

**ZmartWallet Rapport**

**Computer Science 5.th semester Dissertation by;**

**Gruppemedlemmer:** Halil Ibrahim Batbayli & Mark Nitz Christoffersen

**Periode:** 20-04-2015-10-06-2015

**Supervisor:** Anders Børjesson

**Uddannelsested:** Erhvervsakademi Sjælland, Roskilde

Forord

Rapporten De skal til at læse omhandler Wallet App genren inden for SmartPhones, om hvilke typer Wallet Apps der findes på SmartPhone markedet, om hvordan vores projekt, ZmartWallet, adskiller sig fra disse, hvordan vi opnår en transaktion mellem bruger og butik, samt validering af bruger gennem Microsoft Azure Active Directory.

Vi vil gennemgå en Agil udviklings proces ved brug af Scrum og Extreme Programming, og hvordan denne proces har ført os frem til to Applikationer, ZmartWallet og ZmartScanner, og via kode eksempler demonstrere hvordan vi har præsteret at udvikle disse Android Applikationer, som kan udgives på Google Play Store.

Selvom vi arbejder efter Extreme Programming, gør vi også brug af Unified Proces metodologien ved, at vi udfører en markedsanalyse og S.W.O.T analyse, og vil derfor afvige fra Agil Udvikling, da vi følte det var vigtigt først at undersøge markedet for forbrugsmønstre blandt den danske befolkning, og hvilke typer Wallet Apps der allerede findes på markedet for at vurdere om der et reelt forretningsgrundlag for ZmartWallet.

Table of Contents

[1 Indledning 6](#_Toc421623707)

[2 Problemformulering 7](#_Toc421623708)

[3 Problemstillinger 7](#_Toc421623709)

[4 Metoder & Aktiviteter 7](#_Toc421623710)

[4.1 Udviklingsmetode og værktøjer til projektstyring 8](#_Toc421623711)

[4.1.1 Agile Project Management 8](#_Toc421623712)

[4.2 Agile Metoder 9](#_Toc421623713)

[4.3 Selv Organiserende Hold 9](#_Toc421623714)

[4.4 Planning Game 9](#_Toc421623715)

[4.5 Pair Programming 9](#_Toc421623716)

[4.6 Acceptance Tests 10](#_Toc421623717)

[4.7 Small Releases 10](#_Toc421623718)

[4.8 Fælles Kodestandard 10](#_Toc421623719)

[4.9 Simpel Design 10](#_Toc421623720)

[4.10 Refactoring 10](#_Toc421623721)

[4.11 Test Driven Development 10](#_Toc421623722)

[4.12 Version Kontrol / Fælles ejerskab af koden 10](#_Toc421623723)

[4.13 Kontinuerlig Integration 10](#_Toc421623724)

[4.13.1 Scrum 11](#_Toc421623725)

[4.13.2 Afvigelser ved brug af Agile udvikling 12](#_Toc421623726)

[4.14 Værktøjer 12](#_Toc421623727)

[4.14.1 ScrumWise 12](#_Toc421623728)

[4.14.2 Visual Studio udvikler IDE & Xamarin 12](#_Toc421623729)

[5 Mål, Minimumskrav, levering 13](#_Toc421623730)

[6 Projekt Tidsplan 14](#_Toc421623731)

[7 Forretnings Analyse 14](#_Toc421623732)

[7.1 Baggrund og analyse af markedet 14](#_Toc421623733)

[7.2 Danske forbrugeres benyttelse af Mobiltelefon 15](#_Toc421623734)

[7.3 Kundesegment 17](#_Toc421623735)

[7.4 Økonomi 17](#_Toc421623736)

[7.5 Konkurrentanalyse 18](#_Toc421623737)

[7.5.1 Undersøgelse af eksisterende Wallet applikationer og deres funtionalitet 18](#_Toc421623738)

[7.5.2 Fidme 18](#_Toc421623739)

[7.5.3 Konklusion på konkurrentanalyse 19](#_Toc421623740)

[7.6 S.W.O.T. Analyse 19](#_Toc421623741)

[7.6.1 Konlusion på s.w.o.t analyse 19](#_Toc421623742)

[8 Planlægning af udvikling 20](#_Toc421623743)

[8.1.1 Overblik 20](#_Toc421623744)

[9 Produktudvikling 21](#_Toc421623745)

[9.1 Shared Vision 21](#_Toc421623746)

[9.1.1 Produkt Position 21](#_Toc421623747)

[9.1.2 Stakeholder Description (product owner?) 21](#_Toc421623748)

[9.2 Overblik over udviklingsprocess 22](#_Toc421623749)

[9.3 Product Backlog 22](#_Toc421623750)

[9.4 Release Backlog 23](#_Toc421623751)

[Sprint Backlog 23](#_Toc421623752)

[9.5 Scrum board 25](#_Toc421623753)

[9.5.1 Figur Scrum Board 25](#_Toc421623754)

[10 Sprint 1 25](#_Toc421623755)

[10.1 User story 1 – Oprettelse af bruger 25](#_Toc421623756)

[10.2 User story 2 - Login af Bruger 25](#_Toc421623757)

[10.3 Active Directory 26](#_Toc421623758)

[10.4 Implementering af User Story 1+2 – Oprettelse & Login af bruger 26](#_Toc421623759)

[10.5 Evaluering af Sprint 1 33](#_Toc421623760)

[10.5.1 Acceptance Test 33](#_Toc421623761)

[10.6 User Story 1: Opret Bruger Test 33](#_Toc421623762)

[10.7 User Story 2: Login Bruger Test 33](#_Toc421623763)

[10.7.1 Sprint Retrospective 33](#_Toc421623764)

[11 Sprint 2 34](#_Toc421623765)

[11.1 User Story 3: Hent Bruger Information 34](#_Toc421623766)

[11.2 User story 4: Gem bruger information 34](#_Toc421623767)

[11.3 User Story 5: Vis Bruger Information 34](#_Toc421623768)

[11.4 Implementering af User Story 3: Hent bruger information 34](#_Toc421623769)

[11.5 Tasks : 34](#_Toc421623770)

[11.6 Implementering af User Story 4: Gem bruger information 37](#_Toc421623771)

[11.7 Tasks: 37](#_Toc421623772)

[11.8 Implementering af User Story 5: Vis Brugerinformation 38](#_Toc421623773)

[11.9 Evaluering af Sprint 2 40](#_Toc421623774)

[11.9.1 41](#_Toc421623775)

[11.10 User Story 3+4+5: Hent, Gem, Vis Bruger Oplysninger 41](#_Toc421623776)

[11.10.1 Sprint Retrospective 41](#_Toc421623777)

[12 Sprint 3 41](#_Toc421623778)

[12.1 User story 6: Hent Kort 41](#_Toc421623779)

[12.2 User Story 7: Gem kort 41](#_Toc421623780)

[12.3 User story 8: Vis kort 41](#_Toc421623781)

[12.4 User Story 9: Aktiver Kort 41](#_Toc421623782)

[12.5 Implementering af User Story 6: Hent Kort 42](#_Toc421623783)

[12.6 Tasks 42](#_Toc421623784)

[12.7 Implementering af User Story 7: Gem Kort 42](#_Toc421623785)

[12.8 Tasks 42](#_Toc421623786)

[12.9 Implementering af User story 8: Vis kort 44](#_Toc421623787)

[12.10 Tasks 44](#_Toc421623788)

[12.11 44](#_Toc421623789)

[12.12 User story 9: Aktiver kort 47](#_Toc421623790)

[12.13 Tasks: 47](#_Toc421623791)

[12.14 Evaluering af Sprint 3 51](#_Toc421623792)

[12.15 Acceptance Test 51](#_Toc421623793)

[12.16 User Story 6+7+8: Hent, Gem, Vis Kort 51](#_Toc421623794)

[12.17 User story 9: Aktiver kort 51](#_Toc421623795)

[12.18 Sprint Retrospective 51](#_Toc421623796)

[13 Sprint 4 52](#_Toc421623797)

[13.1 User Story 10: Vis Stempelkort 52](#_Toc421623798)

[13.2 User Story 11: Vis QR Kode / Opdater stempler 52](#_Toc421623799)

[13.3 User Story 12 – ZmartScanner App 52](#_Toc421623800)

[13.4 Implementering af User Story 10: Vis Stempelkort 52](#_Toc421623801)

[13.5 Tasks: 52](#_Toc421623802)

[13.6 Implementering af User Story 11: Vis QR Kode / Opdater stempler 55](#_Toc421623803)

[13.7 Tasks: 55](#_Toc421623804)

[13.8 Implementering af User Story 12 : ZmartScanner App 58](#_Toc421623805)

[13.9 Tasks: 58](#_Toc421623806)

[13.10 Zxing.Net.Mobile 58](#_Toc421623807)

[13.10.1 ScanActivity 60](#_Toc421623808)

[13.10.2 60](#_Toc421623809)

[13.10.3 ScannerClient 61](#_Toc421623810)

[13.11 Evaluering af Sprint 4 63](#_Toc421623811)

[13.11.1 Acceptance Test 63](#_Toc421623812)

[13.12 User Story 10: Vis Stempelkort 63](#_Toc421623813)

[13.13 User Story 11: Vis QR Kode / Opdater Stempler 63](#_Toc421623814)

[13.14 User Story 12: Scanner App 63](#_Toc421623815)

[13.14.1 Sprint Retrospective 64](#_Toc421623816)

[14 Konklusion 64](#_Toc421623817)

[14.1.1 Spørgsmål 1 64](#_Toc421623818)

[14.1.2 Spørgsmål 2 64](#_Toc421623819)

[14.1.3 Spørgsmål 3 64](#_Toc421623820)

[Spørgsmål 4 64](#_Toc421623821)

[15 Perspektivering 65](#_Toc421623822)

# Indledning

Formålet med denne rapport vil være at udforske mulighederne i Wallet app genren og udvikle en Wallet applikation til Android platformen, som har til formål at fungere som et alternativ opbevaring og brugen af plastikkort til fx. Bonusordninger, loyalitetskort osv. Frem for alt skal applikationen kunne konkurrere på parametrene bekvemmelighed, fleksibilitet og pris.

For at gøre dette projekt så realistisk som muligt har vi taget kontakt til ReSolve-IT som er en enkeltmands ejet virksomhed med fast tilknyttede udviklere. der udvikler mobil applikationer til primært IOS platformen. ReSolve-IT har et ønske om at få udviklet en Wallet applikation til Android platformen.

Grundet en ekstrem udvikling, er smartphones telefoner blevet en fast del af vores hverdag.

Vi benytter vores smartphones til lt fra kommunikation, shopping og underholdning, samt organisering af vores hverdag. I løbet af de seneste par år, er det i visse tilfælde blevet teknisk muligt, at benytte sin smartphone som betalingsmiddel ved brug af applikationer apps, som et alternativ til de mere traditionelle betalingsmetoder så som, at betale med kontanter, plastickort, samt bankoverførelser via netbank. Hvis vi lige ser bort fra MobilePay, er det dog endnu ikke helt muligt at betale med sin mobil telefon i en fysisk butik i Danmark.

De fleste ejere af en computer er bekendt med E-handel, handel via internettet, det er nemt og tidsbesvarende. Flere og flere benytter nu efterhånden nu også deres mobiltelefon eller tablet til at handle med.

Ifølge DIBS(Dansk Internet Betalings System) årlige rapport fra 2014 om dansk E-handel**[[1]](#footnote-1)**, vil næste store skridt inden for E-handel teknologien vil ske i form af betalingsløsninger via. Mobiltelefonen, når butikker og de mobile betalingsløsninger kan leve op til kunders behov og ønsker.

# Problemformulering

Resolve-It har et ønske om, at få udviklet en wallet applikation kaldet ZmartWallet til Android platformen. ZmartWallet er en Wallet applikation, som har til formål at fungere som et alternativ opbevaring og brugen af plastikkort til fx. bonusordninger og lignende for butikkens kunder. ZmartWallet indeholder en digital version af plastikkortene og sørger derfor for at brugeren har alle sine bonuskort, loyalitetskort, stemplekort mv. samlet ét sted.

Derudover er det hensigten at det digitale kort har designerede QR-stregkoder som kan scannes af den fysiske butik som udsteder kortet ved hjælp af den tilhørende Applikation: ZmartWallet Scanner.

# Problemstillinger

En af de åbenlyse udfordringer er, at udvikle en wallet applikation som er konkurrencedygtig på brugervenlighed og funktionalitet i forhold til eksisterende applikationer, disse parametre er også vigtige i forhold til slutbrugeren. Fungere applikationen ikke optimalt vil slutbrugeren ganske enkelt ikke benytte den, dette er set ud fra kortholders- og butiks synspunkt.

Vi har ud fra ovenstående beskrivelse defineret en række spørgsmål, som har til formål hjælpe os med at løse de problemstillinger der kan være forbundet med udviklingen af ZmartWallet.

1. Hvilke typer Wallet applikationer findes allerede på markedet?
2. Hvordan kan ZmartWallet differentiere sig fra andre Wallet applikationer på markedet?
3. Hvordan kan man registrere optjening og betaling / transaktioner mellem brugere og butikker?
4. Hvordan kan man validere man bruger og sikrer man brugeroplysninger imod svindel?

# Metoder & Aktiviteter

Vi har fundet frem til en række metoder og aktiviteter som skal være medvirkende til, løse problem stillingerne, der er defineret i vores problemformulering. Disse metoder og aktiviteter har til formål, at bidrage til udarbejdelsen af vores research, organisering og produktivitet i forhold til projektstyring og udviklingsprocessen,

For at opnå en vis form for baggrundviden finder vi det vigtigt, at undersøge forbrugsmønstre og trends inden for handel via. smartphones og applikationer og identificering af eksisterende konkurrenter.

Ved at foretage research på disse områder forventer vi også at kunne få et indblik i om der er et reelt forretningsgrundlag for udviklingen af ZmartWallet. ReSolve-IT har naturligvis deres ønsker og minimumskrav, men der kan være forskellige scenarier som de muligvis ikke har tænkt over.

## Udviklingsmetode og værktøjer til projektstyring

Følgende udviklingsmetode og værktøjer vil blive benyttet til, strukturere og sikre et optimalt projekt forløb.

### Agile Project Management

**Til håndtering af projektet har vi besluttet os for at benytte ideen om Agile management som guideline . Agile projekt management har til formål, at optimere projektforløbet, og gøre os mere produktive i forhold til, hvad vi ellers ville være i stand til ved brug af de mere traditionelle metoder såsom ”Waterfall” og UP(Unified process), som kræver meget dokumentation og planlægning før man begynder at producere produktet, hvilket er velegnet til større og længerevarende projekter[[2]](#footnote-2).**

Agile Project Management kan bruges til IT- og ikke IT baserede projekter.

Ikke IT basere projekter kan f.eks. være:

* Hvis en virksomhed skal flytte lokation.
* Omstrukturering af virksomheden.
* Ændring og eller optimering af arbejdsprocesser som man f.eks. kender det fra Lean.

Fælles for dem alle projekttyper eller virksomheder er, at man er i stand til at levere et ”produkt” inde for et relativ kort periode. Da tilgangen i Agile er så fleksibel som den nu er har man mulighed for at ændre eller tilføje til projektet. Sidst nævnte betragter vi som en stor fordel, når man som udvikler skal levere en produkt til en kunde, som muligvis kan have tilføjelser eller ønsker om ændringer til det endelig produkt. Vi kan endda som selv have forslag til ændringer af produktet.

**The Agile Manifesto**

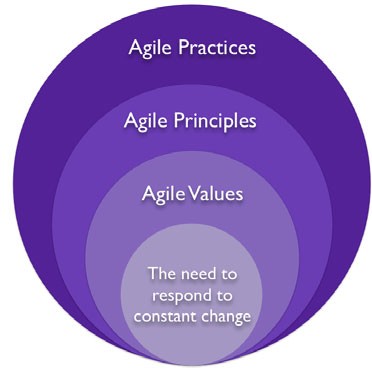
The Agile Manifesto’s oprindelse er fra 2001, hvor ide og koncept er udviklet af 17 personer med forskellige baggrunde inden for IT branchen.

Agile manifesto er en erklæring der er baseret på følgende 4 værdier og 12 principper[[3]](#footnote-3):

**Kendetegnene ved Agile Project Management**

* Udvikler kan levere et funktionelt produkt inde for en relativ kort periode til kunde.
* Ingen- eller begrænset udarbejdelse af dokumentation.
* Daglig kommunikation mellem udviklere højner produktivitet og minimerer fejl.
* Involvering af kunde i løbet at udviklingsproces, kunde har derfor indflydelse i løbet af udviklingsproces og på det endelige slutprodukt.
* Et produkt ved hvert Sprint, som er fuldt funktionelt.

## Agile Metoder

Vi har bevæget os ind på Agile værdier og principper, men for at de 2 begreber hænger sammen er vi nødt til nu at gøre brug af agile metoder, som manifesterer eller implementerer agile værdier og principper.

Billedet **[[4]](#footnote-4)** illustrerer hvordan de tre begreber, principper – værdier og metoder hænger sammen.

I løbet af denne rapport og system udvikling til Android ved brug af Xamarin, vil vi benytte os af en del agile metoder, vi vil forsøge at argumentere for hvorfor vi har valgt at bruge de praktikker vi har brugt, eller valgt ikke at bruge, selvom vi arbejder agilt. Vi har valgt at følge Extreme Programming**[[5]](#footnote-5)**, og udvælge en del metoder som vi mener vil være gældende for os, under denne udviklings proces og rapport udfærdigelse.

