

INGENIØRHØJSKOLEN AARHUS UNIVERSITET

BACHELORPROJEKT

RAMBØLL TILSYN



Arkitektur & Design

PROJEKT NR. 17103

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

VEJLEDER: LARS CHRISTIAN JENSEN

DATO: 19/12-2017

Indholdsfortegnelse

Kapitel 1	Arktitektur	4
1.1	Domænemodel	5
1.2	Klasse diagram	6
Kapitel 2	Design	7
2.1	Firebase database	7
2.2	Applikation	10
2.2.1	Navigationstræ	11
2.2.2	Login	12
2.2.3	Projekt Oversigt	16
2.2.4	Opret Bruger	21
2.2.5	Opret Projekt	24
2.2.6	Registrering på PDF	27
2.2.7	Eksportering	37
	Ordforklaring	39
	Litteratur	40

Læsevejledning

I denne rapport redegøres for valgene bag arkitekturen og designet til Rambøll Tilsyn.

Arkitektur og design dokumentationen kan overordnet inddeles i 2 sektioner:

Første sektion er en beskrivelse af arkitekturen for systemet.

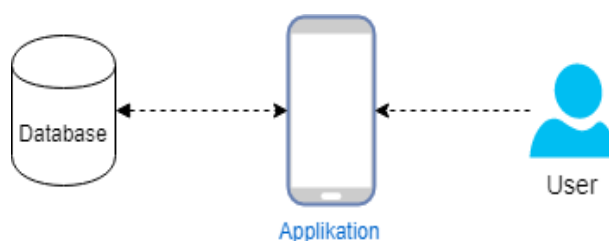
I anden sektion er der en beskrivelse af designvalgene til henholdsvis Firebase og Rambøll Tilsyn.

I slutningen af rapporten findes ordforklaring og litteraturliste.

1 Arktitektur

Dette kapitel viser arkitekturen for applikationen Rambøll Tilsyn.

På Figur 1.1 ses oversigten over systemet og de forskellige elementers relationer til hinanden i systemet.



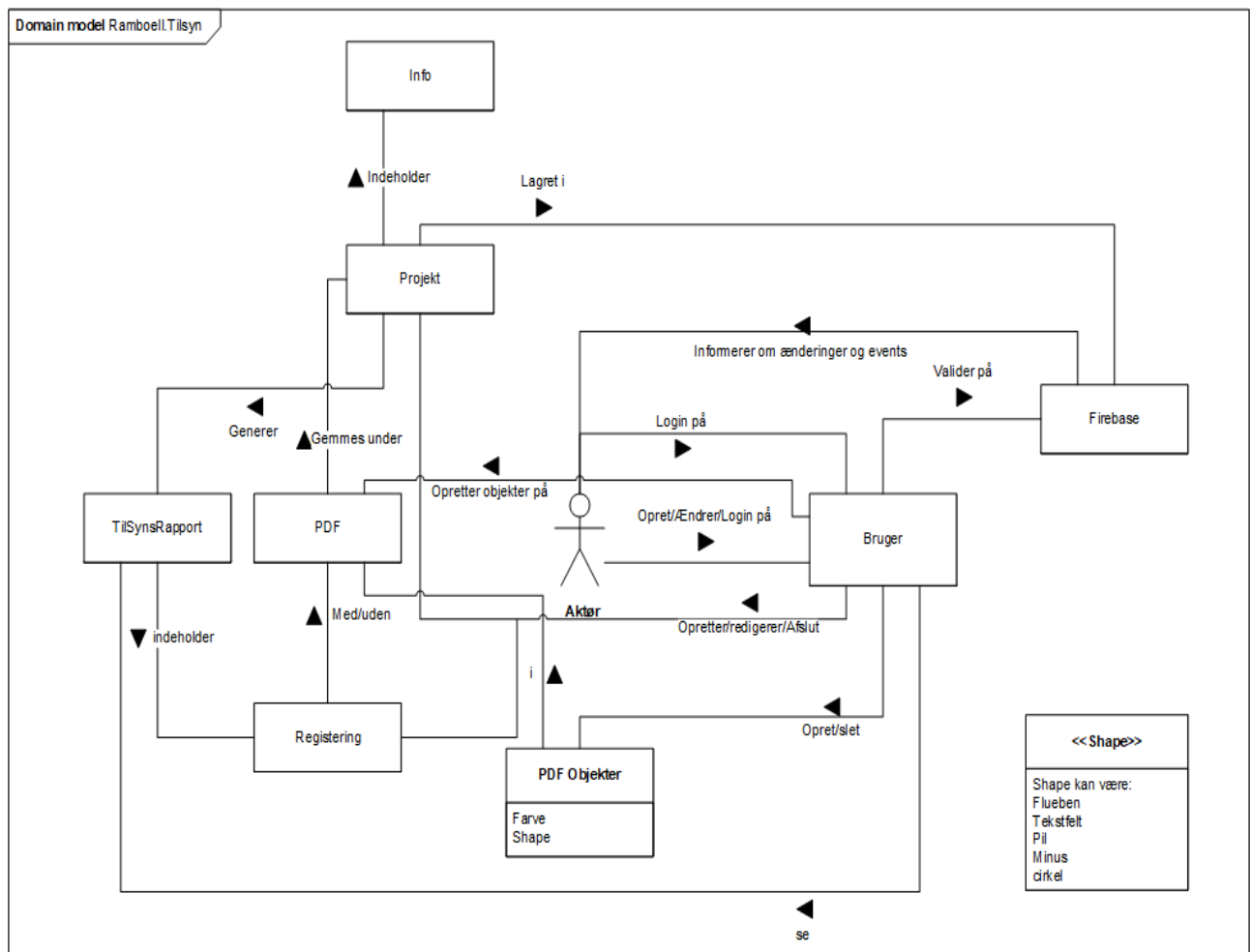
Figur 1.1. Oversigt over systemet.

Det ses på Figur 1.1, at brugeren benytter applikationen. Applikationen kommunikerer via internettet til databasen.

Yderligere beskrivelse af systemet kan findes i Kravspecifikationens Afsnit 2 Systembeskrivelse, som er vedlagt i bilag.

1.1 Domænemodel

Denne domænemodel giver et overblik over, hvordan systemet er bygget op. Domæne modellen viser forbindelserne mellem de forskellige elementer i systemet og hvordan de interagerer med hinanden. Dette gør det lettere at gå fra domænemodellen til implementationen. I dette projekt er der udarbejdet én domænemodel, som ses på Figur 1.2.



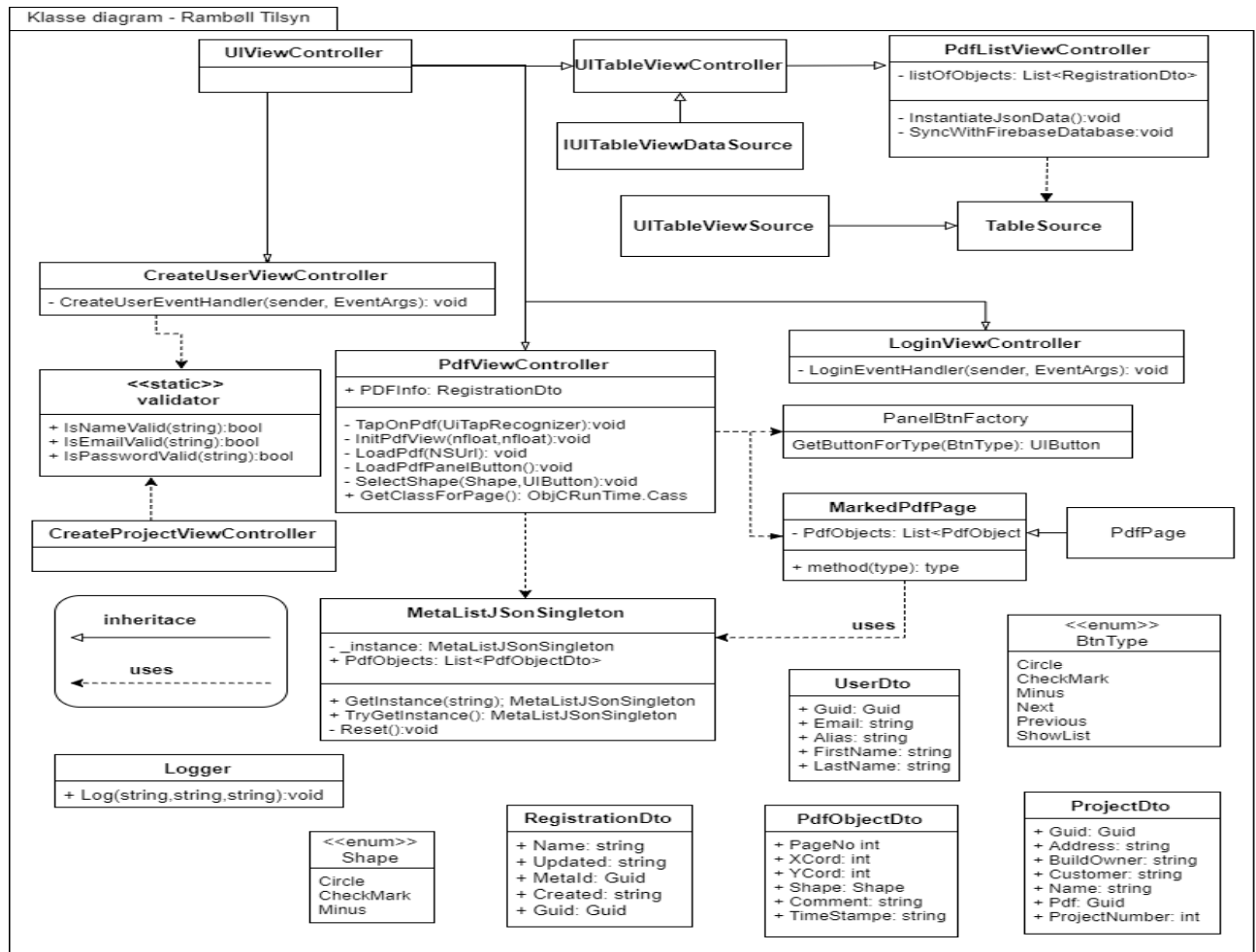
Figur 1.2. Domænemodel for Rambøll Tilsyn.

Brugeren logger ind på Rambøll Tilsyn. Her kan brugeren oprette et projekt, som kan indeholde flere registreringer. I registreringen kan der tilknyttes en PDF-tegning, hvorpå brugeren kan oprette forskellige objekter eller fjerne forkerte objekter.

Når brugeren er færdig med en registrering, vil objekter, PDF-tegning mv. blive gemt i Firebase databasen.

1.2 Klasse diagram

På Figur 1.3 ses der en klasse diagram over Rambøll TilSyn vist i standard UML notation[1]. Klassediagrammet udpeger essentielle funktioner og felter for klassen i applikationen og visuelt viser deres relationer til hinanden.



Figur 1.3. Klassediagram for Rambøll TilSyn.

Her ses, at alle ViewControllers er nedarvet fra UIViewController[2], som er en klasse, der stammer fra Xamarin.iOS frameworket. UITableViewController er en specialiseret klasse fra Xamarin.iOS frameworket, der er nedarvet fra UIViewController. Formålet med denne er at vise en liste af data, som man selv har implementeret. Logger klassen bruger Firebase Analytics[3] for at logge events fra applikationen, som en erstatning for at skrive sine fejlbeskeder ud i Console lokalt[4]. Fordelen ved dette er, at man kan logge diverse events fra alle devices og se det på Firebase Console.

2 Design

Dette kapitel indeholder en beskrivelse af designvalgene for Firebase databasen og applikationen samt, hvordan disse er implementeret.

