

# Undersökning av olika tekniker för tillgång till native funktionalitet i mobila enheter



**LUNDS UNIVERSITET**  
Campus Helsingborg

LTH Ingenjörshögskolan vid Campus Helsingborg  
Institutionen för datavetenskap

Examensarbete:  
Hamza Abdulilah

© Copyright Hamza Abdulilah

LTH Ingenjörshögskolan vid Campus Helsingborg  
Lunds universitet  
Box 882  
251 08 Helsingborg

LTH School of Engineering  
Lund University  
Box 882  
SE-251 08 Helsingborg  
Sweden

Tryckt i Sverige  
Lunds universitet  
Lund [2017]

## Sammanfattning

Detta examensarbete har utförts i samarbete med Cenito Software AB, ett mjukvaruföretag baserat i Malmö. Cenito har en befintlig responsiv site, en SPA (Single Page Application). En responsiv site innebär att denna anpassar storlek och utseende till enheten som försöker få åtkomst till sidan. Denna site är en bas eller första byggsten för andra applikationer som utvecklas av Cenito. Detta examensarbete undersöker möjligheten att optimera sidan för att få den fungera på mobila plattformar, för att exempelvis utnyttja hårdvara som accelerometrar, kamera eller andra funktioner. Följande tre exempel är intressanta för företaget:

- (i). Webbapplikation - Responsiv webbapplikation som fungerar i mobila webbläsare.
- (ii). Webb-Native applikation - Ramverk som Apache Cordova används för att få HTML5 & CSS3 kod från alternativ i, att fungera på en smartphone i form av en webb-native applikation.
- (iii). Native Applikation- Bygga en native applikation från grunden för den specifika mobila plattformen.

Examensarbetet resulterade i en framgångsrik konvertering av företagets webbapplikation till en webb-native applikation. Med hjälp av denna konvertering blev det möjligt att få åtkomst till accelerometrar, kamera, funktionalitet för skanning av QR-koder och push notiser och även implementera dessa funktioner. Det resulterade även i en Android prototyp med skanning funktionalitet för att jämföra denna prototyp med webb-native alternativet, för att kunna ge rekommendationer när ett av alternativen ska väljas och varför. Slutsatsen är att det beror på storleken på företaget, vilka funktioner företaget vill implementera samt vilken marknad man vill nå.

Nyckelord: Webbapplikation, ramverk, mobil plattform, Single Page Application, native funktioner.

## Abstract

This thesis project has been conducted in collaboration with Cenito Software AB, a software-developing company based in Malmö. Cenito has currently a responsive website, a SPA (Single Page Application). A responsive website implies that the size and appearance adapts to the unit that is trying to access the site. This site is being used by Cenito as a basis or first building block for other application that is developed by the company. This project researches the possibility of optimizing the website to run it on a mobile platform, to access hardware like accelerometer, camera or other capabilities. There are three alternatives that the company are interested in:

- (i). Webb application – Works in regular browsers and mobile browsers.
- (ii). Webb Native Application – The use of a framework like Cordova, to convert a SPA into a mobile function.
- (iii). Native Application – Making an application from scratch for a specific mobile OS.

The thesis resulted in a successful conversion of the company's web application to a web-native application. This conversion enabled access to accelerometer, camera, QR-code scanning functionality and push notifications. These features were also implemented into the web-native application. The project also resulted in an Android prototype with scanning functionality to compare this prototype with the web-native option, to provide recommendations to guide the reader when to select one of the options and why. The conclusion is that it depends on the size of the company, what features the company wants to implement and what market one wishes to reach.

Keywords: Webbapplication, framework, mobile platform, Single Page Application, native functions.

## Förord

Ett stort tack till Johan Nilsson och Richard Houltz från Cenito för deras gästvänlighet, för all vägledning från även andra anställda och för möjligheten att utföra detta arbete trots en hektisk period för företaget. Jag vill även rikta ett stort tack till handledare Christian Nyberg och examinator Christin Lindholm för stöttning och hjälp för ett lyckat arbete.

# Innehåll

---

1	Inledning .....	1
1.1	Bakgrund .....	1
1.2	Syfte och Målformulering .....	1
1.3	Problemformuleringar .....	2
1.4	Motivering av Examensarbetet .....	2
1.5	Avgränsningar .....	2
2	Nulägesbeskrivning & Teknisk bakgrund .....	5
2.1	Webbapplikation .....	5
2.1.1	Tekniker för webbutveckling .....	5
2.1.2	JavaScript .....	6
2.1.3	TypeScript .....	8
2.1.4	Angular .....	9
2.2	Webb-native .....	10
2.2.1	Apache Cordova .....	11
2.2.2	Arkitekturen hos en Cordova-applikation .....	12
2.2.3	Cordova plugins .....	13
2.2.4	Plugins struktur .....	13
2.2.5	Visual Studio .....	14
2.2.6	Tools for Apache Cordova .....	15
2.2.7	Ionic2 .....	16
2.3	Native Prototyp .....	18
2.3.1	Android Studio .....	18
2.3.2	Android Aktiviteter .....	20
2.3.3	Google API - Play Services för Android .....	22
2.3.4	Google API Visions Package – Visuell identifiering av objekt .....	23
3	Metod och Analys .....	25
3.1	Tidsplan .....	25
3.2	Arbetsmetod – Inspiration från Scrum .....	26
3.3	Källkritik .....	27
3.3.1	Primära källor .....	27
3.3.2	Sekundära Källor .....	28
3.4	Analys – Verktyg och Arbetsprocess .....	28
3.5	Analys – Utveckling .....	29
4	Resultat .....	31
4.1	Webbapplikation .....	31
4.2	Webb-native – Cordova & Ionic .....	31
4.3	Native Prototyp .....	33
5	Slutsats .....	35
5.1	Reflektion över etisk aspekt - Sekretess .....	36
5.2	Möjligheter till vidareutveckling .....	37
6	Terminologi .....	39
7	Källförteckning .....	41
8	Appendix A Webb-native plugins .....	45
8.1	QR-kod skanning .....	45
8.2	Kamera – Tagning av foto .....	45
8.3	Filer – Hämtning av en JPG fil .....	45
8.4	Push Notiser .....	46
9	Appendix B Android Prototyp .....	47
9.1	Java-fil .....	47
9.2	XML-fil .....	49

# 1 Inledning

---

## 1.1 Bakgrund

Utvecklingen av applikationer kan göras på olika sätt i dagsläget, till exempel som de tre olika alternativen presenterade avsnitt 1.2 Syfte och målformulering. Detta examensarbete kommer kunna leda till att företaget Cenito eller andra företag kommer kunna välja det bästa alternativet för deras applikation. Examensarbetet kommer även kunna gynna Cenito för framtida utveckling av applikationer. Cenito Software är ett mjukvaruföretag baserat i Malmö som utvecklar mobil- och webbapplikationer med cirka 25 anställda. De har mycket erfarenhet inom utveckling av applikationer för maskiner och automationssystem. En av deras kunder är Alfa Laval som är en världsledande tillverkare av pumpar, ventiler och andra typer av maskiner.

## 1.2 Syfte och Målformulering

Syftet med examensarbetet är att undersöka de tre olika alternativen för applikationsutveckling för Cenitos webbapplikation samt utveckla en native prototyp:

- (i). Webbapplikation - Responsiv webbapplikation som fungerar i mobila webbläsare.
- (ii). Webb Native applikation - Använda ramverk som Apache Cordova, för att få HTML5 & CSS3 kod från alternativ ett, att fungera på en smartphone i form av en webb-native applikation samt åtkomst till native funktionalitet i form av plugins.
- (iii). Native Applikation- Bygga en native applikation från grunden för den specifika mobila plattformen.

Cenito har en befintlig responsiv webbsida som fungerar på mobila plattformar, det vill säga alternativ (i) som en applikation direkt i webbläsaren. Det är en specifik site som är en bas eller första byggsten för andra applikationer som Cenito bygger. Detta examensarbete ska hjälpa Cenito att avgöra vilket sätt som är det bästa, vilket alternativ som ger flest möjligheter för deras specifika fall. Detta avgörs utifrån fördelar kontra nackdelar för de tre olika sätten för exempelvis:

- Tillgång till hårdvara t.ex. kamera eller accelerometer.
- Skanning av exempelvis QR-koder.
- Storage (Tillgång till befintliga filer i mobilen).
- Push notiser.
- Kommunikation med andra applikationer.
- Begränsningar med de olika alternativen. Vilka funktioner av ovanstående som kan implementeras eller inte.

Målet är att utifrån undersökningen av alternativ i, ii och iii kunna presentera rekommendationer som förklarar vilket alternativ som är bäst Cenitos webbapplikation. Det ska visas utvärderingar varför vissa alternativ bör väljas eller inte väljas, beroende på applikationens funktionalitet och syfte. En prototyp av en applikation ska utvecklas för att kunna jämföra åtkomst av hårdvara som kamera, Bluetooth och annat nämnt tidigare i detta dokument.

## 1.3 Problemformuleringar

Problemformuleringen är uppdelad i olika delar som rör de olika alternativen för applikationsutveckling. Då Cenito har en befintlig webbapplikation så berör problemformuleringen möjligheterna att bevara så mycket funktionalitet som möjligt. Vilket innebär att undersöka hur mycket av källkoden som kan bevaras och överföras med hjälp av ramverken Cordova eller Ionic2 och sen jämföra detta med någon native prototyp som innehåller viss funktionalitet. Det resulterade i dessa problemformuleringar:

1. Hur ser marknaden ut idag, finns det företag som valt endast ett av alternativen eller har de en kombination? (t.ex. Facebook, Instagram eller annat bättre exempel)
2. Vilken befintlig funktionalitet kan återanvändas eller måste man börja om från början?
3. Vad är det för skillnad mellan alternativ webbapplikation, webb-native och native?
  - a. Vad tillför webb-native ramverk som en webbapplikation inte kan?
  - b. Vad tillför native alternativet jämfört med webb-native?
4. Vilka webb-native ramverk finns det som är jämförbara med Cordova?
5. Vilken funktionalitet ska prototypen ha?
  - a. Hur kommer webb- och webb native applikationerna åt denna funktionalitet i alternativ i, ii, och iii då?

## 1.4 Motivering av Examensarbetet

Motivet för Cenito är att undersöka möjligheten för åtkomst av native funktionalitet samtidigt som källkoden för nuvarande webbapplikationen kan behållas. Andra företag kan dra nytta av detta arbete då tre olika alternativ kommer undersökas och jämföras med varandra för att dra slutsatser och rekommendationer. Företag kan då läsa detta arbete och välja ett alternativ baserat på vad för applikation som de försöker utveckla och vilka funktioner de försöker få åtkomst till. Anledningen till att jag valde att genomföra detta examensarbete är mitt intresse för utveckling av applikationer. Det framtida målet är att vara en utvecklare med färdigheter inom applikationsutveckling för flera plattformar. Exempelvis utveckling av webb- och Android-applikationer och detta examensarbete var rätt steg mot detta mål.

## 1.5 Avgränsningar

Prototyp för en applikation ska utvecklas endast för ett operativsystem. Det valda operativsystem som valts för prototypen är Android. Därför att Android enheter är oftast billiga och har därmed många användare. Prototypen kommer också endast ha en av native funktionerna vilket är kameran för att utnyttja skannings funktionalitet. Denna funktionalitet kommer jämföras med ett skannings-plugin i alternativ (ii). Endast denna funktionalitet kommer jämföras mellan alternativen för att det är redan känt att en native prototyp eller applikation, har tillgång till allt som en plattform (Android i detta fall) har att erbjuda. Då räcker det med att välja en typ av funktionalitet för att kunna jämföra, svara på frågeställningar samt dra slutsatser. Cordova tillhandhåller ett antal så kallade core plugins för native funktioner. Alla core plugins kommer inte testas då det är ett stort antal plugins. Utvalda core plugins kommer undersökas tillsammans med det tidigare nämnda skannings-plugin. Rapporten kommer hålla



sig till dessa för alternativ (ii) och inga andra kommer att undersökas. All exekvering av programkod kommer göras på en enhet med Android operativsystem.



## 2 Nulägesbeskrivning & Teknisk bakgrund

---

Detta kapitel beskriver de program och verktyg som har använts för att genomföra examensarbetet. Beskrivningen är indelad i tre delar som representerar de tidigare diskuterade alternativen för utveckling:

- (i). Webbapplikation - Responsiv webbapplikation som fungerar i mobila webbläsare.
- (ii). Webb Native applikation Ramverk som Apache Cordova används för att få HTML5 & CSS3 kod från alternativ ett, att fungera på en smartphone i form av en webb-native applikation.
- (iii). Native Applikation- Bygga en native applikation från grunden för den specifika mobila plattformen.

### 2.1 Webbapplikation

Första alternativet för utveckling som undersökts i detta arbete är webbapplikation. Tekniker för utveckling av webbapplikationer kommer beskrivas, tillsammans med ett ramverk som Cenito har valt att använda.

#### 2.1.1 Tekniker för webbutveckling

För att kunna utveckla webbapplikationer eller webbsidor krävs det specifika HTML-dokument samt förståelse för hur dessa dokument skrivs. HTML paras väldigt ofta ihop med ett CSS dokument.

