

INGENIØRHØJSKOLEN AARHUS UNIVERSITET

BACHELORPROJEKT

---

RAMBØLL TILSYN



Projektrapport

---

PROJEKT NR. 17103

Navn	Au ID	Studienummer
Ao Li	AU512161	201407737
Morten Sand Knudsen	AU463338	201270955

VEJLEDER: LARS CHRISTIAN JENSEN

DATO: 19/12-2017

ANSLAG: 44438

# Resumé

Rambøll er et landsdækkende rådgivningende ingeniør firma inden for mange forskellige områder, bl.a. byggeri, transport og miljø.

Deres byggetekniske afdeling i Aarhus er interesseret i at digitalisere deres registreringsproces på byggepladsen, som i dag foregår med blyant og papir.

Der er blevet anvendt en række metoder til udviklingen af produktet. Disse metoder har hovedsagligt været workshops, kravspecifikationer og litteratursøgning. Endvidere har der været anvendt en iterativ udviklingsproces i projektet.

Afdelingen har ønsket et produkt, som gør det muligt at lade papir og blyant ligge derhjemme og i stedet benytte enten en smartphone eller tablet på byggepladser mv.

Projektet udmundede i produktet Rambøll Tilsyn, som består af en applikation samt en database. På applikationen vil der kunne oprettes en digitaliseret form for den analog registrering.

Produktet er endt i en prototype, som vil kunne implementeres til et endeligt produkt.

# Abstract

Rambøll is a nation-wide consulting engineering group in a variety of areas such as construction, transport and environment.

Their construction department in Aarhus holds an interest in digitalising the registration process on the building sites which today is carried out with paper and pencil.

A broad array of methods have been utilized in the development of the product at hand. These include workshops, requirement specifications and literature searches. An iteration development process was used for this project.

The department sought a product which would enable them to leave paper and pencils behind and replace them on-site with either a smartphone or a tablet. The product developed is Rambøll Tilsyn. Rambøll Tilsyn is a system which consist of an iOS application to smartphones or tablets with a Firebase backend. In the application a digitalised version of the analogue registration which is currently in place can be created.

The product is currently a prototype which may further be implemented into a final product.

# Indholdsfortegnelse

	Side
<b>Kapitel 1 Indledning</b>	<b>7</b>
<b>Kapitel 2 Kravspecifikation</b>	<b>8</b>
2.1 Aktør-kontekst diagram . . . . .	8
2.2 User stories diagram . . . . .	9
2.3 Funktionelle krav . . . . .	10
2.4 Ikke-funktionelle krav . . . . .	10
2.5 Afgrænsning . . . . .	11
<b>Kapitel 3 Metode</b>	<b>12</b>
3.1 ASE modellen . . . . .	12
3.2 UML, Design og Test . . . . .	13
3.2.1 UML . . . . .	13
3.2.2 Design . . . . .	13
3.2.3 Test . . . . .	13
3.3 Projektstyring . . . . .	13
<b>Kapitel 4 Analyse</b>	<b>14</b>
4.1 Firebase . . . . .	14
4.2 Xamarin . . . . .	14
<b>Kapitel 5 Arkitektur</b>	<b>15</b>
<b>Kapitel 6 Design &amp; Implementering</b>	<b>17</b>
6.1 Firebase . . . . .	17
6.2 Applikation . . . . .	19
6.2.1 Login . . . . .	20
6.2.2 Projekt oversigt . . . . .	22
6.2.3 Registrering på PDF . . . . .	24
6.2.4 Opret Bruger . . . . .	28
<b>Kapitel 7 Test</b>	<b>30</b>
7.1 Automatiseret test . . . . .	30
7.2 Manuel test . . . . .	31
<b>Kapitel 8 Resultater</b>	<b>32</b>
<b>Kapitel 9 Erfaringer</b>	<b>34</b>
<b>Kapitel 10 Fremtidigt arbejde</b>	<b>35</b>

<b>Kapitel 11 Konklusion</b>	<b>36</b>
<b>Ordforklaring</b>	<b>36</b>
<b>Litteratur</b>	<b>38</b>

# Læsevejledning

Denne rapport redegør for den proces som projektet har fulgt og skal give læseren et indblik i projektdokumentationen med vægt på projektets særegenheder. Rapporten følger som udgangspunkt strukturen, der er anbefalet i det vejledende dokument udleveret af IHA / ASE. Da rapporten er en del af den samlede dokumentation for projektet, vil der forekomme henvisninger til tekst og figurer i projekt dokumentationen.

Rapporten kan overordnet inddeles i 5 sektioner:

I første sektion gives der i Kapitel 1 en indledende beskrivelse af det valgte projekt og dets afgrænsninger. I Kapitel 2 bliver kravene til systemet beskrevet.

I anden sektion beskrives der kort i Kapitel 3 de arbejdsgange og metoder, som er blevet brugt under projektet. For en mere uddybende procesbeskrivelse, henvises der til procesrapporten. I Kapitel 4 beskrives kort om den analyse, der er lavet til projektet.

I tredje sektion ses arkitekturen til Rambøll Tilsyn samt, hvordan det er blevet designet, implementeret og testet. Sektionen dækker over Kapitel 5, 6 og 7.

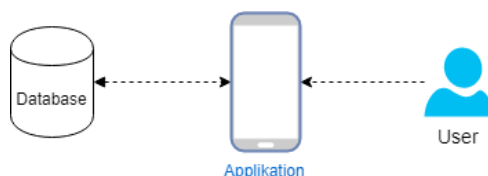
I fjerde sektion af rapporten findes resultater og diskussionen samt, hvilke overvejelser der er gjort for at projektet kan videreudvikles. Fjerde sektion indeholder ligeledes generelle erfaringer. Dette sker i Kapitel 8, 9 og 10.

Femte sektion indbefatter Kapitel 11, hvor der konkluderes på projektet som helhed og der drages paralleller til indledningen.

I slutningen af rapporten findes ordforklaring og litteraturliste.

# 1 Indledning

I løbet af de seneste år er der sket en stor udvikling inden for digitalisering af samfundet og dets arbejdsprocessor[1]. Denne digitale udvikling ønsker de også hos Rambøll. De vil gerne væk fra at skulle have papirstegninger med ud på byggepladsen for at tage notater til et byggeprojekt. I dag er der flere ansatte ved Rambøll som har papir og blyant med ud på byggepladsen for at registre deres observationer. Derfor ønsker Rambøll at få udviklet en applikation, som kan gøre denne arbejdsproces nemmere. Der er blevet udviklet en prototype i form af en applikation med navn Rambøll Tilsyn til dette formål. Rambøll Tilsyn består af en applikation med en tilhørende database, som det kan ses på oversigten på Figur 1.1



**Figur 1.1.** Oversigt over systemet.

Med dette system vil de ansatte hos Rambøll kunne lave deres registreringer direkte i applikationen og derved bliver de fri for at bruge papir og blyant.

## Ansvarsområder

Tabel 1.1 viser en overordnet fordeling af ansvarsområderne i udvikling af produktet Rambøll Tilsyn. Kravspecifikationen er blevet udarbejdet af gruppen i samarbejde med Rambøll.

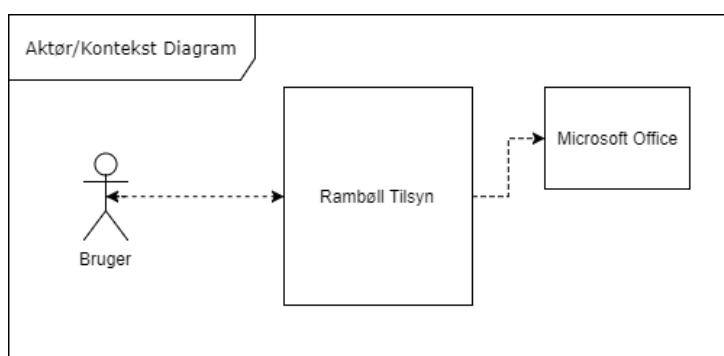
Ansvar	Ao	Morten
Back-end	X	
Applikation	X	X
Dokumentation		X

**Tabel 1.1.** Ansvarsområder for udvikling.

# 2 Kravspecifikation

## 2.1 Aktør-kontekst diagram

På Figur 2.1 ses aktør kontekst diagrammet for Rambøll Tilsyn. Diagrammet viser, hvilke aktører der interagerer med systemet.



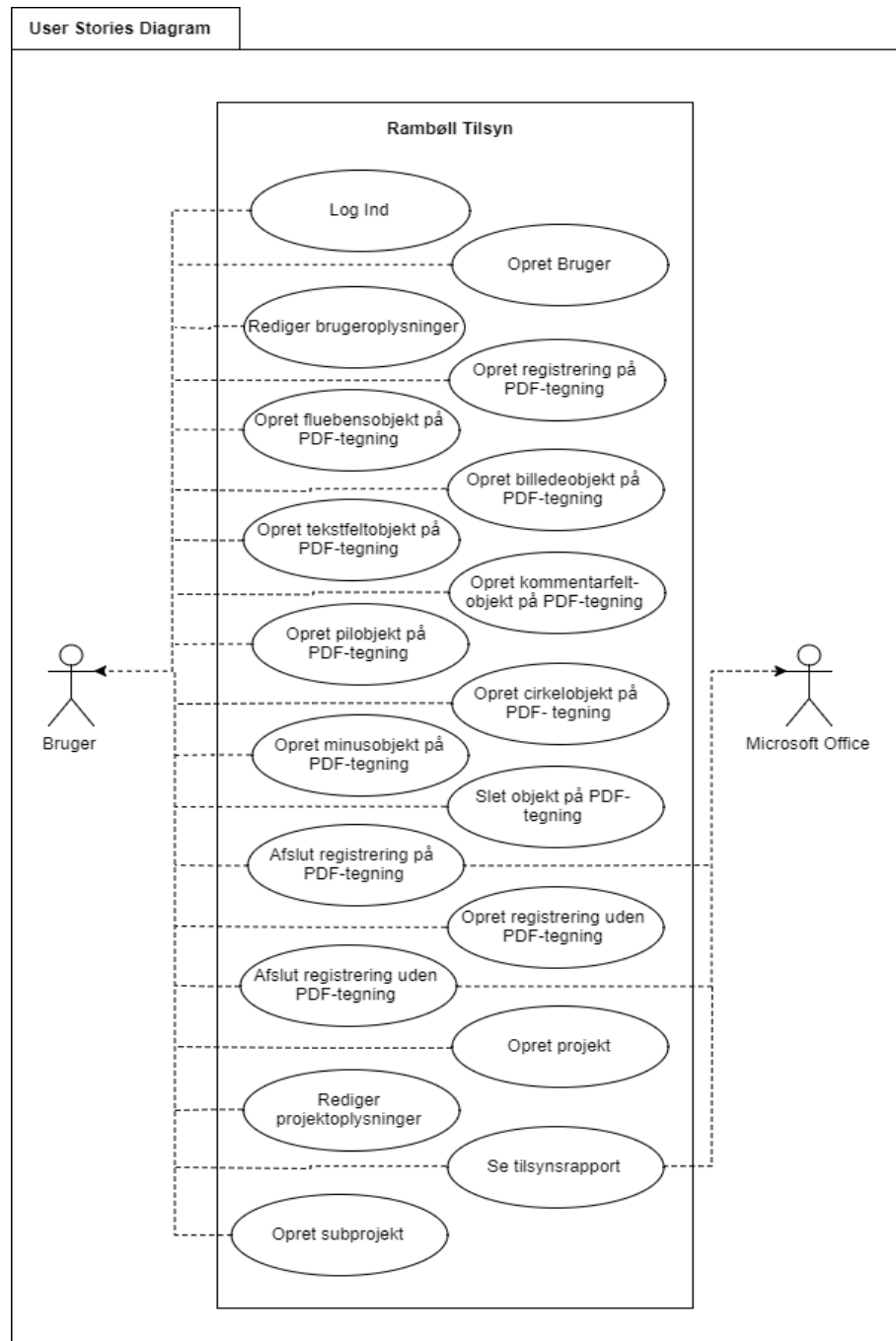
**Figur 2.1.** Aktør-kontekst diagram for Rambøll Tilsyn.

