike databasetabeller. Disse relasjonene kan brukes til å utføre spørringer mot tabeller, kjent som "joins". [54]

Det er tre forskjellige typer databaserelasjoner, hver av dem er oppkalt etter antall tabellrader som kan være involvert i relasjonen. Hver av disse tre relasjonstypene eksisterer mellom to tabeller. [54]

1. **En-til-en**
   Denne relasjonstypen oppstår når hver oppføring i den første tabellen har en, og bare en, motpart i den andre tabellen. En-til-en-relasjoner brukes sjelden, fordi det ofte er mer effektivt å bare sette all informasjonen i en enkelt tabell. Enkelte databasedesignere utnytter dette forholdet ved å lage tabeller som inneholder en delmengde av dataene fra en annen tabell. [10]

2. **En-til-mange**
   Dette er den vanligste typen databaserelasjon. Denne oppstår når hver registrering i tabell A tilsvarer en eller flere registreringer i tabell B, men hver registrering i tabell B tilsvarer bare en registrering i tabell A. [10]

   For eksempel, i en relasjon mellom tabell "lærer" og tabell "student" i en database kalt "skole", vil relasjonen trolig være en en-til-mange relasjon. Dette er fordi hver student bare har en lærer, men hver lærer har flere studenter. En-til-mange-designet bidrar til å eliminere duplikat data som igjen gjør det det mer skalerbart. [10]

3. **Mange-til-mange**
   Mange-til-mange relasjoner oppstår når hver registrering i tabell A tilsvarer en eller flere registreringer i tabell B, og hver registrering i tabell B tilsvarer en eller flere registreringer i tabell A.

   For eksempel vil forholdet mellom tabell "lærer" og tabell "fag" trolig være mange-til-mange, fordi hver lærer kan undervise i mer enn ett fag, og hvert fag kan ha mer enn én lærer. [10]

2.7.2 Skalerbarhet

Skalerbarhet i databaser har tre grunnleggende dimensjoner: Størrelsen på data, antall forespørsler og størrelsen på forespørsler. [9]

Forespørsler kommer i mange størrelser. Transaksjoner påvirker generelt små mengder data, men kan nærme seg tusenvis per sekund. Analytiske spørringer er generelt færre, men kan få tilgang til flere data. Et relatert konsept er elastisk databehandling, systemets evne til å legge til og fjerne kapasitet for å tilpasse seg nye arbeidsmengder. [9]

Man har to ulike tilnærminger til skalering. Disse er vertikal og horisontal skalering. [47]

1. **Vertikal databaseskalering**
   Dette innebærer at databasesystemet kan fullt utnytte maksimalt konfigurerte systemer, inkludert flere prosessorer med stort minne og stor lagringskapasitet. Slike systemer er relativt enkle å administrere, men kan tilby redusert tilgjengelighet. Enhver datamaskin har imidlertid en maksimal konfigurasjon.

   Hvis arbeidsbelastningen utvides utover denne grensen, er valgene enten å overføre til et annet, større system, eller å endre systemet for å oppnå horisontal skalerbarhet. [47]

2. **Horisontal databaseskalering**
   Dette innebærer å legge til flere servere for å arbeide med en enkelt arbeidsbelastning. De fleste horisontalt skalerbare systemene har kompromisser angående funksjonalitet. Hvis et program krever mer funksjonalitet, er det ofte foretrukket å migrere til et vertikalt skalert system. En skalerbar løsning klarer å tilpasse seg stadig større mengder lagrede data ved å skalere opp både antall maskiner og antall lagringsenheter i det totale systemet. [47]

2.8 Modeller og Entitetsklasser

En entitet representerer en enkelt forekomst av domeneobjektet lagret i databasen. Den har attributter, representert som kolonner i tabellene. [6]

En modell representerer vanligvis et objekt som er relatert til problem- eller domeneområdet. I programmet lager vi klasser for å representere objekter. Disse klassene, kjent som modeller, har egenskaper og metoder som definerer objektets adferd. [51]

3 Materialer og metode

3.1 Prosjektorganisasjonen

3.1.1 Prosjektgruppen

Prosjektgruppen består av fire avgangsstudenter fra NTNU i Ålesund. Alle går studiet Bachelor i ingeniørfag - data, ved Institutt for IKT og realfag. Gruppemedlemmenes kandidatatnummer er som følger:

- 10023
- 10043
- 10046
- 10061

3.1.2 Oppdragsgiver

Oppdragsgiveren for dette prosjektet er Multilux AS. Multilux hadde et ønske om å effektivisere salgsprosessen, og ønsket i den sammenheng en applikasjon som kunne fungere som en interaktiv prisliste for deres produkter. Multilux hadde i utgangspunktet en viss idé om hvordan produktet skulle se ut, men de hadde ingen spesifikke tanker om hvilke teknologier som skulle tas i bruk. Vi utarbeidet en kravspesifikasjon sammen med vår kontaktperson i Multilux.

3.1.3 Veileder

Prosjektgruppens veileder har vært Kjell Inge Tomren. Kjell Inge er programansvarlig for studieprogrammet Dataingeniør og er universitetslektor ved Fakultet for informasjonsteknologi og elektroteknikk i Ålesund. Han har også vært foreleser i faget Webteknologi og Systemadministrasjon. Som veileder har han besvart oss på diverse spørsmål. I tillegg til dette har vi hatt møter for å gå gjennom fremgangen i prosjektet.

3.1.4 Prosjektorganisering

Organiseringen i gruppen er i utgangspunktet demokratisk med rollene: prosjektleder, sekretær, frontend- og backend ansvarlig. Det at gruppen er demokratisk innebærer at alle deltar i større valgavgjørelser, og vi går for det valget der flest er enige. Gruppen samarbeider aktivt ved hjelp av kommunikasjonverktøy over internett og møter på skolen.

Rolle	Navn / Kandidatnummer
Oppdragsgiver	Multilux AS
Prosjektleder	10023
Frontend ansvarlig	10043
Sekretær	10046
Backend ansvarlig	10061

Tabell 3.1: Tabell for prosjektorganisering

Oppdragsgiver

Oppdragsgiver sin rolle i prosjektet er å se over produktet og tilføye eventuell informasjon og endringer som trengs. Vi distribuerer alltid nyeste versjon av systemet til telseth.no [1]. Dette gir oppdragsgiver mulighet til å kunne gå inn på websiden og se over hvordan systemet ser ut i løpet av utviklingen. De vil med dette ha mulighet til å gi tilbakemelding til prosjektleder, dersom det er noe de ønsker å forandre på.

Prosjektleder

Prosjektlederen har overblikk over metoder, struktur og oversikt over selve prosjektet. Kommunikasjon mellom oppdragsgiver og prosjektmedlemmene foregår hovedsakelig via prosjektlederen. Prosjektlederen avtaler møter med oppdragsgiver, og finner tidspunkt slik at alle medlemmene kan være delaktige i møtene. Prosjektlederen vil også ha jevnlig kontakt med oppdragsgiver, for å demonstrere prosjektets progresjon. Prosjektlederen har også ansvar for å delegere ulike oppgaver til de resterende medlemmene om nødvendig. I likhet med de andre medlemmene, vil prosjektlederen være delaktig i selve kodingen i prosjektet.

Sekretær

Sekretæren har overordnet ansvar for kommunikasjon med veileder og dokumentering av møter. Testing og kvalitetsikring er også en del av sekretærens oppgave. Sekretæren tester kodeseksjoner etterhvert som enkeltoppgaver blir utført. Det vil deretter bli gitt tilbakemelding av arbeidet til det enkelte medlemmet. I tillegg til dette vil sekretæren, på lik linje med de andre teammedlemmene, være delaktig i selve kodingen.

Frontend ansvarlig

Frontend ansvarlig har et overordnet ansvar for frontend delen av prosjektet. Dette innebærer oversikt over design, brukergrensesnitt, funksjoner og koding av dette. Grunnen bak denne rollen er at hvis prosjektleder eller andre gruppemedlemmer lurer på noe angående frontend, skal frontend ansvarlig alltid vite svaret på de eventuelle spørsmål de skulle ha.

Backend ansvarlig

Det siste medlemmet har ansvar for backend, og implementering av dette i prosjektet. Dette inkluderer alt som har med databaser, server og sikkerhet. Grunnen bak denne rollen er at hvis prosjektleder eller andre gruppemedlemmer lurer på noe angående backend, skal backend ansvarlig alltid vite svaret på de eventuelle spørsmål de skulle ha.

3.1.5 Planlegging

Prosjektet er planlagt basert på innleveringsdato. Dette innebærer at prosjektet skal være klart for produksjonsmiljø før 20. mai. Vi startet med forprosjektrapport (Vedlegg A) og har prøvd å følge best mulig det som var planlagt.

Vi bruker en smidig arbeidsmetode, så resultater pr. leveranse vil bli satt ulikt for hver uke. Dette innebærer at man ikke kan fastslå noen spesifikke datoer.

1. Etter at prosjektplanen er ferdig og design er godkjent vil vi starte utvikling. Dato for innlevering av prosjektplan/forprosjekt er 31. Januar.
2. Vi vil starte med å lage en prototype av programmet for videre godkjenning av oppdragsgiver. Antar dato for ferdigstilling i løpet av Februar.
3. Når prototypen er godkjent vil vi starte utviklingsprosjektet for fullt. Dette innebærer å utvikle alle de ulike komponentene. Her vil ansvar bli delt ut basert på hvem som kan mest om de ulike komponentene. Vi forventer at komponentene er ferdige i løpet av April på grunn av at vi har eksamen i Mars.
4. Når komponentene er ferdige vil vi ha en godkjenning av oppdragsgiver. Vi endrer så på komponentene basert på tilbakemelding fra oppdragsgiver. Vi forventer at dette blir godkjent ikke lenge etter ferdigstilling av komponentene.
5. Vi lager ferdig unit tester og integrasjonstester før vi starter testfasen av systemet. Noe av dette kommer til å bli gjort imens vi lager systemet, så kan hende dette allerede er ferdig når vi har kommet til dette punktet. Forventet ferdig i midten av April.
6. Alpha-testing (testrunde 1). Her vil vi invitere venner og familie til å teste programmet og prøve å ødelegge det. Når testing er gjennomført sitter vi mest sannsynligvis igjen med mye informasjon som kan brukes til å forbedre systemet ytterligere. Forventet ferdig før Mai.
7. Beta-testing (testrunde 2). Dette skal være en testfase der vi forventer at det er få feil i systemet. Testrunde 2 kan gå løpende ut hele Mai.
8. Oppsett av system hos oppdragsgiver og eventuelt oppsett av noen devops metoder osv. Spørs veldig hva oppdragsgiver ønsker. Forventet ferdig i Mai.

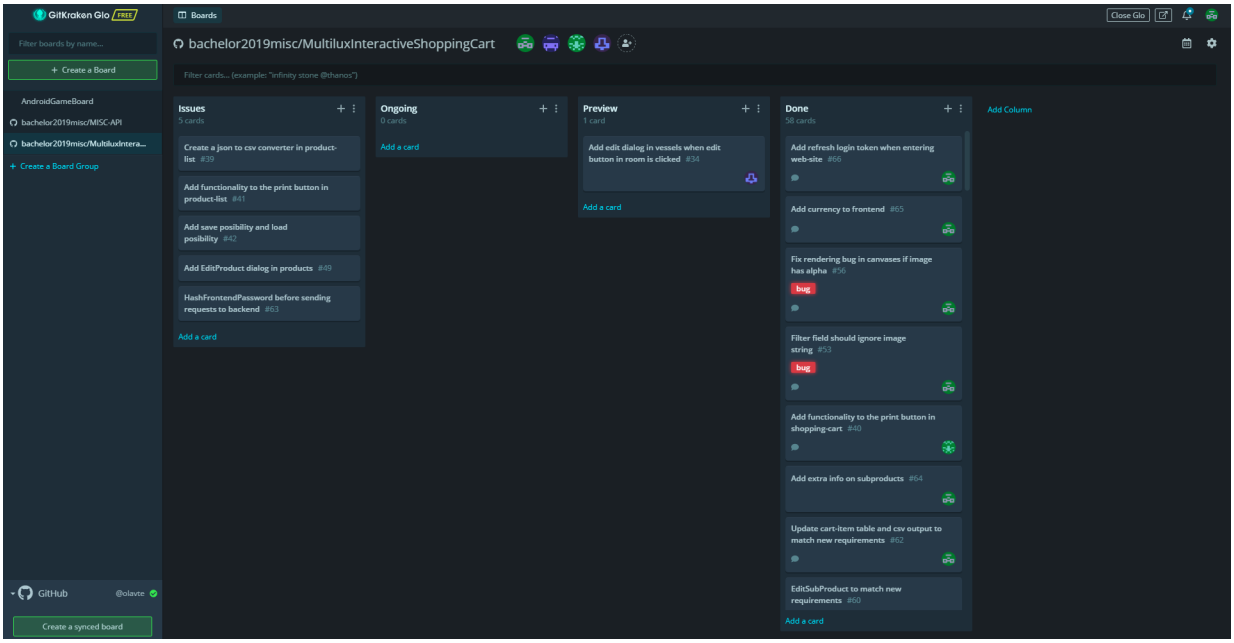
Nr.	Hovedaktivitet	Ansvar	Kostnad	Tid / omfang	Beskrivelse
1	Forprosjekt	Prosjektleder	0,-	~5d	Prosjektplan og oversikt
2	Prototype	Prosjektleder	0,-	~15d	Et prosjekt vi skal vise til oppdragsgiver
3	Server / database	Backend ansvarlig	0,-	~10d	Sette opp database
4	Frontend / design	Frontend ansvarlig	0,-	~10d	Design
5	Backend / logikk	Backend ansvarlig	0,-	~10d	Behandling av innsendt data opp mot database
6	Alpha	Prosjektleder	0,-	~10d	Testing
7	Beta	Prosjektleder	0,-	~10d	Testing
8	Oppsett hos oppdragsgiver	Prosjektleder og oppdragsgiver	Serverkostnader	~5d	Produksjon

Tabell 3.2: Planleggingstabell (Vedlegg A)

3.1.6 Utførelse

Utførelsen av prosjektet gjøres ved at prosjektleder og sekretær definerer "issues" som resten av prosjektgruppen kan plukke fra. Et issue inneholder en tittel som sier hva oppgaven er, beskrivelse på hva som skal gjøres og tasks. Dette er underoppgaver for å gi pekepinne på hvilken rekkefølge ting skal gjøres. I tillegg kan man legge ved et bilde eller fil med eksempel og eventuelt en kommentar ved spesielle tilfeller.

Vi bruker et issueboard med 4 kolonner. Det er delt opp i: issues, ongoing, preview og done. Du ser i figuren under hvordan det ser ut (Figur 3.1).



Figur 3.1: Glo Issueboard

Eksempel på hvordan dette utføres: Når prosjektleder oppretter et issue, oppretter han den i "issues". Et prosjektmedlem velger et issue, setter statusen på issuet til "ongoing" og oppretter en ny branch i Git fra master med issuet sitt id og navn. Da vet de andre i gruppen at den issuet er tatt og at de må velge noen av de andre. Når et issue er ferdig, settes issuet på preview. Dette gjør det slik at noen må se over og godkjenne koden din før issue branchen blir flettet inn i master. Når branchen er flettet inn i master settes issuet til "done".

3.2 Metode

3.2.1 Produktet

Produktet er organisert inn i tre ulike komponenter: frontend, backend og database. På de forskjellige komponentene er det en egen ansvarsrolle som deles ut til deltakerne i prosjektet. Dette innebærer et ansvar der prosjektleder kan forvente at ansvarspersonen har full kontroll over sitt ansvarsområde.

3.2.1.1 Frontend

Frontend er en web applikasjon. Vi bruker Angular rammeverket og nodeJS/npm [53] for kontrollering av avhengigheter.

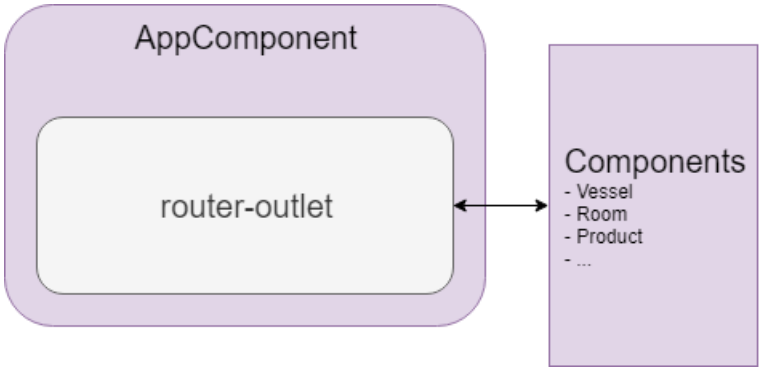
Angular Kort forklart er Angular et rammeverk i JavaScript for å lage applikasjoner, i hovedsak på web. Angular, eller Angular2+ som det også kalles, er en omskriving av AngularJS fra grunnen av. Det er utviklet av Google og hadde sin første utgivelse 14. september 2016. Forskjellen mellom Angular2+ og AngularJS er i hovedsak at Angular2+ har komponentbasert struktur. [55]

TypeScript "TypeScript is JavaScript for application-scale development."[52]. Dette vil si at TypeScript er enkelt forklart JavaScript med litt ekstra egenskaper. Det er er laget av Anders Hejlsberg hos Microsoft, som designet C#. [52]

Bakgrunnen bak TypeScript er at JavaScript alltid har hatt problemer når man skal skalere det til veldig store applikasjoner. Dette kommer av at JavaScript mangler egenskaper som objektorientering, sjekking av datatype og sjekk av feil ved kompilering. Ved bruk av TypeScript får vi disse egenskapene. Vi må definere datatyper, har mulighet for å programmere objektorientert og språket blir kompilert før start for å sjekke etter feil. Dette gir JavaScript utviklere muligheten til å lage store applikasjoner. [52]

Angular Router Angular Router er en del av Angular plattformen. Den tillater utviklere å bygge enkeltside applikasjoner med forskjellige visningsvinduer. Den gjør dette ved å bruke ruten eller url som en indikasjon på hvor brukeren er. [8]

Router-outlet Router-outlet er et tomt element som Angular fyller dynamisk basert på gjeldende rutetilstand. Dette gir muligheten for å lage en enkeltside applikasjon med dynamisk innhold. [32]



Figur 3.2: Router-outlet er i AppComponent og kan bytte komponenter basert på URL-rute

Angular material Dette blir brukt for design og layout i frontend. Angular material er et UI komponentbibliotek som er lagd basert på designstandarder. Siden ingen i prosjektet er profesjonelle designere vil et standardisert bibliotek føre til tidsbesparelser og muligheten for å nå en bredere målgruppe. [23]

HttpClient HttpClient er et bibliotek som simplifiserer HTTP kall til REST API. HttpClient blir sett på som standarden til Angular når du skal gjøre noe relatert til REST. Fordelen for oss ved å bruke denne er at den er strukturelt mer ryddig og krever mindre linjer med koder. Dette gir oss bedre kontroll når der er flere REST kall som skal gjøres asynkront samtidig. [22]

Electron Brukes for å kunne gjøre en Angular web applikasjon om til en .exe applikasjon. [13]

Json2Csv Dette er et bibliotek som konverterer JSON til CSV. [63]

Jspdf Jspdf er et bibilotek som konverterer JSON til PDF. [26]

3.2.1.2 Backend

Backend består av et Node.js RESTful API. Node.js blir brukt til å gjøre en avhengighet opp imot Express. Vi bruker Express avhengigheten til å sette opp en Express server der vi kan definere REST endpoints.

Express ble valgt fordi det er et populært rammeverk til utvikling av serverside-programvare for Node.js plattformen, og i dette tilfellet for å utvikle et tilstandsløst HTTP-API.

Oversikt over ulike bibliotek og definisjoner i prosjektet:

Models En modellfil blir brukt for å representere en tabell i programmet. [14]

Migration En migrasjonsfil blir brukt for å få oversikt over endringer på en tabell i en database. Man definerer hvilke felt som skal være med i tabellen og hvilke datatyper de skal ha. Deretter kan man migrere disse endringene til databasen. Man har også muligheten til å reversere endringene i databasen i etterkant om det er definert i migrasjonsfilen. [14]

Config Konfigurasjonsfilen skal definere hvordan CLI skal koble til en database. Den skal inneholde autentiseringsinformasjon for databasen som den skal kobles til og hvilken dialekt den bruker. [14]

Controllers Controller filene blir brukt til å definere CRUD funksjoner for de forskjellige modellene. Dette innebærer oppretting av funksjoner for å opprette, lese, oppdatere og slette innhold i en tabell. [14]

Bcrypt Bcrypt er et bibliotek vi bruker for å hashe passord til brukere før de blir lagt i databasen. Dette ble brukt fordi det populært og bruker en god metode for å hashe. [37]

Passport Dette er en middelvare for autentisering som er mye brukt i Node.js. Dette bruker vi slik at kun autentiserte brukere kan endre på innhold i databasen. [48]

Nodemon Nodemon er et verktøy vi har tatt i bruk for å kunne automatisk restarte Node.js programmet når det skjer endringer i koden. [44]

3.2.2 Database

Databasen vi har tatt i bruk er en PostgreSQL database [25]. Den blir opprettet på port 5432 på serveren og det trengs et brukernavn og passord for å få tilgang. Alle kall vi gjør til database er med Express sine integrerte spørringer, men det er fortsatt mulig å gjøre ordinære SQL spørringer til databasen hvis det skulle være nødvendig.

3.3 Materialer

3.3.1 Styringshjelpemidler

Glo brukes som et issue tracking system. Her kan du definere kolonner og lage markeringer selv. Prosjektleder kan definere oppgaver som må gjøres, sette antatt arbeidstid osv. Det er fire kolonner som forteller noe om statusen til "issuen" (Figur 3.1).

3.3.2 Utviklingshjelpemidler

Git, GitKraken Vi bruker Git for versjonskontroll og GitKraken som git klient. Grunnen for bruk av GitKraken, er at de fleste fra prosjektgruppen har erfaring med det og vi slipper derfor å bruke mye tid på opplæring i bruk av denne klienten. [7]

IDE, VS Code Vi bruker VS Code fordi løsningen vår er en statisk web app som bruker Node.js for avhengigheter. VS Code er en veldig simpel, men kraftig IDE. Det er store muligheter for utvidelser i VS Code, som for eksempel plugin for PostgreSQL interface. [35]

PgAdmin PgAdmin er et grafisk verktøy for å administrere PostgreSQL databaser. Vi valgte dette fordi det er den mest populære og funksjonsrike open source platformen for PostgreSQL. [38]

GitHub GitHub brukes for å ha versjonskontroll gratis på nettet. GitHub brukes ofte av arbeidsgivere for å se hvilke prosjekt arbeidstaker har deltatt i. Vi gikk for denne siden vi er snart ferdig med studie og skal snart søke jobb. [19]

ARC Dette er en REST klient for testing av rest endpoints. Den gjør det mulig å teste endpoints uten at kallene er implementert i frontend. [2]

WinSCP WinSCP er et program for overføring av filer mellom to enheter. Dette ble brukt for å overføre filer fra en lokal datamaskin til serveren. [62]

4 Resultater

Figur 4.1: MISC — Multilux Interactive Shopping Cart

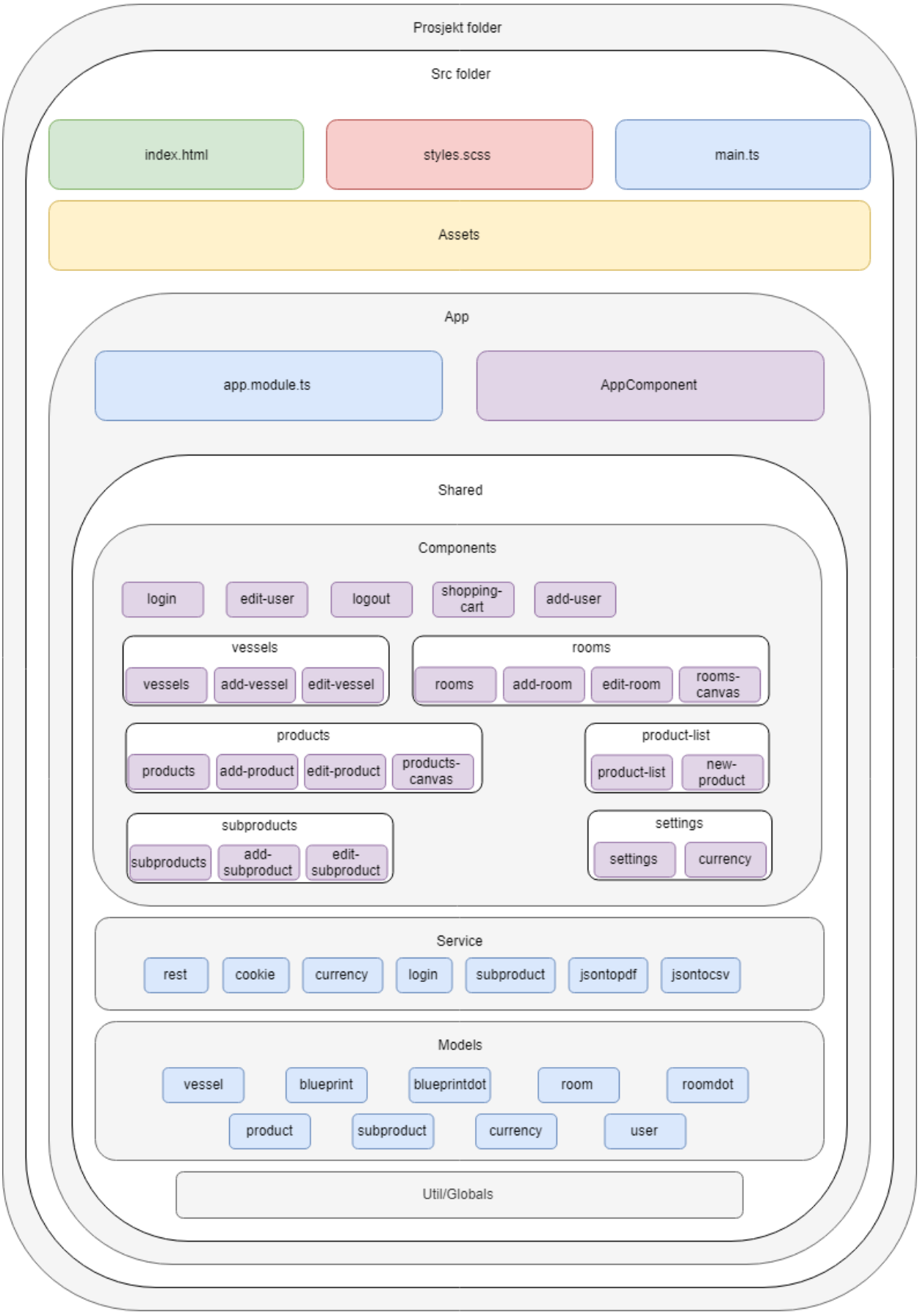
4.1 Struktur

4.1.1 Frontend

4.1.1.1 Filstruktur

Filstrukturen til prosjektet følger standard struktur for Angular prosjekter. Det som særpreger denne strukturen, er at den har mapper som er organisert basert på formålet til koden. Derfor har man en egen mappe for:

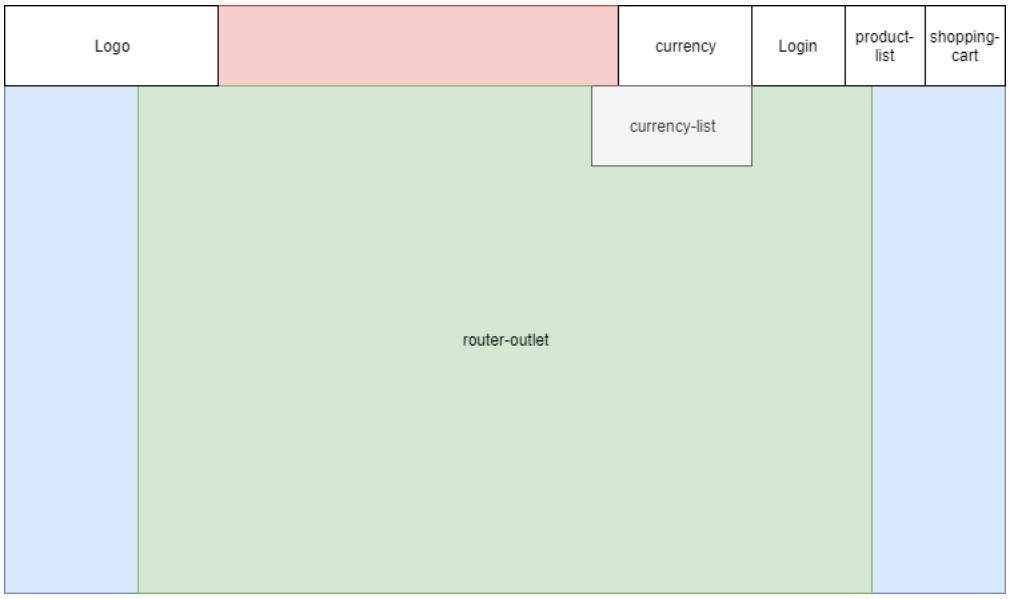
- Assets (bilder og eventuelle ressurser du trenger å ha med).
- Source.
    - Komponenter.
    - Servicer.
    - Models.
    - Utils.



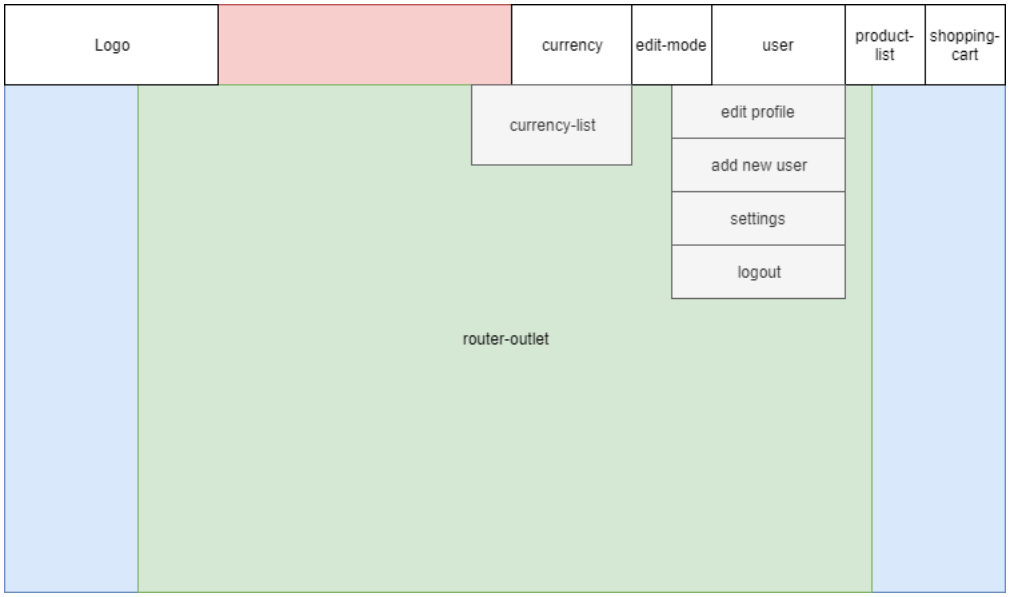
Figur 4.2: Frontend filstruktur

4.1.1.2 HTML-dokumentets struktur

Index.html består alltid av to komponenter. Dette er "AppComponent" og komponenten som ligger i router-outlet (3.2.1.1 side 27). Dette betyr at alt som er konstant bør ligge i AppComponent og det som skal være modulært ligger i router-outlet. Figur 4.3 og 4.4 under viser hvordan dokumentet er strukturert. Blå er body, grønn er router-outlet/hovedseksjonen, rød er header og grå symboliserer at det er skjult.