## Selv Organiserende Hold

Vi er 2 individer til at udarbejde denne rapport, samt udvikling til android, der samarbejder med en back end udvikler samt en IOS udvikler og vores Product Owner, Sinus fra Resolve-It, som ikke er en del af denne rapport. Vi har et ønske om at arbejde i et agilt miljø, og udvikle ZmartWallet App’en, og har i den forbindelse valgt både at arbejde hjemmefra, på skolen, samt i et af Resolve-It’s kontorer. Da vi er et lille hold, er det denne agile metode et godt redskab for os at benytte og tage til os.

Det hjælper os til at arbejde tæt sammen, og stille krav til hinanden fra starten, lære hinanden styrker og svagheder, og evt. trække på hinandens erfaring og lære af hinanden.

## Planning Game

Denne metode har vi ikke benyttet, da metoden forudsætter at man har en vis kendskab eller erfaring inden for Android udvikling for at kunne foretage en realistisk estimering. Vi derfor baseret vores estimering i form af vores Sprints.

## Pair Programming

Vil vi benytte os meget af, da vi begge er under en lærings proces, og vil ved hjælp af denne metode hjælpe hinanden til at kode bedre, eller prøve at levere kode af godt kvalitet, samtidig vil det også være nødvendigt at bruge denne metode, da vi kun er 2 udviklere og arbejder i Xamarin, med Trial version af denne IDE, så giver det god mening at bruge Pair Programming, da der er en del ”nyt” vi skal lære og læse op på, og kan hjælpe hinanden til at nå frem til løsninger mens den ene læser op på noget og den anden koder.

## Acceptance Tests

Dette vil vi også gøre brug af efter hvert sprint, og hver fredag når vi holder Sprint Review med Product Owneren (Sinus), sørge for at Acceptance tests bliver foretaget og feedback modtaget af disse.

## Small Releases

Denne metode vil vi gøre brug af, ved at vi efter hvert sprint vil levere den del af koden vi er færdig med, som virker, og kan bruges af bruger, dog i vores projekt vil disse releases ikke være Slut Bruger releases, men releases, hvori de user stories vi har udført virker efter Acceptance Test kriterier.

## Fælles Kodestandard

Denne metode vil vi også følge, da det er et krav at Applikationen bliver udviklet i Xamarin Cross-Platform IDE, og vi er derfor nødt til at følge et bestemt design pattern, MVVM, for at opnå Cross-Platform mulighed, og vi er derfor også bundet til at udvikle i C# sprog.

Vi vil dog kun udvikle Applikationen til Android platform, og Cross-Platform versionen vil først blive aktuelt i en fremtidig iteration.

Vi vil forsøge at opretholde kode der er let at forstå, ved at give metoderne vi opretter og kalder, navne der indikerer hvad metoden vil gøre, og afholde og fra at lave kommentarer i koden.

## Simpel Design

Ved at følge MVVM pattern, og sørge for at holde koden struktureret, vil vi opnå denne metode, bruger grænse overfladen vil vi også forsøge at holde simpelt.

## Refactoring

Vi vil forsøge at holde os til at koden er struktureret, og hvis vi ændrer i kodens interne struktur (refactoring), ikke ændrer på kodens eksterne adfærd, altså hvis vi foretager en lille ændring, vil det ikke ændre på hvordan koden opfører sig eller måden hvorpå den generer et resultat ikke ændrer sig.

## Test Driven Development

Er en agil metode som vi vil gøre brug af ved at afholde acceptance tets sammen med Resolve-It efter hvert sprint, men vi udfører ingen unit tests fordi backend koden allerede er udviklet og klar til at blive kaldt, samt da vi bruger Xamarin til udvikling**[[6]](#footnote-6)**, og benytter en Trial version, så koster Test Integration udvidelsen for meget til, at vi ikke vil benytte os af denne funktion til Xamarin.

## Version Kontrol / Fælles ejerskab af koden

Vi vil ikke benytte os af denne metode, da vi kun er 2 udviklere der følger Pair Programming, så der vil kun være behov for en udviklings maskine, og ingen grund til version kontrol, da vi vil skiftes til at udvikle på denne maskine.

## Kontinuerlig Integration

Da vi følger Pair Programming i hele processen, er denne metode overflødig for vores vedkommende.

### Scrum

The Agile Manifesto er som sagt en erklæring af 4 værdier og tolv principper, men omfatter ikke ”steps” som skal hjælpe virksomheden og den enkelte udvikler med at opnå en Agile udviklingsproces. Derfor vil vi benytte os af Scrum som er en flere Agil metode man kan benytte til, at opnå Agile udvikling og projektstyring. Scrum er er inspireret af Lean filosofien som bruges til optimering af arbejdsprocesser, der basalt set tager udgangspunkt i produktionsopgaver, altså samlebåndsarbejde.**[[7]](#footnote-7)**

I løbet af vores 4. semester på uddannelsen, har vi lært og benyttet Scrum og er blevet bekendt med dets mange fordele i forbindelse med software udvikling.

Ved at følge Scrum, udvikler man sit produkt i små byggeblokke i en iterativ og trinvis stigende proces. I Scrum har man mindre udvikler teams, der har daglig ansigt-til-ansigt kommunikation og daglige møde, hvilket skaber bedre intern kommunikation, minimere fejlkommunikation og højner produktiviteten som i sidste ende gavner udviklingsprocessen i et projektet.

**Scrum Roller:**

Scrum gør brug af roller og har 3 typer:

**Product Owner:**

I vores tilfælde er product owner Sinus Frank som er ejer af ReSolveIT,. Product Owner er den rolle med en Vision som ønskes realiseret, har det endelig ansvar og bør være til rådighed.

**Scrum Master:**

Vi vil i forbindelse med vores udviklingsproces begge have denne rollen, som Scrum master, da vi kun er to personer.

Scrum Master er den rolle som handler i Product Owners interesse og fungere som en slags projektleder. Scrum Master rollen styrer ikke udvikler teamet, men har til opgave at finde løsninger på eventuelle problemer. På den måder hjælper Scrum master sit udvikler team med at bevare produktiviteten og som er medvirkende med, man opnår de mål der er sat de pågældende sprints .

**Teamet**

Er selve udvikler teamet, som har ansvaret for at dele og håndtere arbejdsmængden imellem sig. Et Scrum udvikler team består ca. af 3-7 personer. Et team omfatter typisk, programmører, Ui designere, system arkitekter og analytikere. I løbet af projektet kommer vi til, at have de fleste ad disse ”hatte” på os i samarbejde med de eksisterende udviklere hos ReSolveIT.

### Afvigelser ved brug af Agile udvikling

Vi er på nuværende tidspunkt allerede bekendt med, at vi kommer til at bryde ”reglerne” om Agile Scrum, da vi først og fremmest skal skrive rapport i forbindelse med vores afgangseksamen, der vil derfor ikke udelukkende være fokus på udvikling i forbindelse med projektet.

Derudover har vi været selektive og kritiske omkring, hvilke Agile metoder der er relevante i forhold til vores projekt . Vi kommer desuden til afvige fra Agile og Scrum i forbindelse med udarbejdelsen af en Business analyse og i forbindelse med dokumentation af den implementering vi vil foretage os i forbindelse med udviklingen af ZmartWallet.

## Værktøjer

Dette afsnit vil beskrive de værktøjer, der vil blive benyttet til projektstyring og udvikling.

### ScrumWise

Vi har valgt at benytte Scrumwise**[[8]](#footnote-8)** som er et online scrum værktøj. Med scrumwise har vi mulighed for at håndtere Agile planlægning altså projektstyring, og version control m.m.

Scrumwise er ikke et værktøj vi har arbejdet med tidligere, men det virker enormt brugervenligt og er i stand til at opfylde vores behov i forbindelse med de principper vi vil i Agile Project Management og Scrum. Scrumwise har et virtuelt scrum board, derudover har vi mulighed for, at oprette de user stories vi kommer til, at arbejde med og inddele dem i sprints, hvilket gør det nemt og overskueligt at håndtere sin Project styring

### Visual Studio udvikler IDE & Xamarin

ResolveIT har fra projektet’s start af haft et krav om, at vi benyttede os af Xamarin i forbindelse med udviklingen af ZmartWallet. Xamarin giver mulighed for Cross plaform udvikling af mobil applikationer. Det der er det er kendetegnende ved Xamarin er, at giver udviklere mulighed for, at udvikle applikationer til f.eks android i C# sproget. Man har i forbindelse med udviklingen mulighed for at benytte Xamarin eller man kan benytte Visual Studio som understøtter Xamarin. Vi har derfor valgt at benytte Visual Studio, som er det IDE (Integrated developer enviroment), vi har bedst kendskab til og vil benytte i forbindelse med vores udviklingsprocess.

Da vores udvikling kommer til at foregå i Xamarin og Visual studio giver det også mulighed for genbrug af virksomhedens forretningslogik, hvis de på et senere tidspunkt skulle have et ønske om at udvikle applikationer til andre platforme. Dette kunne eksempelvis være, at udvikle produkter til Windows Phone eller en desktopløsning såsom WPF(Windows Presentation Foundation).

# Mål, Minimumskrav, levering

Følgende mål i forbindelse med projektet kan betragtes som opgaver skal løses for, at være i stand til at levere en fuldt ud funktionel applikation.

Levering i forbindelse med projektet vil omfatte en applikation i form af ZmartWallet og vil ikke omfatte server side eller database relateret, da dette allerede er sat op af virksomheden. Derudover vil vi levere endnu en applikation i form af en scanner applikation, som butikken skal benytte til, scanne brugerens QR kode på det digitale kort. I forbindelse med udviklingsprocessen vil vi ***jf. Afsnit Metode & aktiviteter***, benytte os af Agile udvikling og definere disse opgaver i form af User stories, som vil omfatte interaktion med server og database, funktionalitet og brugergrænsefladen

**Mål for projektet:**

* Indsamling af funktionelle og Non-Funktionelle krav.
* Baggrunds research af trends indenfor brugen af mobil E-Handel, analyse af virksomhed og konkurrenter i form af eksisterende Wallet Applikationer.
* Implementering af backend for at møde minimumskrav fra ReSolveIT
* Implementering af brugergrænseflade – aktiviteter, menuer, kanapper osv.
* Præsentation og evaluering af applikation hos ResolveIT

**MinimumsKrav:**

Funtionelle krav

* Oprettelse af bruger.
* Login funktion og autentificering af bruger via. Microsoft Azure – Active Directory.
* Brugerprofil.
* Hente liste af digitale kort.
* Aktivere kort.
* QR kode til registrering af bonuspoint og stempler.

Non-funktionelle krav

* Response Time

Transaktioner og opdateringer mellem ZmartWallet og server bør have en hurtig reponse tid ved login, opdatering af liste over kort, registrering af optjente og brugte point m.m.

* Security

Bruger skal autentificeres via. email og password til identificering, der giver brugeren adagang til ZmartWallet og brugerprofil.

* Scalability

ZmartWallet applikationen bør være fleksibel i forhold til eventuelle tilføjelser af nye funktioner.

* Usability

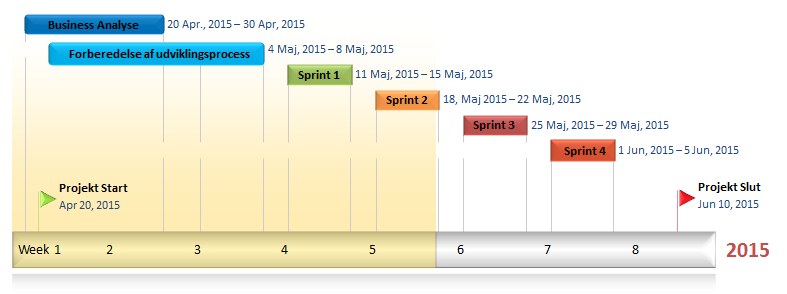
ZmartWallet skal være nemt og bekvemt at benytte for den almene smartphone bruger og bør ikke kræve nogen videre teknisk viden.

**Levering & Evaluering:**

* Ved hvert endt Sprint vil produktet blive præsenteret til ReSolve-IT i form af en demonstration. Er ReSolve-IT tilfredse med det der præsenteret vil udviklingsprocessen forsætte til vi når den sidste fase af hele projektforløbet, som gerne skulle resultere i tilfredsstillende slutprodukter.

# Projekt Tidsplan

Vi er blevet enige om en række milepæle, der skal hjælpe os med, at overholde den fastsatte tidsramme. I den forbindelse har vi fastsat nogle datoer og Sprints, som primært er forbundet med udvikling i form af opgaver og delprodukter, der i sidste ende skal resultere i et slutprodukt. [[9]](#footnote-9)

****

# Forretnings Analyse

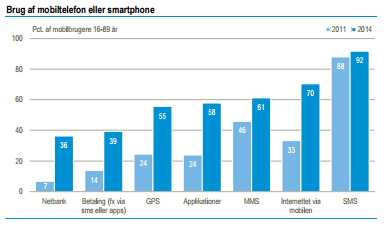
I dette afsnit vil der blive fortaget en forretningsanalyse, der har til formål, at give et indblik i den danske befolknings brug af Mobiltelefoner og applikationer. Efterfølgende vil eventuelle konkurrenter blive identificeret, for derefter at vurdere ReSolve-IT’s og ZmartWallets forretningsgrundlag og konkurrencemæssige position.

## Baggrund og analyse af markedet

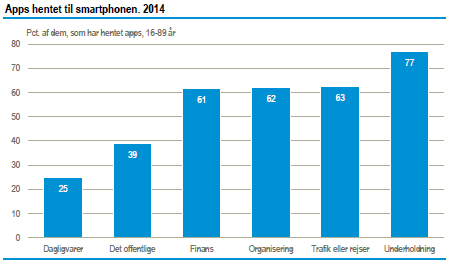
Følgende statistik og analyse af markedet og trends har til formål, at belyse de danske forbrugers købsmønstre ved benyttelse af mobiltelefon, for derefter at vurdere om, der er et reelt forretningsgrundlag for at udvikle en ZmartWallet app. Vi valgt at tage udgangspunkt i de rapporter der udkommer, hvert år fra Dansk Statistik og DIBS, da deres indsamling og analyse af data virker som de mest troværdige og giver et grundigt indblik i dansk e-handel og It-anvendelse blandt den danske befolkning.

## Danske forbrugeres benyttelse af Mobiltelefon

Følgende statistik er udarbejdet af Danmarks statistik. Resultaterne er baseret på 5.457 besvarelser blandt den danske befolkning i alderen 16-74 år via. telefoninterviews og webskemaer. Heraf er antallet af gennemførte besvarelser blandt de 16-74 årige 4.231[[10]](#footnote-10).



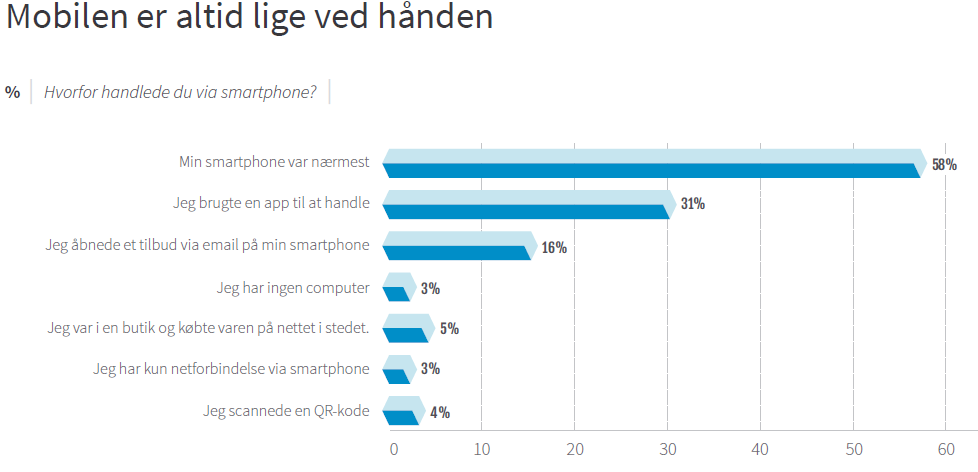
Ifølge rapporten fra Danmarks statistik for 2014, anvender 39% af mobilbrugere deres mobiltelefon og smartphonesmed fx. Sms eller applikationer, hvor elektronisk betaling også foregår via. mobil netbank.



Med udgangspunkt i samme rapport fra Danmarks Statistik kan vi se applikationer der tilhører kategorien dagligvarer kun hentes af 25%, dog er der 62% der har hentet app’s i kategorien organisering og 63% i kategorien trafik og rejser. Disse kategorier betragter vi som de mest relevante i forhold til ZmartWallet og i forbindelse med muligheden for opbevaring og brug af bonuskort, loyalitetskort mv.

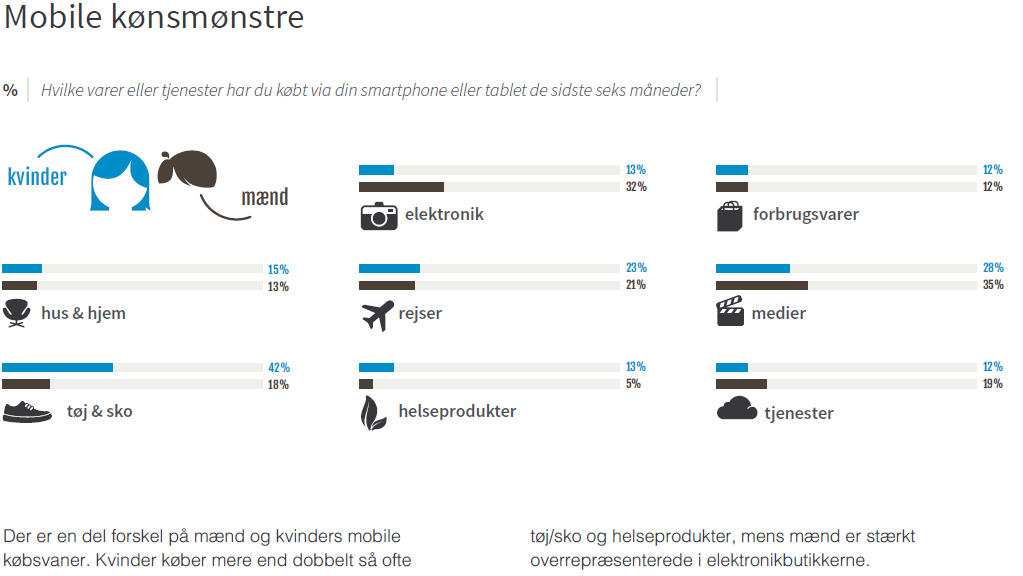
Ud fra dette disse oplysninger har vi dannet os et overordnet indblik i, hvordan de danske forbrugere benytter deres mobiltelefon og smartphone i deres dagligdag.

