BACHELOR TOEGEPASTE INFORMATICA

Bachelorproef

eID IO – Applicatie besturing

Maxim Van de Wynckel

Promotor: Dhr. C. Benoit
Mentor: Dhr. K. De Smedt
Academiejaar 2015-2016

Bachelorproef ingediend tot het behalen van de graad van bachelor in de toegepaste informatica

Download als PDF

Graag zou ik volgende personen willen bedanken bij het helpen van mijn opdracht:

Mijn vader Werner Van de Wynckel: Heeft altijd goed advies, zelfs bij onderwerpen waar hij niet alles van af weet kan hij me nog steeds verder helpen.

Sam Sun (https://twitter.com/samczsun https://github.com/samczsun): Geweldig in het reverse engineering en DRM in Java. Heeft mij geholpen enkele zwakheden in mijn websockets en Javaserver te vinden.

Simon MacDonald (https://github.com/macdonst): De push plugin die hij vrijwillig onderhoud is de enige in zijn soort die voldoende onderhouden wordt. Zelfs met een aantal tekortkomingen ben ik zeker dat hij het perfect zal krijgen mits hij goede support geeft.

Andreas Argelius (https://github.com/argelius): De maker van OnsenIO slaagt erin om een stabiel framework te behouden dat zowel simpel is en voldoet aan alle verwachtingen van de gebruikers. Veel gelijkaardige UI frameworks worden zeer snel bloated met features om gebruikers te lokken, maar dit zorgt vaker voor problemen.

Kristof De Smedt: Geven van de bachelor proef en het beschikbaar stellen van de backend code en kaartlezer.

Christophe Benoit: Begeleider van mijn final work. Bedankt voor het advies bij de tussentijdse evaluatie en helpdesk's.

Vrienden en familieleden: Voor het zo goed mogelijk nakijken van mijn tekst op duidelijke fouten of ontbrekende informatie.

1 INHOUDSOPGAVE

Inleiding

1 BITPOWER

Bitpower is een installatiebedrijf gespecialiseerd in automatisering of sturing van zonwering, opengaande ramen en rolluiken in grote gebouwen. Naast een adviserende rol voor fabrikanten, bieden ze een totaalpakket automatisatie aan: voorstudie, uitvoering, testing en oplevering. Hiervoor werken ze nauw samen met de zonnevakker/schrijnwerkers en de bouwheer om het juiste sturingssysteem te kiezen en conform de voorschriften te installeren. *(Kompass.be - Bitpower, 2016)*

1.1 EID IO

eID IO is een product waarmee je de toegang tot een gebouw of evenement kan beveiligen doormiddel van authenticatie met eID of RFID-tags. Het systeem is cloud gebaseerd waardoor het volledig gemonitord en beheerd kan worden via een online web platform waartoe klanten toegang hebben.

Wanneer er personen zich authentiseren aan een kaartlezer zal dit online in de cloud worden bewaard. Later kunnen deze gegevens dan worden bekeken op het online web platform.

Er wordt verder gewerkt op een reeds bestaand systeem, dit betekend dat er ook rekening moet gehouden worden met eventuele wijzigingen die zullen optreden in de backend.

1.1.1 Reeds geïmplementeerd

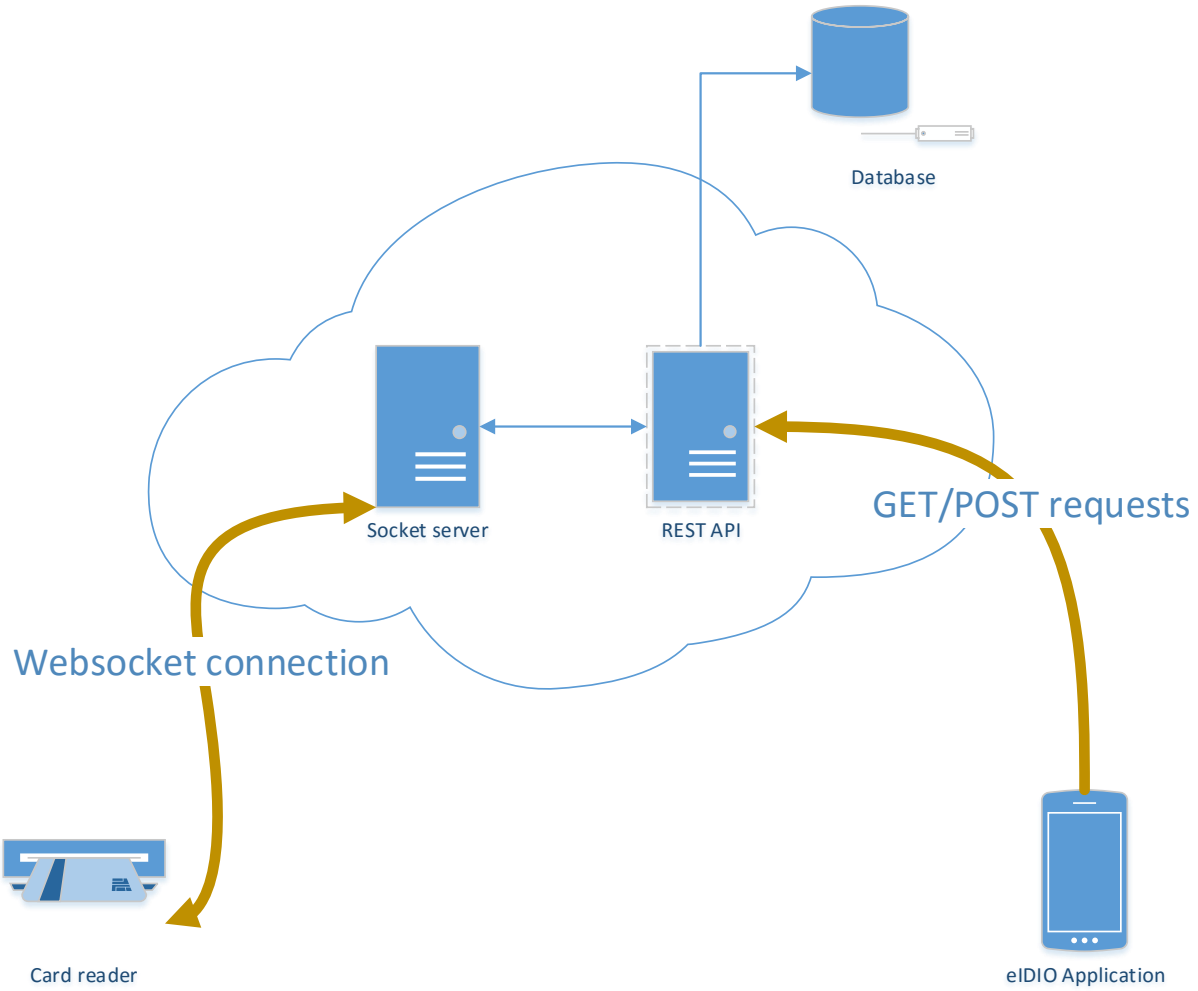
Het systeem wordt al in gebruik genomen en geïnstalleerd. De kaartlezers kunnen bijhouden in de cloud wie er binnenkomt en via een online web paneel kunnen klanten allerhande instellingen configureren van de kaartlezers.

Dit alles gebeurt allemaal via veilige API-calls die:

- IP locked zijn (gedefinieerd door installateur)
- Kaartlezers hebben een serienummer
- Kaartlezers hebben een private key

Gebruikers kunnen online inloggen via een gebruikersnaam en wachtwoord (welke momenteel gehashed wordt met SHA1).

2 OPDRACHT



Mijn opdracht bestond erin om op de bestaande backend code en kaartlezer software verder te werken. De doelstelling was om ervoor te zorgen dat men de deur of poort van een bedrijf of particuliere woning via een iOS en/of Android toestel kan openen. Bovendien moet bij een belsignaal aan een kaartlezer een notificatie verstuurd worden naar gekoppelde mobiele apparaten van de klant waartoe de kaartlezer behoort.

Mits het om toegangscontrole gaat moet alles omtrent de opdracht ook veilig gebeuren zonder dat dit de stabiliteit en snelheid van het geheel omlaag trekt.

Voor deze opdracht moet er verder gewerkt worden op een bestaand (C++) programma voor de kaartlezer die voor een toegangspunt wordt geplaatst en het online paneel waar klanten kunnen kijken wie en op welk moment iemand is binnen gekomen. Zelf moet ik een applicatie schrijven dat moet werken voor zowel iOS en Android toestellen.

2.1 USE CASES

Volgende use cases werken op het idee dat een deur kan geopend worden vanop afstand. Ook zijn er enkele use cases die berusten op de molariteit van het framework voor toekomstige actie implementaties.

2.1.1 Festival – Persoon gebouw

Tijdens een festival wordt er gebeld vanop het gebouw van het personeel. De festival organisator is momenteel op de weide en krijgt een notificatie. Ook wordt hij enkele ogenblikken later via zijn GSM op skype gebeld (Reeds geïmplementeerd). Na bevestigd te hebben dat het een personeelslid is dat zijn RFID kaart binnen heeft laten liggen kiest hij bij de notificatie om de deur te openen.

2.1.2 Bedrijf – Parking

De receptionist(e) is momenteel niet aanwezig maar er is een levering voor het bedrijf en deze dient op de parking te geraken.

De bouwheer die momenteel aan de andere kant van het gebouw is krijgt hier ook een skype call en notificatie zodat hij deze probleemloos kan doorlaten.

2.1.3 Chemisch laboratorium

Bij een laboratorium is het vaak gevaarlijk en niet toegelaten om de werkplek te verlaten zonder eerst te zorgen dat alle producten in een veilige conditie verkeren.

Wanneer er gebeld wordt is het makkelijk voor de laborant om de deur vanop zijn werkplek te openen.

2.1.4 Aanpassen configuratie

Momenteel is dit nog niet geïmplementeerd omdat dit geen doelstelling van het project was. Maar dankzij de molariteit is dit wel een optie voor de toekomst. De applicatie zou kunnen worden uitgebreid om te configureren wie toegang heeft aan de hand van regels. Regels kunnen gaan van simpele zaken zoals leeftijd tot woonplaats (zwembad).

Onderzoek

3 KAARTLEZER

De kaartlezer software is geschreven in C++ voor Windows. Het is een Windows service die een kaartlezer, relais en andere hardware aanstuurt.

Om de deur vanop afstand te kunnen openen moet de kaartlezer in staat zijn om op request van de backend de deur (relais) te openen.

3.1 DEUR OPENEN OP AFSTAND

In het huidige web platform kan men instellen om een deur te openen. Deze dient echter voor opendeurdagen en actualiseert slechts elke minuut. Om de deur live te openen moet er een nieuwe manier bedacht worden om de deur te openen.

3.1.1 Optie 1: Polling

De eerste en makkelijkste methode is polling. Bij deze methode gaat de kaartlezer zelf kijken of er iets gewijzigd is. Het is niet live omdat het op vaste tijdstippen gebeurd.

Polling is de huidige werking die geïmplementeerd was om de configuratie files binnen te halen. Dit is zeer traag momenteel met een interval van 1 minuut. Een idee zou zijn om het pollen tijdelijk te versnellen wanneer de gebruiker in gesprek is met de bouwheer.

Het pollen gebeurt momenteel op een zeer simpele manier, er wordt een volledige request van configuraties opgevraagd ongeacht of er een wijziging gebeurd is. Om dit mogelijk te verbeteren zou er een ping kunnen worden ingebouwd die een “dirty” flag meestuurt om de configuratie te updaten.

3.1.1.1 *Berekening bandbreedte eerste jaren*

x: $Aanvragen\ per\ seconde$
y: $Aantal\ kaartlezers\ eerste\ jaren$

$$x = \frac{y}{30\ seconden}$$

$$x = \frac{250}{30\ seconden} = 8{,}3\ req/sec$$

Berekening van het aantal bandbreedte per maand. Dit is het aantal bytes die gebruikt worden enkel en alleen door polling requests van de kaartlezers.

Er is een gemiddelde van 500 bytes voor de request, dit omvat een HTML request header en enkele parameters en tokens die de kaartlezer definiëren.

Voor de response is een gemiddelde van 210 bytes, deze is kleiner omdat het geen lange tokens bevat en slechts een korte response die moet nagaan of er verdere gegevens moeten worden opgevraagd.