2.1 Firebase database

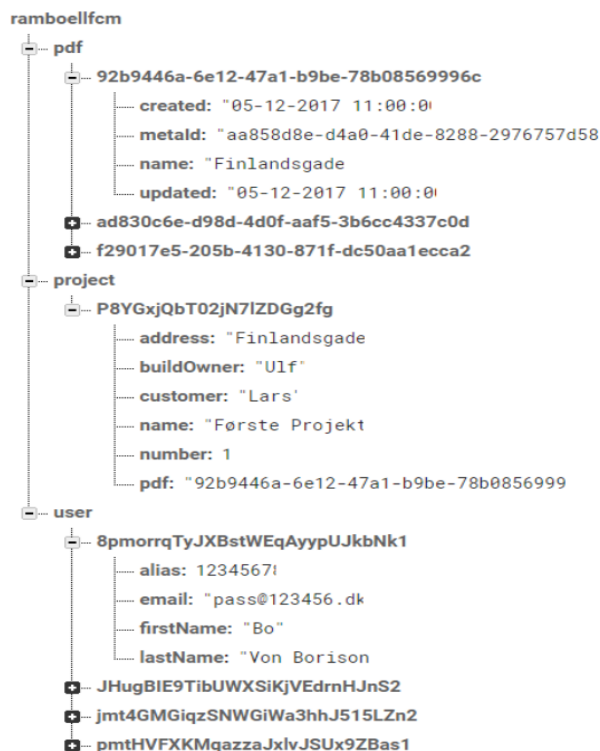
Rambøll Tilsyn benytter tre Firebase produkter for at fungere, Authentication(Auth), Realtime Database og Storage.

Firebase Auth[5] bruges til håndtering af brugere som login og oprettelse. Den indeholder data som e-mail, en hashet password og en unik identifier, der bliver skabt af Firebase Authentication, når brugeren bliver registreret. Firebase tilbyder en Console , som er en hjemmeside, der giver mulighed for at se og manipulere data og konfigurere sin Firebase. På Figur 2.1 kan en oversigt med informationer over brugere ses.

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	hSmJI4yMWbfjFFBy4ri0eD6vngT2
test@test.dk	✉	Dec 10, 2017		jmt4GMGiqzSNWGiWa3hhJ515LZ...

Figur 2.1. Oversigt over authentication data i Firebase Console.

Firebase Realtime Database bliver i forbindelse med applikationen brugt som en platform til at indeholde og give et simpelt overblik over informationen, der er gemt i databasen, som vist på Figur 2.2.



Figur 2.2. Oversigt over databasen i Firebase Console.

På Figur 2.2, ses hvordan applikationens database er struktureret som et træ med en root node[6], der hedder ramboellfcm og som derfra bliver opdelt i 3 noder: **pdf**, **project** og **user**.

Pdf-node indeholder en liste af Guid [7], som er keys til en pdf-node og som bruges til at tilgå de specifikke pdf-noder. Under hver specifik pdf node er der 4 key-value-pair[8]. Disse keys fungerer også som en reference til Firebase Storage, når det bliver nødvendigt at hente PDF filen ned lokalt.

Dette gør, at man kan se, hvilke projekter og pdf'er der findes i databasen og afvente med at downloade PDF'en, indtil man har brug for den. På den måde kan man få en oversigt over, hvilket projekter og pdf'er der er og kun hente de data, man har behov for.

- **created**

Giver en dato for, hvornår objektet er oprettet med en nøjagtighed i sekunder, der kan ændres, hvis det føles nødvendigt.

- **metaId**

En reference til JSON-filen der er gemt i Firebase Storage[9], som indeholder de objekter, der skal tegnes på pdf'en. Disse bliver brugt, når der skal indlæses nye objekter, der bliver oprettet på pdf'en.

- **name**

Navnet på PDF'en som er skrevet i klar tekst og indtastet af brugeren. En guid vil ikke give meget mening for brugeren.

- **updated**

En timestamp der fortæller hvornår JSON-filen, som meta-filen refererer til, sidst er blevet ændret. Dette bruger applikationen til at sammenligne den lokale version af JSON-filen med den, der ligger i Storage.

Project-node har den samme struktur som **pdf-node** af samme grund. Der er i stedet for 6 key-value-pair.

- **address**

Værdien gemt i address vil være et vejnavn, der fortæller, hvor projektet befinder sig.

- **customer**

Denne key indeholder kontaktinformationerne for, hvem der udføres tilsynsrapporter for.

- **name**

Værdien gemt i name vil være navnet på projektet.

- **number**

Rambøll bruger et projektnummer til alle deres projekter. Dette indsætter de selv som brugere og derfor gemmes denne information her

- **pdf**

En reference til en pdf-node, samt en reference til Firebase Storage så applikationen kan downloade PDF'en, når den først har brug for det.

User node indeholder brugerdefineret data, der tilhører brugerne af applikationen. Det id der bliver brugt som key til user stammer fra Firebase Authentication og er en reference til User guid, som ses på Figur 2.1.

- **alias**

Her gemmes telefonnummeret til brugeren.

- **email**

Brugerens e-mail vil blive gemt i både Firebase- Authentication og Storage for at gøre det let tilgængelig at tilgå.

- **firstName**

Fornavnet på brugeren.

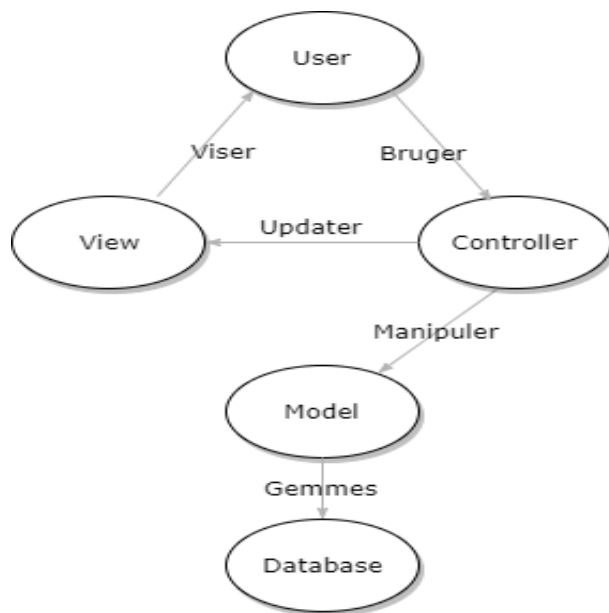
- **lastName**

Efternavnet på brugeren.

2.2 Applikation

Implementeringen af applikationen sker i Xamarin iOS, da der i den indledende analyse blevet overset nogle begrænsninger i forhold til funktionaliteten, når der skulle arbejdes med PDF'er i Xamarin Forms[10].

Til udvikling af applikationen er der brugt et MVC design [11], som vist på Figur 2.3

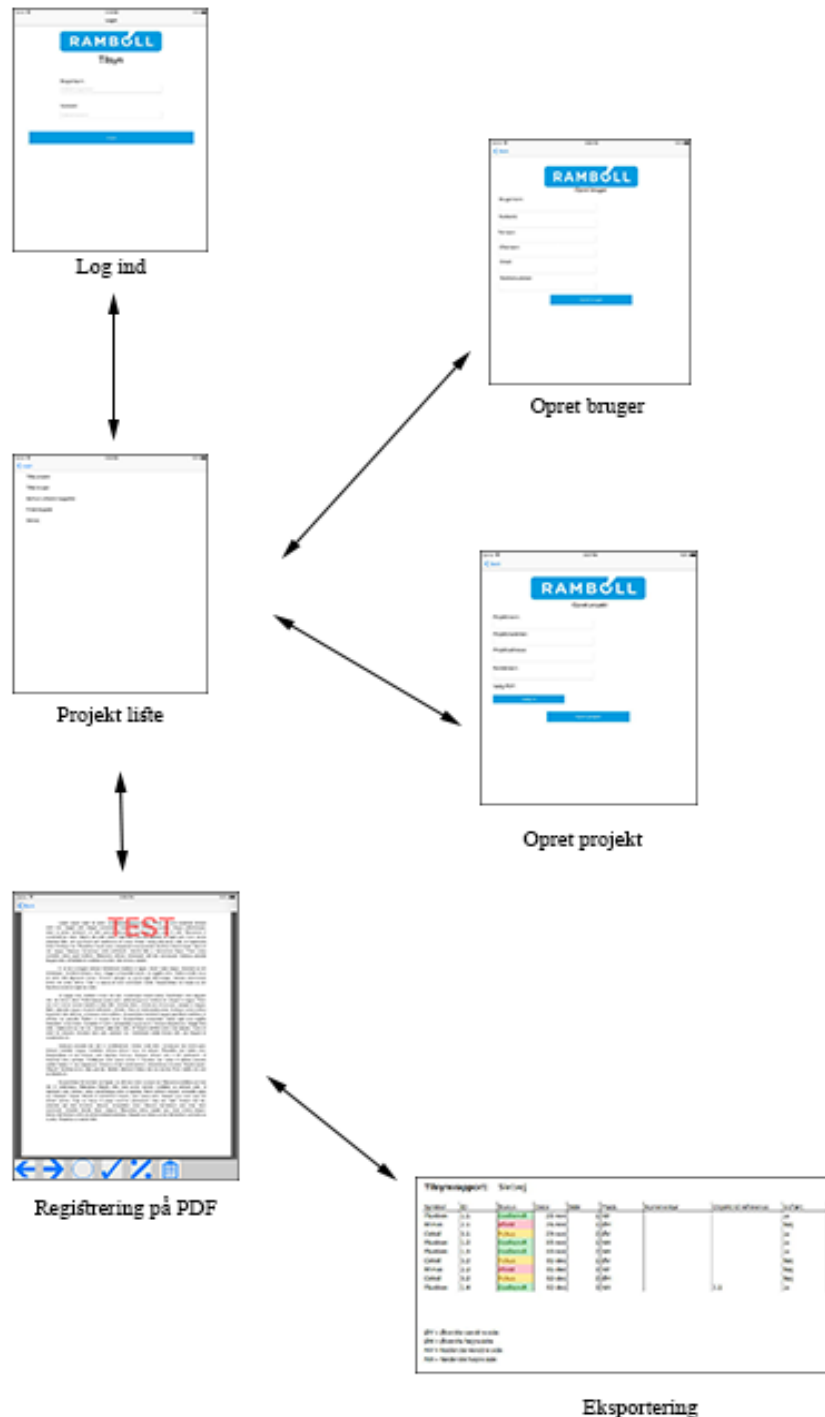


Figur 2.3. MVC design for Rambøll Tilsyn.

Model-View-Controller er et designpattern til opbygning af user interfaces. **Model** indeholder alt data. Det er **Controllerens** opgave at manipulere det til Viewet. **Viewet** er det, som ses af brugeren, og som der kan interageres med. Hvis brugeren interagerer med applikationen, er det Controllerens opgave at få det ned i Modellen og få kaldt det nye view frem. Modellen i Rambøll Tilsyn ligger i Firebase. Der er brugt MVC design pattern til alle views i applikationen.

2.2.1 Navigationstræ

Figur 2.4 viser et navigationstræ for Rambøll Tilsyn. Her ses flowet gennem applikationen og hvordan man kan tilgå de forskellige views.