På venstre side af Figur 2.1 ses brugeren, som er den primære bruger af systemet. Brugeren benytter systemet, som på diagrammet er symboliseret som en blackbox. På højre side ses de sekundære aktører. Disse aktører er dem, som systemet benytter sig af. Microsoft Office er en sekundær aktør, da systemet eksporterer til Excel.



## 2.2 User stories diagram

Figur 2.2 viser systemets User Stories diagram. Dette diagram viser den funktionalitet, som systemet indeholder beskrevet gennem en række User Stories. Udvalgt User Stories beskrives i Afsnit 2.3.



**Figur 2.2.** User Stories diagram for Rambøll Tilsyn.

## 2.3 Funktionelle krav

Følgende er en kort beskrivelse af udvalgte User Stories til Rambøll Tilsyn, som er fundet sammen med Rambøll ved hjælp af MoSCoW analyse [2]. Alle User Stories er fuldt beskrevet med Gherkin i Kravspecifikationens afsnit 3.2 Funktionelle Krav.

### Log-in (CRS-1)

Denne User Story håndterer log-in på applikationen. Her skal brugeren indtaste korrekt brugernavn og kodeord for at få adgang.

### Opret bruger (CRS-2)

Opret bruger User Storien giver en bruger mulighed for at oprette en ny bruger i systemet. Når brugeren er blevet oprettet, vil denne kunne logge ind på applikationen og benytte systemet.

### Opret en registrering på PDF-tegning (CRS-4)

Denne User Story håndterer det, at brugeren ønsker at oprette en registrering på en PDF-tegning til et byggeprojekt.

### Opret fluebensobjekt på PDF-tegning (CRS-5)

I denne User Story får brugeren mulighed for at sætte et flueben på en PDF-tegningen i sin registrering. Fluebenet indikere, at brugeren har godkendt denne del af byggeriet.

### Opret billedeobjekt på PDF-tegning (CRS-6)

Denne User Story giver brugeren mulighed for at tage et billede og placere det på PDF tegningen. Når billedet er taget og placeret på PDF'en, kan brugeren skrive en kort beskrivende tekst til billedet.

### Slet objekt på PDF-tegning (CRS-12)

I denne User Story får brugeren mulighed for at slette et objekt. Hvis brugeren placerer et objekt forkert, eller kommer til at vælge det forkerte objekt, kan brugeren slette det.

### Afslut registrering på PDF-tegning (CRS-13)

Denne User Story håndterer afslutningen af brugerens registrering. Når brugeren har oprettet alle objekter på PDF'en, kan der afsluttes registrering ved tryk på 'Afslut' knappen. En excel-fil vil blive genereret med informationer om de forskellige objekter.

### Opret projekt (CRS-16)

Opret projekt User Storien giver brugeren mulighed for at oprette nye projekter i systemet. Når projektet er oprettet, vil brugere kunne tilgå dette projekt og oprette registreringer.

## 2.4 Ikke-funktionelle krav

De ikke-funktionelle krav definerer de tekniske krav, som systemet skal indeholde. Disse krav beskriver egenskaber, som ikke har indvirkning på systemets funktionelle krav. Dette er eksempelvis antal brugere, der benytter systemet samtidig og hvordan brugerne skal have skrive og læse rettigheder.

Projektets ikke-funktionelle krav kan findes i Kravspecifikationens Afsnit 3.3, som ligger i bilag.

## 2.5 Afgrænsning

For at afgrænse projektet blev der holdt et møde med Rambøll, hvor de kunne komme med kommentar og ideer til systemet.

Der blev lavet en MoSCoW analyse til at prioritere funktionaliteten i systemet. Denne analyse form deler alt funktionaliteten op i fire kategorier *must*, *should*, *could* og *would*. Det blev valgt at systemets *must* User Stories definerer systemet og derved at disse funktionaliteter som er blevet implementeret i projektet, den detaljerede MoSCoW analyse kan findes i kravspecifikations Afsnit 5.1 User stories, prioriteret efter MoSCoW.

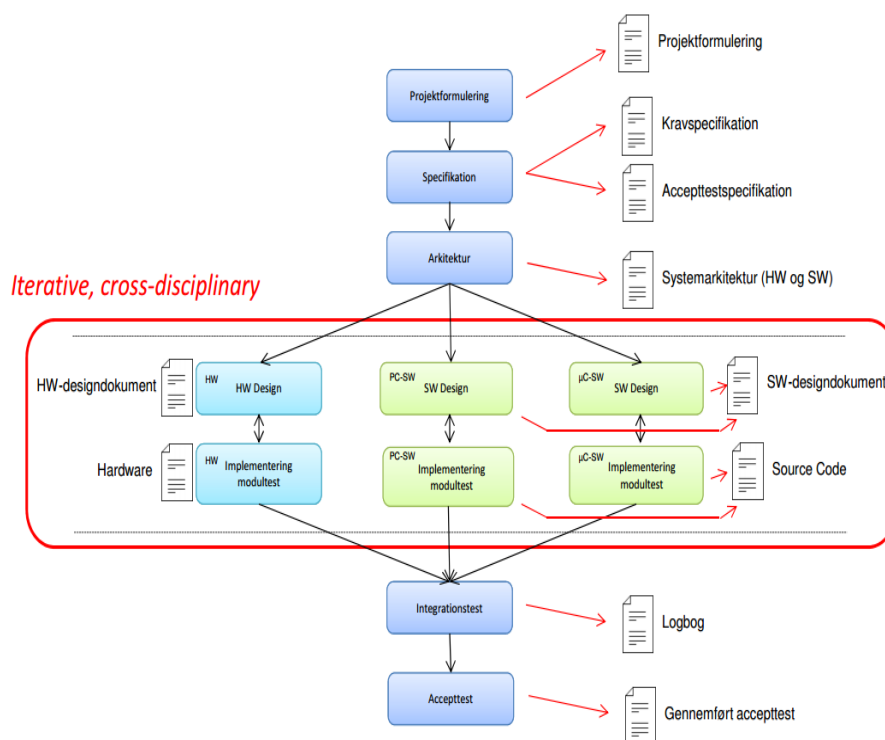
Rambøll ønskede en applikation udviklet med fokus på iOS, da største delen af afdelingen har iPads tilknyttet. Der var dog en længere snak omkring dette, da også et par enkelte brugte Android. Derfor blev det aftalt, at der ville forsøges med at udvikle i cross-platform, som ville kunne fungere til både iOS og Android. For den fulde afgrænsning samt MoSCoW analyse, henvises til Kravspecifikationens Afsnit 5 Afgrænsning.

# 3 Metode

I dette kapitel beskrives om, hvilke metoder og processer der er anvendt igennem projektet. For dybdegående beskrivelse af processen, henvises til Procesrapporten som findes i bilag.

## 3.1 ASE modellen

Den udviklingsmodel, som primært er brugt i projektet, er ASE modellen, som ses på figur 3.1. ASE modellen[3] er en mellemvægtig semi-iterativ udviklingsmodel, som er drevet ud fra en kravspecifikation, der bygger på User Stories. ASE modellen tager udgangspunkt i vandfaldsmodellen til at opbygge et projekt gennem faserne: Projektformulering - Kravspecifikation - Systemarkitektur - Implementering - Test/fejlfinding - Integration og vedligeholdelse [3].



**Figur 3.1.** ASE modellen [3].

For en mere dybdegående beskrivelse af ASE modellen, se Procesrapportens Afsnit 2.3.1 ASE-modellen.

## 3.2 UML, Design og Test

Der er brugt forskellige teknikker og værktøjer inden for design og test til at gennemføre projekt. Disse beskrives i afsnittene herunder.

### 3.2.1 UML

Software designet er taget med udgangspunkt i en domænemodel og UML[4] klasse diagrammer til at vise og specificere grænsefladerne mellem modulerne i softwaren. Ved hjælp af domæne modellen udvikles softwareklasser og hvordan de interagerer med hinanden. Domænemodellen giver et godt overblik over, hvordan de forskellige klasser skal arbejde i systemet. Derudover viser den grænseflader, som fortæller, hvordan de forskellige klasser skal kommunikerer med hinanden. Der er også udarbejdet sekvensdiagrammer, disse går under begrebet adfærdsdiagrammer som viser, hvordan systemet reagerer i forskellige situationer, når brugeren interagerer med systemet.

### 3.2.2 Design

Software design er et værktøj, der bruges blandt udviklere til at skabe software på den bedst mulige måde. Der er mange overvejelser, der skal gøres ved valget af design. Dette blev ofte diskuteret i grupper for at alle er enige om, hvordan udviklingen skal gribes an og der bruges samme design fra start.

### 3.2.3 Test

Softwaretest bruges til at sikre en god kvalitet i koden, der bliver udviklet. Ved at skrive tests til koden, opdages fejlene i kodens opførsel. Dette gør det muligt at finde fejl før udgivelse.

## 3.3 Projektstyring

Styringen af dette projekt er sket gennem dagelige møder i gruppen og et ugentligt vejledermøde.

Der er gennem projektet blevet arbejdet iterativt med udvikling af både dokumentation, rapport og prototypen. Da gruppen har siddet samlet under arbejdsprocessen er ændringer og nye tiltag blevet aftalt løbende. Rollerne i gruppen blev før projektes start fastlagt.

Da den overordnede funktionalitet af systemet skulle bestemmes, sad gruppen og arbejdede i fællesskab for at nå frem til et produkt, der var tilfredsstillende. Gruppen har derfor været inde over kravene til systemet og den grundlæggende opbygning af arkitekturen. Efter alle overordnede krav var blevet fastlagt til Rambøll Tilsyn og der var tilfredshed med systemet, blev gruppen delt op for at arbejde med design og implementering. Der har været tre faste dage, hvor der blev arbejdet med projektet. Her blev ugens problemstillinger lagt op, og der blev holdt daglig stand up møde med opdateringer på, hvor langt folk var, og hvordan disse problemstillinger kunne løses.

Der er blevet brugt Git til at styre versionshistorikken på alle dokumenter og software.

Da gruppen altid sad samlet, var der god kommunikation, om hvor langt de forskellige opgaver var. Dette gav en god iterativ arbejdsproces, da ændringer, blev taget i fællesskab og derved vidste gruppen hele tiden, hvad der skulle ske.

Der er gennem projektet brugt forskellige værktøjer. For en oversigt over disse henvises til Kravspecifikationens Kapitel 6

# 4 Analyse

I projektets start blev der udarbejdet en analyse af systemet. I dette kapitel vil der blive beskrevet de valg, som blev taget til projektet. En mere dybdegående Analyse kan findes i bilag.

