CHALMERS

.

System för larmövervakning via smartklocka

Examensarbete inom Högskoleingenjörsprogrammet i data-teknik

AZER VILIC
ELIN KARLSSON

Institutionen för data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2014

System för larmövervakning via smartklocka.

Azer Vilic, Elin Karlsson

Institutionen för data- och informationsteknik
Chalmers tekniska högskola
412 96 Göteborg
Tel: 031-772 1000
Fax: 031-772 3663

[Bilden föreställer Samsung Gear 2 smartklocka]

Institutionen för data- och informationsteknik
Göteborg, 2014

Sammanfattning

Denna rapport beskriver utvecklingen av ett system för larmövervakning där det är möjligt att kontrollera larmet med en smartklocka. Systemet består av en smartklocka, en smarttelefon, en egenutvecklad webbtjänst samt en redan befintlig webbtjänst. Smarttelefonen kommunicerar med den egenutvecklade webbtjänsten som i sin tur kommunicerar med den befintliga webbtjänsten. Systemet kontrolleras med smartklockan som kommunicerar med smarttelefonen genom Bluetooth. Systemet är heterogent, flera olika plattformar har använts och systemet har utvecklats i ett antal olika miljöer och språk: JavaScript, C#, Java, HTML5, CSS och XML.

Smartklockan som användes i projektet är en Samsung Gear 2 med det nya operativsystemet Tizen. Samsung Gear 2 släpptes på marknaden 2014. En smartklocka är en datoriserad klocka med pekskärm. Smartklockan kan kommunicera med smarttelefonen för att få tillgång till dess funktioner, som till exempel Internet, men kan även användas som en självständig enhet. Utvecklingen av systemet har innehållit olika delmoment. Rapporten beskriver implementeringen av de olika delmomenten samt det slutgiltiga resultatet. En färdig version av systemet har implementerats som klarar av att hämta status på larmet, samt tillkoppla och frånkoppla det med hjälp av både smartklockan och smarttelefonen.

Nyckelord: Smarttelefon, smartklocka, Tizen, blandad miljö, webbtjänst.

Abstract

This thesis describes the development of a system for alarm monitoring where it is possible to control the alarm with a smartwatch. The system is comprised of a smartwatch, a smartphone, a self-developed web service and an already existing web service. The smartphone communicates with the self-developed web service that in return communicates with the already existing web service. However the system is controlled with the smartwatch that communicates with the smartphone via Bluetooth. The system is heterogeneous, many different platforms have been used and the system has been developed in a wide range of environments and languages: JavaScript, C#, Java, HTML5, CSS and XML.

The smartwatch that was used in the project is a Samsung Gear 2, with the new operating system Tizen. Samsung Gear 2 was released on the market in 2014. A smartwatch is a computerized watch with a touch screen. The smartwatch can communicate with the smartphone to get access to its functionalities such as the Internet. However it can also be used as an standalone device. The development of the system has been comprised of different moments. This thesis will describe the implementation of these moments and describe the result. A finished version of the system has been implemented that is capable of retrieving the status of the alarm and also connect and disconnect it with the use of both the smartwatch and smartphone.

Keywords: Smartphone, Smartwatch, Tizen, mixed environment, webservice.

Förord

Denna rapport omfattar ett examensarbete inom institutionen för data- och informationsteknik på Chalmers Tekniska Högskola. Projektet genomfördes på Sigma IT & Management Sweden AB.

Vi vill tacka Linus Vidberg och Sigma Technology för att de gav oss detta examensarbete. Vi vill tacka våra handledare på Sigma, Joakim Jonsson och Niclas Gustafsson för deras stöd. Vi vill även tacka vår handledare på Chalmers Tekniska Högskola, Joachim von Hacht för hans stöd och vägledning med skrivandet av denna rapport.

Göteborg 2014-06-09
Azer Vilic & Elin Karlsson

Ordlista

Android:	*Ett öppet operativsystem främst förekommande i smarttelefoner och tablets. Bygger i grunden på Linux.*
Android projekt:	*Innehåller allt som behövs för att en applikation skall fungera på en Android telefon.*
Android manifest:	*Konfigurationsfil i Android projekt. Beskriver bland annat applikationens namn, rättigheter och version.*
Bluetooth:	*Trådlös kommunikation över kort avstånd.*
C#:	*Objektorienterat programspråk utvecklat av Microsoft. C# är byggt på C++ vilket gör den kompatibel med .NET och common language runtime.*
CSS:	*Cascading Style Sheet, ett språk för att beskriver designen på webbsidor.*
DUID:	*Device User ID, unikt id som hör till en enhet. I detta fall smartklockan.*
Gear Manager:	*Applikation till Samsungs smarttelefoner som har hand om funtkionalitet tillhörande smartklockan som att installera applikationer.*
Märkspråk	*Märkspråk är ett format för dokument bestående av särskilda textkoder. Koden i märkspråket ger instruktioner för hur ett datorprogram som till exempel en webläsare skall presentera bland annat text, bilder och layout. HTML är ett exempel på ett märkspråk.*
HTML:	*HyperText Markup Language, ett märkspråk som används för att utveckla webbsidor.*
HTTP :	*HyperText Transfer Protocol, kommunikationsprotokoll för överföring av webbsidor.*
IIS:	*Internet Information Service, serverprogramvara utvecklat av Microsoft.*
Java:	*Objektorienterat programspråk utvecklat av Sun Microsystems.*

JavaScript:	*Dynamiskt skriptspråk som är prototyp-baserat. Kräver en viss exekveringsmiljö men kan köras på alla operativsystem.*
JSON:	*JavaScript Object Notation, textbaserat format för utbyte av data.*
.NET :	*Ett ramverk utvecklat av Microsoft. Ramverket är utvecklat så att det tillåter kommunikation med applikationer utanför .NET. .NET innehåller även flera klassbibliotek med förkodade lösningar på vanliga problem.*
RESTful:	*Arkitekturbegrepp som används vid utformning av webbtjänster. Varje resurs är unikt adresserbar via URI. Verben POST, GET, PUT och DELETE i HTTP-standarden används för att interagera med objekten.*
SAAgent:	*Android klass som sköter kommunikation mellan smartklocka och smarttelefon.*
Samsung apps:	*Onlinebutik för nedladdning av Samsungs applikationer.*
Samsung Mobile SDK:	*Samsungs mobila utvecklingssats. Ger tillgång till nödvändiga klasser för kommunikation med smartklocka.*
SDK:	*Software Development Kit, en uppsättning av utvecklingsverktyg som tillåter skapandet av applikationer för ett visst mjukvarupaket.*
Webbtjänst:	*En tjänst som gör det möjligt för två enheter att kommunicera över ett nätverk.*
Tizen:	*Operativsystem utvecklat av Samsung, baserad på Linux.*
UML:	*Unified Modelling Language, ett modelleringsspråk utvecklat för att visualisera arkitekturen av ett program genom att rita upp ett diagram.*
URI:	*Uniform Resource Identifier, en textsträng som används för att identifiera en resurs.*
WebKit:	*Webbläsare.*
Wrapper:	*En tjänst som omsluter en annan tjänst och gör det möjligt att använda viss funktionalitet i den senare tjänsten.*
XML:	*Extensible Markup Language, ett märkspråk. Definierar*

regler för att koda dokument som är läsbart både för människor och maskiner.

Innehållsförteckning

1. Inledning

1.1. Bakgrund

Allt fler produkter på marknaden kan kopplas upp till Internet och gränsen mellan datorer och andra enheter börjar suddas ut. En sådan enhet som är under stor utveckling är smartklockan.
En smartklocka är en datoriserad klocka med liknande funktioner och applikationer som en smarttelefon. Interaktionen med smartklockan sker genom en pekskärm.

Sigma är ett konsultföretag och leverantör av tjänster inom informationsteknik, management och informationslogistik. Sigma har tidigare utvecklat ett styrsystem för ett larm, till exempel skall man kunna aktivera larmet när man inte är hemma. Systemet består av en användarapplikation för smarttelefoner, där användaren kan se status på larmet samt tillkoppla eller frånkoppla det. Applikationen kommunicerar med en serverdel i form av en webbtjänst som i sin tur kontrollerar larmet.

Smartklockan är en relativt ny produkt på marknaden men som med stor sannolikhet kommer att bli allt mer populär i framtiden, samt följas av flera andra olika typer av smarta enheter. Sigma tror på en framtida marknad för smarta enheter där smartklockan kommer att få en mer central del. De är angelägna om att få mer kunskap om den nya tekniken som innefattar utvecklingen av applikationer för smartklocka. Sigma är därför intresserade utav att utveckla en ny version av deras tidigare uppdrag, larmsystemet för en smartklocka.

1.2. Syfte

Syftet med examensarbetet är att utöka och vid behov modifiera Sigmas larmsystem så att det kan styras via en smartklocka.

1.3. Avgränsningar

Funktioner för att styra larmet i serverdelen har redan utvecklats av Sigma. Det är vår uppgift att anropa funktionerna och få dem att fungera med smartklockan. En implementering av en webbtjänst kommer att utvecklas för att kommunicera med det befintliga systemet.
Vid tidsbrist kommer systemet att bli en prototyp, men kunskapen och forskningen inom området kommer att vara värdefull för Sigma.

1.4. Mål

Målet med projektet är att utveckla ett fullt fungerande system för att aktivera och deaktivera användarens larm. Utvecklingen kommer att ge Sigma en ökad förståelse för hur tekniken bakom smarta enheter fungerar.