Figur 4.3: Den generelle strukturen er for dokumentet når man ikke er innlogget



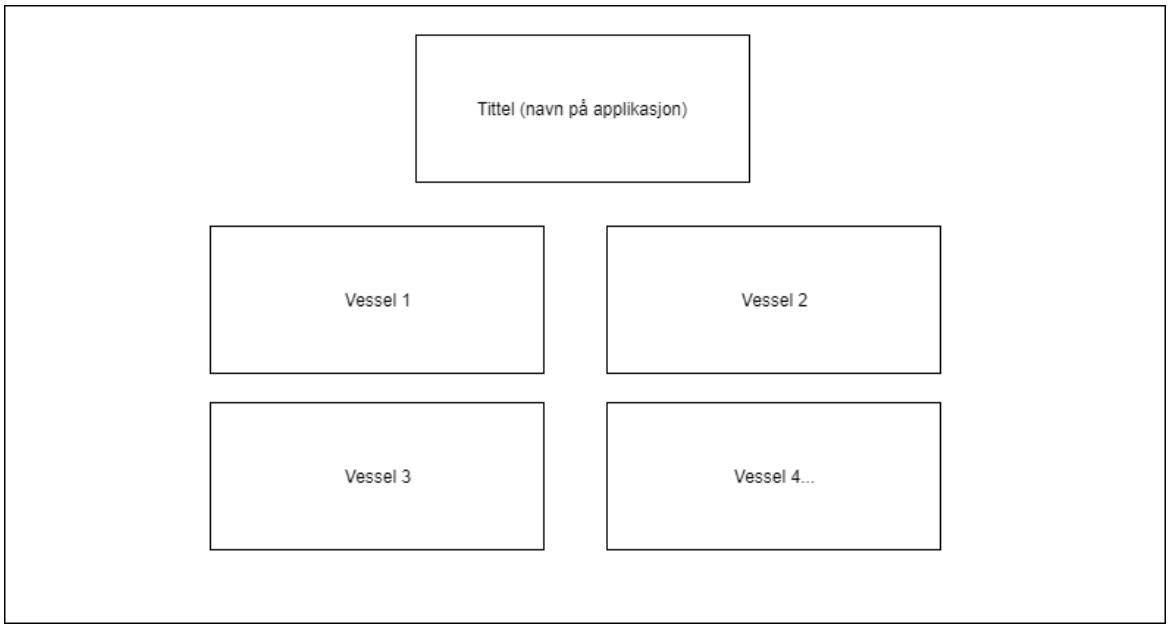
Figur 4.4: Den generelle strukturen er for dokumentet når man er innlogget

Ved å utvide antall menyer når du er innlogget (Figur 4.4) unngår vi flere undermenyer. Dette er god bruk av Steering loven (2.2.2 side 13).

4.1.1.3 Hovedseksjonen

Hovedseksjonen som er en router-outlet har i hovedsak fire forskjellige designvariasjoner:

Variasjon 1



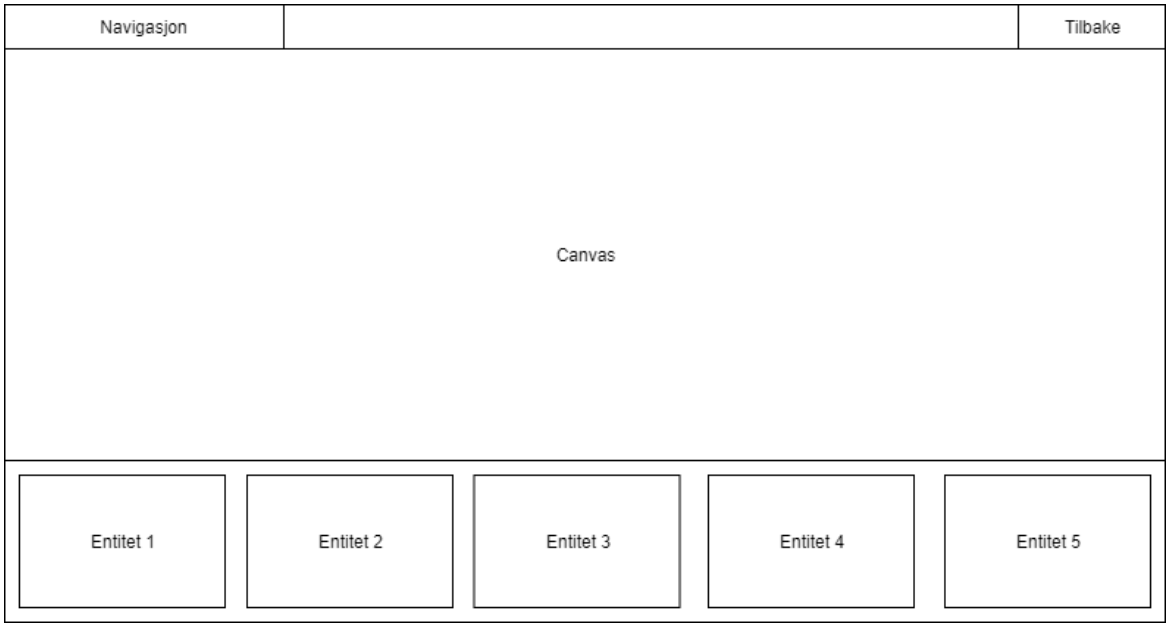
Figur 4.5: Fremsiden, hvor man også velger fartøy

Denne variasjonen dekker hovedsiden og standardsiden. Det vil si at hvis du går til en rute som ikke er definert kommer du hit. Symbolet for alle udefinerte ruter er: **.

Rutene for denne variasjonen er:

```
/
/vessel
/**
```

Variasjon 2



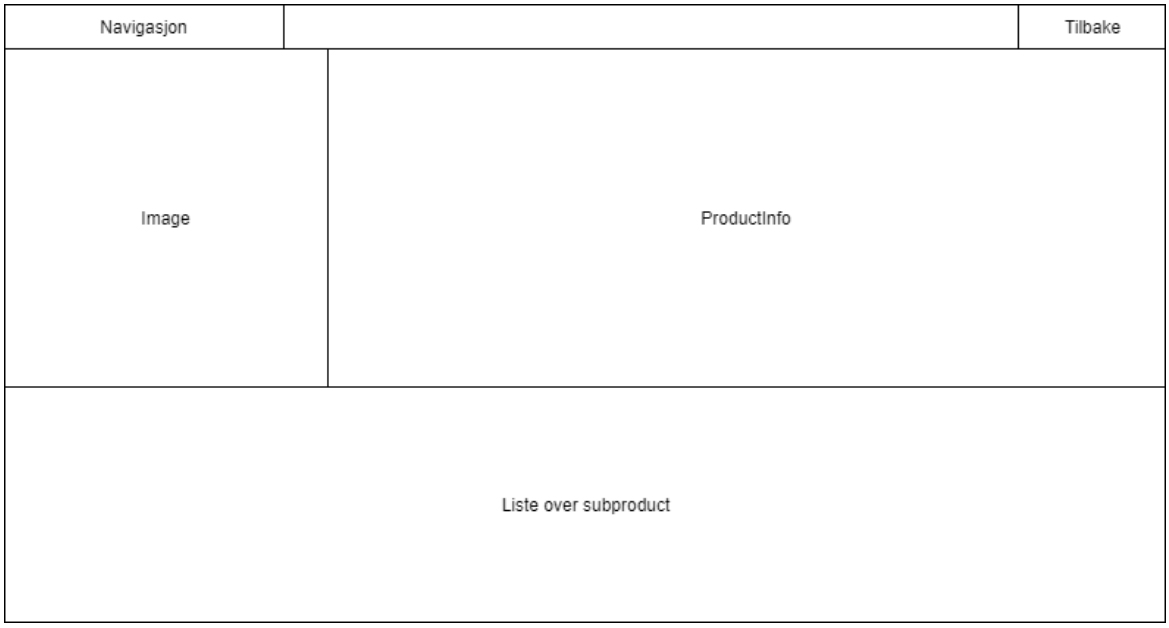
Figur 4.6: Side hvor du har en canvas med flyttbare dotter

Denne variasjonen dekker de sidene i veiviseren som trenger å ha noen trykkbare prikker på et bilde. Det skal også være mulighet for å trykke på elementet til prikkene under.

Rutene for denne variasjonen er:

```
/rooms
/products
```

Variasjon 3



Figur 4.7: Siden der man kan se underproduktene. Disse er elementene som skal i handlelisten.

Denne variasjonen dekker siden for oversikt over underprodukt til et produkt. Denne variasjonen har også tre variasjoner. Forskjellen på disse er navigasjonsindikatoren oppe i venstre hjørne, som endrer seg basert på hvor du er.

Rutene for denne variasjonen er:

```
/subproducts
/product
```

Hvis den blir hentet inn uten rute, forsvinner navigasjon og tilbake knappene.

Variasjon 4

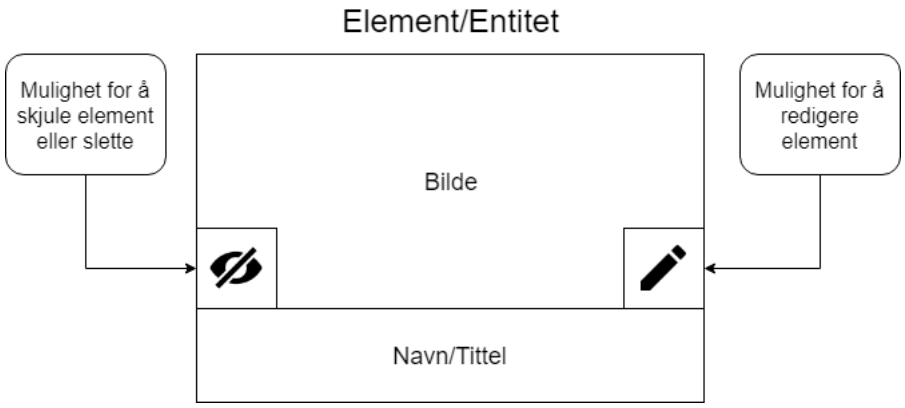
Figur 4.8: Side for visning av liste

Denne variasjonen er for de komponentene som skal vise en liste. Design layouten er i starten den samme, men de har helt ulike komponenter, så selve innholdet er helt ulikt.

Rutene for denne variasjonen er:

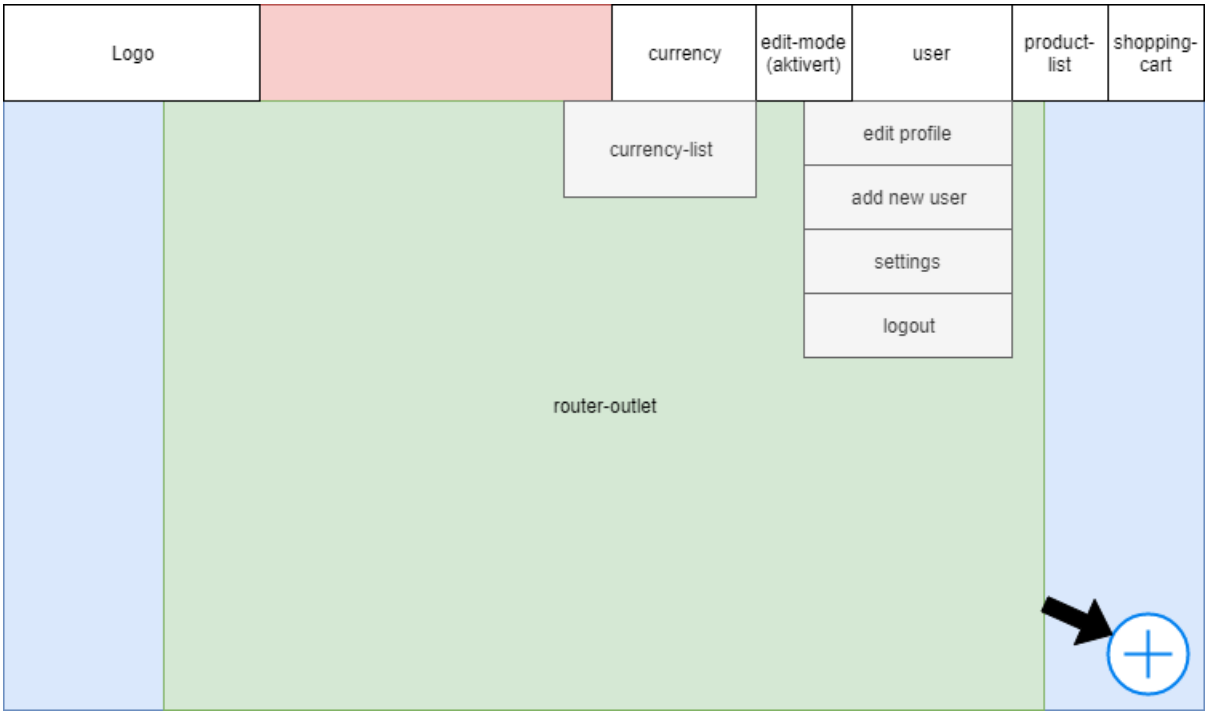
```
/product-list
/shopping-cart
```

4.1.1.4 Redigeringsmodus



Figur 4.9: Entiteter i redigeringsmodus

Vi plasserer mulighet for å redigere elementer og entiteter der du finner dem, så når administrator aktiverer redigeringsmodus vil to nye ikon dukke opp på dem. Ved å gjøre dette følger vi teorien for bra design der vi forutser hva kunden er ute etter (2.2.1 side 11) og plasserer knappene nært slik det også følger Fitts lov (2.2.2 side 12).



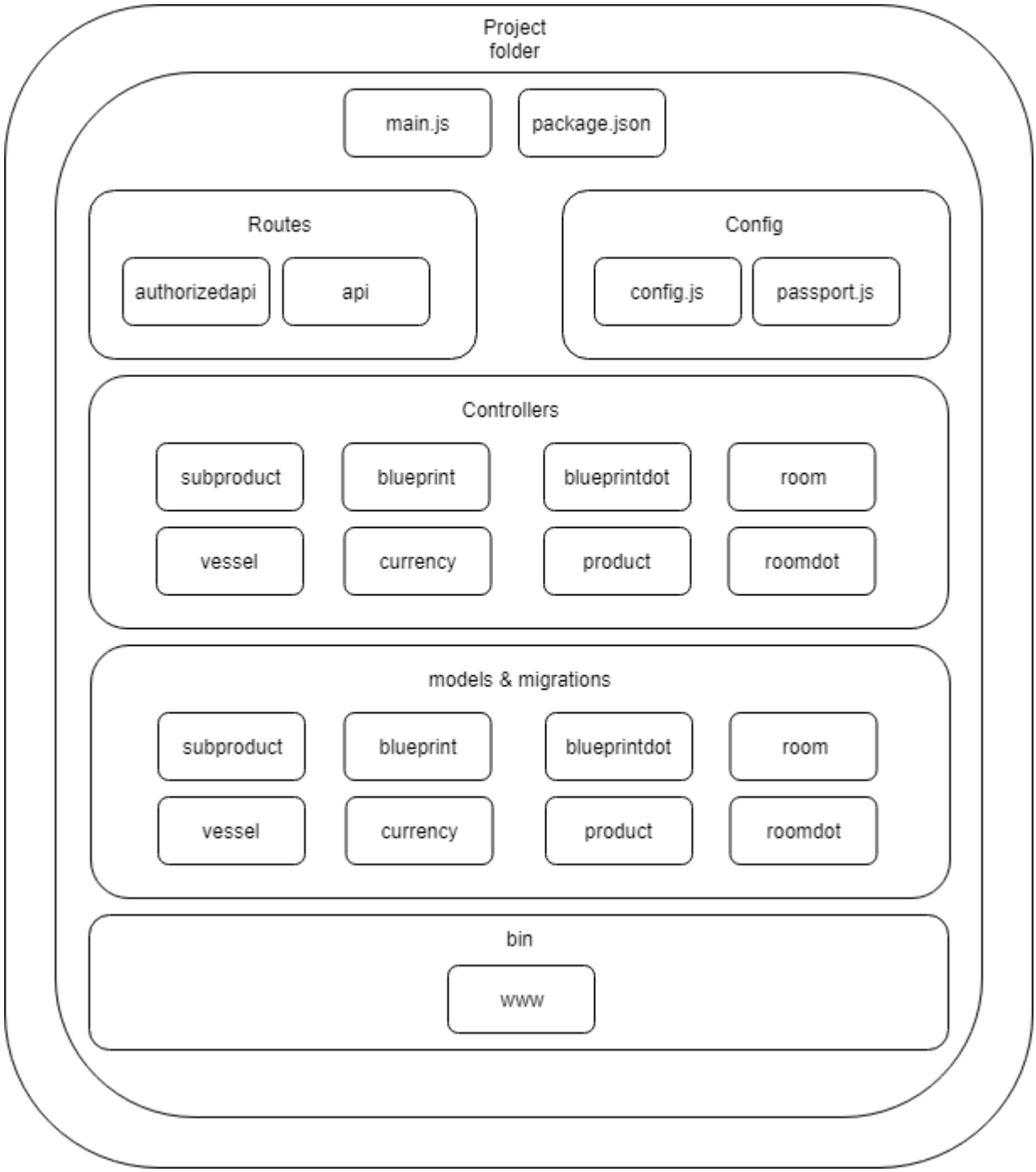
Figur 4.10: FAB har dukket opp i redigeringsmodus av siden

Når administrator aktiverer redigeringsmodus, vil det dukke opp noen ekstra elementer. Det mest konsistente elementet som vil dukke opp er en Floating Action Button (FAB) nede til høyre (Figur 4.10). Denne knappen har alltid som funksjon å legge til et nytt element der du er. Dette gjør at prosessen i å legge til noe nytt er konsistent og vil gjøre det lettere for nye brukere å lære systemet (2.2.1 side 11).

4.1.2 Backend

4.1.2.1 Filstruktur

Filstrukturen ble autogenerert av Express og dannet grunnstrukturen til backend applikasjonen. Den ble deretter endret på for å passe til vårt bruk.



Figur 4.11: Backend filstruktur

Config Denne mappen inneholder av config.js og passport.js.

config.js inneholder informasjon om hvordan Express skal koble til databasen, og passport.js inneholder JWT oppsettet for backend.

Routes Denne mappen inneholder authorizedapi.js og api.js.

authorizedapi.js inneholder API metoder som er satt opp med JWT (2.4.2 side 18) og er adskilt fra api.js da de inneholder API metoder som ikke krever JWT og kan aksesseres av alle.

Controllers Controllers mappen inneholder filer for alle database modellene.

Hver fil inneholder API metoder av typen GET (getbyid, getall osv.) for sin modell. Dette er fordi alle skal kunne hente informasjon fra databasen men kun autoriserte brukere skal kunne ha mulighet til å endre og slette informasjon i databasen.

Models & migration Disse er to ulike mapper, men inneholder de samme filene og er derfor lagt sammen i diagrammet. Models definerer hver databasemodell i egne filer med attributter og datatyper. Filene i migrations definerer også hver databasemodell og skal stemme overens med samsvarande fil i models folderen.

Man må gjør det slik fordi rammeverket vil at det skal gjøres på denne måten. Forskjellen er at filene i migrations blir brukt til å migrere modellene til databasen. Man har også muligheten til å reversere endringene i databasen i etterkant om det er definert i migration filen.

Bin Bin mappen består av en fil kalt www. Denne definerer og starter opp HTTP-serveren. Den skal lytte på hendelser, slik som klienter som prøver å hente informasjon fra serveren.

Main.js Denne filen importerer Express rammeverket inn i applikasjonen og API'ene som har blitt opprettet. Den definerer også feilhåndtering for applikasjonen og annen middelvare som trengs for at applikasjonen skal fungere.

Package.json Package.json fungerer som et manifest for applikasjonen, og inneholder ulike avhengigheter og pakker som blir brukt i prosjektet.

4.1.2.2 API

API'et til MISC er bygd rundt de forskjellige typene av forespørsler. Vi prøver å holde selve endepunktet likt, men at forespørselen skiller dem. (2.5 side 19)

Dette betyr da at:

- GET — hente data fra databasen.
- POST — legge til data i databasen.
- PUT — redigere data i databasen.

Dette gjør at det blir mye lettere for de som er ansvarlig for frontend å anta hvilke endepunkt som finnes i API.

User

Endpoint	Type	Beskrivelse
/add	POST	Legger til en ny bruker.
/user/:username	PUT	Finner bruker og oppdaterer brukernavn/passord.
/login	POST	Sjekker om brukeren eksisterer og om passord er riktig. Om passordet er riktig og brukeren eksisterer vil den sende tilbake en JWT. Denne blir etterpå lagt inn i REST kall på frontend for at brukere skal kunne bruke kall som er kun for autoriserte brukere.
/jwtrefresh	GET	Sender tilbake en ny JWT. Denne vil bli brukt når den eksisterende JWT holder på å gå ut.

Tabell 4.1: User REST kall

Vessel

Endpoint	Type	Beskrivelse
/vessel	GET	Henter alle vessels i databasen.
/vessel	POST	Oppretter først en blueprint, deretter oppretter et vessel der idBlueprint attributtet til vessel blir satt til id'en til blueprinten som akkurat ble opprettet.
/vessel/:id	GET	Henter et vessel basert på id.
/vessel/:id	PUT	Finner et vessel basert på id og deretter oppdaterer den.
/login	DELETE	Finner et vessel basert på id og deretter tar en *soft delete* av den. Dette innebærer at *deletedAt* attributten får verdien til det tidspunktet når det ble slettet, og man ser ikke det spesifikke vesselet lenger om man prøver å hente det. Det eksisterer fortsatt i databasen om det skal gjenopprettes.

Tabell 4.2: Vessel REST kall

Blueprint

Endpoint	Type	Beskrivelse
/blueprint	GET	Henter alle blueprints i databasen.
/blueprint/:id	GET	Henter et blueprint basert på id.
/blueprint/:id	PUT	Finner en blueprint basert på id og deretter oppdaterer den.
/blueprintbyvesselid/:id	GET	Finner først et vessel basert på id og så finner et blueprint der id'en til blueprinten er den samme som blueprint id attributtet som er oppgitt i vessel.

Tabell 4.3: Blueprint REST kall

Blueprintdot

Endpoint	Type	Beskrivelse
/blueprintdot	GET	Henter alle blueprintdots i databasen.
/blueprintdot	POST	Oppretter først et nytt room og deretter oppretter blueprintdot der idRoom attributtet blir satt til id'en til room'et som akkurat ble opprettet.
/blueprintdot/:id	GET	Henter en blueprintdot basert på id.
/blueprintdot/:id	PUT	Finner en blueprintdot basert på id og deretter oppdaterer den.
/blueprintdotbyidroom/:id	GET	Finner først et room basert på id, deretter finner alle blueprintdots der *idRoom* attributtet til blueprintdot er oppgitt i forespørselen.
/blueprintdotbyidvessel/:id	GET	Finner først et vessel basert på id, deretter finner alle blueprintdots hvor idVessel attributtet til blueprintdot er oppgitt i forespørselen.

Tabell 4.4: Blueprintdot REST kall

Room

Endpoint	Type	Beskrivelse
/room	GET	Henter alle rooms i databasen.
/room/:id	GET	Henter et room basert på id.
/room/:id	PUT	Finner et room basert på id og deretter oppdaterer den.

Tabell 4.5: Room REST kall

Roomdot

Endpoint	Type	Beskrivelse
/roomdot	GET	Henter alle roomdots i databasen.
/roomdot	POST	Legger til et ny roomdot.
/roomdot/:id	GET	Henter en roomdot basert på id.
/roomdot/:id	PUT	Finner en roomdot basert på id og deretter oppdaterer den.
/roomdot/:id	DELETE	Finner en roomdot basert på id og deretter sletter den.
/roomdotbyidroom/:id	GET	Finner først et room basert på id, deretter finner alle roomdots der *idRoom* attributtet til roomdot er oppgitt i forespørselen.

Tabell 4.6: Roomdot REST kall

Product

Endpoint	Type	Beskrivelse
/product	GET	Henter alle products i databasen.
/product	POST	Legger til et nytt product.
/product/:id	GET	Henter et product basert på id.
/product/:id	PUT	Finner et product basert på id og deretter oppdaterer den.
/product/:id	DELETE	Finner et product basert på id og deretter sletter den fra databasen.

Tabell 4.7: Product REST kall

Subproduct

Endpoint	Type	Beskrivelse
/subproduct	GET	Henter alle subproducts i databasen.
/subproduct	POST	Legger til et nytt subproduct.
/subproduct/:id	GET	Henter et subproduct basert på id.
/subproduct/:id	PUT	Finner et subproduct basert på id og deretter oppdaterer den.
/subproductbyidproduct/:id	GET	Finner først et product basert på id, deretter finner et subproduct der idProduct attributtet til subproduct er oppgitt i forespørselen.

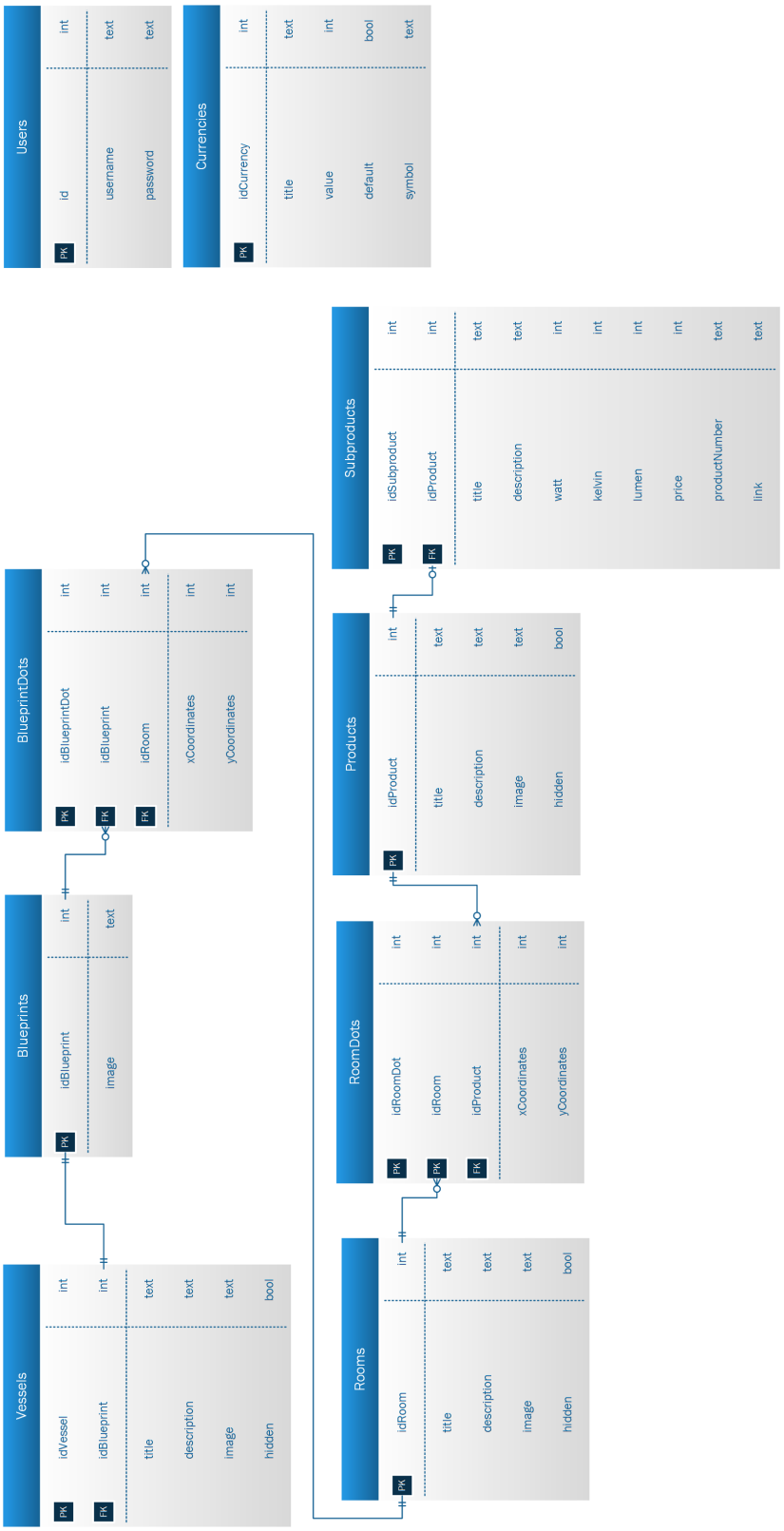
Tabell 4.8: Subproduct REST kall

Currency

Endpoint	Type	Beskrivelse
/currency	GET	Henter alle currencies i databasen.
/currency	POST	Legger til en ny currency.
/currency/:id	GET	Henter en currency basert på id.
/currency/:id	PUT	Finner en currency basert på id og deretter oppdaterer den.
/currency/:id	DELETE	Finner en currency basert på id og deretter sletter den fra databasen.

Tabell 4.9: Currency REST kall

4.1.3 Databasemodell

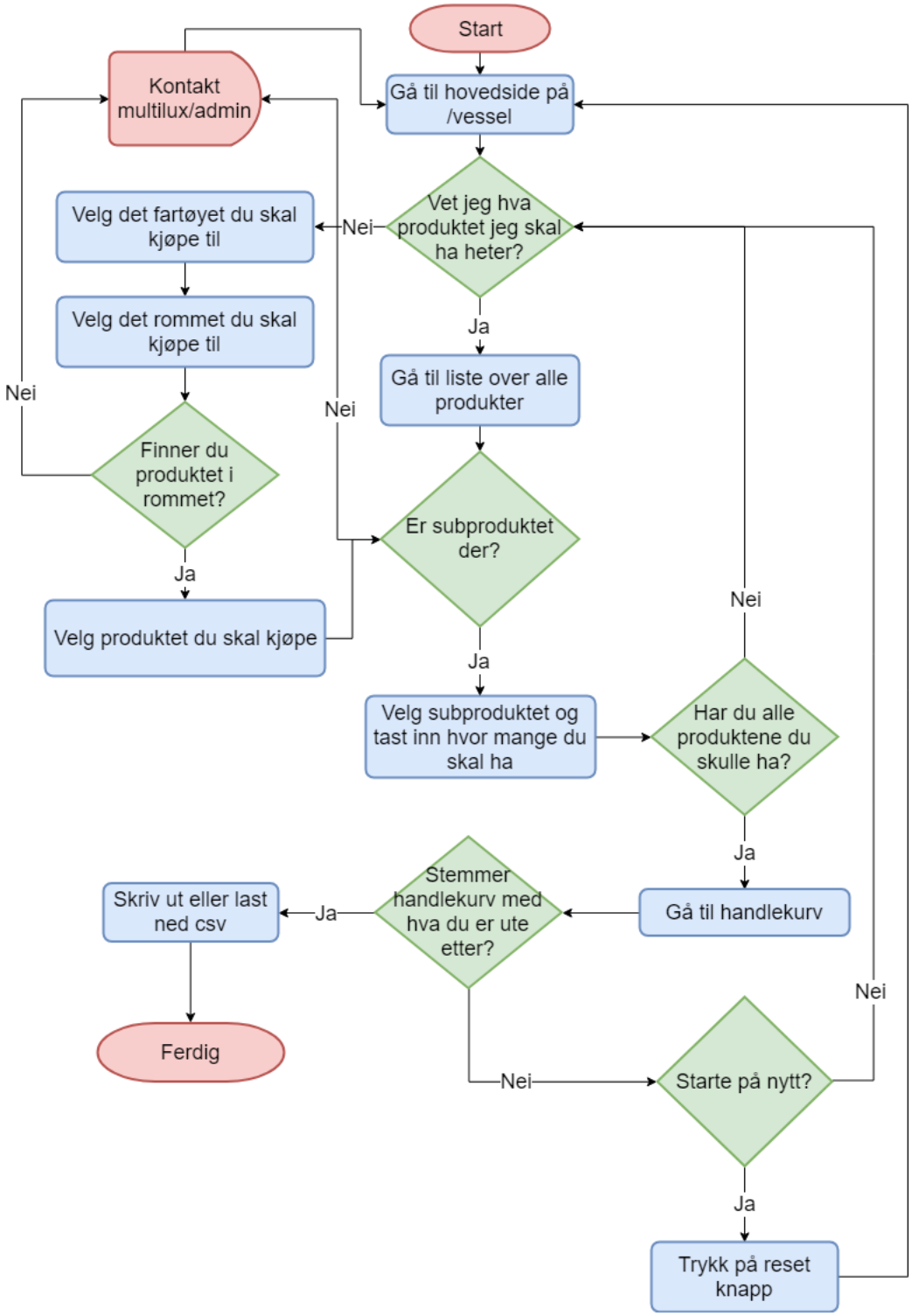


Figur 4.12: Database ER diagram

Databasen er basert på hvilke modeller vi bruker. Særpreget med denne strukturen er hvordan alle tabellene til venstre er koblet sammen. Dette er fordi kravet til oppdragsgiver var å ha en slags veiviser-struktur (2.2.4 side 15), hvor vi tar kunden gjennom alle de forskjellige tabellene steg for steg (Vedlegg B).

De fleste forholdene mellom tabellene er en-til-mange eller mange-til-en, men vi har to tabeller for mange-til-mange forhold (2.7.1 side 20). Dette er tabellene: blueprintdot og roomdot. Dette er egne tabeller, men fungerer egentlig i hovedsak som en forholdstabell mellom blueprint til room og room til product.

4.2 Konseptmodell



Figur 4.13: Flytskjema over systemet

4.3 Interaksjonsdesign og grafisk brukergrensesnitt

Det grafiske brukergrensesnittet vil bli presentert fra to ulike perspektiver: administrator og kunde. Dette for å belyse brukergrensesnittet på en strukturert og oversiktlig måte. Skjermbilder som blir vist er tatt fra versjon 0.9.2 -beta av applikasjonen.

4.3.1 Administrativt perspektiv

Det første underkapittelet vil demonstrere brukergrensesnittet fra et administrativt perspektiv.