## 4.1 Firebase

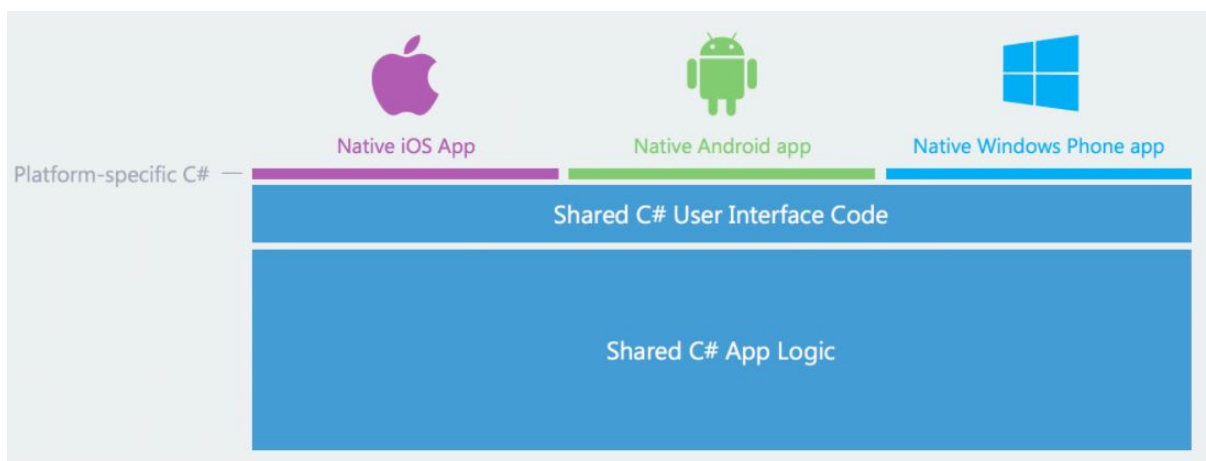
Til håndtering af back-end i systemet er der valgt at bruge Firebase. Firebase[5] er en development platform, som Google har udviklet. Firebase tilbyder flere forskellige funktionaliteter til både iOS og Android.

Firebase er serverless, hvilket betyder at der ikke er behov for at opsætte en server til databasen for at kunne interagere med databasen. Da Rambøll ikke har en server til rådighed, virkede dette, som den optimale løsning.

## 4.2 Xamarin

Applikationen er blevet udviklet i Xamarin Forms[6]. Dette er et cross platform udviklingsværktøj, som giver mulighed for at udnytte share code. Dette gør, at man kan bruge samme kode til både iOS og Android platformen.

Valget bag dette er et ønske fra Rambøll, som gerne ville have et system, der både virkede på iOS og Android, da deres ansatte brugte begge styresystemer.



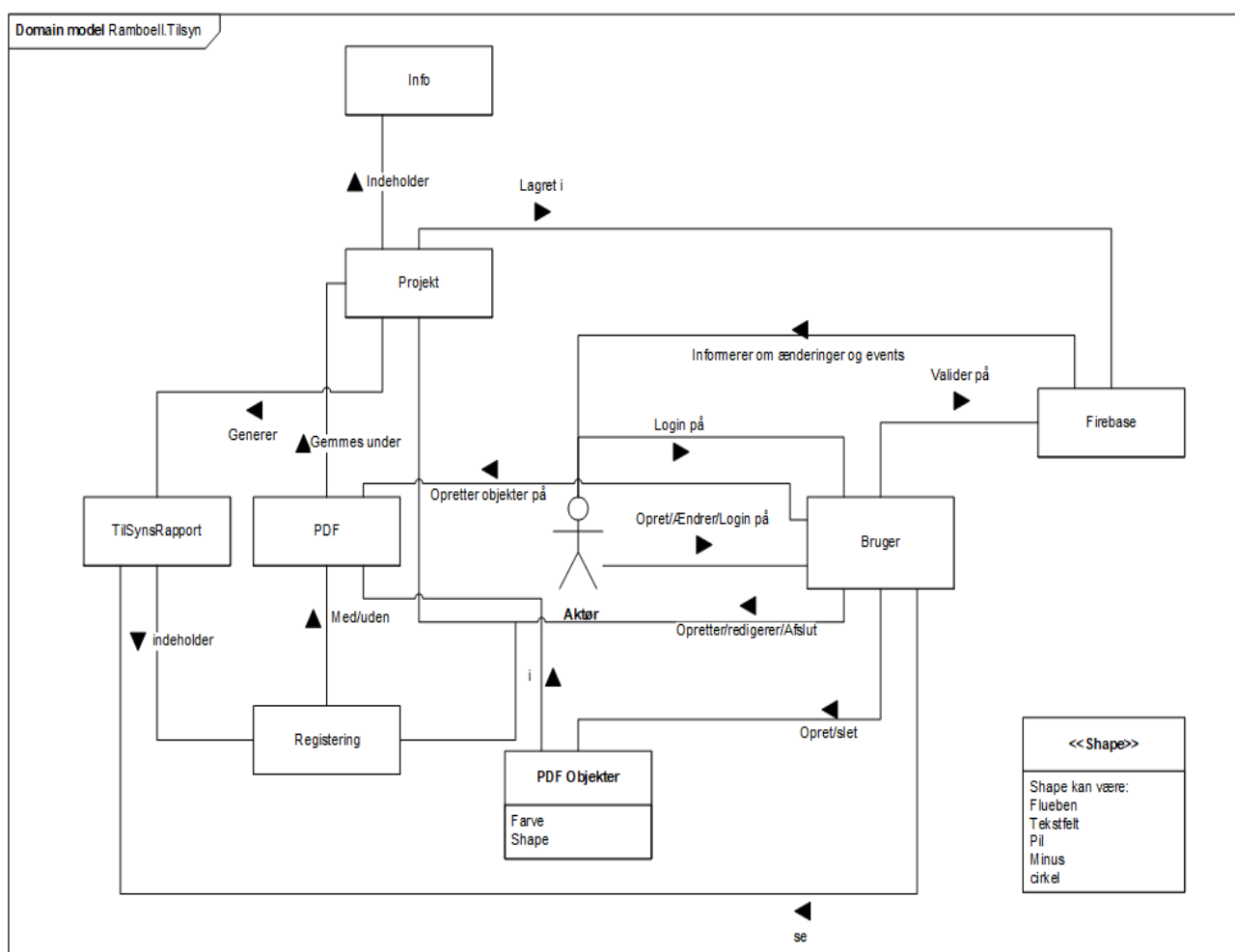
*Figur 4.1.* Kode deling i Xamarin Forms [6].

Fordelen ved Xamarin i forhold til andre cross platform værktøjer er, at det er en indbygget del af Visual Studio, som var programmet, der blev brugt til udvikling af applikationen [7].

# Arkitektur

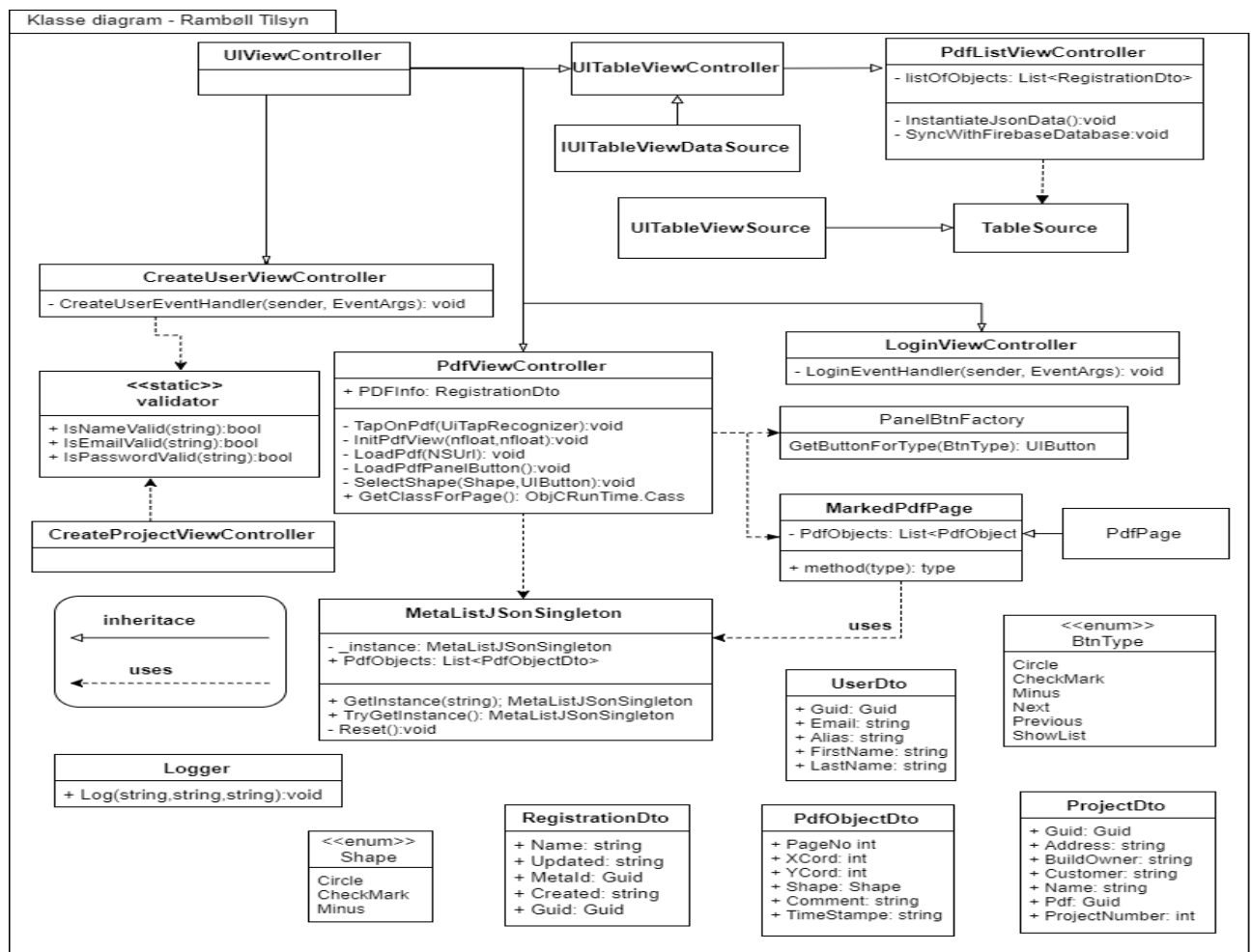
I dette afsnit vil der blive beskrevet den overordnede arkitektur for Rambøll Tilsyn.

På Figur 5.1 ses domæne modellen for Rambøll Tilsyn. Her ses, hvilke dele systemet indeholder og hvilken relation de har til hinanden. For den fulde arkitektur henvises til Arkitektur og Design dokumentationens Afsnit 1 Arkitektur.



**Figur 5.1.** Domænemodell for Rambøll Tilsyn.

På Figur 5.2 ses der en klassediagram over Rambøll TilSyn. Klassediagrammet viser klasserne i Rambøll Tilsyn og hvilke relationer disse har.



**Figur 5.2.** Klassediagram for Rambøll Tilsyn.

Alle ViewControllers er nedarvet fra UIViewController[8]. UIViewController er en klasse, der stammer fra Xamarin.iOS frameworket.

Klassen UITableViewControllers formål er at vise en liste af data, som man selv har implementeret.

Logger klassen bruger Firebase Analytics[9] for at logge events fra applikationen. Dette gør, at man kan bruge fejlmeddelelserne fra Analytics til at fortælle brugeren, hvad der er sket af fejl.



# 6

## Design & Implementering

I dette afsnit vil der blive beskrevet det overordnede design for Rambøll Tilsyn. Derefter vil design for de forskellige dele blive beskrevet med en design-, implementering- og test del.

