

Eksamen rapport OBJ2100

Prosjektplanlegging

For å sikre en effektiv utviklingsprosess, har vi delt prosjektet inn i flere faser: kravspesifikasjon, design, implementering, testing og dokumentasjon.

Kravspesifikasjon

Før vi begynte med design og implementering, har vi grundig analysert oppgaveteksten for å forstå alle krav og funksjoner som systemet skal oppfylle. Viktige funksjoner inkluderer:

- Opprette, søke etter, redigere og slette inventarelementer.
- Bruk av et hierarkisk system for plassering av inventar.
- Håndtering av forskjellige typer inventar: møbler, utsmykning og teknisk utstyr.

Design

Designfasen skal innebære å lage en overordnet arkitektur for systemet. Vi har identifisert hovedkomponentene og deres ansvar:

- **Klientprogrammet:** GUI for interaksjon med brukeren og sender forespørsler til tjenerprogrammet.
- **Tjenerprogrammet:** Behandler forespørsler fra klienten og kommuniserer med databasen.
- **Database:** Lagre inventardata, implementert med SQLite.

Mockups av startsiden

Vi lagde 3 forskjellige mockups av hva startsiden burde se ut som. I alle disse var hovedfokuset vårt å ha noe som enkelt fremstilte informasjonen vi ønsket burde komme frem og formenten ville være viktig. Under denne seksjonen vises de forskjellige mockupene.

Første mockup

Børres inventar

Legg til nytt inventar

Søk i inventar...

Inventartype

Kategori

Innkjøpspris

Plassering

Inventar Type - Kategori

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo

Dato

Pris



Inventar Type - Kategori

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo

Dato

Pris

Andre mockup

Søk i inventar

☐

Innkjøpspris

200 - 5000 kr

Forventet levetid

2 - 5 år

Forventet kassering

2 - 5 år

Antall

50 - 70

Velg: i bruk / ikke i bruk

Tatt ut av bruk ☐ I bruk ☒

Inventar

Legg til nytt inventar

Sorter etter ...

<div><div>Inventar Type - Kategori</div><div> Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo</div></div>	<div>Dato</div> <div>Pris</div>	<div></div> <div></div> <div></div>
<div><div>Inventar Type - Kategori</div><div> Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo</div></div>	<div>Dato</div> <div>Pris</div>	<div></div> <div></div> <div></div>
<div><div>Inventar Type - Kategori</div><div> Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo</div></div>	<div>Dato</div> <div>Pris</div>	<div></div> <div></div> <div></div>

1 2 3 4 5 6 ...

Valg inventar:

Plassering:

Honefoss, tullevøgen/5

Kategori

Tullekategori

Inventartype:

Tulletype

Forkortet beskrivelse:

Tullebeskrivelse

I bruk:

Tatt ut av bruk

Tredje mockup

<

Søk i inventar...

Søkevalg

Innkjøp

23/04/1995

Levetid

20 år

Kassering

5 år

Tullekategori

Tulletype

Antall: 5000

KR

500.00 kr

ANTALL

350

Beskrivelse:

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat.

Plass:

Honefoss, tullevøgen 3/5

Er inventaret i bruk?

Ikke i bruk

Tatt ut av bruk: 2023

Årsak: Lagt på lager

Det aktuelle designet til løsningen ble noe annerledes, ettersom noen utfordringer som forsinket utviklingsprosessen en del. Se til “Utfordringer” for beskrivelse av dette.

Kommunikasjon mellom Klient og Tjener

Kommunikasjonsmønster foregår over sockets, og aksjoner skal være bestemt med kommandoer og typer. Enums for dette.

- **Oppretting, lesing, oppdatering og sletting:** Klienten skal kunne be serveren behandle aksjoner i relasjon til klientens ønsker. Alle objekttyper skal kunne være opprettbare, lesbare, mulige å oppdatere og slettbar.
- **Hent alle:** Klienten skal kunne hente alle av en spesifisert datatype samlet.
- **Søk:** Klienten skal kunne be serveren behandle en søkeaksjon, for å behandle søk i varierende grad av kompleksitet. Altså med og uten filtre.

Oppgavefordeling

Vi benyttet oss av Trello for rollefordeling, og sporing av hvem og hva som ble gjort. I tillegg til dette så arbeidet vi i trunks, altså “git branches” dedikert til utvikling av spesifikk funksjonalitet, og slår sammen kode med “master” branchen til git gjennom GitHub’s Pull Requests, slik at vi kunne ta raske code reviews og se hva som hadde endret og blitt lagt til.

I tillegg til dette så har vi vært i nett stemmechat gjennom hele eksamensperioden, slik at om det var noe, så hevet man bare stemmet, og så hørte resten det og kunne svare umiddelbart.

Funksjonelle og ikke-funksjonelle krav

Funksjonelle

- Opprette inventar-element
 - Løsningen må tillate sluttbrukere å opprette nye inventar-elementer med all nødvendig informasjon.
- Søke etter inventar element

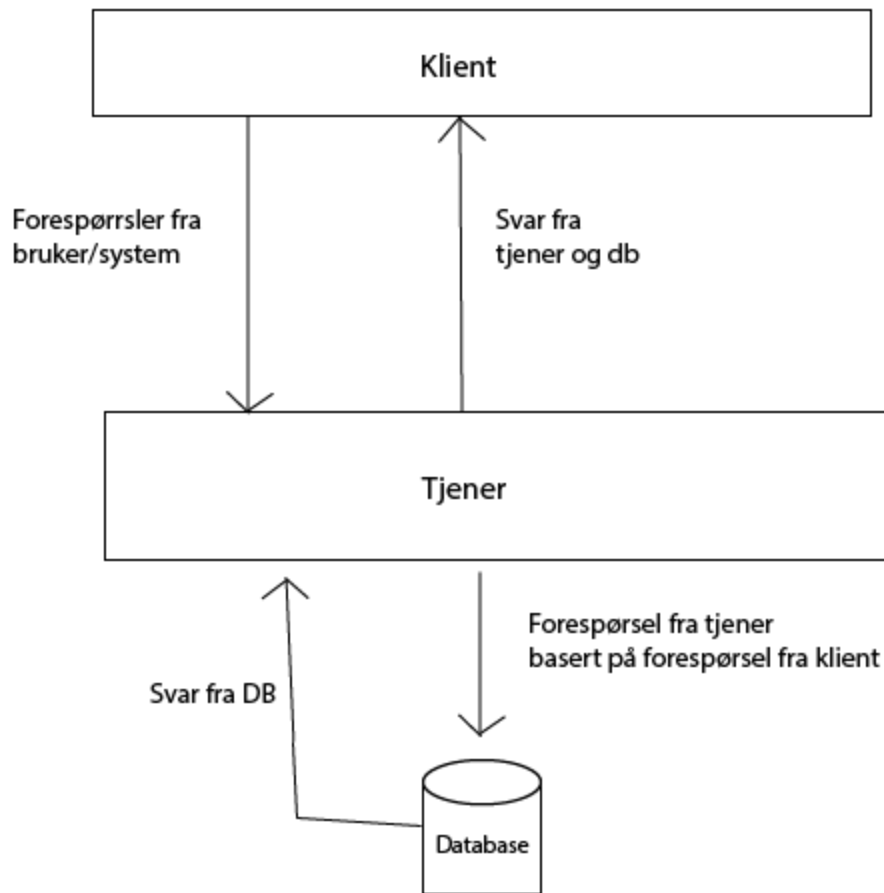
- Løsningen må tilby sluttbrukere kriteriebasert søkefunksjonalitet med relativt høyt detaljnivå.
- Redigere og slette inventar-element
 - Løsningen må tilby sluttbrukere mulighet for å redigere informasjon, eller slette valgte inventar-elementer etter søk.