4.3.1.1 Navigasjonsbar

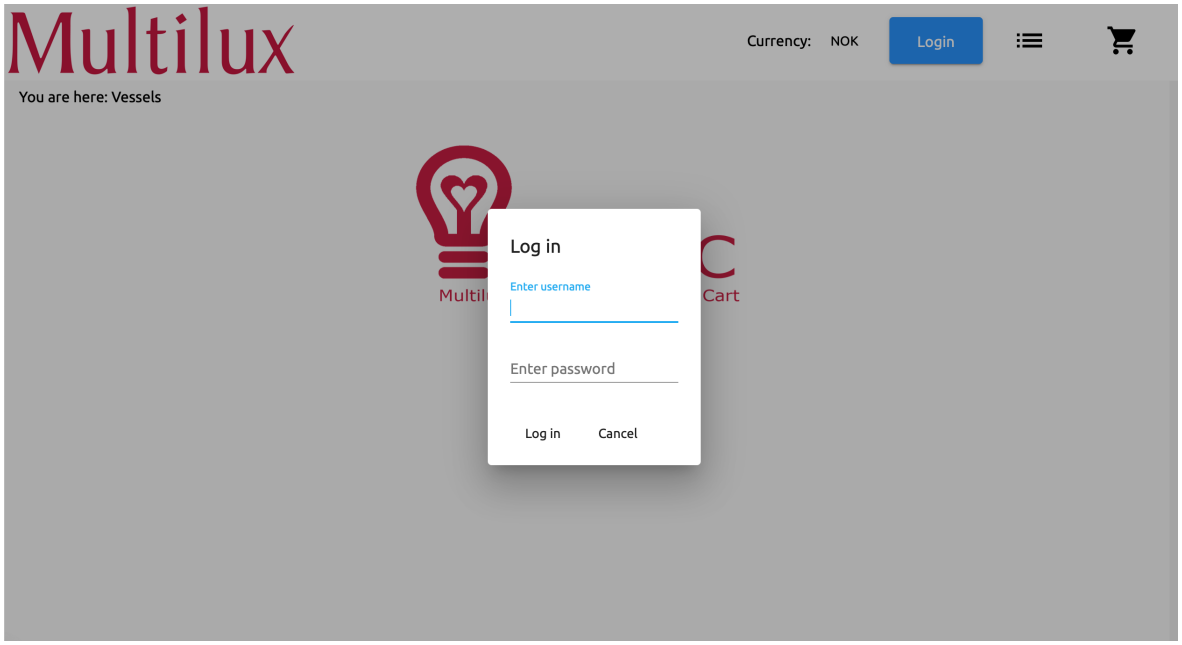
Denne delen tar for seg navigasjonsbaren. Navigasjonsbaren er festet til applikasjonen, hvor kun innholdet i hoved seksjonen oppdateres når brukeren navigerer seg gjennom web applikasjonen. Menyen består av flere valgmuligheter, samt logoen til oppdragsgiver. For å skape balanse og struktur i web applikasjonen var det et bevisst designvalg å ha menyen festet gjennom hele webapplikasjonen. Dette resulterer i at kun innholdet i hovedseksjonen (router-outlet komponenten) oppdaterer seg.

Figur 4.14: Navigasjonsbaren i web applikasjonen

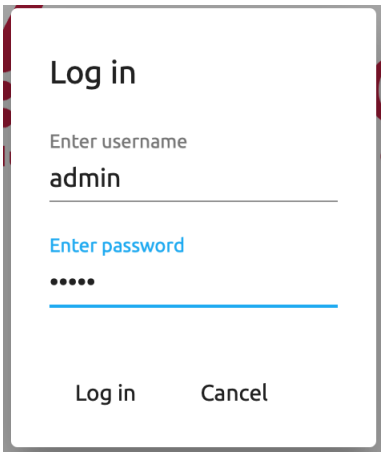
Ovenfor er et skjermbilde av navigasjonsbaren uten innlogget bruker. Navigasjonsbaren inneholder 5 elementer og vises likt fra begge perspektivene.

Logoen (herunder Multilux), fungerer som en hjem knapp og er vedlagt som et bilde. Brukeren kan dermed klikke på denne logoen for å omdirigere seg tilbake til forsiden. “Currency” gir mulighet for brukerne å bytte mellom ulike valutaer som er lagt inn av administrative brukere. “Login” komponenten er kun ment for administrative brukere. Poenget er å la de autoriserte brukerne logge seg inn og legge til blant annet fartøy, produkter, rom og valuta. De to knappene helt til høyre lar kunden gå henholdsvis til en liste over tilgjengelige produkter og handlekurven.

4.3.1.2 Innlogging



Figur 4.15: En dialog boks med login komponenten

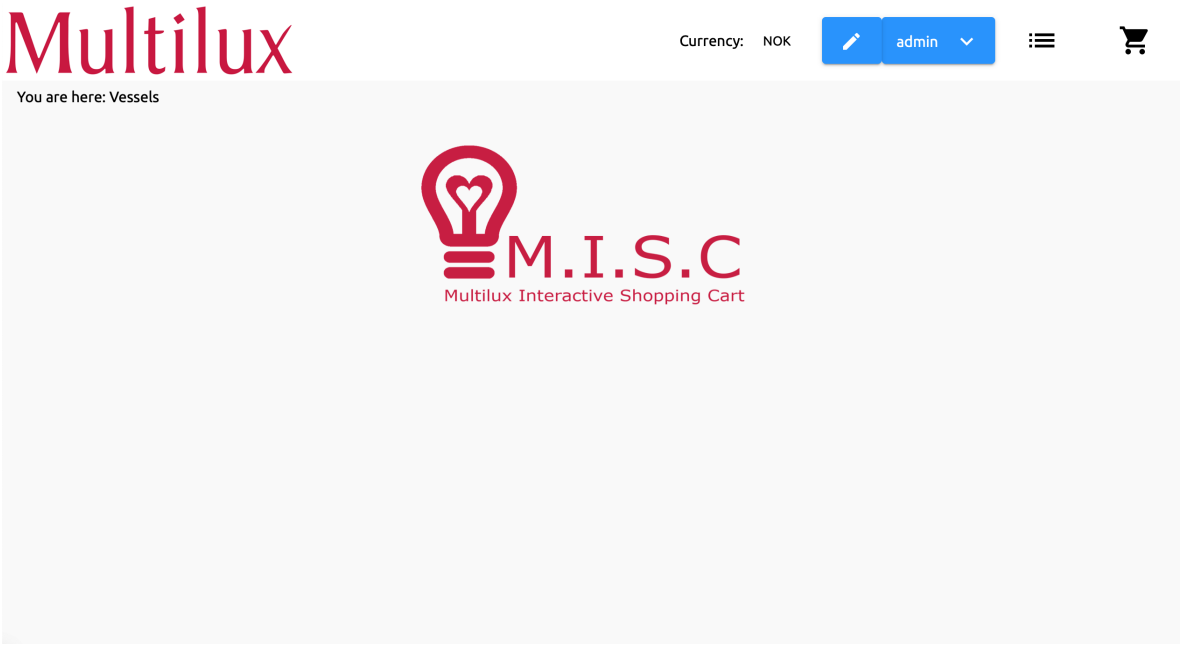


Figur 4.16: Testbruker med brukernavn admin

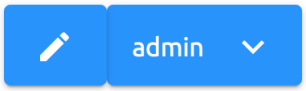
Figur 4.15 og 4.16 ovenfor demonstrerer "login" komponenten. Som vist er det dukket opp et lite vindu i hoved seksjonen. Det er laget en test bruker, her ved navn "admin" og et skjult passord. Etter man trykker på "Log in" henter den det du har skrevet og legger det i JSON som har verdiene username og password. Dette JSON objektet blir sendt til backend med HTTP.

Når backend får forespørselen vil den sjekke at brukeren eksisterer. Om den eksisterer går den videre for å sjekke at passordet er korrekt. Om passordet stemmer vil den opprette en JWT og signere denne, for så å sende tilbake JWTen til brukeren.

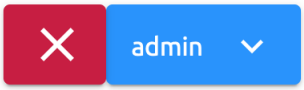
JWTen vil videre bli lagret i en cookie for at frontend skal kunne ta med denne i fremtidige forespørsler til backend. Da vet backend at det er en autorisert forespørsel, og vil godkjenne tilgang til autoriserte metodekall.



Figur 4.17: Forsiden etter at administrator er logget inn



Figur 4.18: Vanlig modus

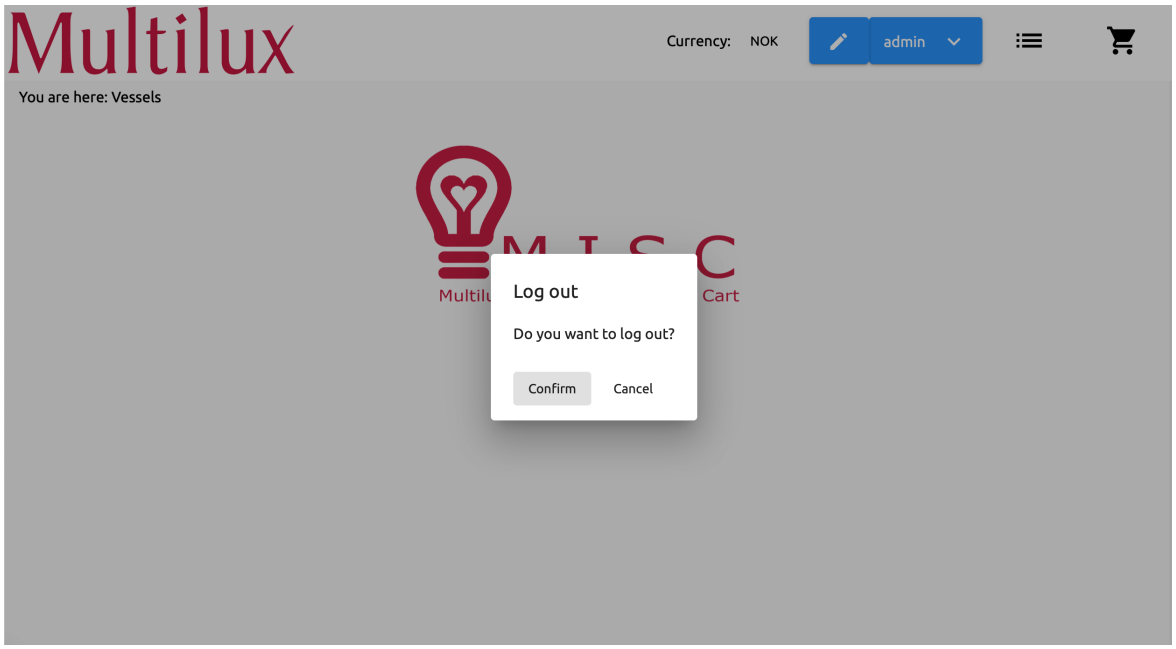


Figur 4.19: Redigeringsmodus

Figurene ovenfor demonstrer de to modusene som er tilgjengelige i web applikasjonen etter at administratoren er logget inn. Den første viser den vanlige modusen (Figur 4.18). Ved å bruke denne modusen har administratoren mulighet til å se hvordan web applikasjonen vil fremstå for kunden. Den andre viser redigeringsmodusen (Figur 4.19). Denne modusen tillater brukeren å gjøre diverse endringer og legge til innhold.

Figur 4.20: “Drop-down” menyen

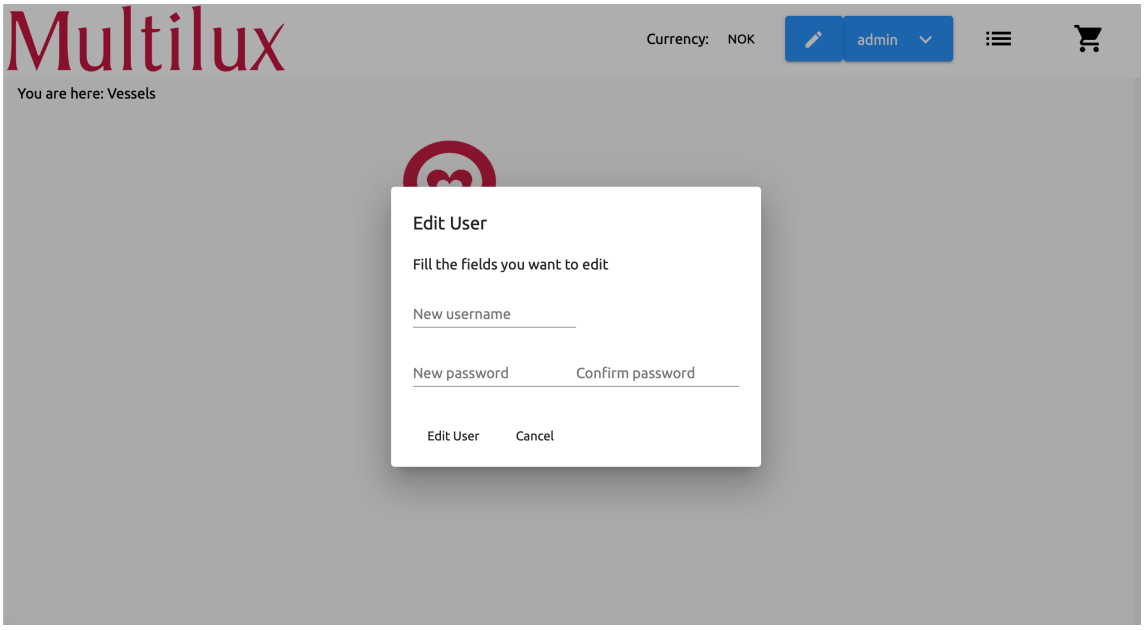
På skjermbildet over vises et utdrag av navigasjonsbaren etter at brukeren har klikket på brukernavnet. Dette vil vises uavhengig av modusen. Drop-down menyen inneholder fire valgalternativer. Disse alternativene er rediger bruker, legge til flere brukere, innstillinger og logge ut.



Figur 4.21: Dialogboks for utlogging

Ovenfor vises dialogen for å logge ut. Den består hovedsakelig av to alternativer, “Cancel”, som avbryter handlingen og “Confirm” som ufører utlogging. I koden fungerer dette ved å sette variabelen for innlogging til usann og slette JWT fra cookies. Vi setter også request header på nytt, slik at den ikke inkluderer JWT.

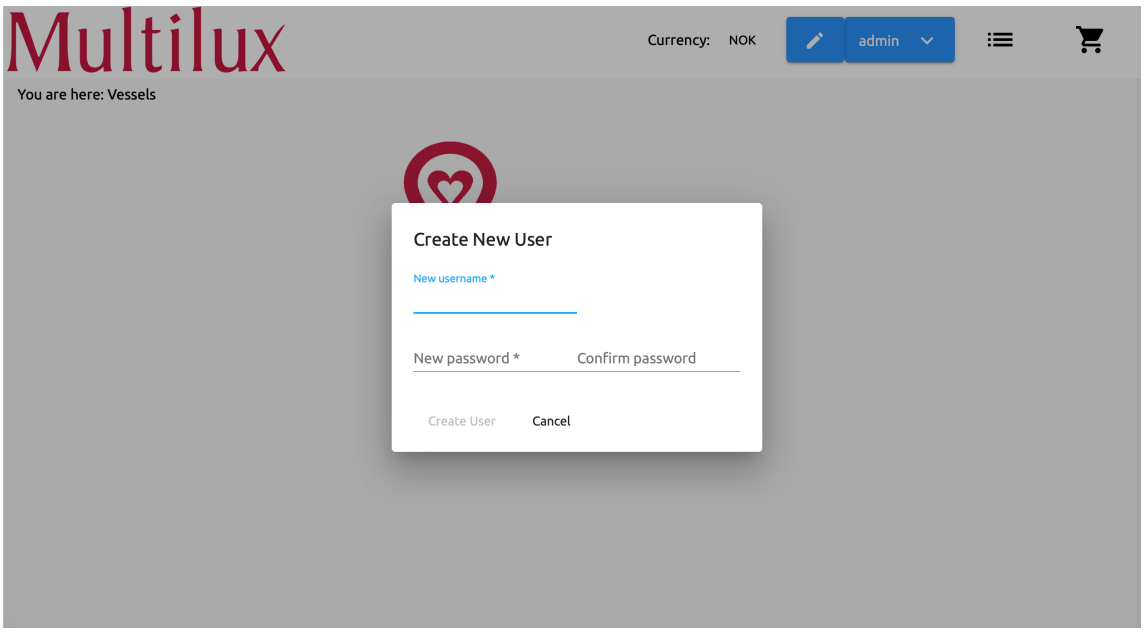
4.3.1.3 Brukere



Figur 4.22: Alternativet "Edit user" fra drop-down menyen

Ovenfor vises komponenten for å redigere bruker. Her har brukeren mulighet til å endre sitt eget brukernavn og passord.

Når backend får forespørselen vil den sjekke at brukeren som sendte forespørselen er den samme som er innlogget i programmet. Om dette er korrekt så vil den utføre brukernavn og/eller passord endringen.

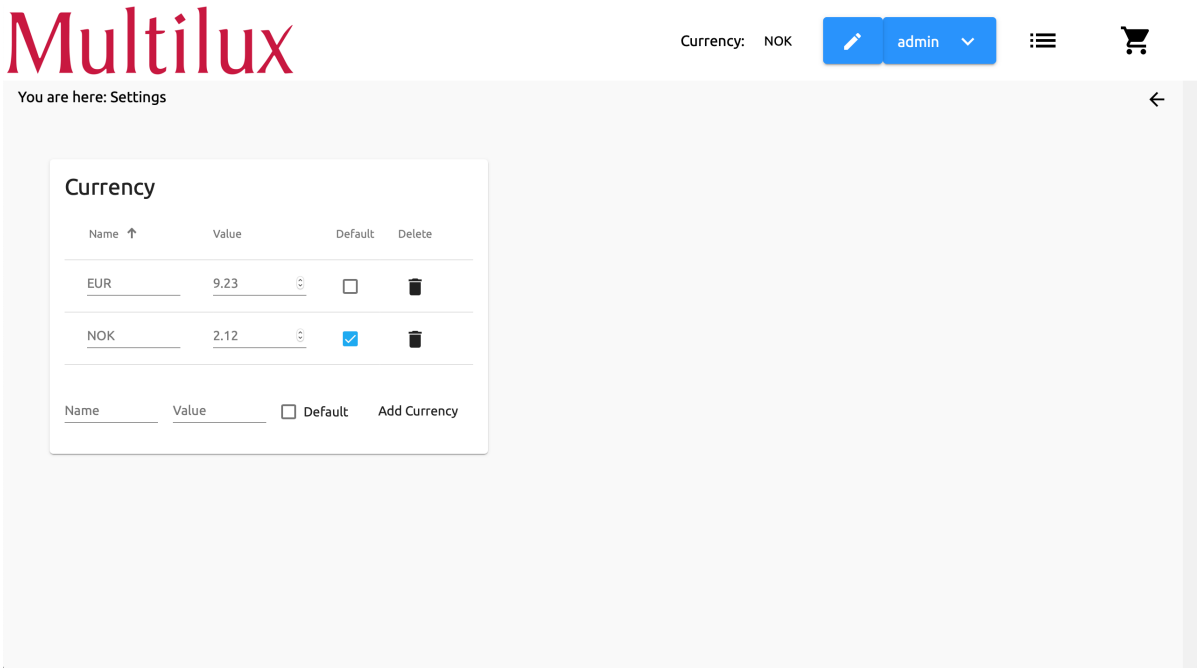


Figur 4.23: Alternativet "Add new user" fra drop-down menyen

I figur 4.23 vises komponenten for å lage ny bruker. I utgangspunktet ønsket oppdragsgiver å være den eneste autoriserte brukeren. Det vil si at han er den eneste som kan logge inn og administrere fartøyene.

Dersom oppdragsgiver i en senere anledning ønsker å utvide antall administrerende brukere, har det blitt laget et alternativ for dette. Brukeren kan tildele et nytt brukernavn og passord. Som vist ovenfor har denne dialogen likt design som "Edit user" (Figur 4.22). Dette er gjort for at applikasjonen skal få et gjennomgående design.

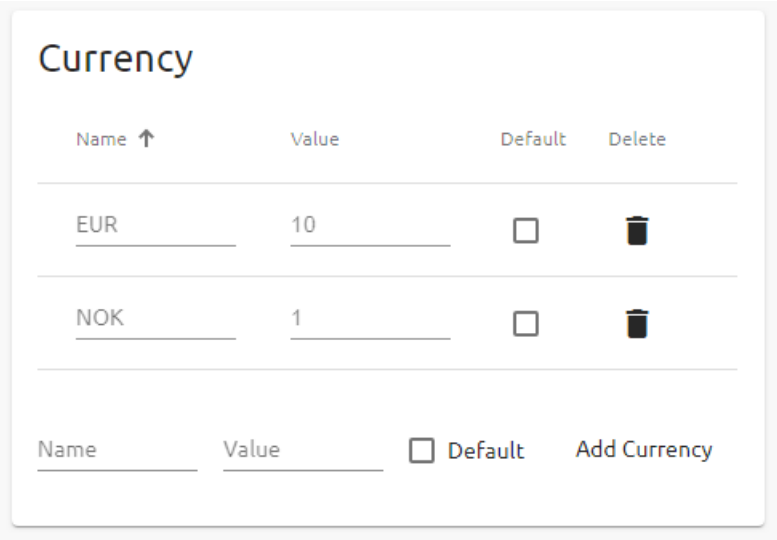
4.3.1.4 Innstillinger



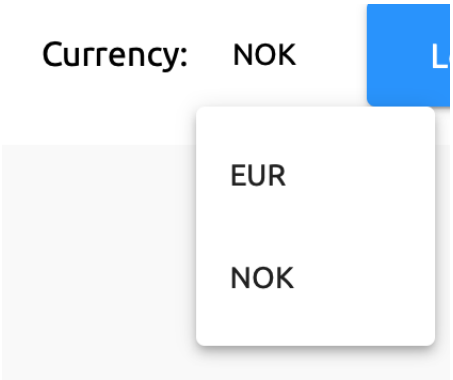
Figur 4.24: Alternativet “Settings” fra drop-down menyen

Denne siden er ment som en samling av innstillingene til web applikasjonen. For øyeblikket inneholder den kun innstillinger for valuta, men det er lagt opp til enkel utvidelse av flere innstillinger. Dette ved at hver innstilling får et tilhørende “Card” komponent som kan bli lagt til "Settings" siden.

Et annet alternativ for håndtering av innstillingene ville vært å lage dialoger i likhet med menyvalgene “Edit profile”, “Add new user” og “Logout”. Dette ville imidlertid ha resultert i et mer uoversiktlig brukergrensesnitt, og drop-down menyen vil i et tilfelle med mange innstillinger bli svært lang. Ifølge Steering loven (2.2.2 side 13) er dette også en dårlig løsning. Dette kommer av at brukeren må navigere musen gjennom en undermeny, i motsetning til å bare trykke på en knapp. Det ble derfor bestemt å lage en enkelt side hvor alle innstillingene kan vises på samme sted, samtidig som vi begrenser størrelsen på drop-down menyen.



Figur 4.25: Innstillinger for valuta



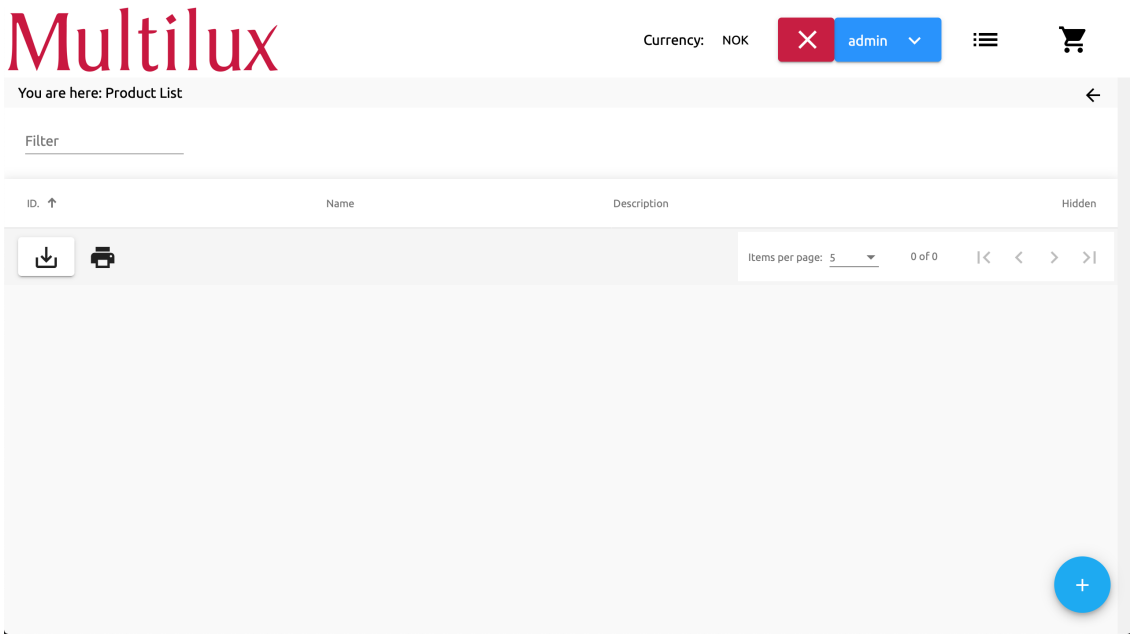
Figur 4.26: Valuta alternativer fra navigasjonsbaren, med norske kroner som standard

Som vist i figur 4.25 ovenfor, er det lagt til to valutaer (Norske Kroner og Euro). Valutaen kan bli slettet fra listen ved å klikke på søppelbøtteikonet til høyre for hver valuta. I tillegg kan en valuta velges som standard i web applikasjonen ved å trykke på avmerkningsknappen. I dette eksempelet er Norske Kroner valgt som standardvaluta.

Det er også mulighet for å legge til flere valutaer nederst i Currency. Multilux ønsket selv at det skulle være statisk valuta som ikke endres automatisk, men som kan justeres ved behov, f.eks. dersom valutakursen endrer seg drastisk.

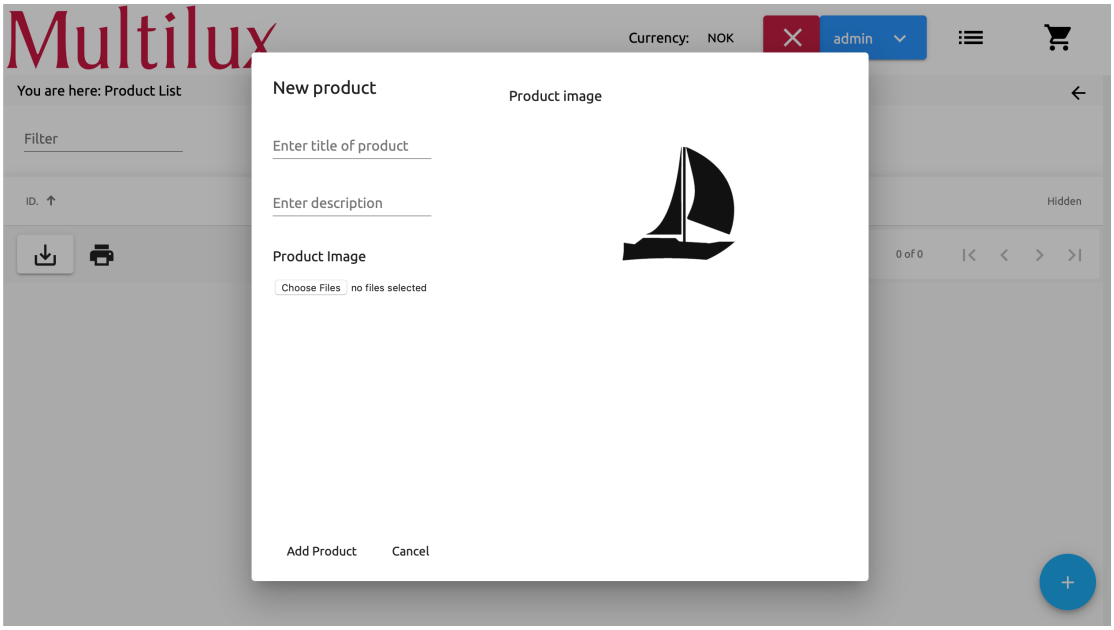
De opprettede valutaene vil dermed bli vist som “Currency” i drop-down menyen på navigasjonsbaren (Figur 4.26).

4.3.1.5 Produktlisten



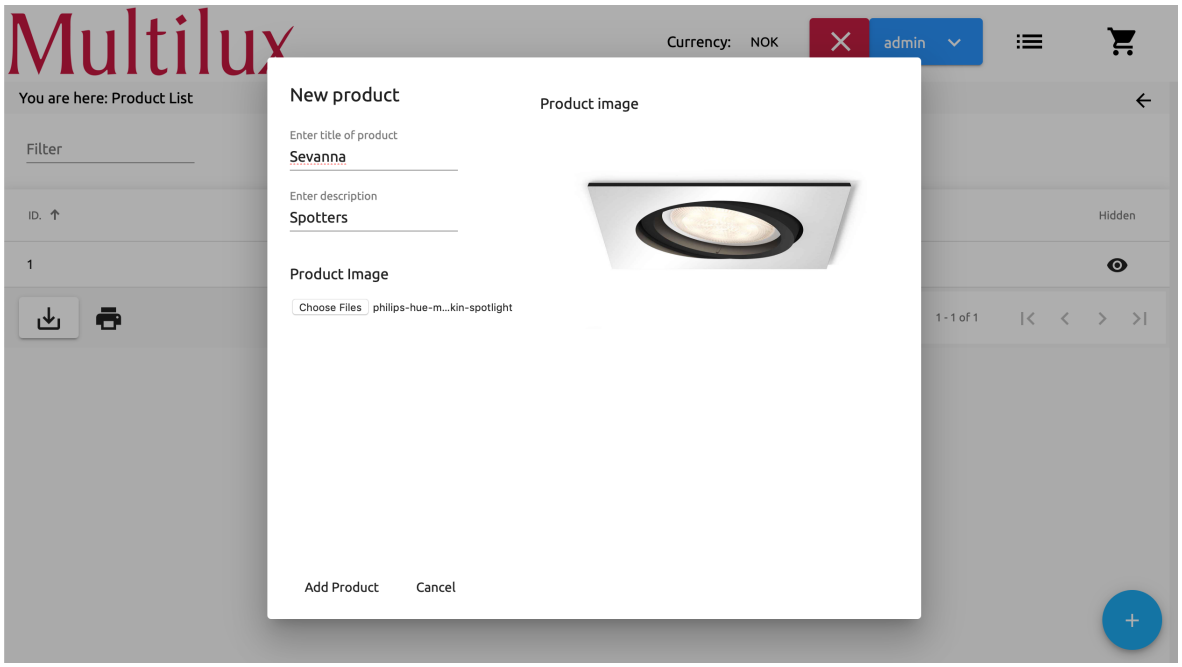
Figur 4.27: Produktlisten i redigeringsmodus uten innhold

Figuren ovenfor demonstrerer produktlisten. Her er produktlisten tom, da det ikke er vedlagt noen produkter enda. For at brukere skal kunne legge til produkter er det nødvendig å være i redigeringsmodus. FAB nede til høyre gir mulighet for å legge til et nytt produkt (Figur 4.27).



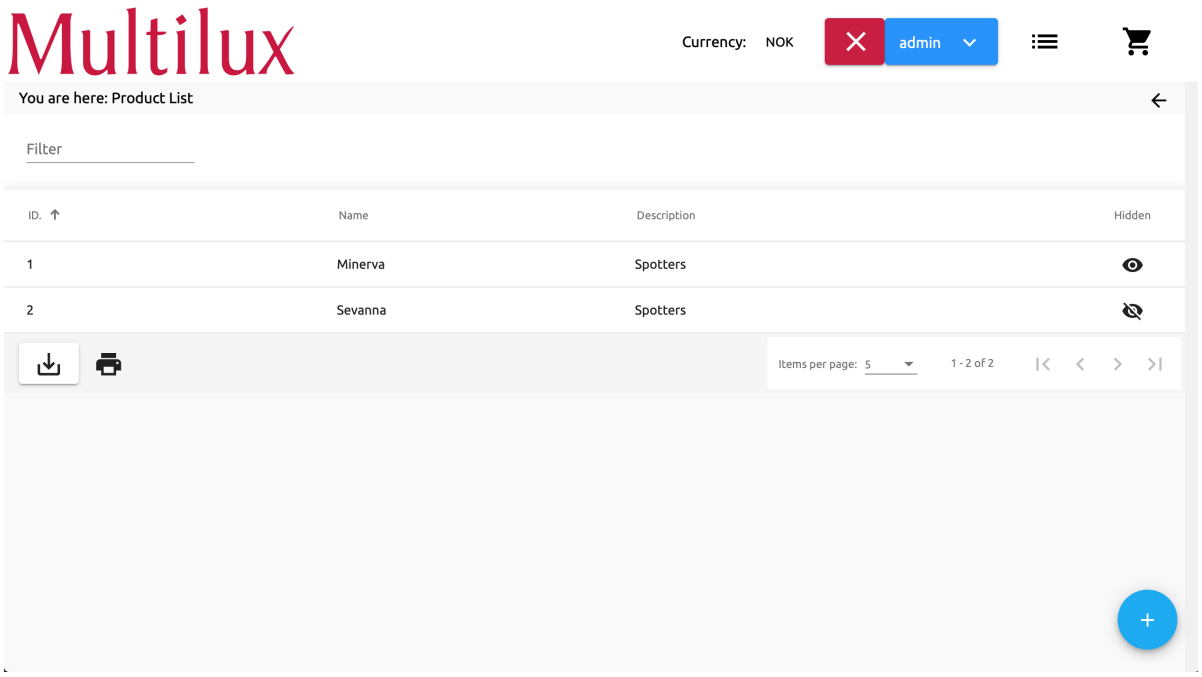
Figur 4.28: New Product dialogboks

Figur 4.28 viser dialog boksen for å legge til nytt produkt etter du har trykt på FAB. Her har brukeren mulighet til å angi en tittel og beskrivelse, samt et bilde av produktet slik at kundene kan se hvordan det ser ut.