$z: bandbreedte\ per\ maand$

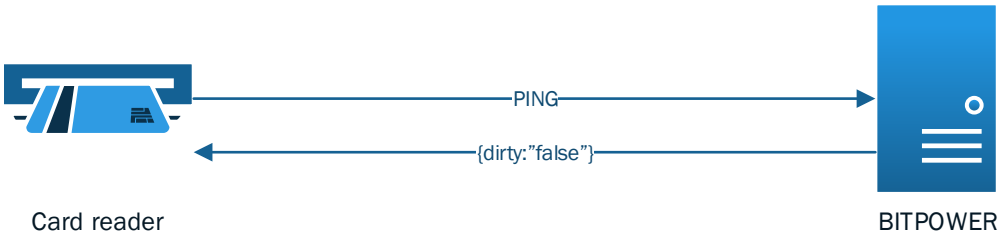
$requestBandbreedte = (x * 60 * 60 * 24 * 30) * 500\ bytes$
$requestBandbreedte = (8{,}3 * 60 * 60 * 24 * 30) * 500\ bytes$
$requestBandbreedte = 107566{,}8\ MB = 11\ GB$

$responseBandbreedte = (x * 60 * 60 * 24 * 30) * 210\ bytes$
$responseBandbreedte = (8{,}3 * 60 * 60 * 24 * 30) * 210\ bytes$
$responseBandbreedte = 4517{,}856\ MB = 4{,}5\ GB$

$z = requestBandbreedte + responseBandbreedte$
$z = 11\ GB + 4{,}5\ GB = 15{,}5\ GB/maand$

3.1.1.2 *Ping request*

De ping request stuurt een simpele ping naar de Bitpower web servers met een hash van de huidige configuratie files. Deze sturen terug of de configuratie (of tijdelijke instellingen) zijn gewijzigd in een simpele “dirty” response.



Wanneer de configuratie gewijzigd is zal de server terugsturen dat de configuratie dirty is en dus moet gewijzigd worden. Vervolgens zal de kaart lezer CPU een grotere eenmalige request sturen naar de server om de nieuwe instelling op te halen. Na het ophalen begint de normale ping request terug opnieuw.

De ping request wordt via het http-protocol verstuurd en de response is in JSON.

3.1.2 Optie 2: Open connectie

Een tweede mogelijkheid zou zijn om een open connectie te laten met de servers van BITPOWER. Echter vereist dit veel resources en is niet makkelijk met de (momenteel) huidige webservers. Dit zou echter wel een snelle manier zijn.

3.1.2.1 *IRC*

Een zeer onveilige oplossing, maar toch belangrijk om te vermelden in dit onderzoek. IRC is een makkelijk protocol om eventueel te gebruiken. Maar zoals reeds vermeld, is dit onveilig en kan dit niet op een webserver draaien.

3.1.2.2 *E-mail*

E-mail is veilig en via Exchange zou er zelf push e-mails kunnen worden ontvangen. Het grote nadeel echter is dat de e-mail geconfigureerd moet zijn op elke kaartlezer en aangezien die naar klanten gaan, is dit niet veilig.

3.1.2.3 *Websockets*

Websockets is een manier om makkelijk met socket connecties te kunnen werken. Wanneer er data wordt gestuurd zal deze een GET-header bevatten wat ervoor zorgt dat er wel meer data wordt doorgestuurd, maar het is een zeer vaak gebruikt protocol dat sinds de integratie met moderne browsers vaak gebruikt wordt.

3.1.2.4 *Google Cloud Message (GCM)*

Google Cloud Messaging zou als oplossing kunnen bieden ondanks dat de client in C++ geschreven is. Google ondersteunt normaal gezien geen clients buiten Android toestellen, dus of het een betrouwbare manier is valt nog af te wachten.

3.1.2.5 *Pushlets*

Pushlets zijn http-streaming clients die eigenlijk continue een pagina laden (ook wel DHTML genoemd). Het principe wordt vaak gebruikt bij webstreaming en wordt in PHP gebruikt met OB_FLUSH. Het enige nadeel is dat webclients zoals apache hier niet voor gemaakt zijn en dus eigenlijk veel meer resources gebruiken dan normaal. Ook zou dit betekenen dat de PHP instelling voor time-outs zou moeten genegeerd worden wat weer voor instabiliteit kan zorgen bij normale requests.

Het principe van Pushlets wordt goed uitgelegd op:
http://www.javaworld.com/article/2076063/java-web-development/pushlets--send-events-from-servlets-to-dhtml-client-browsers.html

3.1.2.6 *RMI (Remote Method Invocation) server*

RMI of Remote Method Invocation server is een server geschreven in Java, die net zoals push services als GCM or APNS een constante connectie in stand houden en enkel iets sturen als er iets moet 'invoked' worden.

Java RMI is echter niet geëncrypteerd en vereist nog verdere SSL tunneling om het echt veilig te maken. Hoewel dit goed gedocumenteerd is server-side dien ik nog steeds de implementatie in C++ te maken voor de client, wat nog ingewikkeld kan worden.

3.1.3 Gekozen methode

Mijn eerste keuze ging uit naar polling, omdat ik hiervoor verder kon werken op het bestaande polling systeem. Ik was echter niet tevreden met de responsetijd, omdat het in vaste intervallen gebeurde.

Omdat polling niet live is en ook veel bandbreedte gebruikt, heb ik gekozen voor een open connectie door middel van websockets. Deze kunnen gedebugd worden in een browser, hebben veel support voor Java en kunnen met WinHTTPD geïmplementeerd worden in de kaartlezer.

3.2 WEBSOCKET CONNECTIE

Mits er gekozen was om websockets te gebruiken, moet er wel een analyse gedaan worden naar deze 'keep alive' connectie. Enkele vragen die in mij opkwamen waren snelheid, veiligheid, implementatie en resource verbruik.

3.2.1 Implementatie

Mits websockets gebruik maken van headers om berichten onder te verdelen, ga je zelf geen protocol moeten uitvinden of implementeren om de berichten mooi onder te verdelen.

De websocket wordt geupgrade door een GET request met de server. De URL scheme voor websockets is WS of WSS voor secure socket connections.

```
GET ws://websocket.example.com/ HTTP/1.1
Origin: http://example.com
Connection: Upgrade
Host: websocket.example.com
Upgrade: websocket
```

De server zal vervolgens volgende response terugsturen om aan te tonen dat de handshake gelukt is.

```
HTTP/1.1 101 WebSocket Protocol Handshake
Date: Wed, 16 Oct 2013 10:07:34 GMT
Connection: Upgrade
Upgrade: WebSocket
```

Verdere implementatie in Java voor de websocket server was zeer makkelijk omdat hier zeer veel recente voorbeelden zijn van servers en protocollen (Agar.io, slither.io,…). Het implementeren in C++ voor de kaartlezer verliep echter niet even vlot.

Zoals reeds vermeld maakte de kaartlezer momenteel gebruik van polling om de config files te updaten. Dit alles werd gedaan met de WinInet API.

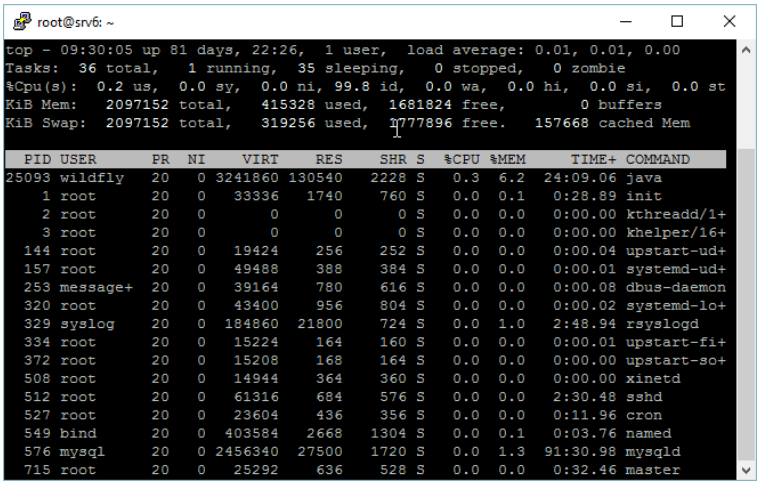
Al snel rees het probleem dat de WinInet API niet gebruikt kon worden om een websocket connectie op te zetten, de WinHTTPD API echter, die zeer hard leek op WinInet kon dit volgens de documentatie wel.

Toen is er een harmonica aan implementaties gebeurd. De WinHTTPD API kon perfect websocket connecties openen maar het zorgde ervoor dat hij problemen gaf met de reeds geïmplementeerde polling.

Na enkele weken werk om SSL werkende te krijgen en zowel het polling als de websockets goed te laten werken was de implementatie zo goed als compleet en kon ik verder met de cleanup om een makkelijk framework te voorzien om nieuwe acties toe te voegen.

3.2.2 Resourceverbruik

Een kennis (Sam Sun) die websockets gebruikte voor DRM. Het idee bij hem was om een constante connectie te hebben tussen zijn DRM-server en draaiende applicaties.



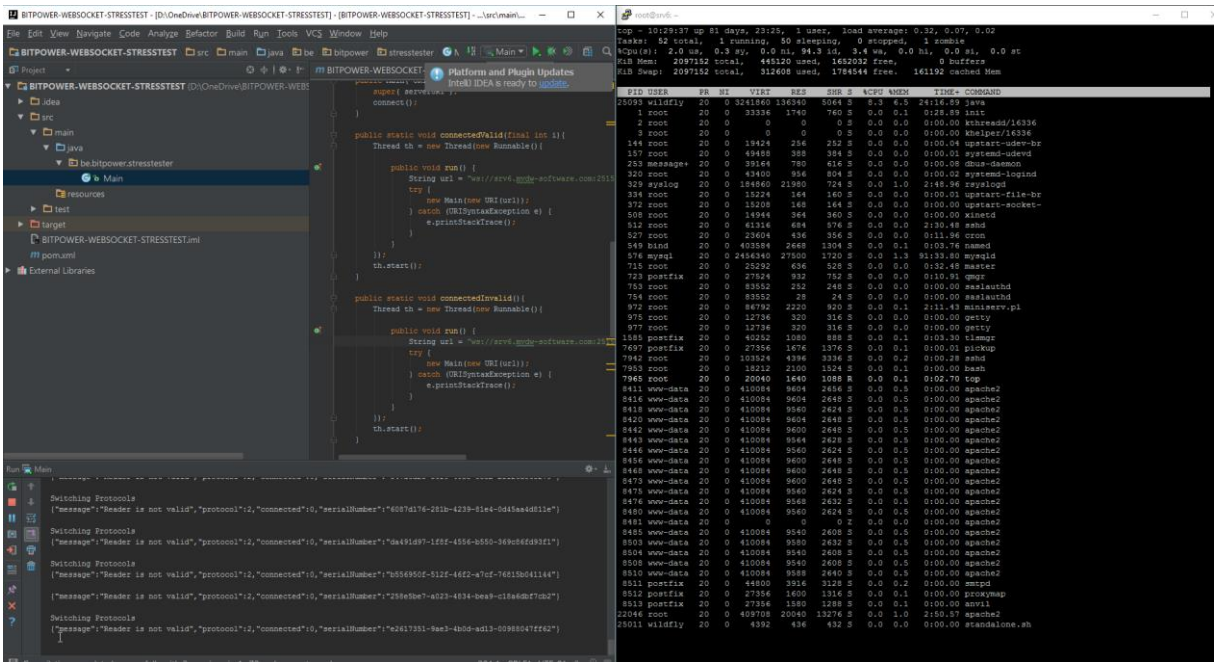
Mits hij deze DRM-server perfect op een VPS van 2GB kon draaien met ongeveer 600-900 connecties was ik er wel van overtuigd dat het resource verbruik niet zo veel ging zijn omdat de omstandigheden grotendeels gelijk waren.

Ook is het belangrijk te vermelden dat zowel bij mijn implementatie als de DRM-server van Sam Sun er niet constant berichten of data dient verstuurd te worden. Dus in principe zijn het allemaal idle connecties.

Toch heb ik volgende berekening gedaan door incrementeel meerdere connecties te maken met gebruik van een Java Agent om te kijken hoeveel memory alles gebruikt.

Volgende stress test werd gedaan op een zeer goedkope 2GB VPS van $40/jaar. Het bewijst dat zelfs met de meeste goedkope hardware het mogelijk is om dit systeem te draaien.

De stresstest omvatte het verbinden van 26 geldige readers en 50+ zelf verzonnen (niet geldige) readers.



