# Konklusion

Projektet er endt ud med en WPF-applikation, en web-applikation og tilhørende databaser. Den ene database kører lokalt, og den anden kører på en Azure-server sammen med web-applikationen. De to databaser synkroniserer automatisk, eller ved at brugeren aktivt beder om det ved et tryk på brugerinterfacet. Der er implementeret flere funktionaliteter på WPF-applikationen end på web-applikationen, da det er her, gruppens største fokus har været. Arbejdet med WPF applikationen startede før introduktionen til *MVP*, *MVC* og *MVVM*. Konsekvensen af dette er manglende afkobling af XAML, *code behind* og *Business Logic Layer* (*BLL*). Dette har gjort det utrolig svært at teste, og derfor er der heller ikke skrevet så mange *unit tests* og integrationstests som ønsket.

Databasetilgangen er i WPF implementeret med ADO.NET, hvilket har givet en del problemer i forhold til *mapping*, og at bevare data-integriteten i selve databasen.

I web-applikationen har vi haft fokus på at få designet og struktureret programmet bedre end vores WPF-applikation. Her har vi brugt *MVC*, og haft fokus på at overholde dette designprincip. Formålet har været at bruge alle de erfaringer, vi har fået under udviklingen af WPF-applikationen, og dermed udvikle en velstruktureret web-applikation.

Databasetilgangen er lavet med *Entity Framework* (*EF*), og mange features og *mapping* sker derfor automatisk. Dette har gjort web-applikationen væsentligt mere læsbar og vedligeholdelsesvenlig end WPF-applikationen.

I projektet er der brugt *continuous integration* i form af et Git-repository. Brugen af Git har gjort det utroligt let at arbejde i det samme *Visual Studio*-projekt på samme tid, og rette de fejl, der kommer, når nye funktionaliteter implementeres. Der har dog været nogle problemer i form af *merging*, når der er blevet ændret i præcis de samme filer, imellem hvert push/pull. Ved disse merges er der nogle få gange gået data tabt, som skulle have været bevaret. På trods af dette opvejer fordelene klart de ulemper, der har måttet være.

Git var ligeledes en nødvendighed, da automatisering af tests i form af *Jenkins* var et ønske, og sammenkoblingen af Git og *Jenkins* var blevet introduceret i I4SWT. Efter at have brugt meget tid på at få det sat op, måtte vi dog konstatere at det ikke var tiden værd, da der uden adgang til selve *Jenkins* serveren ikke kunne laves de nødvendige rettelser for at få *Jenkins* til at kompilere projektet. De første par fejl blev løst over et par uger i samarbejde med Troels Fedder Jensen, men da der blev ved med at dukke nye småting op, gik vi fra brugen af *Jenkins* igen.

Selve gruppearbejdet har været udført med en agil tankegang, og har primært været inspireret af *Scrum*, i form af stand-up møder, retrospektmøder, sprints og backlog. Den agile udviklingsproces har gjort det muligt hele tiden at have et fungerende produkt, som kunne vises til vejlederen, og løbende at tage de vigtigste opgaver. Det har også muliggjort at alle i projektgruppen hele tiden har været klar over hvor langt projektet har været på et givet tidspunkt, og har været i stand til at søge hjælp med det samme, hvis et medlem har været forhindret i at arbejde videre.

Alt dette har resulteret i et enestående slutresultat, med et projekt som opfylder alle de krav, der blev sat i starten af projektet. Det har dog ikke været muligt at implementere mange af de ekstra tilføjelser, grundet tidspres. Dog er der blevet stiftet bekendtskab med nye teknologier, som Azure, hvilket der ikke er blevet undervist i. Ligeledes er der synkronisering mellem to databaser, og en web-applikation, hvilket der ikke har været stort fokus på uden for projektet.