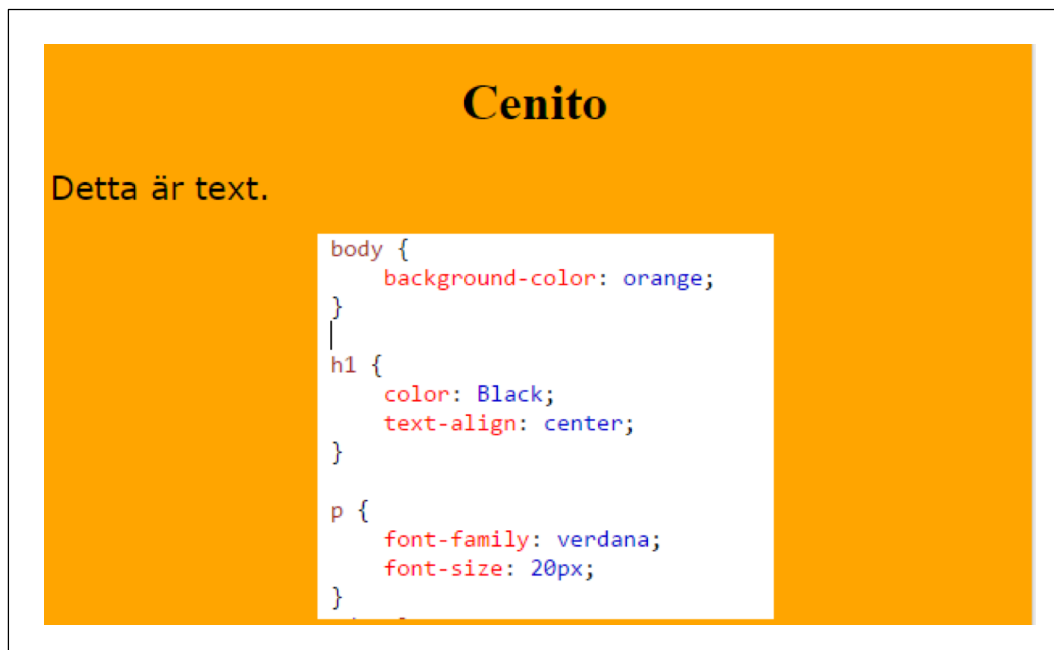
HTML står för HyperText Markup Language och är det språk som används för att skriva HTML-dokument, dessa dokument resulterar i webbsidor efter överföring via internet. Strukturen på dokumenten bestäms av taggar. Dessa taggar läses in av webbläsare för att veta hur innehållet ska visas för användare. I dessa taggar så skrivs innehållet som ska visas som i figur 1. [1]

```
<html>
<head>
<title>Titeln för dokumentet</title>
</head>

<body>
Innehållet i dokumentet som är intressant för användaren.
Kan vara bilder, paragrafer av text etc.
</body>
</html>
```

Figur 1 visar exempel på hur taggar används i ett HTML-dokument.

Dessa taggar kan manipuleras av något som kallas CSS, Cascading Style Sheets. Med hjälp av CSS-dokument går det att bestämma hur innehållet i ett HTML-dokument ska presenteras. Det går exempelvis att ändra typsnitt, färgen på typsnittet eller bakgrunden som figur 2. HTML- och CSS-dokument passar därför ihop. Tack vare CSS finns det möjlighet att göra HTML dokument anpassningsbara beroende på vilken enhet som öppnar det, vare sig det är en mobilenhet eller en PC. Det går att använda sig av CSS direkt i en HTML-fil genom att använda taggen <style> som alternativ till exemplet i figur 2. [2]

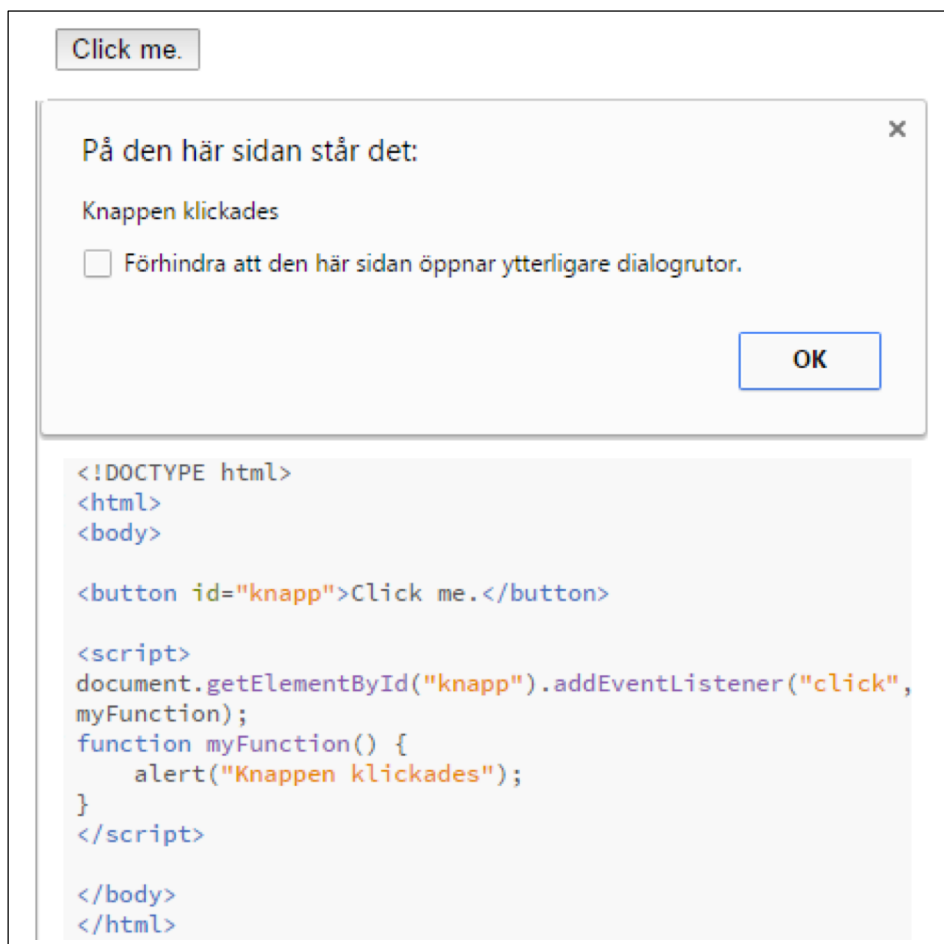


Figur 2 visar hur ett CSS-dokument (vita i figuren) ändrar på HTML-taggaras egenskaper. [2]

### 2.1.2 JavaScript

JavaScript skapades 1995 av en ingenjör som heter Brendan Eich som jobbade på företaget Netscape. Första versionen släpptes 1996 och var tänkt att det skulle heta LiveScript. Men ändrades till JavaScript på grund av ett marknadsföringsbeslut. Några månader efter detta beslut släppte Microsoft JScript med deras browser Internet Explorer 3.

JavaScript är ett skriptspråk som används i samband med webbutveckling. Skriptspråk är ett programmeringsspråk som har stöd för skripts. Dessa skripts är mindre program som utför något som en person i praktiken inte kan utföra. T.ex. uppdatering av en webbsida varannan minut i all evighet. Skripts kan skapas för användning i en webbsite, som möjliggör interaktion med användare på webbsiten. Det kan vara skript som bestämmer hur en webbsida ska agera beroende på vad en användare gör, för att skapa en mer användarvänlig upplevelse. Ett exempel är figur 3 som visar på en händelse efter ett knapptryck. Kod kan läggas till med taggen <script> som exemplet visar, men för större skript brukar det vara en egen JS-fil. [39]



Figur 3 visar hur man kan använda JavaScript kod för att webbsidan ska svara på användarens knapptryck. [Skärmdump]

JavaScript är ett dynamiskt språk. Det innebär t.ex. att variabler inte kan typdefineras och kan då referera till vilka typer som helst. Det vill säga att samma variabel kan referera till ett tal, för att senare referera till en bit text.

```
var variabel = 10;

variabel = "Nu refererar variabeln till denna texten istället";
```

Figur 4 visar på hur JavaScript är ett dynamiskt språk.

Liknande kodrader som visas i figur 4 hade inte fungerat för ett icke-dynamiskt språk. Ett sådant språk är Java. Där alla variabler har specifika typer och en IDE hade inte tillåtit kompilering av ett program med kodrader som påminner om kodraderna ovan. [3]

### 2.1.3 TypeScript

Ett annat programmeringsspråk som används inom webbutveckling är TypeScript. Det är ett språk utvecklat av Microsoft och Anders Hejlsberg, som också är huvudarkitekten för C#. TypeScript är ett superset av JavaScript. Det innebär att TypeScript innehåller allt som ingår i JavaScript, men med extra funktionalitet. Detta betyder också att existerande JavaScript-program, fungerar lika bra under utveckling i TypeScript. [6]

En av TypeScript's funktioner är att det kan ge utvecklaren möjligheten att utveckla statisk programkod, till skillnad från dynamisk som tidigare nämnt i JavaScript. De grundläggande typerna som finns är number, boolean och string som visas i figur 5 för en class *Customer*.

```
class Customer {
  private name: string;
  private phone: number;
  private email: string;

  constructor(name: string, phone: number, email: ) {
    this.name = name;
    this.phone = phone;
    this.email = email;
  }
}
```

Figur 5 visar på ett exempel hur en TypeScript klass kan skrivas för en Customer (kund). [4]

TypeScript ger mer objektorienterade alternativ. Det går att skriva programkod med konstruktorer och attribut bundna till klassen på ett sätt som påminner om strikt objektorienterade programspråk. Dagens webbläsare kan inte tolka TypeScript utan all TypeScript kod kompileras till JavaScript för att sedan kunna köras på webbläsare.

Fat Arrow functions är ett verktyg som utvecklare kan använda som inte är möjligt i JavaScript. Det ett annat sätt att implementera funktioner och metoder på i TypeScript.

```
this.growOld = function(){          this.growOld = () => {
  this.age++;                          this.age++;
}
```

Figur 6 visar exempel på funktion i JavaScript respektive Fat Arrow funktion i TypeScript. [5]

Funktionerna i figur 6 ger exakt samma resultat. TypeScript funktionen till höger ser dock annorlunda ut. Denna typ av funktion indikerar på att allt till vänster om "=>" är argument eller in-parametrar och allt till höger är koden som utförs eller vad metoden skickar tillbaka. Exemplet ovan visar att funktionen inte har några argument och skickar inte tillbaka något

värde. Den utför endast en instruktion där åldern (age), som är ett attribut som vi kan anta redan finns i klassen, adderas med ett. Om koden istället såg ut som figur 7:

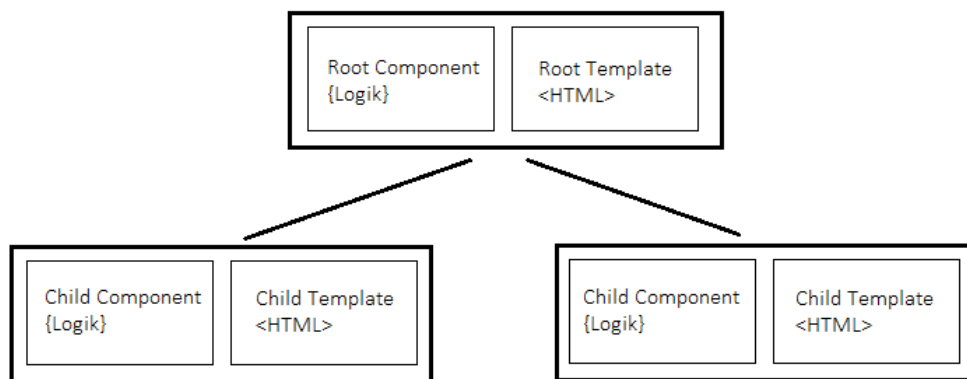
```
this.growOld = (age: number) => { return age++;}
```

Figur 7 visar på ännu ett exempel för Fat Arrow funktion

Här kräver funktionen ett argument (in parameter) av typen number som ska representera åldern. Denna ålder adderas och skickas sedan att sedan kunna tilldelas till growOld attributet. [5]

### 2.1.4 Angular

Angular är ett ramverk som Cenito har använt för att utveckla sin webbapplikation. Det är ett ramverk som är TypeScript-baserat för front-end-utveckling. Det skapades av Google med hjälp av utomstående företag. Utveckling av en webbapplikation i Angular sker med HTML, CSS och TypeScript. Angular ger utvecklare verktygen för att skapa "Single Page Applications". Dessa applikationer består endast av en HTML fil, men med flera byggblock som laddas in i webbapplikationen beroende på vart användaren navigerar sig. [7]



Figur 8 visar byggblocken för en Angular applikation. En template är en mindre del av applikationen som kontrolleras av en komponent. [7]

En applikation utvecklad i Angular består av byggblock som innehåller logik och en vy för byggblocket. Byggblocket innehåller en så kallad komponent för logiken och en template för vyn. En komponent innehåller logik som kontrollerar en mindre del av applikationen, skrivet i en TypeScript-klass. En template är en HTML-fil som definierar hur Angular-ramverket ska visa komponenten för en användare. Ramverket visar byggblocken beroende på var en användare befinner sig i applikationen. Om en användare befinner sig på byggblock A men klickar på byggblock B, så förstörs A och nya byggblock B visas istället. Figur 9 och 10 utgör tillsammans ett exempel på ett byggblock.

```

//egen TypeScript-fil (.ts)
export class HeroListComponent implements OnInit {
  heroes: Hero[];
  selectedHero: Hero;

  constructor (private service: HeroService){}

  ngOnInit(){
    this.heroes = this.service.getHeroes();
  }

  selectedHero(hero: Hero) { this.selectedHero = hero; }
}