Dit zijn de logs van het valideren van de echte en valse kaartlezers:

1461594709285#: {"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}
1461594709367#: {"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}
1461594709450#: {"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}
1461594709451#: {"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}
1461594709457#: {"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}
1461594709457#: {"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}
1461594709529#: {"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}
1461594709531#: {"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}
1461594709540#: {"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}
1461594709558#: {"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}
1461594709620#: {"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}
1461594709622#: {"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}
1461594709624#: {"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}
1461594709630#: {"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}
1461594709631#: {"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}
1461594709631#: {"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}
1461594709638#: {"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}
1461594709641#: {"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}
1461594709694#: {"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}
1461594709787#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"8fa69cc6-eea3-4562-afe3-262278f630b0"}
1461594709803#: {"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}
1461594709826#: {"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}
1461594709843#: {"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}
1461594709847#: {"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}
1461594709856#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"8f96f021-48cf-4fa3-8e51-831e4a16606b"}
1461594709862#: {"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}
1461594709866#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"6bb05978-274a-4359-a08a-48a2a85ecbf1"}
1461594709867#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"9d02fd19-56a6-4294-89c3-9701b2342736"}
1461594709871#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"82d67cac-731a-4057-a839-490594d29646"}
1461594709883#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"9bf46b7a-38ae-490f-9ddc-fb191ff8bf7e"}
1461594709921#: {"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}
1461594709936#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"157eaa37-a769-4a74-9364-c9eedb7b59fd"}
1461594709948#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"faeccc5e-28a0-46f2-8f6b-810ef01b2a4f"}
1461594709954#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"649a5241-a304-4a9b-95a5-0ca5eebb994f"}
1461594710018#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"f89cc8fd-c135-417f-b3b5-c9e4cf299b2c"}
1461594710019#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"0dc7e31f-5bce-4886-a625-d7ee10a1d9ab"}
1461594710029#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"67efb244-f3e3-4a7a-8c09-682346c3972c"}
1461594710045#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"485e9433-db4f-4dfb-8fcf-5d1fc3b003cb"}
1461594710100#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"ca631832-458a-4dc5-8428-4232ef930380"}
1461594710113#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"0c501c44-0c96-4de8-96ff-c5ac090207f3"}
1461594710116#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"71ca3cef-1750-4223-b3f2-c6a7672d3b1b"}
1461594710127#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"980cfa08-b03b-46e8-806e-4954c132bea0"}
1461594710179#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"ccb58589-0fb4-44a4-ba39-4656c0fa9929"}
1461594710200#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"b3c546ac-c9cf-4cab-bc68-dc92fad1a8d2"}
1461594710201#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"544f90ad-8c1d-4b31-ab7f-508963474dfd"}
1461594710208#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"c941f9e0-abb0-4d00-a280-e44c34b43c5b"}
1461594710250#: {"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}
1461594710260#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"80da82aa-ef5f-4a48-aead-936164920b09"}
1461594710280#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"568477f0-8a64-4c2e-b963-5cd51826add2"}
1461594710281#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"6194bf65-5b42-49cc-8838-e25634584ff4"}
1461594710290#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"52891819-f8c6-47aa-ba67-97908cab3213"}
1461594710341#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"71b768c9-216b-4751-92c4-155c68bac507"}
1461594710364#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"87de214a-cbd7-41e7-9916-ba71d56ef7e2"}
1461594710371#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"003b2222-1d27-4ba2-a16b-ab00b150a15b"}
1461594710372#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"ef2b5215-9696-4133-99c7-d69fe36d268f"}
1461594710442#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"c5df8a83-13d3-4c98-a3c3-f33f0461facc"}
1461594710444#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"384bbcf9-7025-42b7-8e8e-4bdbe2592722"}
1461594710446#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"781c7dc6-1a6a-44a0-9e3b-dfd9b226e654"}
1461594710527#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"d3f96c32-b7aa-4a7d-8a6d-364ce2a2de73"}
1461594710528#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"8aa31cee-040b-46fd-9ba5-5e21b07275cd"}
1461594710534#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"517fc039-ab24-4129-9034-243a6ff64e4a"}
1461594710535#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"47a6c6a5-8d51-4d0b-9369-8b858b77e3b5"}
1461594710541#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"5d862b97-85c6-4277-9b01-372bd4646926"}
1461594710607#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"8cb328c8-be46-4755-a59e-64fe7ab28ade"}
1461594710613#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"fe7478f5-73a8-478d-8d71-a4a888cb8434"}
1461594710616#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"94bab7d6-f06c-430f-9580-7dbb6bca60c7"}
1461594710621#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"aee4d2a4-9c38-44a5-b2f8-5e7225897b1f"}
1461594710687#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"8abf11db-30d6-4c78-9197-eea44c6f6598"}
1461594711141#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"45b3bd35-e820-44ef-982d-0de2c5e0759f"}
1461594711142#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"4fbaa1df-4e2d-4d51-9d41-31b0987b29b9"}
1461594711226#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"4eb52195-0ce5-4997-b9aa-2ab30f13dd72"}
1461594711229#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"d48cbc38-6c74-433f-b643-401eb4416a0a"}
1461594711231#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"1ea81272-c12a-4f84-9af2-2da6714d1e49"}
1461594711232#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"7f1797ac-e524-4f62-bcca-7cc4da3fad16"}
1461594712077#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"e26f95ee-1659-450d-87e0-5b7aa58da76c"}
1461594712791#: {"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"6e8fb519-9a64-4998-bf4b-9f82aae6a7f7"}

Na het abrupt afsluiten van de connecties was er nog steeds een kleine load, deze daalde echter van zodra er time-outs optreden op de hartslagen.

Het duurt 3,5 seconden op de test server om 76 readers te connecteren en valideren.

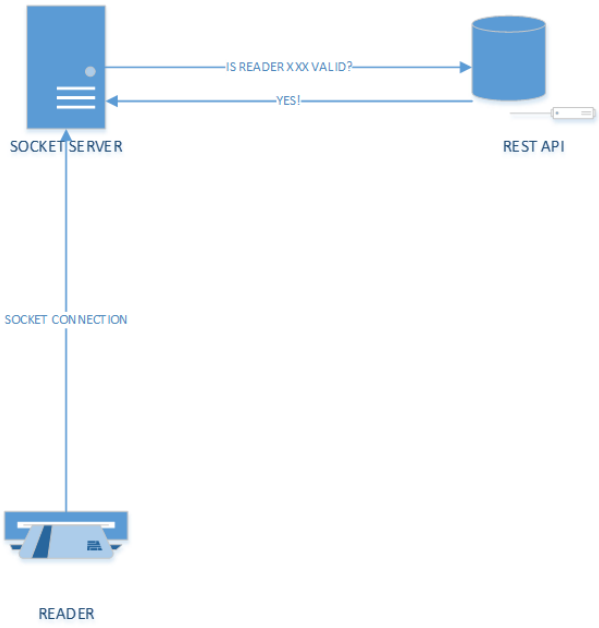
3.2.3 Veiligheid

Websockets kunnen bestaan uit een plain connectie (WS) of een Secure connectie (WSS). Server side heb ik gekozen om twee servers te laten draaien, een voor WS en een andere voor WSS. Dit om het makkelijk te maken voor te debuggen en om aan troubleshooting te doen in de toekomst.

Clientside (kaartlezer) word altijd WSS gekozen.

Stap 1) De kaartlezer zal een secure websocket connectie maken met de socket server.

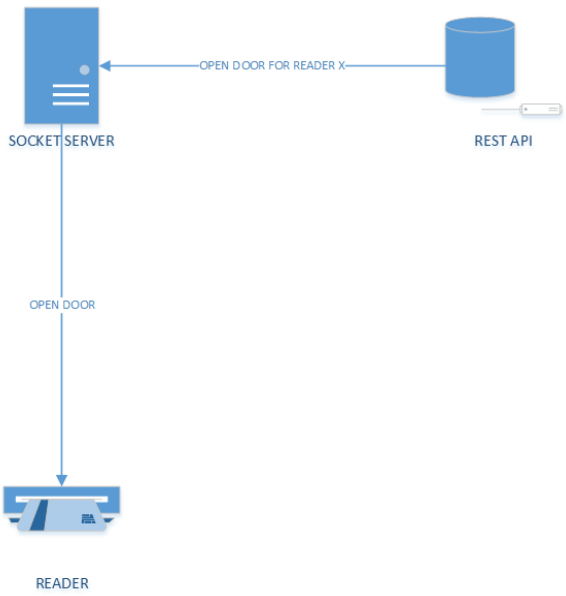
Bij het connecteren wordt het serienummer en de private key gebruikt.



Stap 2) De websocket connectie zal worden goedgekeurd maar achterliggend zullen het serienummer en private key worden gevalideerd met de REST API server.

Indien dit negatief is dan zal de websocket connectie worden verbroken, indien dit positief is zal deze mee opgenomen worden in geconnecteerde kaartlezers.

De private key wordt bijgehouden voor extra verificatie bij stap 3.



Stap 3) Wanneer de applicatie een request maakt met de REST API om de deur te openen (of een andere actie uit te voeren) zal de REST API deze actie doorsturen naar een servlet op de socket server.

De socket server kan deze vervolgens live doorsturen via het websocket protocol.

Hier zijn twee veiligheidsmechanismes voorzien om ervoor te zorgen dat deze servlet veilig kan gebruikt worden:

- De socket server accepteert enkel acties van ingestelde IP-adressen
- De rest api zal zowel het serienummer als de private key doorsturen die nogmaals geverifieerd worden met de geconnecteerde kaartlezers.

Dit in combinatie met een SSL-certificaat voor de servlet maakt het haast onmogelijk om een MITM-attack uit te voeren om een valse request naar deze servlet te sturen.

3.2.4 Hearthbeat

Iets waar ik niet direct problemen mee ondervond was de 'keep alive' van de websockets. Zelfs als ik de kaartlezer abrupt stopte werd de connectie goed verbroken op de server. Het was pas na een stroomuitval dat ik merkte dat de server nog steeds dacht dat deze verbonden was.

Daarom heb ik een hartslag ingebouwd in de client en server om ervoor te zorgen dat deze om de bepaalde tijd iets moet sturen.

Wanneer een kaartlezer een geruime tijd niets van zich laat horen zal de server er van uitgaan dat de kaartlezer inactief is.

Indien dit niet zou worden geïmplementeerd dan zou er een memory leak ontstaan van connecties die open blijven staan. Ook zou het een valse weergave geven in de applicatie voor de online/offline status.

De kaartlezer zelf gaat bij connectieproblemen steeds zelf opnieuw verbinding proberen maken. Indien de hartslag nog niet vervallen is binnen de periode waar er een nieuwe connectie wordt gemaakt, dan zal de vorige connectie vroegtijdig verwijderd worden.

3.2.5 Gegevens doorsturen

Alle gegevens worden doorgestuurd in JSON. Dit is geen vereiste van websockets, maar het was een makkelijke manier om gegevens door te sturen en te lezen in zowel C++ (met JSONCPP) en de java websocket.

3.2.5.1 *Voorbeeld connectie succes*

{"message":"Connected to EIDIO Server","protocol":2,"connected":1,"serialNumber":"288K001140819040"}

Volgende status wordt doorgestuurd wanneer een connective gevalideerd is door de REST API en positief is.

- **Message:** Bericht van de server
- **Protocol:** Protocol versie van de server
- **Connected:** Als de connectie gelukt is dan zal dit 1 zijn
- **SerialNumber:** Serienummer van de kaartlezer

3.2.5.2 *Voorbeeld connectie fail*

{"message":"Reader is not valid","protocol":2,"connected":0,"serialNumber":"ccb58589-0fb4-44a4-ba39-4656c0fa9929"}

Volgende status wordt doorgestuurd wanneer een connectie gevalideerd is door de REST API en negatief is. Na het versturen zal de connectie gesloten worden door de server.

- **Message:** Bericht van de server
- **Protocol:** Protocol versie van de server
- **Connected:** Als de connectie mislukt is zal dit 0 zijn
- **SerialNumber:** Het serienummer dat niet kon worden gevalideerd

3.2.5.3 *Voorbeeld actie naar kaartlezer*

{action:"open",reader:" 288K001140819040",duration:5000}

Dit is een voorbeeld van een actie die van de server naar de kaartlezer wordt gestuurd. Omgekeerd kunnen er geen acties of "vragen" worden verstuurd naar de server.

Optioneel kan de kaartlezer antwoorden op een actie met een vrije JSON response die alsook de actie bevat.

Een actie wordt steeds doorgestuurd met:

- **Action:** De actie naam
- **Reader:** Het SN van de kaartlezer
- **<extra>:** extra gegevens specifiek voor de actie (zoals in bovenstaand voorbeeld de duratie)

Het is de taak van de server om bij te houden welke acties er uitgevoerd moeten worden op de kaartlezer, zodat deze zelf niet moet pollen.

Voorbeelden zijn:

- Software update beschikbaar
- Aanpassing configuratie
- …

3.2.6 Configuratie

Momenteel worden de gegevens van de websocket server hardcoded in de applicatie.

Het is aangeraden om deze server side te laten doorsturen bij de config. Zo kan men in de toekomst wanneer deze kaartlezers op andere plaatsen in de wereld worden geïnstalleerd deze dynamisch laten toekennen om de load te balanceren.

3.3 FYSIEK DE DEUR OPENEN

Volgende opties waren het meest voor de hand liggend om een deur fysiek te openen nadat de bouwheer deze op afstand opent. Hier moet rekening gehouden worden met enkele scenario's en veiligheid.

3.3.1 Optie 1: Elke eID tijdelijk toelaten

Bij deze optie was de oplossing om tijdelijk elke eID toe te laten zodat er nog steeds gelogd kon worden wie er binnen komt. Echter gaf dit het probleem dat wanneer je juist belt omdat je je EID vergeten bent, je niet veel verder staat.