Da vi gerne vil have nogle flere oplysninger om markedet og forbrugsmønstre har vi benyttet os af DIBS årlige rapport fra tredje kvartal 2014, der er udarbejdet af analyseinstituttet YouGov, som omhandler Dansk E-Handel i Danmark, med særlig fokus på mobil handel**[[11]](#footnote-11)** .

Det næste vi skal kigge på er årsagen til, at forbrugeren har benyttet sin mobil telefon frem for sin computer eller tablet.

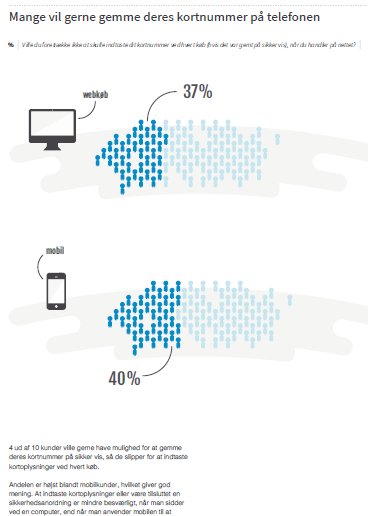
Det interessante ved dette diagram er årsagen i forbindelse med et køb, hvor bekvemmelighed er den primære årsag til brugen af en smartphone, nu viser diagrammet dog ikke hvordan købet er foretaget dvs. Via. en browser eller en applikation.

Vi vil nu kigge nærmere på om der er forskelle på forbrug i forhold til køn**[[12]](#footnote-12)**.



Da ResolveIT selv har udtrykt at ZmartWallets primære målgruppe er kvinder i alderen 20-40. Mener vi at det er interessant, at se kvinder har et større forbrug og hvilke varer eller tjenester de har købt.

Udfra diagrammet kan vi se, at i forhold til kvinder er der et større forbrug hos kvinder i kategorierne tøj og sko, helseprodukter og rejser. I kategorierne hus og hjem og forbrugsvarer er tallene dog relativt lige. På baggrund af disse tal kan ReSolveIT få et indblik i, hvilke butikker de kan forsøge, at indgå aftaler med

Da vi på nuværende tidspunkt har dannet os et overblik over den danske befolknings-forbrugsmønstre ved benyttelse af mobiltelefon,

vil det nu være interessant at se om forbrugerne stoler nok på sikkerhedsnivauet i smartphones og applikationer til, at de vil gemme kortoplysninger på deres mobiltelefon.

Ifølge statistikken fra DIBS**[[13]](#footnote-13)** vil 4 ud af 10 kunder gerne have mulighed for at gemme deres kortnummer på sikker vis, så de slipper for at indtaste kortoplysninger ved hvert køb.

Nu skal huske at have fokus på, at ZmartWallet applikationen omhandler bonuskort, loyalitetskort

og stempelkort, men på baggrund af disse tal kan vi se, der er en vis procentdel af mobilbruger, der ønsker at gemme deres kort oplysninger digitalt på telefonen og benytte dem til at handle med.

Derudover betragter vi det også disse tal som et salgsargument over for en butik, at de kan fremme deres salg, hvis de har en aftale med ZmartWallet aftale. Et par eksempler kunne være kaffebarer,

Tank stationer eller Matas, der kunne øge deres butikssalg.

## Kundesegment

ReSolve-IT’s potentielle kunder og brugere er som udgangspunkt danske butikker og primært kvinder i alderen 20-40 år der er ejere af plastikkort som f.eks. bonusordninger, loyalitetskort m.m. Resolve-IT har en målsætning om, at ZmartWallet skal være gratis at downloade og benytte via. Google Play store. Virksomheder der har en interesse i ZmartWallet konceptet vil have mulighed for, at tilslutte sig til denne uden beregning inde for det første år.

Et andet kundesegment kunne være festivaller såsom Roskilde og Skanderborg festival. Ifølge B.T forsøgte Roskilde festival i 2013 at indføre et elektronisk armbånd**[[14]](#footnote-14)**, som betalingsmiddel for dermed, at slippe for bekymringen om tab eller tyveri. Roskilde festival har dog droppet tiltaget, da armbåndet ikke ungerede optimalt og overvejer at benytte en mobilløsning i stedet.

## Økonomi

Den økonomiske strategi som ReSolveIT har lagt er som følgende;

ZmartWallet er gratis for brugeren at downloade.

ZmartScanner som er tiltænkt butikkerne, er der hvor indtjeningen skal komme fra:

* For at tiltrække butikkerne er første år gratis.
* Månedlig abonnement for butikken.
* Fast gebyr pr. Transaktion, eksempelvis 0,50 kr. pr. Scanning af QR kode.
* Fast gebyr pr. Oprettelse af bruger kort i ZmartWallet.

## Konkurrentanalyse

Følgende konkurrentanalyse som til formål at analysere ZmartWallet og reSolveIT’s potentielle konkurrenters nuværende og fremtidige konkurrenceevene, samt deres strategiske konkurrenceposition.

Ved udarbejdelse af en konkurrentanalyse bør vi have et klart billede af ZmartWallet’s potentielle konkurrenter og markedet inden for Wallet aplikationer.

Den samlede konkurrentanalyse er en kombination af ReSolveIT’s egen analyse af eksisterende konkurrenter på markedet og en S.W.O.T analyse som vi selv har udarbejdet.

Det er på baggrund af analysen også en mulighed for, at vi bliver nød til at ændre strategi, hvis der f.eks eksisterer en konkurrent som tilbyder en Wallet applkation med idegrundlag eller funtionalitet.

### Undersøgelse af eksisterende Wallet applikationer og deres funtionalitet

ReSolveIT har udarbejdet deres egen analyse**[[15]](#footnote-15)**, som er baseret på deres konkurrenters markedsandele i form af antal downloads, altså hvor populære er disse applikationer blandt brugerne. Derudover er der lavet en sammenligning på funktionaliteten, som vi betragter som den vigtigste faktor.

### Fidme

Vi har selv været inde og kigge på Fidme som ResolveIT betragter som deres pt. Største konkurrent.

Fidme er er fransk produceret og udbydes af virksomheden Snapp**[[16]](#footnote-16)** til alle plaformer Apple App store Google Play, Windows Store etc. Med 1-5 millioner downloads må betragtes som en af de største udbydere inden for wallet app kategorien. Fidme giver brugeren mulighed for, at få adgang til mange loyalitetsprogrammer, som har Stregkode og QR kode til scanning. Brugeren har også mulighed for at gemme sine fysiske kort i app’en.

Fidme er i realiteten identisk med ZmartWallet, bortset fra at Fidme har mange flere funtionaliteter, såsom modtagelse af tilbud, nyheder fra favorit butikker, stempelkort, som kan scannes i butikker, hvor præmier kan indløses. Derudover understøtter også andre wallet apps som Apple Passbook, Microsoft Wallet og Samsung Wallet, dog adskiller ZmartWallet sig fra Fidme, da ZmartWallet er interaktivt mellem fysik butik og kunde.

### Konklusion på konkurrentanalyse

Der findes diverse applikationer som tilhører genren wallets, men hovedparten af disse applikationer, har dog dog primært fokus på brugerens mulighed for, at gemme password-beskyttelse og personlige regnskaber, samt applikationer til at opbevaring af sine kreditkort m.m.

Hovedparten af de relativt få applikationer, der hævder at de har loyalitetskort, giver dog kun brugeren mulighed for, at indtaste og/eller scanne stregkode eller medlemsnummer.

Derudover skal brugeren selv skrive kortets navn og evt. uploade logo m.m. Altså en slags mulighed for at ”digitalisere” sine kort, som man i bund og grund kunne have taget et billede af med sit kamera og lagt i mapper i sit galleri.

Den reelle konkurrent er fransk-producerede **FidMe** og en potentiel konkurrent i dansk-producerede **Just Collect**, som dog slår udelukkende på stempelkort. Sidstnævnte kunne i stedet være en samarbejdspartner.

## S.W.O.T. Analyse

Ved at benytte S.W.O.T. modellen undersøger vi om ZmartWallet har et potentiale, ved, at analysere reSolveIT som virksomhed og ZmartWallet’s position i forhold til det eksisterende marked.

|  |  |
| --- | --- |
| Styrker: | Svagheder: |
| * Programmering og database kendskab. * Benyttelse af Cross-platform udvikling. | * Få udviklere hos ReSolve-IT * Få interessenter og etablerede kunder til ZmartWallet. * Mangel på investorer og kapital. |
| Muligheder: | **Trusler:** |
| * Nye interessenter. * Stigende vækst i brug af applikationer. * Stabil vækst inden for mobil teknologi. * Samarbejde med eksisterende dansk konkurrent. | * Eksisterende Wallet applikationer på markedet. |

### Konlusion på s.w.o.t analyse

Vores konlusion på s.w.o.t analysen er, at ReSolveIT er meget afhængig af investorer og deres medarbejdere grundet deres beskedne størrelse som virksomhed. Der er dog bestem et forretningsgrundlag i forhold til ZmartWallet og dermed et grundlag for, at påbegynde udvikling af ZmartWallet applikationen.

På baggrund af den research vi har foretaget er ideen om den digitale ”Wallet” bestemt et koncept, som mobil app industrien betragter som et koncept som vil blive en del af hverdagen blandt smartphone brugere. Det lader dog ikke til, at ”Wallet” konceptet forhold er alvor er kommet til danmark endnu.

Succes kriteriet for ZmartWallet er dog, at man får skaffet en eller flere etableret et samarbejde med et vist antal butikker, som har bonusordninger og loyalitetskort. Derudover er der slutbrugeren som skal kunne se ideen med ”Wallet” konceptet.

# Planlægning af udvikling

Dette kapitel vil være koncentreret om udvikling af ZmartWallet applikationen ved brug af Agile Scrum.

### Overblik

For at beskrivelse har til formål at skabe et overodnet overblik over applikationen og de funktionaliteter der er tiltænkt ZmartWallet.

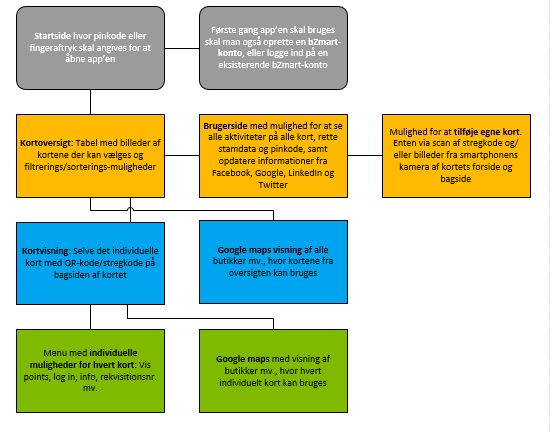
For at kunne benytte ZmartWallet applikationen skal man som bruger oprette sig med et login og en brugerprofil , man har efterfølgende mulighed for, at få vist og redigere sine profiloplysninger.

Når brugeren er logget ind vil man blive præsenteret med liste af forskellige ikke-aktiverede korttyper, som hentes fra en server. Brugeren skal efterfølgende aktivere de kort som han eller hun ønsker at benytte. Findes det pågældende kort ikke i databasen vil brugeren have mulighed for, at tage et billede af det fysiske kort eller scanne stregkoden på bagsiden af kortet og uploade det til sin samling af kort.

Lokation af butikker ved hjælp af Google maps er en feature, som vil blive implementeret i en senere iteration.

Det er som sagt tanken, at man i en senere version af ZmartWallet skal kunne understøtte brug af kreditkort, så man har muligheden for at betale med valuta.

Følgende applikations diagram illustrerer det påtænkte user action flow.



# Produktudvikling

I dette afsnit vil vi påbegynde udviklingen af ZmartWallet og ZmartScanner

## Shared Vision

Shared vision dokumentet **[[17]](#footnote-17)**har til formål at belyse hvad den overordnede vision er for kunde og udvikler.

### Produkt Position

|  |  |
| --- | --- |
| For | ReSolveIT |
| Who | Ønsker at få udviklet en Wallet application til Android plaformen |
| The (product name) | ZmartWallet |
| That | Giver bruger mulighed for, at opbevare sine fysiske bonus og loyalitetskort. |
| Unlike | ZmartWallet er interaktivt mellem fysisk butik og bruger af ZmartWallet App, den opnåede bonus/gevinst kan ”udbetales” med det samme. |
| Our product | Giver brugeren mulighed for, at digitale kort kan benyttes i fysiske butikker via. ZmartWallet. Derudover har ZmartWallet stempelkort ordning. |

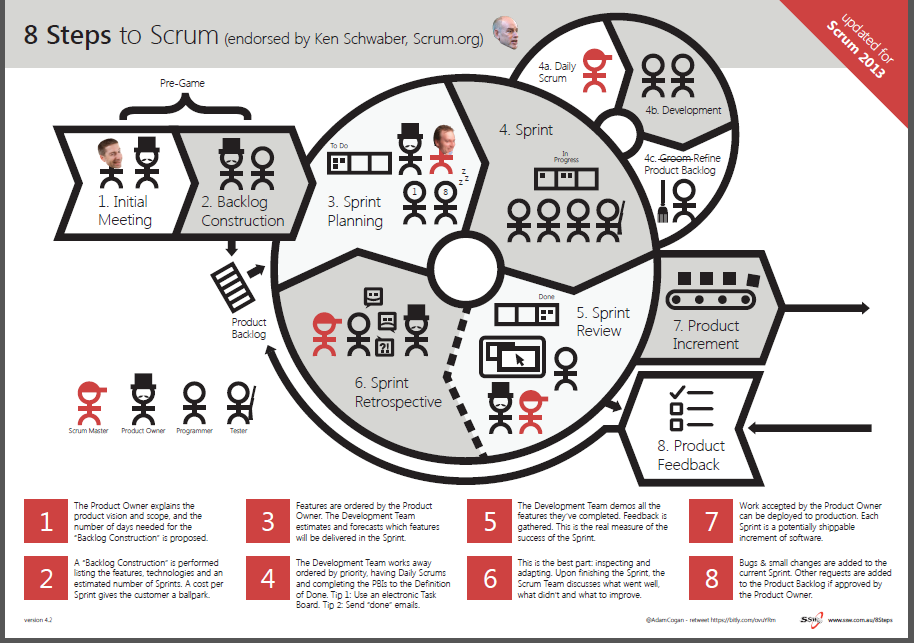
### Stakeholder Description (product owner?)

|  |  |  |
| --- | --- | --- |
| Name | Description | Responsibilities |
| Customer | Sinus Frank ejer af ReSolveIT | Product Owner vil følge produktudviklingen løbende og give feedback. |

I samarbejde med kunden, som i dette tilfælde er resolveIT, udarbejder vi en prioritering af de features der har højest business value for virksomheden. Dette er features der som minimum skal implementeres for at ZmartWallet kan fungere.

## Overblik over udviklingsprocess

Nedenstående illustration viser den process der er forbundet med Scrum.



## Product Backlog

Ved at benytte ScrumWise Online har vi kunnet lave vores product back og oprette alle user stories

Formen på en User Story er typisk defineret som følgende:

***As a (role) I want to be able to…..*** In order to... **[[18]](#footnote-18)** Ved hjælp af dette standard format er vi I stand til at identificere hvem, hvad og hvor den pågældende user story er relevant i forhold til brugeren.

Alle disse user stories og tasks er efterfølgende blevet samlet i product backlog altså et katalog af de features vi vil implementere. Da vi arbejder ud fra principperne Agile og Scrum vil vores projekt og udviklings­process blive inddelt i iterationer, hvilket kaldes Sprints i Agile Scrum. Dette har vi også også mulighed for i Visual Studio Onlne. Vi har ved hjælp af templates mulighed for, at tage hvert enkelt user story og inddele dem i forskellige Sprint alt efter hvilken priotet de har fået. Vi bør derfor have en god mulighed for at styre vores projekt- og tidstyring.

Nedenstående billede er en præsentation af vores produkt backlog som er udarbejdet i ScrumWise.