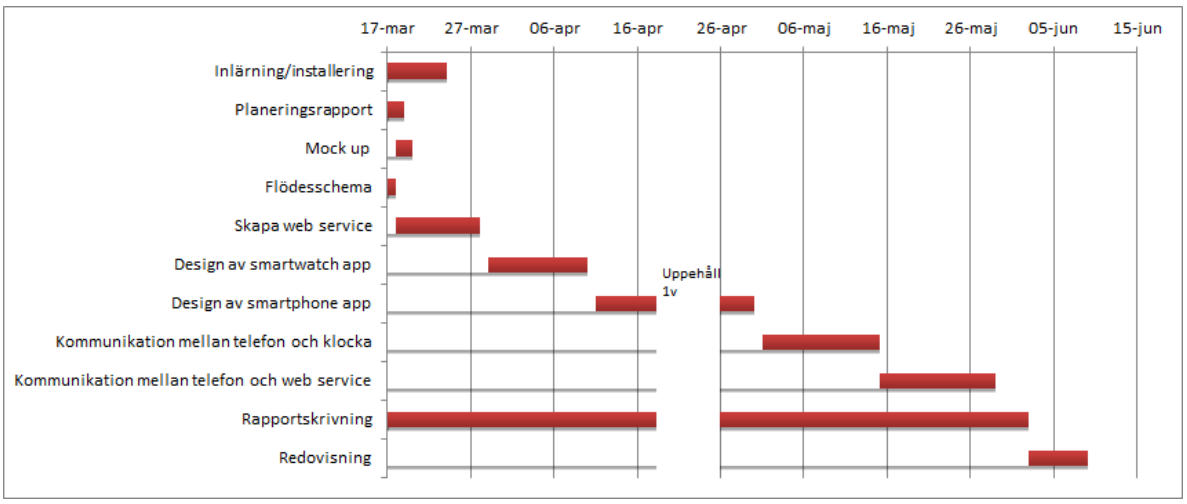
2. Metod

2.1. Deluppgifter

Projektet kommer att delas upp i fyra faser:

1. Första fasen består av instudering av hur Sigmas nuvarande system fungerar, planering som inkluderar mock ups, flödesscheman och user stories.
2. Eftersom SDK och API till Tizen inte fanns tillgänglig under början av projektet kommer den andra fasen bestå av att skapa en serverdel i form av en webbtjänst.
3. Fas tre kommer att bestå av att utveckla en applikation till Android mobiler för att kommunicera med smartklockan och webbtjänsten.
4. Sista fasen kommer att bestå av att utveckla applikationen till smartklockan samt sätta ihop de tre delarna

Utvecklingen kommer att ske på ett iterativt sätt. Varje fas kommer att bestå av planering, produktion, utvärdering samt testning av systemet. Detta kommer att vara en löpande process som återupprepas tills projektet blir färdigställt. Detta görs för att undvika att någon del av projektet inte blir fullt fungerande. I Gantt-schemat nedan visas tidplanen för projektet (figur 2.1).



Figur 2.1 Gantt-schema över tidplanen

3. Teknisk bakgrund

3.1. RESTful Service

REST står för Representional State Transfer och är ett arkitekturbegrepp som används för att utveckla webbtjänster. Alla resurser i en RESTful Service har en egen unik URI. Resurserna kan bli manipulerade genom att använda HTTP:s POST, GET, PUT och DELETE. REST bygger vanligen på HTTP. HTTP är ett kommunikationsprotokoll som används av Internet för att överföra webbsidor och information. HTTPS är en säkrare version av HTTP, där informationen som skickas över nätet krypteras och inte är synlig för utomstående [1]. En webbsida hämtas genom att klienten först skickar en förfrågan till servern med information om vilket objekt den vill hämta, servern svarar sedan genom att skicka det förfrågade objektet tillsammans med en svarstext [2]. Denna struktur gör att RESTful både är användarvänlig och är enkel att utveckla.

3.2. .NET

.NET är ett ramverk utvecklat av Microsoft. Språket som används i .NET är C#, som är ett objektorienterat språk även det utvecklat av Microsoft [3]. Ramverket innehåller ett stort bibliotek och körs i common language runtime. Biblioteket och common language runtime utgör tillsammans .NET.
För att köra webbaserade applikationer och för att få datorn att agera som en server krävs det att IIS (Internet Information Service) är installerat och körs på datorn. IIS är en webbserver bestående av en grupp Internet-servrar.

3.3. HTML5

Webbsidor skrivs i HTML**.** HTML5 är den senaste versionen av HTML och är numera en standard. HTML5 har gjort det enklare att utveckla applikationer som är anpassade för flera olika enheter och plattformar. Detta är viktigt eftersom det idag finns ett stort urval av enheter och plattformar att utveckla till. HTML5 gör det enklare att utveckla till flera av dessa samtidigt [4]. Två andra inbyggda språk, CSS och JavaScript används ofta tillsammans med HTML.

CSS är ett språk som bestämmer stilen på HTML-kod. Man kan till exempel ändra typsnitt, storlekar, positioner och färger. CSS lades till i HTML4 och sparar mycket tid när man designar applikationer, eftersom man i tidigare versioner var tvungen att beskriva stilen i varje HTML-tagg. CSS används även i HTML5[5].

JavaScript är ett skriptspråk, vilket innebär att språket endast behöver rätt exekveringsmiljö och kan köras på de flesta operativsystem utan större förändringar. JavaScript exekveras i en webbläsare och används tillsammans med HTML. Skripten kan läggas i alla HTML-taggar

och läsas i alla webbläsare [6]. Kommunikationen mellan smartklockan och smarttelefonen implementeras med hjälp av JavaScript i Tizen-applikationen. Det finns ett färdigt API att använda för detta [7].

3.4. Smartklocka

Samsung Gear 2 är Samsungs senaste smartklocka som släpptes på markanden 2014. Smartklockan kör operativsystemet Tizen som är utvecklat av Samsung. Operativsystemet kan användas i flera olika typer av smarta enheter som till exempel smarttelefoner, smartklockor och tv-apparater. Utveckling av applikationer i Tizen sker genom HTML5, XML, CSS och JavaScript. WebKit är en webbläsare med öppen källkod. WebKit är inkluderad i smartklockan, men den har inte tillgång till Internet [8]. Applikationerna installeras direkt i smartklockan, därför behöver inte smartklockan Internet för att hämta sidorna.

3.5. Kommunikation mellan smartklocka och smarttelefon

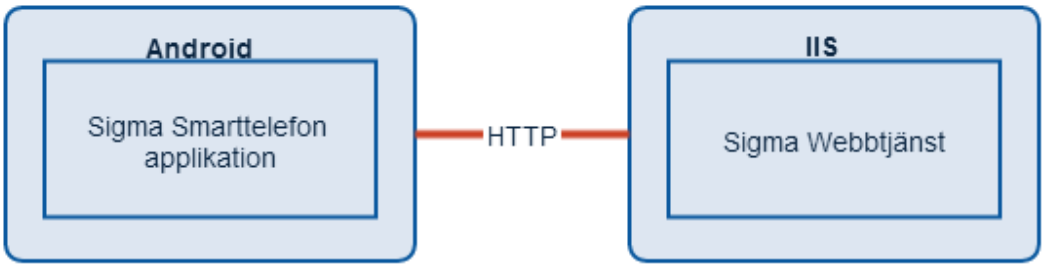
För att smartklockan skall kunna kommunicera med smarttelefonen behöver applikationen Gear Manager installeras genom Samsung apps till smarttelefonen. Gear Manager hanterar alla Gear-funktionaliteter som att ansluta smartklockan till smarttelefonen samt för att installera applikationer och liknande. Smartklockan kan för tillfället endast kommunicera med Samsungs smarttelefoner som har en lägsta Androidversion på 4.3. Kommunikationen mellan smartklockan och smarttelefonen sker genom Bluetooth. Bluetooth är en trådlös standard som används för att skicka data över korta avstånd. Bluetooth kommunicerar på en frekvens av 2.45 gigahertz [9]. Smartklockan och smarttelefonen kommunicerar genom Bluetooth 4.0 LEE [10].

3.6. Android

Android är ett öppet operativsystem som är utvecklat för att användas i smarttelefoner och tablets. Programmeringsspråket i ett Android-projekt är Java. Java är ett objektorienterat språk. För att etablera kommunikation mellan smarttelefonen och smartklockan krävs det att Samsung Mobile SDK är installerat. Detta ger tillgång till paket som gör att smarttelefonen och smartklockan kan kommunicera genom sockets. Det finns i nuläget 16 självständiga paket i Samsung Mobile SDK. Det som används för kommunikationen mellan smarttelefonen och smartklockan är Accessory-paketet. Paketet tillhandahåller ett ramverk kallat Samsung Accessory Framework som hanterar datautbytet mellan enheterna.

3.7. Sigmas befintliga larmsystem till smarttelefon

Sigmas befintliga system kan aktivera ett larm samt deaktivera det och avläsa larmets status genom en applikation på telefonen. Serverdelen består av en webbtjänst, utvecklad i .NET som är en del av Microsofts plattform. Applikationen är utvecklad i C#. Webbtjänsten har fyra resurser som vårt system anropar. Resurserna är Login, GetStatus, SetOn och SetOff. Webbtjänsten skickar data i form av JSON-objekt. I JSON-objekten finns information som meddelande, status, statusAnnex, time och user. Sigma har även skapat en Androidapplikation som kan kontrollera larmet. Figur 3.1 beskriver hur systemet ser ut.