Ikke-funksjonelle

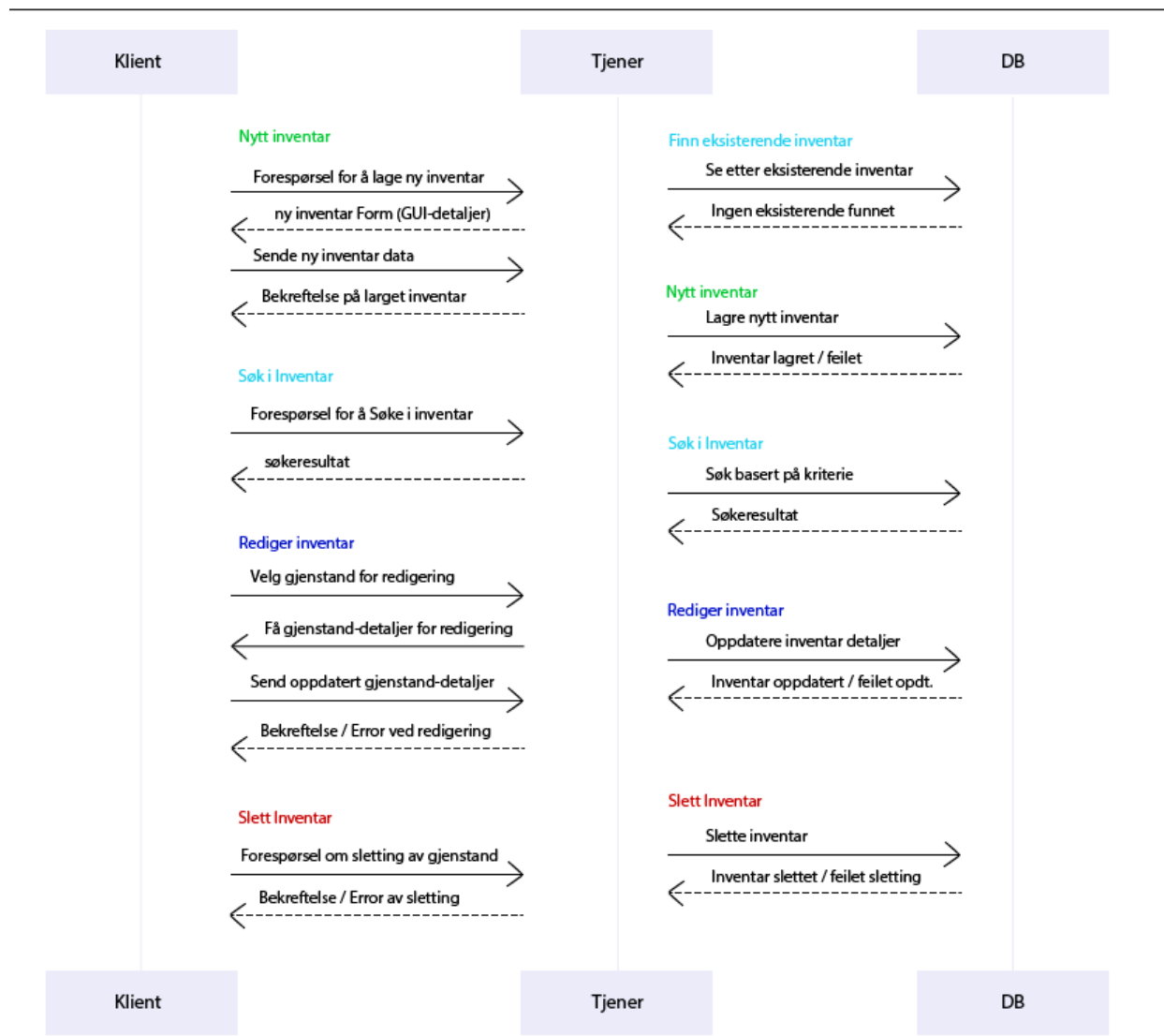
- Ytelse
 - Løsningen bør være raskt og responsivt, og søke- og oppdateringsresultater bør være raske, selv ved store mengder data.
- Brukervennlighet
 - Brukergrensesnittet bør være intuitivt og enkelt å bruke, og feilmeldinger må være klar og forståelige for sluttbrukere.
- Pålitelighet
 - Løsningen bør være robust, og pålitelig, for å minimere risikoen for nedetid og dataredundans.
- Skalerbarhet
 - Løsningen bør kunne skalere systemet horisontalt/vertikalt for å imøtekomme økt datamengde og bruk over tid.

Generell arkitektur

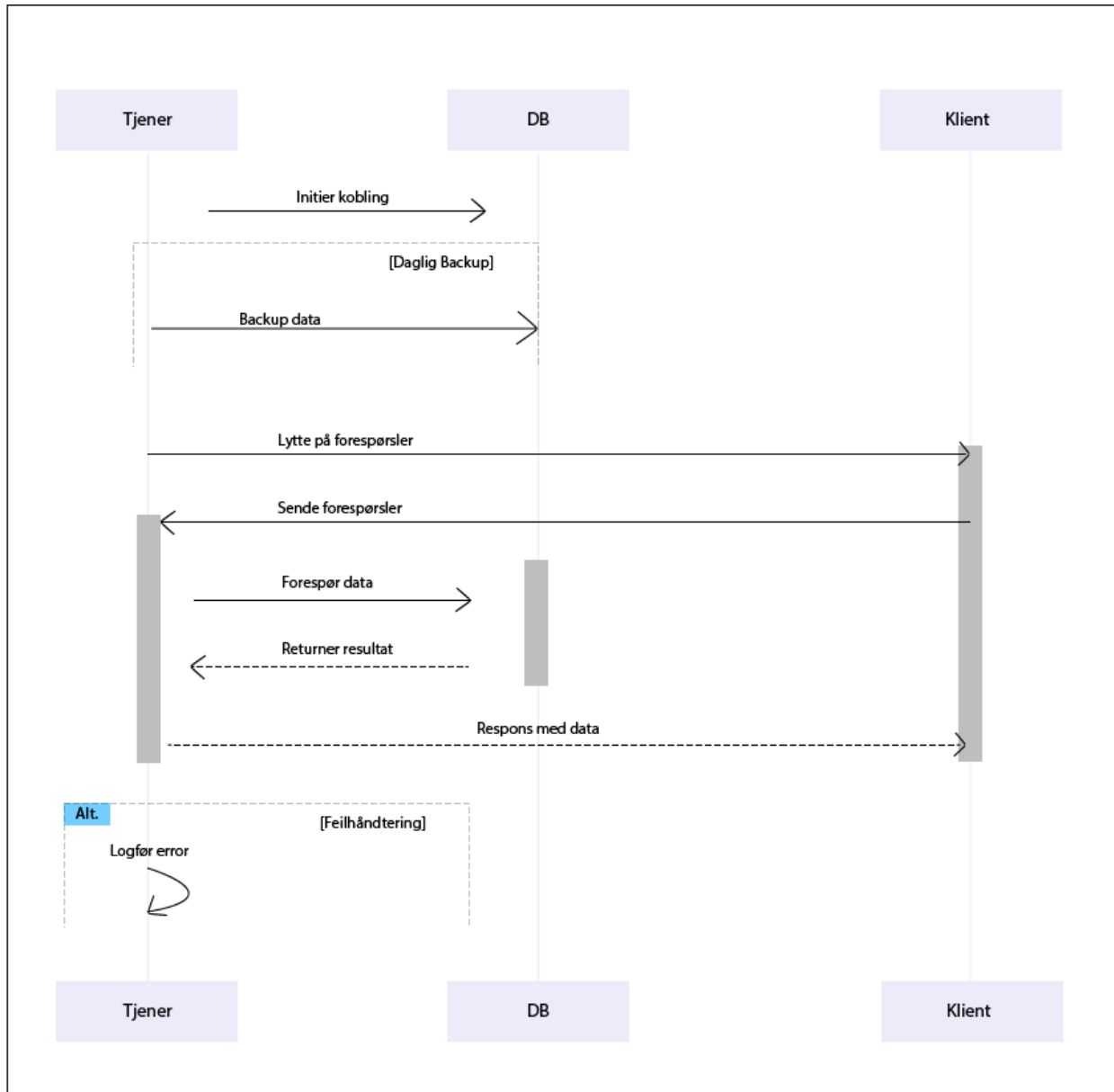
Dette er arkitekturen vi planlegger til hele programmet, det viser forholdet mellom klient /server/database.



Dette er forholdet mellom tjener-klient-db, med user stories:



Oversikt over tjener:



Modularisering

Eksamensoppgavesettet beskriver at det skal være separate programmer, og vi var litt usikre på hvordan dette burde tolkes. Vi så for oss hovedsakelig 2 forskjellige tolkninger av dette:

1. All kode for klient og server/database er i samme pakke, altså samme kildekodemappe i prosjektet, men har ikke lov til å snakke sammen direkte.

2. Kode for klient og server/database logisk separeres på større grad enn bare at de ikke kan direkte kalle på samme kode.

Vi gikk for tolkning 2, og satte opp et multi-module Maven prosjekt (JetBrains, 2023) slik at vi har tre logiske moduler som er separate fra hverandre, så langt så vidt man ikke velger å sette dem som en dependency til hverandre.

Slik prosjektet er satt opp, så har vi disse modulene:

- **client:** Denne modulen inneholder all kode som klienten krever for å kjøre; altså UI og back-end for UIet.
- **server:** Denne modulen inneholder all kode for behandling av klient socket tilkoblinger, behandling av forespørsler, databasekommunikasjon, og diverse annet slikt som JUnit5 tester som simulerer reelle klient tilkoblinger (Bechtold et al., n.d.).
- **shared:** Denne modulen inneholder kode som både **client** og **server** modulene krever for å kompileres. Dette er spesifikt model deklarasjonene, enums for kommandoer og typer, samt som konstante variabler som ikke endres (utenom at man manuelt gjøre det), slikt som serveren's basis portnummer og en switch for å generere dummy data for første oppsett.