3.3.2 Optie 2: De deur openen tot iemand binnenkomt

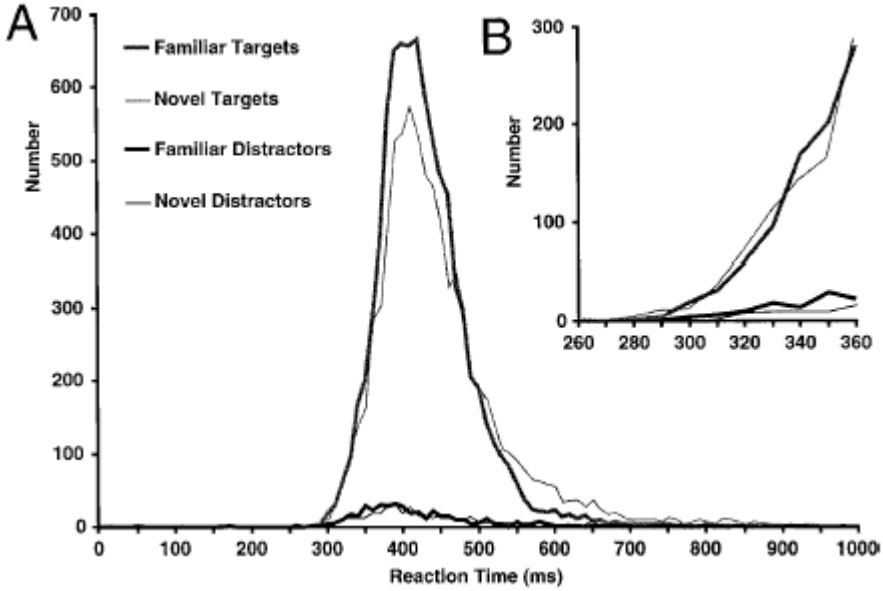
Omdat bij de vorige optie steeds een eID kaart nodig was, heb ik voor dit idee gekozen om de deur open te laten tot iemand binnengekomen is. Dit betekende echter wel dat je dus 's avonds kon terugkomen met een geopende deur. Dit was dus al snel een slecht idee.

3.3.3 Optie 3: De deur openen zoals bij een eID authenticatie

Deze optie heb ik lang aangehouden. Het idee was om de deur/poort X-aantal milliseconden open te houden net zoals dat het zou gebeuren wanneer je je eID authentiseert.

3.3.4 Optie 4: De deur met een extra reactietijd openen

Deze optie is het verlengde van optie 3, hierbij wordt er 750ms reactietijd toegevoegd omdat je niet weet "wanneer" de deur gaat opengaan. Anders dan dat je zelf de controle hebt bij het inscannen van je eID weet je niet wanneer de bouwheer op de knop gaat drukken om de deur te openen.



Grafiek is afkomstig van: http://sccn.ucsd.edu/~arno/jsindexeeg.html

3.3.5 Gekozen implementatie

Op het eerste zich lijkt dit een nutteloos onderzoek, omdat het slechts een klein stukje is van het geheel, maar dankzij dit onderzoek heb ik besloten om een zeer modulair framework te maken dat toelaat om meerdere acties in de toekomst toe te voegen. Momenteel is de geïmplementeerde actie om de deur zoals in een appartementsblok voor een bepaald aantal seconden open te doen, maar dankzij het modulair framework kan dit zeer makkelijk gewijzigd worden.

3.4 RETROSPECT KAARTLEZER

Bitpower heeft mij meerdere keren gevraagd naar de stabiliteit en uitbreidbaarheid van de kaartlezers die geschreven zijn in C++. Dit is een retrospect na het einde van mijn final work met zaken die ik tussendoor heb onderzocht om te kijken of deze mogelijk waren.

3.4.1 Stabiliteit

De kaartlezer is stabiel maar mist op sommige plaatsen foutafhandeling. Omdat C++ zeer lastig kan doen met handlers denk ik dat het zeer snel voor stabiliteitsproblemen zou zorgen als er meerdere kaartlezers gaan worden aangesloten.

3.4.2 Java

Ik heb tijdens een dieptepunt van mijn final work (Januari) geprobeerd of het mogelijk was om de code te porten naar Java. Dit bleek haast onmogelijk doordat de C++ dll via pointers en handlers moet aangesproken worden (niet abstract) en deze ook nergens online gedocumenteerd was.

Voor andere zaken zoals API update polling en websockets was Java perfect.

Java zou een perfecte keuze zijn mits de opdrachtgever ook al heeft laten weten dat er een desktop variant zou worden gemaakt in de toekomst voor PC's aan een receptie. De desktopapplicatie zou enkel voor Windows werken.

3.4.3 C#

Naast het proberen porten naar Java heb ik dit ook geprobeerd in C#, omdat het aanspreken van C++ dll's in C# iets makkelijker gaat.

Echter kwam ik hier met dezelfde problemen dan in Java. Zelfs toen het mogelijk was om de dll file aan te spreken bleef de library veel te abstract om stabiel aan error handling te doen.

Net zoals Java zijn de niet kaartlezer gerichte acties makkelijk te implementeren.

3.4.4 Embedded C

Dit onderdeel heb ik wegens tijdsgebrek niet verder onderzocht of uitgeprobeerd. Ideaal zou zijn om de computer volledig weg te laten (kostprijs omlaag, robuustheid en betrouwbaarheid omhoog) en de volledige besturing via een micro controller te laten gebeuren.

Een micro controller zoals een ATMEL 2560 heeft genoeg geheugen en processing power om een netwerkverbinding tot stand te brengen en de kaartlezer aan te spreken.

Voor een product dat als doelstelling heeft om bij 'Internet Of Things' te behoren is het in mijn ogen vrij omslachtig om aan elk systeem een volledige Windows computer te hangen.

Het programmeren is C op een microcontroller zal met enkele programmeurs een tijdje duren om alles stabiel te krijgen. Maar zodra het product af is blijft het stabiel omdat er geen andere factoren zijn zoals Windows updates of drivers die de stabiliteit in gedrang kunnen brengen. Uiteindelijk zal de kostprijs even duur zijn als een Raspberry Pi.

Het hacken van het systeem zou op software niveau veel moeilijker zijn.

3.4.5 Bron van het probleem

Na het bekijken van de code en het proberen porten naar Java en C# ben ik tot het besluit gekomen dat de bron van het mogelijk stabiliteitsprobleem bij de kaartlezer zit. De enige library die beschikbaar is gesteld door de leverancier is een redelijk slecht gedocumenteerde C++ lib. Hierdoor is het zeer moeilijk om deze naar een andere taal zoals C# of Java te porten zonder zeer 'lelijke' code te schrijven die de C++ dll kan aanspreken.

Het protocol van de kaartlezer reverse engineeren en zelf de library schrijven is een optie maar is zeer tijdrovend en is geen modulaire oplossing.

Mijn advies aan Bitpower is om toekomstgericht te zoeken naar een kaartlezer waarvan er SDK's worden voorzien die zowel werken in C++ als in C#.

Als de kaartlezer eventueel ook in Java zou werken zou dit er tevens voor kunnen zorgen dat de kostprijs honderden euro's goedkoper kan door het gebruik van goedkope (en stabiele) linux CPU-boards zoals een Raspberry Pi of een meer industriële variant.

> *Zelf heb ik enkele embedded systemen met Raspberry Pi's die al met twee jaar uptime aan data logging doe. Ze zijn krachtig en verbruiken zo weinig dat je al snel een kleine Lithium batterij als UPS kan gebruiken.*

Een door Fedict erkende kaartlezer die compatibel is met EID-middleware drivers is de beste optie omdat de hardware van de kaartlezer geen invloed heeft op de implementatie in de code.

Zoals eerder in deze retrospect vermeld is het ooit de bedoeling van Bitpower om kaartlezers te voorzien bij een receptie. Wanneer men de EID-middleware kan gebruiken moet men niet steeds dezelfde 'heavy-duty' kaartlezer voorzien.

Voorbeeld: http://www.connecto.com/nl/eid-kaartlezer-sis-cardreader-fedict/pd/964A49BN019805/993913N8004276

4 BACKEND

De backend bestond aan het begin van mijn final work uit een web paneel dat al volledig functioneel was. Op dit web paneel kunnen klanten en installateurs kaartlezers en gebouwen beheren.

Wijzigingen die ik aan de backend moest aanbrengen was het toevoegen van een API die ervoor zorgt dat de applicatie op een veilige manier gegevens kan opvragen en zich kan authentiseren.

De backend is geschreven in PHP dus dit zal ook de taal zijn die gebruikt gaat worden om de REST API te implementeren.

4.1 VEILIGHEID

Veiligheid is voor dit project zeer belangrijk. De API moet in staat zijn om deuren of poorten te openen en dit is niet iets dat zomaar omzeild mag worden.

De backend heeft reeds een authenticatie voor klanten voorzien, enkel is deze niet gemaakt om extern te benaderen.

Toen ik aan het project begon en verder begon te werken op de reeds bestaande code merkte ik dat hier nog niet veel aandacht aan was besteed. De gevonden verbeterpunten zijn doorgegeven aan de opdrachtgever om deze te wijzigingen. Zelf ben ik met een schone lei begonnen en heb classes gemaakt in de PHP backend om veilige communicatie met de database te voorzien.

Deze classes kunnen later eventueel door Bitpower gebruikt worden om de rest van de UI backend hiervan te laten gebruiken.

4.1.1 Optie 1: Basic TLS-encryptie

Het makkelijkste om de REST API te beveiligen is via TLS-encryptie waarbij de gebruikersnaam en wachtwoord worden doorgestuurd in een reversibel encoding naar een TLS-server. Indien ik voor deze methode zou gekozen hebben, zou ik verder onderzoeken of ik een token zou generen om de volgende requests te authentiseren.

Bij TLS-encryptie worden gegevens en data verstuurd via een veilige tunnel verbinding. Niemand kan deze gegevens intercepteren.

4.1.2 Optie 2: SHA1 TLS-encryptie

De betere versie van Basic TLS-encryptie is om het wachtwoord rechtstreeks in SHA1 door te sturen. SHA1 is de gebruikte encryptie in de backend waardoor dit een veilige manier is om het wachtwoord door te sturen.

Deze SHA1 TLS-encryptie is iets veilig dan de TLS-encryptie, alles gebeurd nog steeds via een SSL-tunnel, maar het wachtwoord is daarbinnen nog geëncrypteerd via SHA1 en word dus niet in leesbare tekst doorgestuurd.

4.1.3 Optie 3: OAuth1

OAuth1 is een volledig ander protocol dan OAuth2 en is een handtekening gebaseerd protocol. Er zijn veel libraties in PHP en Javascript maar deze worden niet meer zo veel up-to-date gehouden.

Het principe van OAuth of Open Authenticatie is dat je 'tokens' krijgt wanneer je een gebruiker laat authentiseren op de server van het bedrijf dat de authenticatie aanbiedt. Met dat token kan de applicatie de communicatie verderzetten.

4.1.4 Optie 4: OAuth2

OAuth2 is een authenticatie protocol ontwikkeld in 2006. Omdat de focus ligt om het zo makkelijk mogelijk te maken voor clients is het ideaal om te gebruiken in de hybride applicatie.

Echter na verder onderzoek over andere mogelijkheden en toepassingen ben ik tot de conclusie gekomen dat er betere mogelijkheden zijn voor een toepassing als dit.

Voor de client kan volgende javascript library gebruikt worden:
https://github.com/andreassolberg/jso

Voor de server kan volgende PHP library gebruikt worden:
https://github.com/bshaffer/oauth2-server-php

4.1.5 Gekozen encryptie

Hoewel OAuth en OAuth2 veilig zijn is het niet nodig om deze te gebruiken mits we geen publieke application interface willen maken.

Daarom zal voor zowel SHA1 TLS-encryptie en Basic TLS-encryptie gekozen worden (voor de applicatie die ik maak SHA1 TLS-encryptie).

Mocht er ooit gekozen worden om 3rd party applicaties toegang te geven tot een grotere API (bijvoorbeeld om gegevens uit te lezen …) kan dit erbij worden genomen.

Dit soort authenticatie in combinatie met OAuth wordt op veel API's zoals MobileVikings, … gebruikt.

Wel stel ik voor dat indien ooit gekozen wordt om naar een Java backend API over te schakelen (en andere developers de API moeten gebruiken) OAuth wordt gebruikt. In java is dit in letterlijk 3-tellen gemaakt en het zorgt ervoor dat de API makkelijk te limiteren is voor de doeleinden.

4.1.5.1 *Protocol verloop – SHA1 TLS-encryptie*

- De gebruiker vult zijn gebruikersgegevens in
- De client zal het wachtwoord naar SHA1 omzetten
- De client verstuurt het versleutelde wachtwoord over TLS naar de server
- De server authentiseert de gebruiker bij elke request die gemaakt wordt.
- Als de authenticatie gelukt is, wordt de <username:<sha1password> naar base64 omgezet en bewaard op de GSM. Naar gelang secure storage wordt deze met AES geëncrypteerd (zie hoofdstuk: mobiele applicatie).
- Bij volgende requests wordt er terug met deze credentials ingelogd om acties uit te voeren
    - Achterliggend zal de backend verifiëren dat het gebouw of kaartlezer dat aangesproken wordt wel degelijk van de ingelogde gebruiker is.

4.2 LINK MET APPLICATIE

De applicatie moet getriggerd worden wanneer er op een bel wordt gedrukt. Om dit te doen moet er met push notifications gewerkt worden.

4.2.1 Verloop

- De gebruiker logt in op de applicatie met zijn klantengegevens
- De backend houd bij welk device geauthentiseerd is (max 2)
- Wanneer er op een bel gedrukt wordt zal de kaartlezer een request sturen naar de backend
- De backend zal naar de geregistreerde mobiele toestellen een request sturen voor re-authenticatie
- De applicatie zal tonen dat er een persoon op de bel heeft gedrukt (met meer informatie waar deze bel zich bevind)
- In de achtergrond zal de applicatie terug een authenticatie uitvoeren

4.2.2 Device push request

Aangezien we met een hybride applicatie werken en bestaande plug-ins gaan gebruiken moeten we onderzoek doen.