Figur 4.29: Eksempel av en ferdig utfylt New Product dialogboks

Her er det blitt laget et eksempel på hvordan dialogboksen kan se ut. Tittelen på produktet er navnet på lyset, en beskrivelse på hva slags lyst det er og et bilde av produktet. Ved å velge “Add product”, blir produktet lagt inn i listen.



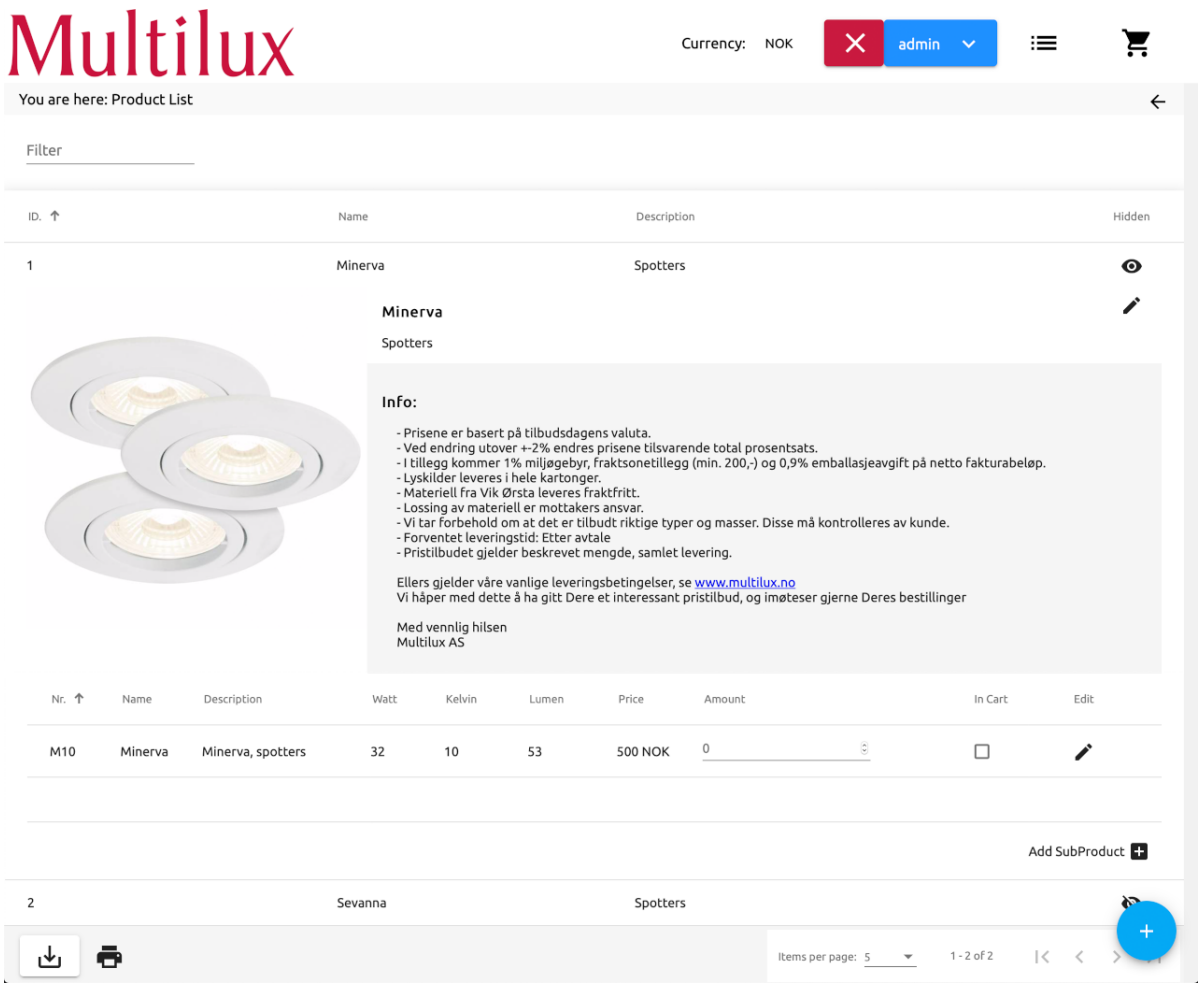
Figur 4.30: Produktlisten med innhold

Figur 4.31: Indikerer synlighet for kunden

Figur 4.32: Indikerer skjult synlighet for kunden

Figur 4.30 viser et overblikk på hvordan produktlisten kan se ut med innhold. Her har brukeren mulighet til å bestemme produktets synlighet for kunden (Figur 4.31 og 4.32). Som vist ovenfor har produktet “Sevanna” skjult synlighet for kunden, mens “Miranda” er synlig.

Hovedpoenget er å gi brukeren mulighet til å skjule produkter. Eksempler her kan være produkter som er utsolgt, eller som ikke er tilgjengelige på markedet før en viss tid. Dersom produktene blir skjult her, vil de også være skjult i rommene de er angitt.

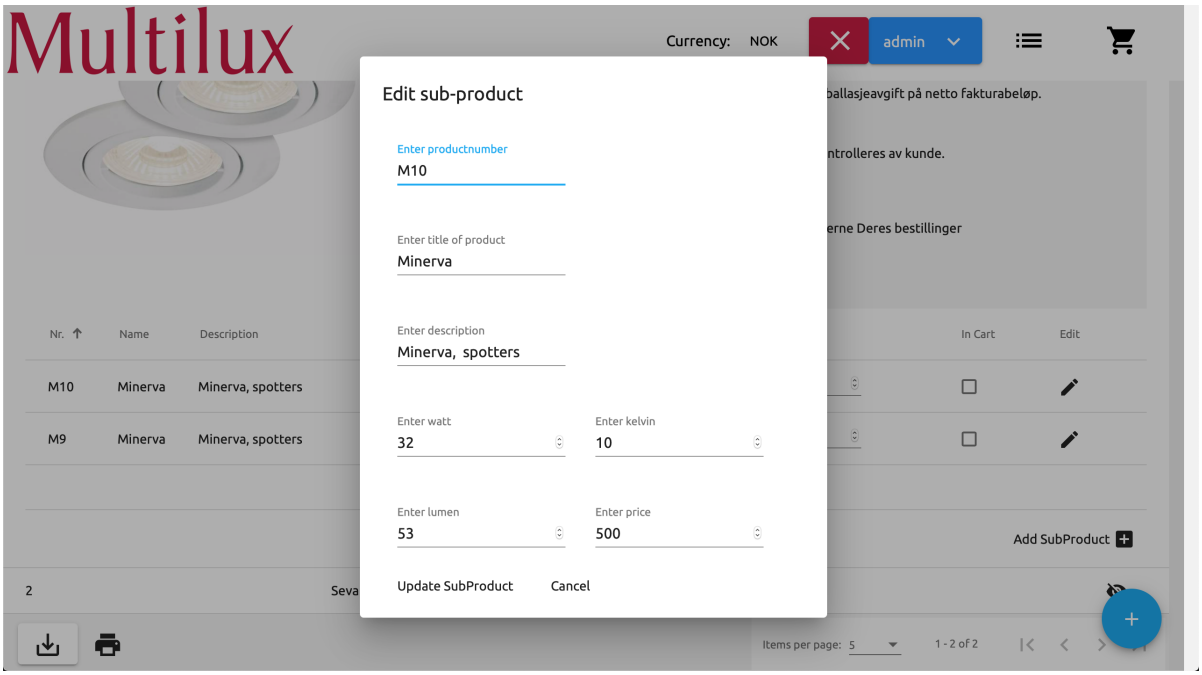


Figur 4.33: Produktlisten etter at brukeren har valgt et produkt

Figur 4.34: Laster ned en CSV for hele produktlisten

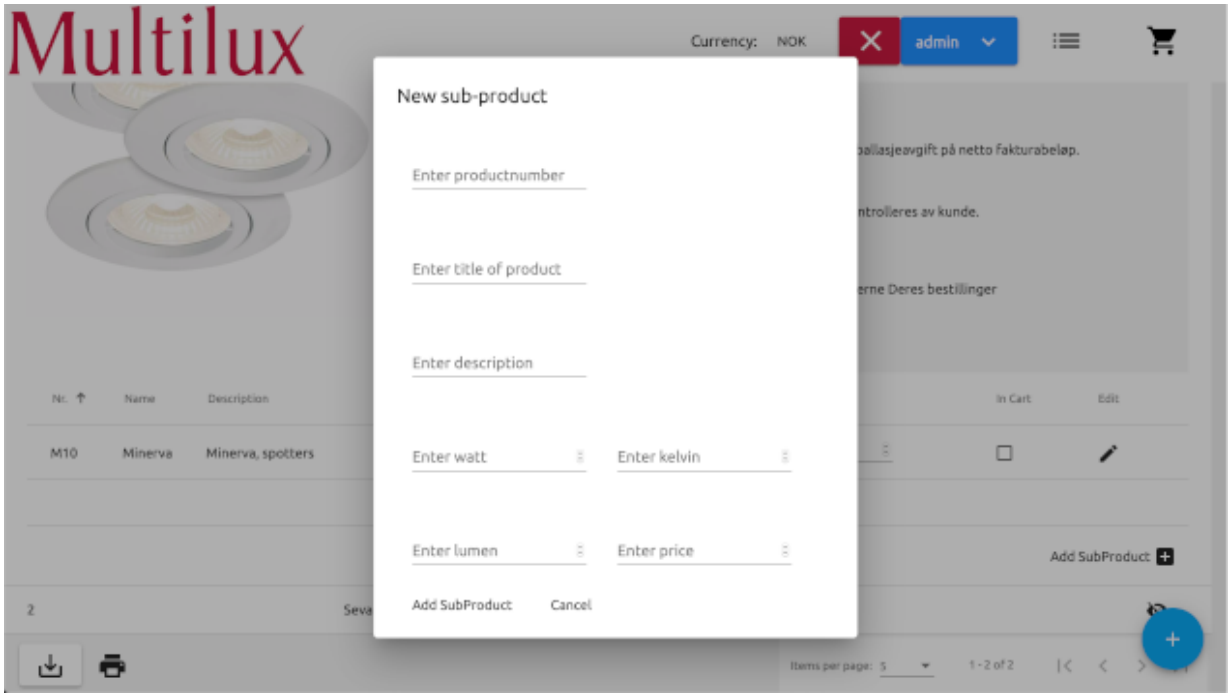
Figur 4.35: Åpner produktlisten som en PDF for utskrift

På figur 4.33 ovenfor kan vi se lister over underprodukter. Disse er produktalternativer under selve hovedproduktet. Bakgrunnen for dette er at det kan være flere versjoner av et produkt, med forskjellige verdier. Disse er samlet under selve hovedproduktet. De to ikonene nederst i venstre hjørne skal la brukeren lagre informasjonen som en CSV fil (Figur 4.34) og skrive ut som PDF fil (Figur 4.35), men denne funksjonaliteten er enda ikke lagt inn i systemet. Tilsvarende funksjonalitet ligger i handlekurven og er dekket på side 79.



Figur 4.36: Edit sub-product dialogboks

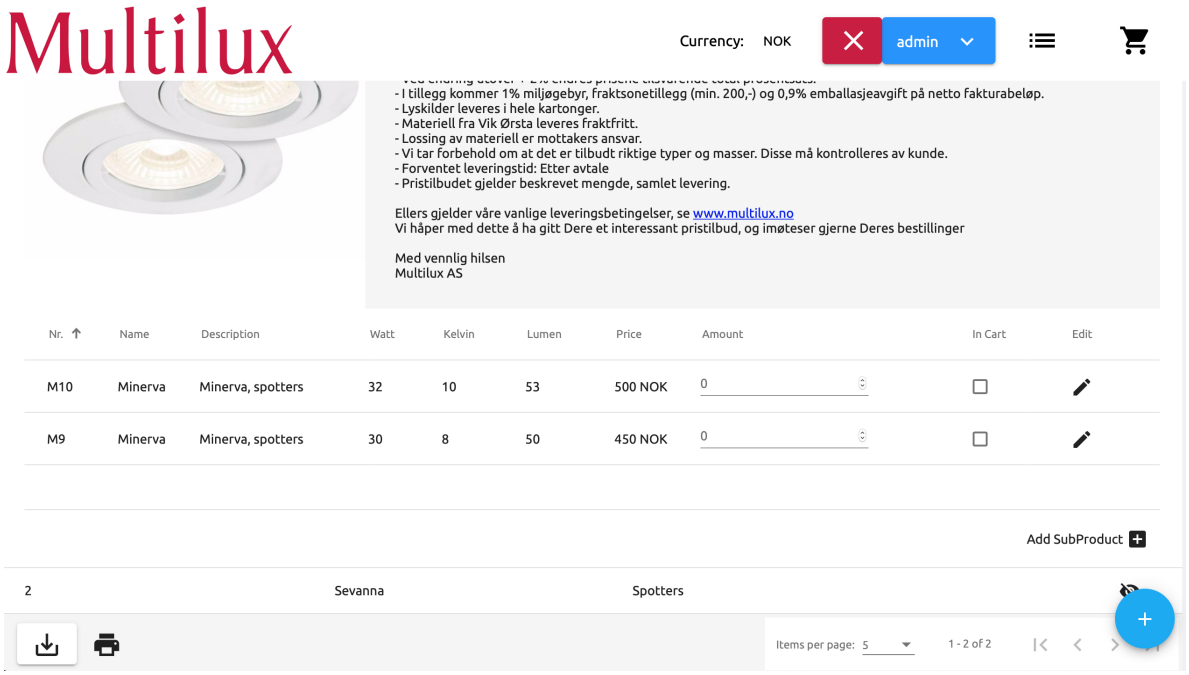
Dersom brukeren ønsker å redigere et eksisterende underprodukt vil dialogen ovenfor vises. Her har brukeren mulighet til å endre på all informasjon som er satt i det enkelte underprodukt.



Figur 4.37: New sub-product dialogboks

I figur 4.37 vises dialogboksen for å legge til et nytt underprodukt. Brukeren har to muligheter til å legge til nytt underprodukt, og følger designet som er valgt gjennom hele web applikasjonen.

Brukeren kan enten benytte seg av FABen eller alternativet “Add SubProduct. Her kan brukeren skrive inn produktnummer, tittel og beskrivelse. De fire siste feltene gir brukeren mulighet til å skrive antall watt, kelvin, lumen og pris. Ved å klikke på “Add SubProduct” vil underproduktet bli lagt til i hovedproduktet, som vist på skjermbildet under.



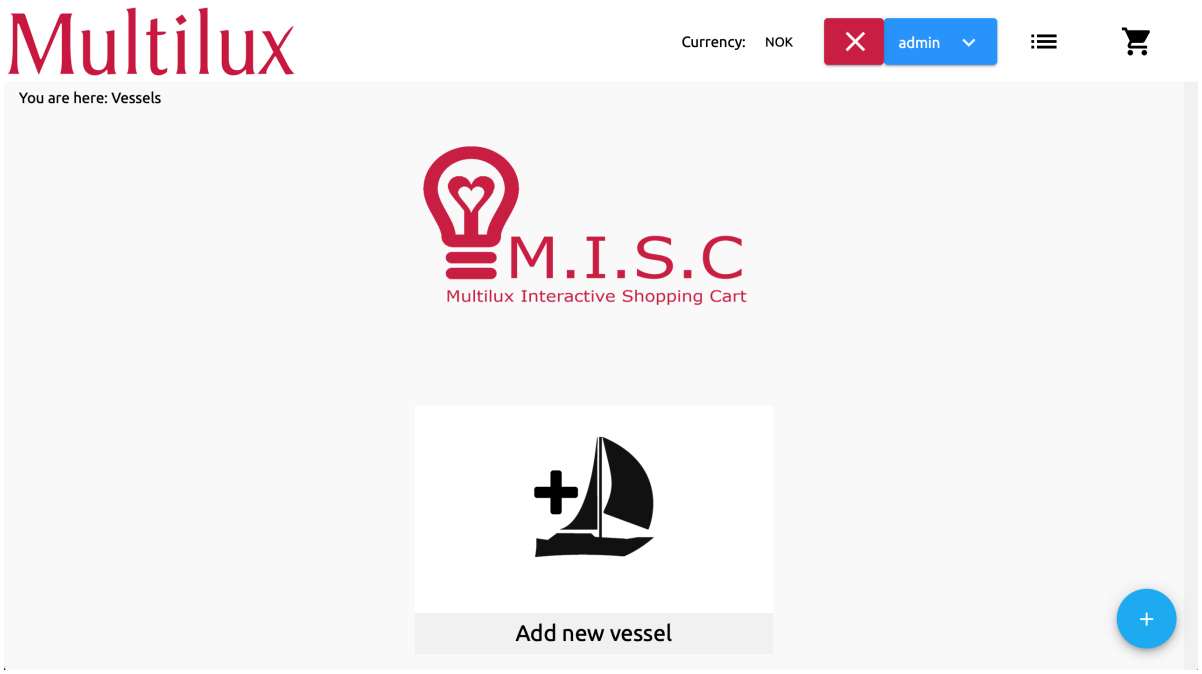
Figur 4.38: Subproduktet er lagt til produktet

4.3.1.6 Forside

Forsiden er delt inn i to deler, en navigasjonsbar og en hovedseksjon, som kan sees fra figur 4.17 på side 50.

I figuren består hovedsekjonen kun av web applikasjonens logo. Dette er fordi det enda ikke er lagt inn fartøy. Det vil i si at når ingen fartøy ligger inne i systemet, eller fartøyene er satt til "ikke synlig", er det slik web applikasjonen ser ut fra kundens perspektiv.

4.3.1.7 Fartøy

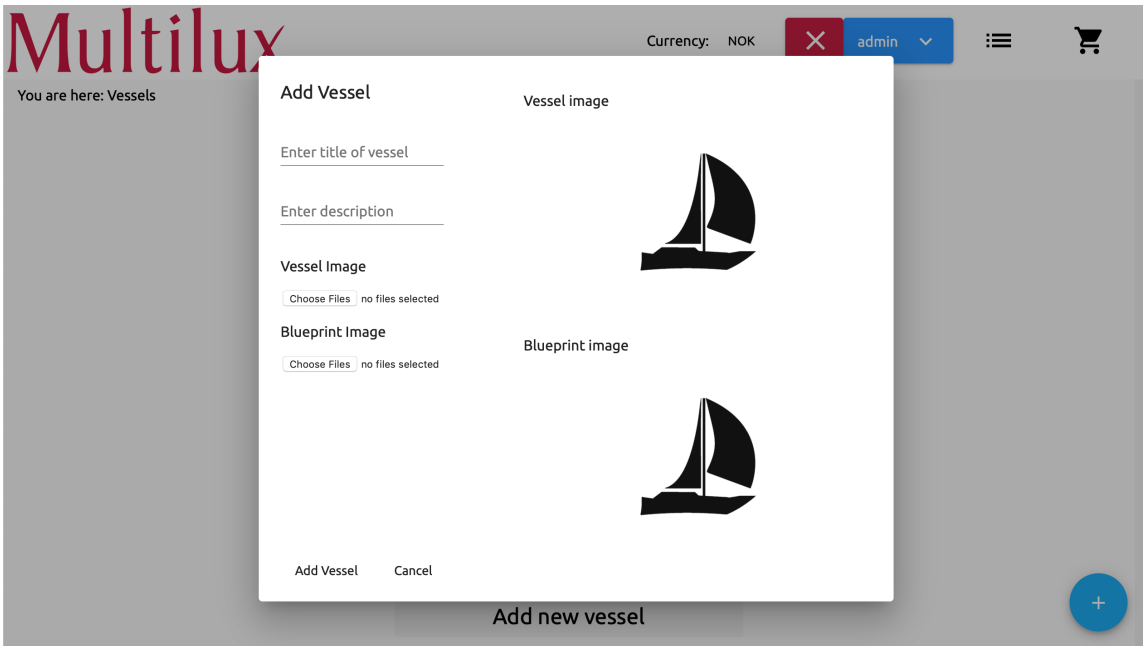


Figur 4.39: Forsiden i redigeringsmodus

Som vist i skjermbildet over har komponenten for å legge til fartøy dukket opp i hovedseksjonen. Brukeren har to valgmuligheter for å legge til nye fartøy. Den ene er å klikke på "Add new vessel" komponenten og den andre er en FAB nederst på høyre hjørne.

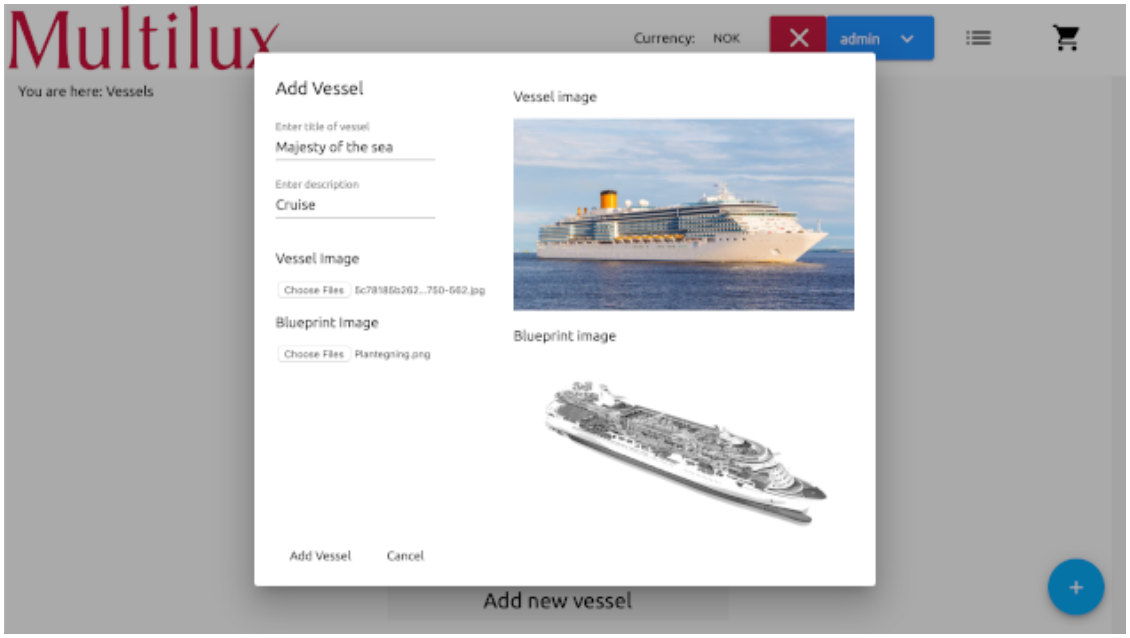
For å få et bedre design og en mer brukervennlig web applikasjon, har det resultert i to valgalternativer. Det skal være mulig å legge til nye fartøy på samme område som fartøyene blir vedlagt. I tillegg skal det være mulig å legge til fartøy uten å lokalisere "Add new vessel" komponenten.

Dersom det blir vedlagt mange fartøy, blir brukeren nødt til å bla lengre ned på siden for å lokalisere "Add new vessel" komponenten. Derfor kan brukeren heller velge FABen. Dette er et design som er gjennomgående i hele web applikasjonen, med samme baktanke.



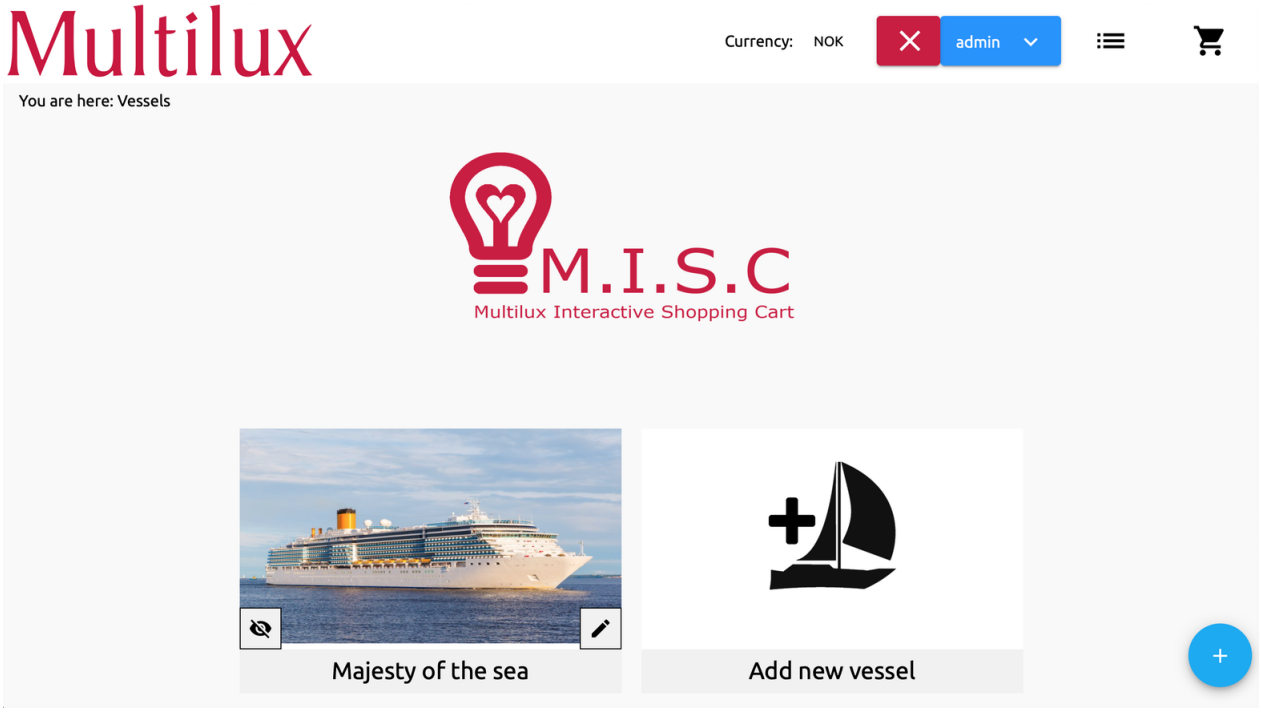
Figur 4.40: Dialogboks for å legge til fartøy

Skjermbildet ovenfor viser dialog boksen for å legge til nye fartøy. Dialogboksen består av en tittel, beskrivelse og to filopplastningsalternativer. Den første filopplastningen er ment som et bilde av fartøyet og den nederste en plantegning av fartøyet.



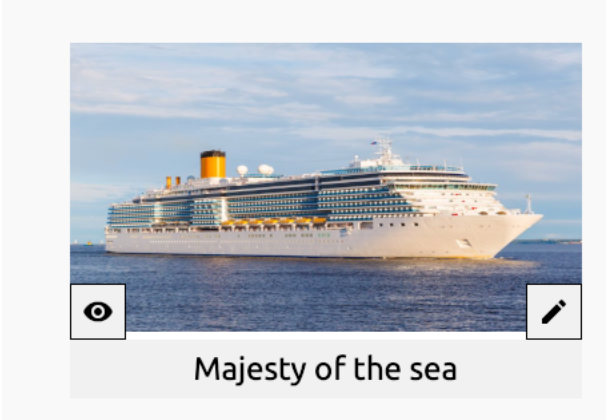
Figur 4.41: Eksempel av en ferdig utfylt Add Vessel dialogboks

Skjermbildet ovenfor demonstrerer et eksempel på hvordan dialogen kan se ut. Navnet til fartøyet er gitt som tittel og beskrivelsen viser hva slags fartøy det er. I denne demonstrasjonen er det blitt vedlagt et fartøy med navn "Majesty of the sea" og beskrivelsen indikerer at det er et cruise skip. Det første bildet er et bilde av selve fartøyet, mens bildet under er en plantegning. Dette er for å gi bedre oversikt over oppbygningen og rommene i fartøyet.



Figur 4.42: Forsiden i redigeringsmodus med et fartøy lagt til

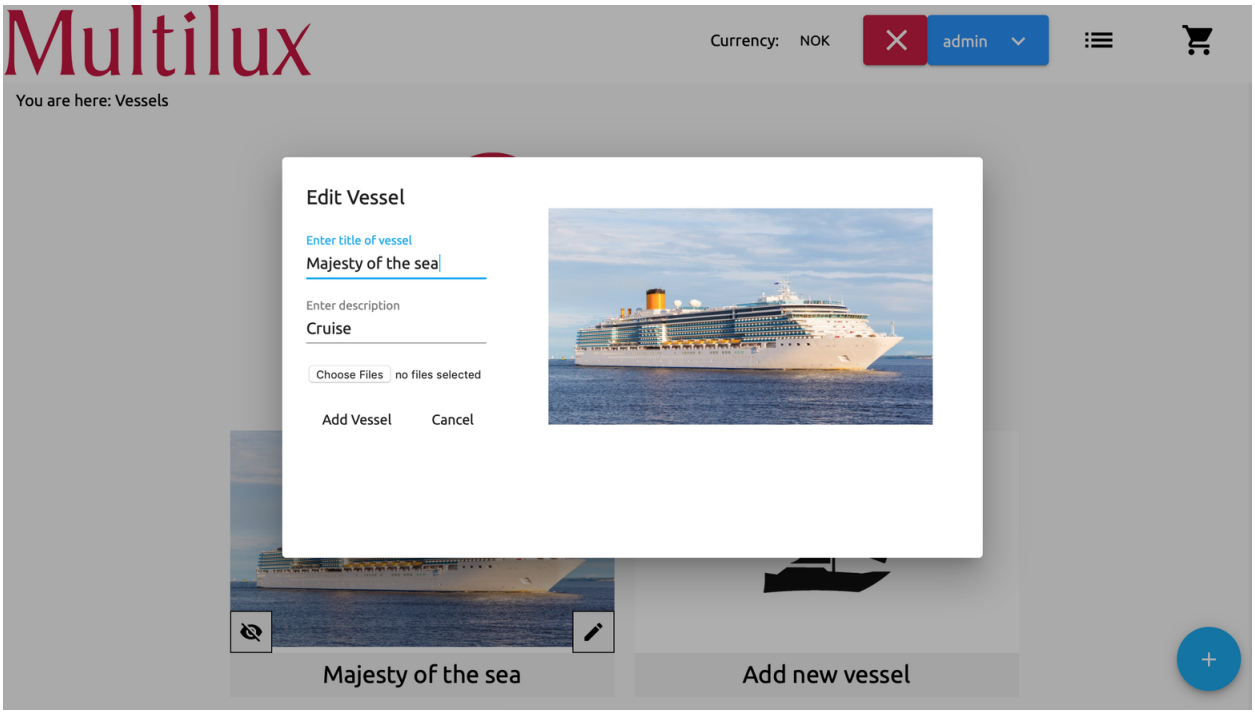
Eksemplet ovenfor viser forsiden med vedlagt fartøy. Alle fartøy som blir vedlagt plasseres til venstre for "Add new vessel" komponenten.