Det er viktig å bemerke her at **client** og **server** modulene kan ikke se hverandre på noen vis, da de ikke er eksportert til Maven's miljø, og har ikke hverandre som krav for fullført kompilering. De kan heller ikke se hverandre via **shared** modulen, ettersom at **shared** modulen ikke eksporterer noe funksjonalitet direkte fra **client** eller **server** modulene, siden forholdet er motsatt vei.

Slik har vi da en logisk barriere mellom **client** og **server** slik at de er fullverdig separate applikasjoner, og som vi føler da møter det oppgaveteksten satte krav på.

Vi har hatt noen vansker med dette også, se til "Utfordringer" for utdypet forklaring.

Tjenerapplikasjonen

Serveren er en socket applikasjon designet med et Model/View/Controller (MVC) mønster. Den lytter etter socket tilkoblinger, og delegerer en ny “ClientHandler” tråd for hver klient. Serveren har også dynamisk portnummer, i den forstand at den har et definert port nummer, men om den er opptatt, så prøver den å inkrementere portnummeret og prøve igjen.

Komponenter

- **Server:**
 - **ServerSocket:** Oppretter og lytter etter innkommende forbindelser. Denne komponenten er ansvarlig for å akseptere nye klienttilkoblinger.
 - **ServerThread:** En egen tråd som kjører serveren og håndterer tilkoblinger. Denne tråden sørger for at serveren kan operere kontinuerlig og håndtere flere tilkoblinger.
 - **Dynamic Port Handling:** Serveren starter på en angitt port, og hvis denne porten allerede er i bruk, forsøker den neste tilgjengelige porten. Dette sikrer at serveren alltid kan starte selv om den foretrukne porten er opptatt.
 - **Start Metode:** Starter serveren og oppretter en ny **ClientHandler** for hver tilkobling. Denne metoden sørger for at serveren begynner å lytte etter innkommende forbindelser.
- **ClientHandler:**
 - **Socket:** Representerer klientforbindelsen. Hver tilkoblet klient får sin egen socket som brukes til kommunikasjon.
 - **Kontrollere:** Instanser av forskjellige kontrollere som håndterer spesifikke funksjoner som inventar, plassering, kategorier, kassering, etc. Disse kontrollerne implementerer forretningslogikken og sikrer at forespørsler behandles korrekt.
 - **Input/Output Streams:** Brukes til å lese fra og skrive til klienten. **ObjectInputStream** og **ObjectOutputStream** brukes for å overføre objekter mellom klienten og serveren.

- **Run Metode:** Overstyrer `run`-metoden for å håndtere innkommende forespørsler og sende tilbake svar. Denne metoden inneholder hovedlogikken for hvordan forespørsler fra klienten behandles.

Behandling av Klientforespørsler

- **Forbindelse Opprettet:** Når en klient kobler til serveren, opprettes en ny `ClientHandler`-tråd for å håndtere denne tilkoblingen. Dette sikrer at flere klienter kan koble til samtidig uten at serverens ytelse påvirkes negativt.
- **Forespørsel Mottatt:** `ClientHandler` leser kommandoer, typer og objekter fra klienten ved hjelp av `ObjectInputStream`. Kommandoene, typene og hvilke type objekter de er bestemmer hvilken operasjon som skal utføres, og objektene inneholder nødvendige data for operasjonen.
- **Behandling:** Basert på kommandoen som mottas, videresendes forespørselen til riktig kontroller (f.eks. `InventarController`) for behandling. Kontrolleren utfører den nødvendige logikken ved hjelp av tjenester og returnerer resultatet. Dette designet følger prinsippet om separasjon av ansvar, der hver kontroller er ansvarlig for en spesifikk del av applikasjonen.
- **Databaseoperasjoner:** Hvis forespørselen krever databaseinteraksjon, utfører kontrolleren operasjoner via `DatabaseConnectionManager`. Dette sikrer at all data som behandles av serveren er synkronisert med den underliggende databasen.
- **Svar Til Klient:** Resultatet av behandlingen sendes tilbake til klienten via `ObjectOutputStream`. Dette kan være en bekreftelse på at operasjonen var vellykket, eller data som klienten har bedt om.

Testing

En av hovedfokusene underveis av server utviklingen var at den skulle ha kodetester for å forsørge at funksjonalitet fungerer som ønsket og forventet. Testene innad i server modulen er splittet inn i 2 kategorier; de som tester Server funksjonalitet i et server scope, altså at testene oppfører seg som at de er Serveren, og tester som tester serveren i et klient scope, altså at de simulerer en klient som kobler til.

Det ble dessverre ikke så mye tid til å opprette mange server-scope tester, men de viktigste i form av databasen ble lagd. Og en tidligere test ble slått sammen med “DatabaseConnectionManager” sin database initialisering metode, slik at databasen kan få test data satt inn.

Det ble derimot opprettet mange forskjellige klient-scope tester. Det er tester for hvert objekt, for søk og for å sjekke at man kan koble til serveren. Objekt testene sjekker at man kan opprette, lese, oppdatere og slette objektet av typen testen er for. Samt blir de brukt til å sjekke at man kan få tak i alle elementer fra den tilsvarende database tabellen i relasjon til objekttypen. De ble også brukt for å sjekke at objekt Models var konstruert på et vis som gir mening, og som fungerer for dems bruksområder. Disse testene har blitt ganske mye brukt for å stramme inn oppførsel av koden.

Vi skal ikke påstå at vi har gjort alt perfekt, eller ikke tatt designvalg som muligens kunne lettet på problemer om andre handlinger ble tatt, men tester hjalp oss i det minste forsørge for å en struktur klar for ekte klienter å bare plugge rett inn i.

Til slutt er det også en test som tester variasjoner av søk klienter kan spørre etter. Dette involverer da å opprette basiske søk som bare er en søkestreng som slår opp i beskrivelser og kategori beskrivelser. Og avanserte søk som kan bestå av ett eller flere filtre. Serveren konstruerer en søke forespørsel basert på hvilke filtre som inkluderes når man bygger søkestreng objektet, og returnerer som et utvidet Inventar objekt, som inneholder informasjon fra andre tabeller og ikke bare fremmednøklerne som det ordinære Inventar objektet inneholder. Dette gjør søking litt enklere, sett fra perspektivet til klienten, og senker unødvendig ressursbruk ved å ikke sende ørten forespørsler om forskjellige objekter, og dems underliggende objekter, og heller bare få alt i ett.

Pga. problemstillinger som utvides på under “Utfordringer,” så har vi endt opp under større tidspress enn ellers ønsket, selv om vi lå ellers godt ann. Dermed kan det være at klienten, altså UIet, ikke har full støtte for søkefunksjoner. Dersom det er uklart hvilke søkefunksjoner som er

implementert til nå, så kan dere se til “SearchController” i client modulen, og “SearchTest” i testene til server modulen. Men for enkelhetsskyld listes de som garantert funker også herunder:

- Enkelt søk uten filtrering; slår opp i inventar.beskrivelse og kategori.kategori.
- Filtrering etter beskrivelse.
- Filtrering etter type.
- Filtrering etter forventet kassering.
- Filtrering etter kategori.
- Filtrering etter levetid.
- Filtrering etter plassering.
- Filtrering etter pris.

Slik serveren konstruerer søk burde la den søke etter disse i hvilken som helst konfigurasjon/rekkefølge. Det vil også bemerkes at selv om ikke alt funker som det burde, mest fordi SQL søkebyggeren er relativt kompleks, så skal alle søk være implementert. Det er bare at noe debugging som skal til for at de gir forventede responser. Det er viktig å bemerke at en god del av disse er implementert på serveren og klienten, men ikke i GUIet.

Dersom det er av interesse, så inkluderes et bilde av testresultatene for søking. De som er grået ut er manuelle tester, slik som nevnt ovenfor, er disse søk som funker; men ikke gir forventet resultat. Dette er mest sannsynlig som følge av SQL søkebyggeren, så hadde vi hatt mer tid, ville vi kunne finpusset dem nok til at de faktisk ga riktig svar. Men fra et teknisk standpunkt skal de alle være implementert. Til og med selv om vi ikke har rukket å lage søketester for alle.