### 6.1 Firebase

I dette afsnit vil designet for Firebase blive gennemgået. For den fulde dokumentation af design og implementering henvises til Arkitektur og Design dokumentationens Afsnit 2.1 Firebase database.

For håndtering af brugere benyttes Firebase Auth [10]. Opbygningen af denne kan ses på Figur 6.1.

A screenshot of the Firebase Console showing a table of authentication data. The table has five columns: Identifier, Providers, Created, Signed In, and User UID. There are five rows of data, each representing a user. The first row has the email 'pass@123456.dk', a provider icon, the date 'Nov 20, 2017', the date 'Dec 10, 2017', and a long alphanumeric User UID. The other rows follow a similar pattern with different email addresses and dates.

Identifier ↓	Providers	Created	Signed In	User UID ↑
pass@123456.dk	✉	Nov 20, 2017	Dec 10, 2017	8pmorrqTyJXBstWEqAyyUJkbNk1
afds@fdsf.gs	✉	Dec 11, 2017	Dec 11, 2017	JHugBIE9TibUWXSiKjVEdmHJnS2
lif@ds.fse	✉	Dec 11, 2017	Dec 11, 2017	XTB1EZo0P2ax3deUTGMDZm8kZ...
tester@asd.df	✉	Dec 11, 2017	Dec 11, 2017	hSmJl4yMWbfjFBy4riOeD6vngT2
test@test.dk	✉	Dec 10, 2017		jmt4GMGiqzSNWGiWa3hhJ515LZ...

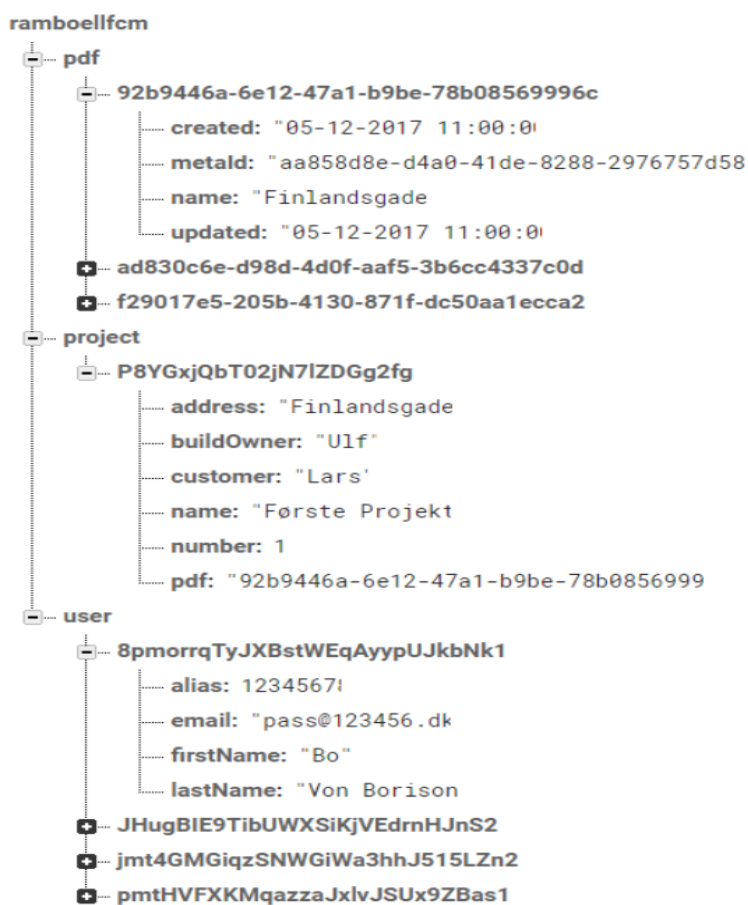
**Figur 6.1.** Oversigt over authentication data i Firebase Console.

Denne indeholder en tabel med en identifier, som er brugerens e-mail.

Provider fortæller, hvordan brugeren har oprettet sig og som hænger sammen med created feltet, der fortæller, hvornår brugeren er oprettet.

Signed In fortæller, hvornår brugeren sidst er logget ind og det sidste felt er brugerens unikke id. Her er også gemt et hashed password, men dette bliver ikke vist pga sikkerhed.

Figur 6.2 viser database strukturen i Firebase.



**Figur 6.2.** Oversigt strukturen i Firebase databasen

Databasen består af tre noder: **pdf**, **projekt** og **user**.

**Pdf** noden indeholder alle PDF-tegninger i systemet samt en reference til informationen omkring oprettede objekter på disse PDF filer.

Hver ny PDF, som bliver tilføjet i systemet, vil blive oprettet som en undernode i **pdf** noden.

**Projekt** noden indeholder noder for hvert projekt, som er oprettet. Disse undernoder indeholder alt det information som tilhører projekterne.

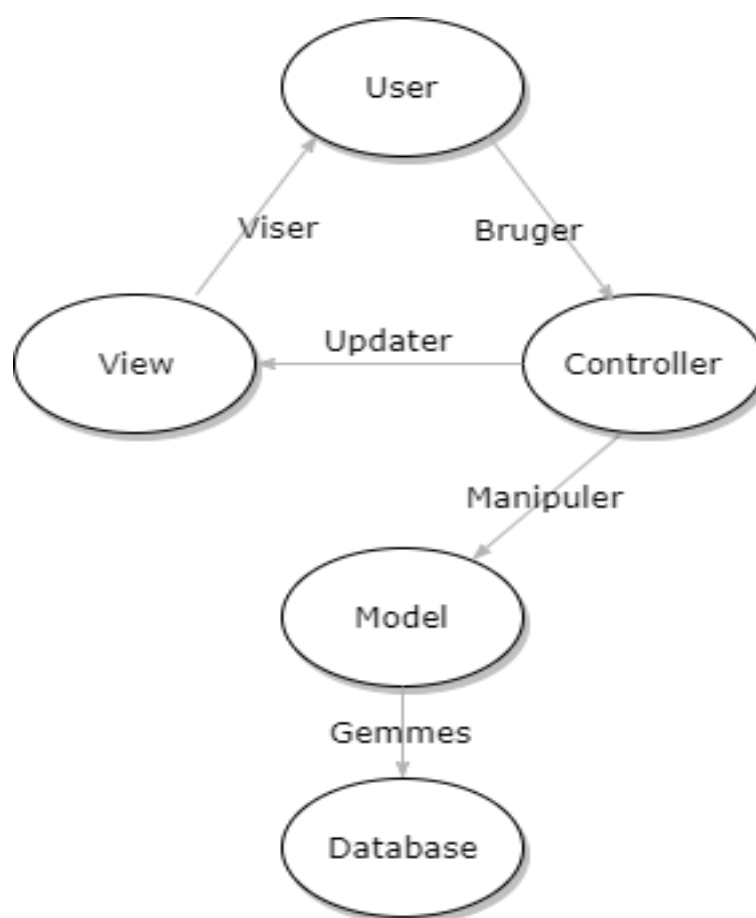
**User** noden indeholder information for brugere. Hver bruger oprettes som en ny undernode.

## 6.2 Applikation

I følgende afsnit bliver der gennemgået design, brugergrænseflade og implementering for User Stories for 'Login', 'Project List', 'Registrering på PDF' og 'Opret Bruger'. For den fulde dokumentation af design og implementering henvises til Arkitektur og Design dokumentationens Afsnit 2.2 og de tilhørende afsnit.

I analysen blev der taget udgangspunkt i at applikationen skulle udvikles i Xamarin Forms [11]. Der blev under udviklingen fundet begrænsninger i Forms framework. Derfor er applikationen implementeret i Xamarin iOS.

Til udvikling af Rambøll Tilsyn er der brugt et MVC design [12], som vist herunder, på Figur 6.3. For beskrivelsen af MVC patternet henvises til Arkitektur og Design dokumentationens afsnit 2.2 Applikation.



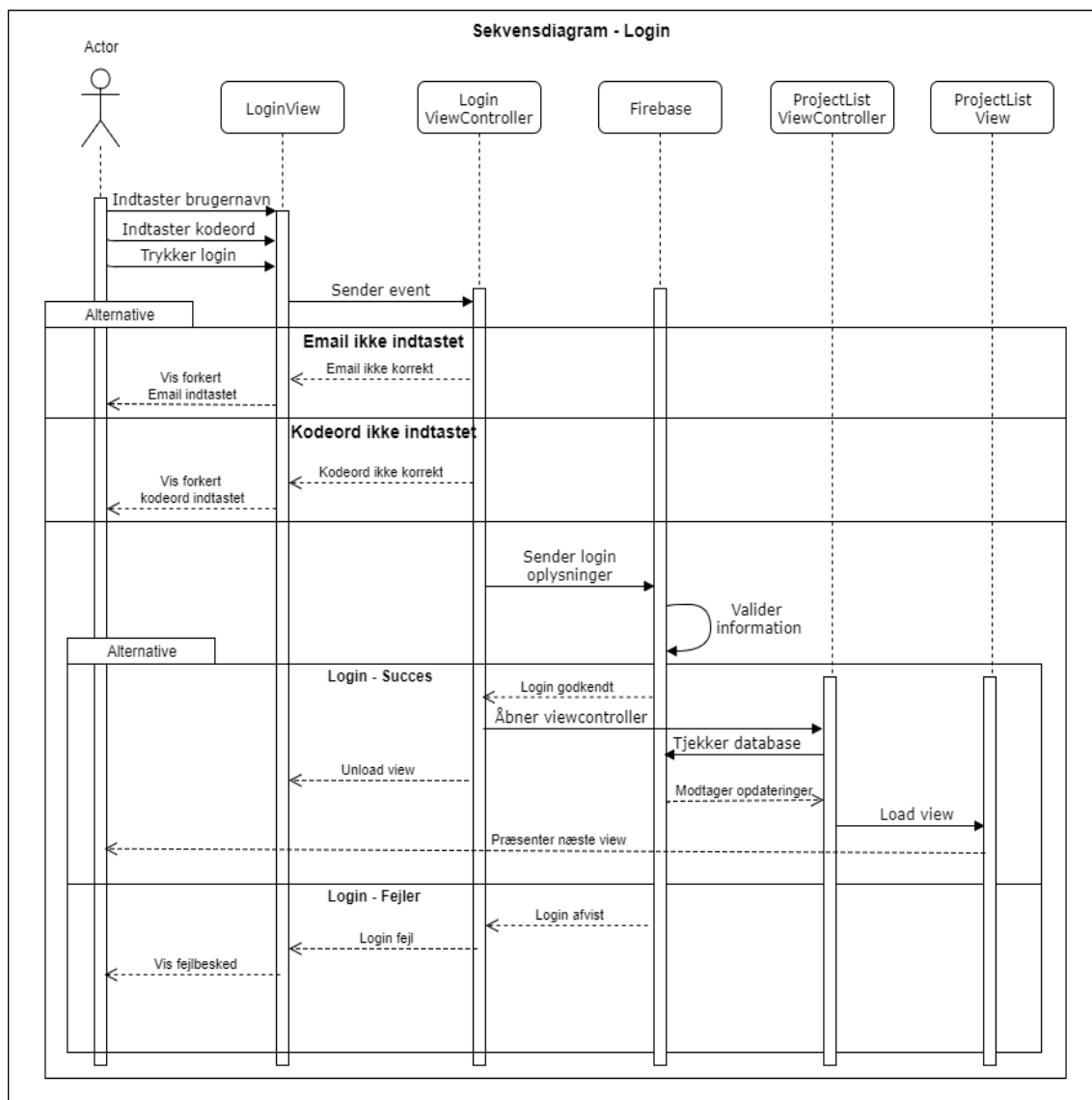
**Figur 6.3.** MVC design for Rambøll Tilsyn.