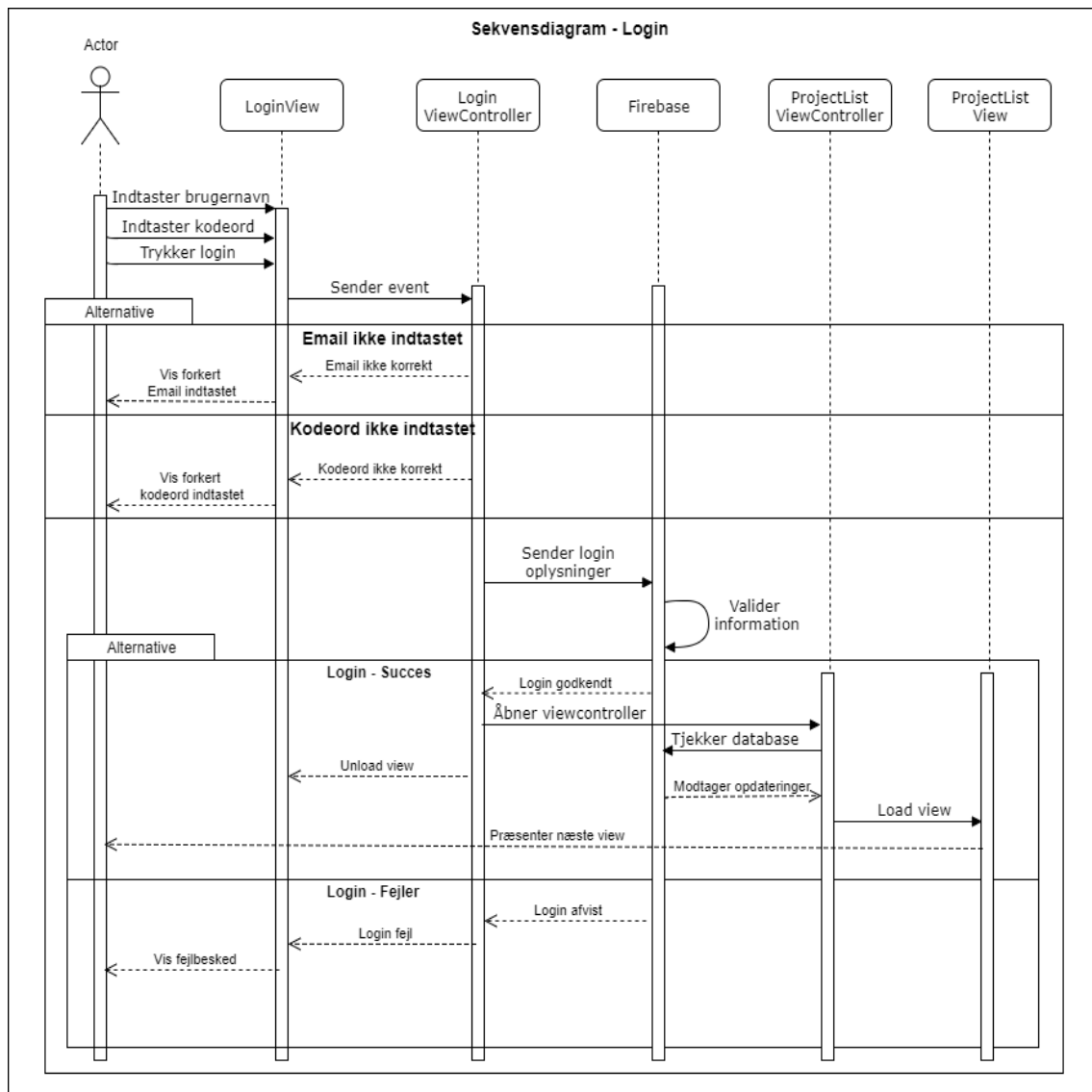
Figur 2.4. Navigasjonstræ for Rambøll Tilsyn.

2.2.2 Login

Dette afsnit indeholder en gennemgang af den grafiske brugergrænseflade, design og implementering af 'Login' viewet i Rambøll Tilsyn.

2.2.2.1 Design

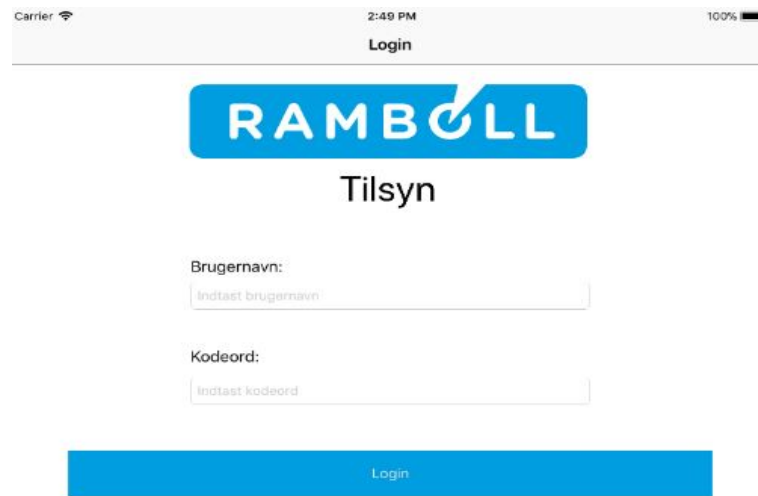
På Figur 2.5 ses sekvensdiagrammet for 'Login' viewet til Rambøll Tilsyn.



Figur 2.5. Sekvensdiagram for 'Login' i Rambøll Tilsyn.

2.2.2.2 Grafisk brugergrænseflade

I 'Login' viewet er der lavet felter til at bruger indtaster sit brugernavn og kodeord. Se Figur 2.6



The screenshot shows a mobile application interface for 'Rambøll Tilsyn'. 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 'RAMBØLL' logo in a blue rounded rectangle, followed by the text 'Tilsyn'. There are two input fields: 'Brugernavn:' with a placeholder 'Indtast brugernavn' and 'Kodeord:' with a placeholder 'Indtast kodeord'. At the bottom, there is a large blue button labeled 'Login'.

Figur 2.6. 'Login' viewet som det er implementeret i Rambøll Tilsyn.

2.2.2.3 Implementering

I dette afsnit vil der blive beskrevet funktionaliteten for de vigtigste funktioner i koden tilhørende 'Login' viewet.

På Figur 2.7 ses funktionen for `CheckIfUserSignedIn()`.

```
1 reference | Ao Li, 5 days ago | 1 author, 3 changes
public void CheckIfUserSignedIn()
{
    listenerHandle = Auth.DefaultInstance.AddAuthStateDidChangeListener((auth, user) => {
        if (user != null)
        {
            Console.WriteLine("User Currently Logged In: {0}", user.Email);

            if (Storyboard.InstantiateViewController("ProjectListViewController") is ProjectListViewController projectList)
            {
                NavigationController.RemoveFromParentViewController();
                NavigationController.PushViewController(projectList, true);
            }
        }
        else
        {
            Console.WriteLine("No Users logged In");
            // No user is signed in.
        }
    });
}
```

Figur 2.7. Kode snip - `CheckIfUserSignedIn()` fra `LoginViewController.cs`

Først tjekker funktionen Firebasees default instance og ser om, der er en bruger, som er logget ind.

Er der en bruger logget ind, hoppes log ind over, og projekt listen bliver vist.

Ellers udskriver den at ingen bruger er logget ind.

På Figur 2.8 ses funktionen for `ViewDidLoad()`.

```
4 references | Msknudsen, 2 days ago | 2 authors, 5 changes
public override void ViewDidLoad()
{
    base.ViewDidLoad();
    LoginBtn.TouchUpInside += LoginBtn_TouchUpInside;
    //LogoutBtn.TouchUpInside += LogoutBtn_TouchUpInside;

    EmailTxt.Placeholder = "Indtast brugernavn";

    PasswordTxt.SecureTextEntry = true;
    PasswordTxt.Placeholder = "Indtast kodeord";

    CheckIfUserSignedIn();
}

3 references | Ao Li, 22 hours ago | 1 author, 2 changes
public override void ViewDidUnload()
{
    base.ViewDidUnload();
    LoginBtn.TouchUpInside -= LoginBtn_TouchUpInside;
}
```

Figur 2.8. Kode snip - `ViewDidLoad()` fra `LoginViewController.cs`

I ViewDidLoad delegeres en eventhandler[12] til LoginBtn.TouchUpInside.

Den indsætter placeholders i tekstfelterne e-mail og kodeord. Derudover maskerer den inputs i kodeords tekstfelt. I ViewDidUnload fjernes eventhandleren igen, da dette er en god kode skik for at undgå memory leak[13].

På Figur 2.9 ses funktionen for LoginBtn.TouchUpInside().



```
2 references | Ao Li, 21 days ago | 1 author, 1 change
private void LoginBtn_TouchUpInside(object sender, EventArgs e)
{
    Auth.DefaultInstance.SignIn(EmailTxt.Text, PasswordTxt.Text, (user, error) => {
        if (error != null)
        {
            AuthErrorCode errorCode;
            if (IntPtr.Size == 8) // 64 bits devices
                errorCode = (AuthErrorCode)((long)error.Code);
            else // 32 bits devices
                errorCode = (AuthErrorCode)((int)error.Code);

            // Possible error codes that SignIn method with email and password could throw
            // Visit https://firebase.google.com/docs/auth/ios/errors for more information
            switch (errorCode)
            {
                case AuthErrorCode.OperationNotAllowed:
                case AuthErrorCode.InvalidEmail:
                case AuthErrorCode.UserDisabled:
                case AuthErrorCode.WrongPassword:
                default:
                    // Print error
                    Console.WriteLine(errorCode.ToString());
                    break;
            }
        }
        else
        {
            if (this.Storyboard.InstantiateViewController("ProjectListViewController") is ProjectListViewController projectList)
            {
                this.NavigationController.PushViewController(projectList, true);
            }
        }
    });
}
```

Figur 2.9. Kode snip - LoginBtn.TouchUpInside() fra LoginViewController.cs

LoginBtn.TouchUpInside tager default instance af Firebase Auth, kalder SignIn metoden for denne instance og indsætter de to parameter skrevet i 'Login' viewet.

Efterfølgende valideres resultatet af log ind forsøget. Hvis der ingen fejl er, altså at e-mail og kodeord er korrekt, så vil projekt listen blive vist.

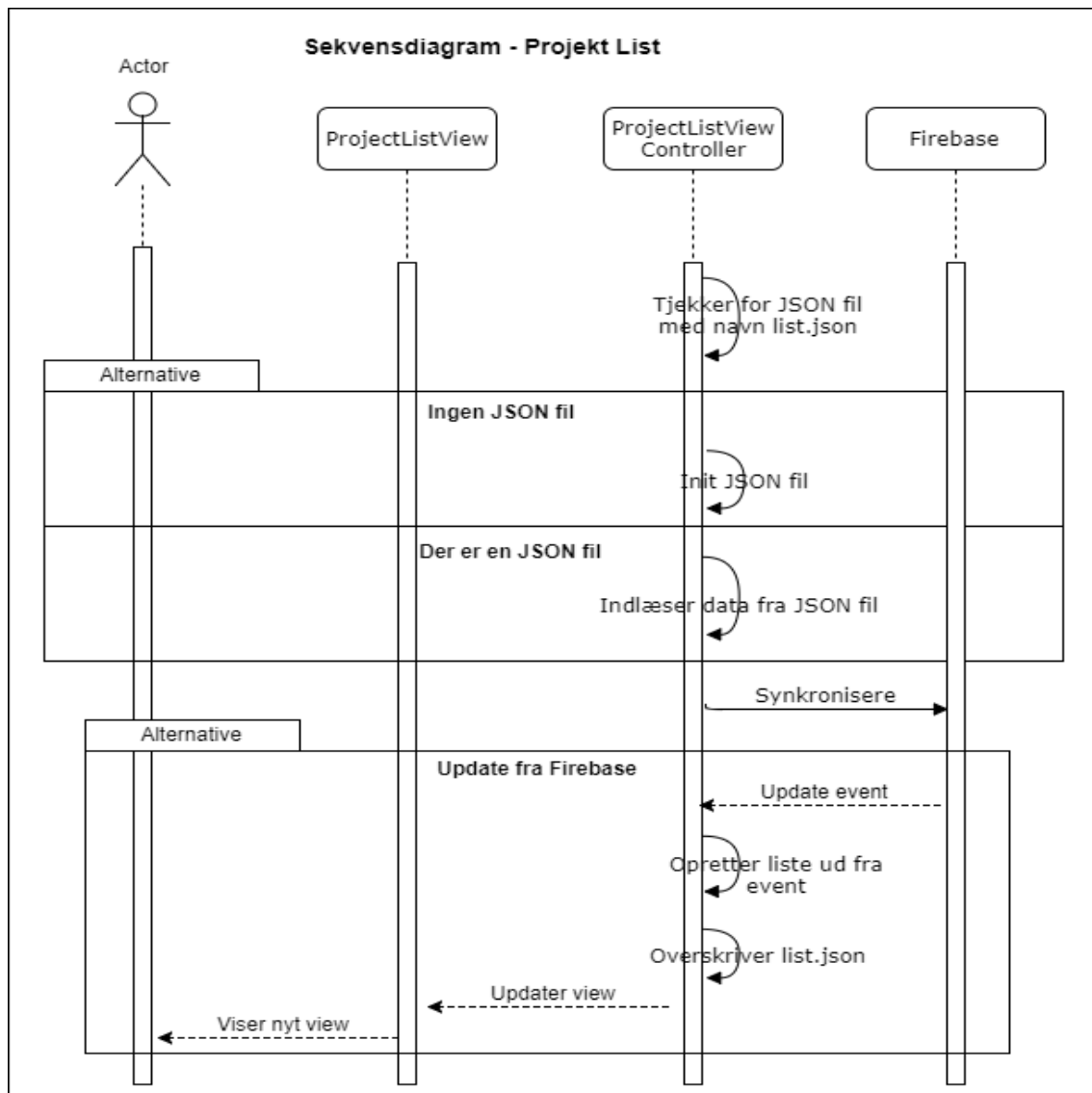
Der kan i switch-casen implementeres fejlhåndtering af diverse fejlkoder.