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

Phonegap-push-plugin is de meest bekende phonegap plugin. Het is duidelijk gedocumenteerd en heeft support voor zowel GCM (android) en APNS (iPhone). De plugin voorziet ook documentatie voor eigen aangepaste payloads om meer informatie met de requests door te sturen.

4.2.2.1 *Google Cloud Messaging (GCM)*

GCM werkt met een registratiesysteem. Om GCM te kunnen gebruiken dien je de device ID bij te houden. Met deze device ID kan je vervolgens rechtstreeks naar deze client een push bericht sturen naar de applicatie.

Deze technologie is vooral gemaakt voor Android en er zijn plug-ins voorzien voor Cordova.

4.2.2.2 *Apple Push Notification Service (APNS)*

APNS is ongeveer hetzelfde als GCM maar dan de Apple variant. Het grote verschil echter is dat de backend op je eigen server kan staan.

De APNS-PHP libary kan hier worden gedownload.
https://code.google.com/p/apns-php/

4.2.2.3 *Windows Notification Service (WNS)*

WNS is de Windows variant voor zowel GCM als APNS echter is deze zeer simpel. Ondanks het geen vereiste is om Windows Phone te supporten heb ik hier toch even onderzoek naar gedaan om na te gaan of eventuele support in de toekomst mogelijk en/of moeilijk is.

Momenteel is de push plugin die ik gebruik niet 100% gemaakt voor windows phones, waardoor het eigenlijk niet mogelijk om dit te testen.

Ik geloof er wel in dat de compatibiliteit voor Windows Phones op komst is, mits er al enkele branches zijn waar er testen naar WNS worden gedaan.

4.2.2.4 *Firebase Cloud Messaging (FCM)*

FCM werd eind Mei 2016 aangekondigd bij Google I/O. Omdat dit op het einde van mijn final work was, heb ik dit niet meer kunnen implementeren, maar ik heb er wel wat onderzoek naar gedaan om te weten te komen wat het juist is en of het inderdaad wel beter is.

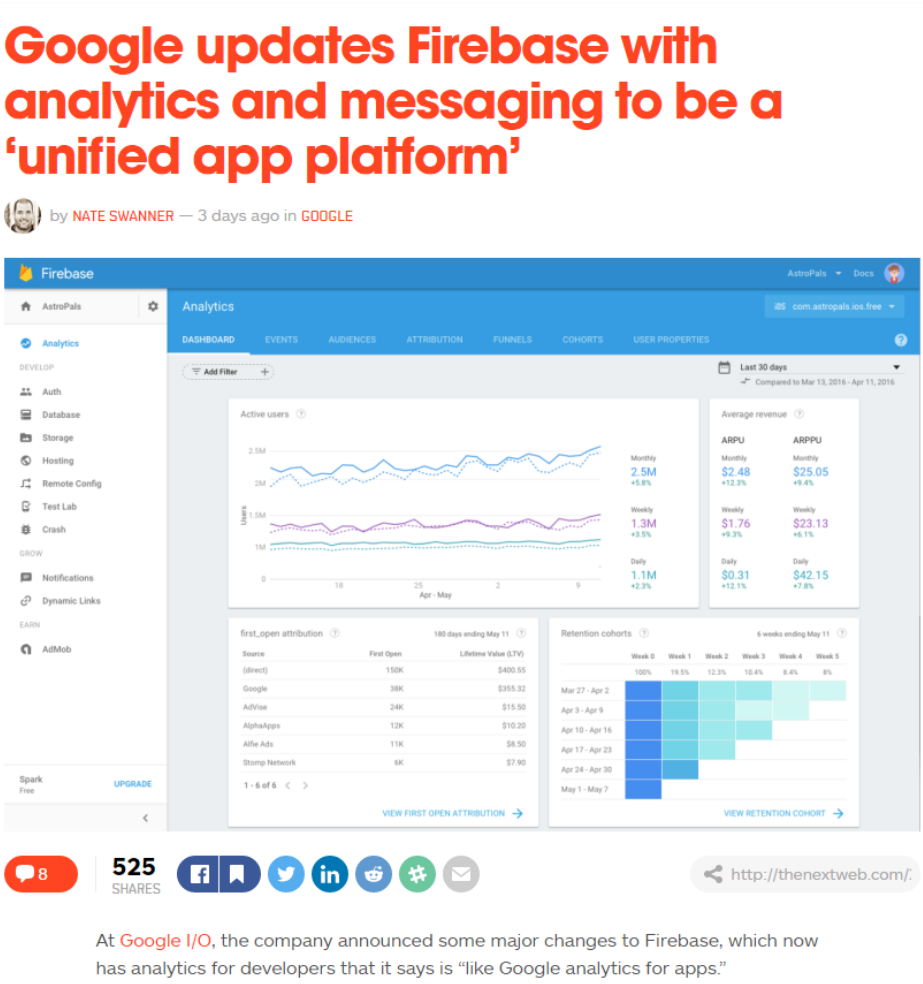
https://developers.google.com/cloud-messaging/faq

Google raadt aan om voor nieuwe applicaties gebruik te maken van FCM ipv. GCM omwille van volgende features:

- Het is cross platform (Web, iOS, Android)
- Data payloads van 4KB en notification payloads van 2KB
- Het werkt verder op de architectuur van GCM
- Server side dient er niets gewijzigd te worden (optimalisaties in de toekomst zijn wel mogelijk)
- Registratie handeling gebeurt automatisch
- **Upstream messaging !!!**

Over dat laatste wil ik nog even uitbreiden. Android N is vooral chat gebaseerd en met FCM hebben ze hier ook aan gedacht. Upstream messaging laat toe om reacties of data te sturen naar de server via het FCM-protocol.

Dit zorgt ervoor dat in de toekomst op een nog snellere een batterij efficiënte manier acties kunnen worden gestuurd naar de server bij een notificatie.



At Google I/O, the company announced some major changes to Firebase, which now has analytics for developers that it says is "like Google analytics for apps."

4.2.2.5 *Implementatie*

De implementatie met Phonegap is sinds Mei 2015 makkelijker dan voordien. Met de nieuwe phonegap-push-plugin (https://github.com/phonegap/phonegap-plugin-push) kan je push berichten sturen met een eigen payload via één API voor alle platformen.

Na onderzoek bleek dat GCM zijn calls voor iOS devices kan doorsturen naar APS en je zelf dus niet de SSL-socket connectie vanuit PHP naar de APS servers moet gebruiken. Je dient hier nog wel steeds de certificaten te genereren.

APNS-Cert.p12	14-Feb-16 16:53	Personal Information Exchange	4 KB
aps.cer	14-Feb-16 18:35	Security Certificate	2 KB
APS.p12	14-Feb-16 18:37	Personal Information Exchange	4 KB
aps_development.cer	08-Feb-16 18:06	Security Certificate	2 KB
APS-DEVELOPMENT.p12	14-Feb-16 16:53	Personal Information Exchange	4 KB
BITPOWER-EIDIO.p12	08-Feb-16 14:18	Personal Information Exchange	2 KB
CertificateSigningRequest.certSigningRe...	08-Feb-16 14:17	CERTSIGNINGREQUEST File	1 KB
EIDIOCert.pem	08-Feb-16 19:10	PEM File	2 KB
EIDIOKey.pem	08-Feb-16 19:10	PEM File	2 KB

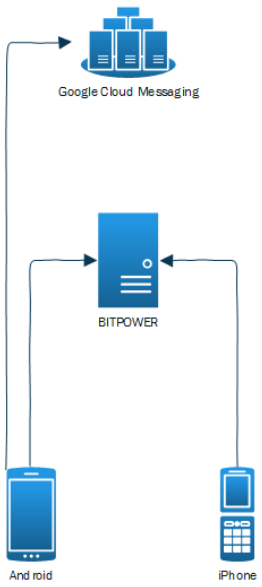
4.3 Push Request

Om het mobiele apparaat te triggeren wanneer er gebeld wordt aan een deur moeten we aan push requests sturen. Deze worden meestal gebruikt voor notificaties en dergelijke, maar er kan een eigen extra payload worden meegestuurd met de push request die dan vervolgens het gebouw en kaartlezer bevat van waar gebeld wordt.

4.3.1 Toestel registratie

Alvorens push requests kunnen worden uitgestuurd moet het mobiel toestel geregistreerd worden. Om dit te doen wordt na het inloggen een request gestuurd die het unieke device ID naar de Bitpower-servers stuurt. Hier worden deze bijgehouden en gelinkt aan een gebruiker.

Er mogen maar een bepaald aantal toestellen geregistreerd worden op een gebruiker. Registratie gebeurt met een FIFO (first in first out) methode.

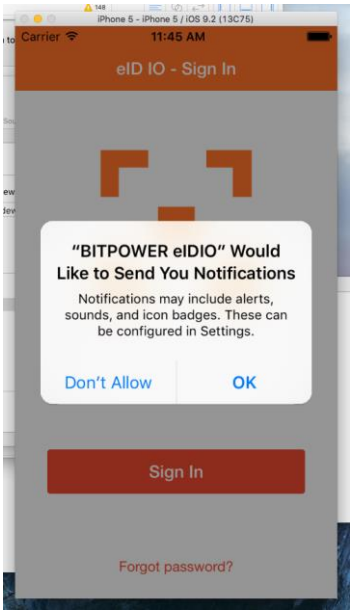


Device Registration

4.3.1.1 *Android (GCM)*

Voor Android zal een request gestuurd worden naar de GCM servers die een uniek device nummer terug geven. Deze UDID wordt vervolgens naar de REST API gestuurd om daar opgeslagen te worden in de database.

Android vereist geen specifieke permissies om een notificatie te ontvangen.



4.3.1.2 *iOS (APNS)*

Voor iOS is het ongeveer hetzelfde als bij Android, echter vereist er meer verificatie tussen applicatie en de APNS-servers d.m.v. op voorhand geregistreerde certificaten.

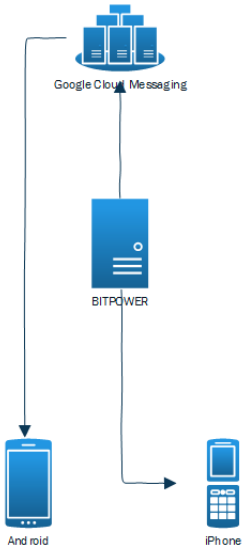
De eerste keer dat de gebruiker de applicatie start op een iOS device zal er een pop-up verschijnen met de melding dat de applicatie notificaties wil sturen. Indien de gebruiker dit niet zou accepteren zal dit betekenen dat deze geen melding krijgt indien er iemand belt.

Omdat het maken van een aparte APNS-implementatie veel werk was en eigenlijk sinds Dec. 2015 niet meer nodig heb ik gekozen om via de push plugin dit via GCM te laten gebeuren. Achterliggend wordt er wel nog steeds met APNS-notificaties gestuurd, maar google zorgt hiervoor.

De keuze om GCM te gebruiken was om het wat makkelijker voor mij te maken om de backend te implementeren, omdat ik dan dezelfde code als GCM kon gebruiken. De notificaties werken zeer goed met deze methode, het enige nadeel is dat de push plugin die ik gebruik geen "knoppen" voorziet om vanuit de notificatie de deur te openen of gesloten te houden.

4.3.2 Toestel notificatie

Zodra er gebeld wordt aan een kaartlezer zullen alle geregistreerde toestellen een notificatie krijgen. Verdere acties zoals het openen van de deur vereisen een geauthentiseerde request.



4.3.2.1 *Android (GCM)*

Voor Android zal de REST API een GCM request sturen met het device ID. Dit gebeurt door middel van een simpele cURL request die verstuurd wordt vanuit een PHP script.

De cURL request bevat de private key die men moet genereren bij het maken van een Google App op http://console.developers.google.com/ . Hier kan men ook als extra veiligheid instellen om enkel requests te aanvaarden die afkomstig zijn van een bepaald IP-adres.

4.3.2.2 *iOS (APNS proxied door GCM)*

Voor iOS is het ongeveer hetzelfde als bij Android, echter vereist het net zoals de registratie een certificaat (private) om de server te authentiseren. Zoals eerder al vermeld gebeurt dit alles via GCM, zodat er geen apart framework moet geïmplementeerd worden server side.

4.3.3 Kaartlezer trigger

De kaartlezer is het component waar alles start wanneer er om toegang wordt gevraagd. Deze zal een request sturen naar de webserver met dezelfde securitygegevens die bij een normale request worden gebruikt zijnde:

- Kaartlezer Serie Nummer
- Private key (opgevraagd bij installatie)

Nadat de kaartlezer is geauthentiseerd zullen de geregistreerde toestellen een notificatie krijgen. De lay-out van de notificatie is zo gekozen dat deze toelaat om snel een overzicht te krijgen van waar er toegang wordt gevraagd en ook om direct een actie te voorzien om de deur te openen of gesloten te laten.

4.4 RETROSPECT BACKEND API

Tijdens dit academiejaar heb ik veel moeten aanpassen in de backend, nu op het einde van mijn project zijn er zaken die ik zou voorstellen aan de opdrachtgever om in de toekomst te wijzigen.