Figur 3.1 Flödesschema över Sigmas tidigare system

4. Genomförande

Detta kapitel ger en övergripande beskrivning av hur systemet skall fungera.

4.1. Funktionella krav

Smarttelefon

1. Smarttelefonen skall kunna avläsa larmets status.
2. Smarttelefonen skall kunna aktivera larmet.
3. Smarttelefonen skall kunna deaktivera larmet.
4. Smarttelefonen skall kunna logga in i systemet.

Smartklocka

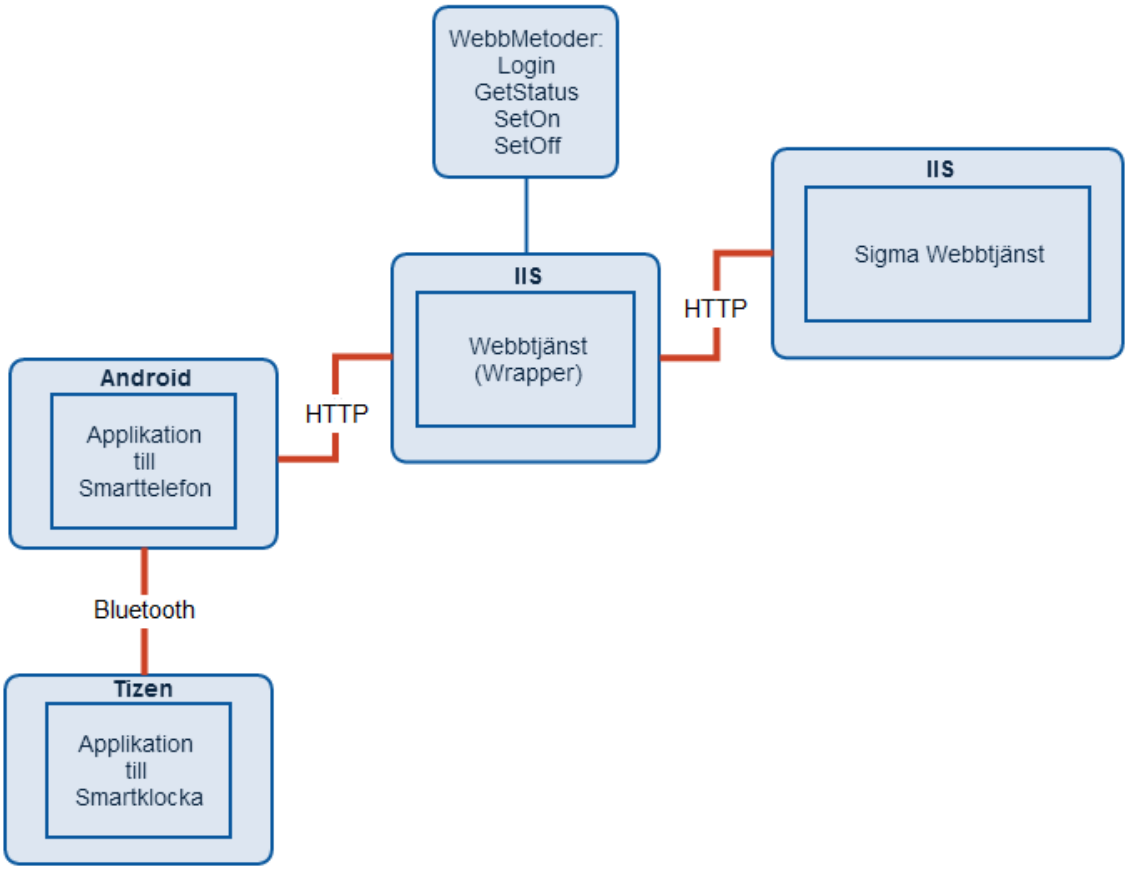
1. Smartklockan skall kunna avläsa larmets status.
2. Smartklockan skall kunna aktivera larmet.
3. Smartklockan skall kunna deaktivera larmet.
4. En pinkod krävs vid aktivering och deaktivering av larmet

För mer utförliga användningsfall se appendix.

4.2. Arkitektur

Systemet består av fyra delar (se figur 4.1):

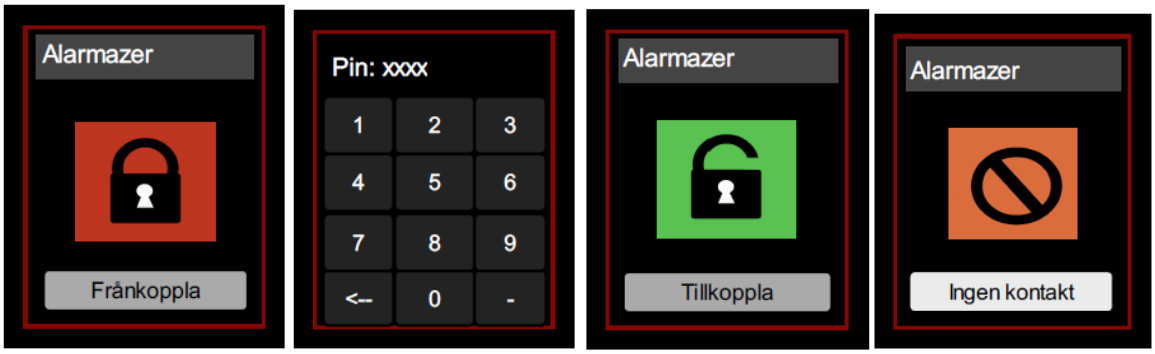
1. Återanvändning av Sigmas befintliga webbtjänst.
2. Egen webbtjänst som används som en wrapper till den befintliga eftersom endast några av den befintliga webbtjänstens metoder skall användas.
3. Ny applikation till smarttelefon. Eftersom smartklockan inte har direkt tillgång till Internet, måste smartklockan först kommunicera med en applikation i smarttelefonen som hanterar webbanropen. Därför bestämdes det att först utveckla en fungerande applikation för smarttelefonen före smartklockan.
4. Applikation till smartklockan.



Figur 4.1 Struktur över larmsystemet

4.3. Preliminärt användargränssnitt

Användargränssnittet planerades i början av projektet (se figur 4.2). Det slutgiltiga resultatet blev mycket likt det planerade gränssnittet.



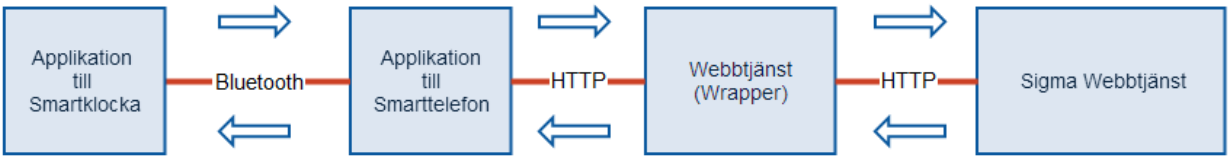
a) *b)* *c)* *d)*

Figur 4.2. Mockups från planeringen av projektet.

4.4. Flöde

Figur 4.3 visar flödet i larmsystemet, för att förtydliga flödet beskrivs nedan vad som sker när larmet stängs av.

1. Användaren klickar på knappen frånkoppla på smartklockans pekskärm (se figur 4.2a).
2. Systemet visar en ny sida på smartklockan, där en knappsats och ett textfält syns (se figur 4.2b).
3. Användaren slår koden som består av fyra siffror. De inslagna siffrorna syns i textfältet.
4. När hela koden är inslagen visas en ny sida på smartklockan med texten “vänligen vänta”, eftersom det tar några sekunder att utföra anropet till webbtjänsten.
5. Smartklockan gör om koden till en array av bytes samt krypterar den innan den skickas via Bluetooth till smarttelefonen.
6. Smarttelefonen tar emot koden, dekrypterar den och gör om den till en textsträng.
7. Smarttelefonen validerar koden.
8. Om koden är godkänd anropar smarttelefonen vår webbtjänsts metod SetOff(); via POST anrop.
9. Vår webbtjänst anropar Sigmas tjänst som slår av larmet och skickar ett JSON-objekt med information tillbaka till vår webbtjänst.
10. Vår webbtjänst får svar från Sigmas tjänst och skickar detta vidare till smarttelefonen.
11. Smarttelefonen får svar från webbtjänsten och gör om textsträngen till ett JSON-objekt samt hämtar meddelandet i objektet.
12. Om allting gick bra skickas en sträng med texten “true” tillbaka till smartklockan via Bluetooth.
13. Smartklockan får svar från smarttelefonen. Eftersom det var rätt kod uppdaterar klockan sitt utseende genom att visa en bild med ett frånkopplat lås samt en knapp med texten “Tillkoppla” (se figur 4.2c).