2.2.3 Projekt Oversigt

Dette afsnit indeholder en gennemgang af den grafiske brugergrænseflade, design og implementering af 'Project List' viewet i Rambøll Tilsyn.

2.2.3.1 Design

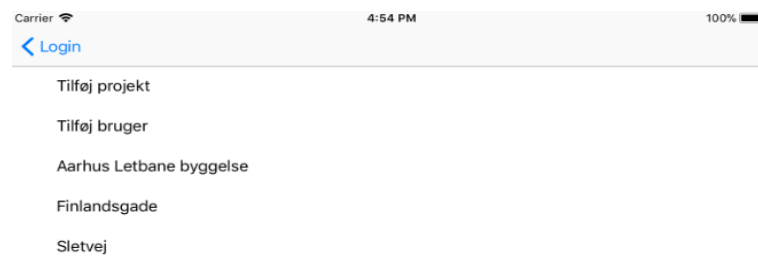
På Figur 2.10 ses sekvensdiagrammet for 'Project List' viewet til Rambøll Tilsyn.



Figur 2.10. Sekvensdiagram for 'Project List' i Rambøll Tilsyn.

2.2.3.2 Grafisk brugergrænseflade

I ProjectListViewet er der en oversigt over, hvilke projekter der ligger i databasen samt mulighed for øverst at tilføje projekt eller bruger. Se Figur 2.11



Figur 2.11. 'Project List' viewet, som det er implementeret i Rambøll Tilsyn.

2.2.3.3 Implementering

I dette afsnit vil der blive beskrevet funktionaliteten for de vigtigste funktioner i koden tilhørende 'Project List' viewet.

På Figur 2.12 ses funktionen for ProjectListViewController().

```
0 references | Ao Li, 2 days ago | 1 author, 5 changes
public ProjectListViewController (IntPtr handle) : base (handle)
{
    _node = Database.DefaultInstance.GetRootReference().GetChild("pdf");
    _node.KeepSynced(true);

    InstantiateJsonFile();
    SyncWithFirebaseDatabase();
}
```

Figur 2.12. Kode snip - ProjectListViewController() fra ProjectListViewController.cs

Det første er, at der hentes en reference node med navn PDF fra Firebase som fortæller, at denne node skal holdes synkroniseret i tilfælde af dataændringer.

Efterfølgende initialiseres JSON-filen som synkroniseres efterfølgende med Firebase.

På Figur 2.13 ses funktionen for InstantiateJsonFile().

```
1 reference | Ao Li, Less than 5 minutes ago | 1 author, 3 changes
private void InstantiateJsonFile()
{
    if (!File.Exists(_file))
    {
        ProjectInfos = new List<RegistrationDto> { new RegistrationDto { Name = "tilføj projekt" }, new RegistrationDto { Name = "tilføj bruger" } };
        File.WriteAllText(_file, JsonConvert.SerializeObject(ProjectInfos));
    }
    else
    {
        var data = File.ReadAllText(_file);
        ProjectInfos = JsonConvert.DeserializeObject<List<RegistrationDto>>(data);
    }
}
```

Figur 2.13. Kode snip - InstantiateJsonFile() fra ProjectListViewController.cs

Der oprettes en JSON-fil, hvis der ikke findes en i forvejen. Denne gemmes lokalt.

Hvis der findes en JSON-fil, læses data fra filen.

På Figur 2.14 ses funktionen for SyncWithFirebaseDatabase().

```
1 reference | Ao LI, 1 day ago | 2 authors, 4 changes
private void SyncWithFirebaseDatabase()
{
    // var s = _node.GetQueryOrderedByKey();
    handleReference = _node.ObserveEvent(DataEventType.Value, (snapshot) =>
    {
        ProjectInfos = new List<RegistrationDto> { new RegistrationDto { Name = "Tilføj projekt"}, new RegistrationDto { Name = "Tilføj bruger" } };

        foreach (var element in snapshot.GetValue<NSDictionary>())
        {
            ProjectInfos.Add(new RegistrationDto
            {
                Guid = new Guid(element.Key.Description),
                Name = element.Value.ValueForKeyPath((NSString)"name").Description,
                MetaId = new Guid(element.Value.ValueForKeyPath((NSString)"metaId").Description),
                Created = element.Value.ValueForKeyPath((NSString)"created").Description,
                Updated = element.Value.ValueForKeyPath((NSString)"updated").Description
            });
            if (File.Exists(_file))
            {
                File.WriteAllText(_file, JsonConvert.SerializeObject(ProjectInfos));
                TableView.ReloadData();
            }
            else
            {
                Console.WriteLine("No json file containing a list of projectInfo found");
            }
        }
    });
}
```

Figur 2.14. Kode snip - SyncWithFirebaseDatabase() fra ProjectListViewController.cs

Noden, der oprettes i konstrukteren, som ses på Figur 2.12, vedhæftes et event, som kaldes hver gang noden ændres.

Når eventet kaldes, læses data fra eventet som gemmer dette i JSON-filen. Efterfølgende fortælles controlleren at viewet skal opdateres.

På Figur 2.15 ses funktionen for ViewDidLoad().

```
4 references | Ao LI, 5 days ago | 1 author, 4 changes
public override void ViewDidLoad()
{
    base.ViewDidLoad();

    TableView = new UITableView(View.Bounds)
    {
        Source = new TableSource(ProjectInfos, this),
        AllowsSelection = true,
        AllowsMultipleSelection = false
    };
}
```

Figur 2.15. Kode snip - ViewDidLoad() fra ProjectListViewController.cs

Her sættes ListViewets source til at være TableSource.

På Figur 2.16 ses funktionen for TableSource().

```

1 reference | Ao Li, 1 day ago | 1 author, 1 change
public TableSource(List<RegistrationDto> projectInfos, ProjectListViewController projectListViewController)
{
    ProjectInfos = projectInfos;
    ProjectListViewController = projectListViewController;
}

```

Figur 2.16. Kode snip - TableSource() fra TableSource.cs

Der initialiseres, hvilke værdier der skal fremvises og der er givet en reference til ProjectListViewControlleren.

På Figur 2.17 ses funktionen for RowSelected().

```

0 reference | Ao Li, 23 hours ago | 2 authors, 6 changes
public override void RowSelected(UITableView tableView, NSIndexPath indexPath)
{
    switch (indexPath.Row)
    {
        case CreateProject:
            if (ProjectListViewController.Storyboard.InstantiateViewController("CreateProjectViewController") is CreateProjectViewController createProjectVC)
            {
                ProjectListViewController.NavigationController.PushViewController(createProjectVC, true);
            }
            break;

        case CreateUser:
            if (ProjectListViewController.Storyboard.InstantiateViewController("CreateUserViewController") is CreateUserViewController createUserVC)
            {
                ProjectListViewController.NavigationController.PushViewController(createUserVC, true);
            }
            break;

        default:
            if (!(ProjectListViewController.Storyboard.InstantiateViewController("PdfViewController") is PdfViewController controller)) return;
            controller.PDFInfo = ProjectInfos[indexPath.Row];
            ProjectListViewController.NavigationController.PushViewController(controller, true);
            break;
    }
}

```

Figur 2.17. Kode snip - RowSelected() fra TableSource.cs

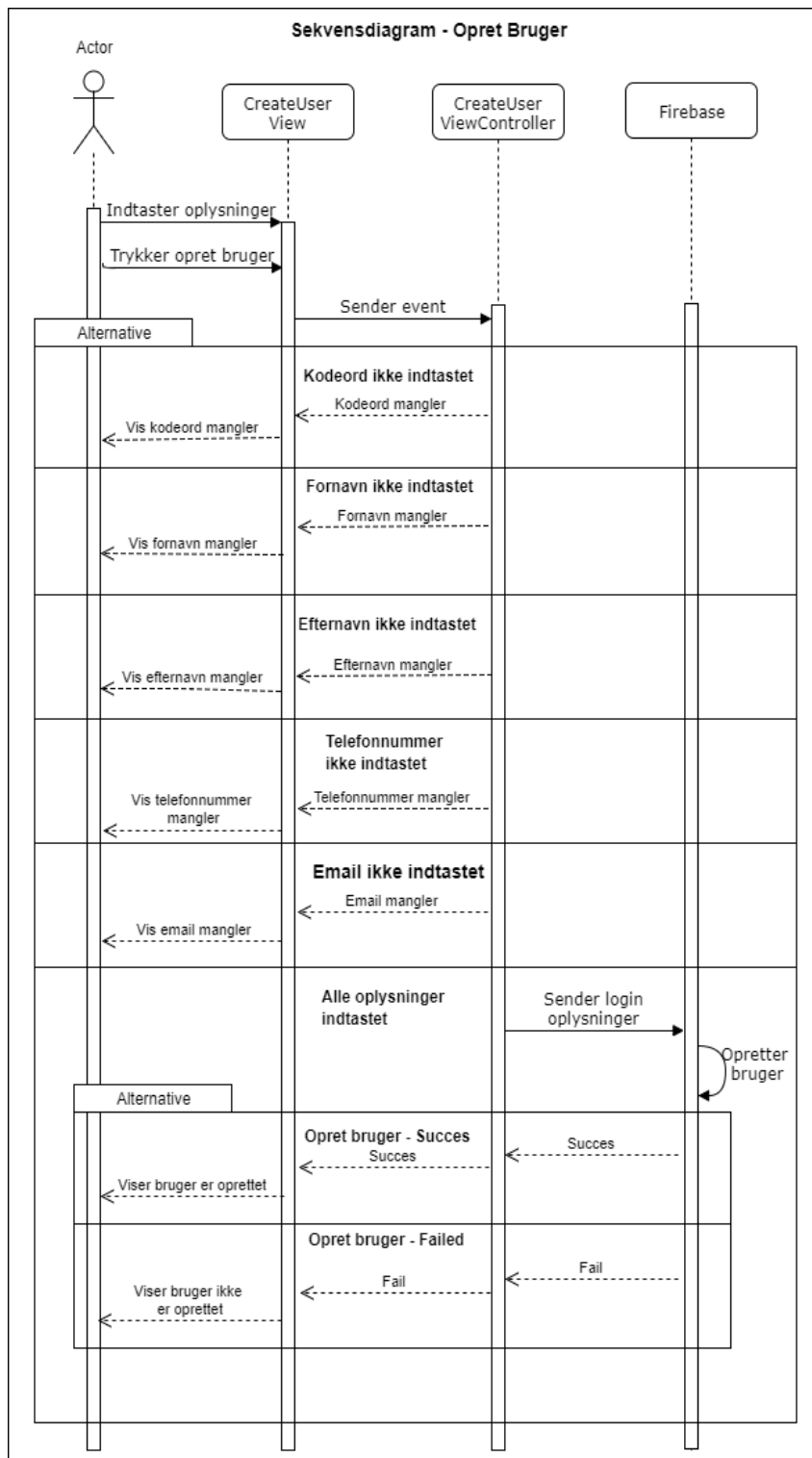
Der er en switch case, casen CreateProject, som giver brugeren mulighed for at navigere til 'Opret projekt' siden. Casen CreateUser navigere brugeren til 'Opret bruger' siden. Alternativt skal den navigere brugeren til 'PDFViewControlleren', for det valgte projekt.