De backend is in grote mate stabiel en dient niet al te veel aanpassingen te krijgen. Wel is het aangeraden om de REST API en de Websocket server (3.2) op eenzelfde server te zetten zodat de communicatie tussen deze twee services kan weggelaten worden om voor een nog kleinere latency te zorgen.

4.4.1 Java backend

Mits de problemen in het begin met de veiligheid en de uitbreiding van een websocket Java backend denk ik dat het aangeraden is om te bekijken of de API backend (en ook de front-end) niet best volledig in Java kan worden geschreven.

De kracht van een backend in Java ben ik vooral te weten gekomen tijdens mijn stage (Februari 2016) en prive projecten die ik onder het jaar heb gedaan.

4.4.1.1 Voordelen

- Makkelijker uit te breiden
- REST API developers werken vaker met Java dan in PHP
- Libraries zijn makkelijker te gebruiken voor bijvoorbeeld GCM, APNS en zelfs WNS
- Makkelijker te schalen in de toekomst naar AWS (Amazon Web Services)
- Een API maken in Java is makkelijker dan in PHP
- De Java backend die nu gebruikt wordt voor de websockets kan geïntegreerd worden in de nieuwe backend. Dit zorgt voor minder latency.

4.4.1.2 Nadelen

- Het is een grote aanpassing voor de front-end. Het beste zou zijn om SpringMVC te gebruiken.
- Meer resourceverbruik in het begin, komt gelijk tot zelfs beter naarmate de sessies stijgen

4.4.2 Stabiliteit REST API servers

Tijdens het testen heb ik vaak gemerkt dat mijn connecties time-outs kregen. Na verder onderzoek op meerdere toestellen ben ik tot de conclusie gekomen dat mogelijk een proxy (DDoS Mitigatie) requests van bepaalde user-agents/IP's begint te weigeren.

Bitpower heeft laten weten dat momenteel de API-site op een shared host staat zolang de applicatie nog niet in productie is. Deze zal in productie naar België worden gemigreerd.

Het is belangrijk om zeker te zijn dat de servers de load aankunnen en de kaartlezers of applicatie niet tegen gaan houden met DDoS Mitigatie.

5 MOBIELE APPLICATIE

De applicatie was uiteindelijk het stuk dat de klanten van Bitpower gingen zien. Er waren enkele vereisten aan verbonden:

- Android ondersteuning
- iOS ondersteuning
- Snel een notificatie bij bellen
- Snelle responsetijd bij acties

De meeste van deze zaken waren al snel opgelost met de analyse van de backend en websocket server.

Uit ervaring wist ik dat notificaties het grote struikelblok ging zijn. De Push plugin die ik voordien vaak gebruikte was zeer moeilijk te implementeren en had zeer vaak bugs. Toch heb ik na vele updates, issue reports en troubleshooting het belangrijkste opgelost gekregen bij de release van push-plugin 1.6.0.

https://github.com/phonegap/phonegap-plugin-push/issues/466
https://github.com/phonegap/phonegap-plugin-push/issues/734

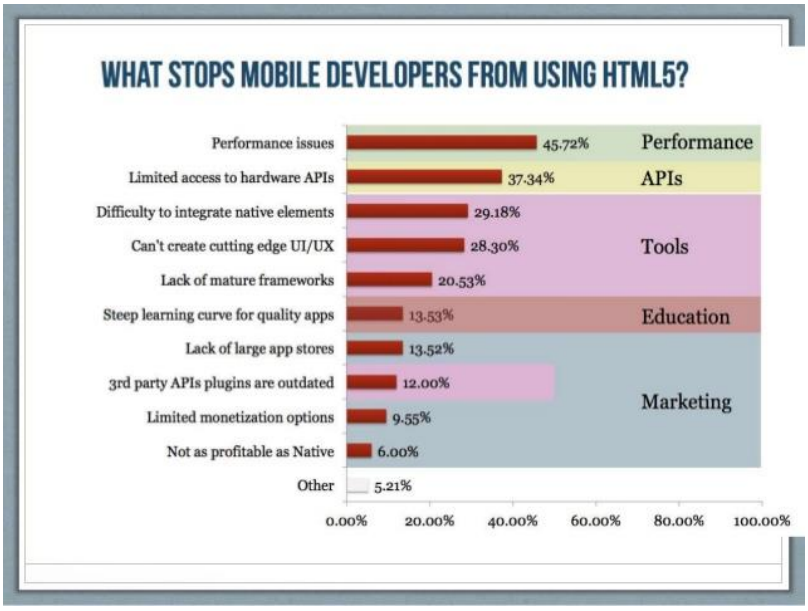
5.1 HYBRIDE APPLICATIE

Omdat mijn kennis om iOS applicaties te maken niet volledig was, heb ik gekozen om niet native te werken. Dit brengt ook enkele voordelen voor de klant:

- De bedrijfsleider (Kristof De Smedt) is een webdeveloper, hybride applicaties in HTML, CSS, JS zijn voor hem makkelijker te onderhouden.
- Sneller compatibel maken voor nieuwe iOS of Android devices en zelf eventueel Windows Phone support.

De nadelen van hybride applicaties zijn:

- Minder snel als native applicaties, al zal dat voor deze opdracht niet hard opvallen
- Native functies aanspreken vereist vaak meer werk, al kan je steeds in de hybride taal blijven
- Opstarttijd van de applicatie is vaak langer mits de UI moet gegenereerd worden
- Aanpassingen aan design patterns (iOS naar flat design, Android naar material design) vereisten dat je zelf de CSS hiervoor gaat maken (of berust op frameworks die mogelijk lang wachten met updaten)



Grafiek is afkomstig van: VisionMobile http://www.visionmobile.com/blog/2013/12/html5-performance-is-fine-what-we-are-missing-is-tools/

5.2 FRAMEWORK

Voor hybride applicaties zijn er verschillende opties. Cordova is het meest gebruikte framework en biedt ook zeer veel native plug-ins aan die het mogelijk maken om beter te interageren met het toestel.

Designers kiezen vaak voor Cordova zonder het gebruik van een UI framework omdat ze zich goed thuis voelen in het designen van sites, wat ongeveer hetzelfde is wanneer je een Cordova site gaat gebruiken.

Als niet-designer en programmeur zou het voor mij het makkelijkste zijn om een UI framework te gebruiken mits ik persoonlijk zelf niet goed ben in grafische vormgeving, dit is iets waar ik rekening mee heb gehouden bij het kiezen en onderzoeken van de frameworks.

Een framework moet voor mij drie dingen kunnen:

1. Het moet toestaan dat van 1 taal een iOS en Android applicatie gemaakt kunnen worden
2. De UI moet makkelijk te maken zijn en aan te passen
3. Er moet een hybride manier zijn om de UI te wijzigingen zodat er geen duplicate code ontstaat

Voor mij moet het framework ook vaak geüpdatet worden: Een framework of library dat vaak geüpdatet wordt, betekend voor mij dat het vaak zal updaten naar nieuwe UI en UX design patterns.

De voor en-na delen van het gebruik van een framework zijn:

Voordelen:

- Het is vaak makkelijker om je eigen design in te brengen dan native
- De frameworks zijn vaak open source en zullen dus door de community geüpdatet worden.

Nadelen:

- Mobile OS updates kunnen mogelijk ervoor zorgen dat het framework niet meer werkt al zal dit vooral om design guidelines gaan en mogelijke native functies.
- De frameworks zijn vaak open source en kunnen mogelijk plots stoppen met updaten
- Vaak dienen deze meer getest te worden op verschillende toestellen

5.2.1 Optie 1: Ionic framework

Ionic is een framework gebaseerd op cordova. Het heeft standaard een mooie lay-out en zelfs een grafische creator die het toelaat om knoppen en andere componenten te slepen. Echter is deze creator zeer gelimiteerd en zal je uiteindelijk zelf aan de CSS moeten wijzigen om het juiste resultaat te verkrijgen.

Het wordt vaak aangeprijst als het meest performante HTML hybride framework door hardware versnelling.

Op het eerste zicht was ik erg geïnteresseerd in dit framework, het beloofde om alles te vergemakkelijken:

- Veel modules te koop zodat je zelf niet veel moet designen
    - Prijs valt goed mee
- Ionic Push om makkelijk push notificaties te sturen
    - Betalend wanneer ionic uit BETA is
- SASS support om makkelijk css aanpassingen te maken
- Veel tools gaande van het genereren van icons en splash screens tot het beheren

Echter waren veel van die tools een teleurstelling. Ze zijn zeer handig en werken zo goed als dat ze voorgesteld worden, maar van zodra je kleine wijzigingen wil maken moet je al snel over schakelen op standaard cordova implementaties in plaats van de ionic wrappers te gebruiken.

Mits een van mijn doelen was om een snel en stabiel framework te vinden, heb ik wat zitten testen of ionic werkelijk sneller is dan phonegap met OnsenUI, de kans is inderdaad dat het een beetje sneller is maar dit is niet merkbaar. Het grootste probleem – namelijk traag laden in het begin, was nog steeds visueel zichtbaar zonder splash screen.

De UI van het framework is vooral voor iOS gericht maar er is een material design beschikbaar. Op het moment dat ik aan mijn final work begon (nu Mei 2016 is dit al wat verbeterd) nog redelijk sober met weinig material designcomponenten.

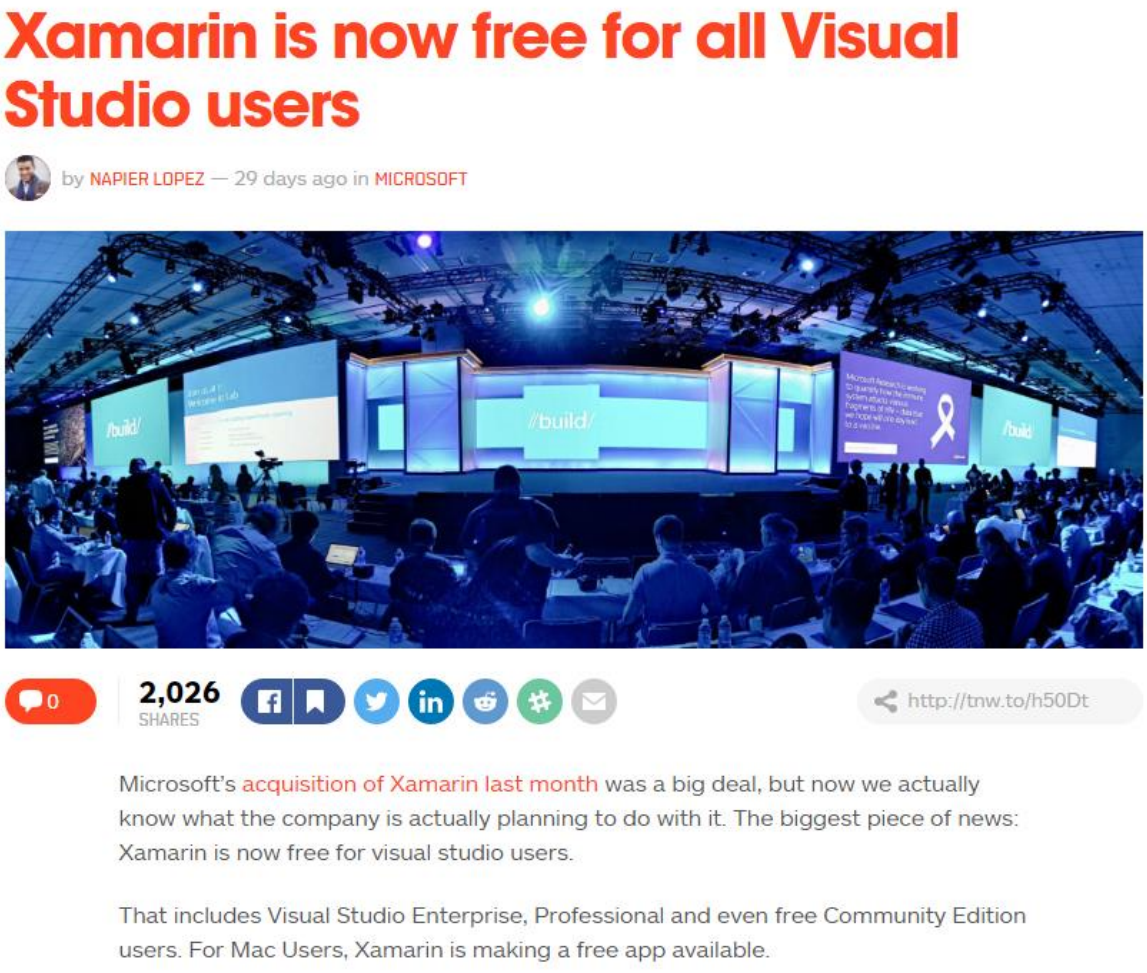
Het blijft een framework waar ik erg hard aan getwijfeld heb om te gebruiken. Ik ben in April aan een prive project begonnen in Ionic dat mij terug tot rust bracht, na een week ben ik naar OnsenUI 2 overgestapt mits de tools niet voldeden aan mijn vereisten.