### 6.2.1 Login

I dette afsnit vises design, brugergrænseflade, implementering og test for 'Login' viewet. For den fulde dokumentation henvises til Arkitektur og Design dokumentationens Afsnit 2.2.2 Login.

#### 6.2.1.1 Design

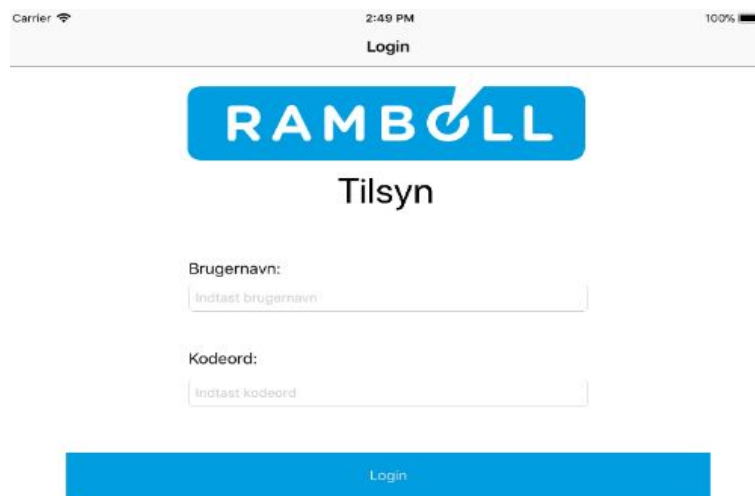
Sekvensdiagrammet for 'Login' viewet til Rambøll Tilsyn kan ses på Figur 6.4. Figuren viser det logiske flow der sker, når brugeren vil logge ind.



**Figur 6.4.** Sekvensdiagram for 'Login' i Rambøll Tilsyn.

### 6.2.1.2 Grafisk brugergrænseflade

Den grafiske brugergrænseflade for 'Login' viewet består af felter til, at brugeren kan indtaste sit brugernavn og kodeord. Se Figur 6.5

The image is a screenshot of a mobile application's login screen. At the top, there is a status bar with 'Carrier', signal strength, '2:49 PM', and '100%' battery. Below this is a header with the word 'Login'. The main content area features the 'RAMBOLL' logo in a blue rounded rectangle, followed by the word 'Tilsyn' in a large, bold, black font. Below 'Tilsyn' are two input fields. The first is labeled 'Brugernavn:' and has a placeholder text 'Indtast brugernavn'. The second is labeled 'Kodeord:' and has a placeholder text 'Indtast kodeord'. At the bottom of the form is a large blue button with the text 'Login' in white.

**Figur 6.5.** 'Login' viewet som det er implementeret i Rambøll Tilsyn.

### 6.2.1.3 Implementering

Når applikationen åbner og login siden kommer frem, starter Firebase med at kontrollere om der allerede er en bruger, som er logget ind. Er der en bruger logget ind, navigeres man direkte videre til 'Project List' viewet. Hvis ikke der er en bruger logget ind skal brugeren logge ind, for at kunne benytte applikationen.

Når login bliver loadet, sættes der en eventhandler på login knappen, så når bruger trykker på den, vil der blive forsøgt at logge ind på applikationen.

Der bliver også indsat placeholders i tekstfelterne for e-mail og kodeord. Når brugeren begynder at skrive kodeordet, vil dette være masked, så kodeordet ikke står i klar tekst.

Når brugeren trykker på login, sendes værdierne fra tekstfelterne til Firebase, som her validerer oplysningerne. Er oplysningerne korrekte, vil brugeren blive navigeret til 'Project List' viewet. Hvis de er forkerte, vil der blive vist en fejlmeddelelse til brugeren.

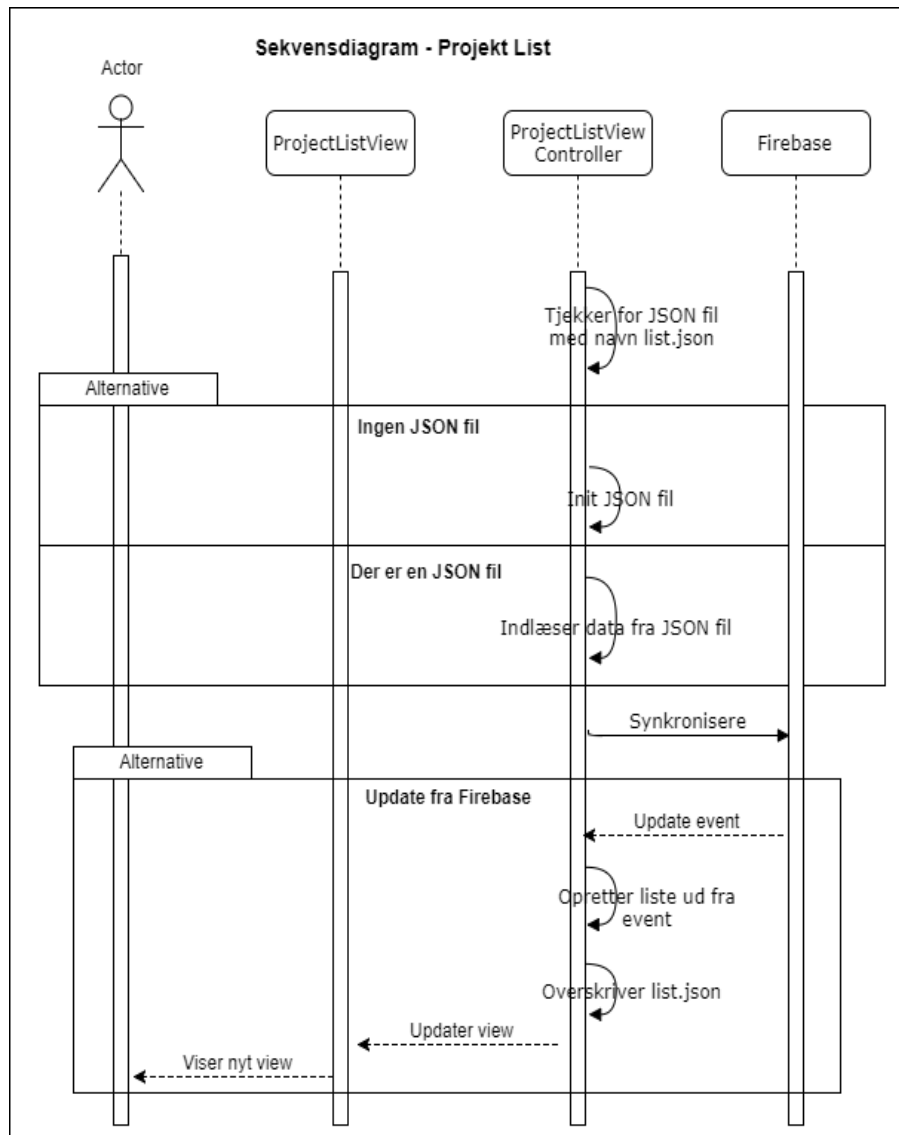
For en mere detaljeret beskrivelse af implementeringen og kode snips, henvises til Arkitektur og Design dokumentations Afsnit 2.2.2 Login.

### 6.2.2 Projekt oversigt

I dette afsnit vises design, brugergrænseflade, implementering og test for 'Project List' viewet. For den fulde dokumentation henvises til Arkitektur og Design dokumentationens Afsnit 2.2.3 Projekt Oversigt.

#### 6.2.2.1 Design

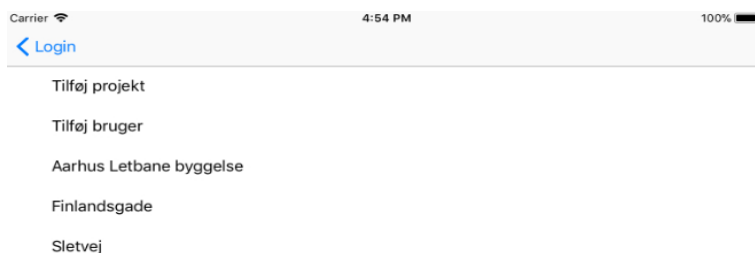
På Figur 6.6 ses sekvensdiagrammet for 'Project List' viewet til Rambøll Tilsyn.



**Figur 6.6.** Sekvensdiagram for 'Project List' i Rambøll Tilsyn.

### 6.2.2.2 Grafisk brugergrænseflade

I 'Project List' viewet er der en oversigt over, hvilke projekter der ligger i databasen, samt mulighed for tilføje projekt eller bruger. Se Figur 6.7



**Figur 6.7.** 'Project List' viewet, som det er implementeret i Rambøll Tilsyn.

### 6.2.2.3 Implementering

Før listen af projekter bliver vist for brugeren, initialiserer 'Project List' viewet en forbindelse til Firebase, hvor den sætter et synkroniseringsevent, som kaldes såfremt, der sker ændringer i PDF-filerne.

Dernæst opretter den en JSON-fil, som indeholder alt projekthinformation fra Firebase.

Controlleren opretter nu et nyt TableView som har sourcen TableSource. Her laver den en liste bestående af alle projekter og muligheden for at tilføje projekt og bruger.

For en mere detaljeret beskrivelse af implementeringen og kode snips, henvises til Arkitektur og Design dokumentations Afsnit 2.2.3 Projekt Oversigt.

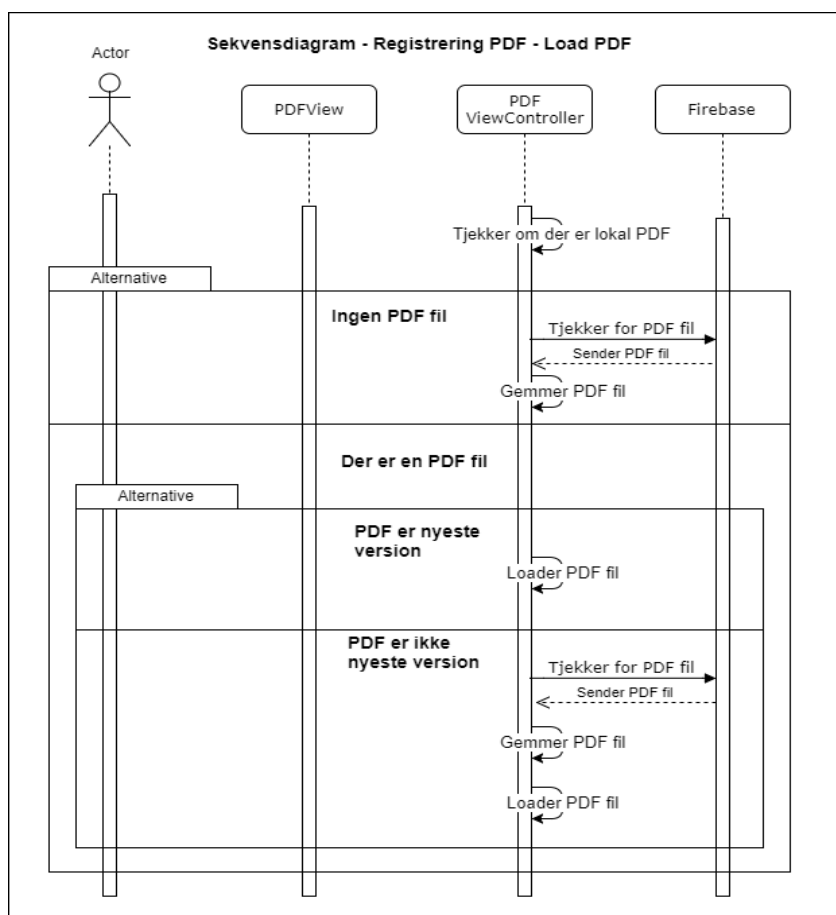
### 6.2.3 Registrering på PDF

I dette afsnit vises design, brugergrænseflade, implementering og test for 'Registrering på PDF' viewet. For den fulde dokumentation henvises til Arkitektur og Design dokumentationens Afsnit 2.2.6 Registrering på PDF.