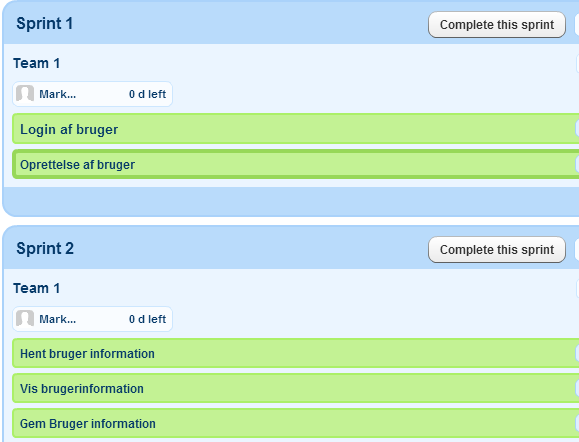
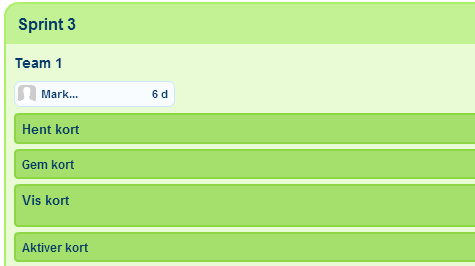
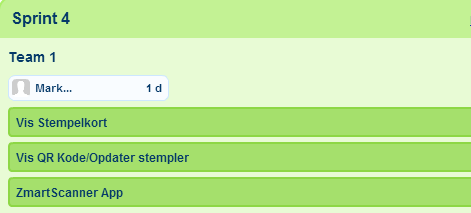
## Release Backlog

Release backlog indeholder de User stories der ikke vil indgå i sprint og vil derfor ikke blive implementeret i løbet af dette projekt.[[19]](#footnote-19)

Sprint Backlog

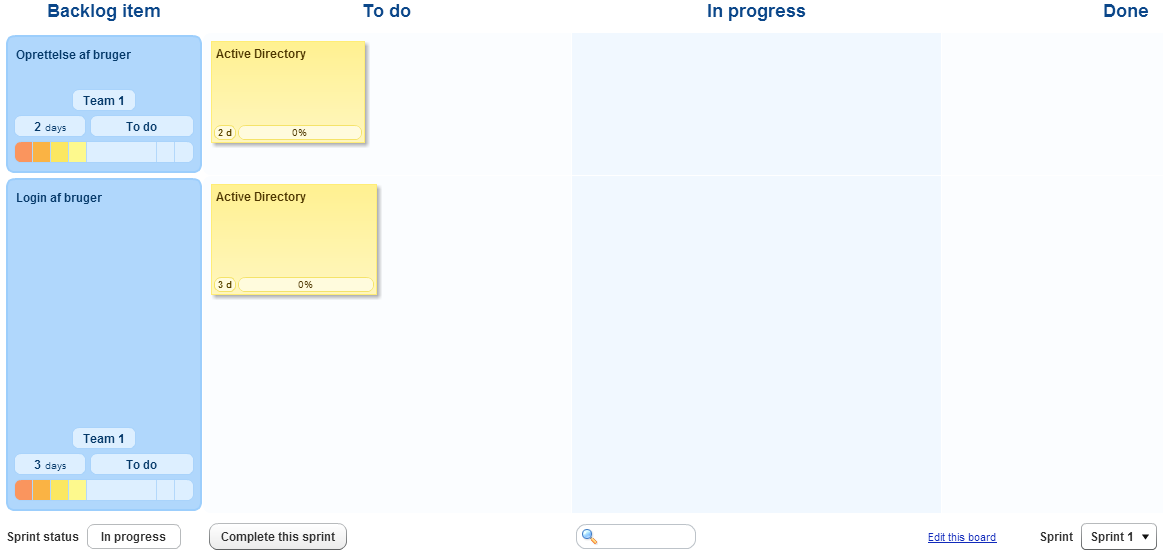
På backgrund af vores Release backlog har vi udarbejdet vores Sprint Backlog, hvor vi nu har taget de enkelte User stories fra vores product Backlog og inddelt dem alt efter hvilket prioritet de har forhold til Business value.

Som det ses er de enkelte user stories indelt i sprint.



## Scrum board

Nedenstående billede er en illustration af vores Scrum board fra ScrumWise, hvor vi har mulighed for få et overblik arbejdsmængden i det sprint, som vi arbejder med, samt status på de enkelte tasks.



### Figur Scrum Board

# Sprint 1

I det første sprint vil vi fokusere på implementeringen af de features, altså funktioner som er alt afslørende i forhold til brug af ZmartWallet.

## User story 1 – Oprettelse af bruger

Som bruger ønsker jeg at være i stand til, at oprette mig som bruger

Når ZmartWallet bliver startet op for første gang vil brugeren vil blive mødt med en login skærm , hvor brugeren har mulighed for at login eller oprette sig som bruger hvis han eller hun ikke har en konti. Dette er ganske enkelt afgørende for at brugeren kan benytte ZmartWallet.

Ved oprettelse vil brugeren blive bedt om at angive en email adresse og et password. Når disse informationer er udfyldt vil brugeren, blive omdirigeret tilbage login skærmen, hvor brugeren vil være i stand til at logge ind og benytte ZmarWallet. Når brugeren bliver omdirigeret, vil dette URL indeholde en AccessToken fra Active Directory, som brugeren ikke kan og må se, og vil i bagrunden blive håndteret og gemt på telefonen.

## User story 2 - Login af Bruger

Som registreret bruger ønsker jeg, at være i stand til at kunne logge ind.

Validering af bruger i form af indtastning af email og password er også afgørende for at man kan bruge ZmartWallet selvom man er oprettet. Dette kan virke som en simpel sikkerheds foranstaltning dog er en det en vigtig en af slagsen.

## Active Directory

Det var et krav fra Resolve-It, at vi skulle implementere et registrerings og login funktion, som skulle foregå via Microsoft Azure Active Directory**[[20]](#footnote-20)**, da firmaet er oprettet som kunde i Microsoft Azure. Vi skulle derfor ikke tænke over eller implementere hvordan backenden fungerer, da det hele var

udviklet og klar til brug, vi skulle kun kalde API’et og hente og sende data via ZmartWallet API til Active Directory.

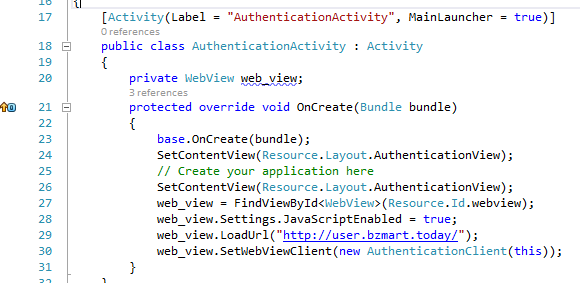
Formålet med Active Directory er, at det er et Authentication Directory, der kan håndtere Enterprise Systemer, Contact Information og samtidig levere mekanisme for central desktop styring og håndtering.

I forbindelse med ZmartWallet’s brug af Active Directory, er det blot bruger registrering og login delen der er relevant, da dette login kan bruges til alle nuværende og fremtidige applikationer som Resolve-It har eller udgiver, og dermed spare en masse tid og er også brugervenlig, ved at brugere kun behøver oprette sig en gang, når de downloader et Resolve-It produkt og kan bruge samme login og password til firmaets andre produkter.

## Implementering af User Story 1+2 – Oprettelse & Login af bruger

Da back-end delen allerede er på plads, og vi i den forbindelse kun skal foretage et kald til API, har vi valgt i vores Android App, at gøre brug at et WEBVIEW, for at vise brugeren en registrerings form, hvor han/hun kan skrive sit navn, sin E-mail adresse, samt et password, og trykke på OPRET knappen. API’et sørger for alle GUI elementer, og oprettelsen i baggrunden, som bliver vist i WEBVIEW. Vi har valgt at bruge et WebView, da de elementer vi skal bruge fra API’et en side fra en hjemmeside, og for ikke at brugeren skal ledes ud af App’en for at komme på denne side fra API, gør WebView os istand til at vise indholdet fra API som om det er en del af Applikationen, og brugeren ikke er klar over at indholdet faktisk er en web side. Vi gør også brug af URL’et til dette WebView til håndtering af AccessToken, som vi vil komme nærmere ind på.

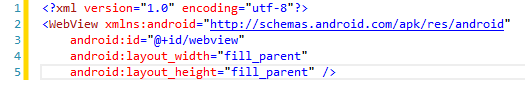
Vi startede derfor med at tilføje en ny Activity, som vi har kaldt; AuthenticationActivity som ser ud således;

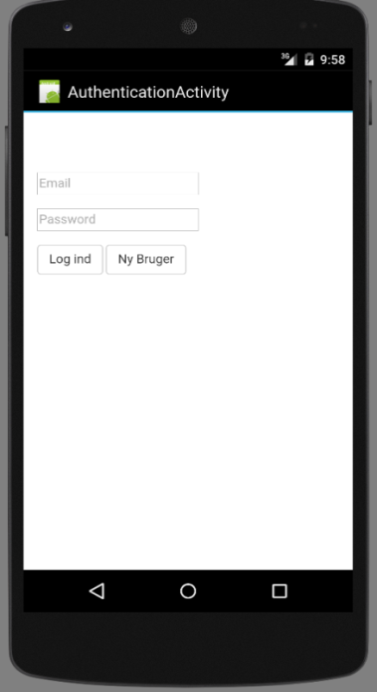
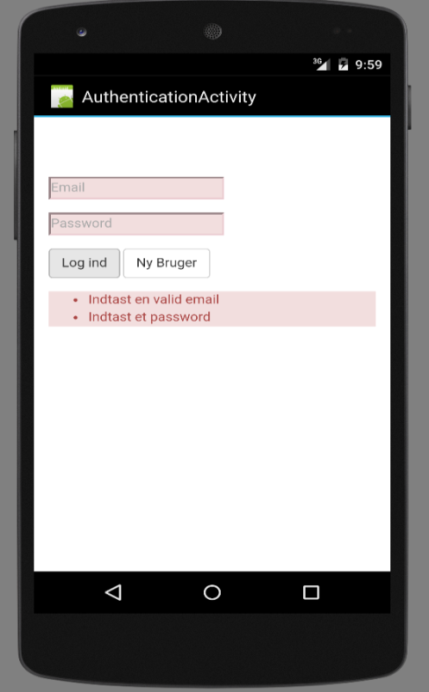
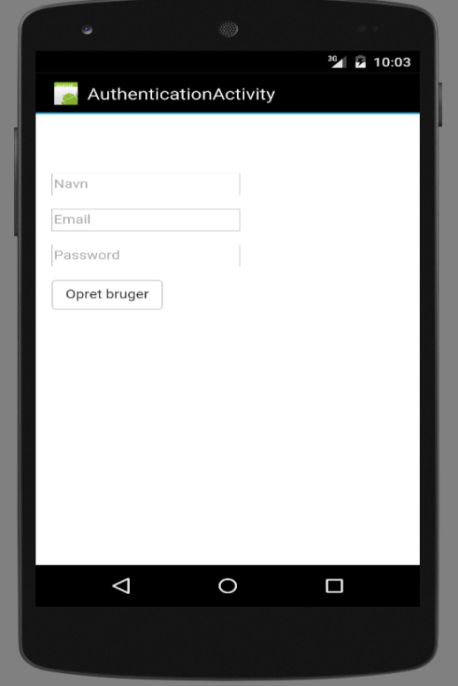


Vi tilføjer et AuthenticationView.AXML, som er GUI Layoutet til denne aktivitet, som også fremgår i AuthenticationActivity, linie 24, hvor vi sætter Content til at være AuthenticationView og sætter

Webview’ets LoadUrl til at være adressen på ZmartWallets API til Active Directory, til Oprettelse eller Login af bruger.

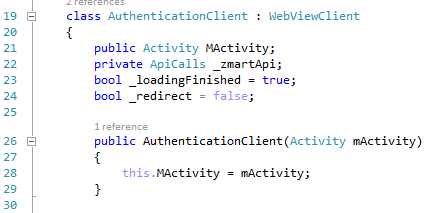
I vores AuthenticationView.AXML indsætter vi et WEBVIEW widget, som kommer til indeholde indholdet af det content der kommer ind via AuthenticationActivity kaldet til ZmartWallet API’et.



Resultatet af ovenstående er illustreret på nedenstående billeder, alt backend er som tidligere nævnt styret af ZmartWallet API’et;

Når brugeren har indtastet navn, email samt password og klikker på Opret bruger knappen, vil WebView nu blive redirected til login siden igen, og her træder Active Directory i gang igen, da denne redirect URL, som er skjult for brugeren, indeholder et AccessToken, som vil være det token der styrer, om denne bruger har adgang til applikationen, samt at brugeren ikke skal indtaste sine login oplysninger hver gang han/hun lukker App’en ned. Denne AccessToken håndterer vi kun i backend koden, da brugeren ikke kan og må kende til dette AccessToken på noget tidspunkt.

Vi har derfor oprettet en Class, AuthenticationClient der implementerer en WebViewClient, som gør os i stand til at hente og håndtere indholdet af vores AuthenticationActivity.



#### 

Den del af ovenstående Class som er interessant, er OnPageFinished metoden, hvor vi kalder den url,som der bliver redirected til gennem ZmartWallet API’et, hvor vi laver en filtrering i den redirect url, og laver en Substring (linje 64), og trækker den del af url strengen ud, som vi ved er den del i

strengen hvor AccessToken’en starter fysisk (linje 60) og gemmer det i en variabel vi kalder for accessToken, som vi kan bruge til at kalde API med til fremtidig brug.

I linje 65 sender vi denne accessToken igennem til vores ApiCalls class, som vi har sat til at være vores API controller, og henter en RefreshToken, som vi så kontrollerer i linje 73. I det tilfælde af, at vores accessToken er ugyldig, vil denne string være tom eller null, og vores kald i linje 76, vil derfor ikke blive startet, og brugeren må forsøge at logge ind igen med email og password.

Linie 65 autherization = \_zmartApi.GetRefreshToken(accessToken);



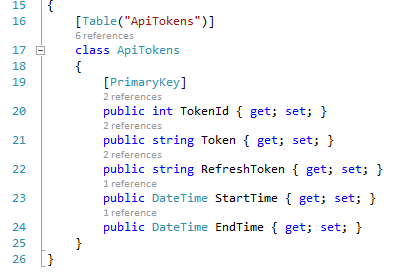
Ovenstående metode er vores kald til ZmartWallet API’et, som leverer en refreshToken til os, som vi kan bruge I vores App, således at når en bruger, opretter sig første gang eller logger ind via login siden, ikke skal tvinges til at logge på hvergang app’en bliver lukket ned. En refreshToken henter selv

en accessToken, hvis denne er udløbet, typisk 2 timer udløbstid på denne, vi har valgt at sætte en udløbstid på 30 minutter som det fremgår af i linje 58. En refreshToken kan holde i op til 90 dage, efter dette skal brugeren logge på igen med email og password.

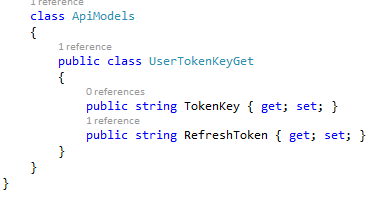
I ovenstående kald, kommer accessToken ind, som altid vil være gyldig, da den kommer ind via vores Login side. Det vil sige, at brugeren er logget ind med gyldig email og password, og skal nu hente en ny RefreshToken.

Kaldet starter ved linje 37, hvor vi sender en httpRequest til en adresse i ZmartWallet Api’et, som skal levere os en refreshToken. Indholdet der vil komme retur, vil være et JSON object, som vi så DeSerializer i linje 46.

Vi har valgt at oprette en Class, ApiTokens, hvor vi vil gemme den information vi får igennem denne JSON, og senere via denne Class, oprette en SQLite Tabel, som Resolve-IT ønsker vi skal gøre, da App’en også skal kunne bruges offline og derfor oprette en lokal Database på telefonen, hvor vi kan gemme alt information i en SQLite Database. Denne Class ud som følgende;



Bemærk linje 16 samt linje 19, her definerer vi gennem et SQLite.Net reference, at denne Class, skal være en Tabel, hvor indholdet skal være Tabel columns, og bruge linje 20 som en PrimaryKey i denne tabel.

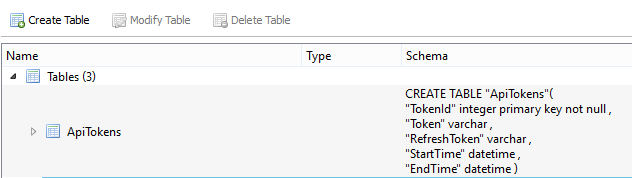
Vi har valgt at oprette en Class til at indeholde alle vores Serializable Classes, som vi vil bruge til at Mappe alt JSON indhold vi vil kalde og sende til ZmartWallet Api’et, denne Class ser på nuværende tidspunkt således ud;

Som man kan se er der en TokenKey, som vil være AccessToken, og en RefreshToken.

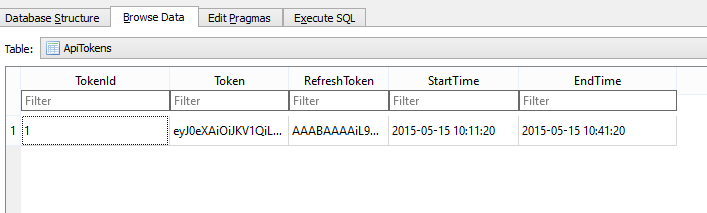
Disse 2 gør vi brug af i vores GetRefreshToken metodekald, fra linie 52 til linie 59, og indsætter/opdaterer dem i vores SQLite Database, linie 62 eller 64 , som er blevet oprettet i linie 35.

For at finde ud af hvilke informationer der kommer igennem vores HttpRequest, sætter vi et breakpoint ved linie 41, og når kaldet bliver udført og stopper ved dette breakpoint, kopierer vi indholdet af content, og indsætter inholdet i et Online JSON Validator**[[21]](#footnote-21)** og kan ved hjæp af denne Validator læse hvad vores Serializable Class skal indeholde, og i dette tilfælde er det TokenKey og RefreshToken, som vi indsætter i vores UserTokenKeyGet Class og vores ApiTokens Class som vi Mapper vores SQLite.Net til, og gør vores oprettelse af Databasen lettere, i form af mindre kode.

Når kaldet er fuldført, kan vi ved hjælp af det indbyggede ADB – Tool, hente vores Database fil ud af telefonen, som vi har sat til at hedde ZmartWalletDB og kan nu ved hjælp af et open source program ved navn, SQLiteBrowser, læse indholdet af vores Database, og se om Tabellen blev oprettet og de korrekte værdier indsat.

