## Arkitektur

Da arkitekturen af projektet skulle besluttes, blev der valgt at bruge en lagdelt model, som ville give et godt og hurtigt overblik over systemet. Grunden til at valget faldt på den, var at ved den lagdelte model er der en gruppering af klasser og pakker. Herved har de klasser, der befinder sig i ét lag, et sammenhængende ansvar for et vigtigt aspekt i systemet i den logiske separation, der er lavet i projektet. Ved at gruppere systemet på denne måde, kan der hurtigt findes ud af, hvor der skal ændres ting, hvis der er noget, der skal ændres. Med valget af at bruge den lagdelte model skulle der så tages en beslutning om, hvor mange lag der skulle inddeles i. Ud fra størrelsen af systemet gav det mest mening at bruge en 3 lagdelt model[[1]](#footnote-1). Den 3 lagdelte model, består af; DAL, BLL og PL.

Et eksempel på hvordan et normalt scenarie kunne se ud, kan ses på Figur 3, som viser hvad der sker, når en bruger indtaster i et felt, der kan autofuldføre. Først ændrer brugeren på teksten på GUI’en, hvorefter GUI’en kalder ned på GUI viewmodel, for at den skal udfylde listen. Dette sker i PL, hvorefter der så kaldes en funktion i klassen AutoComplete i BLL. Denne funktion kalder så en funktion i en klasse i DAL, som laver et database udtræk ud fra den indtastede tekst.



Figur 3: Sekvens diagram for Autofuldførelse, med lag opdeling

Hver af disse lag kan så bestå af flere klasser og pakker, som det kan ses på Figur 4, som viser et package diagram for Pristjek220. Hver pakke er så inddelt under de forskellige lag (PL, BLL og DAL). Hver pakke indeholder så klasser, som har de egenskaber, som pakken beskriver. I programmet ligger de forskellige klasser under pakkens namespace.



Figur 4: Package diagram for Pristjek220

Ved at der er valgt at bruge en lagdelt model og under hvert lag have forskellige klasser, kommer der en separation af applikationsspecifikke tjenester fra de generelle tjenester, som gør, at SRP (Single Responsibility Principle) bliver overholdt. Ved at overholde SRP fås et system, som er mindre sårbart over for ændringer på et senere tidspunkt. Derudover kommer der en separation af højniveaushandlinger fra lavniveaushandlinger. Koblingen og afhængighederne formindskes deraf mellem de forskellige klasser, samt der kommer høj samhørighed. En af de vigtigste ting ved at bruge 3 lags modellen er dog, at der er muligheden for at genbruge kode, sådan at man for eksempelvis kan genbruge de nederste lag af kode til en applikation med en anden brugergrænseflade. Det gøres endnu mere simpelt ved, at de forskellige lag er implementeret med interfaces, som simplificerer en ændring eller udskiftning af dele af systemet. Ved at lave den logiske segmentering mellem de forskellige lag er klarheden af koden øget, for andre der skulle ønske at arbejde videre med koden.

1. Applying\_UML\_and\_Patterns\_\_\_Object-Oriented\_Analysis\_and\_Design\_\_\_3rd\_Edition\_\_Addison\_2004.10\_ Fig 13.2 / SWD Software Architecture I [↑](#footnote-ref-1)