```

Figur 9 visar på en komponent som innehåller en lista med superhjältar. [7]

```

//egen template HTML-fil
<h2>Hero List</h2>
<p><i>Pick a hero from the list</i></p>
<ul>
  <li *ngFor="let hero of heroes" (click)="selectHero(hero)">
    {{hero.name}}
  </li>
</ul>
<hero-detail *ngIf="selectedHero" [hero]="selectedHero"></hero-
detail>

```

Figur 10 visar på en template som låter en användare klicka och välja bland superhjältar som finns i den tidigare visade komponenten. [7]

## 2.2 Webb-native

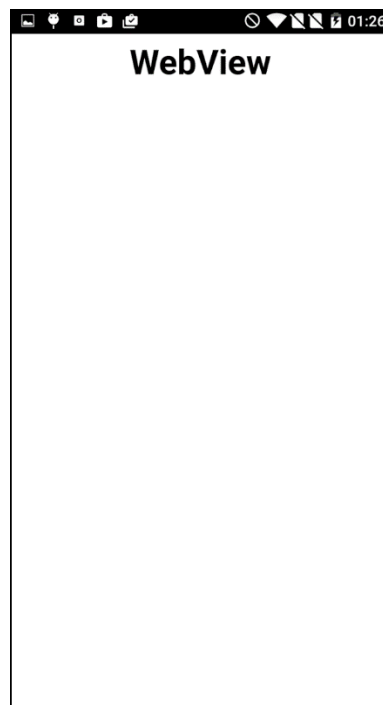
Det andra alternativet som undersöks i detta arbete är webb-native applikation med två olika ramverk. Dessa ramverk ska ge tillgång till native-funktioner som exempelvis skanning av koder, accelerometer och kamera. Även program och verktyg som användes för att utveckla med ramverket beskrivs i detta avsnitt.



### 2.2.1 Apache Cordova

Apache Cordova är ett ramverk för utveckling av applikationer för mobiler skapat av ett företag som heter Nitobi. Adobe systems köpte upp detta företag och lanserade sedan en version av ramverket som heter Phonegap. En version av Phonegap med öppen källkod släpptes också och nämndes Apache Cordova, vilket är den version som används för webb-native undersökningen i examensarbetet.

Apache Cordova är ett utvecklingsramverk med öppen källkod som tillåter användandet av webbt teknologi som till exempel HTML, CSS och även JavaScript för att utveckla applikationer till iOS, Android och även Windows Phone. Detta fungerar genom att en Cordova-applikation körs genom så kallade wrappers, för de specifika plattformarna. Dessa wrappers lindar in koden och visar den i en native WebView. Se figur 11 för hur en WebView ser ut. [8]



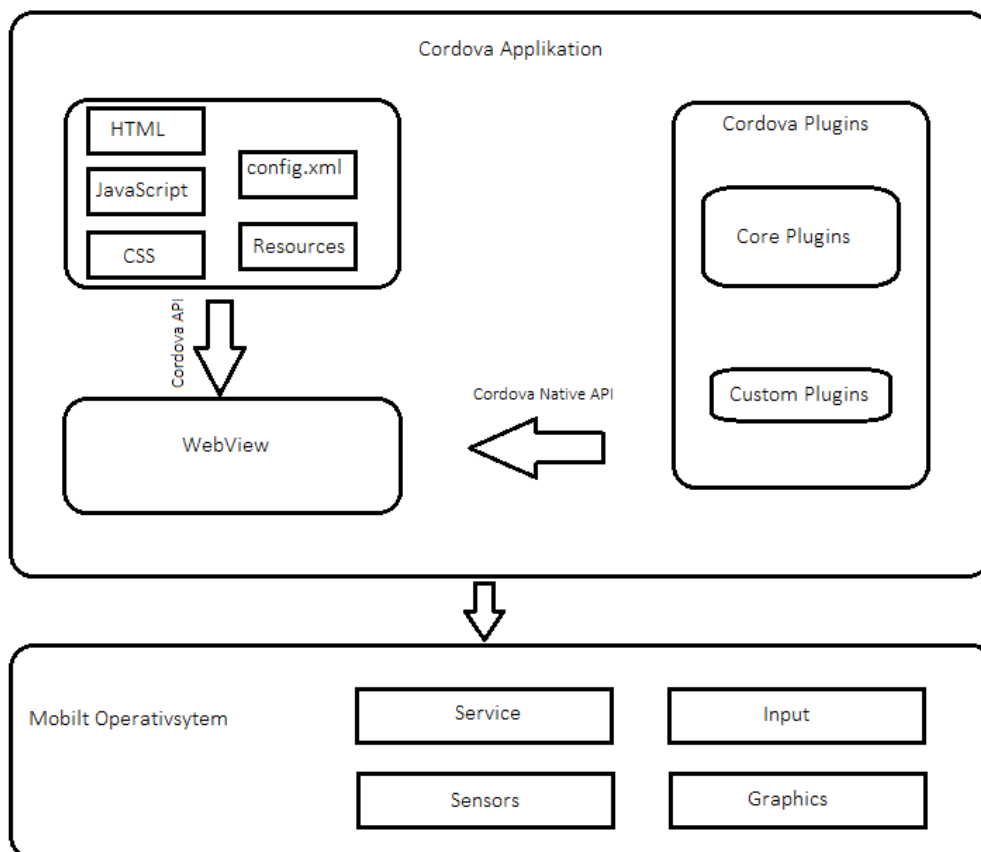
*Figur 11 visar på hur en WebView ser ut vid exekvering i en Android enhet. Innehållet som visas i denna WebView är en HTML-fil med "WebView" som titel. [Skärmdump]*

WebView är en vy som kan användas för att bädda in innehåll från webben. I en WebView ger Cordovas API: er tillgång till telefonens sensorer, filer, nätverk och liknande. Detta ramverk kan användas för att utveckla applikationer till mer än en plattform eftersom samma källkod i en Cordova applikation ska fungera oberoende om det är iOS eller Android. Det finns även en teoretisk möjlighet att paketera en befintlig webbapplikation så den kan exekveras i mobilen, med syftet att få tillgång till native funktionalitet som enhetens kamera. [8]

En Cordova-applikation utvecklas som om det vore en webbsida, med en index.html-sida som första sida med referenser till CSS, JavaScript och andra resurser som behövs. Hur ramverket installeras och kan användas förklaras i sektion 2.2.6.

### 2.2.2 Arkitekturen hos en Cordova-applikation

En Cordova-applikation består av två delar som exekveras i en WebView. Den första delen är programkoden för webbapplikationen. Den andra delen består av Cordovas plugins som exekverar native funktionalitet vid anrop i webbapplikationen. Se figur 12 för ett perspektiv ur högnivå.



Figur 12 visar hur Cordova kommunicerar med enhetens operativsystem och plugins. [8]

Vid skapandet av ett Cordova projekt så initialiseras ett antal filer. Men bara en del av dessa filer är vitala för ett fungerande projekt. De vitala filerna för ett fungerande projekt visas i figur 13 och är följande:

- *Config.xml* är filen som innehåller alla inställningar för applikationen. Vilka plugins som är installerade, vilka plattformar som exempelvis iOS eller Android det kan köras på och vilken version av Cordova som applikationen använder.
- Katalogen *Platforms* innehåller filerna för varje plattform som applikationen kommer köras på.
- Katalogen *www* innehåller webbapplikationen
- Plugins innehåller filer som krävs för att kunna nyttja native funktioner. Vilka plugins som är installerade befinner sig i denna mapp.

bin	2017-04-11 00:05	Filmapp
bld	2017-04-11 00:05	Filmapp
merges	2017-04-11 00:03	Filmapp
platforms	2017-04-11 00:05	Filmapp
plugins	2017-04-11 00:05	Filmapp
res	2017-04-11 00:03	Filmapp
simulation	2017-04-11 00:05	Filmapp
www	2017-04-11 00:16	Filmapp
	2017-04-11 00:03	Textdokument
bower	2017-04-11 00:03	JSON-fil
build	2017-04-11 00:03	JSON-fil
CenitoClientApp.jsproj	2017-04-11 00:03	JavaScript App Pr...
CenitoClientApp.jsproj.user	2017-04-11 00:06	Per-User Project O...
changeList	2017-04-11 00:18	JSON-fil
config	2017-04-17 18:01	XML-fil
package	2017-04-11 00:03	JSON-fil

Figur 13 visar på vilka filer en Cordova applikation består av. [Skärmdump]

### 2.2.3 Cordova plugins

Plugins är en vital del av ramverket Cordova. Då det är plugins som tillför ett användargränssnitt mellan Cordova WebView och native komponenterna eller andra funktioner för den mobila enheten. Detta gränssnitt är det som tillåter utvecklaren att anropa nativekod genom JavaScript. Cordova förser utvecklare med ett antal "core plugins" som ger tillgång till enheters hårdvara som kamera, filer och skanning av QR-koder.

Förutom Cordovas egna core plugins finns det många andra plugins från tredje parter för mer specifika funktioner. Detta innebär då också att utvecklare kan skapa sina egna plugins och använda dessa i sina Cordova-applikationer. Plugins från tredje part kan hittas i exempelvis GitHub. Det finns även plugin-dokumentation för att hjälpa utvecklare skapa egna plugins ifall utvecklare själv vill skapa för en speciell funktion. Hur Cordova plugins installeras förklaras i avsnitt 2.2.6 av dokumentet. [9]

### 2.2.4 Plugins struktur

Alla installerade plugins finns i katalogen "plugins". Följande filer innehåller plugin-funktionalitet, se figur 14:

- "Src" katalogen innehåller native källkod för specifika operativsystem som anropas genom JavaScript, för att få tillgång till funktionaliteten beroende på vilket operativsystem som använts. Källkoden är uppdelad i olika mappar där ena mappen är för iOS och den andra för Android.
- "Tests" katalogen innehåller färdigt test för plugin funktionalitet. Endast nödvändig ifall användare av plugin vill testa innan början på programmering av egen applikation.
- "www" katalogen innehåller JavaScript koden med alla metoder som används för att anropa native källkod som finns i "Src". Detta är gränssnittet som utvecklare ska använda sig av under utvecklingen. [10] [11] [12] [14]

doc	2017-04-24 00:46	Filmapp	
src	2017-04-24 00:46	Filmapp	
tests	2017-04-24 00:46	Filmapp	
types	2017-04-24 00:46	Filmapp	
www	2017-04-24 00:46	Filmapp	
CONTRIBUTING	2017-04-24 00:46	MD-fil	2 kB
LICENSE	2017-04-24 00:46	Fil	12 kB
NOTICE	2017-04-24 00:46	Fil	1 kB
package	2017-04-24 00:46	JSON-fil	2 kB
plugin	2017-04-24 00:46	XML-fil	6 kB
README	2017-04-24 00:46	MD-fil	9 kB
RELEASENOTES	2017-04-24 00:46	MD-fil	10 kB

Figur 14 visar på filerna för en specifik plugin. [Skärmdump], [10], [11], [12], [14]

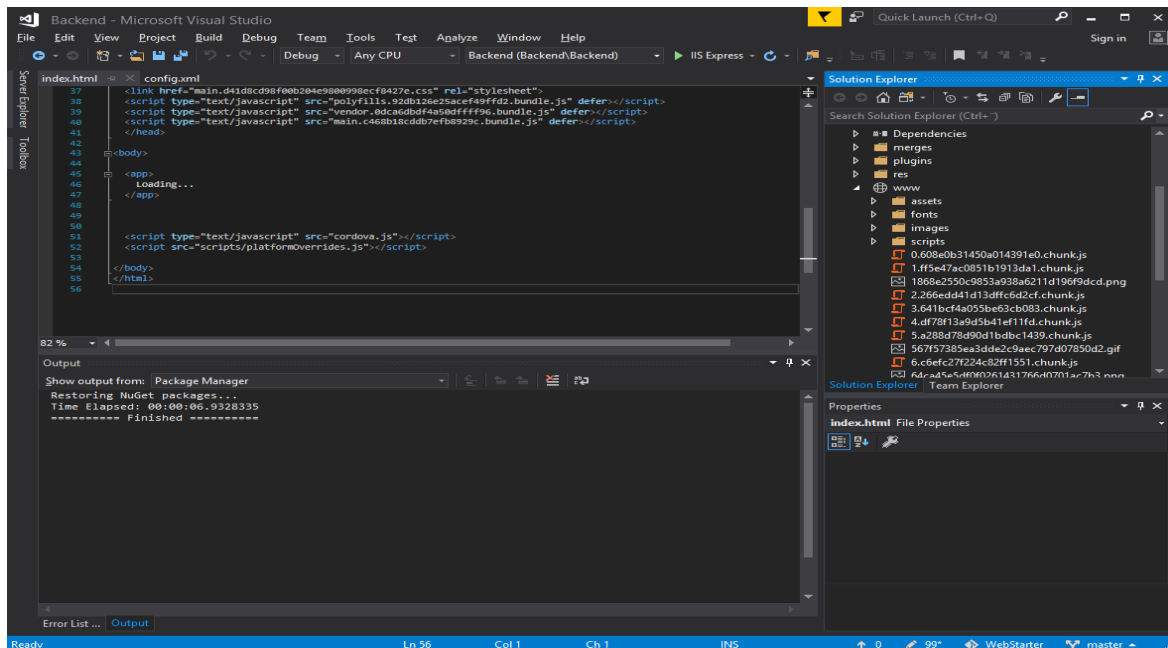
## 2.2.5 Visual Studio

Visual Studio är en IDE, ett verktyg som kan användas för utveckling av applikationer. Microsoft har utvecklat Visual Studio vilket innebär att det främst används för att utveckla applikationer för PC och Windows. Men det är även fullt möjligt att utveckla webb-applikationer och mobilapplikationer. Figur 15 visar på miljön för ett Cordova-projekt. Exempel på programspråk som Visual Studio stödjer:

- C#
- C++
- Visual Basic
- F#
- HTML, CSS & JavaScript
- TypeScript

Visual Studio innehåller en rad verktyg för att kunna skapa skrivbordsapplikationer, webb-applikationer samt mobila applikationer för plattformarna iOS, Android och Windows Phone. Visual Studio innehåller även funktioner för att enkelt kunna synkronisera applikationer med Microsoft Azure, vilket är en samling av integrerade molntjänster som kan anslutas och synkroniseras med olika typer av applikationer. [19]

Exempelvis kan mobila applikationer kopplas till Azure för att utöka applikationerna med olika molntjänster som t.ex. lagring av applikationsdata, autentisera användare, skicka pushnotiser samt lägga till passande backend-funktionalitet i C# eller Node.js. Visual Studio har använts till detta projekt genom ett tillägg som heter "Tools för Apache Cordova". Detta tillägget förklaras mer ingående i avsnitt 2.2.6. [14], [15]



Figur 16 på Visual Studiomiljön för en Cordovaapplikation. [Skärmdump]

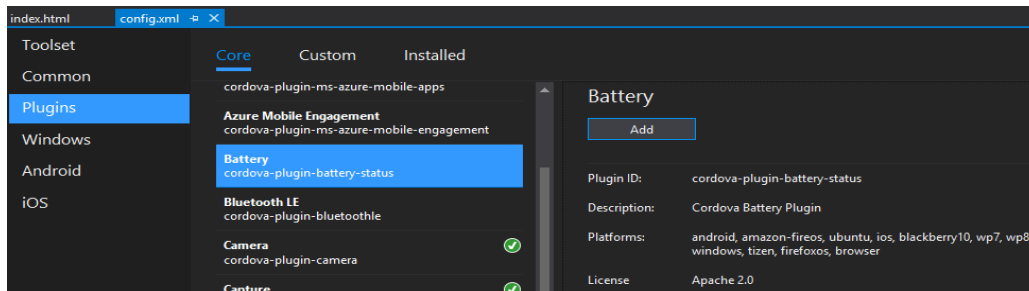
## 2.2.6 Tools for Apache Cordova

Tools for Apache Cordova är ett tillägg som finns tillgängligt i Visual Studio-miljön att ladda ner, som förenklar utveckling av applikationer i Cordova. För att installera tillägget ska utvecklare navigera till Visual Studios hemsida för Tools for Apache Cordova. Ramverket och komponenter som behövs installeras direkt med ett knapptryck, samtidigt som Visual Studio underrättar användaren om vilka av de nedladdade komponenterna som behöver uppdateras. Förutom Apache Cordova visar figur 15 vilka komponenter som installeras.

Komponent	Varför det behövs/installeras
Java JDK (Development kit)	För att kunna använda Android som plattform för Cordova-applikationen.
Android SDK	För att kunna köra Cordova-applikationen i Android-enheter.
Joyent Node.js	För att integrera Apache Cordova med kommandotolken. Olika kommandon vid behov.
Git CLI	För att manuellt kunna lägga till plugins från tredje parter från Github. Detta kommer dock ej behövas då det går att göra direkt i Visual Studio.

Figur 17 visar på vad tillägget installerar förutom Apache Cordova för att kunna utveckla applikationer. [20]

Joyent Node.js ger möjligheten att använda kommandotolken för att utveckla och testa Cordova-applikationer. Alla kommandon som är tillgängliga finns på Cordovas hemsida. Dock kommer kommandotolken endast användas i samband med Ionic i avsnitt 2.4. Anledningen till detta är att "Tools for Apache Cordova" har ett användargränssnitt som förenklar installationsprocessen av plugins. Ett exempel visas i figur 16. Efter att ha navigerat till config.xml och klickat på plugins i vänstermenyn, ska det väljas en plugin i listan och sen klickas på Add. För att hitta config.xml finns filen till höger under Solution Explorer i figur 13. [Skärmdump], [18]



Figur 16 visar hur ett plugin installeras. [Skärmdump]

## 2.2.7 Ionic2

Ionic är ett SDK med öppen källkod för att utveckla mobila applikationer för exempelvis iOS och Android. Det är byggt på Angular och Cordova, vilket innebär att all funktionalitet från Cordova också finns tillgänglig även i Ionic. Pluginfunktionalitet implementeras genom TypeScript. En Ionic-applikation utvecklas i princip på samma sätt som en Angular-applikation som kort beskrevs i avsnitt 2.1.4. [21]

### Plugins i Ionic

Det finns inget tillägg för Ionic i Visual Studio men Joyent Node.js som installerades i samband med installationen används för att installera plugins genom kommandotolken. Till skillnad från Tools for Apache Cordova som har ett användargränssnitt för installation av plugins. För att kunna få tillgång till native plugins i Ionic ska ett paket "Ionic Native" installeras. Detta är ett paket som innehåller native plugins som ska importeras till en TypeScript-fil innan det kan användas. Efter att ha startat ett kommandotolk-fönster i applikationens mapp, visar figur 17 hur paketet installeras.

```
npm install @ionic-native/core --save
```

Figur 17 visar hur native-paketet installeras i Ionic i en kommandotolk.

Detta kommando installerar core plugins som motsvarar tidigare nämnda core plugins i Cordova. Om nya plugins behöver installeras, måste dessa installeras genom ett Ionic-kommando och sedan läggas till i Ionic Native paketet. Detta utförs i kommandotolken genom två steg enligt figur 18.

1. `ionic plugin add "exempel-Plugin1"`
2. `npm install @ionic-native/"exempel-Plugin1" --save`

Figur 18 visar hur ett plugin installeras genom två steg i Ionic.

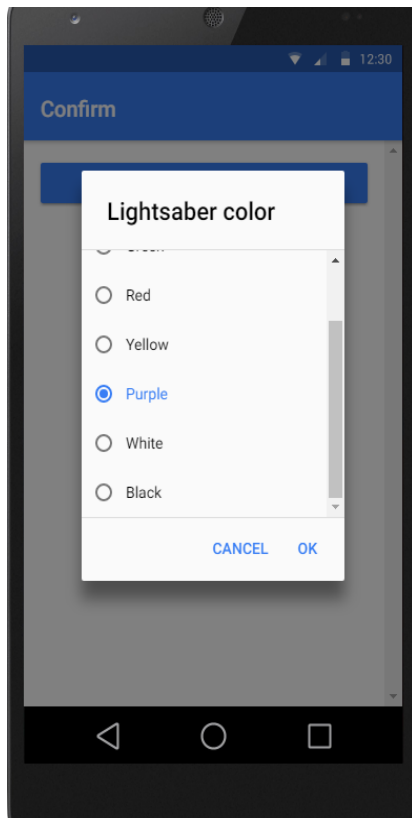
För att få tillgång till plugins som har installerats gäller det att importera det som önskas till sin TypeScript-fil där logiken befinner sig. Detta utförs på följande sätt i klassen enligt exemplet i figur 19. [22]

```
import { exempelPlugin1 } from '@ionic-native/exempelPlugin1';
```

Figur 19 visar hur ett plugin importeras för användning i en TypeScript klass.

## Ionic Components

Eftersom Ionic utvecklas på samma sätt som en Angular webbsite, har detta ramverk specifik funktionalitet vad gäller front-end och design som är inte finns i Cordova. Cordova kan köra webbapplikationer och ge tillgång till plugins men inkluderar inga andra hjälpmedel. Ionic har ett färdigt paket med de så kallade byggstenarna (komponenter) som utvecklare kan välja ifrån. Ionics syfte med dessa komponenter är att enklare kunna skapa ett användargränssnitt för den mobila plattformen. Det kan vara komponenter för knappar, checkboxar eller färdiga menyer som kan nyttjas direkt. Figur 20 visar på ett exempel för så kallade "Alerts" hämtade direkt från Ionic där färg för en lightsaber (lasersvärd) ska väljas efter ett knapptryck. Detta är endast ett exempel av väldigt många som kan väljas. Utvecklaren kan efter att ha importerat ändra koden för att anpassa komponenten till egen applikation. Komponenter importeras även med en template HTML-fil som ej visas i detta exempel. [23]



```
import { AlertController } from 'ionic-
angular';
export class MyPage {
  constructor(public alertCtrl:
AlertController) {
  }

  showRadio() {
    let alert = this.alertCtrl.create();
    alert.setTitle('Lightsaber color');
    alert.addInput({
      type: 'radio',
      label: 'Blue',
      value: 'purple',
      checked: true
    });

    alert.addButton('Cancel');
    alert.addButton({
      text: 'OK',
      handler: data => {
        this.testRadioOpen = false;
        this.testRadioResult = data;
      }
    });
    alert.present();
  }
}
```

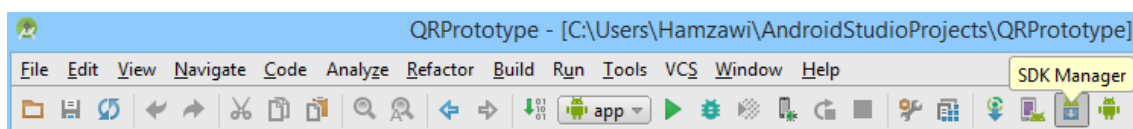
Figur 20 visar på hur en färdig komponent från Ionic kan importeras. [23]

## 2.3 Native Prototyp

Tredje alternativet som undersökts i detta arbete är utveckling av en native prototyp. Denna prototyp ska ha funktionalitet som möjliggör skanning av koder med hjälp av kamera. Kommande kapitel beskriver verktygen som använts för att utveckla denna prototyp.

### 2.3.1 Android Studio

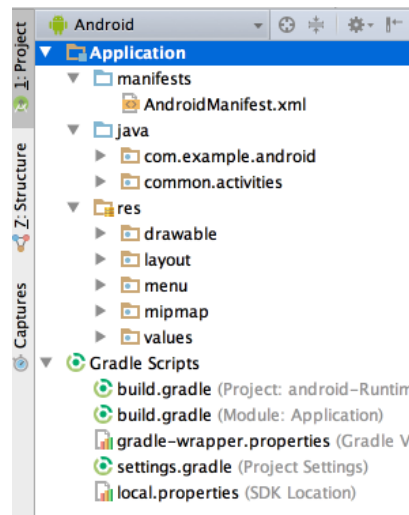
Android Studio är en IDE som är den officiella för Androidutveckling. Denna utvecklingsmiljö stödjer exakt alla enheter som drivs av operativsystemet Android. Prototyper eller applikationer skapas inte bara för mobila enheter i Android Studio, men även för till exempel klockor och TV-apparater. Tillsammans med Android Studio installeras Android SDK Manager. Se i figur 21 för tillgång till SDK manager. [24], [25]



Figur 21 visar var SDK Manager kan hittas i Android Studio längst till höger.

SDK Manager håller reda på vilka SDK som är installerade eftersom olika versioner av Android behöver sin egen version av SDK. Det är viktigt i samband med utveckling av prototyper eller applikationer för äldre Android operativsystem. Då behövs det äldre SDK paket som kan installeras genom Android SDK manager.