2.2.4 Opret Bruger

Dette afsnit indeholder en gennemgang af den grafiske brugergrænseflade, design og implementering af 'Opret Bruger' viewet i Rambøll Tilsyn.

2.2.4.1 Design

På Figur 2.18 ses sekvensdiagrammet for 'Opret bruger' viewet til Rambøll Tilsyn.



Figur 2.18. Sekvensdiagram for 'Opret bruger' i Rambøll Tilsyn.

2.2.4.2 Grafisk brugergrænseflade

I 'Opret Bruger' viewet er der lavet felter til alt den information, som skal indtastes for en ny bruger. Se Figur 2.19

Carrier 3:58 PM 100%

< Back

RAMBOLL

Opret bruger

Brugernavn:

Kodeord:

Fornavn:

Efternavn:

Email:

Telefonnummer:

Opret bruger

Figur 2.19. 'Opret bruger' viewet som det er implementeret i Rambøll Tilsyn.

2.2.4.3 Implementering

I dette afsnit vil der blive beskrevet funktionaliteten for de vigtigste funktioner i koden tilhørende 'Opret bruger' viewet.

På Figur 2.20 ses funktionen for CreateUser().

```

2 references | Ao Li, 1 day ago | 1 author, 1 change
private void CreateUser(object sender, EventArgs e)
{
    var s = Auth.DefaultInstance;
    if (isFieldsValid())

Auth.DefaultInstance.CreateUser(EmailCreateUserTxt.Text, PasswordCreateUserTxt.Text, (user, error) => {
    if (error != null)
    {
        AuthErrorCode errorCode;
        if (IntPtr.Size == 8) // 64 bits devices
            errorCode = (AuthErrorCode)((long)error.Code);
        else // 32 bits devices
            errorCode = (AuthErrorCode)((int)error.Code);

        // Possible error codes that CreateUser method could throw
        switch (errorCode)
        {
            case AuthErrorCode.InvalidEmail:
            case AuthErrorCode.EmailAlreadyInUse:
            case AuthErrorCode.OperationNotAllowed:
            case AuthErrorCode.WeakPassword:
            default:
                Console.WriteLine("Failed to Create User");
                //TODO ERROR HANDLING LOGIC
                break;
        }
    }

else
{
    Console.WriteLine("Succesfully Created A User");
    userNode = userNode.GetChild(user.Uid);
    object[] keys = {
        "alias",
        "email",
        "firstName",
        "lastName"
    };
    object[] values = {
        PhoneCreateUserTxt.Text,
        EmailCreateUserTxt.Text,
        FirstnameCreateUserTxt.Text,
        LastnameCreateUserTxt.Text
    };
    var data = NSDictionary.FromObjectsAndKeys(values, keys, keys.Length);

    userNode.SetValue<NSDictionary>(data);
    });
}

```

Figur 2.20. Kode snip - CreateUser() fra CreateUser.cs

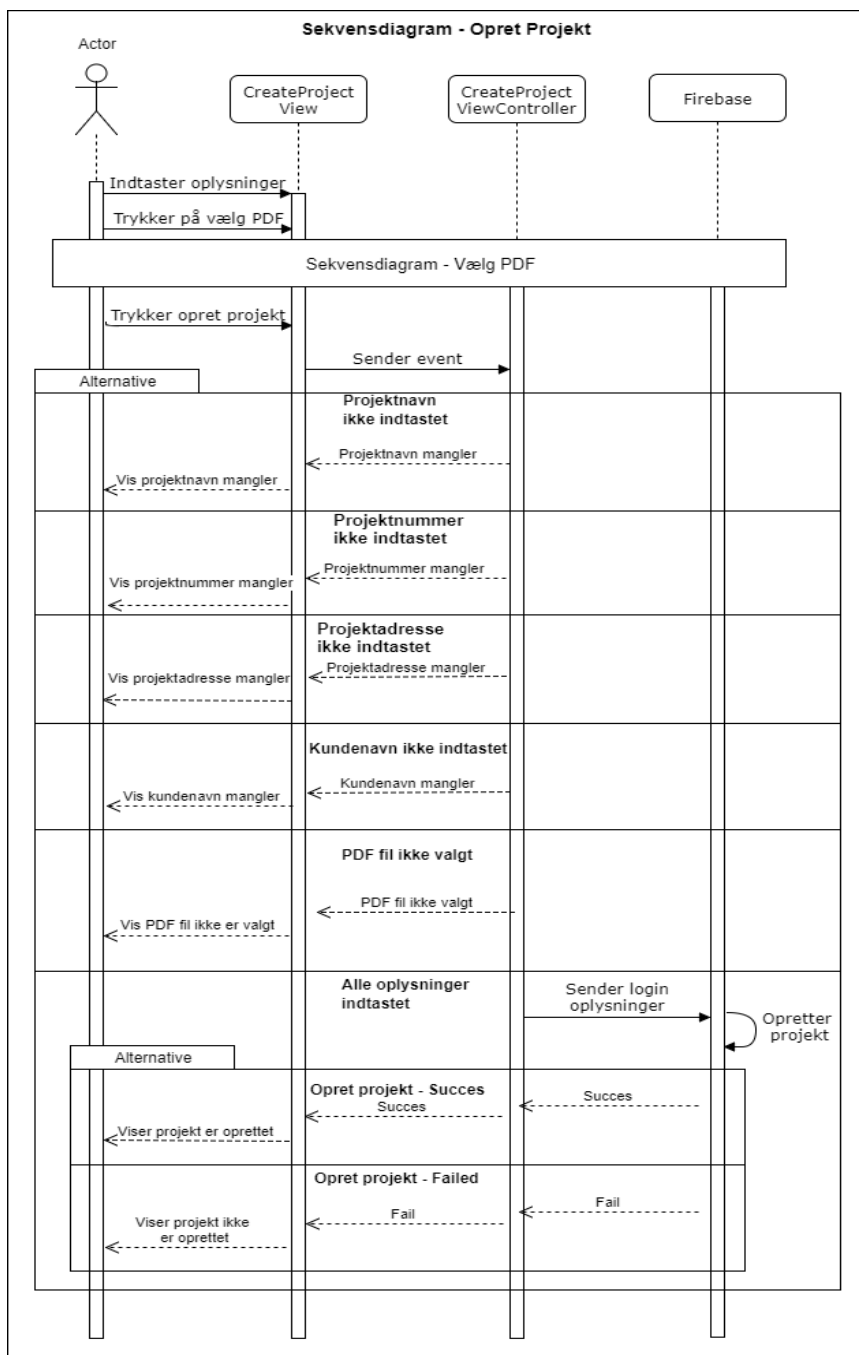
Alle felter valideres. Derefter, hvis alt er godkendt, oprettes brugeren med de input, der er kommet fra viewet. Når brugeren er blevet oprettet uden problemer, skrives informationer som ikke har noget med log ind at gøre: Fornavn, Efternavn osv. ind i databasen.

2.2.5 Opret Projekt

Dette afsnit indeholder en gennemgang af den grafiske brugergrænseflade, design og implementering af 'Opret Projekt' viewet i Rambøll Tilsyn.

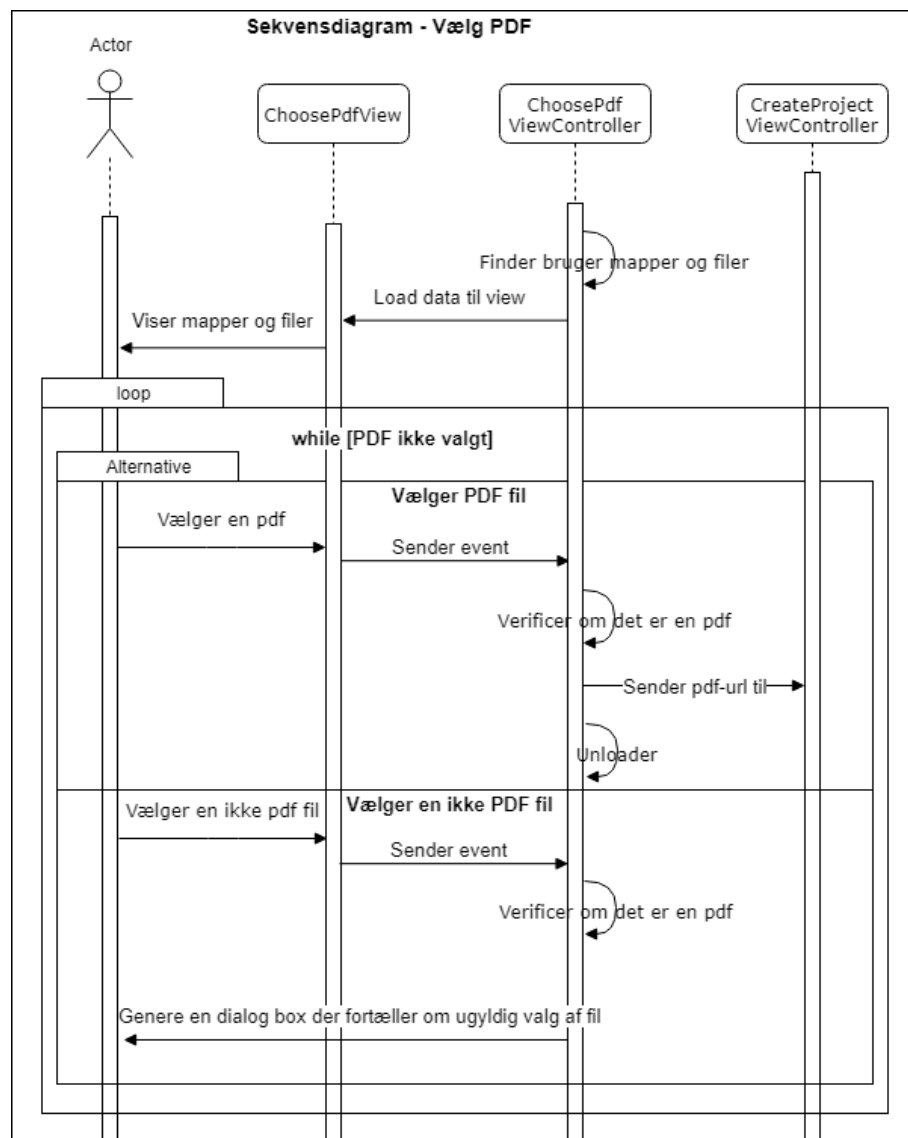
2.2.5.1 Design

På Figur 2.21 ses sekvensdiagrammet for 'Opret Bruger' viewet til Rambøll Tilsyn. Sekvensdiagrammet for 'Vælg PDF' kan ses på Figur 2.22.



Figur 2.21. Sekvensdiagram for 'Opret projekt' i Rambøll Tilsyn.