✓ Søk etter (usn.obj2100.server.clientTests.search)	661 ms
✓ (Testing) implementerings test	356 ms
✓ en gjenstand som eksisterer	41 ms
✓ en gjenstand som ikke eksisterer	19 ms
✓ noe og filtrer etter beskrivelse	17 ms
✓ noe og filtrer etter beskrivelse og forventet kassering	9 ms
✓ noe og filtrer etter beskrivelse og innkjøpsdato	10 ms
✓ noe og filtrer etter beskrivelse og kategori	7 ms
✓ noe og filtrer etter beskrivelse og levetid	15 ms
✓ noe og filtrer etter beskrivelse og plassering	11 ms
✓ noe og filtrer etter beskrivelse, kategori og forventet kassering	8 ms
✓ noe og filtrer etter beskrivelse, kategori og innkjøpsdato	7 ms
✓ noe og filtrer etter beskrivelse, kategori og levetid	8 ms
✓ noe og filtrer etter beskrivelse, kategori og plassering	12 ms
✓ noe og filtrer etter beskrivelse, kategori, plassering og forventet kassering	12 ms
✓ noe og filtrer etter beskrivelse, kategori, plassering og innkjøpsdato	8 ms
✓ noe og filtrer etter beskrivelse, kategori, plassering og levetid	11 ms
✓ noe og filtrer etter beskrivelse, kategori, plassering, levetid og forventet kassering	25 ms
✓ noe og filtrer etter beskrivelse, kategori, plassering, levetid, forventet kassering og innkjøpsdato	15 ms
✓ noe og filtrer etter beskrivelse, kategori, plassering, levetid, innkjøpsdato og type	8 ms
✓ noe og filtrer etter forventet kassering	8 ms
✓ noe og filtrer etter innkjøpsdato	6 ms
✓ noe og filtrer etter kategori	7 ms
✓ noe og filtrer etter levetid	19 ms
○ noe og filtrer etter om den er i bruk	
✓ noe og filtrer etter plassering	8 ms
✓ noe og filtrer etter pris	6 ms
✓ noe og filtrer etter type	8 ms

En annen ting å bemerke er at serveren støtter både direkte lookups view CRUD klienthåndteringen, så klienter kan selvfølgelig hente all dataen de trenger og bare loope over det sånn også. Så det er teknisk sett dobbelt implementert i den forstand. Men søkebyggeren eksisterer for å minske ressursbruk og unødvendig nettverkstrafikk.

Se til “Forbedringer” for kommentar på hva vi kunne gjort bedre i relasjon til server-side testing og Inventar og InventarExtended objektene.

Kapittel 44 av læreboken (Liang, 2021) og dokumentasjonen til JUnit5 (Bechtold et al., n.d.) ble brukt som referanser for hvordan å sette opp tester i Java. Kapittel 33 av læreboken (Liang, 2021) ble også brukt som referanse for basiske konsepter og ideer for klient-server design.

Databasestruktur

Vi gjennomførte 3NF på database designet, men bestemte oss tidlig også om at databasestrukturen burde holdes noe enkelt. Ettersom at eksamen vil gjøre det mer sannsynlig å påføre nødvendige endringer under sprintene.

Det er 6 tabeller i alt.

- Inventar: Hovedtabellen. Den har fremmednøkler til plassering, kategori og kassert tabellene.
- Plassering: Referansetabell som inneholder lokasjoner (plasseringer).
- Kategori: Referansetabell som inneholder kategorier, f.eks. bord, stol, osv.
- KategoriType: Referansetabell til Kategori (FK), inneholder kategorityper, f.eks. møbler, utsmykning, osv.
- Kassert: Referansetabell som inneholder informasjon om hvilke inventar elementer har blitt tatt ut av bruk.
- KassertType: Referansetabell til Kassert (FK), inneholder kasserttyper, f.eks. solgt, kassert, osv.

Inventar tabellen inneholder informasjon om et inventar element, inklusivt når det ble kjøpt og antallet som ble kjøpt da.

“forventetLevetid” kolonnen har en standardverdi “null.”

Kategori kolonnen er en fremmednøkkel som viser til kategori tabellen, hvor det er referanser til forskjellige kategorier. Den har en underliggende referansetabell hvor man har definerte "typer," slik som om det gjelder møbler, utsmykning eller teknisk utstyr.

Plassering kolonnen er en fremmednøkkel som viser til plassering tabellen, hvor man må ha bygg og floy (fløy) definert, men trenger ikke å ha etasje eller rom definert.

Kassert kolonnen er en fremmednøkkel med en standardverdi “null,” som viser til kassert tabellen. Denne har en underliggende referansetabell, som inneholder begrunnelsene “solgt,” “kassert,” “på lager” og “annet.”

kassertType	
id 🔑	integer
begrunnelse 📄	string

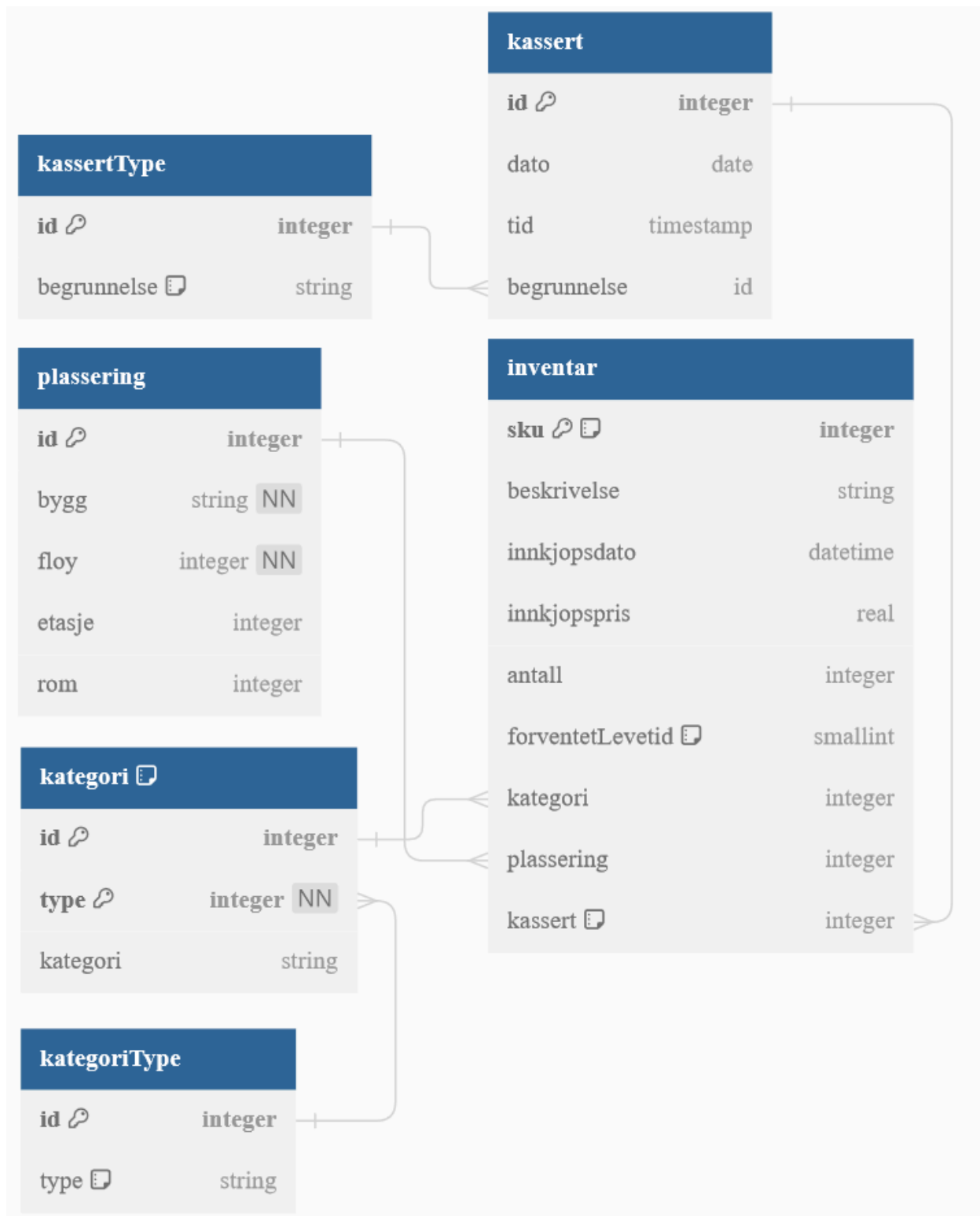
plassering	
id 🔑	integer
bygg	string NN
floy	integer NN
etasje	integer
rom	integer

kategori 📄	
id 🔑	integer
type 🔑	integer NN
kategori	string

kategoriType	
id 🔑	integer
type 📄	string

kassert	
id 🔑	integer
dato	date
tid	timestamp
begrunnelse	id

inventar	
sku 🔑 📄	integer
beskrivelse	string
innkjopsdato	datetime
innkjopspris	real
antall	integer
forventetLevetid 📄	smallint
kategori	integer
plassering	integer
kassert 📄	integer



Klientapplikasjonen

Klientapplikasjonen fungerer som front-end for systemet og gir et grafisk brukergrensesnitt (GUI) som gjør det mulig for brukerne å utføre operasjoner som å legge til, oppdatere, søke og slette poster i inventaret. Kommunikasjonen med serveren skjer via sockets.

Hovedkomponenter

JavaFX-applikasjon:

- **GUI:** Bygget med JavaFX, gir et interaktivt grensesnitt som lar brukerne utføre ulike operasjoner.
- **Controller:** Hver del av GUI-en har en tilhørende kontrollerklasse som håndterer brukerinteraksjoner og formidler data mellom GUI og serveren.”
- **ScreenController:** Controller for de fleste av skjermbildene. Den kaller på såkalte views som har kode til å fremvise et skjermbilde med elementer basert på data regnet ut i Controller klassene.
- **Views:** Klienten har også forskjellige views, de fleste av disse blir kalt i ScreenController, og noen ved initiering i ClientView(hoved view). Ansvaret til view-klassene er å presentere grafikk basert på databehandling i kontrollere.