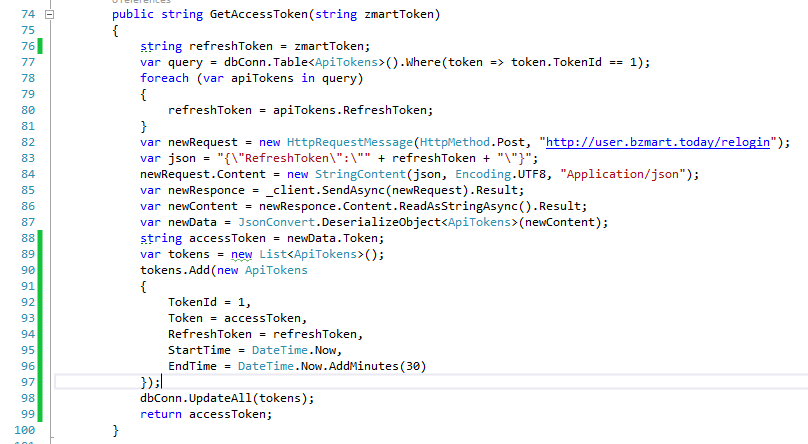


Som det fremgår af ovenstående screen-shot af vores ZmartWalletDB, så er alt oprettet som det skal, vores ApiTokens Class er sat som Tabel, og indholdet af Classen er korrekt indsat som Columns, og TokenId er sat som Primary Key.



Som det fremgår at ovenstående er de korrekte values indsat i Databasen, hvor vi i vores GetRefreshToken metode linie 54 hat sat TokenId = 1, Token samt RefreshToken kommer fra API

kaldet, og i linie 57 bruger DateTime.Now og i linie 58 tilføjer vi 30 minutter og har gemt som vores egen udløbstid for accessToken, denne tid vil være afgørende for vores næste metode, som vi har kaldt GetAccessToken, der vil blive kaldt, hvis tiden for EndTime er overskredet.

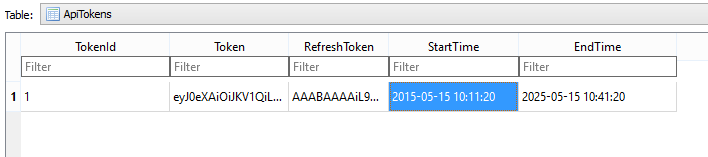


Denne metode ligner meget vores GetRefreshToken, dog kan man i linie 82 se, at kaldet til API er anderledes, samt at det nu er i linie 88 er et accessToken vi henter. I linierne 90 til 97 kan vi se at vi opdaterer indholdet i klassen og opdaterer vores Database i linie 98, og returnerer nu accessToken i linie 99.

Denne fremgangsmåde, med at vi læser indholdet af HttpRequest i form af JSON har vi benyttet til alle vores kald til API’et som vi har fået udleveret af Resolve-It, og dermed kunnet oprette alle vores Serializable Classes samt vores Entity Classes til at oprette vores Database tabeller.

Under udvikling af denne funktionalitet og nuværende implementering, slog det os at applikationen ikke er sikkert, da vi uden videre i form af ADB-Tool, som er et debugging værkstøj der følger med Android SDK, kunne hente Database filen ud uden problemer, og se indholdet af denne, og faktisk også ved hjælp af programmet SQLiteBrowser ændre i indholdet og via ADB-Tool sende denne ”modificerede” database tilbage til telefonen, og derved snyde App’en til at tro at AccessTokens udløbstid f.eks. ikke er udløbet som det fremgår af nedenstående billede, hvor vi har ændret datoen til 10 år frem.

Vi har gjort Resolve-It opmærksom på denne trussel, og løsningen på dette kan f.eks. være kryptering af indholdet i SQLite. Access og RefreshToken vil Resolve-It sørge for noget kryptering af på API siden, og vi skal ikke tænke mere over sikkerheden på nuværende tidspunkt og beholde en lokal database.



## Evaluering af Sprint 1

Vi har nu implementeret de user user stories Sprint 1 indeholder og afholder Sprint Review med product owner som vil foretage acceptance test(s), og vil derefter internt afholde et Sprint Retrospective møde og afslutte sprint 1.

### Acceptance Test

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Projekt Identifikation** | | | | | | | | | | |
| Projekt Navn: | | | | | Projekt Start Dato: | | | | | |
| ZmartWallet | | | | | 20-04-2015 | | | | | |
| Product Owner: | | | | | | | | | | |
| Sinus Frank | | | | | | | | | | |
| Acceptance Kriterier Tests | | | | | | | | | | |
|  | | | | VigtigeFunktioner | | | Test Resultater | | |  |
| Nummer | | Accept kriterier  Beskrivelse | | Ja | | Nej | Accepteret | Afvist | | Kommentarer |
| 1 | | Bruger kan Oprette profil | |  | |  |  |  | |  |
| 2 | | Bruger kan Logge ind | |  | |  |  |  | |  |
| Godkendt | | | | | | | | | | |
| Product Owner | | | Signatur | | | | | | Dato | |
| Ja | Nej | | Sinus Frank - Resolve-It | | | | | | 15-05-15 | |
|  |  | |  | | | | | |  | |

## User Story 1: Opret Bruger Test

Bruger skal vælge Opret bruger og blive sendt videre til Login Form

Bruger skal indtast Navn, Email samt et Password, og klikke på Opret Bruger

Bruger er Oprettet.

## User Story 2: Login Bruger Test

Bruger skal indtaste email samt password og klikke på Login

Bruger indtaster Email samt password og klikker på LOG IND

Bruger bliver logget ind.

### Sprint Retrospective

Vores første møde afholdte vi efter vores sprint review, hvor vi drøftede brugen af SQLite, ifb. Med sikkerhed, men da Resolve-It gerne vil beholde en offline database og hellere vil fokusere på at implementere sikkerhed på API siden, blev vi enige om ikke at gøre mere ved dette, dog ville vi foretrække en anden løsning, hvis det var op til os, hvilket ville være at hente alt indhold online og gøre brug af telefonens hukommelse til at opbevare data, da vi mener en telefon mere eller mindre er online altid.

Vores indbyrdes samarbejde gik godt og vi følte vi havde en god forståelse for hvad vi skulle nå og udføre og udnytte tiden optimalt.

På baggrund af evalueringen i form af en godkendt acceptance test og tilladelse fra ReSolveIt vil vi fortsætte med implementeringen af de user stories som vi har prioriteret i forbindelse med Sprint 2.

# Sprint 2

I dette vil vi koncentrere os om, at implementere de user stories der har med brugerens information atgøre.

## User Story 3: Hent Bruger Information

Som en bruger, ønsker jeg at hente min brugerinformationer, så jeg kan se dem eller ændre i dem inde I App’en.

## User story 4: Gem bruger information

Som en bruger, ønsker jeg at gemme min brugerinformation, så jeg senere kan vise dem eller ændre dem inde i App’en

## User Story 5: Vis Bruger Information

Som en bruger, ønsker jeg at få vist mine bruger informationer, så jeg kan se om de er korrekte og evt. skal rettes i.

## Implementering af User Story 3: Hent bruger information

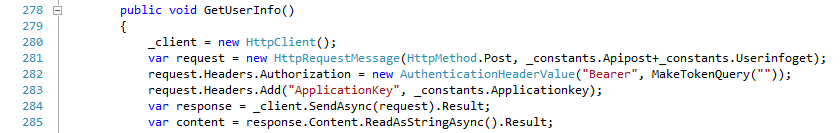
## Tasks :

1. Kald til API.
   1. HttpClient
   2. Active Directory
   3. AccessToken/RefreshToken
   4. JSON

Denne User story omhandler, at brugeren skal kunne være i stand til, at hente sine brugeroplysninger, da brugeren under registrering af sin konto, udelukkende skriver sit navn, samt e-mail adresse.

Til at løse opgaven har vi inddelt User story i ovenstående tasks.

Vi bruger HttpClient til at kalde og sende request til ZmartWallet API’et, og sender en accesstoken med, hvis denne er gyldig for at hente den eller de informationer om brugeren som vi kan arbejde videre med, er accesstoken udløbet sender vi en refreshtoken med, for at forny accesstoken og dermed få adgang til API’et.



Ovenstående billede illustrerer en lille del af det metodekald vi foretager for at hente og gemme brugerens info på telefonen.

Bemærk linje 282, hvor vi starter med at kalde en anden metode, MakeTokenQuery(””), som først skal kontrollere om brugerens accesstoken er gyldig og hvis ikke gøre brug af en refreshtoken for at fornye accesstoken og når dette er gjort fortsætte metoden og dermed hente og gemme brugerens information.



Her foretager vi et kald, hvor vi har gjort brug af SQLiteConnection og kontrollerer databasen for ApiTokens som blev oprettet under User Story 1, og bruger LINQ til at foretage vores query som man kan se i Linie 56, og returnere vores accesstoken, hvis denne er gyldig, hvilket vi kontrollerer i en anden metode CheckToken() som returneres som en boolean.



Her henter vi alt information ud fra vores ApiToken Tabel fra SQLite ved brug af LINQ, linie 72, og gemmer information i variabler, som vi kan bruge til kontrol.

Først starter vi med at sammenligne datoerne for access – og refreshtokens, og sammenligner med tiden NU, linie 83, og hvis denne if statement er sand, så er accesstoken tiden udløbet og må fornyes ved hjælp af endnu en metode, GetAccessToken() som det fremgår af linie 85 og når denne metode er færdig sætter vi vores boolean til at være sand, og returnere dette resultat til den vores oprindelige metode, som så kan fortsætte processen.

Men hvis linie 88 if statement er sand, betyder det at accesstoken er gyldig, og derfor returnere vi en sand boolen og den oprindelige metode kan fortsætte processen.

Linie 92, endnu en if statement, hvis denne er sand, så betyder det at vores Refreshtoken er udløbet, og det er derfor nødvendigt at gennemgå en ny Authentication og fornye både access og refreshtoken, og vi afslutter derfor alle processer og starter en Authentication Activity og lukker den oprindelige metode, som kan startes igen, når Authentication er fuldført.

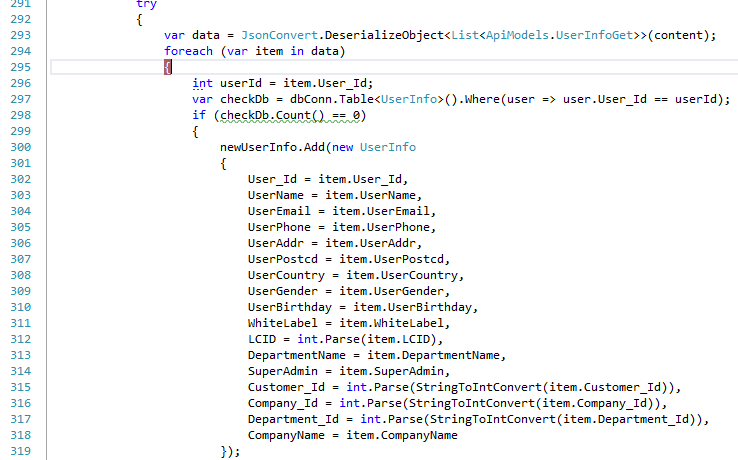
Vi antager at resultatet i ovenstående metoder returnerer sandt og kan fortsætte processen med at hente brugerens information, eller ny authentication er fuldført og vi kan nu fortsætte til user story 4.

## Implementering af User Story 4: Gem bruger information

## Tasks:

1. Gem information på telefon.
   1. SQLite
   2. SQLiteConnection
   3. Deserialize JSON
   4. Opret Class til brug for at gemme information

Når vi får et response fra API’et, kommer der en JSON objekt retur, som er en String, der indeholder brugerens information. Denne information har vi oprettet en Class til at opbevare alle variablerne i dets properties og gemme informationerne i en liste, for så at gemme i SQLite ved hjælp af SQLiteConnection.

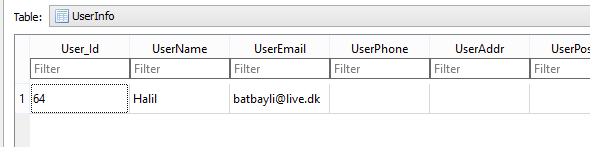


Ovenstående illustrerer denne process, at hente data som JSON, og convertere denne JSON ved hjælp af JSONConvert og Deserialize denne JSON og mappe hver attribute til vores UserInfo Class og dens properties for til sidst at gemme informationen i SQLite.

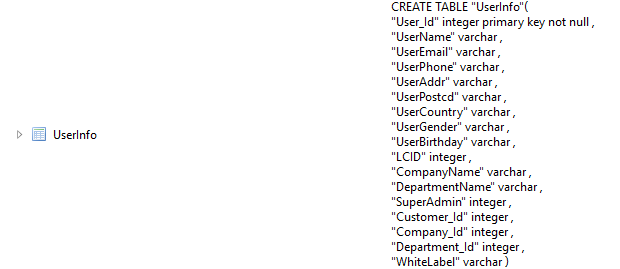
Linie 300, tilføje alle informationer i en Liste af UserInfo, og tildele hver information fra JSON strengen til UserInfo Class’ens properties, og i linie 355 indsætte disse informationer ind i SQLite.



Her et kort udsnit af hvordan UserInfo tabellen bliver udfyldt ved hjælp af ovenstående fremgangsmåde;



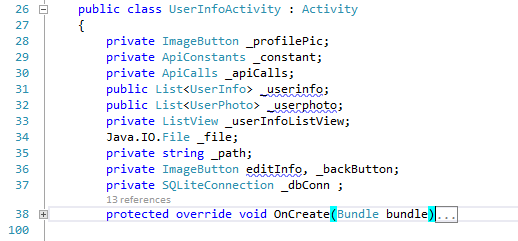
Og selve tabellen der bliver oprettet med minimal kode

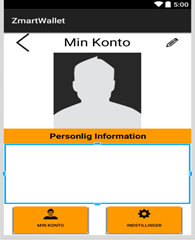


## Implementering af User Story 5: Vis Brugerinformation

Denne user story er en visuel implementering af user story 3 og 4, og vi har i den forbindelse gjort brug af ListView widget til at vise bruger information og oprettet en Adapter Class til at tildele brugerinformation fra SQLite og vise i denne ListView.

Vi har oprettet en UserInfoActivity Class, og et UserInforActivityView.axml til at håndtere koden bag og det visuelle, til at få vist brugerens informationer.

Her et kort udsnit af UserInforActivity;

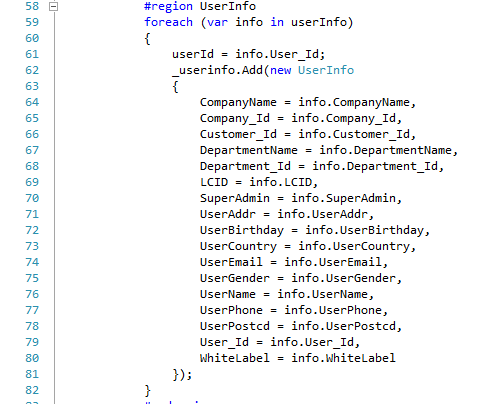
Layoutet ser ud som følgende;

Nogle af knapperne i billedet til højre er ikke implementeret, men det er sådan Resolve-It ønsker layout’et skal se ud, i en fremtidig iteration.

vi sørgede for at få lagt knapperne ind, efter Sprint Review med kunden.

ListView’et vi skal udfylde med brugerens information er den ”kasse” under Personlig Information.

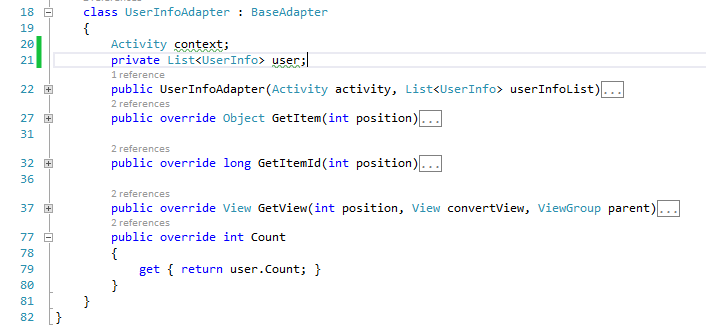
For at arbejde med dette ListView, skal vi først hente brugerens information fra SQLite, hvilket vi har gjort således;

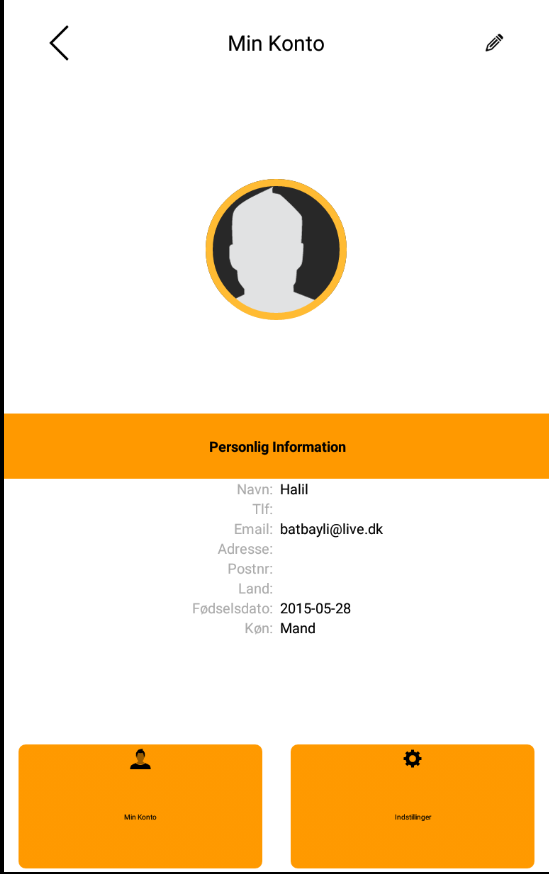


Når vi har hentet den brugerinformation vi skal bruge, så sender vi denne oplysning videre til en Adapter Class vi har oprettet, som skal håndtere informationen, og tilføje den til vores ListView.