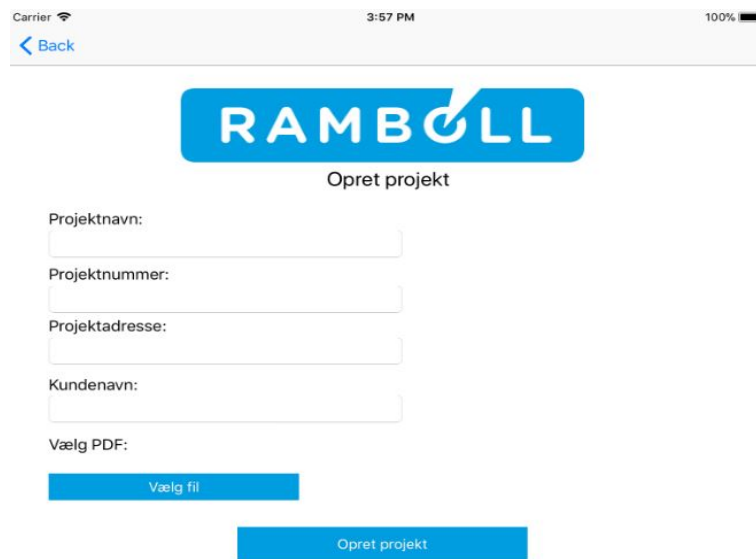
På Figur 2.22 ses sekvensdiagrammet for 'Vælg PDF' i 'Opret Projekt' i Rambøll Tilsyn.



Figur 2.22. Sekvensdiagram for 'Vælg PDF' i 'Opret Projekt' i Rambøll Tilsyn.

2.2.5.2 Grafisk brugergrænseflade

I 'Opret Projekt' viewet er der lavet felter til alle de informationer, som skal tastes ind om et nyt projekt. Se Figur 2.23



Figur 2.23. 'Opret projekt' viewet som det er implementeret i Rambøll Tilsyn.

2.2.5.3 Implementering

Denne funktionalitet er ikke implementeret.

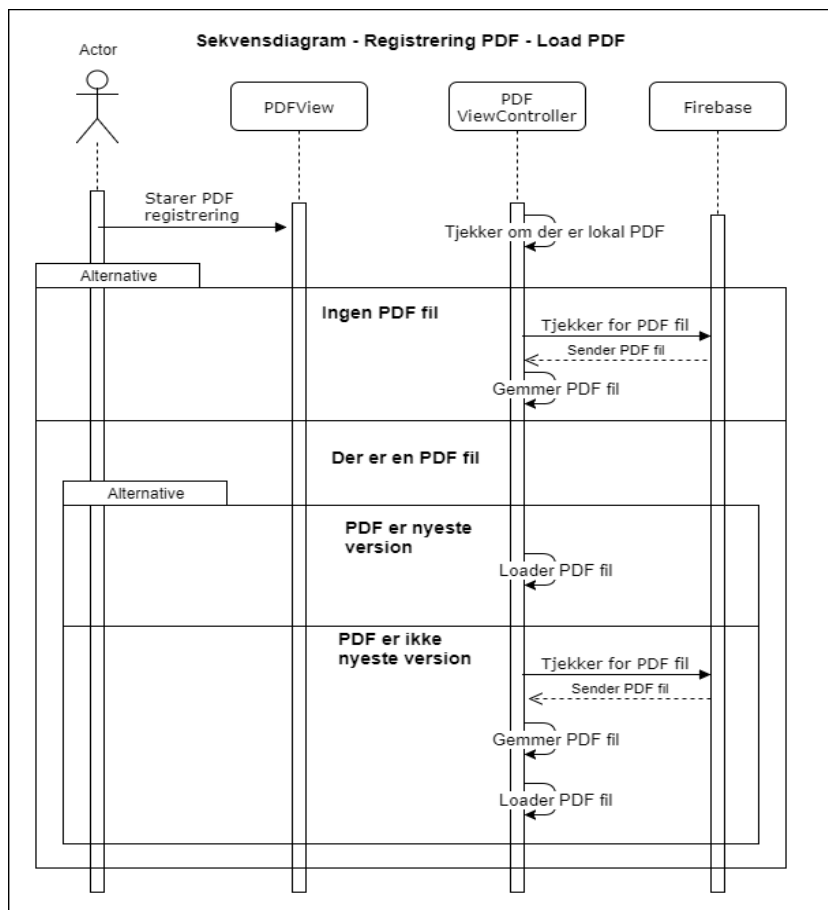
2.2.6 Registrering på PDF

Dette afsnit indeholder en gennemgang af den grafiske brugergrænseflade, design og implementering af 'Registrering på PDF' viewet i Rambøll Tilsyn.

2.2.6.1 Design

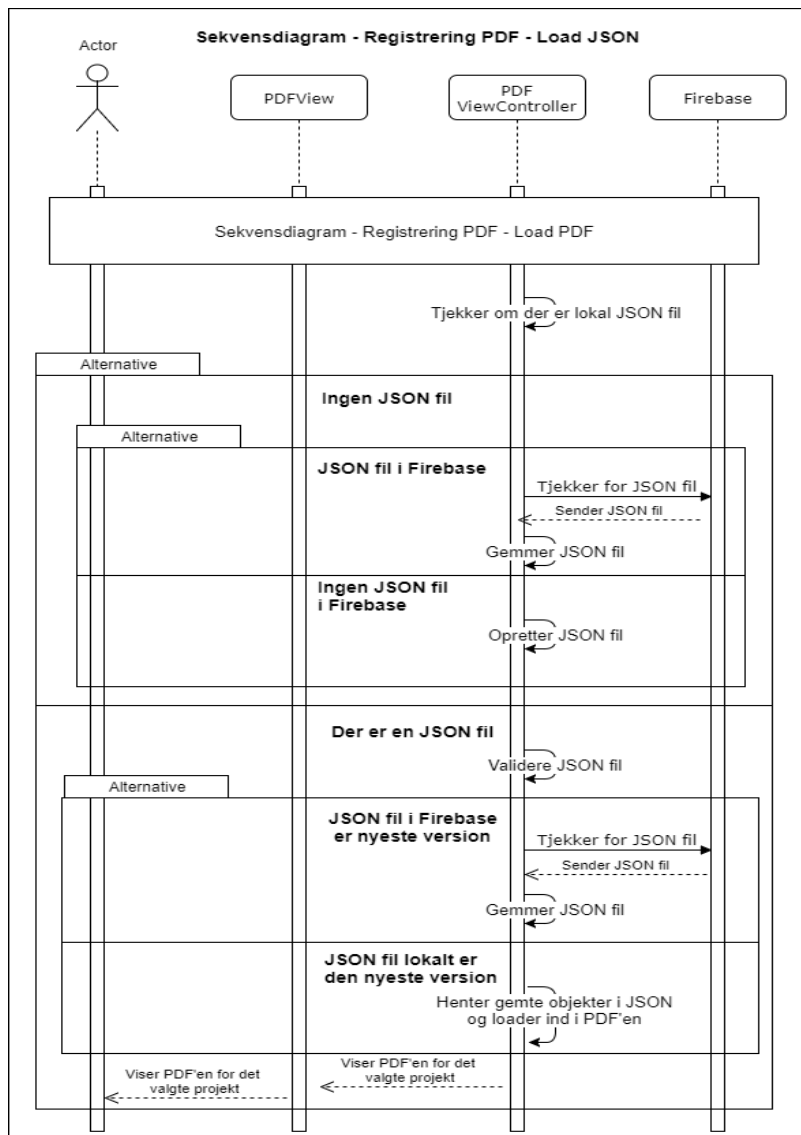
Herunder ses sekvensdiagrammerne for registrering på PDF.

Det første sekvensdiagram, som ses på Figur 2.24, viser hvordan PDF-tegningen bliver loadet ind i applikationen.



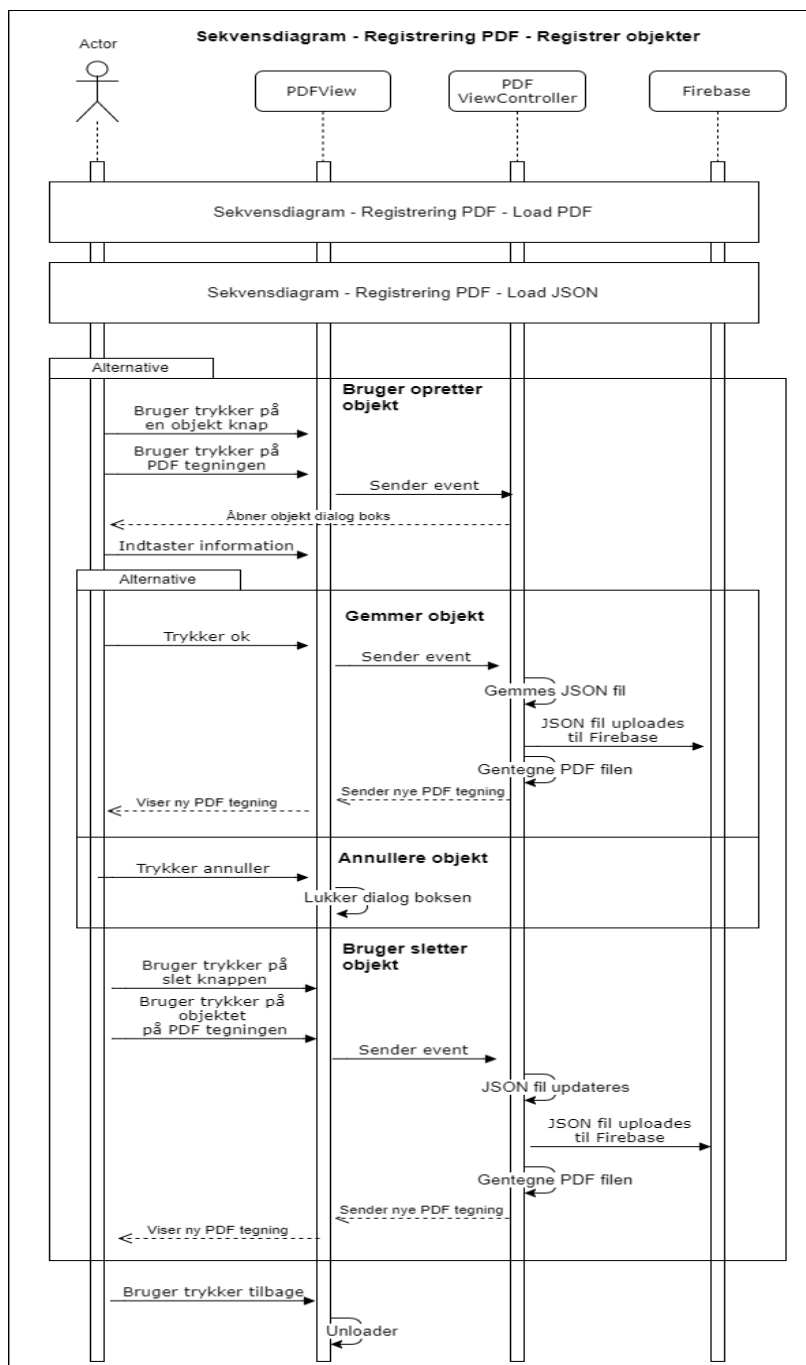
Figur 2.24. Sekvensdiagram for Registrering på PDF - Loading af PDF, i Rambøll Tilsyn.

Næste sekvensdiagram, som ses på Figur 2.25, viser hvordan JSON-filen bliver oprettet. Sekvensen med JSON sker direkte efter at PDF sekvensen er overstået.



Figur 2.25. Sekvensdiagram for Registrering på PDF - Loading af JSON, i Rambøll Tilsyn.

Sidste sekvensdiagram viser, hvordan systemet opfører sig, når brugeren interagerer med applikationen i forbindelse med oprettelse af objekter på PDF-tegningen. Denne sekvens sker i forlængelse af først Loading af PDF og Load JSON. Sekvensdiagrammet for dette kan ses på Figur 2.26.



Figur 2.26. Sekvensdiagram for Registrering på PDF - Registrer på PDF, i Rambøll Tilsyn.