Nettverkskommunikasjon:

- **Socket-forbindelse:** Bruker sockets for å kommunisere med serveren. Klienten oppretter en socketforbindelse til serveren for å sende og motta data.
- **Object Streams:** `ObjectOutputStream` og `ObjectInputStream` brukes for å overføre objekter mellom klienten og serveren.

Brukergrensesnitt (GUI)

GUI-en er designet for å være intuitiv og brukervennlig. Her er noen av de viktigste komponentene:

Hovedvindu:

- **Navigasjonsmeny:** Tilbyr navigasjon til forskjellige deler av applikasjonen som inventar, kassering, kategorier og plasseringer.

Inventar:

- **Listevisning:** Viser en liste over alle inventarposter.
- **Detaljvisning:** Lar brukeren se detaljer om en valgt inventarpost. (fungerer bare etter søk)
- **Søkefunksjon:** Lar brukeren søke etter spesifikke inventarposter basert på ulike kriterier. både enkelt og avansert søkefunksjon.
- **CRUD Operasjoner:** Lar brukeren opprette, lese, oppdatere og slette inventarposter. (fungerer bare etter søk)

Kategorier og Plassering:

- **Administrasjon:** Lar brukeren administrere kategorier og plasseringer.
- **CRUD Operasjoner:** Tilbyr funksjoner for å opprette, lese, oppdatere og slette kategorier og plasseringer. (var planen, men dette rakk vi ikke få til, det ligger klart på server, men vi brukte for lang tid på å implementere klient, til å rekke CRUD på mer en bare Inventar. Selv ikke på Inventar er det mulig å stille på kategorier eller type fordi vi ikke hadde tid til å implementere dette, igjen ligger det klart for det på server-side.

Interaksjon med Brukeren

Navigasjon: Brukeren navigerer gjennom applikasjonen ved hjelp av navigasjonsmenyen. Hver menyvalg åpner en ny del av applikasjonen (f.eks. Inventar, Kassering). (Var planen, per nå har vi bare navigasjon via søkebaren (som aktiveres ved å trykke på søke knappen nede til høyre), legg til nytt inventar-knappen på startsidene eller se på alle inventar knappen).

Data Manipulasjon: Brukeren kan utføre operasjoner som å legge til, oppdatere og slette data gjennom intuitive skjemaer og tabeller. (Sletting er vi ikke helt ferdig med på klient siden, vi fikk lagt til noe i siste minut som som sletter inventar, men det er jo ikke meningen. Vi ville kassere inventar, ikke slette fra databasen. Redigering, lesing og lage ny fungerer, men noe av dataen er hardkodet og ikke basert på input fra bruker som det er ment)

Søk: Brukeren kan søke etter spesifikke inventar ved å bruke søkebaren som er tilgjengelig i hver modul. Det finnes også et avansert søk om man trykker på grønne knapp i søkebar, avansert søking er ikke ferdig implementert.

Interaksjon med Tjeneren

Opprette Forbindelse: Når klientapplikasjonen startes, oppretter den en socketforbindelse til serveren.

Sende Forespørsel: Når brukeren utfører en handling (f.eks. legger til en ny inventar), sender klienten en forespørsel til serveren gjennom socketforbindelsen.

Motta Respons: Serveren behandler forespørselen og sender et svar tilbake til klienten. Klienten mottar dette svaret og oppdaterer GUI-en i henhold til responsen (f.eks. viser en bekreftelse på at inventarposten ble opprettet).

Feilhåndtering: Hvis serveren returnerer en feil (f.eks. på grunn av en ugyldig forespørsel), vises en feilmelding i GUI-en, og brukeren får mulighet til å rette opp feilen. Retroperspektivt skulle vi ønske oss en bedre feilhåndtering, det er ikke alle feil som blir plukket opp eller behandlet riktig, for eksempel om en mister forbindelse til tjener, kunne man blitt presentert med en “reconnect”-knapp.

Utfordringer

Vi benyttet oss originalt av et ordinært Maven prosjekt via JetBrains IntelliJ, men litt ut i starten av løpet valgte vi å splitte prosjektet inn i et multi-modulært Maven prosjekt, for logisk

separasjon av kode. Dette funket flott de 2 første dagene, og for de fleste i gruppen fortsatte det å fungere flott.

Men på den tredje dagen, altså den femtende, så opplevde vi plutselig noe problemer med at prosjektet ikke kunne finne den delte modulen for en i gruppen vår som benyttet seg av Apple MacOS. Vi tok en del tid for å prøve å løse dette, da uten det ville vi jo hatt mye mindre arbeidseffektivitet. Men etterhvert måtte vi bare gi opp og re-prioritere, da vi ikke kunne finne en tydelig begrunnelse for problemstillingen. Det var dessverre ikke noe så enkelt som at Apple MacOS er case sensitive per default og Microsoft Windows ikke er per default, men man kan jo alltid drømme.

Vi endte opp å måtte benytte oss av JetBrains IntelliJ “Code With Me,” som tillater personer med lenken og som blir godkjent, til å få tilgang til IDEet. Dette virket som at det funket bra, foruten at JetBrains ikke har implementert en god del funksjonalitet, som gjør utviklingsprosessen noe verre, da det øker utvikling friksjonen.

Vi fant også ut at det var en figurativ gigantisk kniv som ventet på oss rundt hjørnet. “Code With Me” funker fint, den er responsiv og ganske grei å bruke, i en stund. Etter en varierbar stund så ble den tregere og tregere og tregere. Slik vi kunne se så bruker den unødvendig mye ressurser, oppi 20GB RAM, og nesten alt av CPU kraft tilgjengelig, men det verste var at det var aktivitet spikes hele tiden.

Det var snakk om treghet hvor bare å skrive en setning kunne ta bokstavelig talt opptil ett minutt på det aller verste.

Til syvende sist fikk vi satt opp en ekstra Windows maskin, og byttet over til den, og siden da har vi jobbet like effektivt som før. Men vi føler vi riktig nok har mistet ganske mye tid over mystiske feil som oppstod fra multi-module Maven prosjektet fordi IntelliJ virker til å hele tiden prøve å styre det selv, og ikke la Maven gjøre det, som gjør at vi ikke kunne inkludere deler av konfigurasjonen i VCS slikt som git. Og vi mistet også en god del tid på å prøve alle til å kunne

kode, siden vi kunne jo ikke riktignok kaste vekk en person bare på grunn av et uforventet problem.

Skuffende er det nå uansett, da vi føler vi kunne gjort mye bedre dersom vi hadde hatt mer tid, men det er slik det er. Og vi håper vi har lagd noe adekvat nok. Det vil derfor bemerkes at eventuelle mangler i Ulet, i den forstand at viss funksjonalitet det bare ikke funker, betyr ikke at funksjonaliteten ikke er implementert hos serveren og klienten's back-end.

Forbedringer

Forbedringer i klient

Vi har fått til så mye på tjenersiden at det er veldig leit at vi ikke har rukket å implementere alt vi hadde planlagt på klient. Bl.a. så skulle vi blitt ferdig med en sikker eventHandler klasse som tar seg av eventhandling for alt av interaktive elementer. Vi skulle gitt bedre tilbakemelding på RangeInput i avansert søk, og vi skulle fått avansert søk implementert ferdig med eventHandler, dette rakk vi ikke.

Hjelp-knapp med hjelpe-skuff

Vi har lagt til en søkehjelp i klienten, men den er litt rar, fungerer bare første gang og animasjonen er ikke helt som det skal være, men vi hadde ikke mer tid til den. Man kan aktivere den ved å trykke på Spm.tegnet i søkefeltet. Vi kunne tenkt oss å forbedre animasjonen, og fått teksten til å laste hver gang man trykker på den.

Hovedmeny

Vi har lagt til en hovedmeny på klienten, men vi har ikke rukket å implementere den, det skulle vi gjerne ha gjort da det er raske snarveier til de forskjellige gjøremålene på klient.

Avansert søk

Vi skulle hatt likt å fått brukt mer av det vi la opp til på tjener-del mtp. søk i Inventar. Vi fikk ikke tid til å jobbe noe særlig med avansert søk GUI og event-handling, da vi fikk problemer med OS-kompilator. Vi hadde planer å kombinere søk med alle de forskjellige inputfeltene i avansert søk, men dette blev det altså ikke tid til.

CRUD på alle inventar - visning

Når man starter klient og går direkte på se alle inventar, så vill ikke CRUD fungere på inventarene. Dette er fordi vi gikk fra en Inventar modell til en InventarExtended. Vi rakk ikke å oppdatere hele klienten med den nye datatypen, og får da problemer ved visning på søkeskjermen, når vi skal vise data som ikke egentlig er et søk. Hadde vi hatt mere tid, ville vi ha implementert InventarExtended fullstendig i klient/tjener, og fått tilgang til CRUD som fungerer på både søk og på alle inventar.