Figur 4.43: Fartøy komponenten

Figur 4.44: Ikon for redigering av element

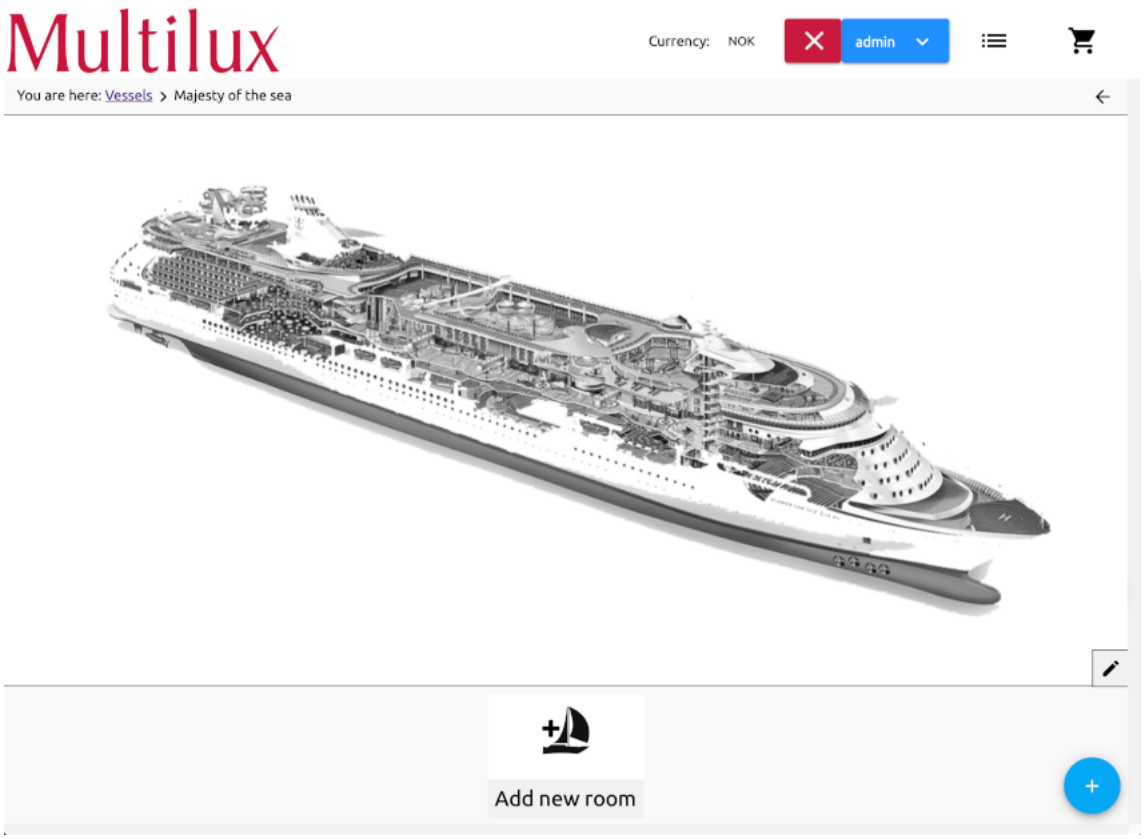
Som vist ovenfor består fartøyet av to ekstra ikoner. Til venstre er ikonet for synlighet (Figur 4.31 og 4.32, side 58) og et ikon for redigering (Figur 4.44). Dette skal gi brukeren mulighet til å redigere fartøyet på en enkel måte.



Figur 4.45: Edit Vessel dialog

Ovenfor vises en dialogboks når brukeren velger å endre fartøyet. Brukeren kan forandre tittelen, beskrivelsen og bilde av selve fartøyet. Som demonstrert ovenfor er endring av plantegningen ikke et alternativ. Grunnen er at plantegningen kan endres på når man skal legge til rom i fartøyet.

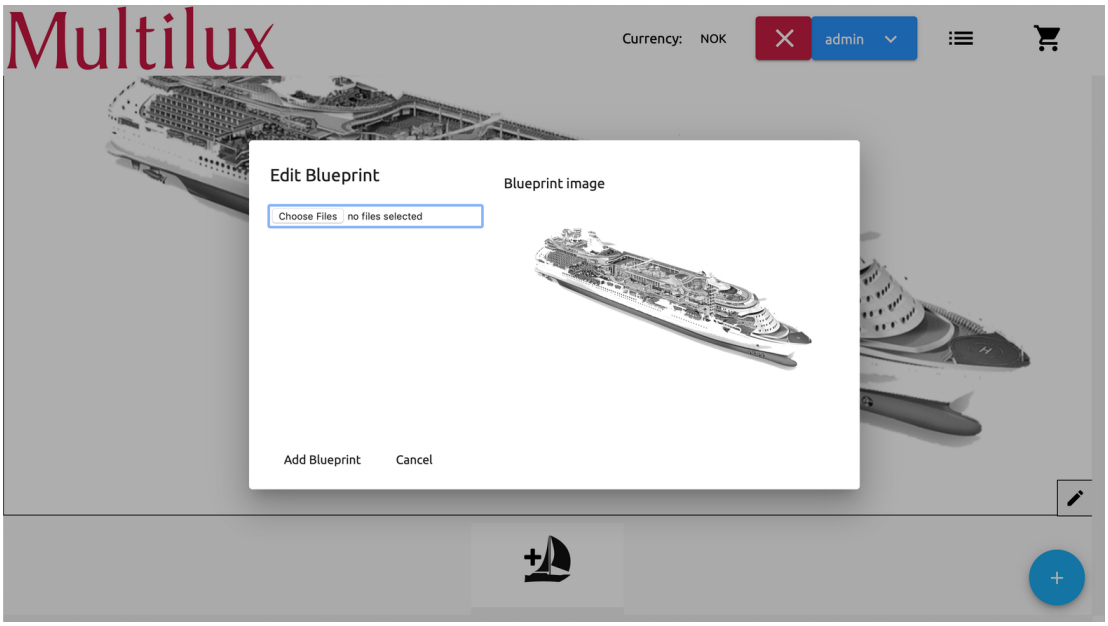
4.3.1.8 Rom



Figur 4.46: Innhold etter valgt fartøy: valg av rom

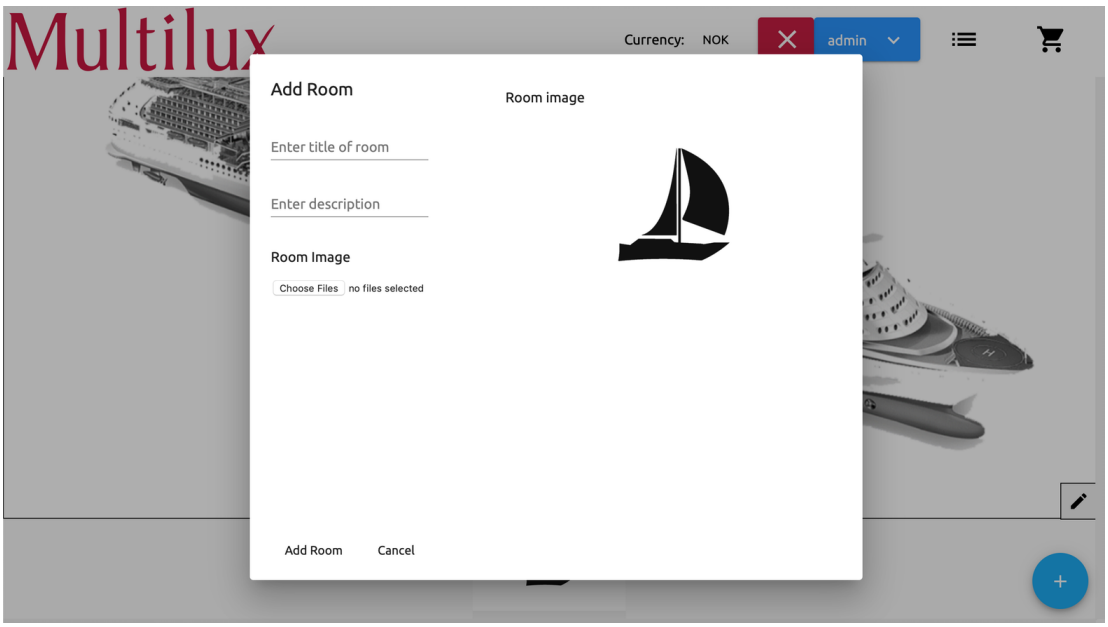
Skjermbildet ovenfor demonstrerer brukergrensesnittet etter at brukeren har klikket seg inn på det vedlagte fartøyet. I likhet med figur 4.43 har plantegningen et redigeringsalternativ. Ved å klikke på denne får brukeren mulighet til å endre bilde av plantegningen.

I likhet med “Add new vessel” seksjonen (Figur 4.39 side 62), har denne delen av webapplikasjonen to valgalternativer for å legge til rom.



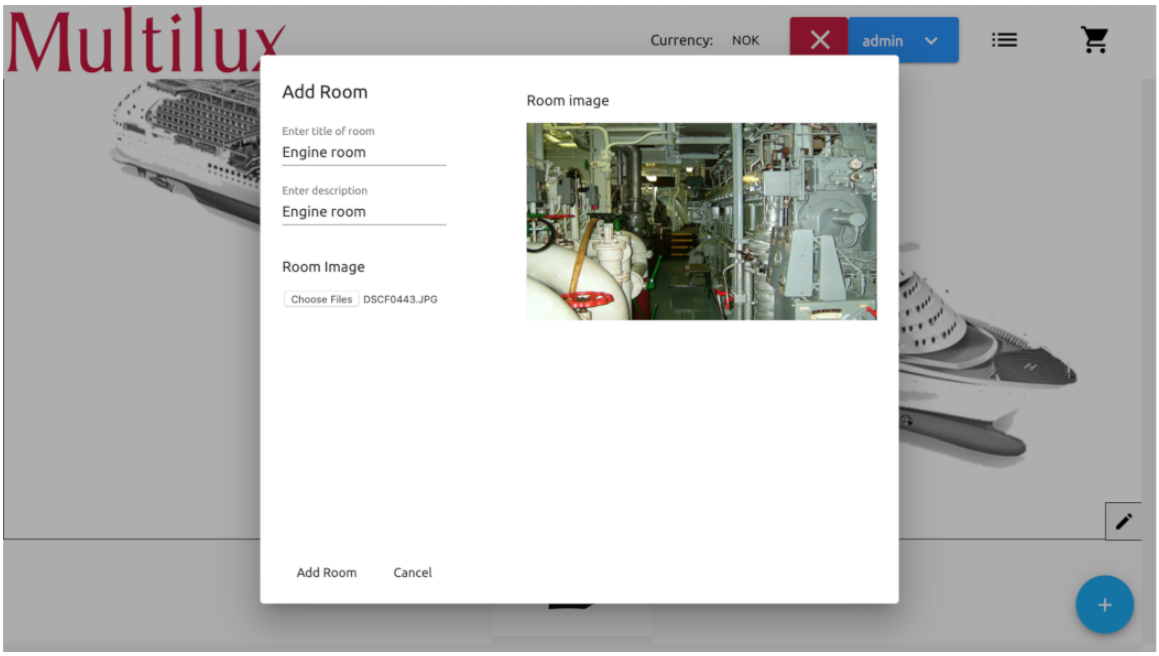
Figur 4.47: Edit Blueprint dialog

Brukeren kan laste opp nytt bilde av plantegningen ved å benytte seg av redigeringsalternativet som vist ovenfor.



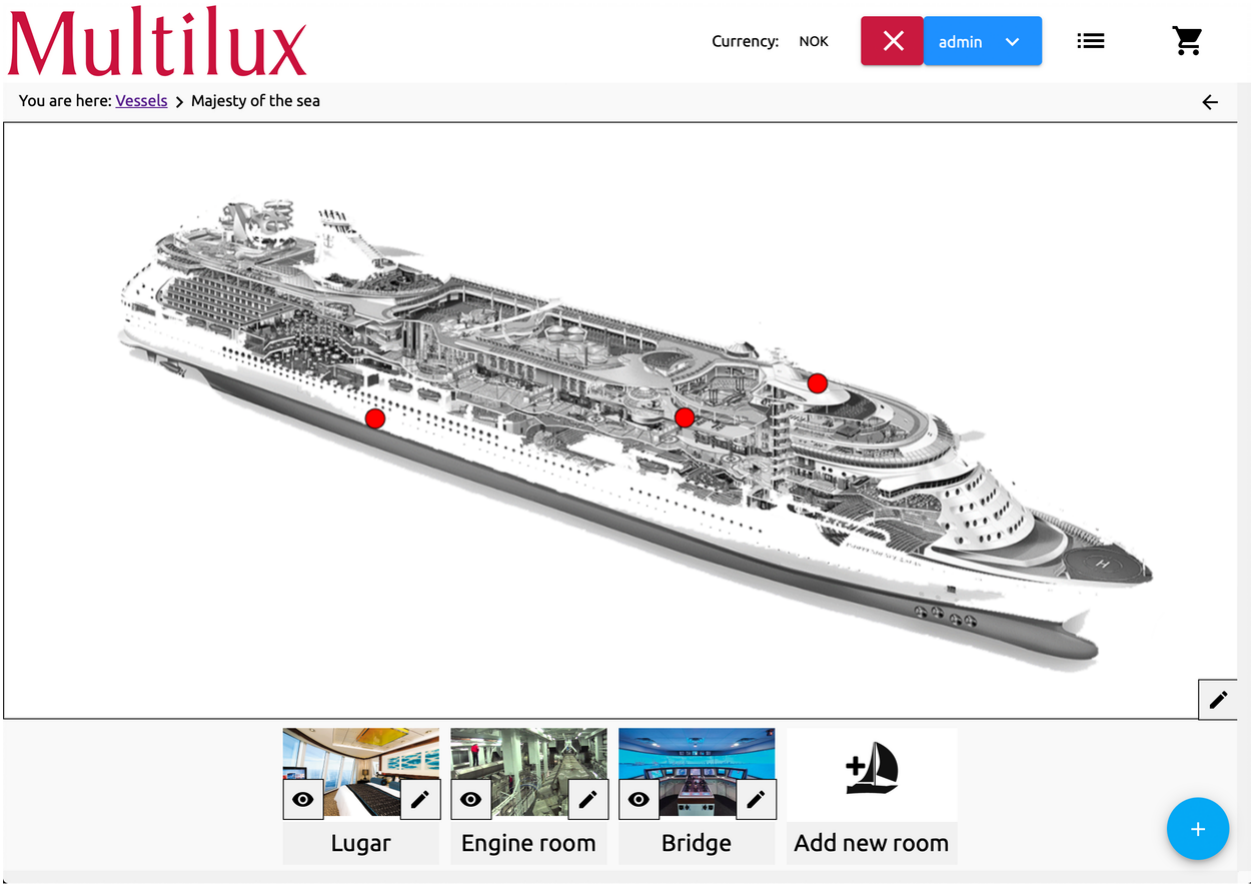
Figur 4.48: Add Room dialog

Ovenfor vises dialogboksen for å legge til rom i fartøyet. Dialogen inneholder en tittel og beskrivelse av rommet, samt mulighet for å laste opp bilde. Når brukeren velger “Add room”, blir rommet automatisk lagt til i hovedseksjonen med tilhørende røde knapper, som vist i neste skjermbilde.



Figur 4.49: Add Room dialog

Skjermbildet ovenfor viser et eksempel på hvordan dialogen kan se ut. Her er det lagt til en nytt rom med navn “Engine room”. Beskrivelsen inneholder en kort forklaring av rommet. Det er også lagt til et bilde av rommet. Dette er for å senere kunne plassere produktene.

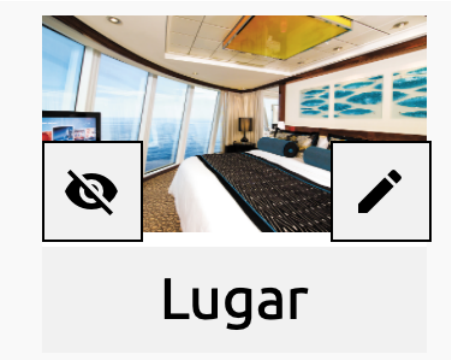


Figur 4.50: Flere rom lagt til fartøyet

Ovenfor vises et eksempel etter at flere rom er lag til i det angitte fartøyet. De røde knappene som vises på plantegningen kan plasseres der rommet befinner seg. Dette var spesifikt ønsket av oppdragsgiver (Vedlegg B).

Det er valgt å ha en liste over fartøyene under plantegningen for bedre oversikt og brukervennlighet. Når musen svever over de røde knappene vil disse rommene vises med blå skrift. Dette vil gi en indikasjon til brukeren om at knappene er plassert på riktig sted.

Under logoen til Multilux vises stiene som er valgt for å komme til nåværende side. Dette er for å gjøre det enkelt for brukere å navigere seg gjennom web applikasjonen. Veiviseren vil være lik for både kunder og administrative brukere.

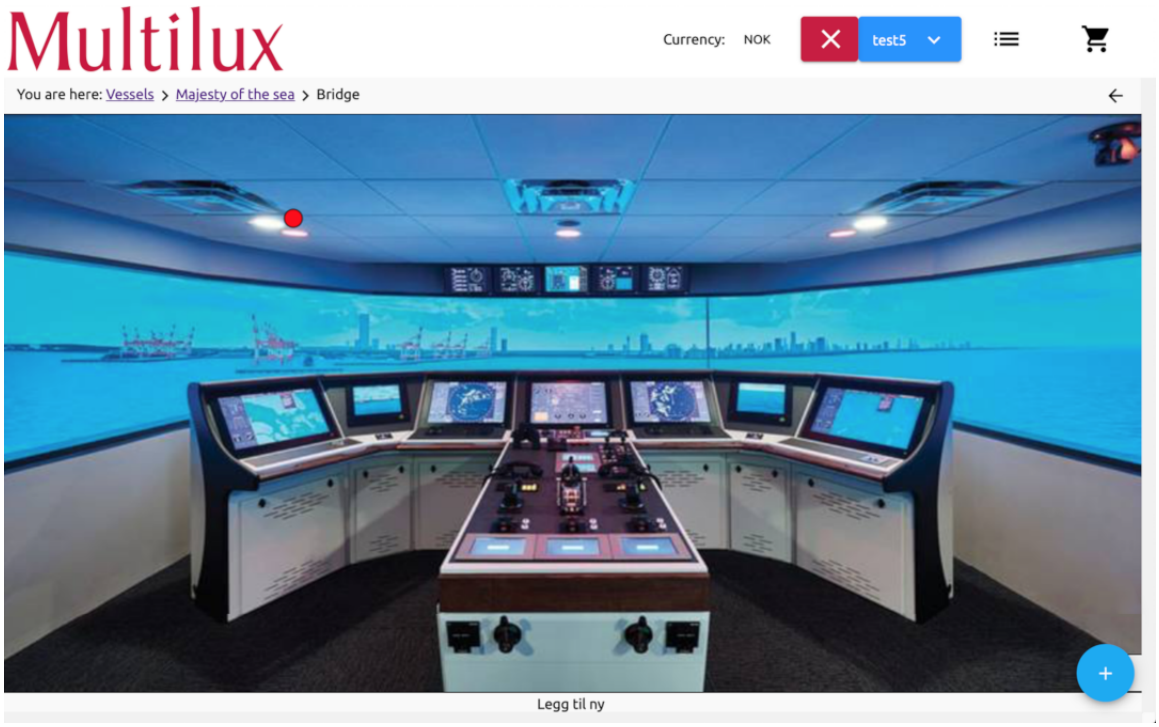


Figur 4.51: Rom komponenten

Figur 4.51 viser bilde av et vedlagt rom. Rommet har to tilhørende ikoner. Dette vil i hovedsak gi samme valgmuligheter som figur 4.43 på side 64. Dersom det valgte fartøyet er skjult, vil alt annet innhold som tilhører denne også være det.

4.3.1.9 Produkt

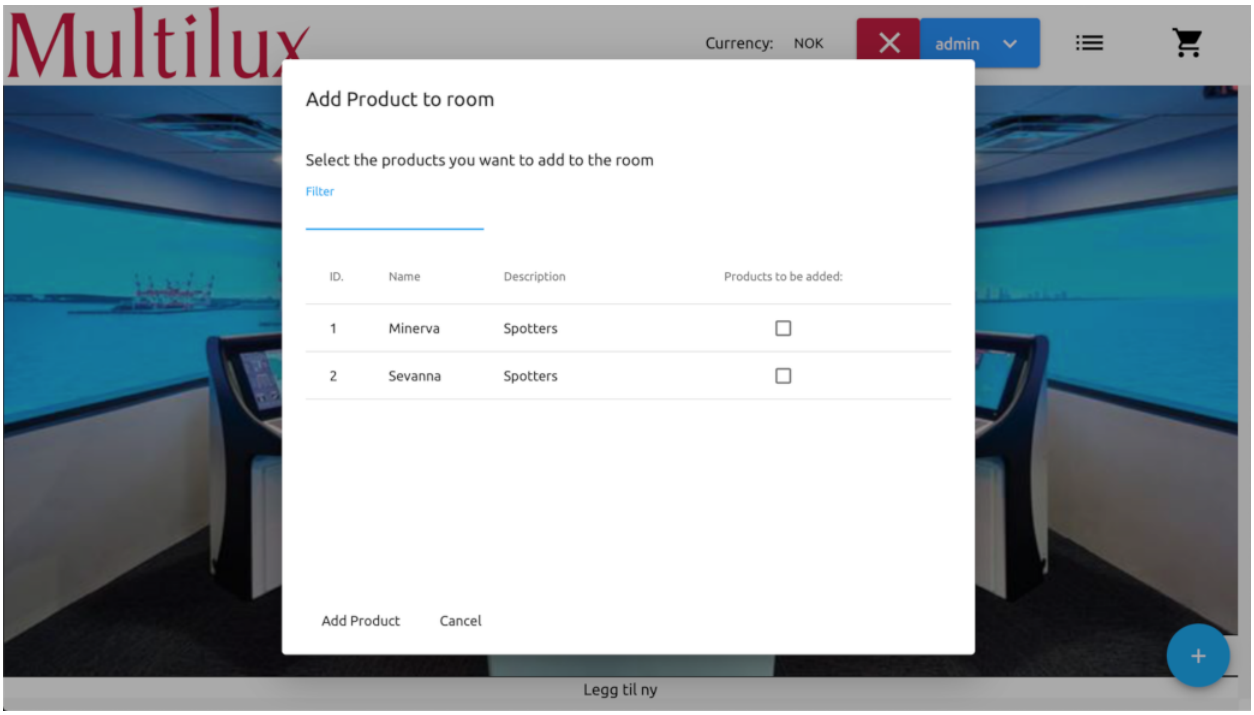
Dersom kunden klikker seg inn på de ulike rommene, vil de bli dirigert til rommet sin side. Her har brukeren mulighet til å plassere og legge til produkter i rommet.



Figur 4.52: Innhold etter valgt rom, valg av produkt

Skjermbildet ovenfor viser brukergrensesnittet etter rommet “Bridge” er valgt. Denne delen av applikasjonen skal la brukere legge til produkter i rommet. Den har også et redigeringsalternativ i likhet med fartøy og rom komponentene, som lar brukere endre bilde av rommet.

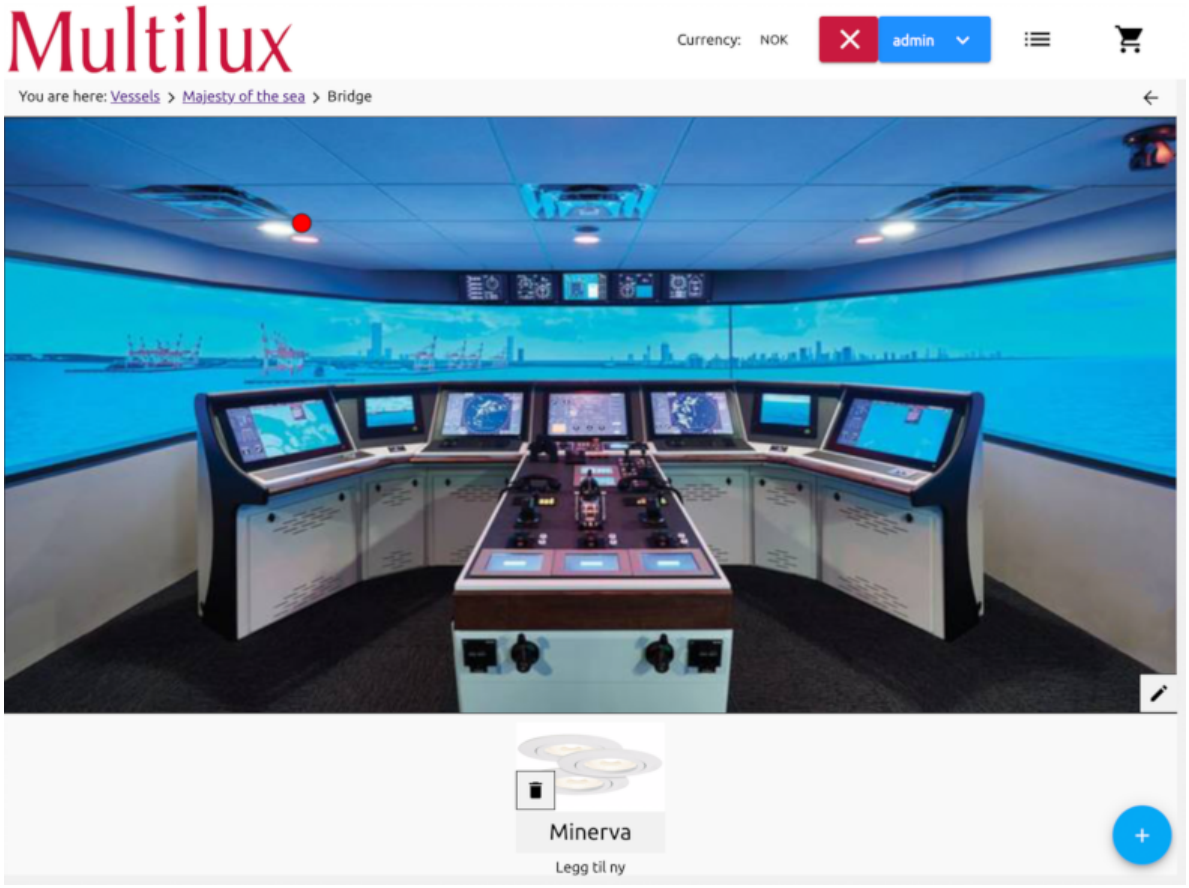
Den blå FAB nede i høgre hjørne på figur 4.52 gir brukeren mulighet for å legge til produkter i rommet, som vist i neste figur.



Figur 4.53: Add Product to room dialogboks

Skjermbildet ovenfor viser en oversikt over produktene som er tilgjengelige. I denne demonstrasjonen er det kun lagt til to produkter. Brukeren kan i utgangspunktet legge til så mange produkter som ønskelig i produktlisten (Figur 4.27 side 56).

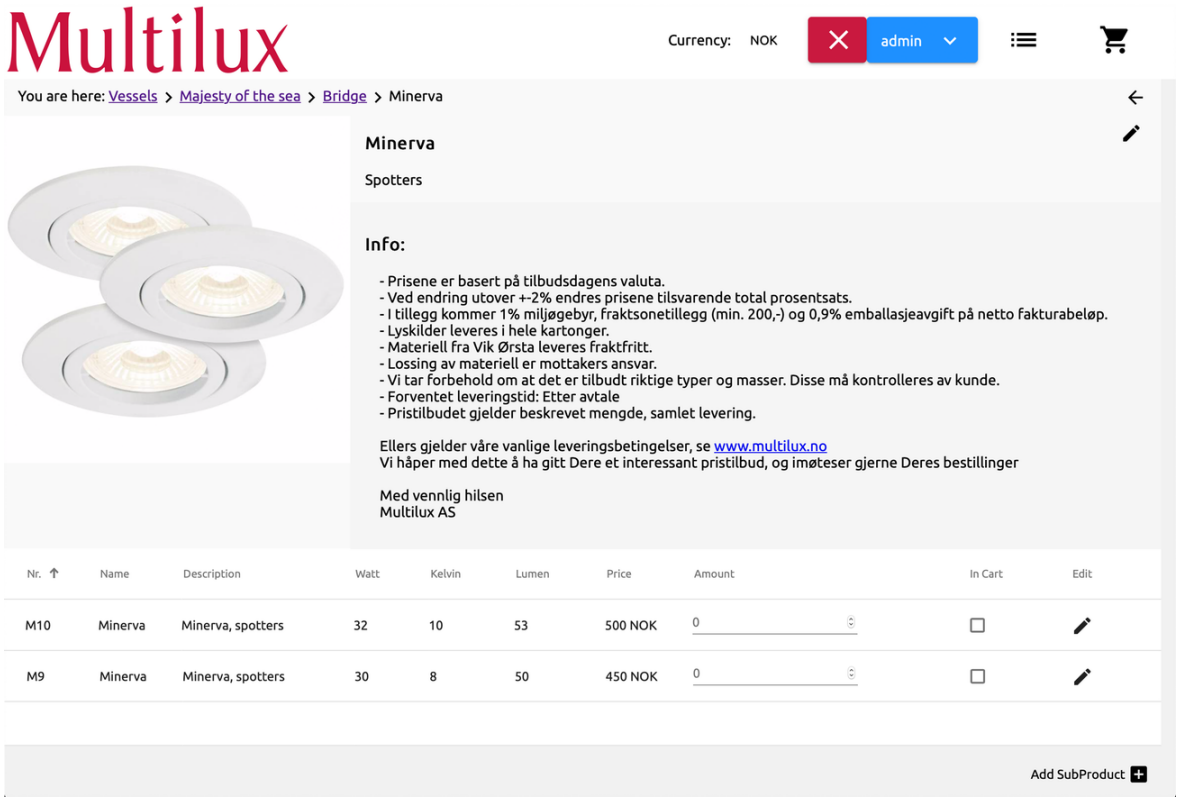
Dialog boksen består av en liste hvor hvert produkt har en id, navn, beskrivelse og en avmerkningsknapp. Avmerkningsknappen gir mulighet for å legge til flere produkter samtidig. Brukeren kan benytte seg at søkefeltet for å navigere seg til ønskede produkt.



Figur 4.54: Produkt lagt til i rom. En rød knapp dukker opp på bildet av rommet.

Som vist i figur 4.54 har en rød knapp dukket opp på bildet av rommet. Det er fordi produktet “Minerva” er lagt til i rommet. Den røde knappen kan plasseres der det er ønskelig. Som vist har også produktet dukket opp under bildet av rommet. Produktet har et søppelbøtteikon på venstre side, som er der for å gjøre det enkelt for brukeren å fjerne produkter fra rommet.

4.3.1.10 Underprodukt



Figur 4.55: Innholdet etter at brukeren har valgt et produkt

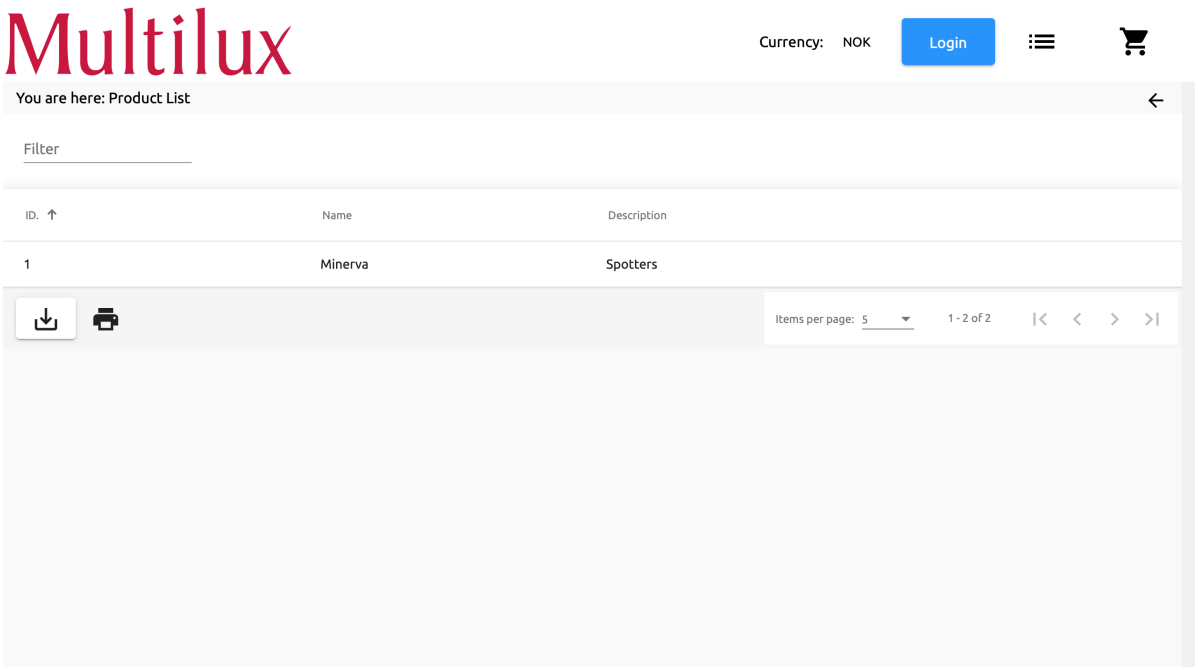
Dersom brukeren klikker på det ønskede produktet vil hovedseksjonen oppdatere seg og man får en mer detaljert visning over produktet og dets underprodukter. Som vist ovenfor har brukeren et redigeringsalternativ i høyre hjørne. Dette gir brukeren mulighet til å redigere det valgte produktet. Informasjonen som står under produkter er standard informasjon som vises under alle produktene. Nederst er listen over underprodukter. Her er det også mulighet for å endre eksisterende underprodukter og legge til nye. Dette er for at brukeren skal slippe å dirigere seg tilbake til produktlisten i navigasjonsbaren.

4.3.2 Kundens perspektiv

Dette underkapittelet tar for seg brukergrensesnittet fra kunden sitt perspektiv.