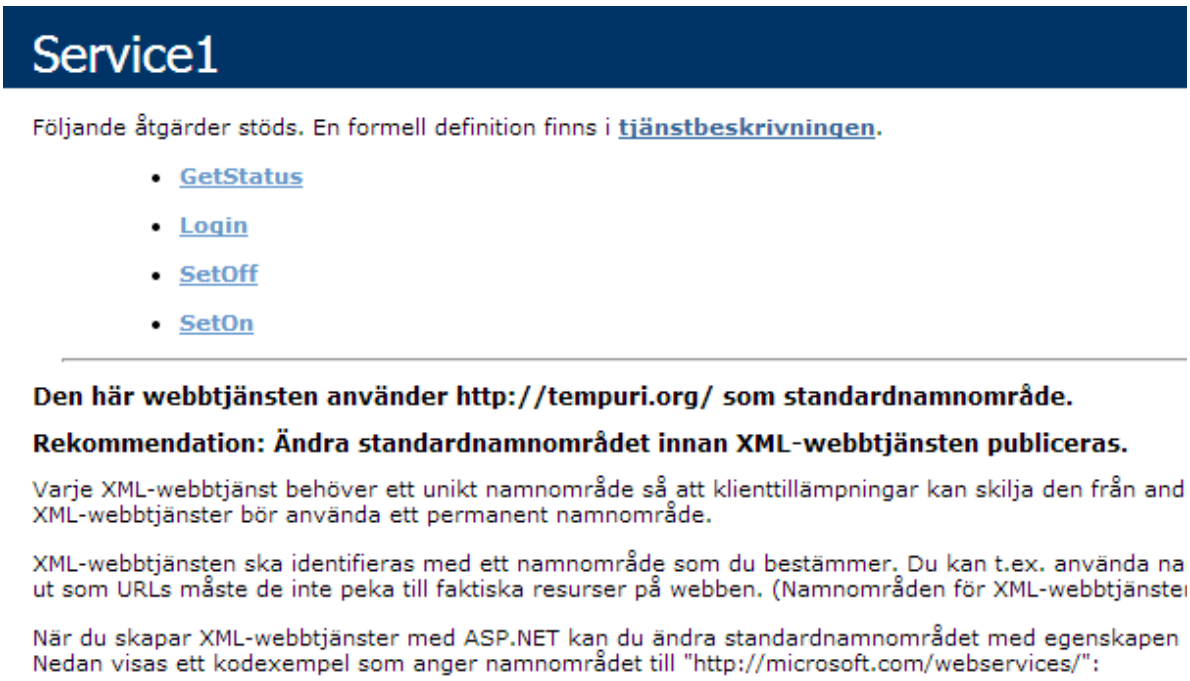
Figur 4.3. Flödet mellan de olika applikationerna i larmsystemet.

4.5. Implementation av webbtjänsten

Inledningsvis utvecklades det en webbtjänst i ASP.NET i C#. Webbtjänsten användes endast för testning och kördes därför lokalt och kunde endast nås av enheter på samma nätverk. Webbtjänsten består av en klass och fungerar som en "wrapper" till Sigmas befintliga webbtjänst. Fyra resurser har valts ut från det befintliga systemet och kan anropas direkt genom vår webbtjänst. En skärmdump över testkörningen av resursmetoderna visas på bild 4.4. Varje resursmetod anropar en URL-sträng, alltså Sigmas webbtjänst. Sigmas webbtjänst svarar med JSON-objekt, i objektet visas det ifall anslutningen lyckades och vilken status larmet har för tillfället. För att kunna köra webbtjänsten krävs det att IIS är installerat på datorn.

De fyra resursmetoder som är deklarerade i webbtjänsten ser ut på följande sätt:

```
public string Login(string username, string psw, string panelname, string pin, string deviceId)
public string GetStatus(string username, string psw, string panelname, string pin, string deviceId)
public string SetOn(string username, string psw, string panelname, string pin, string deviceId)
public string SetOff(string username, string psw, string panelname, string pin, string deviceId)
```



Figur 4.4: Bild av testkörning av webbtjänsten.

Nedan visas ett exempel på hur en resursmetod i webbtjänsten är implementerad, här hämtas status på larmet. Med hjälp av inparametrarna vet man vilken användare det är som frågar efter status samt vilket larm användaren vill nå. Parametrarna läggs till i URL-strängen.

```
[WebMethod]
    public string GetStatus(string username, string psw, string panelname, string pin, string
deviceId)
    {
        WebClient client = new WebClient();
        String downloadString = client.DownloadString("http://…=username..=psw….");
        return downloadString;
    }
```

4.6. Implementation av applikation till smarttelefon

Eftersom smartklockan inte har direkt tillgång till Internet måste smartklockan först kommunicera med en applikation i smarttelefonen som hanterar anropen till webbtjänsten. I det tidigare systemet finns redan en fungerande applikation för smarttelefoner. Däremot fungerar den inte tillsammans med en smartklocka. Implementationen av den nya applikationen till smarttelefonen innehåller enbart de nödvändiga funktionerna som behövs för att smartklockan ska fungera. Det grafiska användargränssnittet har till exempel inte prioriterats.

4.6.1. Grafiskt användargränssnitt

Applikationens grafiska användargränssnitt består av fyra knappar och en textruta (se figur 4.5). Klickar man på Login, loggas man in i systemet och kan sedan kontrollera larmet. Klickar man på Status och är inloggad, hämtas status på larmet, alltså påslagen/avslagen. Klickar man på On och är inloggad, slås larmet på. Klickar man på Off och är inloggad, slås larmet av. Det grafiska användargränssnittet består av XML-filen activity_main.xml.

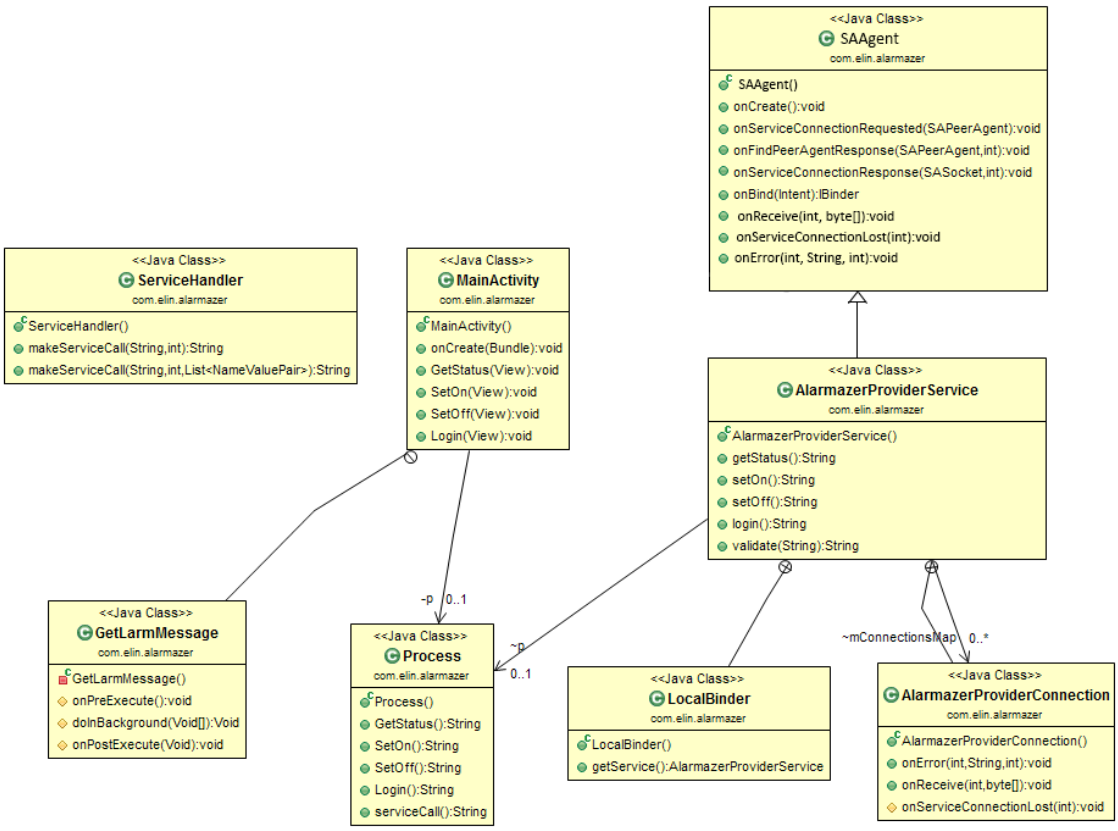
activity_main.xml - XML-filen beskriver hur startsidan skall se ut. Den bestämmer position, storlek och stil på knapparna samt textrutan.

Figur 4.5 applikation till smarttelefon

4.6.2. Förklaring av klasser i applikationen

Applikationen består av fyra Java-klasser samt en XML-fil för startsidan som är beskriven tidigare (se figur 4.6).

- **MainActivity** - Klassen körs när applikationen startas, den tar emot anrop från de fyra knapparna och uppdaterar textfältet med information.
- **ServiceHandler** - Klassen sköter alla anrop till Internet.
- **Process** - Använder klassen ServiceHandler för att utföra webbanrop. Klassen innehåller URL-erna som behövs för att anropa webbtjänsten, samt variablerna som skall skickas med.
- **AlarmazerProviderService** - Klassen sköter all kommunikation med smartklockan, den etablerar en kontakt med smartklockan, tar emot data som



smartklockan skickar, samt skickar data till smartklockan.

Figur 4.6: UML-diagram över smarttelefon applikationen

När applikationen på smarttelefonen startas körs klassen MainActivity. Klassen tar hjälp av activity_main.xml för att rita ut det grafiska användargränssnittet. Klassen innehåller Asynk Task som visar en dialog över processen samtidigt som den gör ett webbanrop. När klassen fått svar från anropet stängs dialogen ner och textfältet uppdateras. Det är även nödvändigt att använda Asynk Task eller liknande trådhantering eftersom programmet annars stannar medan det väntar på svar från webbtjänsten. Används Asynk Task är det möjligt att göra dessa anrop i bakgrunden av programmet och låta resten av programmet fortsätta att köra. För att göra webbanrop används klassen Process.

Klassen Process innehåller fyra metoder, GetStatus(), SetOn(), SetOff(), och Login(). När en av dessa metoder anropas väljs rätt URL och metoden makeServiceCall i klassen ServiveHandler anropas. Strängen som den får tillbaka efter anropet görs om till ett JSON-objekt och de olika delarna läggs i parametrar och returneras till klassen som gjorde anropet.