#### 6.2.3.1 Design

Sekvensdiagrammerne for registrering på PDF-tegning, er opdelt i tre, for at overskueliggøre flowet.

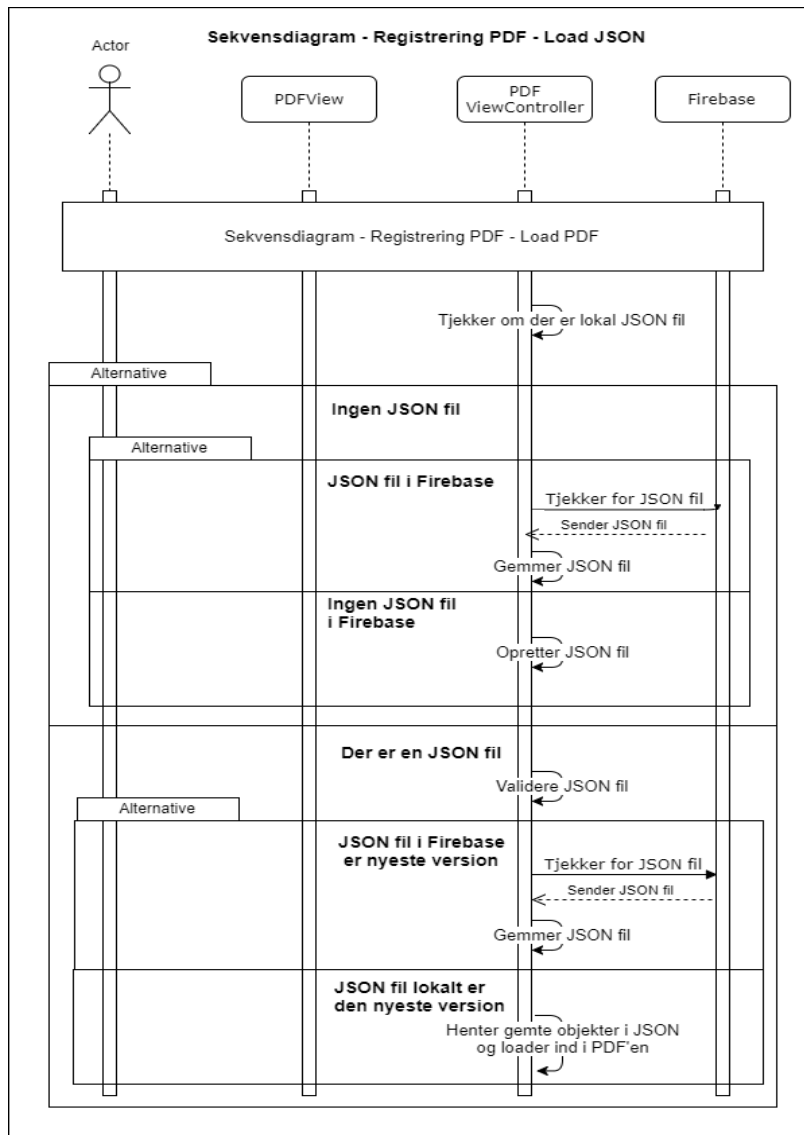
Det første sekvensdiagram viser, hvordan applikationen loader PDF-tegningen ind. Sekvensdiagrammet kan ses på Figur 6.8.



**Figur 6.8.** Sekvensdiagram for Registrering på PDF - Loading af PDF, i Rambøll Tilsyn.

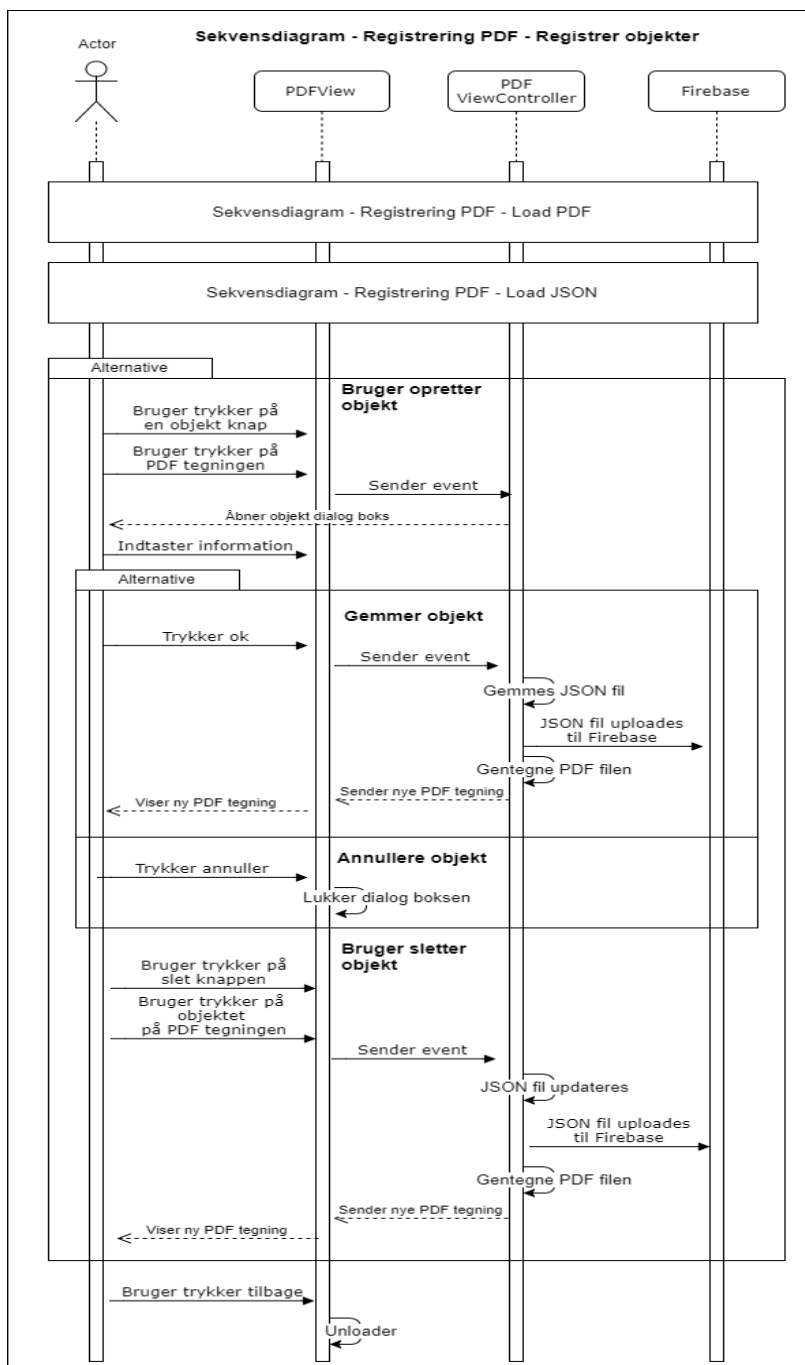


Andet sekvensdiagram viser, hvordan JSON-filen bliver oprettet. Sekvensen med JSON sker direkte efter, at PDF sekvensen er overstået. Sekvensdiagrammet kan ses på Figur 6.9.



**Figur 6.9.** Sekvensdiagram for Registrering på PDF - Loading af JSON, i Rambøll Tilsyn.

Sidste sekvensdiagram for 'Registrering på PDF', sker i forlængelse, af først Loading af PDF og Load JSON og viser, hvordan systemet agerer, når brugeren interagerer med applikationen i forbindelse med oprettelse af objekter på PDF-tegningen. Sekvensdiagrammet for dette, kan ses på Figur 6.10.



**Figur 6.10.** Sekvensdiagram for Registrering på PDF - Registrer på PDF, i Rambøll Tilsyn.

### 6.2.3.2 Grafisk brugergrænseflade

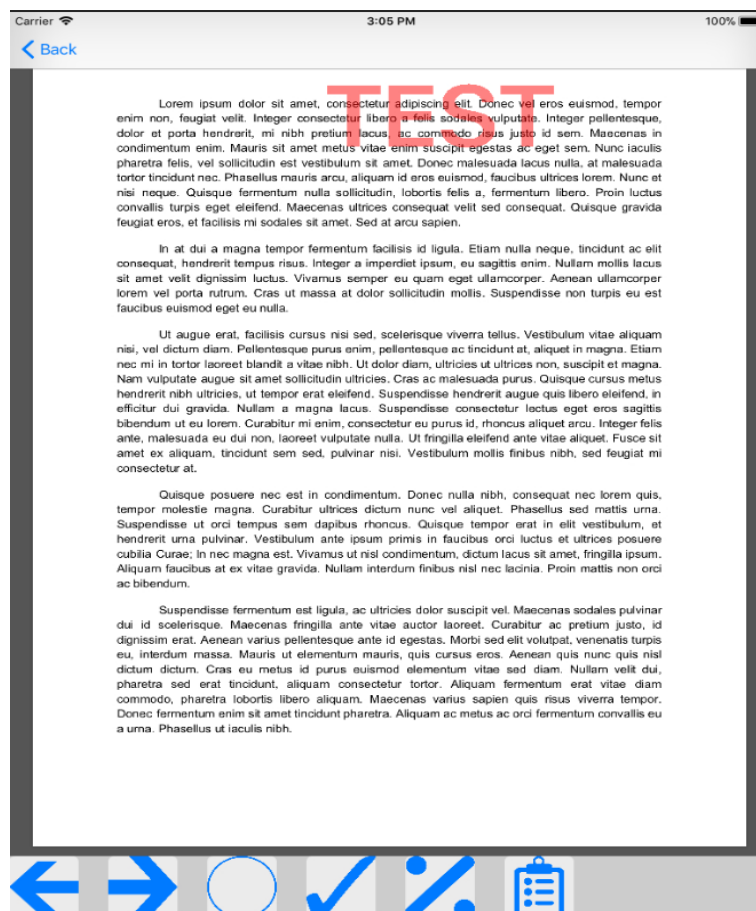
Grænsefladen for 'Registrer på PDF' viewet, som ses på Figur 6.11 består af to views. Det øverste view viser PDF'en, som tilhører projektet.

Det nederste view indeholder knapper, som brugeren kan interagere med. Der er seks forskellige knapper:

De første to er frem og tilbage, hvor brugeren kan bladere mellem de forskellige sider i PDF'en.

De næste tre knapper, er de symboler, som brugeren kan tegne med på PDF'en ved at trykke på knappen og derefter det sted på PDF'en, brugeren ønsker.

Sidste knap er en liste-knap. Her har brugeren mulighed for at afslutte sin registrering.