2.2.6.2 Grafisk brugergrænseflade

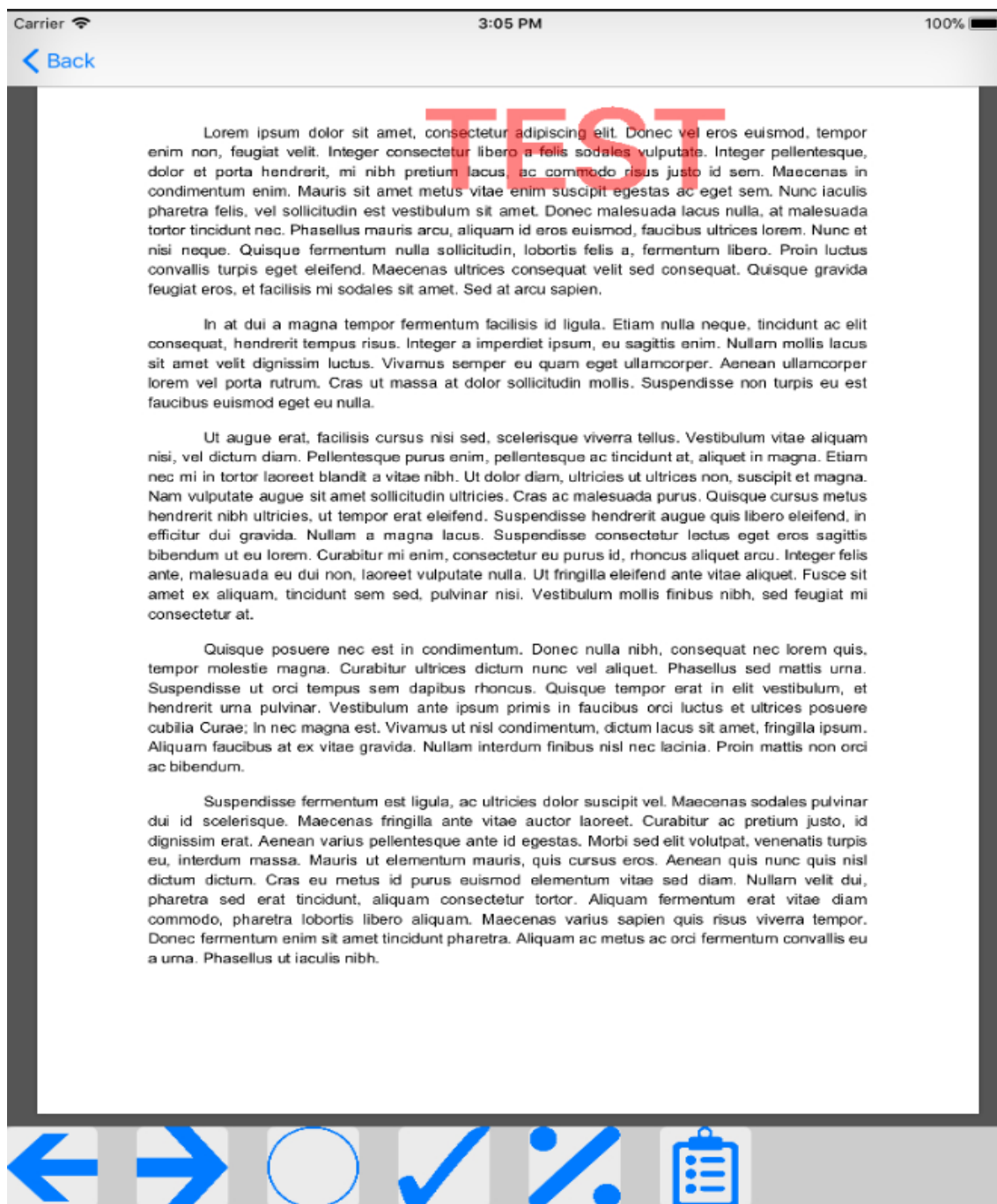
'Registrer på PDF' viewet, som ses på Figur 2.27, består af to views. Det første view viser den PDF, som er valgt til projektet.

Nedenunder ligger der et andet view, der indeholder knapper, som brugeren kan interagere med. Der er seks forskellige knapper:

Brugeren har mulighed på de første to at hoppe en side frem eller en side tilbage.

De næste tre knapper er symboler, som brugeren kan tegne med på PDF'en.

Den sidste knap er en listeknap, som giver brugeren mulighed for at afslutte sin registrering.

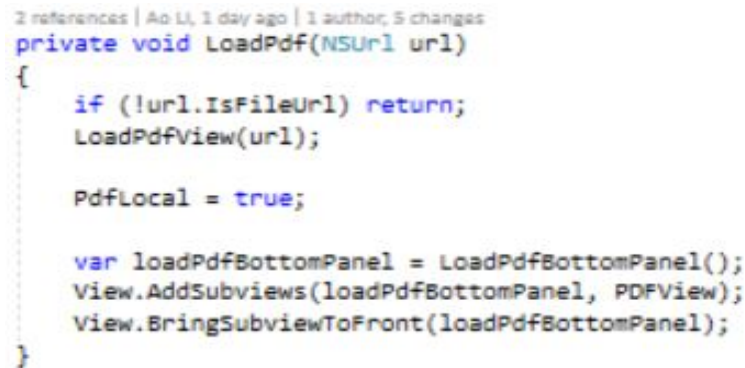


Figur 2.27. Registrer på PDF viewet, som det er implementeret i Rambøll Tilsyn.

2.2.6.3 Implementering

I dette afsnit vil der blive beskrevet funktionaliteten for de vigtigste funktioner i koden tilhørende 'Registrering på PDF' viewet.

På Figur 2.28 ses funktionen for LoadPDF().



```
2 references | Ao Li, 1 day ago | 1 author, 5 changes
private void LoadPdf(NSUrl url)
{
    if (!url.IsFileUrl) return;
    LoadPdfView(url);

    PdfLocal = true;

    var loadPdfBottomPanel = LoadPdfBottomPanel();
    View.AddSubviews(loadPdfBottomPanel, PDFView);
    View.BringSubviewToFront(loadPdfBottomPanel);
}
```

Figur 2.28. Kode snip - LoadPDF() fra PdfViewController.cs

Efter PDF'en er blevet downloadet fra Firebase gemmes den lokalt.

Det kontrolleres om url'en er en filurl. Er det en filurl loades PDF'en ind i viewet.

Når PDF'en er loadet ind i viewet, oprettes symbol knapperne.

Når både PDFViewet og PDFBottomPanel er loadet, loades disse ind i controllerens view.

På Figur 2.29 ses funktionen for LoadBottomPanel().

```
1 reference | Ao Li, 1 day ago | 1 author, 2 changes
private UIView LoadPdfBottomPanel()
{
    var panelView = new UIView
    {
        Frame = new CGRect
        {
            Location = new CGPoint(0, h - panelHeight),
            Width = w,
            Height = panelHeight,
        },
        BackgroundColor = UIColor.LightTextColor
    };
    var pw = panelView.Bounds.Width;
    var ph = panelView.Bounds.Height;
    var pLocal = panelView.Bounds.Location;

    var prePageBtn = PanelBtnFactory.GetButtonForType(PanelBtnFactory.BtnType.PrePage);
    prePageBtn.Frame = new CGRect(pLocal.X, pLocal.Y, 70, ph);
    prePageBtn.TouchUpInside += delegate
    {
        PDFView.GoToPreviousPage(this);
    };

    var nextPageBtn = PanelBtnFactory.GetButtonForType(PanelBtnFactory.BtnType.NxtPage);
    nextPageBtn.Frame = new CGRect(pLocal.X + 100, pLocal.Y, 70, ph);
    nextPageBtn.TouchUpInside += delegate...
    var addCircleBtn = PanelBtnFactory.GetButtonForType(PanelBtnFactory.BtnType.AddCircle);
    addCircleBtn.Frame = new CGRect(pLocal.X + 200, pLocal.Y, 70, ph);
    addCircleBtn.TouchUpInside += delegate...

    var addCheckMarkBtn = PanelBtnFactory.GetButtonForType(PanelBtnFactory.BtnType.AddCheckMark);
    addCheckMarkBtn.Frame = new CGRect(pLocal.X + 300, pLocal.Y, 70, ph);
    addCheckMarkBtn.TouchUpInside += delegate...
    var addMinusBtn = PanelBtnFactory.GetButtonForType(PanelBtnFactory.BtnType.AddMinus);
    addMinusBtn.Frame = new CGRect(pLocal.X + 400, pLocal.Y, 70, ph);
    addMinusBtn.TouchUpInside += delegate
    {
        Console.WriteLine("1 button pressed");
    };
    var showListBtn = PanelBtnFactory.GetButtonForType(PanelBtnFactory.BtnType.ShowList);
    showListBtn.Frame = new CGRect(pLocal.X + 500, pLocal.Y, 70, ph);

    panelView.AddSubviews(prePageBtn, nextPageBtn, addCircleBtn, addCheckMarkBtn, addMinusBtn, showListBtn);

    return panelView;
}
```

Figur 2.29. Kode snip - LoadBottomPanel() fra PdfViewController.cs

Her oprettes UIView, som indeholder knapper, der loades ind fra PanelBottomFactory. PanelBottomFactory er en klasse, som står for at oprette knapperne. Når knapperne er oprettet defineres events på de forskellige knapper.

På Figur 2.30 ses funktionen for TapOnPDF().

```

1 reference | Ao Li, 10 hours ago | 1 author, 5 changes
private void TapOnPdf(UITapGestureRecognizer obj)
{
    var position = obj.LocationInView(obj.View);
    //check which shape we are working with
    if (Shape == Shape.None) return;
    if (PDFView.CurrentPage is MarkedPdfPage markPage)
    {
        var pagePageNumber = markPage.Page.PageNumber;

        var timeNow = new DateTime().ToUniversalTime().ToString("dd-MM-yyyy HH:mm:ss");
        //Add comment when time is right
        pdfObjects.Add(new PdfObjectDto
        {
            XCord = (int)position.X,
            YCord = (int)position.Y,
            PageNo = (int)pagePageNumber,
            Shape = Shape,
            TimeStamp = timeNow
        });

        //Create a reference to the file you want to upload
        File.WriteAllText(MetallocalNSURL.Path, JsonConvert.SerializeObject(pdfObjects));

        //Upload the file to the path "images/rivers.jpg"
        StorageUploadTask uploadTask = _jsonNode.PutFile(MetallocalNSURL, null, (metadata, error) => {
            if (error != null)
            {
                Console.WriteLine("Error");
            }
            else
            {
                DatabaseReference rootNode = Database.DefaultInstance.GetRootReference();
                var key = nameof(RegistrationDto.Updated);
                //update Firebase database
                rootNode
                    .GetChild(Global.Pdf)
                    .GetChild(PDFInfo.Guid.ToString())
                    .GetChild(key)
                    .SetValue(new NSString(timeNow));
            }
        });
        Shape = Shape.None;
        _preSelectedbtn.Selected = false;
        PDFView.SetNeedsDisplay();
    }
}

```

Figur 2.30. Kode snip - TapOnPDF() fra PdfViewController.cs

TapOnPdf() er en eventhandler som håndterer når der detekteres en geusture, som f.eks. Tap. Denne håndterer også tap lokationen og opretter objektet som er valgt af brugeren. Derefter gemmes disse informationer i Firebase.

På Figur 2.31 ses funktionen for `GetClassForPage()`.

```
#region PdfDocumentDelegate
[Export("classForPage")]
0 references | Ao Li, 5 days ago | 1 author, 1 change
public ObjCRuntime.Class GetClassForPage()
{
    return new ObjCRuntime.Class(typeof(MarkedPdfPage));
}
#endregion
```

Figur 2.31. Kode snip - `GetClassForPage()` fra `PdfViewController.cs`

Funktionen `ClassForPage` overrides, som er en objektive C funktion, der normalt returnerer en standard PDF page til vores `typeof(MarkedPdfPage)`.