Sletting/Kassering av inventar

Vi har også lyst til å forbedre sletting av inventar. Det var tenkt at vi skulle ha et skjermbilde for sletting hvor vi velger årsak for kassering. per nå har vi bare sletting av inventaret i db, som ikke funker helt som det skal. Programmet blir ødelagt om vi sletter et inventar, da kan man ikke lenger søke eller gjøre operasjoner. Vi kunne tenkt oss å ferdigstille denne delen.

Rediger/lag ny inventar

Vi har fremdeles noe hardkodet data i ny inventar/rediger inventar. Det ble ikke nok tid til å implementere dataen fra den strukturen vi hadde laget. Det ble litt for komplekst med id-referanser til kategorier og typer, vi vil gjerne bli ferdig med denne delen så man kan reelt benytte seg av inventar-CRUD.

Se på inventar

Vi kunne tenkt oss å markere forskjellige data og datatyper i dette skjermbildet. Vi føler at det ikke holder med "spacing" på lablene som vises frem, og kunne tenkt oss å ha disse i systematisk forskjellige bokser. Dette ble det ikke tid til.

Struktur på klient

Vi er nokså fornøyd med strukturen på klient. Det vi ikke er så fornøyd med er den repetitive koden som ble lagt til som en "last minute" løsning på en del områder. Vi la til noe kode som må refaktoreres som bruker dobbelt opp av metoder som kunne vært enkeltmetoder som var generisk ovenfor alle søk og visninger av flere inventar. Dette hadde vært en av de første tingene vi kunne tenke oss å fikse på klient.

Bruk av Panes/Box

Vi kunne også tenkt oss å gått over bruken av forskjellige panes og box for å se at vi har gjort de riktige valgene. Vi har en følelse av at det kan ha bli gjort feil i hui og hast da det ikke var spesielt god tid til systematisk og grundig analyse i bruk av elementer til forskjellige situasjoner. Det kan også tenke seg at vi kunne hatt bruk for en generalisering i bruk av panes og HBox / VBox for å forhindre gjentakelse og / eller ekstra kode rundt om kring.

Forbedringer i shared modulen

Models

Models, per i dag, er separate definisjoner av database objekt og reflekterer database strukturen relativt 1-til-1. De er Serializable, og behandles gjennom Objects i client og server modulene. Det ville nok vært gunstig å ha hatt en superklasse som utvider Objects, slik at vi kunne lagt til delte metoder slik som å kunne enkelt sjekke hvilken type de, lagring for kommandoer sammen med disse objektene, og metoder for å pakke ut objektene til individuelle variabler og individuelle objekter når det er snakk om lister av objekter.

Noe slikt kunne nok simplificert responsbehandlingen i server modulens ClientHandler klasse.

Client og FakeClient

Under utviklingen av klientsimulering for testing i server modulen ble en klasse i dens test mappe lagd som heter "FakeClient." Denne klienten implementerte originalt funksjonaliteten for å kommunisere med serveren, og man kan se at vi bruker mye av den samme koden i client modulens Client.java klasse.

Ettersom at disse deler en del kode, og har samme bruksområde, så ville det nok vært gunstig å ha slått dem sammen og plassert den sammenslåtte Client inn i shared modulen, slik at den kan brukes av både client modulen og server modulen. Slik kunne vi ha garantert for at klient simulering i client-scope tester i server modulen vil faktisk reflektere den ekte klienten. Slik det er nå kan Client og FakeClient ende opp med å gro mer ulike over tid, som kan forårsake tester som fungerer hos serveren's simulerte klienter, men ikke for ekte klienter.

I samme gren ville det nok vært gunstig å flytte over client modulens ClientController klasse til shared modulen, da det er gjennom denne ordinære klienter er ment til å kommunisere med Client klassen. Nesten som at Client egentlig er en Service i kontekst av MVC.

Inventar og InventarExtended models

Inventar modellen er den originale klassen som definerer forholdet mellom objekt-og-database. Senere i prosjektets utvikling ble forhold endret, se “Modularisering” og “Testing,” og det ble opprettet en ny model som var ment til å inneholde data fra andre tabeller relatert til det objektet gjaldt for. Altså slikt som Inventar objektet’s kategori, plassering og kassering, slik at man ikke bare får tilbake Inventar objektet med fremmednøklerne for kategori, plassering og kassering istedenfor dems datum.

Siden de er nesten identiske ville det vært veldig gunstig å ha hatt en felles Inventar superklasse, og så noe som en InventarSimple og InventarExtended underklasse som inneholder variabler og funksjoner unikt til dems bruksområde. Det ville, kortsiktig, minsket kode duplikasjon. Og langsiktig, ville dette sikre mot at Inventar og InventarExtended modellene blir avvikende fra hverandre, som kan og vil føre til uforventet oppførsel når man bruker Inventar ovenfor InventarExtended.

I det minste ble InventarExtended lagd for litt de samme grunnene at man har primitive typer som “int” og også mer avanserte som “Integer.”

Vedlegg

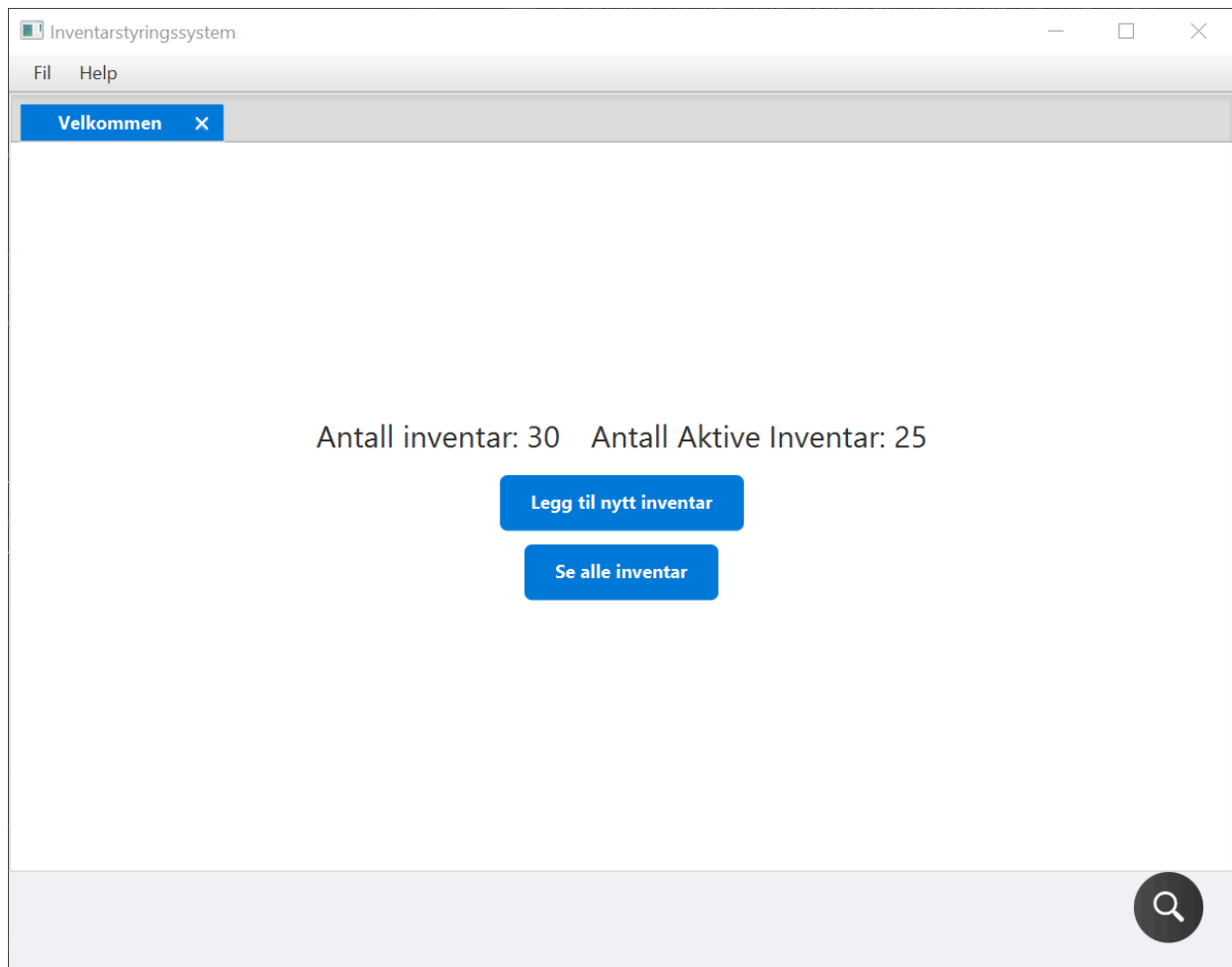
Hvordan bygger man prosjektet?