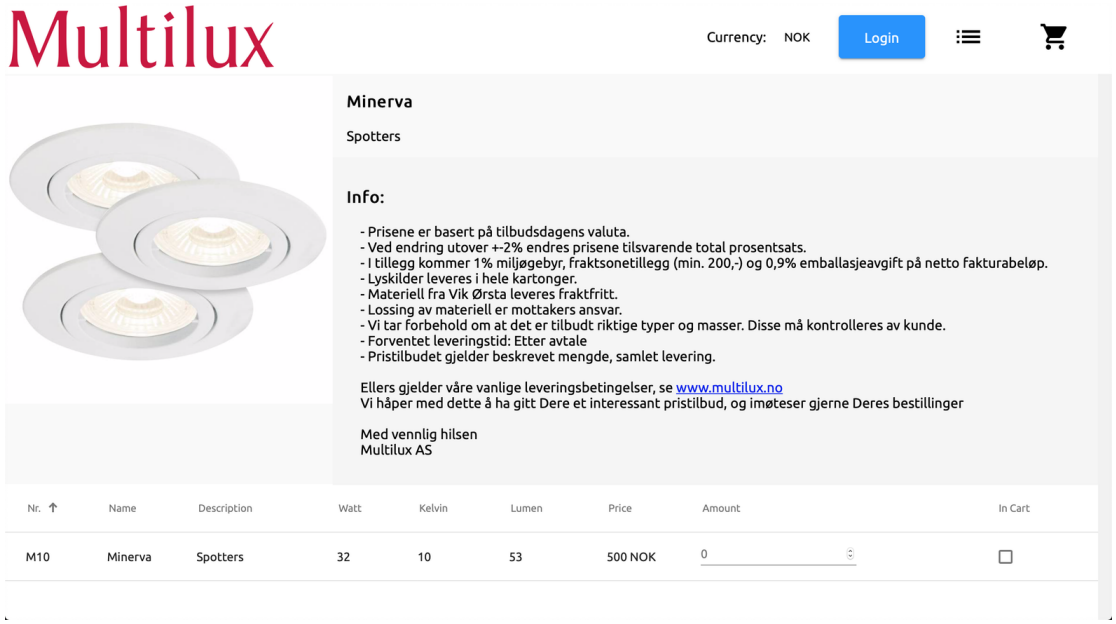
Kundene har ikke autoritet til å logge seg inn på web applikasjonen. I utgangspunket gir dette kunden kun tre alternativer som kan benyttes i navigasjonsbaren. Kunden kan bytte valuta (Figur 4.26 side 55), se lister over produkter og gå til handlekurven.

4.3.2.1 Produktliste



Figur 4.56: Produktlisten, liste over tilgjengelige produkter

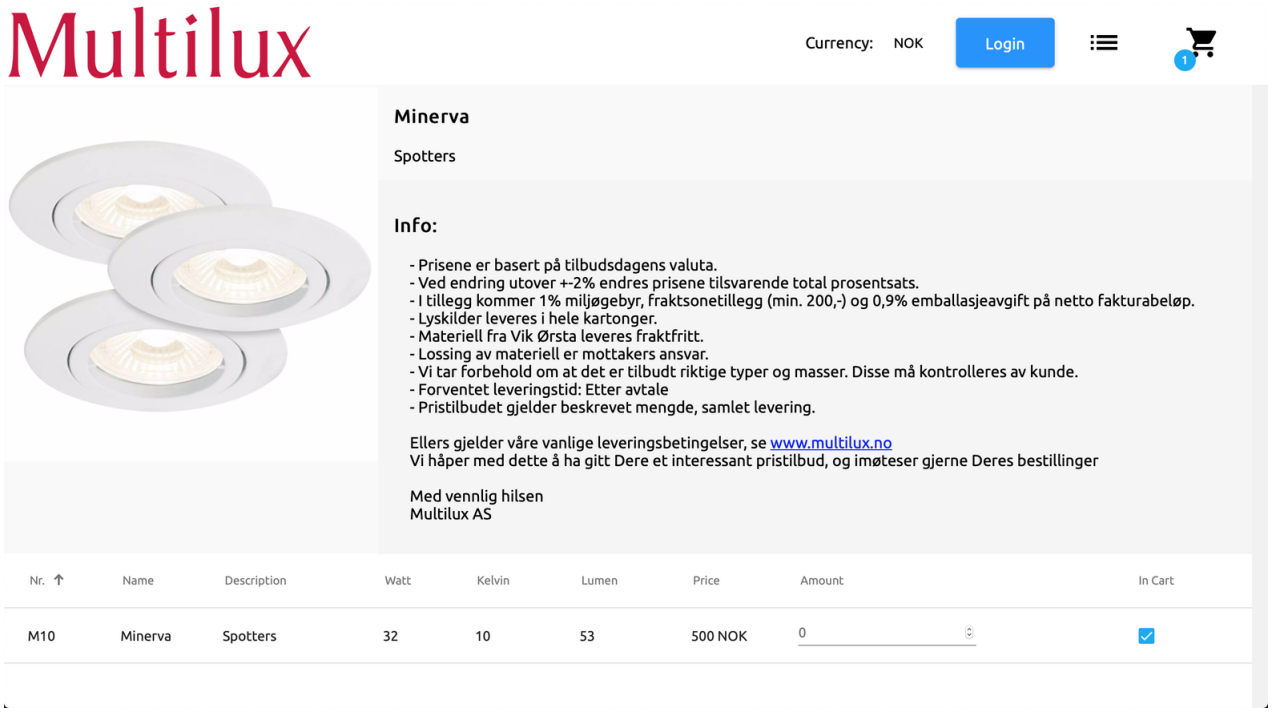
Skjermbildet ovenfor viser en liste med produktene som er tilgjengelig for kunden. Kunden kan allerede her velge ønskede produkter og legge det i handlekurven.



Figur 4.57: Produktlisten etter kunden har klikket på et produkt

Dersom kunden klikker på et produkt vil hovedseksjonen oppdatere seg med informasjon om produktet. Som vist ovenfor, har det dukket opp en liste under bildet. Listen inneholder produktet sin id, navn og beskrivelse. I tillegg vises produktets watt, kelvin, lumen og pris.

Kunden har mulighet for å velge antall produkter. Det er ikke satt noen grense på hvor mange antall produkter kunden kan velge. Ved å klikke på avmerkningsknappen, vil produktet bli lagt til i handlekurven.

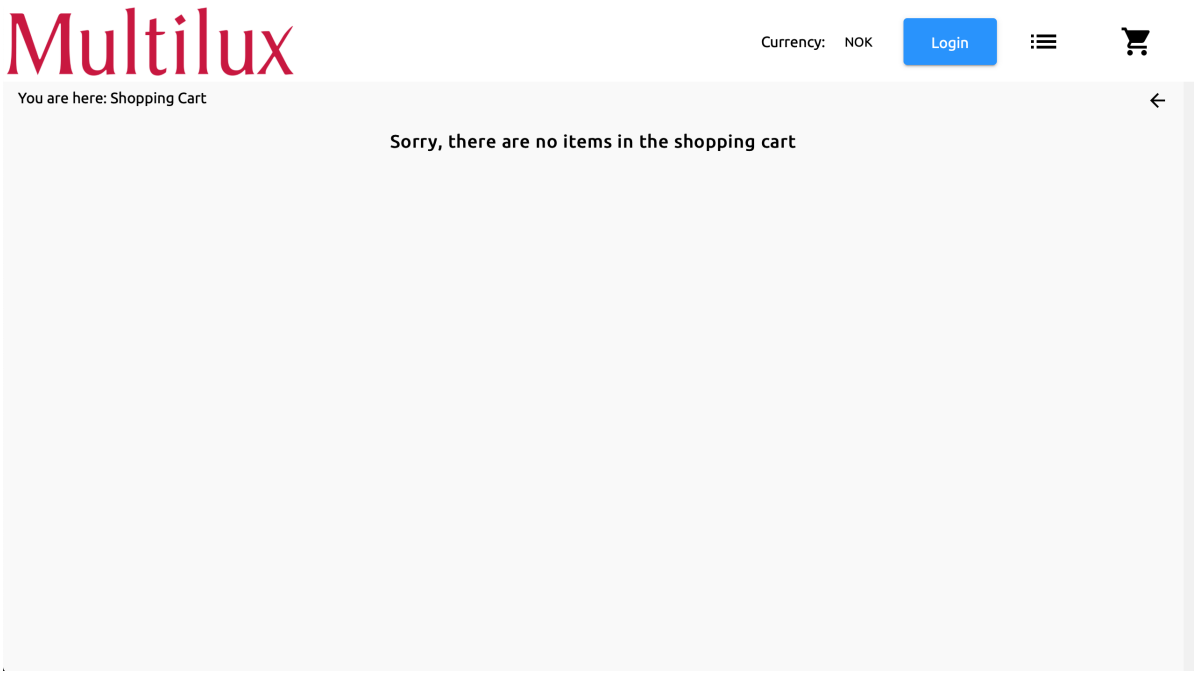


Figur 4.58: Avmerkningsknappen er huket av

Figur 4.59: Handlekurvikonet med en vare lagt til handlekurven

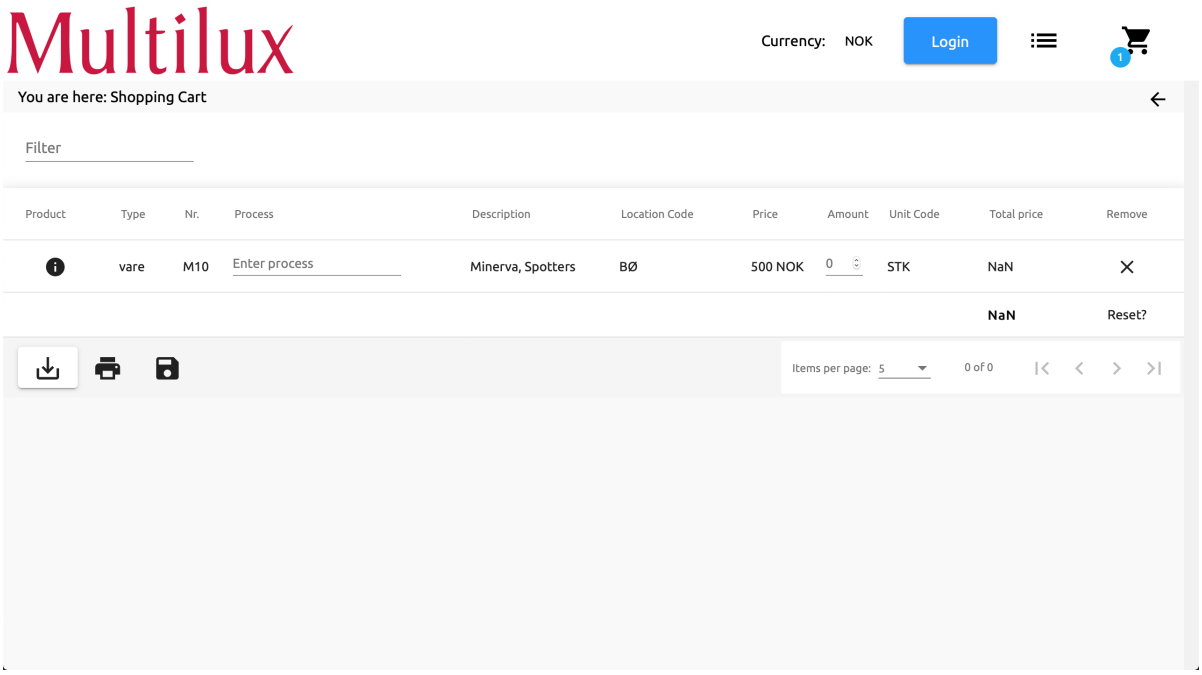
Ovenfor vises en demonstrasjon av resultatet etter at kunden har huket av for å legge i handlekurven. En blå rund-ing med tall kan sees på handlekurv-ikonet (Figur 4.59). I figurene ovenfor er det kun lagt til ett produkt i han-dlekurven.

4.3.2.2 Handlekurv



Figur 4.60: Handlekurven uten valgte produkter

Skjermbildet ovenfor viser handlekurven til kunden dersom den er tom. Det vises en tekst som forteller kunde at det ikke har lagt til noen produkter i handlekurven enda.



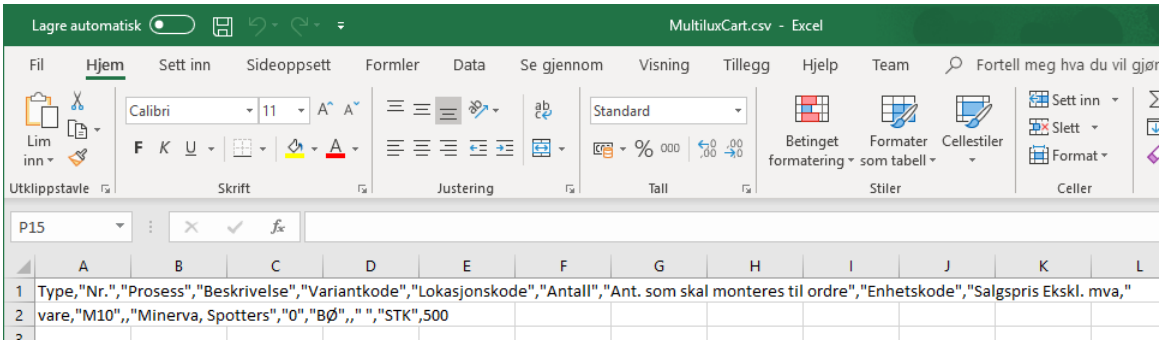
Figur 4.61: Handlekurven med ett valgt produkt

Figur 4.62: Informasjonsikonet

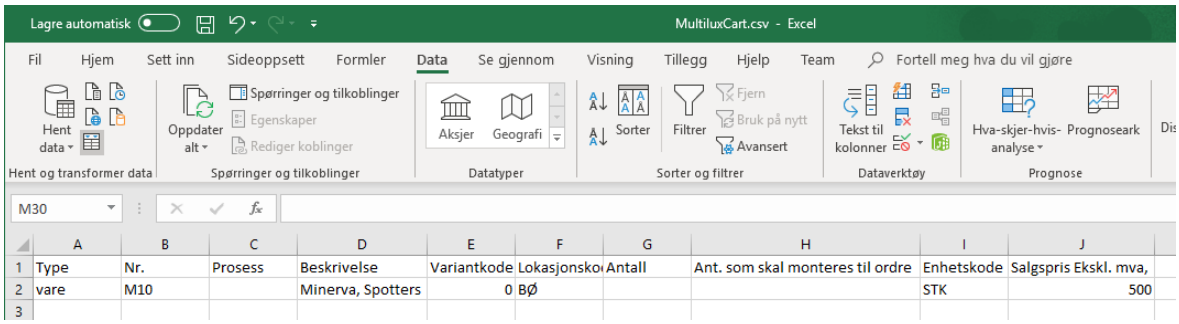
Figur 4.63: Lagre ikon

Figur 4.61 ovenfor viser handlekurven til kunden etter at et produkt er lagt til. Her har kunden mulighet til å klikke på informasjonsikonet (Figur 4.62) for å navigere seg til informasjonssiden for produktet. Det vises også type (vare), prosess, beskrivelse, lokasjon og pris.

Kunden har mulighet til å øke antallet av produkter og få en total pris. Oppdragsgiver gir kundene sine avtaler og rabatter, dette vil dermed bli tatt med i totalprisen. Kunden kan velge å fjerne enkelte produkter eller tømme hele listen med produkter. Dette gjøres ved å velge "Reset" alternativet. Nede til venstre vises tre ikoner for nedlasting (Figur 4.34 side 59), utskrift (Figur 4.35 side 59) og lagring (Figur 4.63).



Figur 4.64: CSV fil generert fra handlekurven



Figur 4.65: CSV filen formatert i Excel

Ved å klikke på ikonet for å laste ned CSV (Figur 4.34 side 59), vil kunden lagre produktene i en CSV fil på maskinen sin. Denne filen kan sendes til Multilux hvor de så åpner den i Excel (Figur 4.64) hvor man kan bruke "Tekst til kolonner" funksjonen for å gjøre dataen mer lesbar (Figur 4.65).

Multilux ønsket denne funksjonaliteten siden de da kan laste filen inn i deres Microsoft NAV-system. System deres genererer da et tilbud i PDF format som de da sender ut til sine kunder. Grunnen bak at de ville vi skulle bruke CSV filer var at selgeren kunne justere og se over produktene før de sender det tilbake til kunden som et tilbud.

Multilux Shopping Cart

Type	Nr.	Process	Description	Location Code	Price	Amount	Unit Code	Total Price
vare	M10		Minerva, Spotters	BØ	500	0	STK	NaN

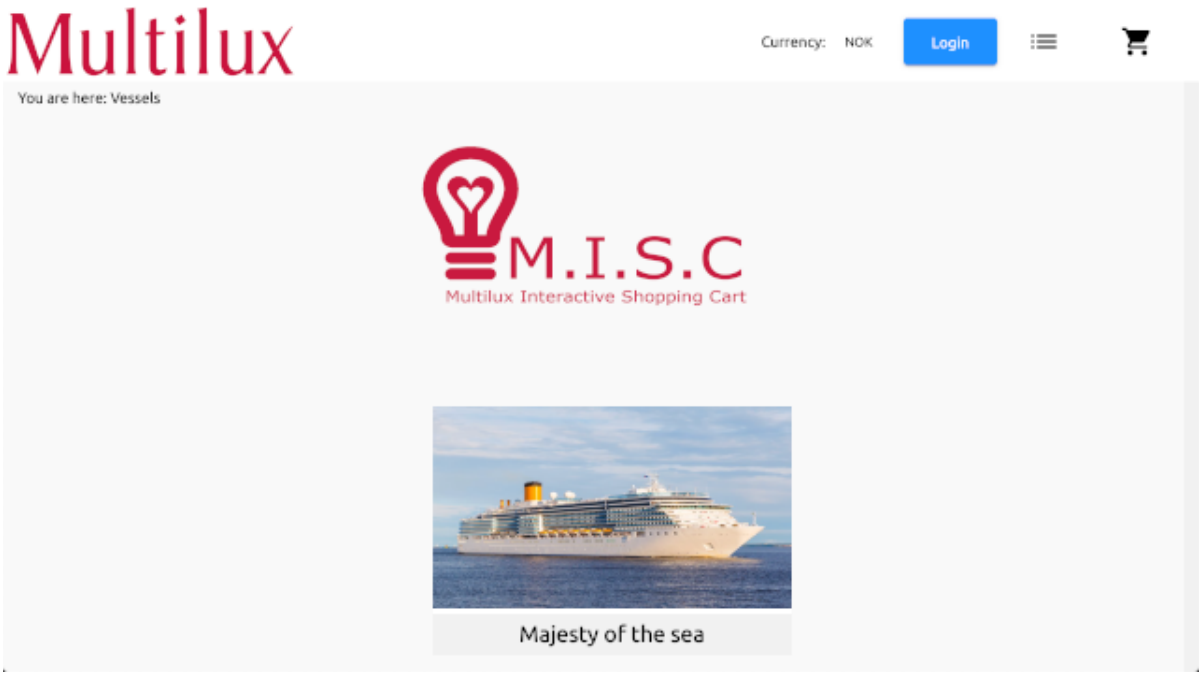
Total Price: NaN

Figur 4.66: PDF generert fra handlekurven

Ikonet for utskrift (Figur 4.35 side 59) gir kunden mulighet til å skrive ut handlekurven. Dersom kunden klikker på dette alternativet, vil det åpnes en PDF i et nytt vindu i nettleseren (Figur 4.66). Fra denne kan kunden skrive ut filen fra skriver.

Figur 4.63 viser ikonet som et alternativ for lagring. Den har foreløpig ingen funksjon. Baktanken var å la brukere laste ned prislisten som en json, slik at de senere kan laste det opp på en annen maskin.

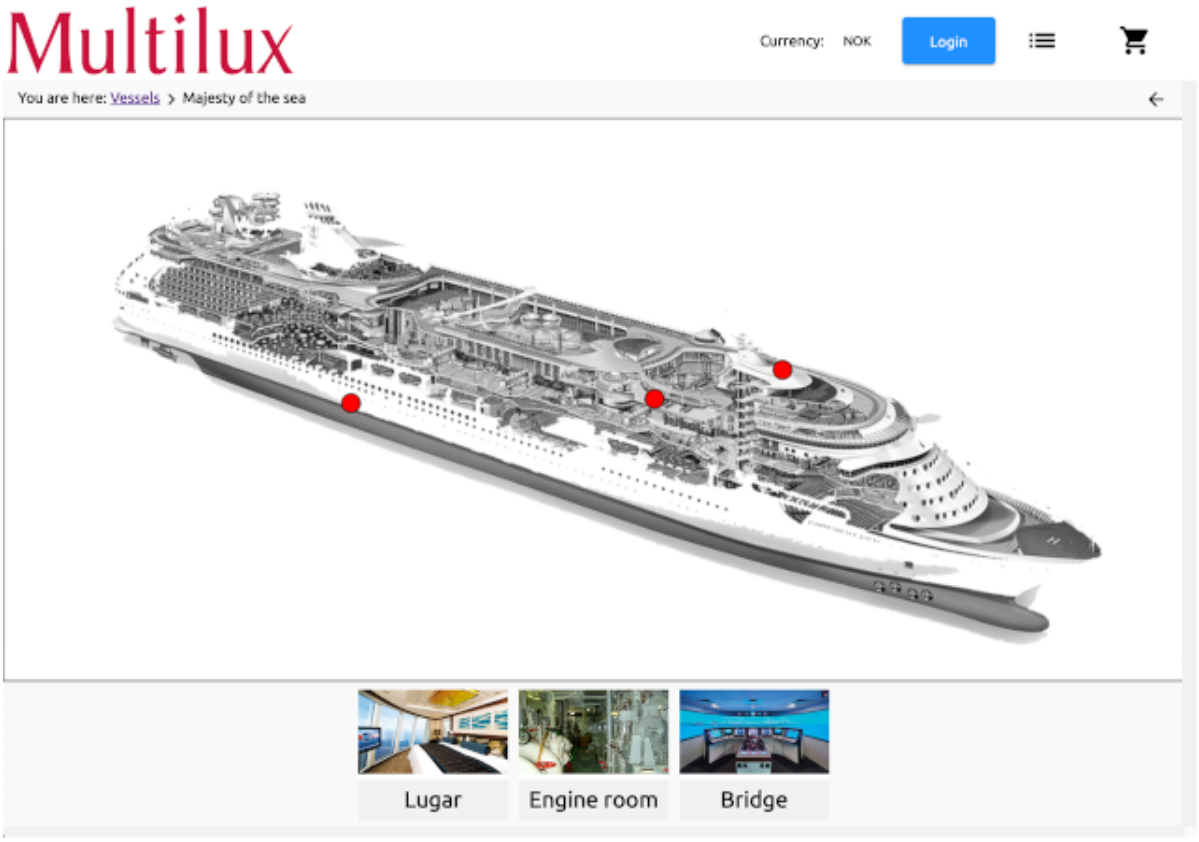
4.3.2.3 Forside



Figur 4.67: Forsiden sett fra kundens perspektiv

Skjermbildet ovenfor viser forsiden til web applikasjonen fra kundens perspektiv. Her vil alle fartøyene som er blitt lagt inn fra administrative brukere vises. Det er nå lagt til et fartøy som er tilgjengelig for kundene. Bildet av fartøyet har dukket opp i hoved seksjonen under logoen til web applikasjonen.

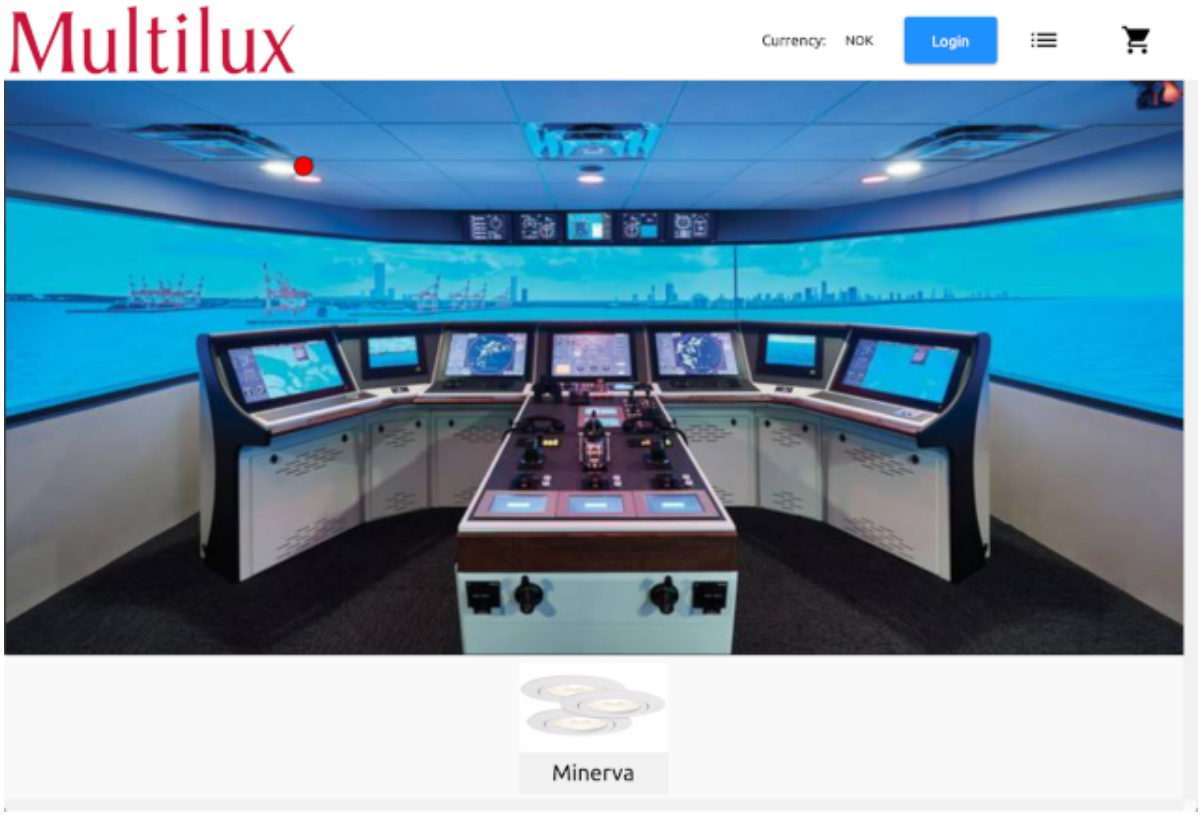
4.3.2.4 Velge rom



Figur 4.68: Plantegning med rom

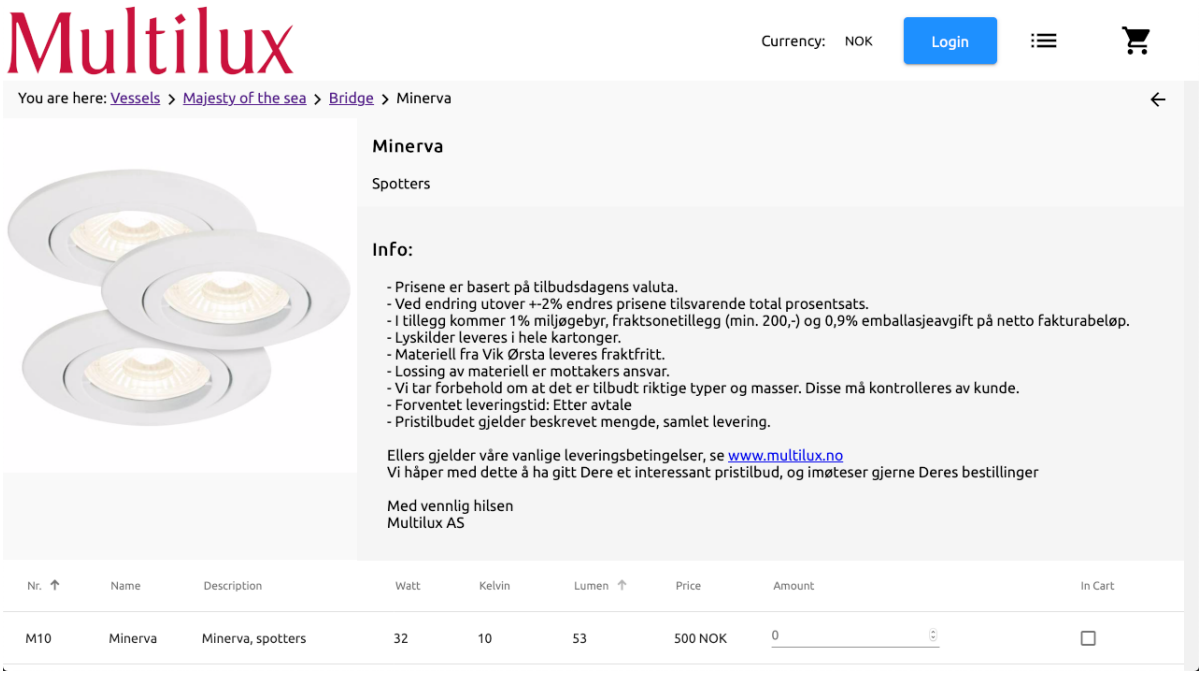
Når kunden klikker seg inn på et fartøy, vil plantegningen til det gitte fartøyet vises i hovedseksjonen. Skjermbildet ovenfor viser plantegningen med rommene. Kunden kan velge med å klikke på de røde knappene, eller rommene under plantegningen.

4.3.2.5 Velge produkt



Figur 4.69: Rommets produkter

Skjermbildet ovenfor viser innholdet i hovedseksjonen etter at kunden har valgt et rom. Rommet “Bridge” er valgt. Kunden kan velge mellom å klikke på den røde knappen, eller produktet under bildet.



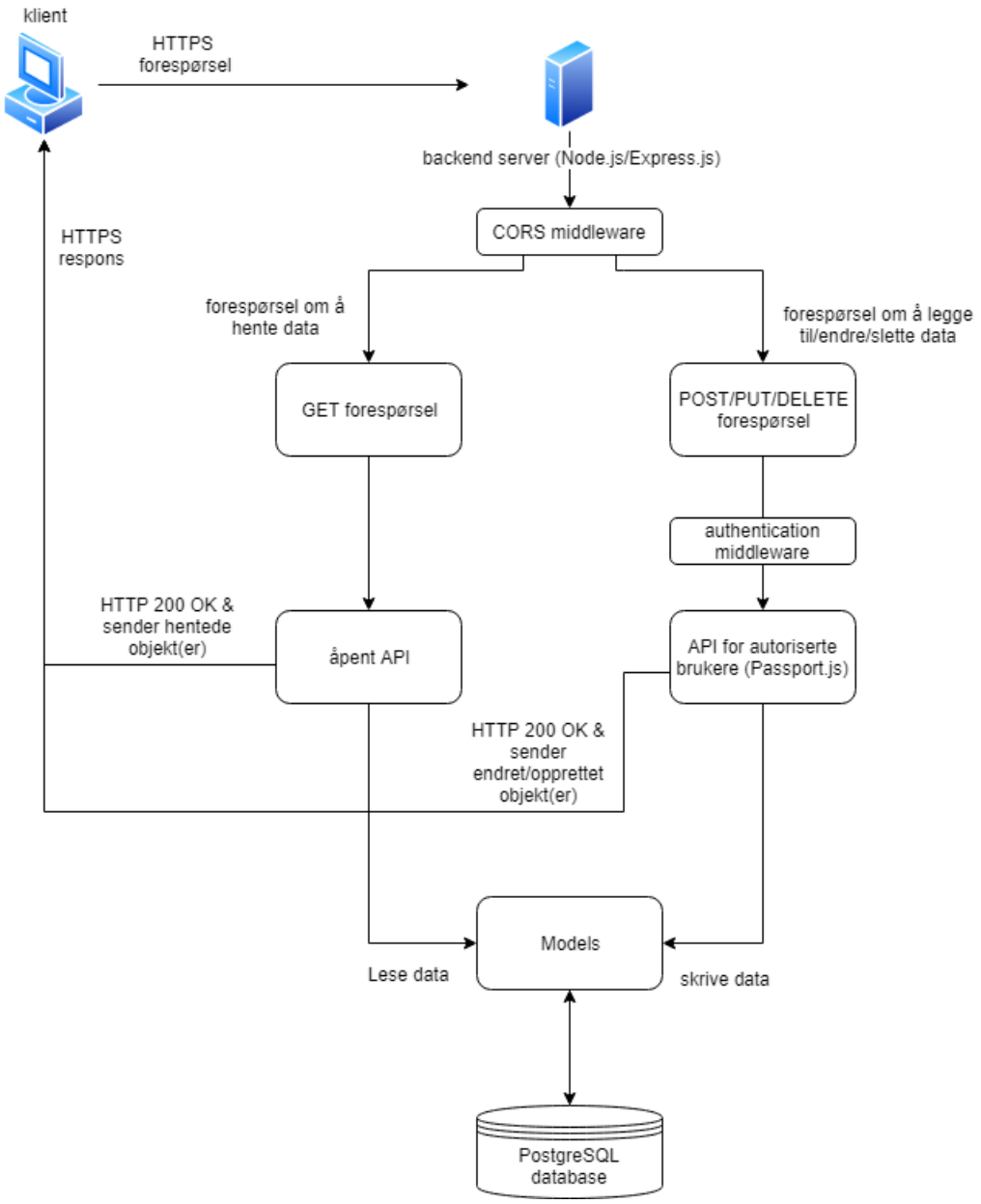
Figur 4.70: Produktinformasjon og underprodukter

Ovenfor vises et bilde av innholdet etter at kunden har valgt et produkt. Her vises diverse informasjon om produktet.

