# Valget for repository patternet

Gennem Pristjek220’s udvikling blev der erfaret, at det var besværligt at unit teste BLL da der var for hård binding mellem BLL og DAL. Det vil sige at det ikke var muligt at isolere BLL fra DAL, og da unit tests kræver at klassen, som skal testes, skal isoleres fra resten af koden, var det ikke muligt at teste BLL på denne måde. Et andet problem var, at meget af det kode, som blev skrevet, blev duplikeret, da der var brug for det forskellige steder.

Der blev derfor implementeret et Repository pattern for at separere BLL fra DAL. Derved blev det muligt at isolere BLL, så det kunne unit testes da repositoriet kan substitueres. Repository pattern kommer dog stadig med den ulempe, at det så har en hård binding ned til databasen, og ikke kan isoleres fra den. Derfor er det ikke blevet unit testet, men er i stedet blevet integrationstestet med databasen. Udover at lave en separation mellem BLL og DAL giver repositoriet også den fordel, at den laver et abstraktionslag til databasen, hvor man på den måde kan samle al adgang ned til databasen, og derved undgå at samme kode skal skrives flere steder.



Figur : Implementering af Repository pattern i Pristjek220.

På Figur 1 kan der ses, hvordan Repository pattern’et er blevet implementeret i Pristjek220. De forskellige repositories indeholder CRUD[[1]](#footnote-1) funktionerne, for den tabel de hører til. ProductRepository indeholder derved funktionerne til Product tabellen i databasen. I Repository klassen, som de specifikke repositories nedarver fra, ligger de generelle funktioner som Add og Remove for at undgå duplikeret kode. Unit of Work er lavet som et access point til repositoriesne fra BLL.

1. Create, Read, Update og Delete [↑](#footnote-ref-1)