Og herunder vores UserInfoAdapter der implementerer BaseAdapter



Metoden der er ansvarlig for at udfylde ListView er linie 37, GetView, hvor vi opretter en række TextView’s og via den liste med brugerinformation vi sendte igennem fra UserInfoActivity, kan trække alle informationerne ud og delegere dem ud til de forskellige TextViews, og i sidste ende få vist brugerinformationerne korrekt.

Illustrationen som ses til højre er resultatet af ovenstående process;

Grunden til at slut resultatet ikke helt ligner vores implenterings billede er fordi vores Emulator bruger en højere opløsning end det vi har implementeret, men det er Resolve-It efter Sprint Review’et informeret om og har accepteret, at ovenstående på nuværende stadie ser ud som den gør.

## Evaluering af Sprint 2

Vi har nu implementeret de user user stories Sprint 2 indeholder og afholder Sprint Review med product owner som vil foretage acceptance test(s), og vil derefter internt afholde et Sprint Retrospective møde og afslutte sprint 2.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Projekt Identifikation** | | | | | | | | | | |
| Projekt Navn: | | | | | Projekt Start Dato: | | | | | |
| ZmartWallet | | | | | 20-04-2015 | | | | | |
| Product Owner: | | | | | | | | | | |
| Sinus Frank | | | | | | | | | | |
| Acceptance Kriterier Tests | | | | | | | | | | |
|  | | | | VigtigeFunktioner | | | Test Resultater | | |  |
| Nummer | | Accept kriterier  Beskrivelse | | Ja | | Nej | Accepteret | Afvist | | Kommentarer |
| 3 | | Hent Bruger Information | |  | |  |  |  | |  |
| 4 | | Gem bruger Information | |  | |  |  |  | |  |
| 5 | | Vis bruger Information | |  | |  |  |  | |  |
| Godkendt | | | | | | | | | | |
| Product Owner | | | Signatur | | | | | | Dato | |
| Ja | Nej | | Sinus Frank - Resolve-It | | | | | | 22-05-15 | |
|  |  | |  | | | | | |  | |

### 

## User Story 3+4+5: Hent, Gem, Vis Bruger Oplysninger

Bruger klikker på Bruger informations knappen

Bruger kan se sine oplysninger i GUI

### Sprint Retrospective

Forløbet af sprint 2 gik godt, vi stødte på ingen problemer og fulgte planen og fik implementeret alt vi skulle til tiden og kunne få udført en godkendt acceptance test.

# Sprint 3

I dette sprint vi vil hente kort fra API og implementere aktivering af disse kort.

## User story 6: Hent Kort

Som en bruger ønsker jeg at hente alle kort, for at kunne se en liste af kort.

## User Story 7: Gem kort

Som bruger ønsker jeg at kunne gemme kort fra Api’et, så kortene kan vises i form af en liste.

## User story 8: Vis kort

Som bruger ønsker jeg, at få vist en liste af alle kort.

## User Story 9: Aktiver Kort

Som en bruger, ønsker jeg at klikke på et kort, for at aktivere det og gøre kortet personligt.

## Implementering af User Story 6: Hent Kort

## Tasks

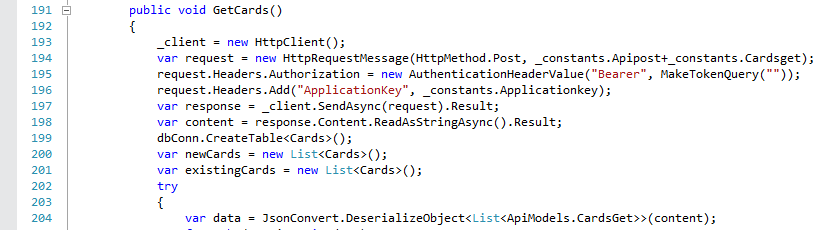
Kald til Api.

1. Kald til Api.
2. HTTPClient.
3. Active Directory.
4. Acees Token og Refresh Token.
5. JSON.
6. Oprettelse af classes for at kunne gemme data.

Denne user story omhandler, at bruger skal være i stad til at hente alle kort fra ZmartWallet Api’et.

Da denne tilgang minder meget om fremgangmåden i user story 3 vil vi ikke gå i specifikke detaljer omkring implementering.

Som det fremgår af nedenstående code snippet, er den reelle forskel i linje 194, hvor der nu kaldes på en constant.Cardsget og på 199 og 200, hvor det er Class **Cards** , der benyttes til at gemme data i.



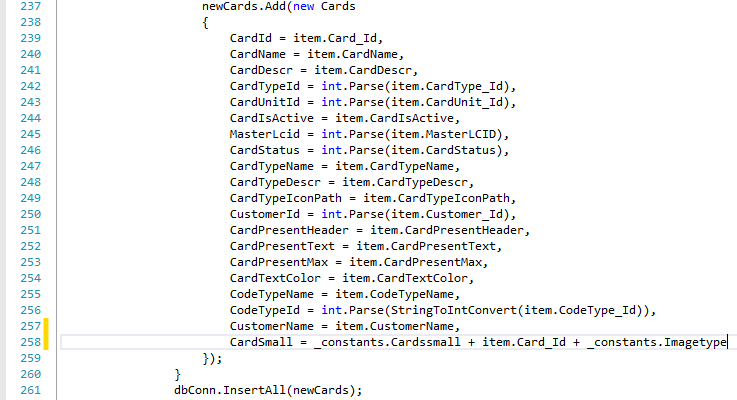
## Implementering af User Story 7: Gem Kort

## Tasks

1. Gem kort på telefon.
2. SqlLite.
3. SqlConnection.
4. Deserialize JSON.
5. Brug af eksisterende Classes til at gemme data.

Implementerigen for denne user story er basalt set lig med fremgangsmåden i user story 4, hvor vi benytter samme tilgang i forhold til at deserialize JSON og SQLConnection til at gemme i Sqllite databasen.

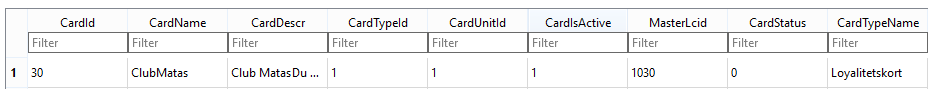
Da vi arbejder med billeder , som kan optage en del plads på telefonen, har vi valgt kun at gemme en url i sqllite databasen, som referer til et billede af det pågældende kort fra ZmartWallet Api’et, for dermed at spare på lagringsplads.

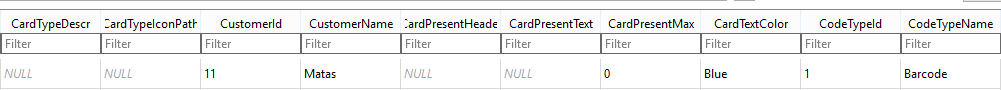


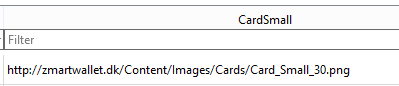
Ved linje 258 definerer vi path til et specifikt kort, hvor der benyttes Cardsmall og hvilken billed type, der bruges, hvilket er illustreret nedenfor.

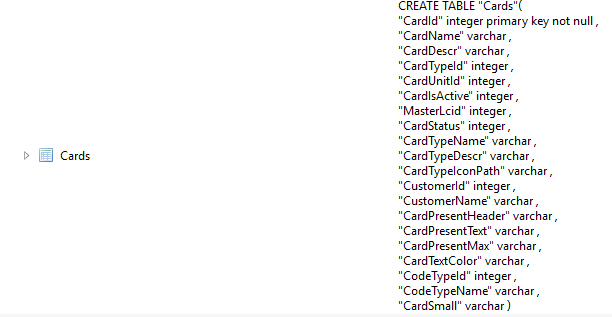




Nedstående billede er en illustration af hvordan kort information bliver lagret i databasen.







## Implementering af User story 8: Vis kort

## Tasks

1. Hent alle kort fra SQlite (url path)
   1. SQLiteConnection
   2. LINQ
2. Load billeder fra url i en liste i App’en
   1. GridView til at vise billeder
   2. GridViewAdapter til håndtering af billeder
   3. LazyLoad billeder

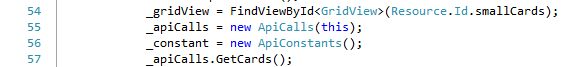
## 

Til at løse denne User Story, har vi som vi hidtil gjort, gjort brug af SQLiteConnection og LINQ til at query databasen og hente information ud som vi skal buge, og i dette tilfælde skal vi hente alle kort fra SQLite, hvilket vi har gjort som følger;





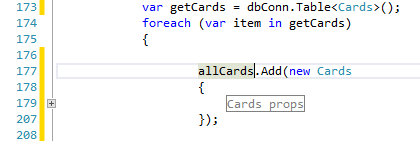
Her laver vi en Liste af Cards som skal indeholde alle kort fra SQLite vi henter. (Linie 43 og 61)



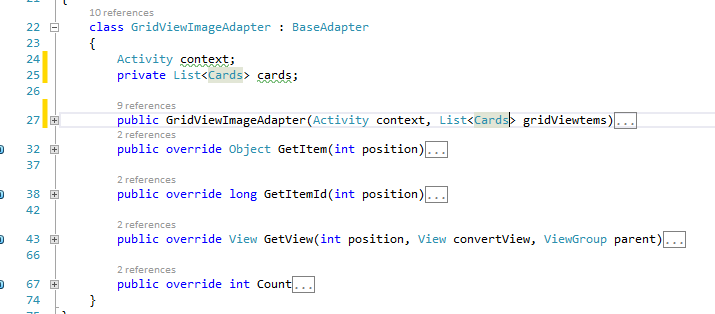
Her instancierer vi vores GridView som vi skal udfylde med kort, og kalder API’et om at hente alle kort og gemme i SQLite, som var i User Story 6+7, (Linie 54 og 57)



Her kalder vi vores GridViewImageAdapter, som skal sørge for LazyLoad af vores billeder (Linie 75), vi sender listen af kort som vi har udfyldt ved hjælp af SQLiteConnection (Linie 173-208);

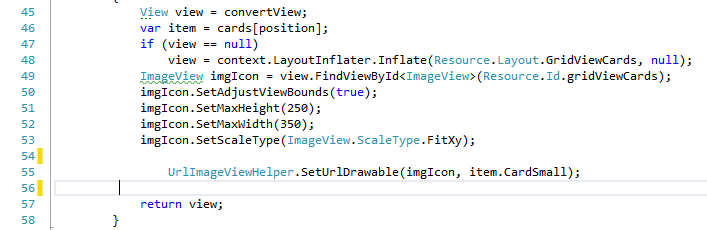


Vi opretter vores GridViewImageAdapter Class som extender en BaseAdapter



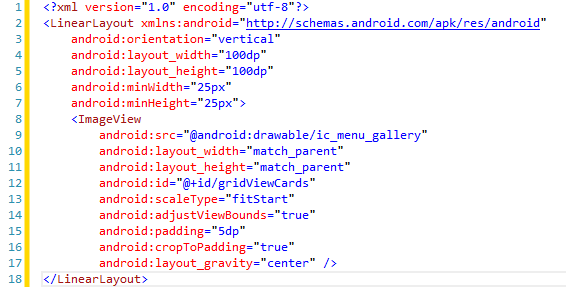
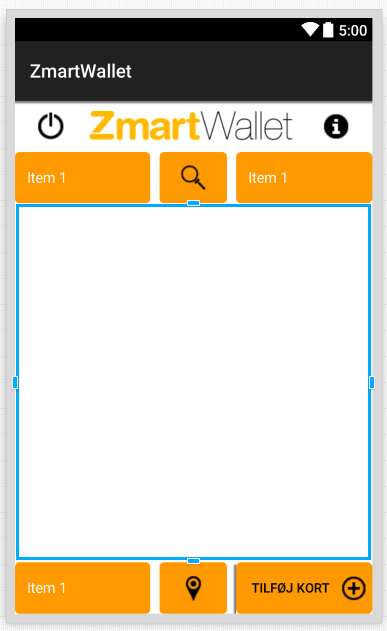
På linie 25 og 27 kan vi se den liste af Kort vi skal arbejde med

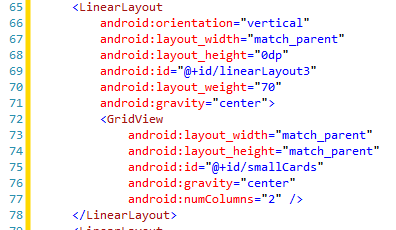
Resten af metoderne i denne Class er standard, og den vigtige metode her er linie 43 GetView Metoden som sørger for billedbehandlingen, samt LazyLoading af billederne.



Her er indholdet af denne metode, vi bruger en ImageView (linie 49) til at holde hvert enkelt billede i den Liste af Kort vi arbejder med, og bestemmer hvilket kort position vi skal bruge til denne ImageView ved at bruge linie 46 og linie 55.

LazyLoad implementeringen var den nemme løsning (Linie 55), da der allerede er implementeret et library til dette formål, UrlImageViewHelper Library som kan hentes som en extension til Xamarin, som sørger for alt arbejdet, med at hente et billede, når det kræves, dvs. at App’en må maximalt hente X antal billeder svarende til X antal hukommelse som der bliver defineret, 1024 som standard, og denne Class vil sørge for at fjerne de ældste billeder der er hentet hvis telefonen er ved at løbe tør for hukommelse, og hente dem igen, hvis brugeren begynder at rulle tilbage igen mod start positionen, dette giver en meget responsivt GUI og god brugervenlighed, da brugeren ikke oplever at der bliver hentet billeder i baggrunden.



Her er vores ImageView der skal bruges i vores GridViewImageAdapter til at holde hverenkelt billede, for så at blive sendt return til vores GridView

Og her vores GridView der skal holde ImageView fra Adapteren, læg mære til linie 77, hvor vi definerer max 2 rækker horizontalt, og udfylder GridView’et lodret, dvs man kan rulle listen ned eller opad.

Vores billede af vores GUI indeholder elementer, som Resolve-It ønskede vi skulle have med, dog ikke implementere koden bag endnu, men efter vores Sprint Review ønskede kunde at se hvordan GUI ville se ud hvis disse elementer var med, og vi satte derfor hurtigt elementerne ind, som er en række knapper samt Spinners, der vil blive implementeret i senere Sprint’s.

Resolve-It er tilfredse med Resultatet, og vi kan fortsætte til næste User Story.

Ovenstående er screenshot af indholdet af vores GridView, når GridViewImageAdapter har været i gang og benyttet LazyLoad til at hente billeder fra URL.

## User story 9: Aktiver kort

## Tasks:

1. Klikke/Vælge et kort i Gridview
   1. SQLiteConnection
   2. UrlImageViewHelper(LazyLoad)
   3. HttpClient
   4. JSON
   5. Knytte kort til bruger

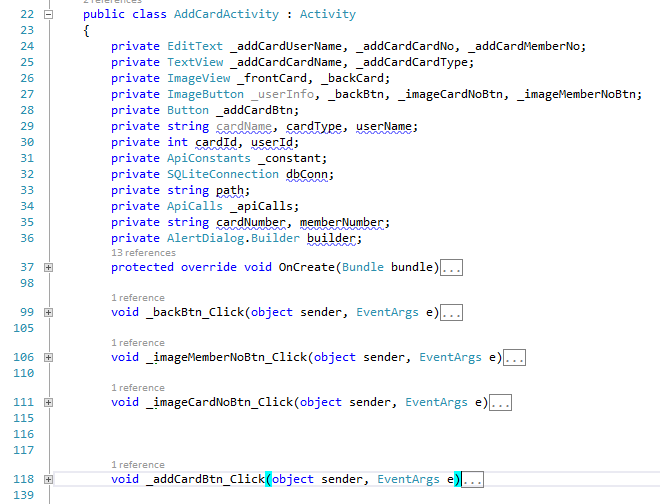
For at gennemføre implementering for aktivering af denne User story, har vi inddelt opgaven i mindre tasks, som er nødvendige for, at kunne aktivere et kort.

Når en bruger henter kort fra API, kommer disse ind med en status; ***isAktive=no*** og aktivering af et kort vil efterfølgende ændre denne til ***isActive=Yes*** denne status er vigtig, da kortene som er aktive, får ”trykket” brugerens navn samt et kort nummer og medlemsnummer på den URL hvorfra kortet vil blive vist fra. (Dette er udført af API’et og er ikke noget vi skal implementere).