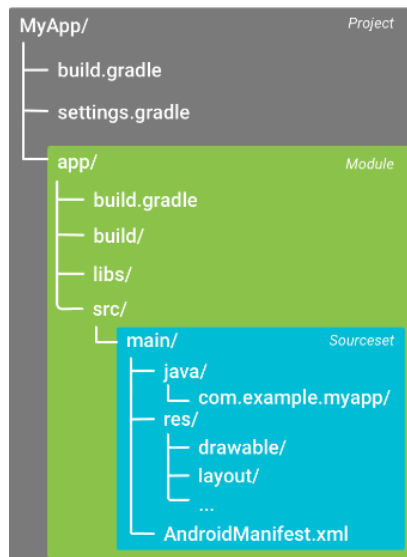




Figur 22 visar på strukturen för ett påbörjat Android Studio projekt. [Skärmdump]

När det skapas ett nytt Android Studio projekt skapas även en struktur (se figur 22) och ett antal filer som visas i figur 14. Av dessa filer är det tre stycken som står för utvecklingen och Gradle Scripts som står för extra kompilerings-inställningar. Dessa filer är följande:

- **Res:** En förkortning för Resources. Inget relaterat till programkod sparas i denna mapp. Alla resurser som behövs för applikationen sparas i denna fil i form av bilder, ikoner och skisser för det tänkta användargränssnittet.
- **Java:** I denna mapp skapas det Java-filer som innehåller programkoden. Denna mapp ska innehålla all funktionalitet och logik för ett Android Studio projekt.
- **AndroidManifest.xml:** Beskriver vad applikationen består av. Vilka Androidversioner som ska stödjäs och vilka behörigheter applikationen har (exempelvis internet, kamera eller annan hårdvara).
- **Gradle Scripts:** Används för att konfigurera bygget och kompileringen av applikationen. Figur 23 visar de olika nivåerna i ett projekt.
  - **Build.gradle (Project):** Konfiguration av inställningar som gäller för hela projektet i Android Studio, d.v.s. inställningar på projektnivå. Det kan t.ex. vara version av Android SDK som projektet ska använda för kompilering. Figur 23 visar på projektnivå och applikationsnivå.
  - **Build.gradle (Module):** Konfiguration av inställningar för applikationsnivå. Här deklaras beroenden för att kompileras som exempelvis utomstående API:er för att kunna nyttja dess funktioner. [26]



Figur 23 visar på de två olika nivåerna för en build.gradle fil. En i projektnivå respektive module(applikationsnivå). [Skärmdump], [26]

### 2.3.2 Android Aktiviteter

En Android applikation består vanligtvis av flera aktiviteter som en användare navigerar till och från. Dessa aktiviteter går igenom olika tillstånd beroende på vart användaren navigerar. Figur 24 är en aktivitet som visar ett meddelande i nedre delen av skärmen vid tryck på knappen "CLICK ME".

```

(activity_main.xml)

<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Click me"
    android:id="@+id/buttonClickMe"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
/>
</RelativeLayout>

(MainActivity.java)

Button clickMe = (Button) findViewById(R.id.buttonClickMe);
clickMe.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        Toast toast = Toast.makeText(
            getApplicationContext(), "Button was clicked!",
            Toast.LENGTH_LONG );
        toast.show();
    }
}

```

Figur 24 visar på en aktivitet med en knapp "CLICK ME", som visar ett meddelande (Toast) vid knapptryck. [Skärmdump]

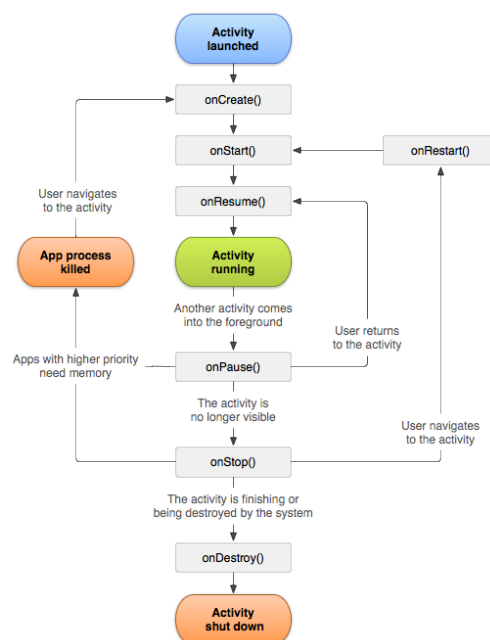
Aktiviteter i Android består av en layout och en Java-fil. Java-filen innehåller all logik och programkod för den specifika aktiviteten. En layout är ett XML-dokument som beskriver de synliga resurserna för användaren, till exempel knappar, listor och menyer. XML-dokument påminner om HTML-dokument när det gäller design då det används taggar som ska resultera i ett användargränssnitt. Ett XML-dokument måste alltid ha ett element som ska vara av typ *View* innan andra taggar deklarerar. Detta element ska vara förälder till alla taggar som adderas i efterhand. Figur 24 är ett exempel på en aktivitet med en layout XML-fil med en Java fil. I detta exempel är *<RelativeLayout>* förälder till taggen för knappen. Notera att knappen har ett identifikations attribut *id* från figur 24:

```
android:id="@+id/buttonClickMe"
```

Alla taggar måste ha id attribut eftersom attributet används för att hitta den specifika taggen i Java-filen. I MainActivity.java ser vi att ett Button attribut skapas och tilldelas genom metoden *findViewById()*. Metoden behöver in-parameter id för att hitta taggen i XML dokumentet:

```
Button clickMe = (Button) findViewById(R.id.buttonClickMe);
```

På detta vis har knappen hittats och tilldelats ett attribut som möjliggör programmering av denna knapp. I figur 24 har knappen *clickMe* en lyssnare som väntar på att en användare ska trycka på knappen. [27], [28], [29]



Figur 25 visar de olika tillstånden för en aktivitet. [30]

En aktivitet består huvudsakligen av sex stycken olika tillstånd som anropas av operativsystemet som visas i figur 25. Varje tillstånd har ett eget syfte och exekveras därefter. Dessa tillstånd implementeras i en Java-fil som tillhör en Android aktivitet.

De sex tillstånden är Javametoder som består av:

- **onCreate():** Anropas när aktiviteten startas. Användargränssnittet för aktiviteten skapas och visas i enheten. Denna metod måste implementeras av utvecklaren och ska vanligtvis innehålla enkel logik som skapandet av listor för relevanta objekt, knappar eller annat relaterat till användargränssnitt. Java-kod för knapptryck från exempelvis figur 21 ska implementeras eller exekveras inuti denna metod.
- **onStart():** Anropas av operativsystemet när aktiviteten exekveras efter att ha blivit stoppad, exempelvis när en användare lämnar en aktivitet för en annan.
- **onResume():** Anropas av operativsystemet efter aktiviteten har anropat onStart(). Vid detta anrop så kan användare interagera med aktiviteten. Detta tillstånd bibehålls tills en annan händelse avbryter. Exempelvis ett telefonsamtal.
- **onPause():** Anropas av operativsystemet när en aktivitet hamnar i bakgrunden av en applikation men ännu inte förstörts.
- **onStop():** När aktiviteten inte är synlig för en användare så har operativsystemet anropat onStop(). Exempel på användning är sparandet av data till en databas, då detta är innan en aktivitet förstörs.
- **onDestroy():** Anropas när en aktivitet ska förstöras. [30]

Metoden *onCreate()* måste implementeras av utvecklare för att en aktivitet ska exekveras. Resterande metoder behövs inte i aktivitetens Java-fil, men rekommenderas för en effektiv applikation.

### 2.3.3 Google API - Play Services för Android

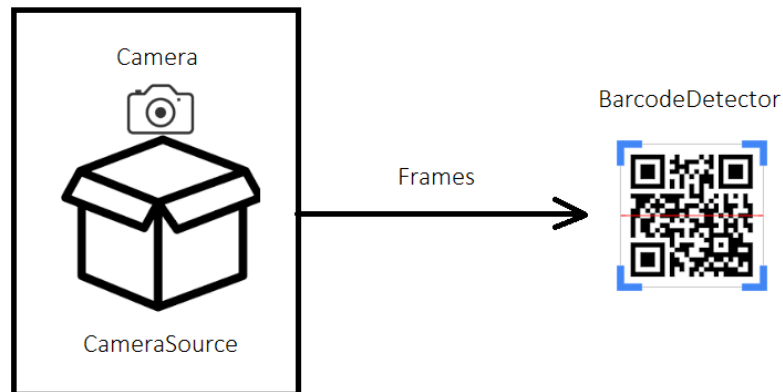
Google Play Services är ett API som användes vid utveckling av prototypen. Detta API gör det möjligt för en applikation att nyttja en mängd olika funktioner. Exempel på sådana funktioner är Google Maps, push notiser och framförallt i detta fall skanning av koder. För att kunna använda Googles API bör paketet "Google Play Services" och "Google Repository" installeras för applikationen eller prototypen. Dessa hittas genom att gå in i Android SDK Manager och sedan klicka på SDK Tools. Efter installation adderas paketet under dependencies (beroenden) i build.gradle filen som figur 23 (applikationsnivå). [31]

```
apply plugin: 'com.android.application'
dependencies {
    compile 'com.google.android.gms:play-services:10.2.4'
```

Figur 26 visar hur Google Play Services paketet läggs till i build.gradle för att kompileras. [31]

### 2.3.4 Google API Visions Package – Visuell identifiering av objekt

Ett paket som använts för att utveckla prototypen är "Visions". Paketet innehåller funktionalitet för visuell identifiering av objekt. För att möjliggöra identifiering av objekt i en applikation används två subpaket inom paketet "Visions". Dessa subpaket illustreras i figur 27 vilket är *CameraSource* och *BarcodeDetector* som användes för prototypen. [34]



Figur 27 visar *CameraSource* och *BarcodeDetector* som användes i prototypen.

*CameraSource* är ett paket som hanterar kameran. *CameraSource* tar emot ramar från kameran när användare riktar den mot ett objekt för identifiering. Dessa ramar skickas sedan till en detektor som hanterar text, ansikten eller kod. Utvecklare rekommenderas att bestämma hur snabbt eller hur många ramar i sekunden som ska skickas till en detektor. Ramar skickas väldigt snabbt till en detektor utan någon förinställning och det riskerar att en detektor ignorerar vissa ramar ifall en hastighet inte är specificerad för *CameraSource*. [32]

*BarcodeDetector* identifierar skanningskoder från ramar som kommer från *CameraSource*. Detektorn jämför objektet i ramarna med skanningskoder som stöds i paketet. Det är inte endast QR-koder som stöds därför rekommenderas det att specificera vilken typ av kod som ramarna ska jämföras med för bättre prestanda. Figur 28 visar ett exempel på hur en instans av *BarcodeDetector* skapas och hur den begränsas till att endast stödja QR-kod format. [33]

```
BarcodeDetector barcodeDetector = new  
BarcodeDetector.Builder(this)  
    .setBarcodeFormats(Barcode.QR_CODE)  
    .build();
```

Figur 28 visar på hur en instans av *BarcodeDetector* skapas. [Skärmdump]



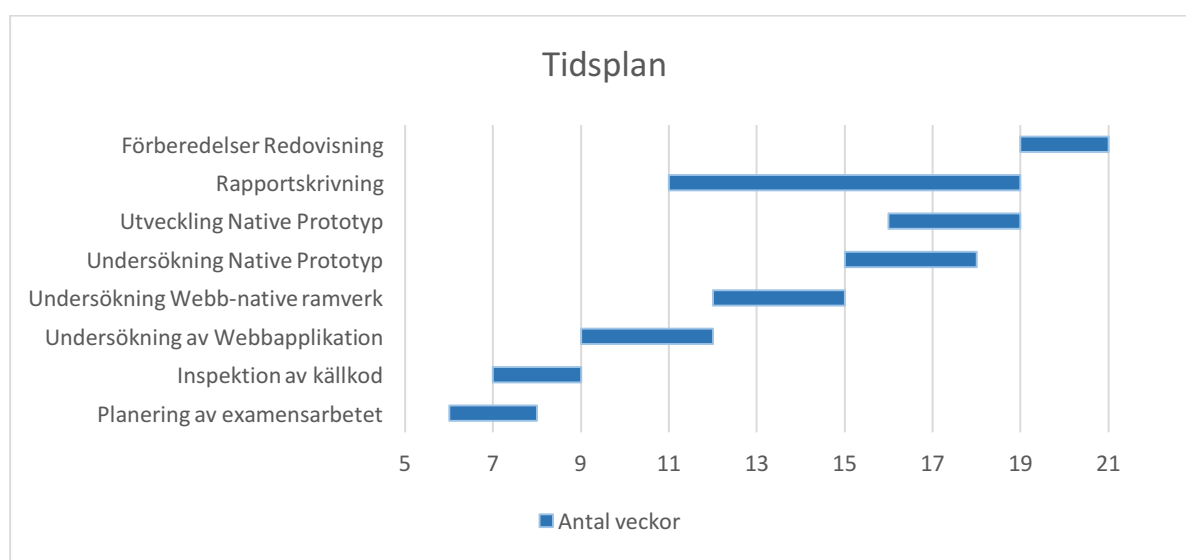
## 3 Metod och Analys

Detta avsnitt beskriver metoder som använts under detta projekt. Hur tidsplanen såg ut, vilka metoder som tillämpades och vilka faser som projektet bestod av. I detta avsnitt ingår det även en analysering av arbetsprocessen och utvecklingen.

### 3.1 Tidsplan

Enligt tidsplanen i figur 29 kan examensarbetet delas upp i olika faser, från planering av arbete till förberedelse inför redovisning. Arbetsbelastningen var dock störst i de följande tre faserna:

- (i). Undersökning av Webbapplikation
- (ii). Undersökning av Webb-native ramverk
- (iii). Undersökning & utveckling Native Prototyp



Figur 29 visar på tidsplanen som var utgångspunkten för arbetet från och med vecka 6 2017.

Fas (i) och (ii) innehöll mycket som var nytt och där tidigare erfarenheter inom programmering inte var till så stor hjälp. I fas (iii) kunde erfarenheter av Javaprogrammering från kurser i utbildningen användas. Den största arbetsbelastningen hamnade därmed på de två första faserna. I dessa faser var det viktigt att samla in och bearbeta information. Detta genomfördes genom att prova informationen praktiskt, genom att skriva kortare testprogram vars resultat under exekvering kunde jämföras med dokumentationen som användes.