Dersom det skulle være ønskelig eller nødvendig å bygge prosjektet, så kan man se til “README.md” filen i root av prosjektet innkludert i zip filer. Den forklarer nødvendige steg for å komme seg opp og kjøre, og det er denne prosessen vi har brukt mens vi har utviklet eksamensløsningen.

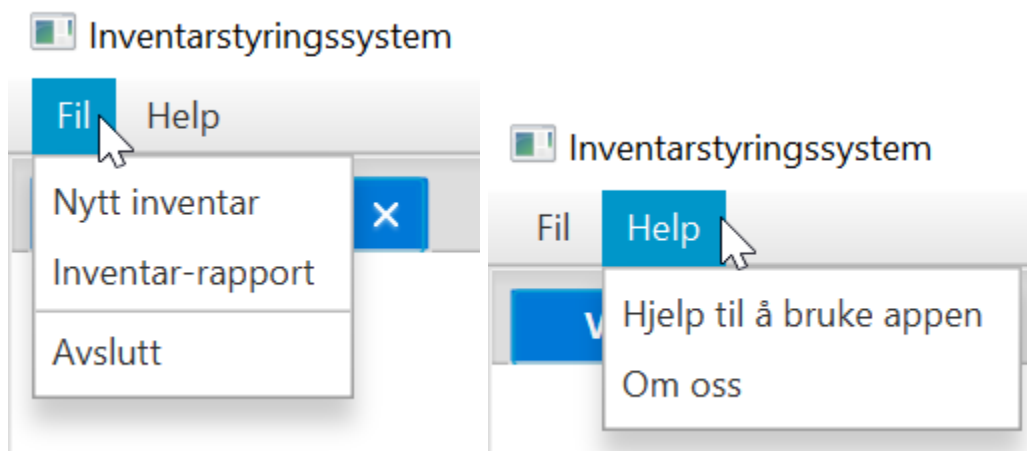
Eksempler av klient (bilder)

I den medfølgende zip filen, under mappen “docs” så ligger det også en kort lydløs videosnutt som viser klienten i bruk. Dersom det skulle være noen problemer med at prosjektet ikke vil bygges, f.eks. med at det prøves å bygges på MacOS, og antagelig Linux også, så kan man også bruke den for å få en ide på hvordan ting ser ut og henger sammen.

Start side



Menu Bar



Se alle inventar elementer

Inventarstyringssystem

FilHelp

VelkommenAlle inventar

Beskrivelse	Innkjøps Pris	Innkjøps Dato	Actions
Stol	100.0 kr	2022-01-01T10:00	<div>ViewEditDelete</div>
Maleri	200.0 kr	2022-02-01T11:00	<div>ViewEditDelete</div>
Lyse brun, mahogany	300.0 kr	2024-12-24T22:38:21	<div>ViewEditDelete</div>
Gaming stol/pult kombo ting	50.0 kr	2021-01-13T12:14:13	<div>ViewEditDelete</div>
Bilde av mamma	74.38 kr	2025-01-13T12:14:13	<div>ViewEditDelete</div>
Description 6	100.0 kr	2022-01-01T10:00	<div>ViewEditDelete</div>
Description 7	200.0 kr	2022-02-01T11:00	<div>ViewEditDelete</div>
Description 8	300.0 kr	2022-03-01T12:00	<div>ViewEditDelete</div>
Description 9	400.0 kr	2022-04-01T13:00	<div>ViewEditDelete</div>
Description 10	500.0 kr	2022-05-01T14:00	<div>ViewEditDelete</div>

Nytt inventar

Inventarstyringssystem

Fil

Help

Velkommen

Nytt inventar

×

Type:

Kategori:

Innkjøpsdato:

Innkjøpspris:

Plassering:

Antall:

Forventet levetid (kun møbler):

Beskrivelse:

Legg til

Inventarstyringssystem

FilHelp

VelkommenNytt inventar

Type:

Mobler

Kategori:

Skap

Innkjøpsdato:

17.05.2024

Innkjøpspris:

212

Plassering:

USN-Bø

Antall:

33

Forventet levetid (kun møbler):

3

Beskrivelse:

Brun, mønstret.

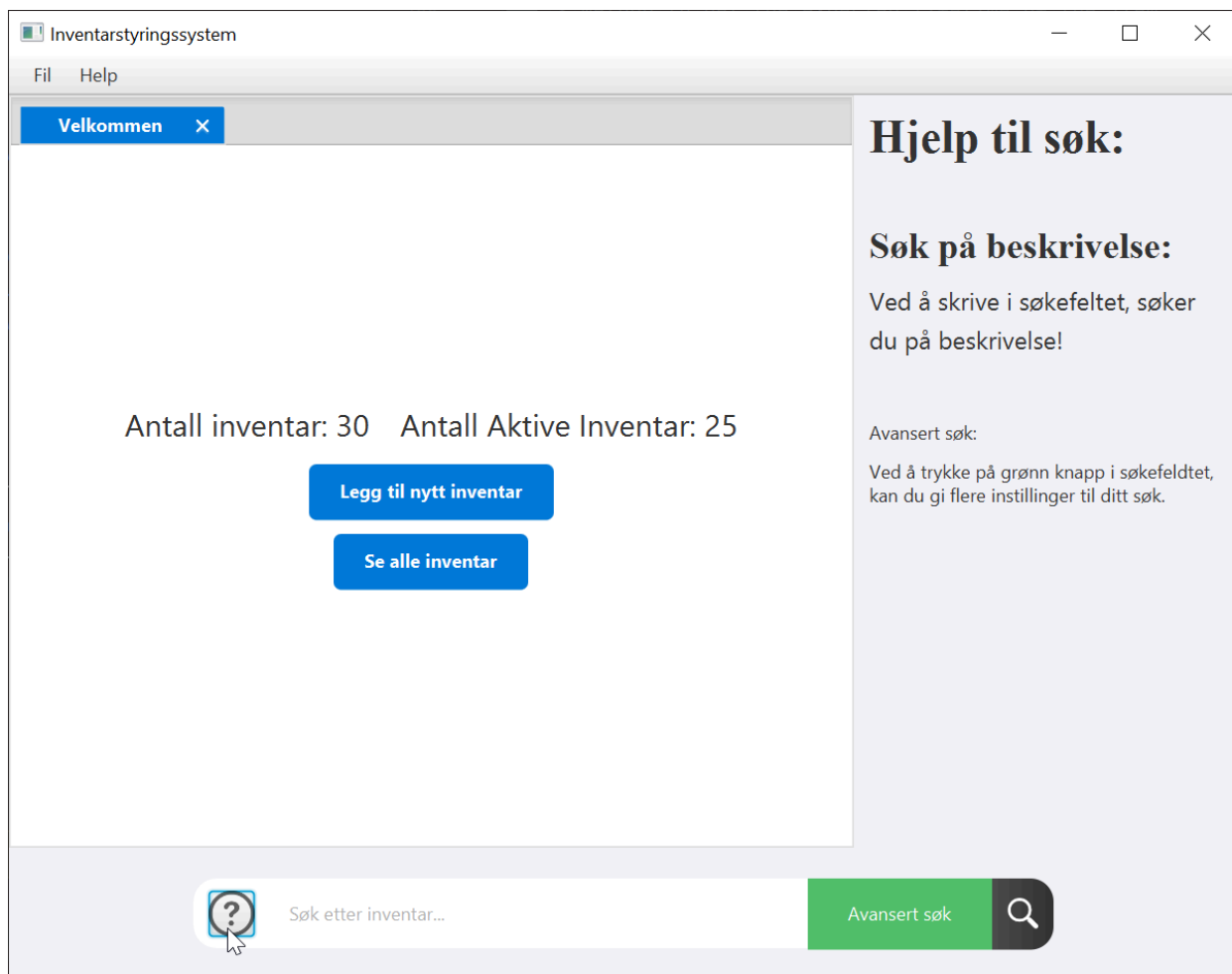
Legg til

Søkemeny

?

Søk etter inventar...

Avansert søk



Inventarstyringssystem

FilHelp

Avansert søk

Inventartype

Velg inventartype...

Velg kategori...

Beskrivelse

Søk på beskrivelse...

Plassering

Søk på plassering...

Innkjøpsår

Velg år...