Kunden kan klikke på avmerkningsknappen for å legge til produktet i handlekurven. Dersom kunden kun skal ha dette produktet vil hendelsesforløpet heretter være likt som demonstrert under Produktliste (Side 75) og Handlekurv (Side 78).

4.4 Distribusjon og Administrasjon

4.4.1 Distribusjon



Figur 4.71: Backend styr

Vi har satt opp en midlertidig server for å vise resultatet til bedriften. Den har operativsystemet Debian (Linux) og inneholder alle delene av programmet: frontend, backend og database.

4.4.1.1 Backend

PM2 er en prosess manager for Node.js applikasjoner som ble installert for å gjøre det mulig å kjøre programmet i bakgrunnen som en service i systemet. Etter at man har lagt til applikasjonen i listen over prosesser så får man se prosess id, status for prosessen og minnebruk. Applikasjoner/prosesser som kjører fra PM2 vil bli startet på nytt automatisk om applikasjonen krasjer eller blir stengt ned og er en viktig grunn til å ta det i bruk.

Etter at PM2 ble konfigurert, ble Nginx satt opp som revers proxy slik at brukerne har en måte å aksessere applikasjonen på. Nginx deler opp backend og frontend slik at forespørsler om å gå til APIet går til riktig server og at forespørsler som går til nettsiden kommer til riktig fil.

4.4.1.2 Frontend

Frontend består av et Angular program og for å gjøre det produksjonsklart ble programmet bygd i produksjonsmodus for at koden skal kunne kjøre på web serveren skikkelig. Deretter overførte vi filene til serveren og la de i riktig mappe. Den vil da servere statiske nettsider til klienter.

4.4.1.3 Database

Databasen (2.7 side 20) er av typen PostgreSQL og vi måtte derfor installere PostgreSQL pakken på serveren. PostgreSQL servicen startet automatisk etter installasjon. Den ble så koblet opp til standard databasen for å teste tilkoblingen.

Etter en database med korrekte rettigheter og roller ble opprettet, måtte fjernstyring til PostgreSQL serveren aktiveres. Som standard er det kun lokale program og lignende som kan koble til PostgreSQL serveren, og det var derfor nødvendig å endre på konfigurasjonsfiler for at databasen skulle være åpen for trafikk fra resten av internett. Når både backend og database var satt opp ble tabeller migrert over fra backend til databasen.

4.4.2 Administrasjon

Vedlikehold vil av serveren (2.6 side 19) vil generelt bestå av oppdatering og oppgraderinger av operativsystemet. For å overvåke ressursbruken på serveren brukes blant annet PM2 for å sjekke minne- og CPU bruk for prosessene som kjører og andre innebyde kommandoer for overvåking av ressursene.

Vedlikehold av databasen vil i stor grad bestå av å ta backup av data ved oppretting av dump filer. Ideen bak denne metoden er å generere en tekst fil med SQL kommandoer. Når denne blir matet tilbake til serveren så vil den gjenskape databasen i samme stand som den var i når filen ble opprettet. PostgreSQL har en kommando pg-dump som utfører dette. Denne backup prosedyren kan bli utført fra hvilken som helst ekstern vert som har tilgang til databasen.

Om et av programmene har kode som skal oppdateres vil vi rekompilere programmet og overføre det til serveren igjen.

5 Drøfting

5.1 Evaluering av resultatet

5.1.1 Evaluering av MISC

5.1.1.1 Oppbygging

Vi hadde i utgangspunktet tenkt å skrive applikasjonen i Java, men gjorde en beslutning om å bygge systemet som en tilstandsløs web applikasjon istedet. Vi føler dette er mer skalerbart enn en ordinær Java applikasjon.

Med en tilstandsløs web applikasjon har vi en web frontend som sender HTTP forespørsler til et REST API som har en kobling med en relasjonsdatabase. Grunnen til dette er at vi er kjent med denne måten å sette opp systemer på, og at det er en fremtidrettet måte å løse problemet på. Ved å bruke et tilstandløst system kan frontend byttes ut uten at det påvirker resten av systemet. Dette gjør at man lett kan utvide systemet til flere plattformer enn bare web.

Når systemet er delt i to kan vi også mye lettere fordele arbeidsoppgaver og ansvarsområder. Vi har med dette kunne brukt god tid på utvikling av backend og database uten at det har påvirket den koden vi har skrevet i frontend. I starten brukte vi også testdata for å kunne teste data i frontend uten å påvirke backend.

Det at vi valgte PostgreSQL som database kommer av at noen av oss skal jobbe med PostgreSQL i fremtidig jobb og ville derfor bruke bacheloren til å skaffe seg nyttig og relevant kunnskap. Vi har ikke hatt noen problemer med PostgreSQL, og føler det er en god løsning for database.

Vi har en Debian server for testing av publisering. Vi valgte denne serveren fordi den tilhører et medlem i gruppen og var derfor allerede tilgjengelig. Valget med å bruke Nginx som web server og revers proxy, kommer av at det er en god konkurrent til Apache. Nginx krever lite ressurser mens den takler stor belastning og er derfor veldig skalerbart. Det er også lett å sette opp Nginx med HTTPS.

5.1.1.2 Brukervennlighet

Målet med prosjektet var å lage en interaktiv prisliste. Systemet skulle være brukervennlig, effektivt, sikkert og utvidbart. Brukervennligheten er viktig siden også de uten mye kunnskap om data skal kunne bruke systemet. Vi har forsøkt å lage systemet enkelt og veldig selvforklarende. Dette har vi gjort ved å begrense hvor mange forskjellige ting du kan trykke på på hver side. Når kunden kommer inn på hovedsiden har kunden i grunn bare tre valg: velge fartøy, gå til produktlisten eller gå til handlekurven. Dette har vi gjort bevisst med tanke på forutinntatt oppfatning (2.2.1 side 11) hvor vi forutser kundens mål og ønsker og legger opp for at det er lett å finne frem til det kunden er ute etter.

Det at applikasjonen er på web øker også brukervennligheten, siden brukeren kan bruke applikasjonen hvor som helst. Applikasjonen er ikke designet for mobil, dette kommer av at applikasjonen bare er for næringsliv. Layouten til applikasjonen er ellers basert på standarder og bruker også Angular Material for å videre følge design og layout standarder. Vi føler at dette gjør det lett for kunder å finne fram til det de leter etter og at kan finne funksjoner der de forventer at de er.

5.1.1.3 Design

Planen for design var å lage det så simpelt og stilrent som mulig. Dette kommer av at programmet skal være for næringliv og at funksjonalitet er mer viktig enn at det ser bra ut. Generell design på komponenter og elementer kommer fra Angular Material. Disse er ferdig designede komponenter, laget av profesjonelle grafiske designere. Dette gjør at vi kan bruke disse komponentene og anta at vi da følger en standard for godt design.

Grafikk Vi har designet et par egne ikon som brukes i applikasjonen. Dette kommer av at vi følte applikasjonen trengte et par kjennetegn og visuelle hjelpemiddler. Foreksempel hvis et fartøy har et udefinert bilde, har vi et standard bilde for det. Vi har også designet en egen logo som er inspirert av Multilux. Dette er for å gi MISC sitt eget særpreg og at vi trengte ikoner for fav og eventuelt .exe ikon.

Navigasjonsbar Navigasjonsbaren gir en fin oversikt over de viktigeste funksjonene. Vi føler at den er simpel og oversiktlig. En begrensning vi har nå er at ved forminskning av web vinduet til det minimale, vil designet til navigasjonsbaren endre seg. Dette resulterer i at logoen ("Multilux") vil bli plassert under den resterende navigasjonsbaren som vist i skjermbildet under.

Figur 5.1: Navigasjonsbaren i et lite vindu

I forhold til det som var planlagt hadde vi ikke tenkt å ha produktliste og valuta i navigasjonsbaren. Dette ble lagt inn etterhvert da vi så at dette kunne være praktisk.

Dialoger Dialogbokser gjør at vi kan vise et skjema uten at det ødelegger for layouten til selve applikasjonen. Ulempen med dialogbokser er at det blir ofte litt tungvindt når du må innom en dialogboks hver gang du skal inn med input.

På de fleste inputfelt er der ingen sjekk på om det som blir skrevet inn er godkjent. Det er sjekk for dette i backend, men mangler sjekk i frontend. Dette resulterer i at brukeren ikke blir informert om at den dataen brukeren taster inn er feil og at forespørselen ikke har gått gjennom.

Innlogging Innlogging var for vår del ganske greit siden applikasjonen i seg selv skulle være åpen, og de som skulle kunne logge inn var administratorer. Vi trengte derfor ikke å implementere roller for å skille mellom brukere og administratorer. Vi hadde ingen begrensninger under utikling, men vi har en begrensning for videre utvikling hvis oppdraggiver ønsker å ha flere roller i systemet.

Dette kommer av at vi ikke har lagt til muligheten for å sette rolle på brukeren. Grunnen til dette er at det ikke trengtes utifra de kravspesifikasjoner vi fikk. Der er bare en rolle som var nødvendig i dette systemet; administrator. Muligheten for å legge til flere roller var derfor aldri nødvendig. Dette kan løses ved å endre på bruker tabellen i databasen og sette rollene på de eksisterende brukerene manuelt.

Produktliste Vi trengte produktlisten slik vi kunne ha en plass å plassere alle produktene. Dette kommer av at produktene er unike og det ville da vært ugunstig hvis man måtte lage nye produkt for hvert rom. Vi gikk heller for en løsning der du kan velge produkt fra en samling av produkter. Ved å ha det slik kan også kunder som vet navnet på de produktene de skal ha filtrere produkter i listen og finne produktet direkte uten å måtte gå gjennom veiviseren. Produktlisten var ikke planlagt fra starten av, men ble lagt til etter at vi så det var nødvendig å ha en side der vi hadde oversikt over alle produktene.

Handlekurv Handlekurven er siste steget før kunden laster ned CSV. Grunnen bak handlekurven er at vi må ha en plass å vise de produktene som er valgt. Dette er slik kunden kan se over de valgte produktene og eventuelt justere verdier før kunden skriver ut CSV og sender det til selger. Handlekurven var en av de komponentene som ble ofte endret, grunnet misforståelser fra oppdraggiver. Dette kom mye av at vi ikke hadde kunnskap om fagfeltet til oppdraggiver. Oppdragsgiver måtte derfor gi oss en god forklaring på hvilke felt som måtte være med og hvorfor.

Innstillinger Innstillinger er laget for å kunne utvides ved en senere anledning. Den er også laget med tanke på enkelhet og effektivitet. Innstillinger var ikke planlagt fra starten av og oppsto først når vi så at vi måtte ha et sted å plassere innstillingene for valuta. Vi bestemte oss for å gå for en utvidbar løsning istedet for å lage én innstilling bare for valuta.

Veiviser Oppdragsgiver ønsket å ha en slags veiviser der kunden kunne finne fram til et produkt gjennom å velge fartøy og rom. Vi hadde i utgangspunktet egentlig tenkt at rutene for de forskjellige stegene i veiviseren skulle være enkle.

For eksempel skulle oversikten over rom egentlig ha ruten "/rooms". Vi gikk vekk fra dette siden man da ikke kunne linke direkte til et rom eller produkt. Det vi endte opp med var at du hadde rute og idene for hvilke fartøy, rom og produkt du hadde valgt. Med dette kan foreksempel en selger kopiere linken og sende det til en kunde. Kunden kan da gå direkte til det fartøyet, rommet eller produktet selgeren ville vise kunden.

Canvas med flyttbare prikker Vi gikk for en ordinær HTML-canvas element fordi vi ikke tok oss tid til å finne et rammeverk for dette. Grunnen til at vi ikke hadde tid var at det var viktig å få dette på plass tidlig slik vi kunne gjøre klart en prototype.

Vi vurderte eventuelt å gå for JQuery siden det har funksjoner for dragable objekt, men vi ville ha full kontroll av elementet noe vi ikke får med JQuery. Hvis vi hadde gått for JQuery hadde vi hatt et par problemer med endring av størrelse på vindu. Når vi bruker HTML-canvas element kan vi lage egne funksjoner for å regne ut kollisjon, samt x og y i forhold til dimensjonene på vinduet.

Den største begrensningen med canvas elementene er at det bare er en person som har jobbet med dem. Dette vil bety at hvis noe er galt med canvas er det mest sannsynligvis bare denne personen som kan rette det. Eventuelt må noen bruke mange timer på å sette seg inn i koden bak canvas elementet.

Det var også egentlig planlagt at vi skulle lage et felles canvas element for begge sidene som benytter seg av det, men koden på de to forskjellige canvasene ble så ulik at vi endte opp med å lage to ulike canvas komponenter.

5.1.1.4 Biblioteker

Vi har prøvd å bruke minst mulig eksterne biblioteker, men går for de bibliotekene som er sett på som standarder. Dette fører til at vi har veldig god kontroll på den koden vi har, og hvis det er noe vi er usikre på er det mye informasjon på nettet om de brukte bibliotekene.

5.1.2 Evaluering av Angular som rammeverk

Vi valgte å bruke Angular som web-rammerk. Hovedgrunnen bak dette er at vi hadde ønsket å benytte anledningen til å lære oss et nytt web rammeverk. Samtidig er Angular et at de mest populære rammeverk for web utvikling. Utvikling av frontend i Angular er oversiktlig og komponentene påvirker ikke hverandre. Vi opplevde sjeldent at filer og scripts påvirket hverandre, noe som førte til mindre bugs.

Angular har veldig god dokumentasjon og det finnes mange guides og guider på hvordan man kan lage forskjellige komponenter og tjenester. Dette gjorde arbeidet mye lettere.

Det beste med Angular som et rammeverk for oss er at det er lett å redigere komponenter uten at det påvirker resterende kode i prosjektet. Dette er ideelt dersom kunden vil endre på noe.

5.1.3 Evaluering av Node.js som servermiljø

Hovedgrunnen til at vi gikk for Node.js som servermiljø er at det er et populært servermiljø for utvikling av serverprogramvare. Man slipper også å kombinere forskjellige programmeringsspråk da JavaScript blir brukt til både frontend og backend.

Node.js har også gode rammeverk for utvikling av REST API (2.5 side 19). Til dette valgte vi Express.js som er et godt dokumentert rammeverk.

På en annen side kan APIet til Node.js være ustabilt. Grunnen til dette er at det skjer endringer i APIet som deretter må endres i vår kodebase for at det skal være kompatibelt. JavaScript er heller ikke like robust som andre programmeringsspråk med tanke på support av bibliotek og utføring av andre enkle oppgaver [33].

5.1.4 Evaluering av sikkerheten i systemet

5.1.4.1 Tofaktorautentisering

Det hadde vært bra for sikkerheten i systemet om vi brukte tofaktorautentisering (2.4.2 side 19) for å autentisere brukere, men på grunn av tidsbegrensninger har vi ikke implementert dette. Det er også ikke et så stort behov i dette systemet, da vi ikke holder på sensitiv informasjon. Verste scenariet er om man sletter databasen, men i dette tilfellet har vi backup av databasen.

5.1.4.2 Passordsikkerhet

For å holde passord i databasen sikre har vi brukt bcrypt som passord hashing funksjon (2.4.2 side 18). Da unngår man at passord ligger i klartekst i databasen. Grunnen til at vi valgte bcrypt er at det er en langsom metode. I tillegg til å inkorporere en 'salt' for å beskytte mot 'rainbow table' angrep, er bcrypt en adaptiv funksjon: over tid kan iterasjonstallet økes for å gjøre det langsommere, slik at det forblir motstandsdyktig mot 'brute force' søkeangrep, selv med økt beregningskraft. Siden bcrypt er en godt kjent metode anser vi dette som en god løsning for sikring av passord i databasen.

5.1.4.3 HTTPS

Vi har også sikret transportlaget ved at vi bruker HTTPS (2.4.2 side 18). Dette sørger for at informasjon som går frem og tilbake mellom server og klient er sikret. Dette innebærer at dersom noen overvåker nettverkstrafikken til en bruker av nettstedet, kan de ikke fange opp brukernavn og passord når det blir sendt til serveren.

5.1.5 JSON Web Token

Vi har tatt i bruk JWT for å kunne autentisere brukere (2.4.2 side 18). Dermed kan vi skille hvem som skal ha tilgang til å endre data i databasen og hvem som bare skal ha tilgang til å hente data.

5.1.5.1 Informasjonskapsler

Informasjonskapsler blir brukt for å kunne lagre variabler. Det er lagret lokalt på datamaskinen til brukeren. Disse variablene vil inneholde en brukers JWT for å kunne sende en autorisert forespørsel til server, og det vil også være en variabel for å lagre brukernavn.

5.1.6 Evaluering av distribusjon og administrasjon

Måten vi distribuerer frontend, backend og databasen på nå er via en Linux server. Der er flere grunner til dette valget. Linux er mer stabilt enn for eksempel Windows servere og har bedre sikkerhet. Det er også mindre ressurskrevende og krever derfor ikke at man ofte må oppgradere hardware. På en annen side er Linux servere litt mindre brukervennlige, da man ofte jobber fra et terminalvindu istedet for et grafisk brukergrensesnitt. Det er også ikke alltid tilfelle at alle Linux distribusjoner har langtidssupport. [27]

5.1.7 Evaluering av responsiviteten

Applikasjonen er generelt rask, men vi bruker innlasting animasjoner som indikasjoner på at den jobber på de plassene den henter mye data. Dette er slik at brukeren ikke skal tro at systemet har hengt seg opp. På den andre siden mangler vi indikasjoner på at en forspørsel har blitt sendt, motatt, avbrutt eller feilet. Dette er noe vi eventuelt kan legge til ved en senere anledning.

5.1.7.1 Støtte av nettlesere

Vi har brukt Google Chrome som hovednettleser, men det fungerer også i Mozilla Firefox. Vi har ikke tatt hensyn til at programmet skal fungere i Internet Explorer eller Edge.

5.1.7.2 Android og iOS

Vi har ikke hatt noe fokus på at programmet skal se bra ut på mobil. Dette kommer av at programmet skal brukes i næringsliv og kundene vil da derfor mest sannsynlig sitte på desktop når de bruker programmet.

5.1.8 Evaluering av database-designet

Designet av databasen har endret seg flere ganger iløpet av prosjektet. Dette kommer av at vi lagde en ganske kompleks struktur i staren, men innså etterhvert som vi utviklet at vi kunne simplifisere systemet. Noen endringer kom også av misforståelser eller endringer fra oppdragsgiver.

5.2 Evaluering av prosjektet

5.2.1 Utviklingsmetode

I Teoretisk grunnlag (2.1.2 side 10) ble Scrum nevnt som en utviklingsmetode innad i prosjektet. I underkapittel (5.2.3.4) (Avvik fra utviklingsmetode), har vi poengtert grunnlaget på hvorfor vi ikke fulgte denne utviklingsmetoden.

Selv om vi har falt bort fra flere av Scrum sine prinsipper, er det noen vi har valgt å ta i bruk. En av prinsippene vi har brukt er product backlog. I prosjektet har vi delt opp diverse arbeid i oppgaver. Disse oppgavene har vi samlet i en product backlog. Etterhvert som oppgavene blir løst, blir de sett over og godkjent av de andre medlemmene i gruppen. Etter at oppgaven er blitt godkjent blir den flettet sammen med resten av prosjektet.

5.2.2 Håndtering av utfordringer

Gruppen har håndtert alle problemer og utfordringer på en demokratisk måte. Vi presenterer problemet på vår kommunikasjonsplattform også diskuterer vi oss frem til en ideell løsning. Dette fører til at alle føler seg inkluderte, noe som øker motivasjon. Vi føler dette har vært en bra måte å løse middels til store utfordringer.

5.2.3 Avvik

5.2.3.1 Distribusjon

Oppdragsgiver ville også i starten ha mulighet for å ha applikasjonen som en .exe applikasjon. Dette var slik at de kunne sende kundene sine en .exe applikasjon de kunne ha på datamaskinen sin. Vi oppnådde dette ved å importere et eksternt bibliotek som heter Electron (3.2.1.1 side 28).

Etterhvert som oppgaven pågikk har vi ikke testet at programmet fortsatt kjører gjennom Electron, noe det ikke gjør i nyeste versjon. Ettersom vi endte opp med en web løsning er det derimot ikke heller nødvendig med en .exe applikasjon siden kundene til oppdragsgiver uansett trenger tilgang til internett for å kunne bruke applikasjonen.

5.2.3.2 Samarbeid

Det har vært få utfordringer når det gjelder samarbeidet i prosjektet. En utfordring vi møtte var medlemmenes tilgjengelighet. Ettersom enkelte medlemmer hadde arbeid ved siden av, ble det vanskelig å planlegge tidspunkt alle var tilgjengelige på. Dette resulterte som oftest i at kun enkelte av medlemmene ble med i de aktuelle møtene. Vi har uansett holdt god kommunikasjon over nett, så vi har ikke vært veldig avhengige av å måtte møtes fysisk.

5.2.3.3 Avvik i produktet

Vi hadde originalt tenkt at når kundene holder musen over prikkene for valg av rom eller produkt så skulle det komme opp en tooltip. Denne skulle inneholde produktnavn og bilde slik at brukerne enkelt kunne se hvilke rom eller element prikkene representerer. På grunn av tidsbegrensninger fikk vi desverre ikke tid til å implementere dette.

Vi mangler også en “Add new product” knapp ved siden av rommene i products siden. Vi har FAB knapp, men det skulle også vært en ved siden av rommene slik at designet gjennomgår hele applikasjonen. Grunnen til denne mangelen må ha vært at vi allerede har FAB knapp så vi la aldri merke til det før på slutten av prosjektet. Vi har derfor ikke hatt tid til å fikse dette før fristen.

5.2.3.4 Avvik fra utviklingsmetode

I utgangspunktet var planen å følge den smidige arbeidsmetoden på best mulig måte. Det viste seg at denne metoden ikke ble like gunstig som vi i utgangspunktet hadde sett for oss. Det er flere grunner til at det ikke ble tatt i bruk smidige metoder i vårt prosjekt. Flere av prosjektets medlemmer har lang vei til skolen, som også ble vårt felles møtested. Med både tid og kostnader i betraktning, ble det bestemt at vi heller skulle samarbeide over nettet.

En vesentlig del i Scrum, er at utviklingsteamet ikke har definerte roller, noe vi har motstridt. Bakgrunnen for at vi trengte definerte roller var i utgangspunktet at vi har valgt å bruke rammeverk og utviklingsmetoder, som vi i utgangspunktet hadde lite kunnskap om. Det ville dermed vært tidskrevende dersom alle medlemmene skulle tilnærmet seg lik kunnskap innenfor alle områdene. Vi valgte derfor å tildele hverandre roller. Tankegangen vår ble dermed "det er bedre å ha mye kunnskap om noe, enn lite kunnskap om alt".

Vi har på mange måter fraskrevet oss det meste som har med smidige metoder å gjøre. Det har heller ikke blitt holdt de andre møtene som er essensielle innenfor Scrum. Det har vært vanskelig å opprettholde en tett dialog med oppdragsgiver. Dette er fordi oppdragsgiver befinner seg henholdsvis i Bø, Telemark. Det var da vanskelig å planlegge møter og det meste kommunikasjon foregikk over mail.

5.2.4 Videre utvikling

Akkurat nå jobber Multilux med å skaffe seg rettigheter på bilder de kan bruke i programmet. Dette betyr at programmet står stille akkurat nå. Det er ellers store muligheter for å utvide programmet siden alt av kode er i individuelle komponenter. Det er funksjonaliteter som ikke er på plass ennå, men som vi tenker å legge til i løpet av sommeren når oppdragsgiver har skaffet bildene som trengs.

5.2.5 Hva har vi lært?

I løpet av prosjektet har vi fått god innsikt i hvordan utviklingsprosjekter fungerer i praksis. Vi har også fått tilnærmet god kunnskap innenfor Angular og Node.js. Vi har også lært oss hvordan vi kan samarbeide som et lag og utveksle kunnskap oss i mellom.

På den andre siden har vi også lært litt om hvor vanskelig det er å faktisk følge arbeidsmetodikker. Det er veldig lett å droppe det når du mener du kan gjøre arbeidet mye raskere hvis du gjør det på din måte. Problemet med dette er at ikke alle er med på hvordan du gjør det og at du som gjør det på din måte må eventuelt bruke mye tid på å forklare hvordan du gjør det. Sånn totalt sett ender du opp med å bruke mer tid ved å gjøre det på din måte fordi noen av de du er på gruppe med ikke klarer å henge med.

6 Konklusjon

Hensikten med prosjektet har vært å lage en digital løsning for en interaktiv prisliste der kunden selv kan finne produkter og dens tilhørende informasjon som spesifikasjoner og pris. Ved prosjektstart diskuterte vi følgende problemstilling: "Hvordan kan vi lage en brukervennlig og sikker interaktiv prisliste som holder seg oppdatert med dagens utvalg, spesifikasjoner, priser og valuta?" Dette er da hovedproblemstillingen vi har besvart med de metodene og resultatene som er presentert i denne rapporten.

Resultatene viser at prosjektet:

- Kan enkelt oppdateres med nytt innhold.
- Oppfyller kravene stilt av oppdragsgiver.
- Har brukervennlighet i fokus.
- Har innlogging slik det kan skilles mellom hvem som bare kan lese og hvem som kan redigere.
- Har et tilstandsløst REST API som kan kobles opp mot andre systemer ved eventuelle utvidelser.
- Bruker standarder innen kryptering.
- Kan settes opp med HTTPS for sikre tilkoblinger.
- Tilbyr funksjonalitet som eventuelle lignende program ikke har. For eksempel CSV for eksportering av produkter.
- Reduserer unødvendig etterarbeid hos både kunder og selgere.

De metodene og teknologiene vi har brukt er ikke en fasit for denne oppgaven, siden det finnes mange forskjellige måter å løse den på. Vår løsning er et valg vi har gjort utifra egne ønsker, erfaringer og meninger. Før vi bestemte oss for hvilke metoder og teknologi vi skulle bruke, gjorde vi en del undersøkelser på om de oppfylte alle kravene vi og oppdragsgiver stilte til oppgaven. Vi har derfor kunne stått bak valget vårt gjennom hele prosjektet. Det har vært vesentlig for oss å ha kunnskap om prinsippene for et tilstandsløst system. Selv om vi var nye til rammeverket, kunne vi prinsippene bak et godt system og baserte oss på det. Selv om vi synes resultatet ble bra er dette bare en av mange måter å løse oppgaven på.

Prosjektet har gitt gruppen masse erfaring innen det å jobbe i prosjektgruppe. Selv om vi ikke har klart å holde en god smidig arbeidsmetodikk har gruppen jobbet godt sammen. Vi jobbet inkrementelt og oppgaver kunne derfor delegeres til gruppemedlemmene uten at det var nødvendig å vente på at annen kode var ferdig før de kunne begynne på sitt. Gruppen har fått uvurderlig kunnskap innen versjonskontroll, og prosjektet har også gitt oss erfaring i hvor utfordringene med å avtale møter som passer for alle når avstandene er store.

Dette prosjektet har vært svært lærerikt. Vi har lært oss et nytt rammeverk som ingen av oss kunne noe om i starten av prosjektet. Vi har med dette innsett at selv om man ikke kan selve rammeverket kan man bruke prinsippene vi har lært gjennom studiet til å forstå og tilnærme seg baktanken til rammeverket. På grunn av denne erfaringen har vi nå et bedre utgangspunkt for å lære oss nye rammeverk. Prosjektet dekker også viktige felt i næringlivet, som: web-utvikling, API, database og distribusjon. Dette er kunnskaper vi kommer til å ta med oss videre etter endt studie.

Som drøftingen og evalueringen viser, er vi fornøyd med selve løsningen på oppgaven og det resultatet vi kan vise til. På den andre siden skulle vi ønske at vi hadde hatt større fokus på en smidig arbeidmetode og innarbeidet en fast rutine tidlig slik at gruppemedlemmene kunne ha lagt opp sine tidsplaner basert på det. Vi håper at oppdragsgiver får god nytte av systemet.

Vi kan med denne rapporten konkludere at det utviklede systemet oppfyller krav fra oppdragsgiver, samt at prosjektets problemstillinger er besvart med presentasjon av metode og resultat.

Vedlegg

A Forprosjektrapport

FORPROSJEKT - RAPPORT

FOR BACHELOROPPGAVE

NTNU
Kunnskap for en bedre verden

TITTEL:
Interaktiv prisliste belysning

KANDIDATNUMMER(E):
Olav Telseth, Simon Kalsnes Synnes, Einar Weltan og Kristin Hagen

DATO:	EMNEKODE: *	EMNE:	DOKUMENT TILGANG:
	IE303612	**Bacheloroppgave (Data, Elkraft)**	- Åpen

STUDIUM:	ANT SIDER/VEDLEGG:	BIBL. NR:
BACHELOR I INGENIØRFAG - DATA	/	- Ikke i bruk -

OPPDRAGSGIVER(E)/VEILEDER(E):
Multilux

OPPGAVE/SAMMENDRAG:
Oppgaven går ut på å lage en interaktiv prisliste der kunden selv kan finne produkter og dens tilhørende pris. Produktene er belysning til båter/shipping, og er fordelt på de forskjellige typer fartøy (Cruice, supply, fiskebåt etc.).

Postadresse
Høgskolen i Ålesund
N-6025 Ålesund
Norway

Besøksadresse
Larsgårdsvegen 2
Internett
www.hials.no

Telefon
70 16 12 00
Epostadresse
postmottak@hials.no

Telefax
70 16 13 00

Bankkonto
7694 05 00636
Foretaksregisteret
NO 971 572 140

Denne oppgaven er en eksamensbesvarelse utført av student(er) ved NTNU i Ålesund.

INNHOLD

1 INNLEDNING

Multilux har hovedkontor i Bø i Telemark og er en betydelig leverandør av belysning for samferdsel, infrastruktur, idrettsarenaer, samt en mengde andre utendørs bygg og anlegg.

De har lenge brukt Excel for salg av lys på båt, og de ønsker nå å gå over til en interaktiv prisliste. Målet med dette er at kunden selv kan finne produkter og dens tilhørende pris, og dermed avlaste selgerne i Multilux AS. Prislisten tar utgangpunkt i belysning til båter/shipping, og er fordelt på de forskjellige typer fartøy (Cruise, supply, fiskerbåt etc.).

Grunnen til at vi valgte denne oppgaven er at vi har tidligere erfaringer med interaksjon mot 2D modeller, distribuerte og mobile applikasjoner samt behandling fra API. Vi synes det er spennende med en interaktiv 2D modell som en søkemotor for opplisting av mulige produkt i en spesifikk del av båten.

2 BEGREPER

- Frontend - den delen av programvaren som ligger nærmest brukeren. Koden som former det du ser på skjermen og interaksjoner mellom disse elementene.
- Backend - Behandler dataflyten mellom data og det visuelle
- Server - En data som står og kjører backenden til programmet vårt
- Database – Tabeller med data som vi lagrer
- Merge-request – at noen ser over før du merger inn i master

3 PROSJEKTORGANISASJON

3.1 Prosjektgruppe

Studentnummer(e)
772449
460137
460135
130329

3.1.1 Oppgaver for prosjektgruppen – organisering

Gruppen vil fungere demokratisk og vil diskutere seg fram til hver beslutning. Vi har en prosjektleder, men denne rollen vil fungere mest som en person ansvarlig for fremgangen i prosjektet siden vi ønsker en mer demokratisk tilnærming til beslutningstagning.

3.1.2 Oppgaver for prosjektleder

Utføre oppgaver, delegere oppgaver, holde framdrift i prosjektet og vise framdriften til oppdragsgiver.

3.1.3 Oppgaver for øvrige medlem(mer)

Utføre arbeidsoppgaver, gjøre beslutninger sammen med andre gruppemedlemmer.

Ansvarsområder for øvrige medlemmer blir delegert mens prosjektet pågår. Dette kan være ansvarsområder som: backend ansvarlig, frontendansvarlig eller database ansvarlig osv.

3.2 Styringsgruppe (veileder og kontaktperson oppdragsgiver)

4 AVTALER

4.1 Avtale med oppdragsgiver

4.2 Arbeidssted og ressurser

Siden Multilux er plassert i Bø i Telemark vil vi ikke ha mulighet til å jobbe i deres lokaler uten å ta fly eller lignende opp dit. Vi kommer derfor til å jobbe mest fra skolen eller hjemme.

Multilux vil gi oss tilgang på de ressursene vi trenger, som bilder og ekspert kunnskap.

Vi har kontakt med Stian Nielsen som er Distriktssjef for Vestlandet i Multilux.

Angående datasikkerhet skal det kun være Multilux som har administrativ tilgang til systemet. Hvis vi skal ha brukere må vi også ta hensyn til GDPR.

