Designmønstre – dokumentation

I dette afsnit, vil der blive beskrevet de forskellige designmønstre der er blevet anvendt i dette system, samt hvordan de er bruges af systemet.

# Repository Pattern

https://msdn.microsoft.com/en-us/library/ff649690.aspx

I mange applikationer, er det forretningslogikken der tilgår data og databaser. Dog kan direkte tilgang introducere indtil flere problemer, som gentagne kode, højere risiko for programmeringsfejl, dårlig abstraktion og lav testbarhed.

For at kunne løse disse problemer, introduceres et Repository mønster til vores implementering. Repository mønstret separerer logikken mellem forretningslogikken og datakilden, som en database. En illustration af dette, ses i Figur 1. Repositoryet sørger derfor at tilgå datakilden for data og mapper dataen til forretningslogikken. Ved at logikken separeres, opnås der at alt data tilgang foregår igennem Repositoryet, så der opnås en højere abstraktion. Herved har man opnået højere testbarhed af forretningslogikken.



Figur 1 Repository Pattern (MSDN)

## Anvendelse

Repository mønstret anvendes i både Data Access Layer (DAL) for applikation og webapplikation.

(Ufærdigt)

# Unit of Work

<http://martinfowler.com/eaaCatalog/unitOfWork.html>

Unit Of Work er et design mønster, hvorpå man holder styr på hvilke transaktioner forretningslogikken laver til databasen. I et Unit Of Work, gøres alle de transaktioner som forretningslogikken ønsker at gøre, og derefter vil Unit of Work holde styr på hvad der skal ske på database-siden, og commit og rollback de ændringer, som forretningslogikken har gjort. På den måde opnås der noget logik, kan opretholde en liste af ændringer der skal ske og give det videre til databasen.

## Anvendelse

Unit of Work mønstret anvendes i både Data Access Layer (DAL) for applikation og webapplikation. (ufærdigt)

# Facade Pattern

# ASP.NET MVC