Klassen ServiceHandler kan göra POST och GET anrop och returnerar resultatet av anropet i form av en sträng. Klassen har en metod makeServiceCall() som tar en URL-sträng som inparameter, en metodtyp som antingen kan vara POST eller GET, samt en lista med variabler som inparametrar. Det är möjligt att göra anrop med eller utan parametrar.

När smarttelefonen får ett meddelande från smartklockan körs klassen AlarmazerProviderService. Klassen utför alla uppgifter i en arbetstråd så att resten av applikationen kan fortsätta att köras som vanligt. Klassen använder Process för att göra anrop till webbtjänsten precis på samma sätt som MainActivity. (AlarmazerProviderService beskrivs mer utförligt under kapitel 4.8.3 Smarttelefonens kommunikation med smartklockan).
Kopplingen mellan klasserna visas även i UML-diagrammet ovan (se figur 4.6).

4.7. Implementation av applikation till smartklocka

4.7.1. Certifikat/kodsignering

För att kunna testköra och kompilera koden på smartklockan och inte enbart på emulatorn behövs ett certifikat från Samsung. Detta är ett krav från Samsung för att skydda sin applikation från modifikation och för att förstärka äganderätten av applikationen. Detta försäkrar även att applikationen enbart kan köras på testenheter som man själv specificerat. Kodsignering behövs även för att ladda upp sin applikation till Samsung Apps [11].
Det går att generera en certifikat-förfrågan genom Tizen. Detta görs genom att trycka på “generate certificate request” och fylla i personuppgifter samt DUID (Device Unique Identifier). DUID hämtas genom att koppla in smartklockan i datorn med hjälp av en USB-kabel, öppna Tizen IDE, högerklicka på enheten i “Connection Explorer” och sedan välja “properties”. DUID finns under “Info”. För att få certifikatet godkänt måste certifikat-

förfrågan skickas till Samsung Developer Center via mail. För att göra detta på ett säkert sätt skall certifikatet först krypteras med hjälp av PGP. Man skapar först sin egen PGP-nyckel och efter det importerar man Samsung Developer Center PGP key. Efter det krypterar man sin "certificate request" med Samsungs publika nyckel och mailar sedan sin egna publika nyckel till Samsung tillsammans med det krypterade certifikatet. När Samsung developer center har dekrypterat filen och verifierat den skapas ett registrerings-certifikat som krypteras med användarens publika nyckel och skickas tillbaka via mail. Registrerings-certifikatet, som är en XML-fil, dekrypteras av användarens privata nyckel och kan sedan registreras i Tizen IDE med hjälp av "register certificate" knappen.

4.7.2. Applikationstyper för smartklocka

Det finns tre olika typer av Tizen wear applikationer:

1. **Standalone** - En applikation som kan användas direkt på smartklockan, utan koppling till smarttelefon. Ett exempel på detta är musikspelaren.
2. **Linked(master-follower)** - Applikationen består av två separata delar, en Andriod-applikation som körs på smarttelefonen samt en Tizen-applikation som körs på smartklockan. För att applikationen på smartklockan skall fungera krävs det att man först har installerat värdapplikationen på smarttelefonen.
3. **Integrated** - Består av en Android-applikation som innehåller en widget till smartklockan. Widgeten installeras automatiskt på smartklockan när applikationen installeras på smarttelefonen. Till skillnad från typen Linked kan man inte välja att enbart ha applikationen i mobilen. Raderas applikationen från smartklockan raderas den även från mobilen [12].

Eftersom det tidigare larmsystemet finns tillgängligt för smarttelefoner och om man i framtiden vill använda vårt system som ett tillbehör har typen Linked valts. Tanken är att man kan välja att lägga till applikationen i smartklockan och att den skall kunna fungera med det tidigare larmsystemet med några få modifieringar. Vi har utvecklat systemet så att det existerar en fullt fungerande applikation även på smarttelefonen där smartklockan fungerar som en förlängning. Typen Linked gör det möjligt att lägga ut en fullt fungerande applikation för smarttelefoner på Google Play. Applikationen kan användas av alla Android-telefoner. Man kan samtidigt lägga ut en applikation för smartklockan på Samsung apps. Denna applikation kan användas av de som äger en smartklocka från Samsung, samt har laddat ner applikationen för smarttelefonen.

4.8. Uppsättning av kommunikation mellan enheterna

För att de två applikationerna skall ha möjlighet att kommunicera med varandra krävs det att vissa filer existerar i de två projekten samt att vissa inställningar är definierade i konfigurationsfilerna. Följande stycken beskriver vad som behöver läggas till i de olika

filerna för att en koppling skall lyckas.

4.8.1. Android manifest

Följande rader skall läggas till i Android-projektets manifest-fil.
Raderna nedan ger tillåtelse till applikationen att använda Samsungs accessory framework, Bluetooth, smartklocka och notifikationer.

```
<uses-permission
android:name="com.samsung.accessory.permission.ACCESSORY_FRAMEWORK" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="com.samsung.wmanager.APP"/>
<uses-permission android:name="com.samsung.wmanager.ENABLE_NOTIFICATION"/>
[12]
```

Nedan specificeras vilken klass i projektet som sköter kommunikationen med smartklockan. Denna klass ärver SAAgent. I vårt projekt har vi döpt klassen till AlarmazerProviderService.

```
<service android:name="com.elin.alarmazer.AlarmazerProviderService" >

[12]
```

Följande kod används av Gear Mobile SDK vid installation av applikationen.

```
<receiver android:name="com.samsung.android.sdk.accessory.RegisterUponInstallReceiver" >
      <intent-filter>
         <action android:name="android.accessory.device.action.REGISTER_AFTER_INSTALL"
/>
      </intent-filter>
</receiver>
<receiver android:name="com.samsung.android.sdk.accessory
.ServiceConnectionIndicationBroadcastReceiver">
     <intent-filter>
          <action android:name="android.accessory.service.action
          .ACCESSORY_SERVICE_CONNECTION_IND" />
     </intent-filter>
</receiver>
[12]
```

Nedan specificeras vart XML-filen som beskriver service profilen har placerats, här kallat accessoryservices.xml. Filen beskrivs mer utförligt under rubriken Service profil.

```
 <meta-data
         android:name="AccessoryServicesLocation"
         android:value="/res/xml/accessoryservices.xml" />
[12]
```

Eftersom vi har valt att utveckla en applikation av typen Linked skall också följande rader läggas till i manifestet. Den första taggen beskriver namnet på applikationen till smarttelefonen. Den andra taggen beskriver namnet på paketet.

```
<meta-data
 android:name="master_app_name"
 android:value="AlarmazerProvider" />
<meta-data
 android:name="master_app_packagename"
 android:value="com.elin.alarmazer" />
[12]
```

Följande rader är frivilliga att lägga till.

```
<meta-data
 android:name="master_app_samsungapps_deeplink"
 android:value="Samsungapps deeplink URL" />
<meta-data
 android:name="master_app_playstore_deeplink"
 android:value="playstore deeplink URL" />
[12]
```

De används ifall man endast har installerat applikationen på smartklockan men inte på smarttelefonen. Eftersom applikationerna inte har publicerats på Google Playstore eller Samsung apps har dessa rader inte tagits med i manifestet i nuläget. Den första taggen beskriver länken till applikationen för smartklockan som ska ligga ute på Samsung apps. Den andra taggen beskriver länken till applikationen för smarttelefonen som skall ligga på Google Play.

4.8.2. Service profil

Service profilen är en XML-fil som förväntas ligga i mappen res/xml både i Android- och Tizen-projektet. Filerna används till att definiera kopplingen mellan enheterna. Applikationerna i de båda enheterna måste ha information om applikationsnamn, roller och vilken kanal som skall användas för överföring av data. Under <serviceProfile> specificeras ett gemensamt id och namn för Android- och Tizen-applikationerna samt en gemensam servicekanal, i det här fallet är id=”104”. Det som skiljer sig mellan de två filerna är application name, vilket skall vara namnet på respektive applikation, och role, vilket beskriver vilken roll applikationen har. Android-applikationen har alltid role=”provider”, eftersom den är värdapplikationen, Tizen har role=”consumer”. En annan sak som skiljer dem åt är att endast värdapplikationen behöver beskriva sökvägen till service implementationen.[12]

Android accessoryservices.xml

```
<resources>
   <application name="AlarmazerProvider" >
      <serviceProfile
         id="/system/Alarmazer"
         name="Alarmazer"
         role="provider"
serviceImpl="com.elin.alarmazer.AlarmazerPr
oviderService"
         version="1.0"
         serviceLimit="ANY"
              serviceTimeout="10">
         <supportedTransports>
            <transport type="TRANSPORT_BT"
/>
         </supportedTransports>
         <serviceChannel
            id="104"
            dataRate="low"
            priority="low"
            reliability= "enable"/>
   </serviceProfile>
   </application>
</resources>
```

Tizen accessoryservices.xml

```
<resources>
   <application name="Alarmazer" >
      <serviceProfile
         id="/system/Alarmazer"
         name="Alarmazer"
         role="consumer"
         version="2.0" >
         <supportedTransports>
            <transport
type="TRANSPORT_BT" />
         </supportedTransports>
         <serviceChannel
            id="104"
            dataRate="low"
            priority="low"
            reliability="enable" >
         </serviceChannel>
      </serviceProfile>
   </application>
</resources>
```

I config.xml i projektet i Tizen läggs följande rad till för att specificera placeringen av serviceprofilen.

```
<tizen:metadata key="AccessoryServicesLocation" value="res/xml/accessoryservices.xml"/>
```
[12]

4.8.3. Smarttelefonens kommunikation med smartklockan

För att smarttelefonen och smartklockan skall kunna kommunicera med varandra måste det finnas en klass i Android-projektet som ärver klassen SAAgent, det är denna klass som tar emot och skickar data via Bluetooth. Nedan följer förklaringar av vissa metoder som är ärvda av SAAgent.

onServiceConnectionRequested() - När Samsung Accessory Framework får en anslutningsbegäran från konsument delen som stämmer överens med värdapplikationen kallar Samsung Accessory Framework på onServiceConnectionRequested(). Om en begäran accepteras anropar Samsung Accessory Framework på onServiceConnectionResponse() för att skicka ett SAAgent objekt som matchar med konsument delen.

send(channel, data) och **secureSend(channel, data)** - skickar data genom kanalen. Vi har valt att använda oss av secureSend(), eftersom meddelandet krypteras och inte kan avläsas av utomstående. Detta är viktigt eftersom vi både skickar en kod samt att vi skickar status på larmet. Variabeln channel skall ha samma nummer som valdes i filen accessoryservices.xml, i det här fallet 104.

onServiceConnectionLost() - anropas när kontakten mellan enheterna har brutits.