Fas (i) av arbetet var att undersöka teknikerna för en webbapplikation och förstå hur HTML, CSS och JavaScript hänger ihop inom webbutveckling. Detta gjordes genom att undersöka information publicerad av utvecklare av populära webbläsare. Exempelvis så har Mozilla Firefox dokumentation som visade hur HTML- eller CSS-dokument kunde skrivas. Det fanns även en komplett introduktion och dokumentation för JavaScript som inte var anpassad för en specifik webbläsare. Många handledningar (tutorials) kunde användas för att skriva kod och komma igång. Avsikten med denna fas var att få en förståelse varför en webbapplikation

inte kan använda t.ex. accelerometer i en enhet, men även att förstå programkoden som Cenitos webbapplikation består av. All kunskap som inhämtades under fas (i) användes i nästa fas eftersom fas (ii) kräver grundkunskaper i webbutveckling.

Fas (ii) av arbetet bestod av att inhämta information för att förstå hur Cordova kan paketera webbapplikationer till en webb-native applikation, samt hur native plugins från avsnitt 1.2 kan implementeras. Källan var givetvis Cordovas egen dokumentation. Där tillhandhålls kod-exempel på hur olika plugins kunde implementeras för JavaScript eller TypeScript samt de olika metoderna tillgängliga för varje plugin. Här var all information från fas (i) väldigt viktig p.g.a. att programkoden var skriven i JavaScript. Fas (i) bidrog även till att öka förståelsen för hur Cenitos webbapplikation är uppbyggd och hur den kunde konverteras till en webb-native applikation. Detta var av stor betydelse därför att dokumentation som användes från Cordova kunde endast beskriva hur man arbetar med en ny applikation. Det fanns ingen beskrivning för hur en existerande webbapplikation kan konverteras. Den tidigare fasen förenklade även processen vid skrivandet av programkod för att få tillgång till plugins och utföra test.

Under fas (iii) så påbörjades ett arbete med en Bluetooth-prototyp. Men företaget föreslog ändring på detta för att istället göra en prototyp som skannar koder. Denna prototyp skulle utvecklas för Android med funktionalitet där kameran används för att skanna och identifiera koder. På grund av att denna fas av arbetet skulle göras på en ny plattform, med nya verktyg och annan typ av programspråk så kunde inget utnyttjas från de tidigare faserna. Då Android själva är de som skapat Android Studio så blev det naturligt att direkt använda deras dokumentation för utveckling av prototypen. Som vid tidigare faser handlade det i inledningen om att hämta och bearbeta information för att applicera denna praktiskt i form av exekverbar programkod.

## 3.2 Arbetsmetod – Inspiration från Scrum

Scrum är en metod som används för en agil systemutvecklingsprocess. En agil utvecklingsprocess består av flera principer och två av dessa som Scrum uppfyller är följande:

1. Högsta prioritet är att se till att kunden är nöjd genom tidig och kontinuerlig leverans av mjukvara.
2. Ändring av krav är alltid välkomna, även sent i utvecklingsprocessen.

De två principerna uppfylls med hjälp av de byggstenar som Scrum består av. En av dessa är PBI som står för Product Backlog Items. PBI kan jämföras med en kravlista där kunder vid behov kan gå in och ändra på prioriteringen av krav, men även lägga till eller ta bort krav. Detta säkerställer att utvecklare prioriterar de funktioner som har mest värde för kunderna. Implementeringen sker sedan under mindre faser som kallas för Sprints. Dessa faser kan vara mellan 1 och 4 veckor och under en Sprint har utvecklare ett kort möte dagligen som kallas för Daily Scrum. Under detta möte diskuterar utvecklarna vad som har implementerats sedan igår, vad som ska göras till imorgon och vilka eventuella hinder som finns. Daily Scrums är viktigt eftersom att ett stort projekt kan bestå av många mindre grupper och dessa grupper



behöver synkronisera med varandra för att kunna uppdatera sig angående varandras utvecklingsprocesser. [33], [34], [35]

Detta examensarbete utfördes endast av en person vilket gör det svårt att ta hjälp av olika projektmetodiker. Istället inspirerades arbetet av några enstaka principer från en metod som Scrum, som består av PBI och Daily Scrums. Dessa två principerna inspirerade dock hur arbetet skulle planeras. För varje fas (i) examensarbetet sammanställdes en lista på vad som ska göras och i vilken ordning. Detta påminner om en PBI. Denna lista sammanställdes ensam utan kommunikation med företaget. Detta eftersom företaget tillät examensarbetaren ha full kontroll över utvecklingsprocessen och själv bestämma prioriteter. Varje dag genomfördes en utvärdering som liknade en Daily Scrum. Under denna utvärderades vad som uppnåtts, vad som ska göras till nästa dag, om några hinder finns och om dessa hinder kräver en ändring i PBI.

## 3.3 Källkritik

### 3.3.1 Primära källor

- *Typescriptlang.org* är en primärkälla för information angående TypeScript med avseende för att informera och presentera fakta i form av programkod. Källan är fristående och underhålls av Microsoft vilket gör den pålitlig då Microsoft själva skapade och designade TypeScript.
- *Angular.io* är den källa som använts för att inhämta information angående ramverket Angular. Källan är fristående och en primärkälla vilket gör den trovärdig då det är skaparna av Angular som använder denna källa för att informera och lära ut utvecklare på nätet hur Angular används. Skaparna av Angular består av utvecklare och experter från Google.
- Ramverket Cordova är ett ramverk med öppen källkod som förses av *apache.cordova.org*. Det är en organisation som skapade ramverket vilket gör den trovärdig. Den är fristående och används för att informera och presentera ramverket för utvecklare som vill skapa applikationer med hjälp av webbt teknologi.
- *Ionicframework.com* är källan som använts för inhämtning av information för Ionic2. Källan är trovärdig på grund av att det är en primär och fristående då det är den enda som primära källan. Källan används för att informera och presentera utvecklare för Ionic.
- Två olika källor har använts för Visual Studio och verktyget Tools for Apache Cordova. Båda källorna används för att informera och presentera fakta i syfte att lära utvecklare använda programmet eller verktyget.
- *Docs.microsoft.com* har använts för att förstå hur Tools for Apache Cordova ska användas, och hur utveckling med detta tillägg går till. Källan är pålitlig då Microsoft är skaparna av tillägget och är oberoende av andra källor.
  - *Taco.visualstudio.com* är en källa inom Microsoft- och Visual Studio-nätverket. Källan beskriver hur plugins installeras och vilka möjligheter som finns för Tools for Apache Cordova. Källan är trovärdig eftersom den underhålls av utvecklare från Microsoft som har skapat tillägget.
- För utveckling av Android Prototypen användes två stycken källor. Båda används i syfte att informera och presentera programkod för utvecklare. *Developer.android.com* är ett

nätverk skapat av grundarna av Android för att lära ut utveckling i ett Android operativsystem. Detta gör källan pålitlig i samband med att det är en primärkälla. Detsamma gäller för *developer.google.com* som är ett nätverk skapat för utvecklare av Google för att få information om API:er och hur dessa ska användas.

### 3.3.2 Sekundära Källor

- *Scrumguides.org* är en organisation och källan har använts i samband med metodiken som projektet inspirerades av. Det är en sekundär källa som är beroende då den refererar till dokument skapade av de personer som grundade Scrum, vilket gör den trovärdig. Syftet är att informera om hur organisationer kan använda sig av Scrum för sina projekt.
- *Agilealliance.org* är en källa som använts som inspiration för projekt metodiken. Det är en källa som uppmuntrar användningen av Agila metodiker. Källan är pålitlig då den refererar till ett manifest som alla agila metodiker utgår efter.
- *Developer.mozilla.org* är en sekundär källa som använts i samband med JavaScript. Det är en organisation med syftet att ha informationen tillgänglig för utvecklare som vill skapa webb relaterade projekt. Källan är trovärdig p.g.a. att de själva driver en webbläsare som använder sig utav de tekniker som de dokumenterar och presenterar för utvecklare som vill lära sig
- *JavaScript: The Good Parts* är en bok som använts till detta projekt. Det är en källa som förklarar och presenterar de olika delarna i JavaScript som funktioner, objekt och metoder. Det är en källa med hög trovärdighet då författaren var en senior JavaScript-arkitekt 2008 då boken släpptes.

## 3.4 Analys – Verktyg och Arbetsprocess

Att tidigare erfarenhet av webbutveckling saknades resulterade i en arbetsam start med examensarbetet. Trots detta gick det under den första fasen av examensarbete att förstå varför företag är intresserade av ett ramverk som Cordova. JavaScript är ett populärt programspråk med många ramverk och API:er för utveckling av grafiska gränssnitt. Exempelvis så är Ionic ett ramverk som förenklar utvecklingen av användargränssnittet tack vare alla färdiga komponenter som finns tillgängliga. Jämfört med Android och Java där utvecklaren måste själv göra det från grunden. Detta gör JavaScript ett mer passande programspråk än Java för grafiska användargränssnitt. I kombination med möjligheten att utveckla native funktionalitet för flera plattformar med JavaScript, gör programspråket väldigt flexibelt då det inte längre är begränsat till webben.

Något som gör JavaScript till ett flexibelt programspråk är exempelvis programkod som ska generera en meddelanderuta. I Android måste en instans av meddelanderuta-klass skapas, bestämma med en metod vad meddelandet ska innehålla och sedan skapa det. Till skillnad från JavaScript som använder sig av `alert("Meddelande text")`.

En aspekt som underskattades vid undersökningen av alternativ två webb-native är installation av programvaror och verktyg. Visual Studio är ett väldigt stort program som kräver mycket tid vid installation. Cordovatillägget i Visual Studio behövde också tid att installeras, eftersom tillägget består av ett flertal komponenter. Även om arbetet gick enligt tidsplan

skulle det varit tidsbesparande att förbereda alla program eller verktyg tidigare för att på så vis kunna börja undersöka och utveckla direkt.

### 3.5 Analys – Utveckling

Utveckling genom tillägget Tools for Apache Cordova i Visual Studio visade sig förenkla användningen av Cordova. Främst genom tidsbesparing där alla komponenter som behövs för Cordova installeras direkt i samband med tillägget. Alternativet vore att söka upp alla komponenter var för sig och installera dessa, vilket inte var optimalt. Med hjälp av både Ionic och Cordova kunde Cenitos befintliga webbapplikation konverteras till webb-native utan några hinder. Det gick även att få tillgång till native-funktionalitet lika framgångsrikt genom båda ramverken. Det fanns skillnader i programkoden men ingenting som påverkade prestandan. Ingenting under exekvering av programkoden visade att det var bättre att använda just Cordova eller Ionic för tillgång av native funktionalitet. Ionic visade sig ha större möjligheter vid utveckling av användargränssnitt, tack vare de färdiga komponenterna som finns att tillgå. Det innebär också att i ett scenario då det inte finns en existerande webbapplikation så är Ionic möjligen ett bättre val. Ionics möjligheter med de färdiga komponenterna testades aldrig då examensarbetet avgränsades till att gälla native plugins.

Utveckling av native prototypen gick som förutspått lite snabbare än webb-native alternativet. Tidigare erfarenhet av Java var till stor fördel till skillnad från de två tidigare alternativen med webbutveckling. När prototypen skulle utvecklas i Android Studio behövdes det optimeringar som inte var nödvändiga i en webb-native applikation. Exempel på detta var deklareringsberoenden och att prototypen har behörighet att använda kameran i enheten. Dessa optimeringar behövdes inte i Cordova eller Ionic då allt var färdigt i plugins som installerades. Skulle en fullständig applikation utvecklas så kan det vara ett hinder att behöva göra optimeringar för varje operativsystem. Troligtvis innehåller en fullständig applikation många beroenden och flera behörigheter som behöver optimeras. Något som talar för användning av en native prototyp är att det är möjligt att få tillgång till all funktionalitet som finns i en enhet. Ifall prototypen som utvecklades behöver tillgång till Androids email klient är detta möjligt med några rader programkod. Detta är inte möjligt i en webb-native applikation, endast om en email plugin från ramverket finns att referera till.



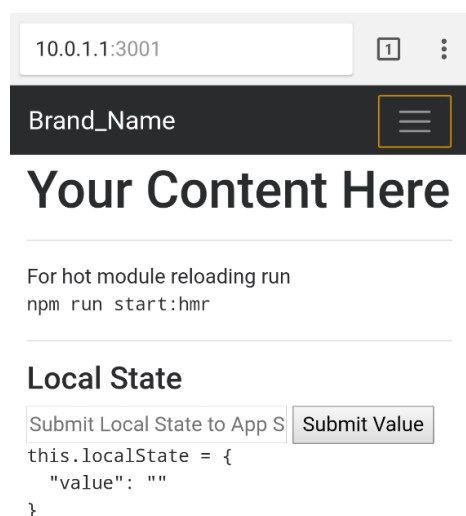
## 4 Resultat

---

Detta kapitel behandlar resultaten för varje fas som diskuterades i kapitel 3 metod. Eventuella resultat av kortare programkod kommer presenteras tillsammans med skärmdumpar för de tre olika alternativen för applikationsutveckling.

### 4.1 Webbapplikation

Första fasen resulterade inte kod, utan att sätta sig in i hur Cenitos site är uppbyggd, hur de olika webbt teknikerna används och hur Cenitos applikation ser ut när den exekveras genom en mobilbrowser. Se figur 30 som visar hur applikationen såg ut i enhetens browser.