På dette billede kan vi se vores GridView Click event som skal starte aktiverings processen af et kort;



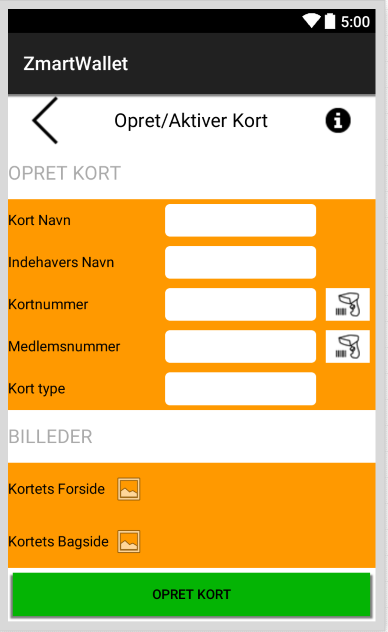
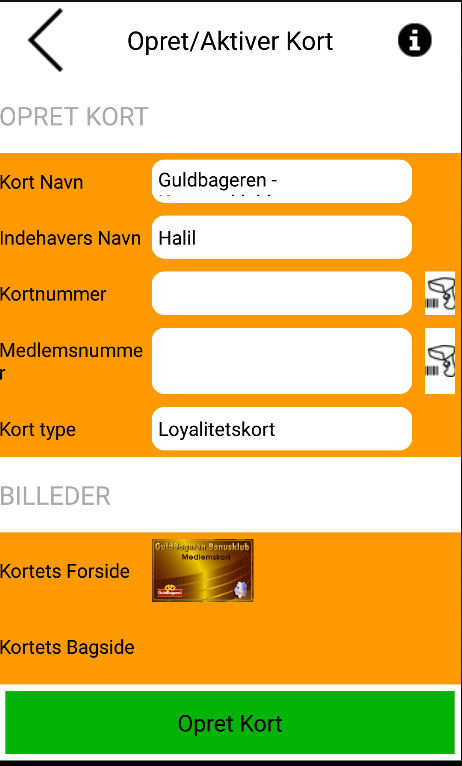
Vi bruge igen SQLiteConnection og LINQ til at hente information, og vi bruger et AdapterView.ItemClick event (Linie 98) for at hente information om det kort vi trykker på i GridView og benytter linie 105 for at definere et ID for dette kort, som passer til et kort nummer, og bruger dette ID i linie 107-108 for, at lave en query med SQLiteConnection og hente information om dette kort, for senere at sende denne information til en ny Activity, som Intent Extra, og starter vores AddCardActivity (linie 121-128). Linie 132-136 er en feature der skal implementeres i en senere Sprint. Herunder vores AddCardActivity Class;



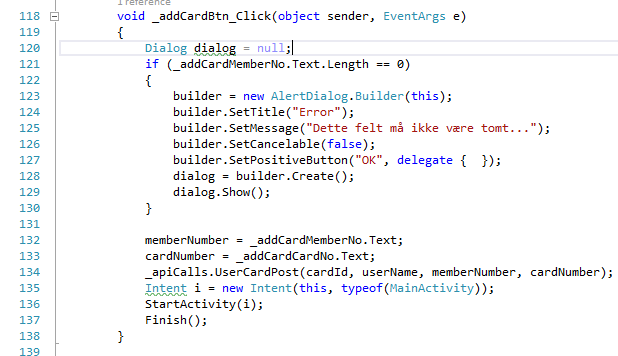
Og her vores måde at hente Intent Extras fra MainActivity;



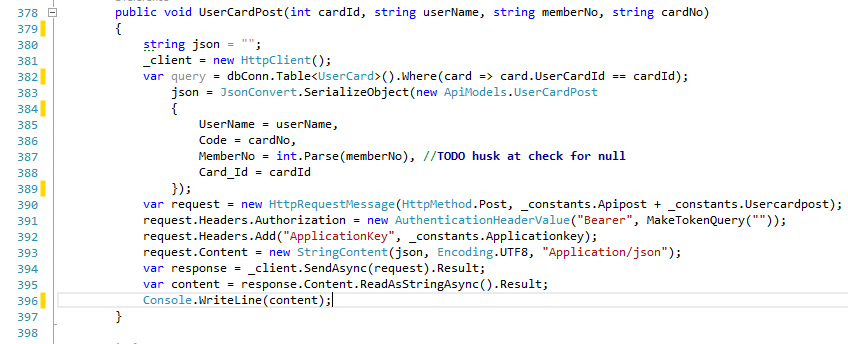
Bemærk linie 58, hvor vi kører metoden i en UITråd da vi benytter os af en AlertDialog til at få vist den information vi ønsker at arbejde med og sende til API’et for at gemme aktiveringen af kortet.



Ovenstående billede viser hvordan vores AddCardView ser ud, og hvordan denne bliver udfyldt i Emulatoren, når en bruger vælger et kort i GridView og sender information igennem via Intent som skal bruges til at udfylde de nødvendige felter, hvor brugeren selv skal udfylde et kort nummer og medlemsnummer, og ved at trykke på Opret Kort knappen, vil dette kort blive aktiveret, og fremgå som aktiveret i GridView. Vores metode til denne funktion ser ud således;



Bemærk linie 132-134, hvor vi sender disse parametre med til vores API kald til at aktivere et kort.



Og igen et kald til API’et denne gang sender vi data til API, som sørger for at behandle disse og indsætte hvor de skal indsættes, hvorpå vi nu i App’en kan hente ny data, og indsætte og opdatere kort (User Story 3+4+5), og brugeren kan nu i GridView se et aktiveret kort, som vil være 100% synlige, hvor inaktive kort er 25% synlige.

Ovenstående billede viser aktive kort til venstre og inaktive kort til højre.

## Evaluering af Sprint 3

Vi har nu implementeret de user user stories Sprint 3 indeholder og afholder Sprint Review med product owner som vil foretage acceptance test(s), og vil derefter internt afholde et Sprint Retrospective møde og afslutte sprint 3.

## Acceptance Test

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Projekt Identifikation** | | | | | | | | | | |
| Projekt Navn: | | | | | Projekt Start Dato: | | | | | |
| ZmartWallet | | | | | 20-04-2015 | | | | | |
| Product Owner: | | | | | | | | | | |
| Sinus Frank | | | | | | | | | | |
| Acceptance Kriterier Tests | | | | | | | | | | |
|  | | | | VigtigeFunktioner | | | Test Resultater | | |  |
| Nummer | | Accept kriterier  Beskrivelse | | Ja | | Nej | Accepteret | Afvist | | Kommentarer |
| 6 | | Hent kort | |  | |  |  |  | |  |
| 7 | | Vis kort | |  | |  |  |  | |  |
| 8 | | Gem kort | |  | |  |  |  | |  |
| 9 | | Aktiver kort | |  | |  |  |  | |  |
| Godkendt | | | | | | | | | | |
| Product Owner | | | Signatur | | | | | | Dato | |
| Ja | Nej | | Sinus Frank - Resolve-It | | | | | | 29-05-15 | |
|  |  | |  | | | | | |  | |

## User Story 6+7+8: Hent, Gem, Vis Kort

Bruger kan se sine kort når han logger ind

## User story 9: Aktiver kort

Bruger klikker på et kort han vil aktivere

Bruger udfylder sit kortnummer, medlemsnummer og klikker på OPRET KORT

Kort bliver aktiveret

## Sprint Retrospective

Denne sprint gik uden de store udfordringer, da vi allerede har beskæftiget os i sprint 2 med at hente data og gemme data og var derfor allerede forberedt på denne sprint, dog havde vi en lille udfordring i Aktiver kort delen, vi ikke kunne få til at fungere, men fandt ud af at løsningen var at bruge Task<T> til at løse opgaven. Vi overholdte tidsplanen og kunne fremvise en godkendt acceptance test og dermed fortsætte til næste sprint.

# Sprint 4

I dette sprint vil vi implementere visning af stempel kort, QR kode på kort, implementere Scanner App og transaction fra scanner app til API

## User Story 10: Vis Stempelkort

Som en bruger, ønsker jeg at vise mit stempelkort, så jeg kan se hvor mange stempler jeg har optjent

## User Story 11: Vis QR Kode / Opdater stempler

Som en bruger, ønsker jeg at få vist min QR kode på mit kort, så butikken kan scanne den.

## User Story 12 – ZmartScanner App

Som en bruger, ønsker jeg at kunne scanne en QR Kode på ZmartWallet App, for at kunne se om kunden har opnået en bonus/rabat.

## Implementering af User Story 10: Vis Stempelkort

Som en bruger, ønsker jeg at vise mit stempelkort, så jeg kan se hvor mange stempler jeg har optjent

## Tasks:

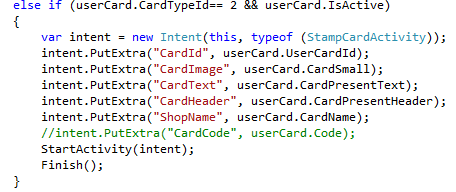
1. Klikke/vælge Kort i gridview
   1. SQLiteConnection
   2. GridView og ImageView til stempler
   3. Gridview Adapter
   4. UrlImageViewHelper(LazyLoad)
   5. JSON
   6. Kald til API

For at gennemføre implementering af denne User Story, har vi opdelt denne User Story i mindre tasks, som er nødvendige for at få vist et stempelkort og antal af stempler en bruger har optjent.

Når en bruger vælger/klikker på et stempelkort i App’ens MainActivity, foretager vi et kald til databasen, for at hive kort information ud ved hjælp af SQLiteConnection;

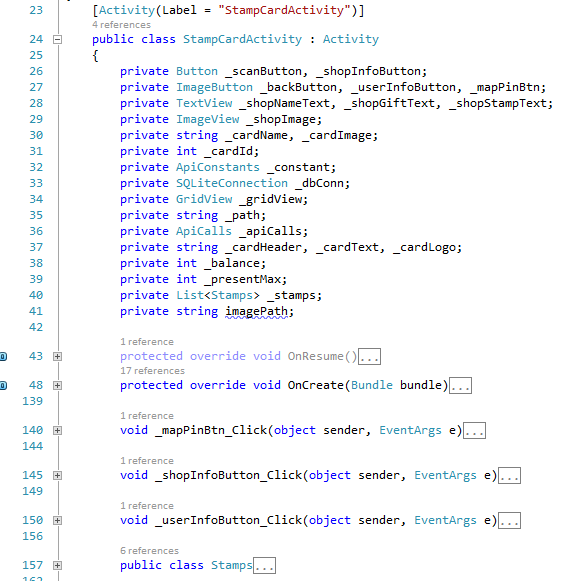


Ovenstående query vil returnere et resultat, og hvis dette resultat er lig med 1, så har vi fundet det kort vi skal bruge og kan fortsætte vores process, med at få vist det valgte stempelkort. Vi har foretaget et foreach loop og foretager denne IF statement;



Altså hvis kort type er lige med 2, hvilket er stempelkort i databasen, og hvis kortet er aktivt, så foretager vi ovenstående kald for at komme til StampCardActivity og putter parametre med som skal bruges i denne Activity.

StampCardActivity som skal behandle de ”Extra” der kommer fra MainActivity ser ud således;



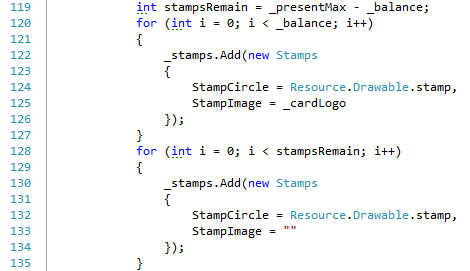
Og vores OnCreate metode ;

Fra linie 90-94 henter vi alle ”Extra” fra tidligere Intent, som vi nu kan bruge for at fortsætte med at få vist vores stempelkort.

Vi bruger igen lazyload til at hente billede fra API og få vist i et ImageView (linie 98).

Fra linie 102-108 foretager vi igen kald til SQLite for at hente information ud fra databasen,som vi kan bruge, f.eks. linie 104 hvor mange stempler en bruger har optjent og linie 105, hvor mange stempler en bruger skal bruge for at opnå en rabat/bonus.

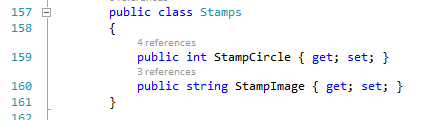
For at vi visuelt kan vise hvor mange stempler en bruger har og hvor mange brugeren mangler har vi valgt at lave en kalkulering først ved hjælp af 2 FOR loops som ser ud således;



Linie 119 finder vi først ud af hvor mange stempler en bruger mangler til at opnå en bonus, denne variabel bruger vi senere. Lad os antage at;

\_presentMax = 5 og \_balance = 1, dvs. at brugeren mangler 4 stempler for at opnå bonus.

Linie 120-126 laver vi vores første FOR loop for at tilføje et ”stempel” billede i vores View for at indikere, at brugeren har X-antal stempler, i vores antagelse har brugeren 1 stempler.

Vi har oprettet en Class, der kan ”holde” vores stempler. 

En meget simpel Class, der skal indeholde en Drawable, som er baggrunds billede med en cirkel, og vores billede af stemplet som bliver hentet i en URL.

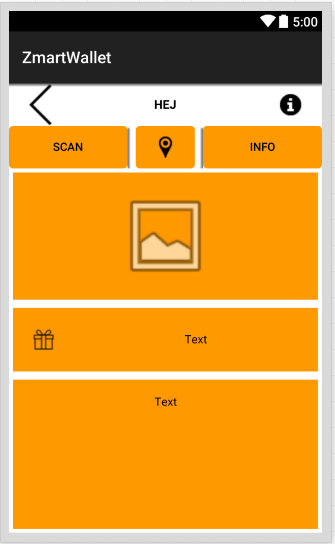
I løbet af vores FOR loop tilføjer vi en drawable samt stempel, og tilføjer til en Liste af Stamps, som vi skal bruge i vores StampViewAdapter, som skal behandle billede visning af stempler.

Linie 128-134 foretager vi vores andet FOR loop, for at vise brugeren hvor mange stempler brugeren mangler, og i vores antagelse mangler brugeren 4 stempler.

Vi foretager samme process med at gemme informationen i vores ”holder” Class Stamps, men denne gang så sender vi kun en Drawable med, da der ikke skal være et billede af stempel med her.

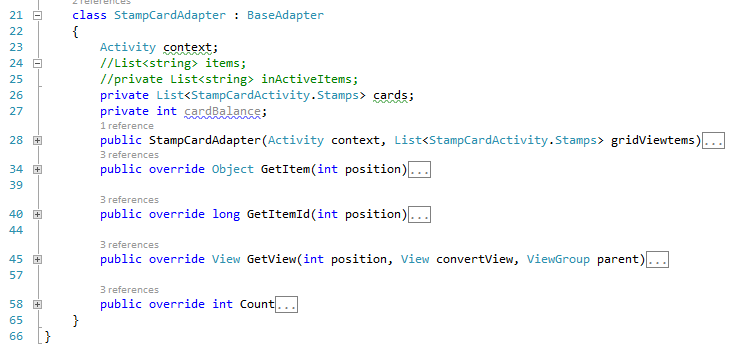
Oventående fremgang producerer følgende View, som vi har kaldt;

StampCardView.Axml



Ovenstående Billede til venstre, viser hvordan Layoutet ser ud uden data, og ovenstående billede til højre viser hvordan vores Vis StempelKort ser ud i App’en, og kan se hvordan vores FOR loops har produceret 5 cirkler og et billede af et stempel, dvs. at brugeren korrekt mangler 4 stempler endnu for at opnå sin bonus/gave.

Her er vores StampCardAdapter som sørger for ovenstående;



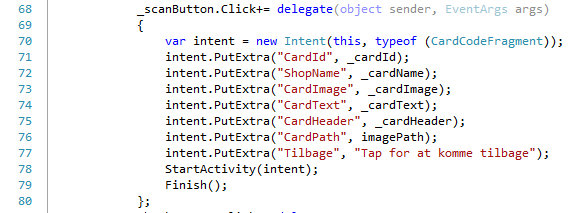
Vi har før, i User Story 5, Vis bruger information, vist hvordan en BaseAdapter fungerer, og vil ikke komme nærmere ind i ovenstående Kode.

## Implementering af User Story 11: Vis QR Kode / Opdater stempler

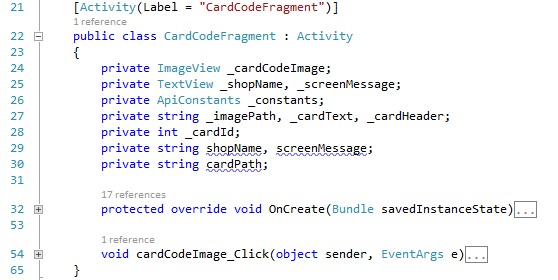
## Tasks:

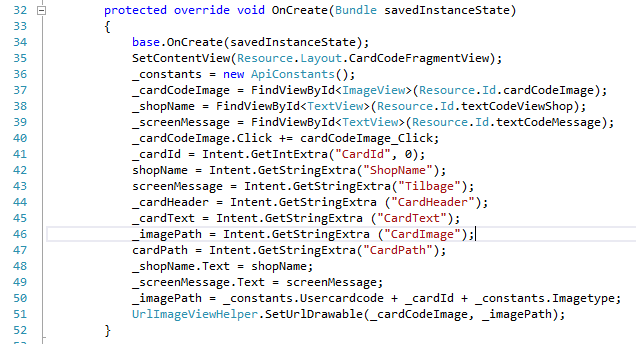
1. Intent
2. API kald
3. UrlImageViewHelper(LazyLoad)
4. Opdatering af stempler
5. User story 10

For at gennemføre ovenstående User Story, har vi gjort det simpelt, ved at sende data fra StampCardActivity til vores CardCodeFragment Activity, for at få vist et billede af en QR kode, som vi henter via et URL og viser i et ImageView.



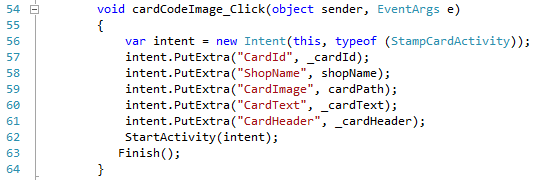
Fra linie 68-80 i StampCardActivity foretager vi et kald, når brugeren klikker på SCAN knappen og åbner CardCodeFragment som ser ud således;



Ovenstående Class er ansvarlig for at få vist brugeren den QR kode der er nødvendig for at kunne få scannet i butikken og opdatere sine stempler og evt. modtage en bonus, hvis denne er opnået.