4.8.4. Smartklockans kommunikation med smarttelefonen

Applikationen till smartklockan innehåller en fil kallad main.js. Kommunikationen med smarttelefonen sköts genom denna fil på konsumentsidan. Nedan följer förklaringar på funktioner som används för kommunikationen i JavaScript-filen.

onerror(err) anropas när ett fel har uppstått, till exempel om det inte går att etablera en koppling med värdapplikationen.

```
function onerror(err) {
	console.log("err [" + err.name + "] msg[" + err.message + "]");
}
```

onconnect anropas när enheterna skall etablera en koppling och värdapplikationen har skickat en socket som svar på förfrågan.

```
var agentCallback = {
	onconnect : function(socket) {
		SASocket = socket;
		SASocket.setSocketStatusListener(function(reason){
			console.log("Service connection lost, Reason : [" + reason + "]");
			disconnect();
		});
	},
	onerror : onerror
};
```

Funktionen används för att etablera en koppling mellan enheterna om anropet kommer från rätt applikation sätts en lyssnare på kopplingen så att applikationen kan ta emot data från smarttelefonen.

```
var peerAgentFindCallback = {
	onpeeragentfound : function(peerAgent) {
		try {
			if (peerAgent.appName == ProviderAppName) {
				SAAgent.setServiceConnectionListener(agentCallback);
				SAAgent.requestServiceConnection(peerAgent);
			} else {
				alert("Not expected app!! : " + peerAgent.appName);
			}
		} catch(err) {
			console.log("exception [" + err.name + "] msg[" + err.message + "]");
		}
	},
	onerror : onerror
}
```

onsuccess(agents) anropas när applikationen har lyckats att etablera en koppling mellan enheterna.

```
function onsuccess(agents) {
	try {
		if (agents.length > 0) {
			SAAgent = agents[0];
			SAAgent.setPeerAgentFindListener(peerAgentFindCallback);
			SAAgent.findPeerAgents();
		} else {
			alert("Not found SAAgent!!");
		}
	} catch(err) {
		console.log("exception [" + err.name + "] msg[" + err.message + "]");
	}
}
```

connect() anropas när en koppling skall göras mellan de två enheterna, till exempel om man vill skicka något till smarttelefonen måste denna metod först ha anropats. Metoden etablerar en koppling om detta inte redan finns.
disconnect() tar bort kopplingen mellan enheterna.

```
function connect() {
        if (SASocket) {
      return false;
   }
        try {
                webapis.sa.requestSAAgent(onsuccess, onerror);
        } catch(err) {
                console.log("exception [" + err.name + "] msg[" + err.message + "]");
        }
}

function disconnect() {
        try {
                if (SASocket != null) {
                        SASocket.close();
                        SASocket = null;
                        createHTML("closeConnection");
                }
        } catch(err) {
                console.log("exception [" + err.name + "] msg[" + err.message + "]");
        }
}
```

4.8.5. Exempel på kommunikationen

Nedan följer ett exempel på kommunikation mellan smartklocka och smarttelefon. Exemplet visar hur smartklockan hämtar status på larmet genom smarttelefonen.

Funktionen getStatus() anropas när smartklockan ska hämta larmets status.

1. Funktionen connect() anrops för att se till att det finns en koppling mellan enheterna, om det inte finns någon koppling etableras en.
2. Raden sätter en lyssnare till svaret som smartklockan skickar tillbaka efter anropet. Svaret kommer att skickas till funktionen onreceiveStatus.
3. Raden skickar en krypterad textsträng med ordet “status” till smarttelefonen.

```
// Funktion i filen main.js i Tizen projektet för smartklockan
function getStatus() {
	try {
		connect();                                                    //1
		SASocket.setDataReceiveListener(onreceiveStatus);             // 2
		SASocket.sendSecureData(CHANNELID, "status");                 // 3
	} catch (err) {
		console.log("exception [" + err.name + "] msg[" + err.message + "]");
	}
}
```

Varje gång smarttelefonen tar emot ett meddelande från applikationen i smartklockan anropas metoden onReceive(int channel, byte[] data);

1. Metoden gör om arrayen av bytes till en sträng.
2. Eftersom alla meddelanden som skickas från smartklockan kommer till denna metod måste vi veta vad det är smartklockan vill göra.
3. Eftersom textsträngen “status” skickades anropas metoden getStatus(). Metoden avläser status på larmet som har beskrivits tidigare och lägger status i variabeln mess. Om larmet är tillkopplat blir mess=“armed” om larmet är frånkopplat blir mess=“disarmed” och om det av någon anledning inte gick att hämta status på larmet blir mess=“no_contact”.
4. En koppling till smartklockan skapas.
5. En ny tråd bildas för överföringen av strängen.
6. Strängen omvandlas till bytes, krypteras och sänds sedan till smartklockan

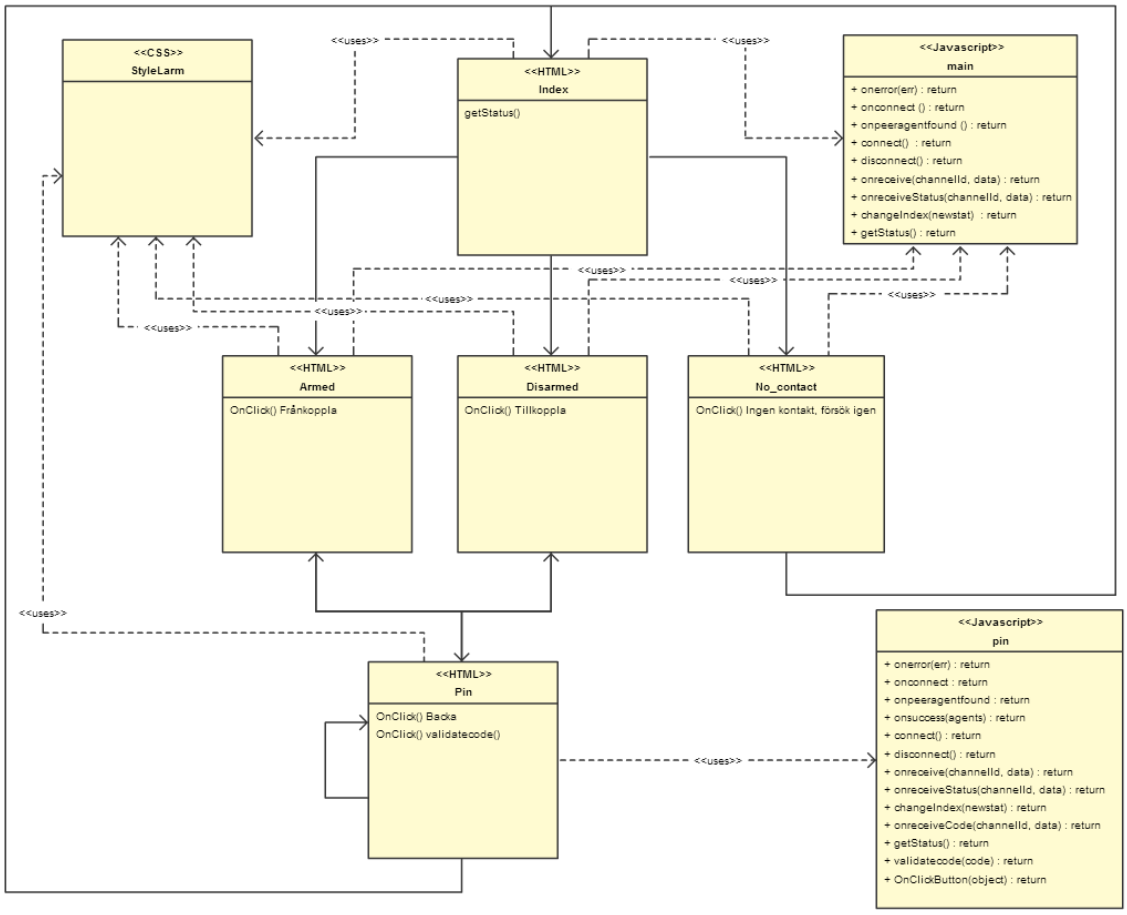
```
//Metod i klassen AlarmazerProvide.java  i Android projektet för smarttelefon
@Override
public void onReceive(int channelId, byte[] data) {
                String sentString = new String(data);                                   // 1
                String mess ="";
                if(sentString.equals("status")){                                        // 2
                        mess = getStatus();                                             // 3
                }
.
.
.
final String message = mess;
final AlarmazerProviderConnection uHandler = mConnectionsMap.get(Integer
.parseInt(String.valueOf(mConnectionId)));                                              //4
        if(uHandler == null){
                Log.e(TAG,"Error, can not get Connection handler");
                return;
        }
new Thread(new Runnable() {                                                             // 5
        public void run() {
                try {
                        uHandler.secureSend(CHANNEL_ID, message.getBytes());            // 6
                } catch (IOException e) {
                e.printStackTrace();
                }
        }
}).start();
```