Figur 30 visar på Cenito webbapplikation genom Android enhetens browser. [Skärmdump]

### 4.2 Webb-native – Cordova & Ionic

Konverteringen från webbapplikation till webb-native resulterade i ett användargränssnitt som liknade figur 30 med undantaget att applikationen exekverades som en webb-native applikation. Det innebär att det inte existerade ett adressfält som i figuren vilket resulterar i utseende som påminner om en applikation och inte en webbsida. Eftersom Cenitos applikation är utvecklad i Angular placerades all programkod för användning av plugins i root-komponenten för applikationen. Detta ger tillgång till plugins oavsett vilket par av komponent och template-filerna som visas för användaren.

Denna konvertering genomfördes efter att ha kompilerat Cenitos applikation då den är utvecklad med TypeScript. Kompileringen genererar JS-filer som kan exekveras i en webbläsare. Ett Cordova-projekt skapades i Visual Studio och i projektets *www-katalog* placerades de kompilerade JS-filerna. Konverteringen i Ionic genomfördes på samma sätt då ett sådant projekt också innehåller en *www-katalog* eftersom det har Cordova som grund. Detta resulterade i att Cenitos webbapplikation klassades som webb-native och därmed tillgång till native funktionalitet.

I webb-native implementerades plugins för accelerometer, QR-kod skanning, kamera, hämtning av bilder eller filer samt push notiser. Appendix A avsnitt 8.4 visar på programkod som gör det möjligt att använda sig av push notiser med Cordovas plugin. Denna programkod ska kompletteras med en backend från företaget som möjliggör utskick av push notiser för att säkerställa att det fungerar för mer än en enhet.

Figur 31 visar ett exempel på hur Accelerometer plugins användes tillsammans med skanning av QR-kod tillsammans. Accelerometern beräknar accelerationen på tre olika håll X, Y och Z frekvent på ett intervall på 500 millisekunder. Ifall absolutbeloppet av dessa tre blir större än 25 startades en QR-kod skanner. Figur 32 visar hur enheten ser ut när *this.Scan()* exekveras och Appendix A i kapitel 8 visar hur metoden *this.Scan()* var implementerad med resterande plugins.

```
document.addEventListener("deviceready", () => {
  console.log("Ready to access plugins");

  let watchID: WatchHandle = navigator.accelerometer.watchAcceleration(
    (acceleration: Acceleration) => {
      x1 = acceleration.x;
      y1 = acceleration.y;
      z1 = acceleration.z;
    },
    () => { alert("Error!"); }, {frequency: 500}
  );
  setInterval(() => {
    let shake = Math.abs(x1 - x2 + y1 - y2 + z1 - z2);

    if ( shake > 25) {
      console.log("Device Motion detected");
      this.Scan();
    }
    x2 = x1;
    y2 = y1;
    z2 = z1;
  }, 500);
}, false );
```

Figur 31 visar hur kod för Accelerometer ser ut för att starta metoden *Scan()* efter en skakning.

Applikationen hade samma utseende oberoende om det var Cordova eller Ionic som användes. Det var dessutom ingen skillnad i hur programkoden skulle skrivas förutom ett mindre undantag. Undantaget gäller hur utvecklare får åtkomst till pluginmetoderna. I figur 31 ser vi att attribut *watchID* tilldelas med hjälp av *navigator.accelerometer.watchAcceleration()*. Där *navigator*attributet är det som ger tillgång till accelerometers plugin-metoder och attribut. I Ionic uppnås detta med imports som deklarerats i programkoden. Figur 32 visar hur Accelerometer importerats och sedan använts för tilldelning. Figuren har ytterligare skillnader eftersom Ionic och Cordova har olika gränssnitt för hur utvecklare ska använda plugins, men efter tilldelning av attribut är programkoden densamma. Ett exempel är *subscribe(acceleration)* metoden i figur 32. Metoden talar om när ramverket ska lyssna på accelerometern genom variabeln *acceleration*. För att sluta lyssna på accelerometern kan metoden *unsubscribe()* anropas. Motsvarigheten i Cordova för *navigator* i figur 31 är *navigator.accelerometer.clearWatch(watchID)* för att sluta lyssna.

```
//Länggt upp i klassen: import {DeviceMotion} from 'ionic-native';
//Sedan:
let watchID =
DeviceMotion.watchAcceleration({frequency:200}).subscribe(acceleration);
```

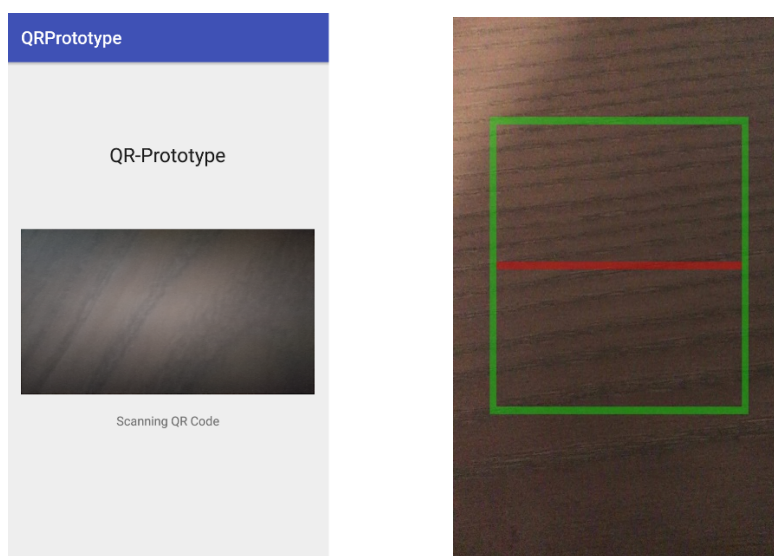
Figur 32 visar hur programkoden skrivs för Accelerometer i Ionic för att nyttja dess funktioner.

### 4.3 Native Prototyp

Fas (iii) av arbetet resulterade i en native prototyp som kan skanna QR-koder. I webb-native applikationen så tar skannings-funktionaliteten hela skärmen för enheten. Detta är inte fallet för native prototypen utan utvecklaren kan själv bestämma. En klass *SurfaceView* i Android Studio är en vy som kan användas i en aktivitet för att bestämma innehållet i vyn. I denna vyn bestämdes storleken i XML filen, sedan justerades *CameraSource* storlek till att matcha vyn i programkoden som figur 33. Resultatet av prototypens utseende visas i figur 34 i jämförelse med webb-native alternativet för QR skanning.

```
barcodeDetector = new BarcodeDetector.Builder(this)
    .setBarcodeFormats(Barcode.QR_CODE)
    .build();
cameraPreview = (SurfaceView) findViewById(R.id.scanningPreview);
cameraSource = new CameraSource
    .Builder(this, barcodeDetector)
    .setRequestedPreviewSize(640, 480)
    .build();
```

Figur 33 visar hur kamerahanteraren anpassar kamera-fönstrets storlek till storleken på *SurfaceView* variabeln *cameraPreview*.



Figur 34 visar hur enheten såg ut vid QR-skanning för native respektive webb-native alternativet.





## 5 Slutsats

---

Detta projekt har varit väldigt arbetsamt med tanke behovet av att lära sig ett nytt programspråk. Mycket tid gick åt att läsa in teori och information och försöka förstå hur det ska användas praktiskt. Under många tillfällen kändes det som att projektet hade hamnat vid ett dödläge på grund av brist av tidigare erfarenhet av JavaScript. Detta förstärktes av att det inte fanns dokumentation eller information på hur en webbapplikation kunde konverteras till webb-native med Cordova. Endast om att det var teoretiskt möjligt. Under projektet var det aldrig aktuellt att vara delaktig i företagets Daily Scrums eller annat relaterat till utveckling. Anledningen till detta var att företaget hade en extrem arbetsbelastning och detta projekt genomfördes sidan om. Trots detta så försågs projektet tillräckligt med handledning för att bryta de dödlägen som uppstod under varje fas. Resultaten som åstadkommits i detta arbete kompletterar syftet genom att det gjorde det möjligt för företaget att välja ett alternativ för sin applikation.

1. *Hur ser marknaden ut idag, finns det företag som valt endast ett av alternativen eller har de en kombination?*

De större företagen som har råd att anställa olika sorters ingenjörer har alltid en webbapplikation och en native-applikation för att nå så många användare som möjligt. Företag som exempelvis Instagram och Facebook ger möjligheten att använda nästan alla funktioner genom webbläsaren vare sig det är en telefon eller en PC, samtidigt som en native-applikation finns för en bättre användarupplevelse. Ett mindre företag som består av 25 anställda där alla inte är utvecklare kanske inte har råd att anställa iOS och Android ingenjörer. Vilket gör webb-native alternativet ett mer ekonomiskt alternativ. I slutändan beror det på vad företaget har råd med, vad de tjänar på att ha något av dessa alternativen och vilken funktionalitet applikationen ska ha. Exempelvis finns Skånetrafikens applikation för alla operativsystem, vilket är ganska naturligt för att maximera intäkterna. Webb-native alternativet har inga kända plugins skapade med avsikt att förenkla utveckling av betal- och banktjänster.

2. *Vilken befintlig funktionalitet kan återanvändas eller måste man börja om från början?*

Under planering av arbetet tillsammans med företaget var deras önskan att behålla så mycket funktionalitet som möjligt, utan att behöva kompromissa för att få tillgång till native funktioner vid konvertering till webb-native. Sett till resultatet av konverteringen så kan all befintlig funktionalitet behållas om webbapplikationen är utvecklad med hjälp av HTML, CSS och JavaScript eller TypeScript. Något som företaget var extra nöjda med är att login-funktionen fungerade i applikationen. Vilket innebär också att de kan koppla sina färdiga backend-program till applikationer som körs genom Cordova.

3. *Vad är det för skillnad mellan alternativ webbapplikation, webb-native och native?*
  - a. *Vad tillför webb-native ramverk som en webbapplikation inte kan?*
  - b. *Vad tillför native alternativet jämfört med webb-native?*

Den största skillnaden som separerar alternativ webbapplikation från resterande är brist på tillgång till native funktionalitet. Det är inte möjligt att skapa en webbapplikation för en mobilenhet som kan skanna koder eller använda accelerometer. Vilket gör att det alternativet väljs bort när det handlar om att utveckla den typen av funktioner. Det som skiljer webb-native och native alternativet från varandra är tillgänglighet av operativsystemet. Eftersom QR-koder kan innehålla varierande typer av information för exempelvis kontaktinformation, email eller sms kan dessa funktioner behövas vid olika scenarion. Efter skanning av en QR-kod som innehåller kontaktinformation skulle det vara logiskt att få upp kontaktlistan i enheten. Detta är inte omöjligt i webb-native men problematiskt då det skulle behövas ett separat plugin för kontaktlistan. I jämförelse med native då kontaktlistan är mer åtkomlig. Kontentan av dessa resonemang är att åtkomst av native-funktionalitet är lika bra när det gäller core plugins. I webb-native alternativet kan resultatet av native funktionerna användas inom applikationen men inte utanför. Medans native applikationer kan utvecklas för att nå annan funktionalitet inom operativsystemet.

#### *4. Vilka webb-native ramverk finns det som är jämförbara med Cordova?*

De enda ramverken som kan jämföras med Cordova är Phonegap och Ionic. På grund av att dessa ramverk just använder Cordova men har funktionalitet som Cordova möjligtvis saknar. Phonegap är en marknadsförd version och det är väldigt få skillnader från Cordova. Ionic däremot innehåller något som Cordova saknar i sin version med öppen källkod. Det är verktyg för utveckling av grafiska användargränssnitt och specifikt de komponenter som är tillgängliga att börja utveckla med. Om arbetet var baserat på att utveckla en webb-native applikation från grunden hade Ionic varit tidsbesparande. Eftersom det kan argumenteras att det är bättre att ha ett färdigt användargränssnitt innan utveckling.

#### *5. Vilken funktionalitet ska prototypen ha?*

##### *a. Hur kommer webb - och webb-native applikationerna åt denna funktionalitet?*

Funktionaliteten för prototypen bestämdes till att vara skanning av koder. Då det finns många olika kod-format att välja mellan valdes QR-koder till formatet som prototypen ska kunna skanna. Åtkomst till skanningsfunktionalitet var inte möjligt för en webbapplikation under detta projekt. Webb-native alternativet får åtkomst till funktionen genom installering av ett plugin i Cordova eller Ionic. Detta plugin innehåller ett gränssnitt med metoder för användning av skanningsfunktionerna. Användning av dessa metoder resulterar i uppstart av skanningsfunktion med hjälp av kameran. Efter en slutförd skanning återgår enheten till webb-native applikationen för att visa resultat av skanningen.

## 5.1 Reflektion över etisk aspekt - Sekretess

En etisk aspekt som tillämpades och diskuterades med företaget är sekretess. Webbapplikationen från Cenito som användes till detta arbete är en byggsten som är en grund för framtida utveckling av applikationer. Även om det inte handlade om personlig information eller identitet, så är webbapplikationen programkod som jag har ansvar över och måste se till att den ej hamnar i fel händer. Att inte publicera den offentligen på exempelvis GitHub eller andra källor där människor kan komma åt programkoden. Då det är i princip arbete utfört av

företaget som ska säljas till kunder i form av applikationer. I Internet är det i dagsläget väldigt enkelt att kopiera och klistra programkod och få tag på fullständiga applikationer eller annan typ av programkod.

