# Valget for repository patternet

Gennem Pristjek220’s udvikling blev der erfaret at det ikke rigtig kunne lade sig gøre unit teste Buisness logic layer (BLL) da der var en for hård binding imellem BLL og Data access layeret (DAL). Hvilket vil sige at det ikke var muligt at isolere BLL fra DAL, og da unit test kræves at klassen som skal tests skal isoleres fra resten af koden, var det ikke muligt at teste BLL. Et andet problem var at meget af det kode som blev skrevet blev publiceret da der var brug for det forskellige steder.

Der blev derfor implementeret et Repository pattern for at separerer BLL fra DAL. Derved blev det muligt at isolere BLL så det kunne unit tests da repository pattern’et 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, og er derfor ikke blevet unit testet, men i stedet integrations testet med databasen. Udover at lave en separation imellem BLL og DAL giver repositoriet også den fordel at den laver et abstraktions lag til databasen, hvor man på den måde kunne samle alt adgang ned til databasen og derved undgå at samme kode skulle skrives flere steder.



Figur : Implementering af Repository pattern i Pristjek220.

På Figur 1 kan der ses hvordan Repository patterenet er blevet implementeret i projektet. Hvor de forskellige repositories indeholder CRUD[[1]](#footnote-1) funktionerne for de tabeler de hører til altså ProductRepository indeholder funktionerne til Product tabellen i databasen. I Repository klassen som de specifikke repositories nedarver fra er de generelle funktioner som Add og Remove for at undgå duplikeret kode. Unit of Work er lavet som et access point til repositoriesne fra buisness logic layeret.

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