1. Smartklockan får ett svar av smarttelefonen. Eftersom vi tidigare satte funktionen onreceiveStatus som en lyssnare till svaret kommer denna metod att anropas.
2. Om status på larmet är förändrat ska smartklockans utseende uppdateras

```
// Funktion i filen main.js i Tizen projektet för smartklockan
function onreceiveStatus(channelId, data) {                                    //1
        changeIndex(data);
}

function changeIndex(newstat) {                                                //2
        //Only change page if we are not already in it
        var path = window.location.pathname;
        var page = path.split("/").pop();

        if (newstat.valueOf() == armed.valueOf()) {
                if(page!="armed.html"){
                        document.location.href = "armed.html";
                }
                status="armed";

        } else if (newstat.valueOf() == disarmed.valueOf()) {
                if(page!="disarmed.html"){
                        document.location.href = "disarmed.html";
                }
                status="disarmed";

        } else {
                if(page!="no_contact.html"){
                        document.location.href = "no_contact.html";
                }
                status="no_contact";
        }
}
```

4.9. Larmapplikationen för smartklocka

Applikationen för smartklockan består av HTML-filer, CSS-filer, XML-filer och JavaScript-filer. I figur 4.7 visas ett UML-diagram över applikationen.



Figur 4.7 UML-diagram över applikationen till smartklockan.

HTML:

- **armed.html** - Filen visas när larmet är tillslaget.
- **disarmed.html** - Filen visas när larmet är frånslaget.
- **index.html** - Startsidan. Innan applikationen kan visa status på larmet måste den först hämtas. Index visar en “Vänligen vänta” - sida medan anropet till webbtjänsten utförs.
- **pin.html** - Filen visar en knappsats där man skall skriva in sin pinkod.
- **no_contact.html** - Filen visas när ett fel har inträffat, till exempel om Bluetooth är inaktiverat eller om Internet är avstängt.

CSS:

- **styleLarm.css** - Innehåller all design av HTML-sidorna i projektet.

JavaScript:

- **main.js** - används av alla HTML-filer utom pin.html. Den sköter kommunikationen med smarttelefonen och uppdaterar utseendet på smartklockan beroende på status på larmet. Varje gång applikationen startas hämtas status på larmet så att smartklockan alltid är uppdaterad. Om det av någon anledning inte går att hämta status visas en fel-sida som beskriver problemet.
- **pin.js** - används av pin.html för validering av koden. Klassen kommunicerar även med smarttelefonen.

XML:

- **accessoryservices.xml** - används till att koppla konsumentapplikationen till värdapplikationen.
- **config.xml** - konfigurations fil, innehåller information om startsida, applikations namn med mera.

HTML-filerna armed.html, disarmed.html och no_contact.html har alla samma layout. En överskrift med applikationsnamet finns längst upp på skärmen. En bild har valts för att beskriva status på larmet, detta för att det både blir visuellt snyggare och att man snabbt skall kunna se vilken status det är på larmet. En röd bakgrund har valts för tillkopplat och en grön för frånkopplat. Under bilden finns en knapp som tar användaren till pinkodsidan om man befinner sig i armed.html eller disarmed.html. Befinner man sig däremot i no_contact.html och klickar på knappen gör smartklockan ett nytt försök att hämta status och uppdaterar utseendet på smartklockan till armed.html, disarmed.html eller no_contact beroende på larmets status. HTML-filerna använder metoderna i JavaScripts-filen main.js för bland annat kommunikation med smarttelefonen. Pinkod sidan består av en knappsats och en textruta. Koden består av fyra siffror. När användaren har klickat på fyra siffror skickas koden automatiskt direkt till smarttelefonen för validering. JavaScript-filen pin.js innehåller alla metoder som pin.html använder. Alla HTML-filer använder CSS-filen styleLarm.css för designen på sidorna. När status på larmet avläses visas alltid en väntsida som finns i index.html. UML- diagrammet nedan beskriver sambandet mellan de olika filerna.

4.10. Driftsättning

För att driftsätta systemet krävs det att dessa steg utförs.

1. Starta den lokala webbtjänsten.
2. Se till att smarttelefonen har tillgång till samma trådlösa nätverk som webbtjänsten.
3. Installera Android-applikationen AlarmazerProvider på smarttelefonen med hjälp av USB-kabel.

4. Installera Tizen-applikationen Alarmazer på smartklockan med hjälp av USB-kabel.
5. När applikationen har installerats är applikationen klar att användas.

5. Resultat

Projektets mål har uppnåtts genom att vi har utvecklat en fungerande Tizen-applikation till en Samsung Gear 2 smartklocka, en Android-applikation till smarttelefon och en webbtjänst. Det går att aktivera ett larm, samt deaktivera larmet och avläsa larmets status genom smartklockan. Samma funktionalitet finns även i applikationen till smarttelefonen. Båda applikationerna fungerar som planerat och interagerar med varandra. Det blev inga avgränsningar beträffande vad som var ursprungsmålet och dess funktionalitet med varken Tizen-applikationen eller Android-applikationen. Webbtjänsten som vi skapat mot Sigmas webbtjänst har körts lokalt under projektets gång. Detta för att webbtjänsten har använts i ett testningssyfte och har varit en wrapper till Sigmas webbtjänst.

5.1 Användargränssnitt

Nedan visas användargränssnittet av den färdiga applikationen i smartklockan.

Figur 5.3 Ingen kontakt.

Figur 5.1 Larm frånslaget.

Figur 5.2 Larm tillslaget.

Figur *5.4 Pinkod*

Figur 5.5 Vänligen vänta

6. Diskussion

Projektet har innefattat många olika programmeringsspråk och blandad utvecklingsmiljö. Trots detta fungerar systemet som det skall utan större fördröjningar. Det är viktigt att varje del i ett projekt fungerar individuellt och att kopplingen mellan dessa fungerar korrekt. Något som varit väldigt givande var att göra mockups och planera hur Tizen-applikationen skulle fungera och se ut i god tid. Detta gjorde att det inte blev några oklarheter under själva kodningen.

6.1. Hämtning av information

Det har inte funnits mycket hjälp att hitta på Internet beträffande Tizen eftersom att Tizen för bärbara enheter nyligen släpptes. Den största informationskällan har varit Samsungs webbsida för Gear utvecklare. Sidan innehåller dokumentet "Getting started" för Gear-applikationer, vilket är ett dokument som beskriver hur man kommer igång med ett projekt. Dokumentet var enkelt att förstå och tog upp de viktigaste delarna. En sak som det inte tog upp var registrering av certifikat. Certifikatet behövdes för att köra applikationen i smartklockan och inte bara i Tizen-emulatorn. Eftersom vi inte hade läst något om detta krävdes det en del felsökning innan vi förstod att det var detta som hindrade oss från att kompilera koden på smartklockan. Att generera ett certifikat samt att få det registrerat var en relativt krånglig process. Vi har därför i vår rapport gett en mycket utförlig beskrivning av hur man gör för att generera och registrera ett certifikat. Webbsidan innehåller även ett antal exempel-applikationer till Tizen för smartklockan. Dessa har varit givande att studera och ger en bra förståelse för hur smartklockan och smarttelefonen kommunicerar och interagerar med varandra.

6.2. Blandad miljö

I projektet har det används flera olika programmeringsspråk och plattformar. Varje delmoment i systemet har utvecklats i ett eget programmeringsspråk. Smattelefonen har utvecklats i Java, webbstjänsten i C# och smartklockan i HTML5 och JavaScript. Det har varit smidigt att få dessa olika programmeringsspråk att kommunicera med varandra. Utbudet av programmeringsspråk samt plattformar är stort. De olika språken används till olika saker och är mer passande för olika uppgifter. Företagen som utvecklar plattformarna konkurrerar även med varandra och vill att deras egen plattform skall vara störst och smidigast. Det finns både fördelar och nackdelar med detta. En fördel är att företagen vill att deras plattform skall vara enkel att utveckla till samt att den skall fungera med flera olika typer av enheter. En stor nackdel är att man som utvecklare måste lära sig flera olika språk och utveckla till flera olika plattformar för att en applikation skall bli kompatibel med så många enheter som möjligt. Just utvecklingen av applikationer till Samsungs smartklocka där smartklockan och smarttelefonen kommunicerar med varandra kräver att man utvecklar i olika språk. Detta är en konsekvens av att Samsung valt att deras smartklockor skall köra

Tizen samtidigt som deras smarttelefoner i nuläget kör Android. En fördel som Samsungs plattform Tizen har är att applikationerna programmeras på precis samma sätt som en webbapplikation och skulle därför, med några få modifieringar, även kunna användas på alla enheter som har en inbyggd webbläsare. Att den använder sig utav HTML5 är även en fördel, eftersom den anpassar layouten utefter storleken på skärmen. Detta är viktigt, eftersom enheterna idag har väldigt olika utseenden.