Ovenstående kode producerer følgende i ZmartWallet App’en;

Denne QR kode kan nu scannes af butikken og når brugeren ”Tapper” på skærmen (linie 54-64)for at vende tilbage vil opdateringen af stempler automatisk blive kaldt af StampCardActivity og brugeren kan få vist hvor mange stempler der er optjent.



Opdatering af stempler er et metodekald til API, fra butikken, og vil blive registreret på brugerens konto så snart scanningen er foretaget, og denne opdatering bliver kaldt af os i App’en hvergang User Story 10 bliver startet, altså hver gang at StampCardActivity bliver kaldt, så foretager vi et kald til API’et for at hente de nyeste data og gemme i SQLite, og så hente data ud fra SQLite som vi kan bruge til at få vist en eventuel opdatering.



Linie 67 kalder følgende metode til API’et;



Ovenstående kald vil altid hente ny data ind fra API og enten indsætte ny data, eller opdatere, og når denne metode er færdig, vil antallet af stempler en bruger har optjent være opdateret, og blive vist korrekt i App’en, som bliver kaldt i StampCardActivity som vi har beskrevet i User Story 10, Vis kort.

## Implementering af User Story 12 : ZmartScanner App

## Tasks:

1. Scanne QR Kode på en ZmartWallet App
   1. Kald til Api
   2. WebView

Resolve It ønsker en simpel Scanner App, der skal være istand til at scanne en QR-Kode fra ZmartWallet Applikationen, og dermed kalde ZmartWallet API’et og kontrollere om QR-Koden er gyldig.

App’en har til formål, at en fysisk butik, feks. Baresso, skal kunne installere denne App på en af butikkens telefoner (eller flere), og skal bruges ved kassen af en kassemedarbejder, som kan bruge denne App til at Scanne en kundes QR-Kode og ZmartScanner App’en vil via et kald til API’et, gennem QR-Koden som indeholder et unikt ID til denne kunde, samt et url til API’et, butikkens ID, og sender denne QR-Kode afsted til API’et som klarer alt arbejdet gennem Stored Procedures i API’et, som vil returnere et svar til ZmartScanner App’en, om denne QR kode er gyldig, og hvis den er, komme med en meddelelse om at brugerens QR-Kode er gyldig og at transaktionen er gennemført.

Hvis kunden er Baresso, som før nævnt, vil denne meddelelse fortælle ZmartScanner, at kunden f.eks. har 5 ud 9 stempler registreret, og hvis kunden f.eks. har 9 ud af 9 stempler vil vedkommende være i stand til at få en gratis kop kaffe, og kassemedarbejderen kan yde den påkrævede service.

Da applikationen skulle være simpel, og kun udføre en specifik opgave, at scanne en QR-Kode og sende denne QR-Kode der er en url til ZmartWallet API, har vi valgt at bruge et WebView til at håndtere meddelelsen der kommer retur fra API, Zxing.Net.Mobile Library der er til at hente på Xamarin Component Library (Samme som Visual Studio Extensions) til at klare vores Scan af QR-Koder og WebViewClient til at håndtere load af WebView.

## Zxing.Net.Mobile

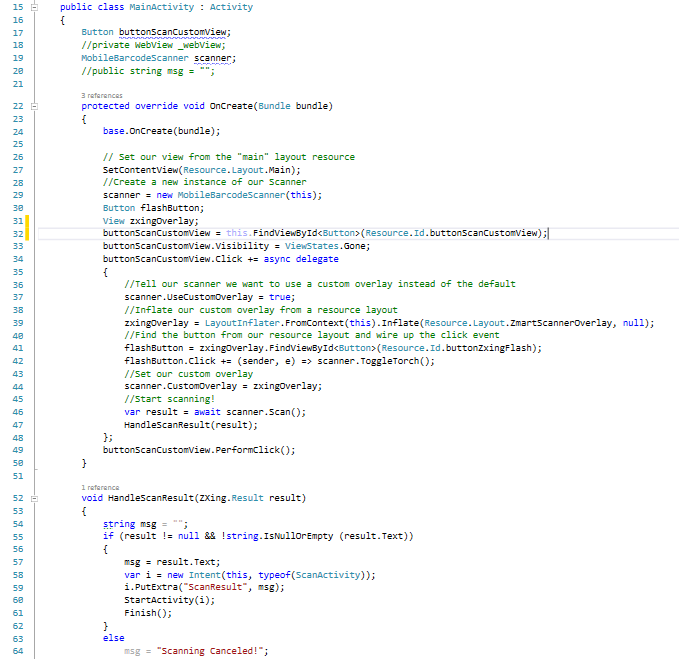
Dette library sørger for at scanne en QR-Kode, eller stregkode, og asynkront returnere et scannet resultat.

Vores brug af Scanner implementering er simpelt, og vi har valgt at bruge den i vores MainActivity, da brugeren (kasseassistent) ikke behøver logge ind, da dette foregår online via kaldet gennem API’et og login funktionen foregår i WebView’et, hvis dette er påkrævet.

Herunder er vores MainActivity Source Kode;

Vi har valgt at lade en knap starte vores Scanner funktion, men denne knap vil ikke være synlig for brugeren, da vi simulerer en knap tryk funktion for at starte Scanneren, hvergang MainActivity bliver kaldt, for at spare brugeren for at skulle starte Scanner funktionen, når en QR-Kode f.eks er blevet udført.

Denne simulering af knap-tryk foregår i linie 49, som starter en asynkron Scanner funktion fra linie 34 til linie 48, hvor scanneren vil ”lytte” til en succesfuld scan af en QR- eller stregkode og når linie 46 får et resultat, bliver denne sendt til vores HandleScanResult metode i linie 52.



Når HandleScanResult metoden får et resultat der ikke er null eller tomt, bliver dette resultat, som er en string, hvilket i vores tilfælde er en URL til ZmartWallet API, og sender denne URL videre til en Class der kan håndtere dette resutat.

### ScanActivity

Denne Class har til formål at modtage en URL fra MainActivity.HandleScan() metoden samt at håndtere denne.

Herunder en simpel implementering af et WebView;



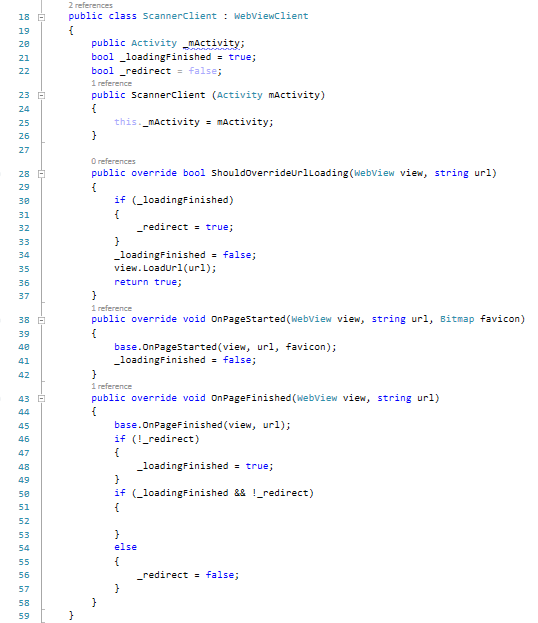
Overstående Class, modtager en String der kommer gennem MainActivity.HandleScanResult() som er sendt via det Intent der er startet derfra og indeholder den string vi skal bruge, som bliver hentet i linie 25, og vi har her den URL der skal bruges til at starte WebView’et med indhold (linie 30), og sætter WebViewClient til en Class vi har oprettet og kaldt ScannerClient som håndterer loading af Webviewet.

### 

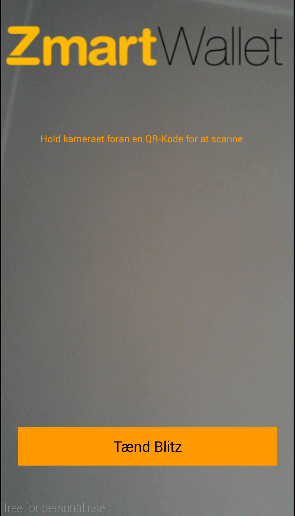
### ScannerClient

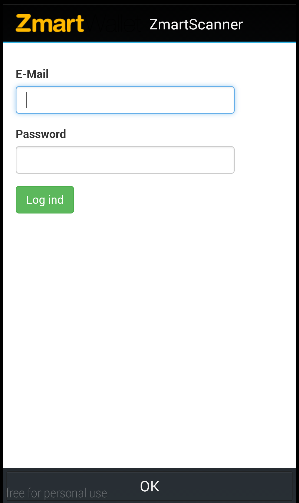
Herunder indholdet af denne ScannerClient Class, som klarer at loade indholdet af webviet, ved den URL der skal bruges, som kommer gennem ScanActivity.

Linie 28, 38 samt linie 43 håndterer URL.

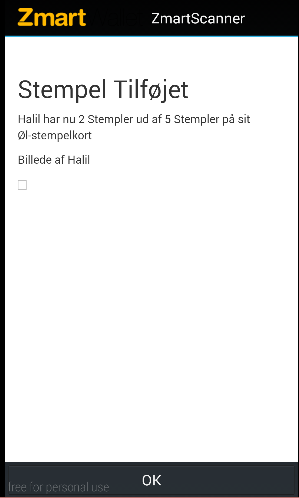


Da vi ikke skal sørge for yderligere håndtering af indholdet af dette WebView, er det ikke nødvendigt at implementere yderligere kode, da API sørger for at sende det indhold med der skal bruges. Denne Class benyttes til at loade indholdet som WebView’et skal vise.

Dette billede illustrerer hvordan Scanner App’en starter op, dvs. App’en søger/venter på at en QR Kode bliver vist foran den.



Når en QR Kode er scannet af butiks medarbejderen, vil App’en lave en kontrol og anmode Scanner App’en om at logge ind via Active Directory.

Når login funktionen er udført vil App’en kalde API’et via den URL der er i QR Koden og præsentere følgende skærm der oplyser medarbejderen hvor mange stempler en bruger har optjent



Og her er interaktionen der fortæller butiksmedarbejderen, at der skal udbetales en ”gave” da det sidste stempel er blevet brugt, og kortet vil nu i API’et være inaktivt og skal aktiveres igen af brugeren på sin smartphone.

## Evaluering af Sprint 4

Vi har nu implementeret de user user stories Sprint 4 indeholder og afholder Sprint Review med product owner som vil foretage acceptance test(s), og vil derefter internt afholde et Sprint Retrospective møde og afslutte sprint 4.

### Acceptance Test

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Projekt Identifikation** | | | | | | | | | | |
| Projekt Navn: | | | | | Projekt Start Dato: | | | | | |
| ZmartWallet | | | | | 20-04-2015 | | | | | |
| Product Owner: | | | | | | | | | | |
| Sinus Frank | | | | | | | | | | |
| Acceptance Kriterier Tests | | | | | | | | | | |
|  | | | | VigtigeFunktioner | | | Test Resultater | | |  |
| Nummer | | Accept kriterier  Beskrivelse | | Ja | | Nej | Accepteret | Afvist | | Kommentarer |
| 10 | | Vis StempelKort | |  | |  |  |  | |  |
| 11 | | Vis QR Kode / Opdater stempler | |  | |  |  |  | |  |
| 12 | | Scanner App | |  | |  |  |  | |  |
| Godkendt | | | | | | | | | | |
| Product Owner | | | Signatur | | | | | | Dato | |
| Ja | Nej | | Sinus Frank - Resolve-It | | | | | | 05-06-15 | |

## User Story 10: Vis Stempelkort

Bruger klikker på et spempelkort

Bruger kan se sit stempelkort med antal af stempler

## User Story 11: Vis QR Kode / Opdater Stempler

Bruger klikker på SCAN knappen

Bruger kan se sin QR Kode på kort

Bruger lukker QR Kode

Bruger kan se opdatering af stempler

## User Story 12: Scanner App

Bruger kan scanne QR Kode

Bruger kan logge ind på Scanner App

Bruger kan se optjente bonus/rabat

Bruger kan udbetale bonus/rabat hvis denne er opnået

### Sprint Retrospective

Alt forløb som det skulle, og vi kunne overholde tidsplanen og fremvise en godkendt acceptance test.

Vi måtte ændre en smule i koden her i løbet af sprintet, da API kaldet blev ændret i forbindelse med aktivering af stempelkort, som vi fik klaret i samarbejde med Alexander(API backed udvikler).

# Konklusion

Gennem forløbet af denne rapport har vi arbejdet på, at undersøge og give svar på vores problem definition.

### Spørgsmål 1

Vi er kommet frem til, at applikationer der tilhører Wallet genren er på vej frem. Der dukker flere og flere applikationer frem til SmartPhones, som tilbyder diverse former for digital opbevaring af brugernes fysiske kort. Der findes F.eks FidMe, just-Collect, Wallyy, Google Wallet og et utal af andre udbydere, der alle tilbyder digitial opbevaring af kort.

### Spørgsmål 2

Vi har udviklet ZmartWallet som differencierer sig fra de ”normale” Wallet app’s ved, at ZmartWallet er interaktivt, og der er fysisk interaktion mellem butik og brugeren Smartphone, og kan bruges som et betalingsmiddel ved, at brugeren kan opnå en bonus/rabat i form af stempelkort og stempler.

### Spørgsmål 3

Vi har opnået en registrering af optjening af stempler, når en bruger får scannet sin QR Kode på sit digitale kort i en fysisk butik, ved at benytte en online transaktion mellem ZmartScanner App’en og ZmartWallet API’et.

Spørgsmål 4

Ved at benytte Microsoft Azure Active Directory, har vi opnået en sikkerhed for brugeren og ZmartWallet, i form af en oprettelse af en unik bruger, en login funktion af denne bruger og dermed en validerings proces, der skal sikre at en bruger’s konto ikke kan blive misbrugt.

# Perspektivering

Vi er overordnet godt tilfredse med resultatet af denne rapport og vores fremgangsmåde med at løse opgaverne. Vi har forsøgt at være grundige med vores undersøgelse og analyser, forsøgt at holde os til Extreme Programming og arbejde agilt og udnytte Scrum til at styre udviklings processen, selvom vi er gået lidt udover XP ved at lave analyser og dokumentering, som ikke er en del af denne metodologi.

Under forløbet har vi dog forsøgt at gøre Resolve-It opmærksom på evt. fremtidige komplikationer, i form af sikkerhed og også i form af app’ens præstation, da det måske kræver for meget af en SmartPhone, at skulle gemme data lokalt i form af SQLite, og hente data fra denne, når de vigtigste kald der skal foretages, foregår gennem ZmartWallet API’et, føler vi at SQLite er overflødig, og skulle vi som udviklere fortsætte med at udvikle for Resolve-It, ville dette være noget af det første vi ville ændre på.

Vores interne samarbejde har været rigtig godt, og vi har fordelt arbejdsopgaverne godt og trukket på hinandens styrker og bidraget med viden og hjælp til hinanden, når dette har været påkrævet. Vi har opnået at præstere at aflevere ikke kun en men to Android App’s til Resolve-It, som kan i deres nuværende form blive udgivet hos Google Play, hvilket vi selv er stolte over at have præsteret.

Vi har fået bekræftet overfor os selv, at vi kan samarbejde med andre udviklere, der arbejder med andre platforme end os selv, Alexander der har Java baggrund, Kewin med IOS baggrund, Sinus der er database Administrator og os selv, Mark og Halil med C# baggrund, og fået dette samarbede til at fungere og i sidste ende producere to App’s der interagerer med hinanden, og måske dermed været med til at udvikle den første generation af denne type Wallet App, hvilket vi er meget spændte over, og vil med stor interesse følge med i hvordan ZmartWallet bliver modtaget af butikkerne og brugerne.

1. <http://www.dibs.dk/dansk-ehandel> [↑](#footnote-ref-1)
2. <http://upedu.org/> [↑](#footnote-ref-2)
3. <http://agilemanifesto.org/> [↑](#footnote-ref-3)
4. <https://msdn.microsoft.com/en-us/library/hh273032%28v=vs.88%29.aspx> [↑](#footnote-ref-4)
5. <http://www.extremeprogramming.org/> [↑](#footnote-ref-5)
6. <http://developer.xamarin.com/> [↑](#footnote-ref-6)
7. <http://www.lean.org/WhatsLean/Principles.cfm> [↑](#footnote-ref-7)
8. <https://www.scrumwise.com/> [↑](#footnote-ref-8)
9. Bilag 1 Projekt Tidsplan [↑](#footnote-ref-9)
10. <http://www.dst.dk/pukora/epub/upload/18686/itbef.pdf> [↑](#footnote-ref-10)
11. <http://www.dibs.dk/dansk-ehandel> [↑](#footnote-ref-11)
12. <http://www.dibs.dk/dansk-ehandel> [↑](#footnote-ref-12)
13. <http://www.dibs.dk/dansk-ehandel> [↑](#footnote-ref-13)
14. <http://www.mobilsiden.dk/nyheder/roskilde-festival-dropper-betalingsarmband-overvejer-istedet-mobilen,lid.31757/> [↑](#footnote-ref-14)
15. Bilag 1 ReSolveIT Konkurrentanalyse [↑](#footnote-ref-15)
16. <http://www.fidme.com/en> [↑](#footnote-ref-16)
17. Agile Software Engineering with Visual Studio, second edition [↑](#footnote-ref-17)
18. Agile Software Engineering with Visual Studio, 2nd Edition [↑](#footnote-ref-18)
19. Bilag 3 User Stories der ikke indgår i implementeringen. [↑](#footnote-ref-19)
20. <http://azure.microsoft.com/da-dk/services/active-directory/> [↑](#footnote-ref-20)
21. <http://jsonlint.com/> [↑](#footnote-ref-21)