**Figur 6.11.** 'Registrer på PDF' viewet, som det er implementeret i Rambøll Tilsyn.

### 6.2.3.3 Implementering

Når der navigeres ind i 'Registrering på PDF' viewet, loades PDF'en fra Firebase og gemmes lokalt. Den loades herefter ind i viewet. Efterfølgende oprettes et UIView, hvori symbol-knapperne oprettes. Når begge dele er oprettet og loadet, hentes information fra den JSON fil der blev hentet i 'Project List' viewet, og så tegnes de objekter, som er gemt i PDF'en.

Når disse er tegnet, vises PDF'en for brugeren.

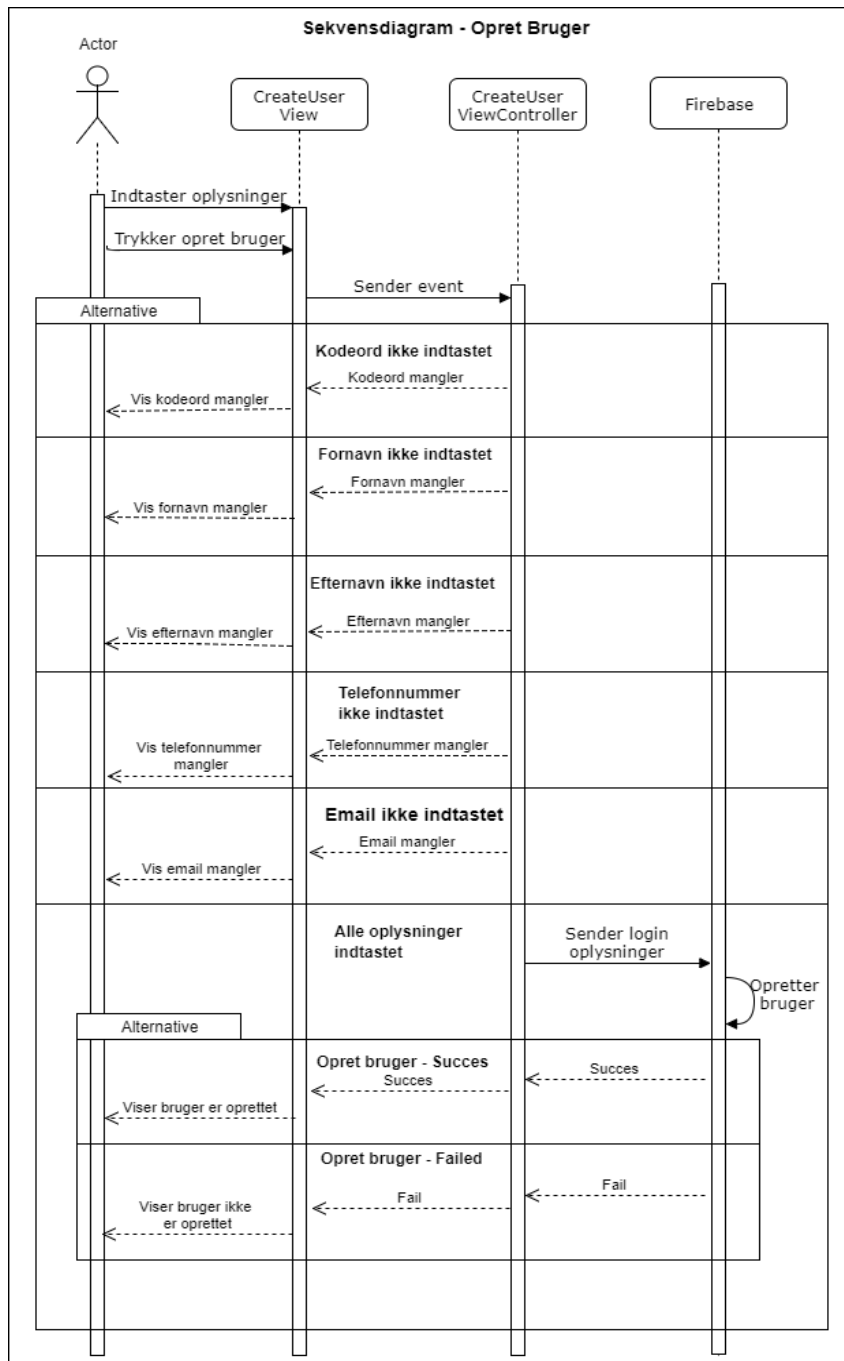
For en mere detajeret beskrivelse af implementeringen og kode snips, henvises til Arkitektur og Design dokumentations Afsnit 2.2.6 Registrering på PDF.

### 6.2.4 Opret Bruger

Dette afsnit indeholder en gennemgang af den grafiske brugergrænseflade, design og implementering af 'Opret Bruger' viewet i Rambøll Tilsyn. For den fulde dokumentation henvises til Arkitektur og Design dokumentationens Afsnit 2.2.4 Opret Bruger.

#### 6.2.4.1 Design

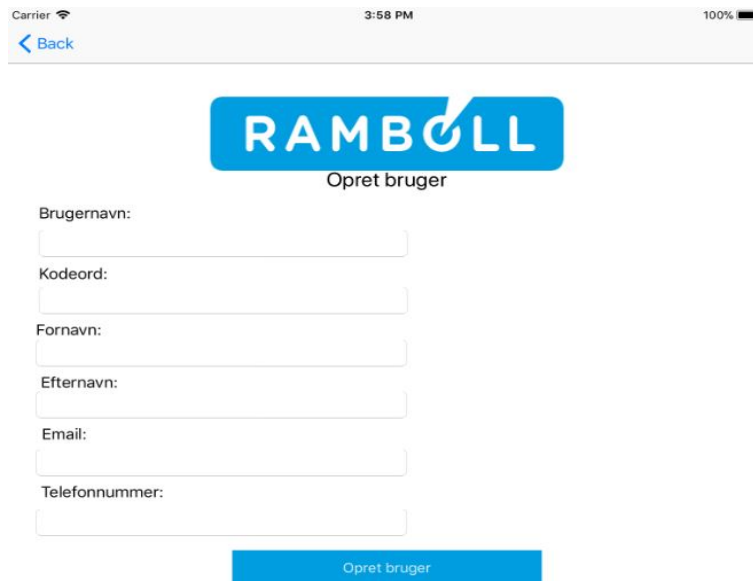
På Figur 6.12 ses sekvensdiagrammet for 'Opret bruger' viewet til Rambøll Tilsyn.



**Figur 6.12.** Sekvensdiagram for 'Opret bruger' i Rambøll Tilsyn.

#### 6.2.4.2 Grafisk brugergrænseflade

I 'Opret bruger' viewet er der lavet felter til alt den information, som skal tastes ind om en bruger. Se Figur 6.13



**Figur 6.13.** 'Opret bruger' viewet, som det er implementeret i Rambøll Tilsyn.

#### 6.2.4.3 Implementering

Først valideres alle felterne og hvis inputs i alle felter bliver godkendte, oprettes brugeren der er indtastet i viewet. Når brugeren er oprettet korrekt, skrives informationen, der ikke er log ind specifikke ind i databasen. Dette er information som; fornavn og efternavn.

For en mere deltageret beskrivelse af implementeringen og kode snips, henvises til Arkitektur og Design dokumentations Afsnit 2.2.4 Opret Bruger.

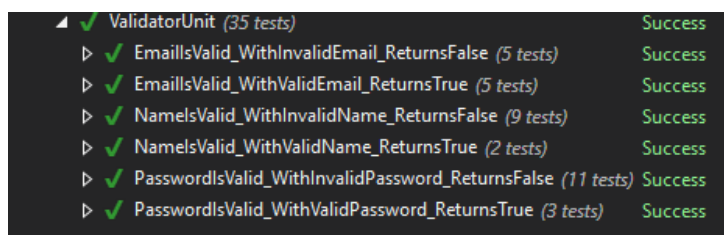
# 7 Test

Dette kapitel indeholder en kort beskrivelse af de tests som er blevet udført i forhold til validering af Rambøll Tilsyn.

## 7.1 Automatiseret test

I følgende afsnit beskrives unit test, der er lavet. Frameworket Nunit[13] er blevet brugt til testing af klasser. Den fulde automatiserede test dokumentation, henvises i Modul- og Integrationstest dokumentet Afsnit 1 Unit Test.

**Validator** Validator-klassen er en statisk klasse, som håndterer validering af input-felterne i applikationen. Applikationens testcases ses på Figur 7.1.



**Figur 7.1.** Screenshot af test sessionen på ValidatorUnit.

En af unit testene, som kan ses på Figur 7.1, er for funktionen PasswordIsValid. Her bliver der testet for både valide og invalide kodeord.

Funktionen PasswordIsValid kontrollerer at et password opfylder følgende krav:

- Minimum 6 karakter lang
- Minimum et stort bogstav
- Minimum et lille bogstav
- Minimum et tal
- Speciel tegn er valgfrit
- Må ikke have mellemrum

Validator-klassen er den eneste klasse, der automatisk testes, da det er den eneste klasse, der indeholder en algoritme. Koden, som benyttes i prototypen, ligger i forskellige frameworks og derfor tages der højde for at disse er vel implementeret og gennemtestet. Dette er grunden til, at der ikke implementeret mere automatisk test.

## 7.2 Manuel test

Denne sektion indeholder de manuelle tests for 'Login' og 'Opret bruger' viewene. Der er skrevet tilsvarende test cases for alle de implementerede views. Disse kan findes i Modul- og Integrationstest dokumentets Afsnit 2 Manuelle Test.

### Login tests

#### Pass/fail criteria:

Alle steps skal verificeres på Rambøll Tilsyn applikationen:

- Verificer at bruger kan logge ind med korrekte brugeroplysninger.
- Verificer at bruger ikke kan logge ind med forkert e-mail.
- Verificer at bruger ikke kan logge ind med forkert kodeord.

### Opret bruger test

#### Pass/fail criteria:

Alle steps skal verificeres på Rambøll Tilsyn applikationen:

- Verificer at en bruger kan oprettes, hvis alle felter er udfyldte korrekt.
- Verificer at en bruger ikke kan oprettes, hvis e-mail feltet mangler at blive udfyldt.
- Verificer at en bruger ikke kan oprettes, hvis kodeord feltet mangler at blive udfyldt.
- Verificer at en bruger ikke kan oprettes, hvis fornavn feltet mangler at blive udfyldt.
- Verificer at en bruger ikke kan oprettes, hvis efternavn feltet mangler at blive udfyldt.
- Verificer at en bruger ikke kan oprettes, hvis telefonnummer feltet mangler at blive udfyldt.
- Verificer at en e-mail skal indeholde et snabel-a (@) for at blive godkendt.
- Verificer at kodeordet skal overholde kravene omkring minimum 6 cifre, et stort bogstav, et lille bogstav og et tal.

De manuelle tests til Rambøll Tilsyn er med til at sikre at både funktionalitet og forløb i applikationen fungerer.

# 8

## Resultater

Dette kapitel gennemgår resultaterne, som er opnået ved arbejde med Rambøll Tilsyn. Herunder vises en tabel over User Stories, som er opdelt efter MoSCoW, og ud fra hver User Story kan resultatet ses. For resultaterne baseret på User Story niveau henvises til Accepttestspecifikationen, som er vedlagt i bilag.