6.3. Etiska aspekter

Det debatteras mycket om att människan i dagens samhälle är ständigt uppkopplad och åtkomlig via sin smarttelefon vart de än befinner sig. Därför kan det nog i framtiden debatteras mycket om smartklockan när den blir allt mer populär. Genom smartklockan har man tillgång till sms och samtal direkt via handleden. Smartklockan är inte något som är skapat för att stoppas ner i fickan, därför kan den möjligt ge en känsla utav att alltid vara väldigt åtkomlig.

6.4. Miljö aspekter

Samsung är först av de stora varumärkena som tillverkar smarttelefoner som börjat certifiera enligt TCO Certified SmartPhones. Samsungs nya smarttelefon Galaxy S4 är certifierad enligt TCO. Detta betyder att Galaxy S4 möter både miljökrav och krav på bra arbetsförhållanden i produktionen[13]. Förhoppningsvis kommer Samsungs nya smartklockor också att certifieras enligt TCO.

7. Rekommendationer och framtida arbete

Nedan följer några funktioner som diskuterades men inte lades till eftersom det inte fanns tid samt att de inte var inräknade i planeringen.

1. **Lista.** En lista som visar information om när larmet aktiverades eller deaktiverades. Listan skulle innehåll datum och tid, samt användare. Till detta behövs en databas som sparar informationen.
2. **Notifikationer.** Ett annat förslag var att klockan skulle få en notifikation varje gång någon aktiverar eller deaktiverar larmet.

8. Referenser

[1] Question: HTTP and HTTPS - What is the difference? *Ask Doug.* http://www.virtu-software.com/ask-doug/QandA.asp?q=7 (2014-04-30)

[2] Simtec Limited. (2014) Introduction. *HTTPWatch.* http://www.httpwatch.com/HTTPgallery/introduction/ (2014-04-29)

[3](2014) http://en.wikibooks.org/wiki/C_Sharp_Programming (2014-05-07)

[4] HTML5 introduction. *w3schools.com.* http://www.w3schools.com/HTML/HTML5_intro.asp (2014-04-29)

[5] CSS introduction. *w3schools.com.* http://www.w3schools.com/CSS/CSS_intro.asp (2014-04-29)

[6] JavaScript introduction. *w3schools.com.* http://www.w3schools.com/js/js_intro.asp (2014-04-29)

[7] Tizen. (2012) About. *Tizen.* https://www.tizen.org/about (2014-04-29)

[8] Tizen. (2014) Tizen SDK for Wearable 1.0.0b1 Release Notes. *Tizen.* https://developer.tizen.org/fr/downloads/sdk/release-notes/tizen-sdk-wearable-1.0.0b1?langswitch=fr (2014-04-13)

[9] Franklin, Layton, C.F, J.L http://electronics.howstuffworks.com/bluetooth2.htm (2014-04-29)

[10] Davies, C.D (2014) http://www.slashgear.com/samsung-gear-2-and-gear-2-neo-smartwatches-official-22317783/ (2014-04-30)

[11] Samsung (2014) http://developer.samsung.com/samsung-gear (2014-05-02)

[12] Samsung ElectronicsCo. (2014) Samsung Gear Application Getting started. *Samsung Developers.* http://img-developer.samsung.com/contents/cmm/SamsungGearApplication_GettingStarted_1.0.pdf (2014-05-16)

[13] Samsung Galaxy S4 first to achieve sustainability certification for smartphones. (2014) TCO Development http://tcodevelopment.com/news/samsung-galaxy-s4-first-to-achieve-sustainability-certification-for-smartphones/(2014-05-16)

Appendix

Användningsfall

Nedan listas systemets användningsfall.

Aktivera larmet:

1. Användaren klickar på knappen tillkoppla på smartklockans pekskärm.
2. Systemet visar en ny sida på smartklockan, där en knappsats och ett textfält syns.
3. Användaren slår koden som består av fyra siffror, de inslagna siffrorna syns i textfältet.
4. När hela koden är inslagen visas en ny sida på smartklockan med texten vänligen vänta eftersom det tar några sekunder att göra webbanropet.
5. Smartklockan gör om koden till en array av bytes samt krypterar den innan den skickas via Bluetooth till smarttelefonen.
6. Smarttelefonen tar emot koden, dekrypterar den och gör om den till en textsträng.
7. Smarttelefonen validerar koden.
8. Om koden är godkänd anropar smarttelefonen vår webbtjänsts metod SetOn(); via POST anrop.
9. Vår webbtjänst anropar Sigmas tjänst som slår av larmet och skickar ett JSON-objekt med information tillbaka till vår webbtjänst.
10. Vår webbtjänst får svar från Sigmas tjänst och skickar detta vidare till smarttelefonen.
11. Smarttelefonen får svar från webbtjänsten och gör om textsträngen till ett JSON-objekt samt hämtar meddelandet i objektet
12. Om allting gick bra skickas en sträng med texten “true” tillbaka till smartklockan via Bluetooth.
13. Smartklockan får svar från smarttelefonen, eftersom det var rätt kod uppdaterar klockan sitt utseende genom att visa en bild med ett frånkopplat lås samt en knapp med texten “Frånkoppla”.

Deaktivera larmet:

1. Användaren klickar på knappen frånkoppla på smartklockans pekskärm.
2. Systemet visar en ny sida på smartklockan, där en knappsats och ett textfält syns.
3. Användaren slår koden som består av fyra siffror, de inslagna siffrorna syns i textfältet.
4. När hela koden är inslagen visas en ny sida på smartklockan med texten vänligen vänta eftersom det tar några sekunder att göra webbanropet.
5. Smartklockan gör om koden till en array av bytes samt krypterar den innan den skickas via Bluetooth till smarttelefonen.
6. Smarttelefonen tar emot koden, dekrypterar den och gör om den till en textsträng.
7. Smarttelefonen vailderar koden.

8. Om koden är godkänd anropar smarttelefonen vår webbtjänsts metod SetOn(); via POST anrop.
9. Vår webbtjänst anropar Sigmas tjänst som slår av larmet och skickar ett JSON-objekt med informaton tillbaka till vår webbtjänst.
10. Vår webbtjänst får svar från Sigmas tjänst och skickar detta vidare till smarttelefonen.
11. Smarttelefonen får svar från webbtjänsten och gör om textsträngen till ett JSON-objekt samt hämtar meddelandet i objektet.
12. Om allting gick bra skickas en sträng med texten “true” tillbaka till smartklockan via Bluetooth.
13. Smartklockan får svar från smarttelefonen, eftersom det var rätt kod så uppdaterar klockan sitt utseende genom att visa en bild med ett frånkopplat lås samt en knapp med texten “Tillkoppla”.

Avläsning av larmets status

1. Varje gång statussidan laddas skapas det en koppling mellan smartklockan och smarttelefonen.
2. Smartklockan omvandlar strängen “status” till en array av bytes, krypterar arrayen och skickar den sedan till smarttelefonen via Bluetooth.
3. Smarttelefonen tar emot datan, dekrypterar den samt omvandlar den till en sträng.
4. Smarttelefonen anropar vår webbtjänsts metod GetStatus(); via POST anrop.
5. Vår webbtjänst anropar Sigmas tjänst som läser larmets status och skickar ett JSON-objekt med informaton tillbaka till vår webbtjänst.
6. Vår webbtjänst får svar från Sigmas tjänst och skickar detta vidare till smarttelefonen.
7. Smarttelefonen får svar från webbtjänsten och gör om textsträngen till ett JSON-objekt samt hämtar meddelandet i objektet.
8. Om allting gick bra skickas en sträng med texten “armed” eller “disarmed” tillbaka till smartklockan via Bluetooth. Om någonting gick fel i webbanropet, skickas strängen “no_contact”. Strängarna görs först om till arrayer av bytes samt krypteras innan de sänds.
9. Smartklockan får svar från smarttelefonen, svaret dekrypteras och görs om till en sträng. Om strängen innehåller ordet “armed” visas status sidan som beskriver att larmet är tillkopplat. Om strängen innehåller ordet “disarmed” visas status sidan som beskriver att larmet är frånkopplat. Om strängen innehåller “no_contact” visas status sidan som beskriver att något gick fel.

Hantering av pinkodsidan:

Pinkoden består av fyra stycken siffror. När en siffra trycks ned skrivs den automatiskt ut på skärmen efter texten “Pin:”. När fyra siffror är intryckta skickas koden automatiskt till smarttelefonen för validering. Under tiden koden valideras visas sidan med texten “Vänligen vänta”.

För att backa tillbaka till status sidan, klicka på knappen längst ner åt vänster i pinkodsidan.

För att sudda den senaste siffran, tryck på knappen längst ner åt höger på pinkodsidan.

Ingen kontakt med smarttelefon:

1. Om smartklockan inte har kontakt med smarttelefonen eller om smarttelefonen inte har kontakt med Internet så kan smartklockan inte visa status på larmet och kommer att visa en fel-sida som beskriver problemet.
2. Efter att användaren har kontrollerat Bluetooth kopplingen mellan smartklockan och smarttelefonen samt kontrollerat att smarttelefonen har tillgång till Internet kan användaren klicka på knappen “Ingen kontakt, försök igen”.
3. Efter att användaren har klickat på knappen försöker smartklockan att hämta status igen och visar sidan med texten ”Vänligen vänta”.
4. Beroende på om status kan avläsas och vilken status det i så fall är, visas någon av de tre status-sidorna.