## 5.2 Möjligheter till vidareutveckling

Möjligheterna att vidareutveckla detta arbete är väldigt stora därför att den största delen av arbetet kretsade kring tillgången av native-funktionalitet. En vidareutveckling av arbetet skulle kunna vara utveckling av en egen plugin till Cordova. En plugin med native funktionalitet som inte finns listad under deras dokumentation. Ett exempel skulle kunna vara en plugin som innehåller email eller SMS funktioner. I projektet användes det en färdig webbapplikation. En möjlig vidareutveckling på arbetet vore att inte endast göra en native prototyp från grunden utan även en webb-native applikation i Ionic eller Cordova. För att testa ramverkets begränsningar inom utveckling praktiskt.



## 6 Terminologi

---

Site	En webbplats/hemsida.
IDE	Integrated Development Environment, miljö för utveckling.
SDK	Software Development kit. Behövs för att utveckla mjukvara till specifika plattformar.
Back-end	Data access delen av en webbapplikation.
Front end	Användargränssnittets delen av applikationen
API	Application programming interface
Browser	Webbläsare som exempelvis Internet Explorer, Chrome och Mozilla FireFox.
Cross-plattform applikation	En applikation som fungerar bra på flera plattformar med samma källkod, t.ex. iOS och Android
Native funktionalitet	Funktionalitet som endast är åtkomlig för den specifika enheten.
HTML	HyperText Markup Language används för att skriva websites.
CSS	Cascading Style Sheets används för styla HTML dokument.



## 7 Källförteckning

---

### Mozilla Developer Network

[1] *HTML*

[https://developer.mozilla.org/en-US/docs/Learn/Getting\\_started\\_with\\_the\\_web/HTML\\_basics](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics)

[Access: 13e februari 2017]

[2] *CSS*

[https://developer.mozilla.org/en-US/docs/Learn/Getting\\_started\\_with\\_the\\_web/CSS\\_basics](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics)

[Access: 13e februari 2017]

[3] *JavaScript*

[https://developer.mozilla.org/sv-SE/docs/Web/JavaScript/A\\_re-introduction\\_to\\_JavaScript](https://developer.mozilla.org/sv-SE/docs/Web/JavaScript/A_re-introduction_to_JavaScript)

[Access: 14 februari 2016]

### TypeScript

[4] *Classes*

<https://www.typescriptlang.org/docs/handbook/classes.html>

[Access: 15 februari 2017]

[5] *Functions*

<https://www.typescriptlang.org/docs/handbook/functions.html>

[Access: 15 februari 2017]

[6] *Class Declaration*

<https://github.com/Microsoft/TypeScript/blob/master/doc/spec.md#1>

[Access: 15 februari 2017]

### Angular

[7] *Architecture*

<https://angular.io/docs/ts/latest/guide/architecture.html>

[Access: 15 februari 2016]

### Cordova

[8] *Architecture & Overview*

<https://cordova.apache.org/docs/en/latest/guide/overview/index.html>

[Access: 22 mars 2017]

[9] *Plugins*

<https://cordova.apache.org/docs/en/latest/guide/cli/index.html#add-plugins>

[Access: 22 mars 2017]

[10] *Camera*

<https://cordova.apache.org/docs/en/latest/reference/cordova-plugin-camera/index.html>

[Access: 22 mars 2017]

[11] *Device Motion (Accelerometer)*

<https://cordova.apache.org/docs/en/latest/guide/overview/index.html>

[Access: 22 mars 2017]

[12] *Push Notifications*

<https://github.com/phonegap/phonegap-plugin-push>

[Access: 23 mars 2017]

[13] *BarcodeScanner (QR Code Scanner)*

<https://github.com/phonegap/phonegap-plugin-barcodescanner>

[Access: 23 mars 2017]

## **Visual Studio & Tools for Apache Cordova**

[14] *Official site* <https://www.visualstudio.com/vs/ide/>

[Access: 22 mars 2017]

[15] *Cross-Platform Mobile Development*

<https://docs.microsoft.com/sv-se/visualstudio/cross-platform/cross-platform-mobile-development-in-visual-studio#HTML>

[Access: 22 mars 2017]

[16] *Tools for Apache Cordova*

<https://taco.visualstudio.com/en-us/docs/install-vs-tools-apache-cordova/>

[Access: 22 mars 2017]

[17] *Manage & Add plugins*

<https://taco.visualstudio.com/en-us/docs/manage-plugins/>

[Access: 22 mars 2017]

[18] *Core Plugins*

<https://docs.microsoft.com/sv-se/visualstudio/cross-platform/tools-for-cordova/take-further/core-plugins>

[Access: 22 mars 2017]

[19] *Microsoft Azure Mobile Apps*

<https://docs.microsoft.com/en-us/azure/app-service-mobile/app-service-mobile-value-prop>

[Access: 22 mars 2017]

[20] *Cordova Extra Components*

<https://taco.visualstudio.com/en-us/docs/install-vs-tools-apache-cordova/#choose>

[Access: 22 mars 2017]



## **Ionic2**

[21] *Overview*

<https://ionicframework.com/docs/components/#overview>

[Access: 25 mars 2017]

[22] *Native Plugins*

<https://ionicframework.com/docs/native/>

[Access: 25 mars 2017]

[23] *Components*

<https://ionicframework.com/docs/components/#alert-confirm>

[Access: 25 mars 2017]

## **Android Studio & Development**

[24] *Android Studio Official Site*

<https://developer.android.com/studio/features.html>

[Access: 29 April 2017]

[25] *SDK Manager*

<https://developer.android.com/studio/command-line/sdkmanager.html>

[Access: 29 April 2017]

[26] *Gradle Build Settings*

<https://developer.android.com/studio/build/index.html#settings-file>

[Access: 29 April 2017]

[27] *Button Widget*

<https://developer.android.com/reference/android/widget/Button.html>

[Access: 29 April 2017]

[28] *Layouts*

<https://developer.android.com/guide/topics/ui/declaring-layout.html>

[Access: 29 April 2017]

[29] *XML*

[https://www.w3schools.com/xml/xml\\_what.asp](https://www.w3schools.com/xml/xml_what.asp)

[Access: 29 April 2017]

[29] *Activities*

<https://developer.android.com/guide/components/activities/intro-activities.html>

[Access: 29 April 2017]

[30] *Activity-lifecycle*

<https://developer.android.com/guide/components/activities/activity-lifecycle.html>

[Access: 29 April 2017]

## Google API

[31] *Setup Google Play Services*

<https://developers.google.com/android/guides/setup>

[Access: 29 April 2017]

[32] *Camera Source*

<https://developers.google.com/android/reference/com/google/android/gms/vision/CameraSource>

[Access: 29 April 2017]

[33] *BarcodeDetector Documentation*

<https://developers.google.com/android/reference/com/google/android/gms/vision/barcode/BarcodeDetector>

[Access: 29 April 2017]

[34] *Google API – Visions package*

<https://developers.google.com/android/reference/com/google/android/gms/vision/package-summary>

[Access: 29 April 2017]

## Scrum

[35] *Scrum Daily Events*

<http://www.scrumguides.org/scrum-guide.html#events-daily>

[Access: 5 Maj 2017]

[36] *Scrum Product Backlog*

<http://www.scrumguides.org/scrum-guide.html#artifacts-productbacklog>

[Access: 5 Maj 2017]

[37] *Agile Manifest*

<https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/>

[Access: 5 Maj 2017]

[38]

<http://scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf#zoom=100>

[Access: 5 Maj 2017]

## JavaScript

[39] Crockford, Douglas. 2008. *JavaScript: The Good Parts*. O'Reilly Media, Inc

## 8 Appendix A Webb-native plugins

---

### 8.1 QR-kod skanning

```
// QR-Scanner programkod
private Scan() {
  cordova.plugins.barcodeScanner.scan(
    (result: any) => {
      alert("We got a barcode\n" +
        "Result: " + result.text + "\n" +
        "Format: " + result.format + "\n" +
        "Cancelled: " + result.cancelled);
    }, (error: any) => {
      alert("Scanning failed " + error);
    }, {
      "preferFrontCamera" : false, // iOS och Android
      "showFlipCameraButton" : true, // iOS och Android
      "prompt" : "Place a barcode inside the scan area", // Android endast
      "formats" : "QR_CODE,PDF_417",
      "orientation" : "landscape" // Android endast (portrait|landscape)
    }
  );
}
```

### 8.2 Kamera – Tagning av foto

```
// Kamera programkod för tagning av foto
private capturePhoto() {
  navigator.device.capture.captureImage(
    (captures: MediaFile[]) => {console.log(captures.length + " captured photos");
  },
  (error: CaptureError) => {alert( "Error " + error.message); },
  { limit: 3 }
  );
}
```

### 8.3 Filer – Hämtning av en JPG fil

```
// Hämtar ett en .JPG fil från enheten
private getFile() {
  navigator.camera.getPicture(
    (data: string) => {alert("Got photo"); },
    (message: string) => {alert("Failed: " + message); },
    {
      allowEdit: true,
      destinationType: Camera.DestinationType.FILE_URI,
      encodingType: Camera.EncodingType.JPEG,
      sourceType: Camera.PictureSourceType.PHOTOLIBRARY,
      quality: 80
    }
  );
}
```

## 8.4 Push Notiser

```
let push = PushNotification.init({
  android: {
    senderID: "12345679" /* Värdet för SenderID ska ersättas med ett nyckelvärde från
                        en backend tjänst för fungerande push notiser */
  },
  ios: {
    alert: "true",
    badge: true,
    sound: 'false'
  },
  windows: {}
});

push.on('registration', (data) => {
  console.log(data.registrationId);
});

push.on('notification', (data) => {
  console.log(data.message);
  console.log(data.title);
  console.log(data.additionalData);
});

push.on('error', (e) => {
  console.log(e.message);
});
```

## 9 Appendix B Android Prototyp

### 9.1 Java-fil

```
public class MainActivity extends AppCompatActivity {

    private SurfaceView cameraPreview;
    private TextView textResult;
    private BarcodeDetector barcodeDetector;
    private CameraSource cameraSource;

    final int RequestCameraPermissionID = 1001; // ÅtkomstID för kameran

    public static final int EMAIL = 2;
    public static final int PHONE = 4;
    public static final int SMS = 6;
    public static final int TEXT = 7;
    public static final int URL = 8;
    public static final int WIFI = 9;

    // Säkerställer tillstånd av kamera-användning under exekvering.
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
    @NonNull int[] grantResults) {
        switch (requestCode) {

            case RequestCameraPermissionID: {
                if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {

                    if (ActivityCompat.checkSelfPermission(this,
                    android.Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {

                        return;
                    }
                    try {
                        cameraSource.start(cameraPreview.getHolder());
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            }
        }
    }

    // Skapar alla komponenter som behövs för prototypen i denna metod.
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        cameraPreview = (SurfaceView) findViewById(R.id.scanningPreview);
        textResult = (TextView) findViewById(R.id.textResult);

        barcodeDetector = new BarcodeDetector.Builder(this)
            .setBarcodeFormats(Barcode.QR_CODE)
            .build();

        cameraSource = new CameraSource
            .Builder(this, barcodeDetector)
            .setRequestedPreviewSize(640, 480)
            .build();
    }
}
```

```

cameraPreview.getHolder().addCallback(new SurfaceHolder.Callback() {
    @Override
    // Säkerställer kamera-användning för ytan där kamera-vyn ska placeras.
    public void surfaceCreated(SurfaceHolder holder) {

        if (ActivityCompat.checkSelfPermission(getApplicationContext(),
android.Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(MainActivity.this, new
String[]{android.Manifest.permission.CAMERA}, RequestCameraPermissionID );
            return;
        }

        try {
            cameraSource.start(cameraPreview.getHolder());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @Override
    //Metod för ändringar i ytan, minimal kod då det sker minimala förändringar.
    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {

    }

    @Override
    //När ytan förstörs stängs även kameran ned med cameraSource.stop()
    public void surfaceDestroyed(SurfaceHolder holder) {
        cameraSource.stop();
    }
});

//Detektorns setProcessor() tillåter skanning av ramar under tiden kameran används
barcodeDetector.setProcessor(new Detector.Processor<Barcode>() {
    @Override
    public void release() { }

public void receiveDetections(Detector.Detections<Barcode> detections) {

    final SparseArray<Barcode> qrCodes = detections.getDetectedItems();

    if(qrCodes.size() != 0){
        textResult.post(new Runnable() {
            @Override
            public void run() {

                textResult.setText(qrCodes.valueAt(0).displayValue);

            }
        });
    }
});
}

```

## 9.2 XML-fil

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context="com.example.hamzawi.qrprototype.MainActivity">

    <SurfaceView
        android:id="@+id/scanningPreview"
        android:layout_width="match_parent"
        android:layout_centerInParent="true"
        android:layout_height="185dp" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/resultText"
        android:textAlignment="center"
        android:layout_marginTop="20dp"
        android:id="@+id/textResult"
        android:layout_below="@+id/scanningPreview"
        android:layout_centerHorizontal="true" />

</RelativeLayout>
```