Avtalt rapportering er annenhver uke.

4.3 Gruppenormer – samarbeidsregler – holdninger

Gruppen er enig i at man skal være ærlige og si fra dersom man trenger hjelp. Vi skal ha en pågående diskusjon for å dekke alle uklarheter og spørsmål. Alle skal være åpen for tilbakemeldinger og forslag.
Vi skal være profesjonelle og sette oppdragsgiver i fokus. Dette innebærer å ha en god informasjonsflyt der vi holder de avtaler som blir gjort.

Dersom det oppstår faglige uenigheter i gruppen, så er det flertallet som bestemmer innretning og videre arbeid. Dersom det ikke er noe flertall, bestemmer prosjektleder.

Alle medlemmer skal møtes på avtalte tidspunkt. Dersom enkelte blir forsinket eller ikke har mulighet til å møte, skal det gis beskjed til de andre på gruppen.

En dataingeniør burde sette oppdragsgiver i fokus og kommunisere godt slik at produktet som skal utvikles blir så likt det oppdragsgiveren ser for seg som mulig.

5 PROSJEKTBESKRIVELSE

5.1 Problemstilling - målsetting – hensikt

Multilux har lenge brukt Excel for salg av lys på båt. De vil nå gå over til en interaktiv prisliste. Målet med dette er at kunden selv kan finne produkter og dens tilhørende pris, og dermed avlaste selgerne i Multilux AS. Prislisten tar utgangpunkt i belysning til båter/shipping, og er fordelt på de forskjellige typer fartøy (Cruise, supply, fiskerbåt etc.).

Hovedmålet er å lage en interaktiv prisliste som både kundene og selgerne kan bruke.

Delmålene i prosjektet er å få ferdig alle de forskjellige komponentene vi må ha. Dette innebærer database, server, klient-applikasjon og administrasjons-applikasjon. Ellers er der noen delmål for mer administrativt og ressursbasert arbeid som innsamling av alle nødvendige plantegninger og bilder, felles bruk av versjonskontroll og møter med oppdragsgiver.

5.2 Krav til løsning eller prosjektresultat – spesifikasjon

Kravet til prosjektresultatet er et program der brukere kan lett finne fram til produktet de leter etter, samt legge alle ønskede produkter i en samlet oversikt (handlekurv e.l.) slik at man kan se total pris. Prosjektet skal også være modulært så selgeren kan redigere det man kan finne i programmet. Dette innebærer endring av priser priser, legge til / fjerne båter, produkter, bilder osv.

5.3 Planlagt framgangsmåte(r) for utviklingsarbeidet – metode(r)

Fremgangsmåten for utviklingsarbeidet vil bli basert på hvordan skoleåret er lagt opp. Vi jobber med forprosjekt frem til 31. Januar og vil ha klar en prosjektplan til Februar.

I starten av Februar vil vi begynne med prosjektarbeidet. Det er planlagt å bruke Scrum, som er en smidig arbeidsmetode. Vi vil ha møter med oppdragsgiver annenhver uke.

Styrken med Scrum er at man vil få god og jevn kommunikasjon med oppdragsgiver slik at de får periodiske oppdateringer på framdriften til prosjektet. Dette gjør at oppdragsgiver kan komme med tilbakemeldinger og si fra hvis vi gjør noe som ikke følger deres forventninger.

En av svakhetene til Scrum er at hvis ikke alle er dedikerte og samarbeidsvillige vil ikke resultatet bli like bra.

5.4 Informasjonsinnsamling – utført og planlagt

Den eksisterende løsningen består av ulike Excel dokument med produkter og tilhørende priser som blir manuelt redigert av selgerne.

Ytterligere informasjon om det eksisterende systemet kan vi få fra oppdragsgiver.

5.5 Vurdering – analyse av risiko

I tabellen under vises mulige farer for prosjektet, samt konsekvenser, tiltak og sannsynlighet for at faren skal oppstå.

Fare	Konsekvens	Tiltak	Sannsynlighet
Sykdom blant gruppemedlemmer	Forsinket arbeid, eller mer arbeid på de andre	Andre på gruppen må gjøre mer arbeid eller be om utsettelse	Lav
Feilaktig tolkning av problemstillingen	Arbeid må gjøres på nytt og prosjektet kan ta lenger tid enn antatt	Avklare med oppdragsgiver	Lav
Uenighet blant gruppemedlemmene	Prosjektet stoppes opp	Følge gruppens normer og regler	Lav
Tap av data	Arbeid må gjøres på nytt og prosjektet kan ta lenger tid enn antatt	Lagre i skytjenester / Git, ta regelmessig backup.	Middels
Mangel på kunnskap blant medlemmer	Arbeidet stopper opp og prosjektet kan ta lenger tid enn antatt.	Medlemmer må lese seg opp og få den kunnskapen som er nødvendig for å fullføre prosjektet.	Lav

5.6 Hovedaktiviteter i videre arbeid

Nr	Hovedakt.	Ansvar	Kostnad	Tid/omfang	Beskrivelse
1	Forprosjekt	Prosjektleder	0,-	~5d	Prosjektplan og oversikt
2	Prototype	Prosjektleder	0,-	~15d	Et prosjekt som skal vises til oppdragsgiver slik vi vet hva dem vil ha
3	Server/database	Ikke avklart	0,-	~10d	Sette opp en database struktur som er optimalisert
4	Frontend/ design	Ikke avklart	0,-	~10d	Brukervennelig design
5	Backend/ Logic	Ikke avklart	0,-	~10d	Logikk for behandling av data
6	Alpha	Prosjektleder	0,-	~10d	Testing
7	Beta	Prosjektleder	0,-	~10d	Testing
8	Oppsett hos deres systemer	Prosjektleder + noen derifra	De trenger server	~5d	Produksjon
Tot			0,-	~75d	

5.7 Framdriftsplan – styring av prosjektet

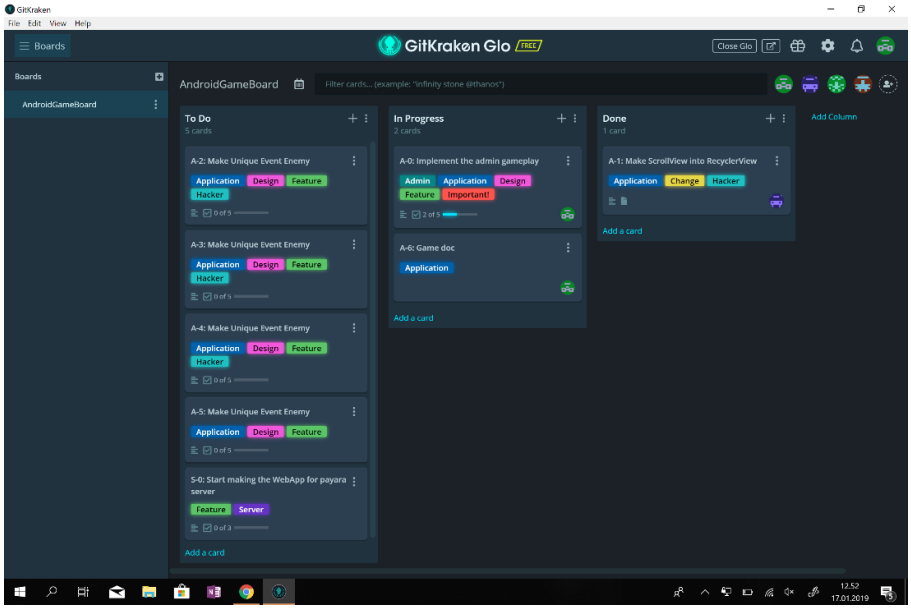
5.7.1 Hovedplan

Vi har en smidig arbeidsmetode så resultater pr. Leveranse vil bli satt ulikt for hver uke. Dette innebærer at man ikke kan fastslå noen spesifikke datoer.

- Etter at prosjektplanen er ferdig og design er godkjent vil vi starte utvikling. Dato for innlevering av prosjektplan/forprosjekt er 31. Januar.
- Vi vil starte med å lage en prototype av programmet for videre godkjenning av oppdragsgiver. Antar dato for ferdigstilling i løpet av Februar.
- Når prototypen er godkjent vil vi starte utviklingsprosjektet for fullt. Dette innebærer å utvikle alle de ulike komponentene. Her vil ansvar bli delt ut basert på hvem som kan mest om de ulike komponentene. Vi forventer at komponentene er ferdige i løpet av April på grunn av at vi har eksamen i Mars.
- Når komponentene er ferdige vil vi ha en godkjenning av oppdragsgiver. Vi endrer så på komponentene basert på tilbakemelding fra oppdragsgiver. Vi forventer at dette blir godkjent ikke lenge etter ferdigstilling av komponentene.
- Vi lager ferdig unit tester og integrasjonstester før vi starter testfasen av systemet. Noe av dette kommer til å bli gjort imens vi lager systemet, så kan hende dette allerede er ferdig når vi har kommet til dette punktet. Forventet ferdig i midten av April.
- Alpha-testing (testrunde 1). Her vil vi invitere venner og familie til å teste programmet og prøve å ødelegge det. Når testing er gjennomført sitter vi mest sannsynligvis igjen med mye informasjon som kan brukes til å forbedre systemet ytterligere. Forventet ferdig før Mai.
- Beta-testing (testrunde 2). Dette skal være en testfase der vi forventer at det er få feil i systemet. Testrunde 2 kan gå løpende ut hele Mai.
- Oppsett av system hos oppdragsgiver og eventuelt oppsett av noen devops metoder osv. Spørs veldig hva oppdragsgiver ønsker. Forventet ferdig i Mai.

5.7.2 Styringshjelpemidler

- Vi kommer til å bruke GitKraken som git-klient. Dette er fordi GitKraken er en god git-klient samtidig som den har innebygd issue tracking, som gjør det lettere og raskere å jobbe med git. Dette vil også gi oss en måte å sette opp oppgaver som må gjøres og få oversikt over hva som er ferdig.
- Bildet under viser hvordan issue tracking i fungerer i GitKraken. Her kan du definere kolonner og lage markeringer selv. Prosjektleder kan definere oppgaver som må gjøres, sette antatt arbeidstid osv. Vi kommer til å ha en kolonne som vi vil kalle arkiv. Her legger man issues som er ferdige og er klare for å kunne sende til oppdragsgiver.



5.7.3 Utviklingshjelpemidler

- Git plattform
- Issue tracking system: GitKraken Glo Boards
- IDE (Integrert Utviklingsmiljø som f.eks NetBeans / IntelliJ)

5.7.4 Intern kontroll – evaluering

- Oppfølging av framdrift vil bli gjort av prosjektleder ved hjelp av issue tracking systemet.
- Merge-request vil bli brukt når en endring er gjort. Alle endringer i prosjektet skal bli godkjent av en annen i gruppen. Dette er slik at ikke noen legger til noe som ødelegger systemet.
- På fullføringer og leveringer skal det bli godkjent av prosjektleder samt oppdragsgiver.

5.8 Beslutninger – beslutningsprosess

Gruppen vil fungere demokratisk og vil diskutere seg fram til hver beslutning. Det er slik vi har jobbet i forprosjektet og planen er å jobbe på samme måte fremover.

6 DOKUMENTASJON

6.1 Rapporter og tekniske dokumenter

- Dokumentasjon i form av en README skal skrives for hver komponent i systemet.
- Hvert medlem skal dokumentere hvilke endringer de har gjort i prosjektet før disse blir pushet inn i master branchen i git.
- For godkjenning vil vi bruke "merge request". Dette gjør at noen andre må godkjenne arbeid før endringer går ut i produksjon. Eventuelle unntak kan gjøres.
- Kopiering vil gjøres automatisk når vi bruker versjonkontroll. Ellers vil vi skrive dokumentene våre i skybaserte tjenester som gjør at de vil være lagret der.
- Programvaren vil bli oppbevart hos gitlab/bitbucket.
- For å vedlikeholde programvaren vil vi legge inn unit testing og integrasjonstester.

7 PLANLAGTE MØTER OG RAPPORTER

7.1 Møter

7.1.1 Møter med styringsgruppen

- Møter blir gjort etter diskusjon i gruppen.

7.1.2 Prosjektmøter

- Vi har et planlagt møte med oppdragsgiver i uke 8. Hensikten er å snakke om prosjektet på en veldig detaljert måte slik at alle har en felles forståelse på hva oppgaven virkelig går ut på. Til denne datoen vil vi også ha ferdig en prototype slik vi kan få tilbakemelding på hvordan produktet skal se ut og fungere.
- Scrum møter blir annenhver uke, men vil også ta noen løpende møter hvis der er behov for å snakke sammen og diskutere.

7.2 Periodiske rapporter

7.2.1 Framdriftsrapporter (inkl. milepæl)

- Planlagte rapportformer blir basert på de ulike komponentene i systemet. Det vil bli en framdriftsrapport for hver komponent. Formatet på disse rapportene blir først dokumentasjon på selve komponenten, deretter blir det en tidslinje som viser fremgang og veien videre.
- Datoer for disse rapportene er ikke fastsatt og vil bli bestemt etter hvert som prosjektet pågår. Planen er at rapportene skal være på en skybasert dokumentplattform slik oppdragsgiver hele tiden kan gå inn og sjekke nyeste oppdatering.

8 PLANLAGT AVVIKSBEHANDLING

- Hvis framdrift/innhold ikke går som planlagt omdiskuterer vi prosjektet med oppdragsgiver og oppdaterer backlog. Eventuelt justerer vi arbeidsmengde.
- Prosedyre for endringer er først diskusjon i gruppen, diskusjon opp mot oppdragsgiver og tilslutt oppdatering av backlog. Endringer er lett å foreta når man bruker en smidig arbeidsmetode som Scrum.
- Prosjektleder tar ansvar for fremdrift, så hvis der er avvik som hindrer framdrift finner prosjektleder en løsning.

9 UTSTYRSBEHOV/FORUTSETNINGER FOR GJENNOMFØRING

Hvis oppdragsgiver skal ha programmet oppe å kjørende på nettet trenger de å hoste en server/database som vi kan bruke.

For å gjennomføre oppgaven trenger vi også plantegninger og bilder av rom og utstyr for alle fartøy.

10 REFERANSER

VEDLEGG

Vedlegg 1: Powerpoint presentasjon fra Multilux: Prosjekt-Interaktiv-prisliste.pptx

B Oppgavens idé, powerpoint fra Multilux

INTERAKTIV PRISLISTE

Multilux

Målet med en interaktiv prisliste er at kunden selv kan finne produkter og dens tilhørende pris, og dermed avlaste selgerne i Multilux AS.
Prislisten tar utgangspunkt i belysning til båter/shipping, og er fordelt på de forskjellige typer fartøy (Cruice, supply, fiskebåt etc.).

Multilux

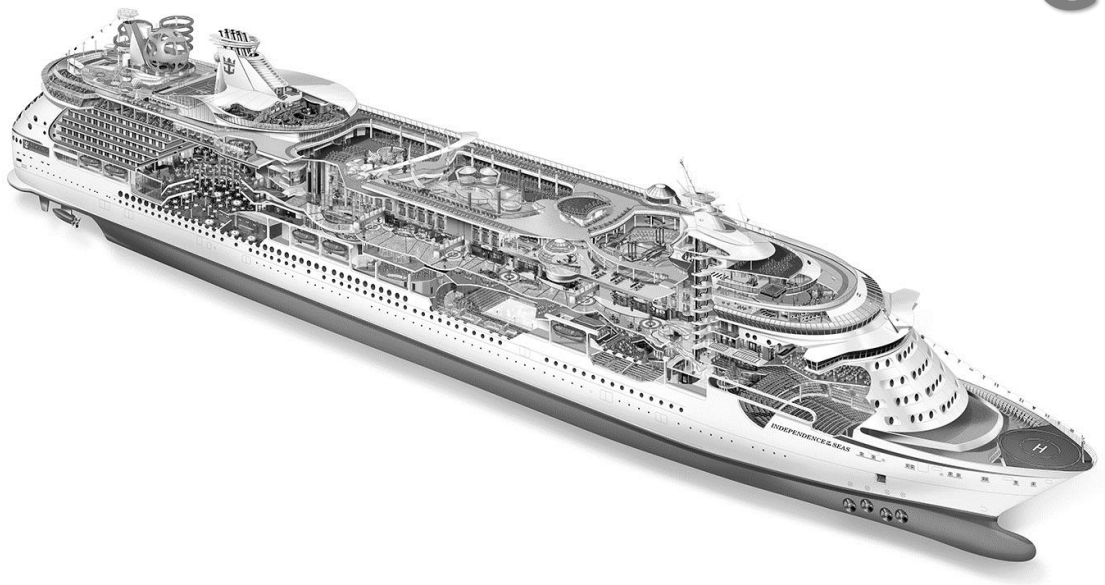
Forsiden begynner med en innholdsliste over de forskjellige fartøyene, og er linket videre til plantegning over det aktuelle farøyet.

- Cruise vessel
- Supply ship
- Car/passanger vessel
- RO/RO
- Oil/gas/chemicals
- Fishing vessel

Multilux

Tilbake

CRUISE

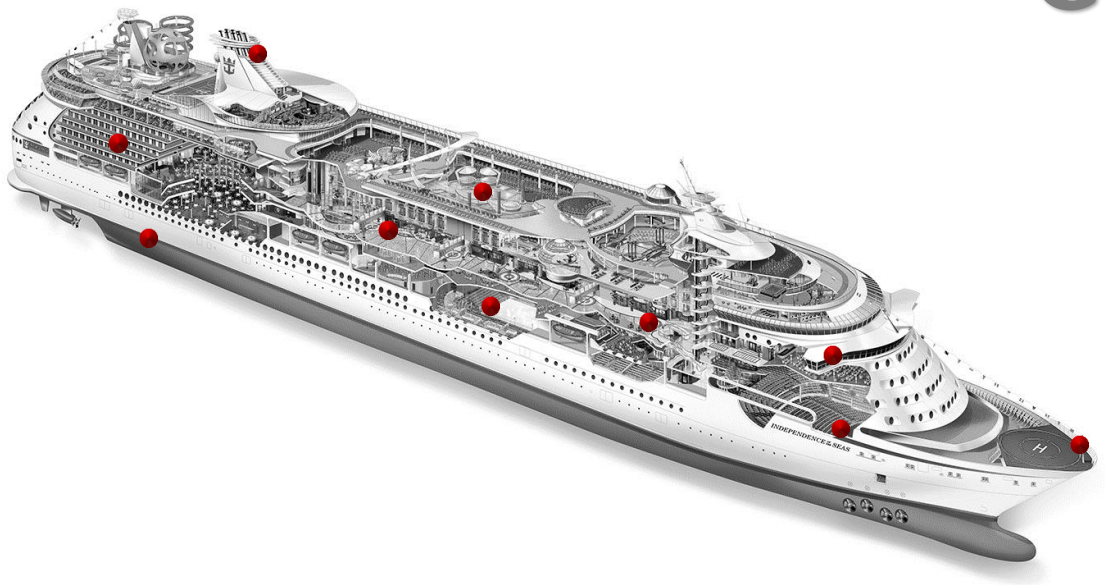


Inne på en fartøyskategori vil man få
opp et tverrsnitt av det valgte fartøyet.

Multilux

Tilbake

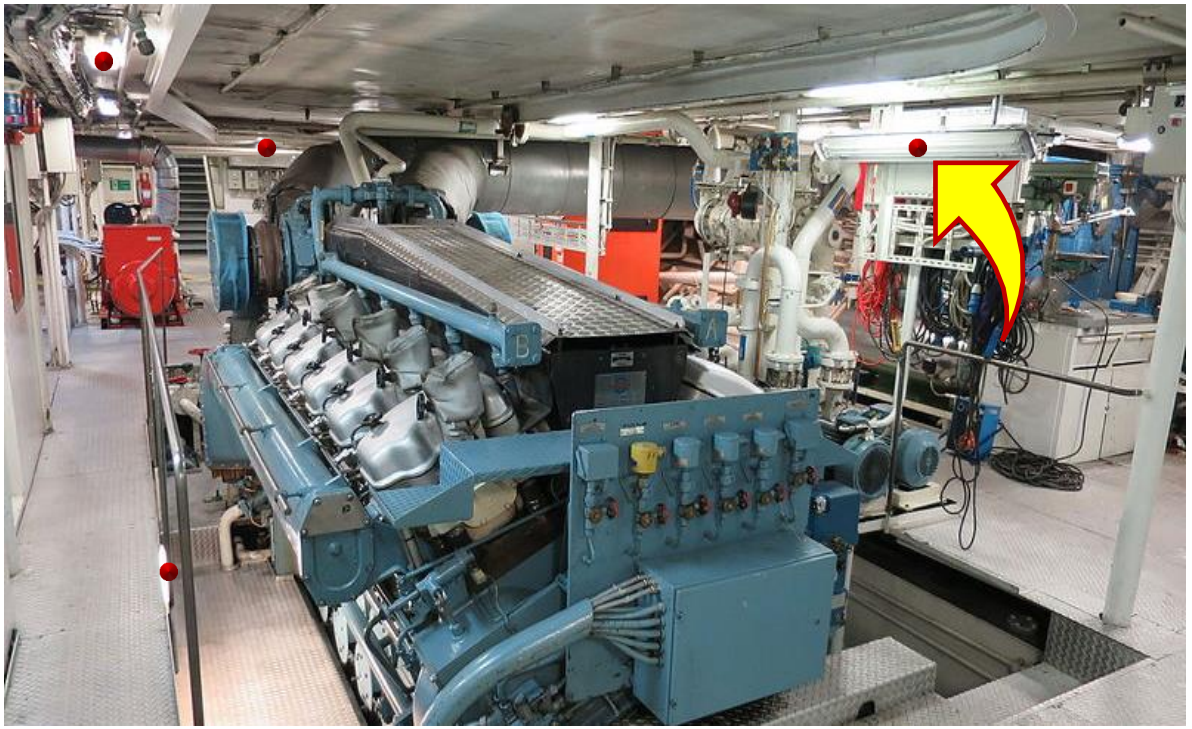
CRUISE



På plantegningen vil det være hyperlinker til de forskjellige rommene på fartøyet. Eks: Maskinrom, lugar, baug, bro osv..
Man kan hele tiden gå tilbake til forrige side ved å trykke på «Tilbake».

Multilux

Tilbake



Klikker man f.eks på «Maskinrom», vil man bli tatt videre til et maskinrom hvor det er nye hyperkoblinger mot produktene.

Multilux

Tilbake

Product name	Watt tot.	°K	Lm (output)	Replace	Price before discount		Discout	Amount	Total Price
Minerva	10	4000	1104	1x18W fluo					
	21	4000	2614	2x18W fluo					
	23	4000	2765	1x36W fluo					
	37	4000	5009	2x36W fluo					
	50	4000	6730	2x54W fluo					
	28	4000	3639	1x58W fluo					
	51	4000	6591	2x49W fluo					
	60	4000	7584	2x58W fluo					
	68	4000	9144	2x80W fluo					

The luminaire is also available with 3000K, 5000K and 6500K light source and shatterproof glass. Possibility of DALI or 1-10V.

FDV

Etter at man har klikket på et produkt i maskinrommet, kommer anbefalt produkt fra Multilux opp. Her vil det bli oppført listepriser, og kunden fyller selv inn rabatt og antall. Totalpris blir deretter regnet ut automatisk. Denne «handlelisten» vil bli lagret på en egen side på tvers av alle fartøy og produkter.
På denne siden vil det linkes til produktdatablad (FDV), noe som også er mulig fra «handlelisten». Handlelisten må kunne skrives ut/lagres.
Listeprisene bør kunne endres med en valutafaktor. Det brukes DKK og EUR i innkjøp av disse produktene, men de selges i NOK.

Multilux

C Kildekode i zip fil

Referanser

[1] Misc - multilux interactive shopping cart, 2019.
https://www.telseth.no/.

[2] advancedrestclient.com. Advanced rest client, sett 15. mai 2019.
https://install.advancedrestclient.com/install.

[3] Dan Arias. Adding salt to hashing: A better way to store passwords, sett mai 2019.
https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/.

[4] Jeff Atwood. Dictionary attacks 101, sett mai 2019.
https://blog.codinghorror.com/dictionary-attacks-101/.

[5] Linux Audit. Linux hardening steps, sett 8. mai 2019.
https://linux-audit.com/linux-server-hardening-most-important-steps-to-secure-systems/.

[6] Scottish Qualifications Authority. Fundamentals of database design, sett 12. mai 2019.
https://www.sqa.org.uk/e-learning/MDBS01CD/page_06.htm.

[7] Axosoft. git-client, sett 15. mai 2019.
https://www.gitkraken.com/git-client.

[8] Ahmed Bouchefra. A complete guide to routing in angular, sett 19. mai 2019.
https://www.smashingmagazine.com/2018/11/a-complete-guide-to-routing-in-angular/.

[9] Tony Branson. A glimpse into database scalability, sett 9. mai 2019.
https://www.infosecurity-magazine.com/opinions/a-glimpse-into-database-scalability/.

[10] Mike Chapple. Introduction to database relationships, sett 12. mai 2019.
https://www.lifewire.com/database-relationships-1019729/.

[11] Symantec Corporation. What is a man-in-the-middle attack?, mai 2019.
https://us.norton.com/internetsecurity-wifi-what-is-a-man-in-the-middle-attack.html.

[12] DigitalOcean. How to set up a node.js application for production on debian 9, sett mai 2019.
https://www.digitalocean.com/community/tutorials/how-to-set-up-a-node-js-application-for-production-on-debian-9.

[13] Electron. Electron, sett mai 2019.
https://electronjs.org/docs/tutorial/about.

[14] Expressjs. Express application generator, sett 15. mai 2019.
https://expressjs.com/en/starter/generator.html.

[15] Internet Engineering Task Force. Json web token (jwt), mai 2015.
https://tools.ietf.org/html/rfc7519.

[16] Internet Engineering Task Force. Pkcs #1: Rsa cryptography specifications version 2.2, november 2016.
https://tools.ietf.org/html/rfc8017https://tools.ietf.org/html/rfc8017https://tools.ietf.org/html/rfc8017.

[17] Internet Engineering Task Force. Elliptic curve cryptography (ecc) cipher suites for transport layer security (tls) versions 1.2 and earlier, august 2018.
https://tools.ietf.org/html/rfc8422https://tools.ietf.org/html/rfc8422https://tools.ietf.org/html/rfc8422.

[18] Jędrek Fulara. Component-based architecture in angular 2, sett september 2016.
https://www.sparkbit.pl/component-based-architecture-angular-2/.

[19] GitHub. About, sett 19. mai 2019.
https://github.com/about.

[20] Mehmet Goktürk. 37. fitts's law, sett 16. mai 2019.
https://www.interaction-design.org/literature/book/the-glossary-of-human-computer-interaction/fitts-s-law.

[21] Google. Architecture overview, sett mai 2019.
https://angular.io/guide/architecture.

[22] Google. Httpclient, sett mai 2019.
https://angular.io/guide/http.

[23] Google. Angular material, sett mai 2019.
https://material.angular.io/.

[24] Network Working Group. Hmac: Keyed-hashing for message authentication, februar 1997.
https://tools.ietf.org/html/rfc2104https://tools.ietf.org/html/rfc2104https://tools.ietf.org/html/rfc2104.

[25] PostgreSQL Global Development Group. postgresql, sett 15. mai 2019.
https://www.postgresql.org/.

[26] James Hall. jspdf, sett 15. mai 2019.
https://www.npmjs.com/package/jspdf.

[27] Ionos. Linux vs. windows: a comparison of the best web server solutions, sett 11. mai 2019.
https://www.ionos.com/digitalguide/server/know-how/linux-vs-windows-the-big-server-check/.

[28] Darril Gibson James Michael Stewart, Mike Chapple. *CISSP: Certified Information Systems Security Professional Study Guide*. John Wiley & Sons Inc, Indianapolis, Indiana, 2008. ISBN 9781118314173.

[29] Jeff Johnson. *Designing with the Mind in Mind*. Morgan Kaufmann Publishers, Burlington, Massachusetts, 2014. ISBN 9780124079144.

[30] JWT. Introduction to json web tokens, sett april 2019.
https://jwt.io/introduction/.

[31] Kaspersky. What's a brute force attack?, sett mai 2019.
https://www.kaspersky.com/resource-center/definitions/brute-force-attack.

[32] Nate Lapinski. The three pillars of angular routing. angular router series introduction., sett 19. mai 2019.
https://blog.angularindepth.com/the-three-pillars-of-angular-routing-angular-router-series-introduction-fb34e4e8758e.

[33] Kiran Malvi. The positive and negative aspects of node.js web app development, sett 15. mai 2019.
https://www.mindinventory.com/blog/pros-and-cons-of-node-js-web-app-development/.

[34] Microsoft. Typescript - javascript that scales, sett 18. mai 2019.
https://www.typescriptlang.org/.

[35] Microsoft. Visual studio code, sett 15. mai 2019.
https://code.visualstudio.com/.

[36] Multilux. Om oss, sett 15. mai 2019.
https://multilux.no/om-oss/.

[37] R. A. Karthika P. Sriramya. *PROVIDING PASSWORD SECURITY BY SALTED PASSWORD HASHING USING BCRYPT ALGORITHM*. Asian Research Publishing Network, Islamabad, Pakistan, 2015.

[38] pgAdmin. pgadmin, sett 15. mai 2019.
https://www.pgadmin.org/.

[39] Vasyl Redka. Understanding restful api, sett 9. mai 2019.
https://mlsdev.com/blog/81-a-beginner-s-tutorial-for-understanding-restful-api.

[40] Margaret Rouse. security token (authentication token), sett 18. mai 2019.
https://searchsecurity.techtarget.com/definition/security-token.

[41] Margaret Rouse. two-factor authentication (2fa), sett 18. mai 2019.
https://searchsecurity.techtarget.com/definition/two-factor-authentication.

[42] Mohammed Sanaulla. Cohesion and coupling: Two oo design principles, sett 18. mai 2019.
https://sanaulla.info/2008/06/26/cohesion-and-coupling-two-oo-design-principles/.

[43] Bruce Schneier. Real-world passwords, 14. desember 2006.
https://www.schneier.com/blog/archives/2006/12/realworld_passw.html.

[44] Remy Sharp. nodemon, sett 12. mai 2019.
https://github.com/remy/nodemon#nodemon.

[45] Nasjonal sikkerhetsmyndighet. Hypertext transport protocol secure (https), sett 5. mai 2019.
https://www.nsm.stat.no/publikasjoner/regelverk/veiledninger/veiledning-for-systemteknisk-sikkerhet/hypertext-transport-protocol-secure/.

[46] Næringslivets sikkerhetsråd. Mørketallsundersøkelsen - informasjonssikkerhet og datakriminalitet, sett 18. mai 2019.
https://www.nsr-org.no/moerketall/.

[47] Bunmi Sowande. The essentials of database scalability: Vertical & horizontal, sett 12. mai 2019.
https://blog.turbonomic.com/blog/on-technology/the-essentials-of-database-scalability-vertical-horizontal/.

[48] Bunmi Sowande. Overview, sett 12. mai 2019.
http://www.passportjs.org/docs/.

[49] Girts Strazdins. Lecture 4: The relational model, 2018. ID202912 Systemutvikling og modellering.

[50] TargetInternet. An introduction to agile working, sett 8. mai 2019.
https://www.targetinternet.com/an-introduction-to-agile-working/.

[51] Tutorialspoint. Dbms - data models, sett 12. mai 2019.
https://www.tutorialspoint.com/dbms/dbms_data_models.htm.

[52] tutorialspoint. Typescript - overview, sett 19. mai 2019.
https://https://www.tutorialspoint.com/typescript/typescript_overview.htm.

[53] w3schools. Node.js npm, sett mai 2019.
https://www.w3schools.com/nodejs/nodejs_npm.asp.

[54] Hao Wang. Lecture 4: The relational model, 2017. ID202912 Datamodellering og databaseapplikasjoner.

[55] Wikipedia. Angular (web framework), sett mai 2019.
https://en.wikipedia.org/wiki/Angular_(web_framework).

[56] Wikipedia. Application programming interface, sett 18. mai 2019.
https://en.wikipedia.org/wiki/Application_programming_interface.

[57] Wikipedia. Cohesion (computer science), sett 18. mai 2019.
https://en.wikipedia.org/wiki/Cohesion_(computer_science).

[58] Wikipedia. Coupling (computer programming), sett 18. mai 2019.
https://en.wikipedia.org/wiki/Coupling_(computer_programming).

[59] Wikipedia. Cascading style sheets, sett 18. mai 2019.
https://en.wikipedia.org/wiki/Cascading_Style_Sheets.

[60] Wikipedia. Html, sett 18. mai 2019.
https://en.wikipedia.org/wiki/HTML.

[61] Wikipedia. Wizard (software), sett 15. mai 2019.
https://en.wikipedia.org/wiki/Wizard_(software).

[62] WinSCP. Winscp, 9. mai 2019.
https://winscp.net/eng/index.php.

[63] Mirco Zeiss. json2csv, sett mai 2019.
https://www.npmjs.com/package/json2csv.