På Figur 2.32 ses funktionen for `Draw()`.

```
0 references | Ao Li, 5 days ago | 1 author, 1 change
public override void Draw(PdfDisplayBox box, CoreGraphics.CGContext context)
{
    // Draw original content
    base.Draw(box, context);
    var pdfDocument = this.Document;
    using (context)
    {
        // Draw watermark underlay
        UIGraphics.PushContext(context);
        context.SaveState();
        var pageBounds = this.GetBoundsForBox(box);
        context.TranslateCTM(0.0f, pageBounds.Size.Height);
        context.ScaleCTM(1.0f, -1.0f);
        //context.RotateCTM((float) (Math.PI / 4.0f));

        Console.WriteLine($"{pageBounds}");

        var attributes = new UIStringAttributes()
        {
            ForegroundColor = UIColor.FromRGBA(255, 0, 0, 125),
            Font = UIFont.BoldSystemFontOfSize(84)
        };

        var text = new NSAttributedString("TEST", attributes);

        text.DrawString(new CGPoint(250, 40));
        RenderObjects(context);
        context.RestoreState();
        UIGraphics.PopContext();
    }
}
```

Figur 2.32. Kode snip - `Draw()` fra `MarkedPdfPage.cs`

Når PDF-pagen bliver oprettet, kaldes `draw`, og denne er overridet til at tegne på PDF'en med de objekter, som bliver hentet fra JSON-filen. Disse tegnes ind på PDF'en.

På Figur 2.33 ses Enum Shape.

```
3 references | Ao Li, 5 days ago | 1 author, 1 change  
public enum Shape  
{  
    CheckMark,  
    Circle,  
    Minus  
}
```

Figur 2.33. Kode snip - Enum Shape for objects fra MarkedPdfPage.cs

Definitionen på vores symboler.

På Figur 2.34 ses MetaListJsonSingleton.

```
9 references | Ao Li, 10 hours ago | 1 author, 3 changes  
public class MetaListJsonSingleton  
{  
    private static MetaListJsonSingleton _instance;  
    private string _path = "";  
    6 references | Ao Li, 2 days ago | 1 author, 2 changes  
    public List<PdfObjectDto> PdfObjects { get; private set; }  
  
    1 reference | Ao Li, 2 days ago | 1 author, 2 changes  
    private MetaListJsonSingleton(string path)  
    {  
        _path = path;  
        var txt = File.ReadAllText(path);  
        PdfObjects = JsonConvert.DeserializeObject<List<PdfObjectDto>>(txt) ?? new List<PdfObjectDto>();  
    }  
    2 references | Ao Li, 2 days ago | 1 author, 2 changes  
    public static MetaListJsonSingleton GetInstance(string path)  
    {  
        return _instance ?? (_instance = new MetaListJsonSingleton(path));  
    }  
    1 reference | Ao Li, 2 days ago | 1 author, 1 change  
    public void Reset()  
    {  
        _instance = null;  
    }  
  
    2 references | Ao Li, 2 days ago | 1 author, 1 change  
    public static MetaListJsonSingleton TryGetInstance()  
    {  
        return _instance;  
    }  
}
```

Figur 2.34. Kode snip - MetaListJsonSingleton fra MetaListJsonSingleton.cs

Denne Singleton bliver brugt til at hente PDF objekter fra den lokale JSON-fil.

På Figur 2.35 ses funktionen for RenderObjects().

```

1 reference | Ao Li, 10 hours ago | 1 author, 4 changes
private void RenderObjects(CGRect rect, CGContext g)
{
    var listOfData = MetaListJsonSingleton.TryGetInstance();
    if (listOfData?.PdfObjects != null)
        using (g)
        {
            for (var i = 0; i < listOfData.PdfObjects.Count; i++)
            {
                var item = listOfData.PdfObjects[i];
                if (Page.PageNumber == item.PageNo)
                {
                    var w = item.XCord * ScaledWidth;
                    var h = item.YCord * ScaledHeight;
                    var size = new CGRect( w - 25, h - 25, 50, 50);
                    CGImage img = null;
                    switch (item.Shape)
                    {
                        case Shape.Circle:
                            img = UIImage.FromBundle("circle").CGImage;
                            break;
                        case Shape.CheckMark:
                            img = UIImage.FromBundle("checkmarkpdf").CGImage;
                            break;
                        case Shape.Minus:
                            img = UIImage.FromBundle("minuspdf.png").CGImage;
                            break;
                        default:
                            Console.WriteLine("Using an invalid Shape ");
                            break;

                            throw new Exception("Using an invalid Shape ");
                    }
                }
            }
            #endif
            var attributes = new NSAttributedString()
            {
                ForegroundColor = UIColor.FromRGBA(255, 0, 0, 125),
                Font = UIFont.BoldSystemFontOfSize(12)
            };
            var text1 = new NSAttributedString($"{i+1}", attributes);
            text1.DrawString(new CGPoint(w+25, h-25));
            //g.RotateCTM((float) (Math.PI / 12.0f));
            g.DrawImage(size, img);
        }
}

```

Figur 2.35. Kode snip - RenderObjects() fra MarkedPdfPage.cs

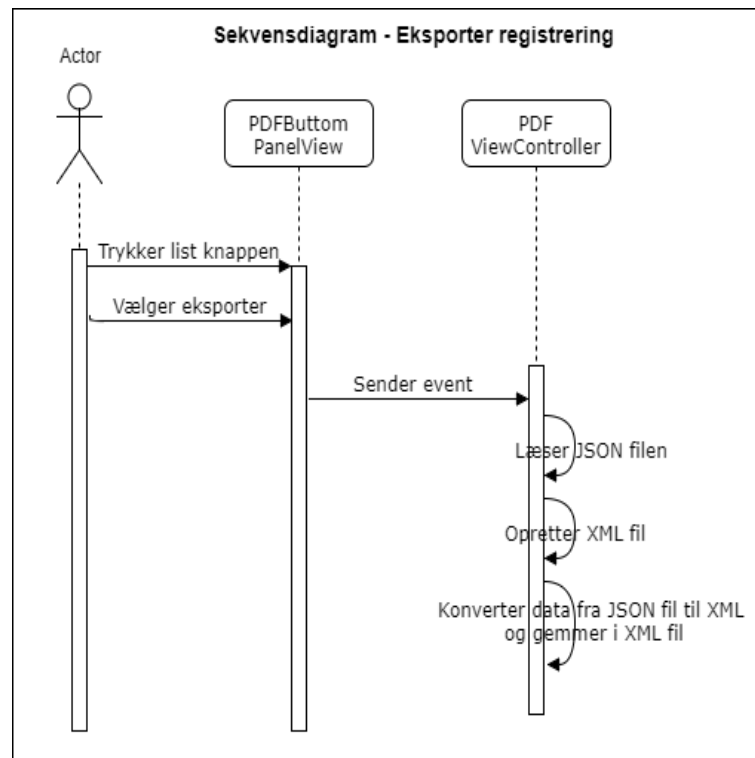
Funktionen RenderObjects() står for at tegne de objekter der tilhører PDF'en.

2.2.7 Eksportering

Dette afsnit indeholder en gennemgang af den grafiske brugergrænseflade, design og implementering af 'Export viewet' i Rambøll Tilsyn.

2.2.7.1 Design

På Figur 2.36 ses sekvensdiagrammet for 'Export' viewet til Rambøll Tilsyn.



Figur 2.36. Sekvensdiagram for Eksportering i Rambøll Tilsyn.

2.2.7.2 Grafisk brugergrænseflade

Efter at have eksporteret registreringen til en XML-fil, vil den se ud som på Figur 2.37.

Tilsynsrapport: Sletvej

Symbol	ID	Status	Dato	Side	Plads	Kommentar	Objekt id reference	Udført
Flueben	1.1	Godkendt	28-nov		1 NV			Ja
Minus	2.1	Afvist	28-nov		1 ØH			Nej
Cirkel	3.1	Fokus	29-nov		2 ØV			Ja
Flueben	1.2	Godkendt	30-nov		1 NH			Ja
Flueben	1.3	Godkendt	30-nov		2 NH			Ja
Cirkel	3.2	Fokus	01-dec		1 ØV			Nej
Minus	2.2	Afvist	01-dec		3 NV			Nej
Cirkel	3.2	Fokus	02-dec		2 ØH			Nej
Flueben	1.4	Godkendt	02-dec		3 NH		3.1	Ja

ØV = Øverste venstre side

ØH = Øverste højre side

NV = Nederste venstre side

NH = Nederste højre side

Figur 2.37. Export viewet som det vil se ud, når det er åbent i excel.

Tabellen består af linjer, hvor hvert objekt har følgende information:

Symbol: Er navnet på den type symbol, som er oprettet.

ID: Er objektets id. Dette gives ud fra, hvilken type symbol det er.

Status: Viser statussen på objektet. Status følger typen af objekt.

Dato: Viser dato for hvornår objektet er oprettet.

Side: Viser hvilken side i PDF-tegningen objektet er oprettet på.

Plads: Viser hvor på den tilhørende side at objektet er placeret.

Kommentar: Giver mulighed for at skrive kommentar til objektet.

Objekt id reference: Hvis et objekt bliver ændret fra f.eks. et minus til et flueben, vil der blive oprettet et nyt fluebens objekt, som får reference id til det minus, som er ændret.

Udført: Viser om objektet er blevet udført. F.eks. at alle minus og cirkel objekter er blevet ændret til flueben.

2.2.7.3 Implementering

Denne funktionalitet er ikke implementeret.

Ordforklaring

Forkortelse	Forklaring
Auth	Firestore Authentication
Console	Firebases web tilgang
Framework	Miljø et program laves til at kunne udføres
Hashet	En funktion som hjælper med at krypter data
Internet	En trådløs internetforbindelse som f.eks. 3G eller Wifi
JSON	JavaScript Object Notation
Keys	Nøgle
MVC	Model-View-Controller
Overrides	At overskrive en eksisterende funktion
PDF	Portable Document Format
XML	Extensible Markup Language

Litteratur

- [1] UML.org. Uml. URL <http://www.uml.org/>. Last Visited d. 24.11.2017.
- [2] Apple Inc. Uiviewcontroller - uikit | apple developer documentation, . URL <https://developer.apple.com/documentation/uikit/uiviewcontroller>. Last Visited d. 16.12.2017.
- [3] Google. Firebase analytics. URL <https://firebase.google.com/docs/analytics/>. Last Visited d. 15.12.2017.
- [4] Techopedia Inc. What is a console (con)? - definition from techopedia, . URL <https://www.techopedia.com/definition/1906/console-con>. Last Visited d. 15.12.2017.
- [5] Firebase. Firebase authentication, . URL <https://firebase.google.com/products/auth/>. Last Visited d. 18.10.2017.
- [6] Techopedia Inc. What is a root node? - definition from techopedia, . URL <https://www.techopedia.com/definition/21839/root-node>. Last Visited d. 07.12.2017.
- [7] Vangie Beal. What is guid? webopedia definition. URL <https://www.webopedia.com/TERM/G/GUID.html>. Last Visited d. 07.12.2017.
- [8] Techopedia Inc. What is key-value pair (kvp)? - definition from techopedia, . URL <https://www.techopedia.com/definition/13645/key-value-pair-kvp>. Last Visited d. 07.12.2017.
- [9] Firebase. Firebase storage, . URL <https://firebase.google.com/products/storage/>. Last Visited d. 19.10.2017.
- [10] Xamarin. Forms. URL <https://www.xamarin.com/forms>. Last Visited d. 15.11.2017.
- [11] TutorialTeacher. Mvc. URL <http://www.tutorialsteacher.com/mvc/mvc-architecture>. Last Visited d. 04.12.2017.
- [12] Techopedia Inc. What is an event handler in c#? - definition from techopedia, . URL <https://www.techopedia.com/definition/3220/event-handler-c>. Last Visited d. 15.12.2017.
- [13] Techopedia Inc. What is a memory leak? - definition from techopedia, . URL <https://www.techopedia.com/definition/3838/memory-leak>. Last Visited d. 15.12.2017.