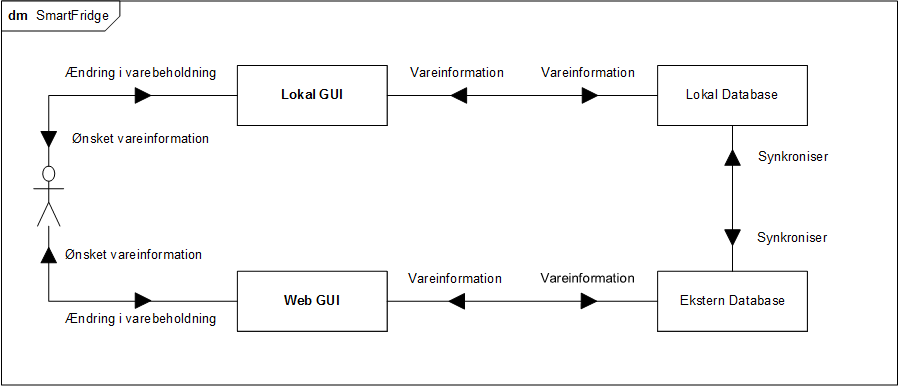
# Systemarkitektur

## Domænemodel

Domænemodellen (Figur 1) beskriver den overordnede kommunikation i systemet. Brugeren interagere enten med den lokale GUI eller Web GUI’en. Den lokale GUI får sin information om varebeholdningen fra den lokale database, og har mulighed for at tilføje, fjerne og ændre i data’en. Web GUI’en får sin information fra den eksterne database og har samme muligheder for ændring af data som den lokale GUI. Den lokale database og den eksterne database synkroneres af applikationen som styrer den lokale GUI.

Domænemodel: Kommunikationen imellem de to databaser kan ikke være ”synkronisering” måske ”synkroniser data” ?



Figur 1 - Domænemodel af SmartFridge

## Fridge app

Følgende sekvensdiagrammer beskriver hver forløbet af en use case.



Figur 2 – Sekvensdiagram for UC Se Vare

Sekvensdiagrammet på Figur 2 beskriver hvordan de forskellige lag i applikationen kommunikere med hinanden. Brugeren interagere med UI, hvorefter Logik laget sørger for de rigtige data bliver vidst på UI’et. Logik-laget henter dens information fra datalaget. Se vare use casen dækker over flere lister. Det varierer hvilke data der bliver hentet til logik laget afhængigt af hvilke liste der vælges af brugeren.

Der mangler kald efter ”vælger data”. UI’en kalder sig selv med ”Vis vare data”.

 Figur 3 – Sekvensdiagram for UC Tilføj Vare

Tilføj vares loop påbegyndes når Bruger indtaster vareinformationerne. Bruger kan tilføje lige så mange vare han/hun ønsker indtil der vælges ”Tilføj og Afslut”. De varer der tilføjes, før der trykkes på ”Tilføj og Afslut”, skal kunne ses i en midlertidig liste.

Hver enkelt vare bliver ikke gemt direkte ned i datalaget når der trykkes tilføj. De gemmes i en intern liste, hvilken der bliver kørt igennem og tilføjet til databasen når der trykkes ”tilføj og afslut”



Figur 4 – Sekvensdiagram for UC Fjern Vare

Figur 4 forudsætter at der allerede er tilføjet et item til den liste hvor item skal fjernes fra.



Figur 5 – Sekvensdiagram for UC Synkronisér

Synkroniseringen mellem den lokale- og den eksterne database, bliver i denne use case påbegyndt af brugeren, men sker også automatisk vha. et framework hvert 10. minut. Lykkedes den påbegyndte synkronisering ikke vil dette bliver indikeret i GUI’en.Kaldet ”påbegynd synkronisering” er ikke asynkron, hvilket vi fandt ud af til acceptesten i dag. Måske det skal rettes i programmet?



Figur 6 – Sekvensdiagram for UC Rediger Vare

## Web app

Sekvens diagrammerne for web applikationen er opdelt i to. En klient-side hvor applikationen kan tilgås fra en enhed med internet forbindelse og server-side hvor applikationen kørers på og returnere html kode til klient-sidens browser. Forbindelsen imellem klienten og serverne sker vha. http-protokollen. Databasen der refereres til i følgende sekvensdiagrammer symbolisere den eksterne database.



Figur 7 – Sekvensdiagram for hvordan en liste åbnes Skal kaldende fra view->controller, fra controller->DAL og fra DAL ->Database ikke være synkrone kald?

Sekvensdiagrammet ’Åben liste’ skal være udført før sekvensdiagrammerne for AddItem, DeleteItem og EditItem på henholdsvis Figur 8, Figur 9 og Figur 10, kan udføres. Diagrammerne beskriver hvordan der kommunikeres imellem lagene.



Figur 8 – Sekvensdiagram for UC Add Item Stavefejl i ”kalder AddItemViewl” Er det samme skrifttykkelse på alt tekst i samtlige diagramer i dette dokument? I stedet for indtaster data sker kaldet først når der trykkes tilføj. Indtaster data skal stadig være der, men der sker ikke noget før der trykkes tilføj. Ligesom I WPF gemmes data først i databasen når der trykkes ”tilføj og afslut”. Samme som forrige med synkrone kald. Også for delete Item



Figur 9 – Sekvensdiagram for UC Fjern Item



Figur 10 – Sekvensdiagram for UC Edit Item Pilene er rammer ikke helt hvor de skal halvejs igennem diagrammet. Synkrone kald.