Must	Resultat
Log ind (CRS-1)	Fuld funktionsdygtig
Opret bruger (CRS-2)	Fuld funktionsdygtig
Opret en registrering på PDF-tegning (CRS-4)	Fuld funktionsdygtig
Opret fluebensobjekt på PDF-tegning (CRS-5)	Fuld funktionsdygtigt
Opret minusobjekt på PDF-tegning (CRS-11)	Fuld funktionsdygtigt
Slet objekt på PDF-tegning (CRS-12)	Ikke implementeret
Afslut registrering på PDF-tegning (CRS-13)	Ikke implementeret
Opret projekt (CRS-16)	Ikke implementeret
Should	
Opret billedeobjekt på PDF-tegning (CRS-6)	Ikke implementeret
Opret tekstfeltobjekt på PDF-tegning (CRS-7)	Ikke implementeret
Opret cirkelobjekt på PDF-tegning (CRS-10)	Fuld funktionsdygtigt
Could	
Rediger brugeroplysninger (CRS-3)	Ikke implementeret
Opret kommentarfeltobjekt på PDF-tegning (CRS-8)	Ikke implementeret
Opret pilobjekt på PDF-tegning (CRS-9)	Ikke implementeret
Won't Have	
Opret en registrering uden PDF tegning (CRS-14)	Ikke implementeret
Afslut registrering uden PDF tegning (CRS-15)	Ikke implementeret
Rediger projektoplysninger (CRS-17)	Ikke implementeret
Se tilsynsrapporter (CRS-18)	Ikke implementeret
Opret sub entreprise (CRS-19)	Ikke implementeret



Resultatet af Rambøll Tilsyn anses som 'ikke helt tilfredsstillende'. Der er startet på en prototype til systemet. Der mangler enkelte funktionaliteter fra **MUST** kategorien. Hvis disse var blevet implementeret, ville prototypen have været færdig, og dette ville have givet et tilfredsstillende resultat for gruppen.

Der blev i slutningen af projektet lagt fokus på at få implementeret funktionalitet, så flowet igennem applikationen fungerede. Her menes der, at brugeren kunne logge ind, oprette en registrering og efterfølgende lave objekter i denne registrering.

For at kunne gøre flowet muligt, uden at skulle hardcode brugere, valgte gruppen at få implementeret, funktionalitet så der gennem applikationen kunne oprettes brugere.

Til dry run af accepttesten var der nogle observationer omkring manglende fejlmeddelelser ved oprettelse af en bruger, og at efter oprettelse af en ny bruger, blev den nye bruger automatisk logget ind. Dette var nogle mindre fejl, hvor at fejlmeddelelser blev tilføjet og der nu ikke skiftes bruger, når en ny bruger oprettes.

# 9 Erfaringer

Dette kapitel har til formål at beskrive de samlede erfaringer for projektgruppen.

Der er igennem projektet blevet opnået en masse erfaringer i gruppen.

Der var fra Rambølls side først et ønske om en iOS applikation, men da ingen i gruppen havde MacBooks og Rambøll ikke stillede det til rådighed, blev det aftalt, at der skulle skrives i Xamarin, som er et cross-platform værktøj, der giver mulighed for at udvikle i iOS og Android. I starten af projektet blev der udviklet i Xamarin Forms, da dette gjorde det muligt at genbruge både UI og Business logic mellem både iOS og Android applikationen. Da der skulle udvikles mere avanceret funktionalitet omkring PDF registrering, blev mulighederne i Xamarin Forms for begrænset[11]. Der blev i steftet skiftet til native Xamarin.iOS udvikling, som gav mulighed for at implementere funktionaliteten. Gruppen fandt også ud af at selvom, der udvikles i Xamarin, skulle der bruges en fysisk MacBook til at kompilere koden.

Cross-platform udviklingen har sine fordele og ulemper. At der i Xamarin f.eks. kan udvikles i C# er en fordel, da dette er et sprog gruppen har haft undervisning i og derfor skal der ikke læres et nyt programmeringssprog.

En af ulemperne var, at der skulle tages højde for mange forskellige enheder i code behind for, at UI ser ens ud på alle enheder. Hvis der f.eks. blev udviklet i Android studio, kunne man få det samme layout ligegyldig, hvilken enhed man deployerede på.

Gruppen har erfaret, at iOS udvikling kan være besværlig, hvis ikke der udvikles i Swift[14] eller Objective-C[15]. Hvis der skal udvikles i Swift, skal man have tilegnet sig en MacBook, som kan kører Xcode[16]. Det er ikke ligesom f.eks. Android Studio, som er et gratis værktøj at hente og benytte. Hvis gruppen skulle lave dette projekt igen, ville der blive udviklet native i Swift og man vil have udviklingsværktøj i form af MacBooks til rådighed og derved er alle problemer med byggemiljø til at kompilere koden løst.

Der kan tages mange erfaringer omkring applikationsudviklingen med til fremtidige projekter.

# 10 Fremtidigt arbejde

I forlængelse af projektet Rambøll Tilsyn er der mange muligheder for videreudvikling og optimering.

## **Fra prototype til produkt**

Det første arbejde, der skal udføres, er implementeringen af de User Stories, som ikke er blevet implementeret i prototypen. Her tænkes bl.a. på User Stories fra **Must**-kategorien, da dette vil give et minimum viable product, som vil kunne gøre, at Rambøll kunne begynde at bruge systemet, mens resten blev implementeret.

## **Database**

Der er i projektet blevet lavet et produkt med en Firebase database. Et andet punkt til fremtidigt arbejde kunne være at få flyttet database delen over på Rambølls egne servere. Dette kommer an på om de ønsker at have databasen intern i huset eller om de tænker, at produktet fungerer bedre med Firebase databasen og blot skulle overtage administrationen af denne.

## **Byggeserver**

Der kunne med fordel opsættes en iOS byggeserver, som ville kunne stå for continuous integration[17] test. Dette vil gøre, at hver gang noget kode bliver committet til serveren, vil denne kontrollere om alt fortsat fungerer efter hensigten.

## **Fremtidig funktionalitet**

Når alle User Stories er implementeret, vil der ligge en opgave i at kunne ændre objekter efter de er oprettet og gemt. Hvis der f.eks. oprettes et minus objekt første gang, der laves en registrering på et projekt, og dette skal ændres til et flueben næste gang at der bliver lavet en registrering. Minus objektet skal her kunne ændres ved at brugeren laver et long press på objektet. Der vil komme et vindue op, hvor brugeren kan ændre symbolet og skrive en kommentar hertil. Dette vil så blive gemt og vil komme med i tilsynsdokumentet, når dette bliver eksporteret.

En anden fremtidig udvikling til systemet vil være 3D rendering af tegninger. Dette skal kunne gøres så brugeren i stedet for at registre på en PDF-tegning vil kunne få en 3D tegning af byggeprojektet, som der kan laves registreringer på.

Når ovenstående er implementeret, uden fremtidig funktionalitet, vil det sidste skridt være at overlevere systemet til Rambøll, så det kan blive implementeret i deres systemer og arbejdsprocesser.

# 11 Konklusion

Dette kapitel har til formål, at beskrive den samlede konklusion for projektet.

Formålet med projektet var at udvikle et system til Rambøll, for at kunne digitalisere deres registreringsprocesser på byggepladser rundt om i Danmark og som i dag udføres med papir og blyant.

Dette udmundede i produktet Rambøll Tilsyn, som består af en applikation samt en database.

Firebase står for databasedelen og Authentication i forhold til brugere. Dette har betydet, at der kunne drages nytte af en masse funktionalitet, validering af brugeroplysninger samt at holde kodeord sikre.

I applikationen, som er implementeret som iOS applikation, er der mulighed for at oprette brugere, oprette projekter, samt oprette registreringer. Dette udgør største delen af det flow, som blev designet for systemet. En prototype vil kunne bruges som pilotprojekt og testes i rigtige arbejdssituationer, mens der blev arbejdet på et endeligt produkt.

Det er lykkedes at implementere de fleste essentielle funktioner i prototypen, med nogle undtagelser, herunder 'Afslut registrering på PDF' og 'Slet objekt på PDF-tegning'.

Prototypen er blevet testet i et lukket miljø og er derved ikke blevet brugertestet ude i rigtige arbejdssituationer. Dette kan begrænse prototypens pålidelighed til, om den vil fungere ved første test eller om skulle opstå der er nogle uforudsete komplikationer.

Rambøll Tilsyn, som prototypen er i dag er den næsten et minimum viable product, og vil med videre arbejde kunne sættes i drift inden for en overskuelig tidshorisont.

# Ordforklaring

Forkortelse	Forklaring
Android	Android Operation System
Console	Firebases web tilgang
Domæne navn	Domain name[18]
Firebase	Fælles betegnelse for Firebase database og SDK til firebase
Framework	Miljø et program laves til at kunne udføres
Internet	En trådløs internetforbindelse som f.eks. 3G eller Wifi
iOS	iPhone Operating Sytem
iOS.X	Mindste OS krav til Xamarin
JSON	JavaScript Object Notation
Lokal navn	Den del af e-mail der står før "@"
MVC	Model-View-Controller
PDF	Portable Document Format
Rambøll Tilsyn	Applikationen der implementeres i dette projekt
Swift	Apples programmeringssprog
UI	Grafisk brugergrænseflade
UML	Unified Modeling Language

# Litteratur

- [1] Erhvervsstyrelsen. Redegørelse for danmarks digitale vækst 2016. URL <https://erhvervsstyrelsen.dk/redegoerelse-danmarks-digitale-vaekst-2016>. Last Visited d. 15.12.2017.
- [2] Agile Business Consortium. Moscow - analysis tool. URL <https://www.agilebusiness.org/content/moscow-prioritisation-0>. Last Visited 12.09.2017.
- [3] Torben Gregersen. Rapport vejledning, 2017. Vejledning til udfaerdigelse af projektrapporter v1.2.
- [4] UML.org. Uml. URL <http://www.uml.org/>. Last Visited d. 24.11.2017.
- [5] Google Firebase. Firebase, . URL <https://firebase.google.com/>. Last Visited d. 20.09.2017.
- [6] Xamarin. Xamarin platform. URL <https://www.xamarin.com/platform>. Last Visited d. 17.10.2017.
- [7] Microsoft. Visual Studio, 2017. URL <https://www.visualstudio.com/>. Last Visited 23.09.2017.
- [8] Apple Inc. Uiviewcontroller - uikit | apple developer documentation, . URL <https://developer.apple.com/documentation/uikit/uiviewcontroller>. Last Visited d. 16.12.2017.
- [9] Google. Firebase analytics. URL <https://firebase.google.com/docs/analytics/>. Last Visited d. 15.12.2017.
- [10] Firebase. Firebase authentication, . URL <https://firebase.google.com/products/auth/>. Last Visited d. 18.10.2017.
- [11] Xamarin. Forms. URL <https://www.xamarin.com/forms>. Last Visited d. 15.11.2017.
- [12] TutorialTeacher. Mvc. URL <http://www.tutorialsteacher.com/mvc/mvc-architecture>. Last Visited d. 04.12.2017.
- [13] Unit testing framework for .net. Framework. URL <http://nunit.github.io/>.
- [14] Apple. Swift, . URL <https://developer.apple.com/swift/>. Last Visited d. 07.12.2017.
- [15] Techopedia Inc. What is objective-c (objc)? - definition from techopedia, . URL <https://www.techopedia.com/definition/10205/objective-c-objc>. Last Visited d. 13.12.2017.
- [16] Apple. Xcode, . URL <https://developer.apple.com/xcode/>. Last Visited d. 07.12.2017.
- [17] Martin Fowler. Continuous integration. URL <https://martinfowler.com/articles/continuousIntegration.html>. Last Visited d. 14.12.2017.

- [18] Techopedia Inc. What is a domain name? - definition from techopedia, . URL **https://www.techopedia.com/definition/1327/domain-name**. Last Visited d. 15.12.2017.