Innkjøps pris

Forventet levetid

Søk

Velkommen

Antall inventar: 30 Antall Aktive Inventar: 25

Legg til nytt inventar

Se alle inventar

Søk

Ikke sortert:

Inventarstyringssystem

Fil

Help

Velkommen

stol

Beskrivelse	Innkjøps Pris	Innkjøps Dato	Actions
Stol	100.0 kr	2022-01-01T10:00	<div>View</div> <div>Edit</div> <div>Delete</div>
Gaming stol/pult kombo ting	50.0 kr	2021-01-13T12:14:13	<div>View</div> <div>Edit</div> <div>Delete</div>

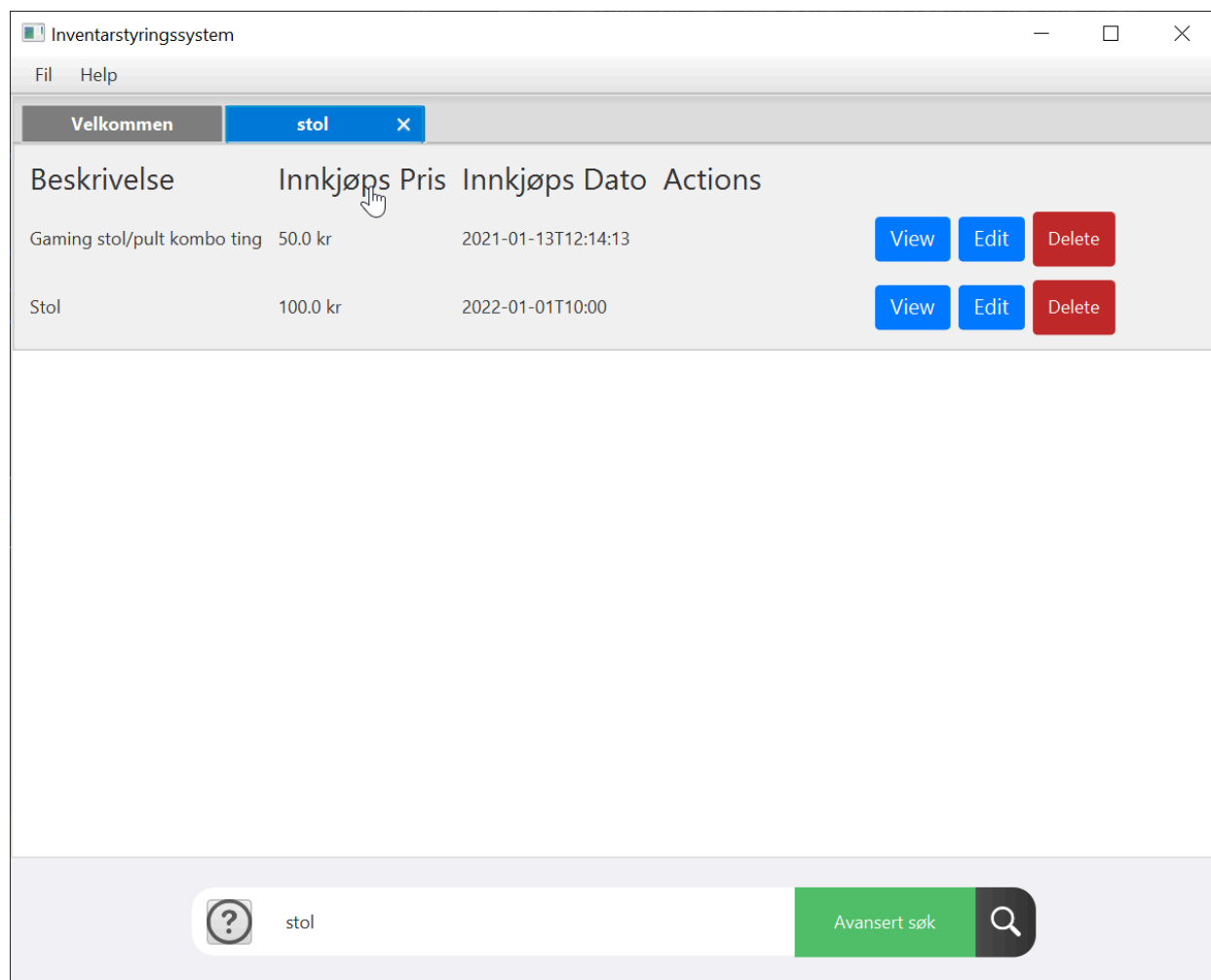
?

stol

Avansert søk

Q

Sortert:



Klikket på "View:"

Inventarstyringssystem

Fil Help

Velkommen stol Se på Gaming stol/pul X

Beskrivelse:

Gaming stol/pult kombo ting

Type:

Møbler

Kategori:

testkategori

Innkjøpsdato:

2021-01-13T12:14:13

Innkjøpspris:

50.0

Plassering:


Bygg B/2/3/202


Antall:

85

Forventet levetid:

3

 stol

Avansert søk 

Klikket på “Edit:”

Inventarstyringssystem

FilHelp

VelkommenstolRediger Gaming stol/pul X

Type:

Møbler

Kategori:

Innkjøpsdato:

Innkjøpspris:

50.0

Plassering:

Bygg B/2/3/202

Antall:

85

Forventet levetid (kun møbler):

3

Beskrivelse:

Legg til

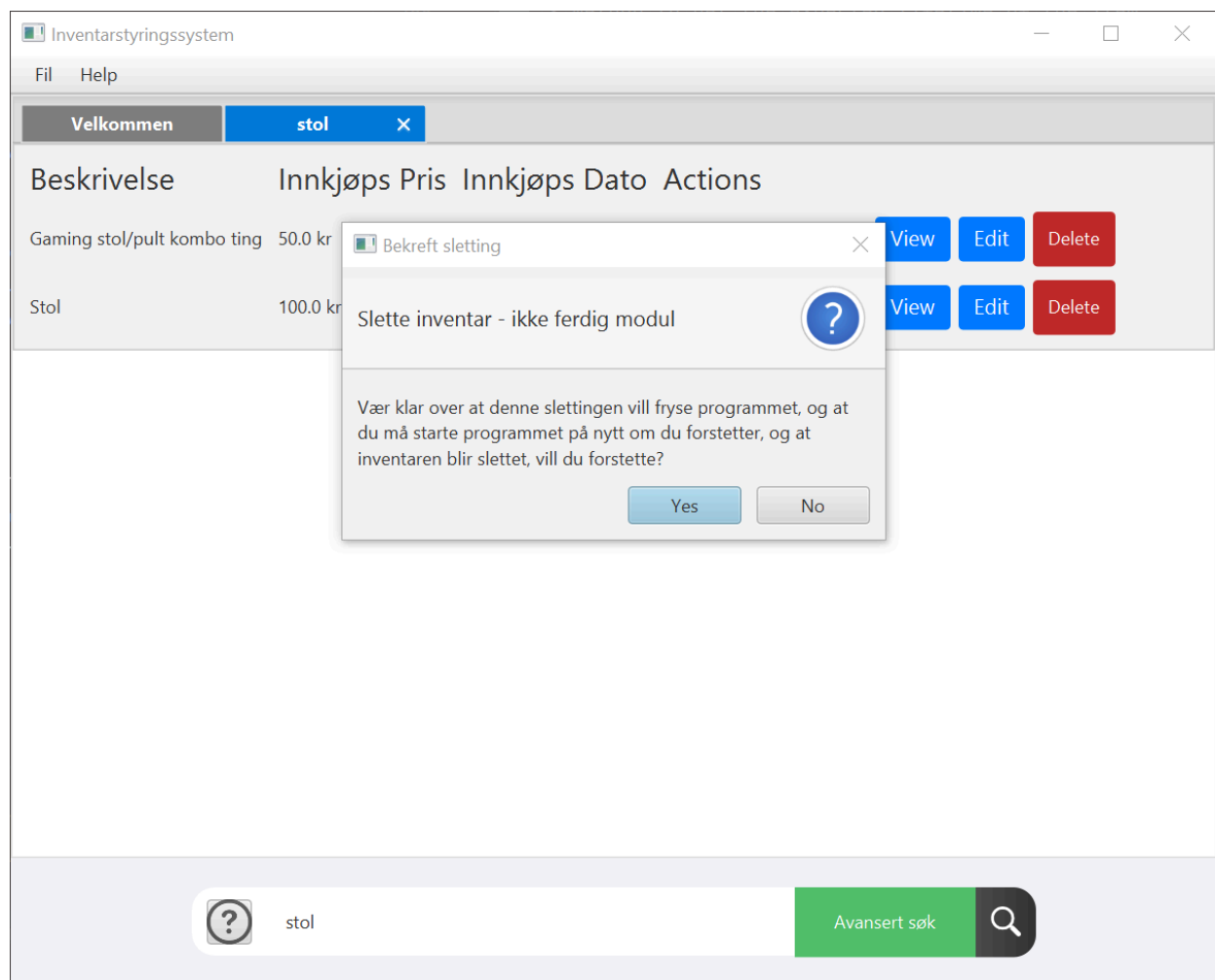
?

stol

Avansert søk

Q

Klikket på “Delete:”



Litteraturliste

Bechtold, S., Brannen, S., Link, J., Merdes, M., Philipp, M., Rancourt, J. D., & Stein, C. (n.d.). *JUnit 5 user guide*. JUnit5. Hentet mai 13, 2024, fra <https://junit.org/junit5/docs/current/user-guide>

Liang, Y. D. (2021). *Introduction to Java programming and data structures, comprehensive version* (12th ed.). Pearson.

JetBrains. (2023, februar 17). *Multi-module projects*. Hentet mai 13, 2024, fra <https://www.jetbrains.com/guide/java/tutorials/marco-codes-maven/multi-module-projects/>