Om kort samen te vatten wat Ionic is: *Het is het perfect framework met de perfecte tools. Maar van zodra je iets meer moet doen dan de tools bieden, begin je vanaf nul.*

5.2.2 Optie 2: Xamarin

Ondanks dat xamarin geen web gebaseerd framework is, is het toch makkelijk om de lay out met drag en drop te wijzigen. Echter is het betalend en vereist nog veel research wat mogelijk lang kan duren.

Wijziging April 2016: Begin April 2016 werd aangekondigd dat Xamarin gratis beschikbaar wordt gesteld voor alle Visual Studio (community) gebruikers. Dit komt door de overname van Xamarin door Microsoft in Maart 2016.



Xamarin is now free for all Visual Studio users

by NAPIER LOPEZ — 29 days ago in MICROSOFT

0 2,026 SHARES http://tnw.to/h50Dt

Microsoft's acquisition of Xamarin last month was a big deal, but now we actually know what the company is actually planning to do with it. The biggest piece of news: Xamarin is now free for visual studio users.

That includes Visual Studio Enterprise, Professional and even free Community Edition users. For Mac Users, Xamarin is making a free app available.

5.2.3 Optie 3: Phonegap met jQuery Mobile

jQuery is een framework waar ik zeer vertrouwd in ben en ook populair is, daarom vond ik het nodig om ook te controleren of hetzelfde gezegd kon worden over het mobile framework. Hoewel de onderliggende functies veel mogelijkheden hebben zoals swypen, touch controls, … is de lay out sterk verouderd en voelt het zeer "website"-achtig aan.

jQuery is zeer bekend onder web developers en het verbaasd me ook niet dat veel web designers dit kiezen omdat ze de functies door en door kennen. Echter is het zeer moeilijk om een MVC-applicatie te maken met enkel jQuery, waardoor je applicatie al snel meerdere frameworks bevat (trager laden).

5.2.4 Optie 4: Intel XDK

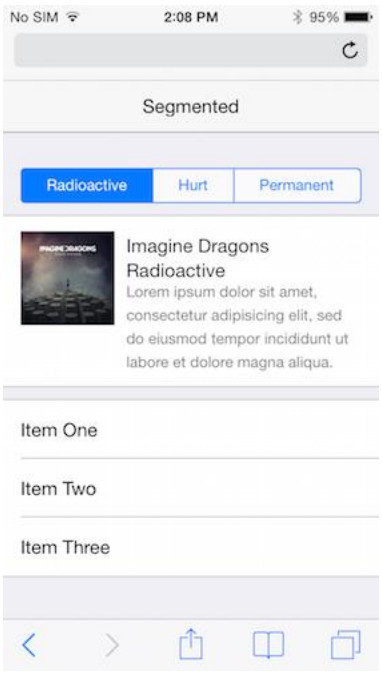
Intel XDK is slechts een ontwikkelomgeving. Maar komt met handige tools zoals live-preview en vele frameworks waaruit je kan kiezen. Van op het eerste zicht lijkt deze IDE het beste om de applicatie te maken.

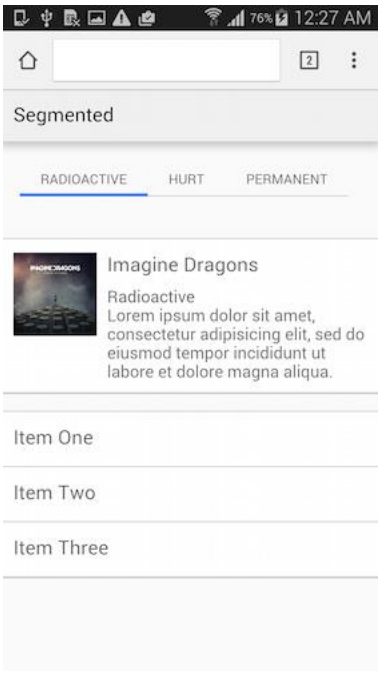
In eerste instantie ben ik begonnen om mijn applicatie met deze IDE te schrijven, maar omdat alles met drag and drop werkte was het zeer moeilijk om een eigen inbreng te geven aan de lay-out van de applicatie en design patterns.

5.2.5 Optie 5: ChocolateUI

ChocolateUI is een UI framework zoals OnsenUI. Het grote voordeel is dat de lay-out voor Windows Phone er zeer goed uit ziet, het nadeel is dat de UI bij iOS en Android er zeer gedemodeerd uitziet.

http://chocolatechip-ui.github.io/index.html





5.2.6 Optie 6: Phonegap met OnsenUI

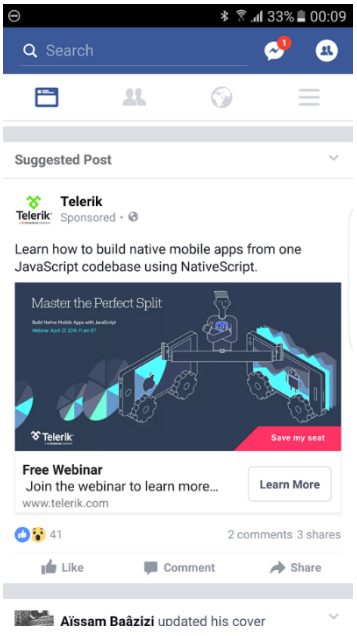
OnsenUI is een layout template specifiek voor tablets of smartphones. Het wordt vaak geüpdatet en beschikt over een een template color palette waardoor de maintainer van de applicatie makkelijk het framework kan updaten zonder alle kleuren terug juist te zetten.

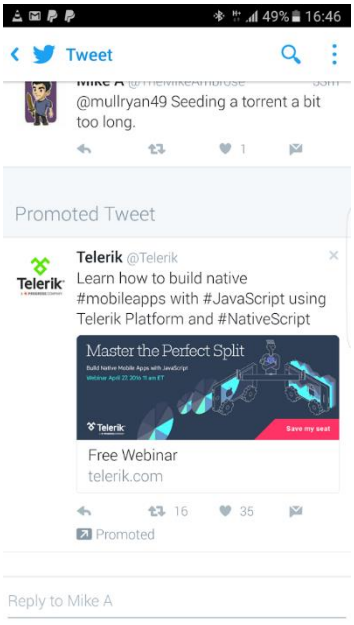
Phonegap is een Cordova gebaseerd framework zoals Ionic maar dan zonder alle extra mogelijkheden die Ionic bied. Het is simpel in gebruik en er is zeer veel vrijheid op vlak van lay-out.

OnsenUI zelf bevat de optie om AngularJS of React te gebruiken. Zelf maak ik al enkele jaren applicaties in AngularJS dus was het voor mij een snelle keuze om hiervoor te gaan.

Bovendien vind ik React meer een “library” aan hulpmiddelen en AngularJS een echt MVC framework waar je out of the box iets mee kan doen.

5.2.7 Optie 7: Kendu UI



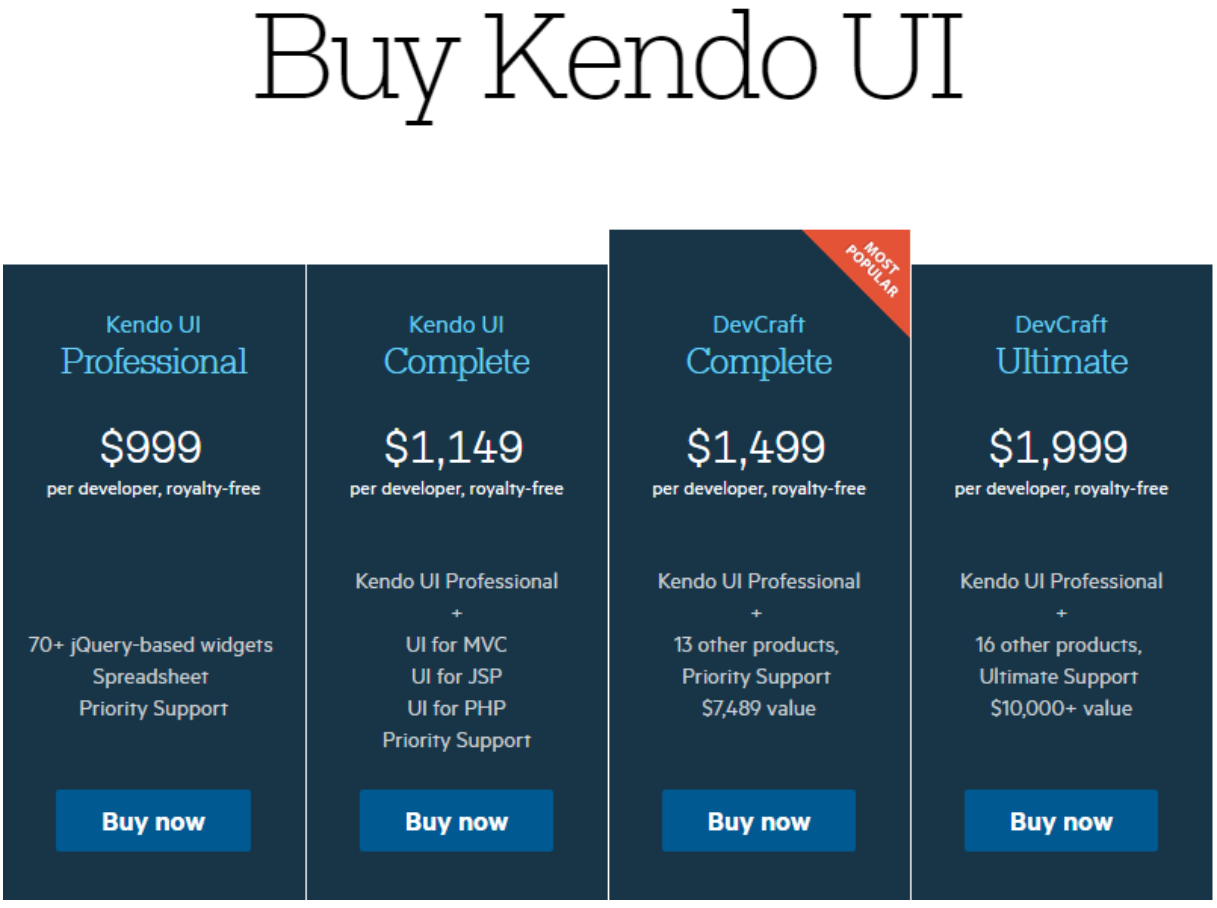


Kendu UI is een framework dat ik bekeken heb midden mijn final work, mits ik overspoeld werd met reclame voor dit framework op twitter en facebook (dus vond ik het de moeite om even te bekijken).

Telerik is de maker van enkele UI frameworks voor HTML5 en .NET.

Het framework zag er veelbelovend uit toen ik de componenten bekeek, het was niets speciaal maar wel makkelijk en simpel.

Echter bleek mijn geluk van korte duur toen bleek dat dit betalend was en nog minder toen ik de prijs te zien kreeg.



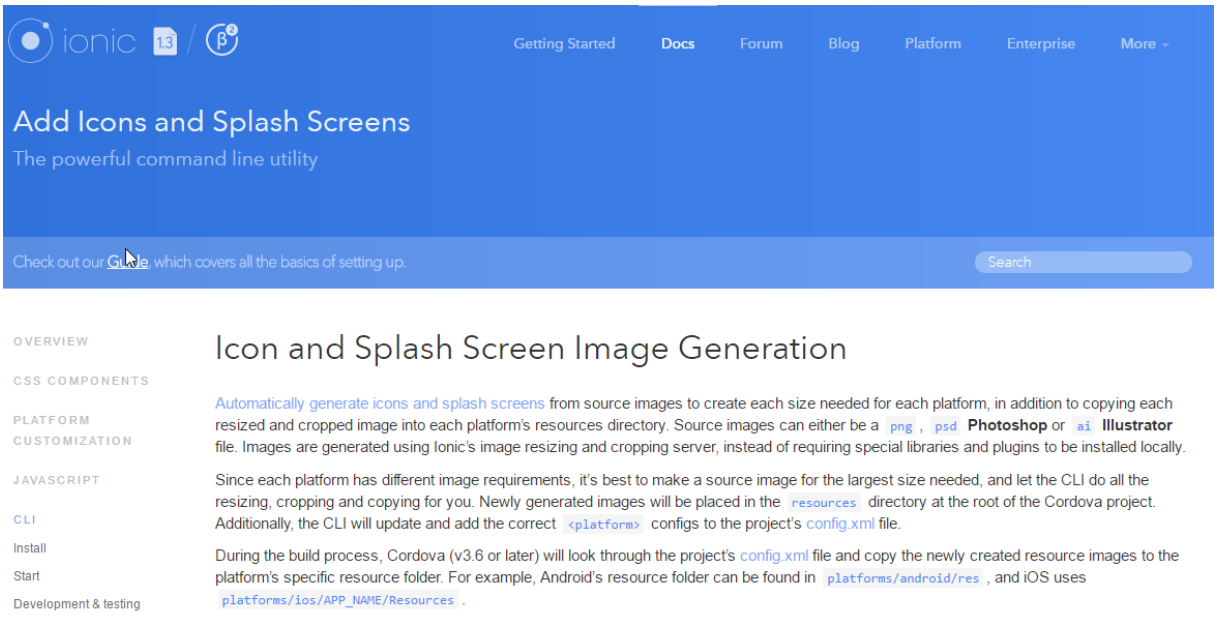
5.2.8 Gekozen framenwork

Uiteindelijk na enkele frameworks getest te hebben is er gekozen voor Phonegap met OnsenUI als lay-out framework om volgende redenen:

- Het framework is me bekend van voorgaand onderzoek
- Makkelijk te gebruiken, vereist geen account of IDE. Dit maakt het ook makkelijker om het project door te geven aan het bedrijf
- Je wordt niet geforceerd om een bepaalde lay-out template te gebruiken. Je hebt hier de eigen keuze in
- OnsenUI is makkelijk in gebruik en heeft ook tools om het kleurenpallet automatisch te updaten indien een nieuwe versie van OnsenUI uitkomt.
- Uit vorige updates weet ik dat ze trends volgen. In Maart van mijn final work hebben ze bijvoorbeeld support toegevoegd voor material design. En in Februari 2015 (voor het final work) hebben ze nog enkele coole features toegevoegd die alles veel makkelijker maken.

