# Design, implementering og test - Database

I dette afsnit vil designprocessen, implementering samt test af database-delen af systemet blive beskrevet. Her vil blive beskrevet Data Access Layer for både Fridge App og Web App, med de overvejelser der er blevet gjort i designprocessen og implementering af DAL for begge applikationer.

For mere detaljeret gennemgang, henvises til Design, implementering og test for databasen i projektdokumentation.

## Fridge app

I dette afsnit vil designprocessen, implementering samt test af database-delen for Fridge app’en blive beskrevet, samt de overvejelser der er blevet gjort for database-tilgang fra applikationen.

### Design

I dette afsnit vil designprocessean blive beskrevet, hvor de designovervejelser der er gjort i forhold til DAL for Fridge App.

#### Teknologi

Før at der kunne udarbejdes et design, var det nødvendigt først at bestemme hvilken teknologi man ville anvende til at implementere DAL. Her stod de oplagte valg mellem ADO.NET og Entity Framework, da det var de teknologier der var blevet nævnt i database-kursuset. Her blev det valgt at anvende ADO.NET for lærings skyld, samt at der ville blive overvejet at anvende Entity Framework til Web app’en.

#### Objekt model

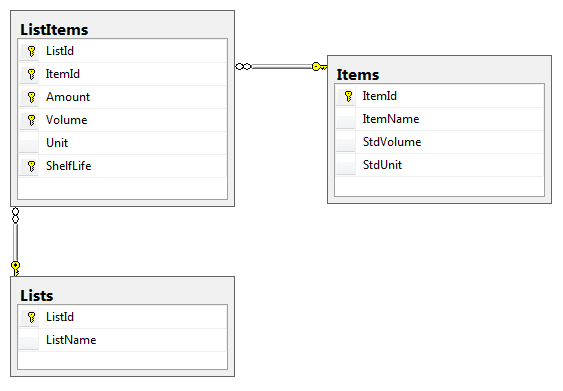
Da der arbejdes med relationelle databser, skulle der udarbejdes en objekt model, hvor at data kunne gemmes korrekt. Her blev det diskuteret hvilke objekter der var en nødvendig at kunne gemme i databasen. Her blev de følgende entities identificeret:

* Liste (List) – F.eks. køleskab, indkøbsliste. Disse lister skal kunne indeholde de varer som brugeren sætter i køleskabet og hvad de har på deres indkøbsliste. Denne indeholder listens navn.
* Vare (ListItem) – Den konkrete vare, som er på en given liste. Det er f.eks. 2x 1 liters mælk i køleskabet. Dette indeholder antal, mængde, enhed og holdbarhedsdato.
* Varetype (Item) – For at brugeren ikke skulle indtaste varetypen hver gang, er varetyper gemt, som f.eks. mælk, æg, hakket oksekød osv. Varetypen skal også have en standard mængde og enhed.

Da disse entities var identificeret, kunne der udarbejdes en objekt model, som ses på Figur 1, der er udarbejdet vha. DDS-lite. Der er udnyttet et Mange-til-mange mellem ’List’ og ’Item’, hvor weak entityen ’ListItem’ er imellem. På den måde var det muligt at have den konkrete vare og varetypen adskilt på en fordelagtig måde. Objektmodellen med attributter kan ses på Figur 2.

C:\Users\Mathis\Desktop\SmartFridgeV1.jpg

Figur Objekt model



Figur Objekt model med attributter

#### Repository mønster

Som en del af designet, er Repository mønstret blevet anvendt. Repository mønstret giver mulighed at have ét sted, hvor at alt databasetilgang foregår. Formålet med dette er at vi seperare databasetilgangen for forretningslogikken, der er med til at gøre systemet lettere at vedligeholde, samt lettere læsligt. Udover det, giver Repository mønstret også mere testbarhed, da det vil være muligt at mocke repositoryet ud og derved kan gøre mere forretningslogik testbart. For en mere detaljeret beskrivelse af repository mønstret, henvises til Repository mønster-afsnittet i projektdokumentationen.

### Implementering

### Test