Ondanks dat ik voor OnsenUI had gekozen heb ik toch nog wel enkele keren getwijfeld om toch niet over te schakelen naar Ionic. Na het proberen porten naar Ionic kwam ik echter met zoveel problemen zoals trage push notificaties, UI glitches door de alpha versie van IonicMaterial dat ik mijn besluit terug heb bevestigd om bij OnsenUI te blijven.

Wel heb ik de Ionic Resource tool gebruikt om mij te helpen met het resizen van splash screens en icons.



5.3 VEILIGHEID

Op het eerste zicht is een normale Android of iOS applicatie even makkelijk te reverse engineeren als een hybride applicatie als het neerkomt op het sniffen van packets die verstuurd worden. In principe is het echter geen probleem als iemand anders de API kan achterhalen zolang de applicatie geen sensitieve informatie bevat.

Een hybride HTML-applicatie is zeer makkelijk te 'reverse engineeren' aangezien alle HTML en JS-files als resources in de package zitten.

http://developer.telerik.com/featured/securing-phonegapcordova-hybrid-mobile-app/

Bovenstaand artikel bevat enkele zeer interessante denkpistes om een hybride applicatie te beschermen

5.3.1 Javascript remote laden

Een van de zaken die wordt besproken in bovenstaand artikel is dat de javascript remote geladen kan worden nadat authenticatie (van de bouwheer) is voltooid. Dit zorgt er echter voor dat deze steeds opnieuw moeten gedownload worden en het is geen probleem mits de javascript geen sensitieve informatie bevat.

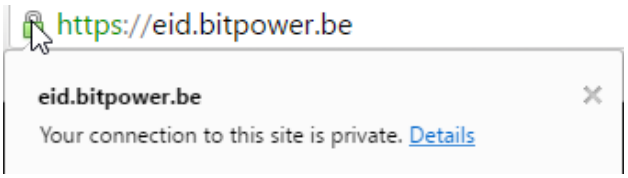
Het is wel aangeraden wanneer je ervoor wil zorgen dat bepaalde functionaliteiten niet makkelijk te achterhalen zijn door de concurrentie.

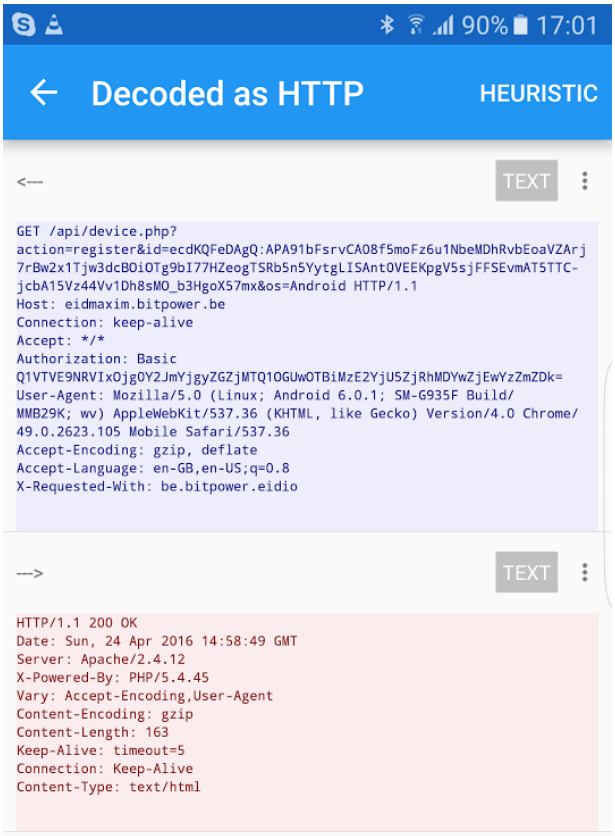
5.3.2 Data beveiligd versturen

Alle interactie met de EIDIO-servers zouden moeten gebeuren via HTTPS om ervoor te zorgen dat men geen gegevens zal kunnen achterhalen met bijvoorbeeld wireshark.

De enige manier om gegevens te achterhalen is door een applicatie als Packet Capture, die lokaal een proxy opzetten die het naar HTTP kan decoderen.

Tijdens mijn final work gebruikte ik een test API die geen certificaat had, echter heeft de echte API een HTTPS certificaat en zal alle communicatie dus veilig verlopen.



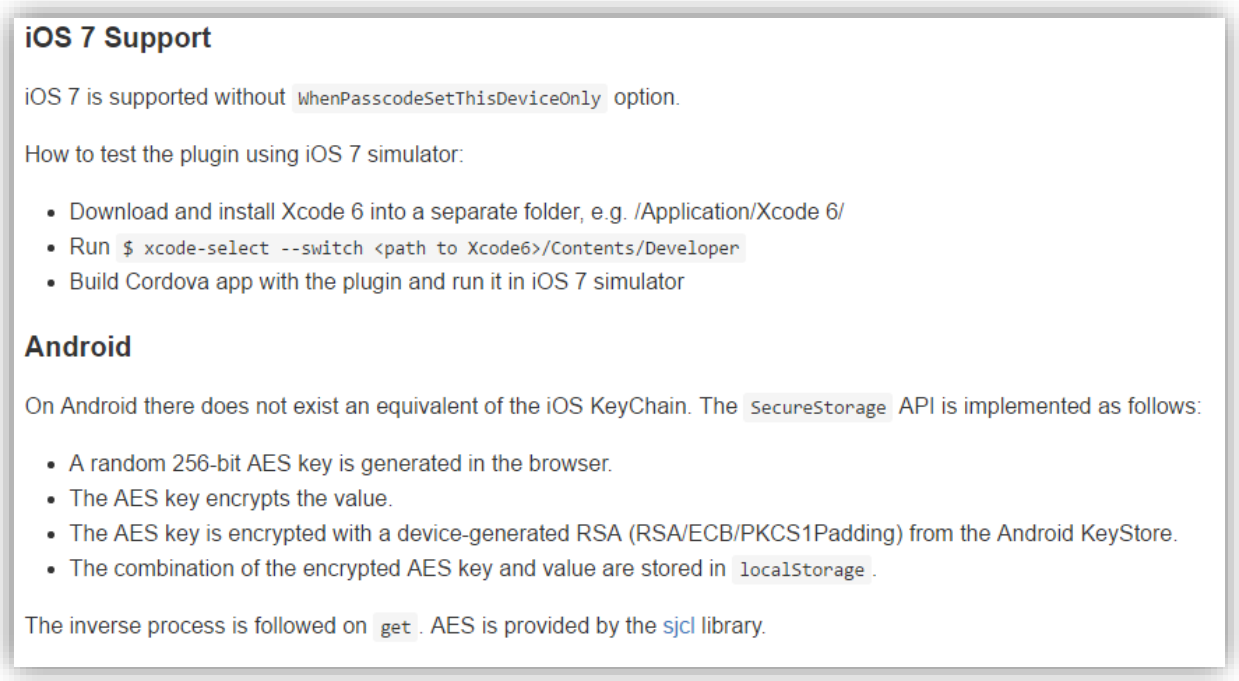


5.3.3 Data beveiligd bewaren

Zoals aangeraden in het artikel wordt er gebruikgemaakt van localStorage in plaats van een database plugin te gebruiken. Alle gegevens in de localStorage zijn enkel toegankelijk in de EIDIO app. Het wachtwoord wordt als SHA1 hash bewaard.

Om iOS KeyChains te gebruiken en ook Android beter te beveiligen heb ik gebruik gemaakt van Secure storage https://github.com/Crypho/cordova-plugin-secure-storage

Dit werkt verder op "localStorage" om deze nog beter te beveiligen met KeyChains voor iOS en AES-encryptie bij Android.



iOS 7 Support

iOS 7 is supported without `WhenPasscodeSetThisDeviceOnly` option.

How to test the plugin using iOS 7 simulator:

- Download and install Xcode 6 into a separate folder, e.g. /Application/Xcode 6/
- Run `$ xcode-select --switch <path to Xcode6>/Contents/Developer`
- Build Cordova app with the plugin and run it in iOS 7 simulator

Android

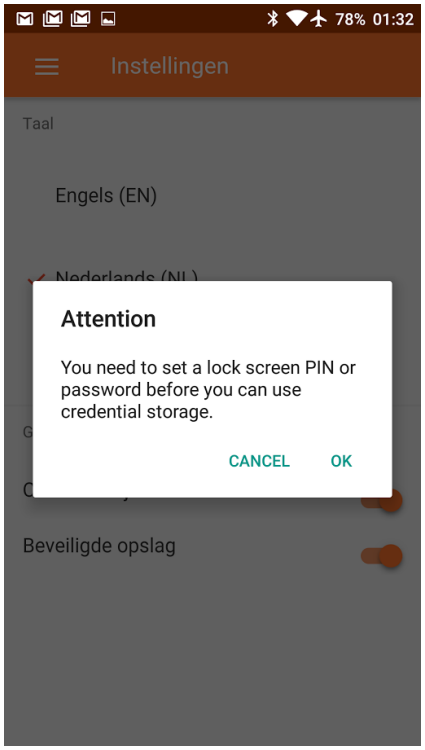
On Android there does not exist an equivalent of the iOS KeyChain. The `SecureStorage` API is implemented as follows:

- A random 256-bit AES key is generated in the browser.
- The AES key encrypts the value.
- The AES key is encrypted with a device-generated RSA (RSA/ECB/PKCS1Padding) from the Android KeyStore.
- The combination of the encrypted AES key and value are stored in `localStorage`.

The inverse process is followed on `get`. AES is provided by the sjcl library.

Dit heeft wel enkele nadelen:

- Het laden, saven en verwijderen van content uit de secure storage moet asynchroon gebeuren en dus moeten er callbacks meegegeven worden.
- Er moet een lock screen wachtwoord, pincode of patroon worden ingesteld. Dit wordt gebruikt als private key.
    - Mocht de applicatie ooit ingebouwd worden in bijvoorbeeld een vaste tablet op de muur, dient dit voor gebruiksvriendelijke redenen uitgezet te worden.
    - NOTA: Voor iOS moet er geen lock screen worden gebruikt mits er gebruik wordt gemaakt van keychains.



5.3.4 Wees voorzichtig met plug-ins

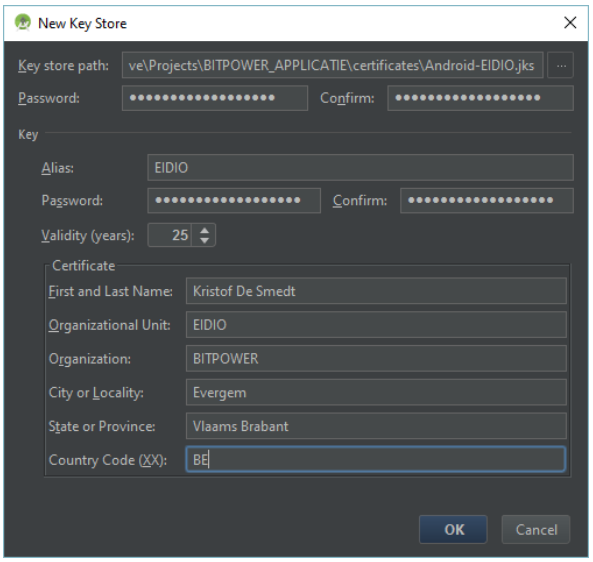
Plug-ins zijn steeds een black box waar je niets van afweet zonder zelf in de code te gaan duiken. Ik ben zeer kritisch geweest met de plug-ins die ik installeerde of nodig had om ervoor te zorgen dat deze van betrouwbare bronnen kwamen zoals de push plug-in die door een werknemer van Adobe is geschreven.

5.4 RELEASE KLAAR

Na zelf enkele applicaties te hebben gemaakt weet ik dat het steeds toch nog wat werk is om de applicatie effectief te uploaden. De eIDIO applicatie zou echter release klaar zijn zowel intern als de benodigde assets om deze te publiceren.

5.4.1 Android

Voor android moeten er enkele zaken zoals een keystore en artwork worden aangemaakt. Deze informatie is nodig om de applicatie op de Play Store te zetten.

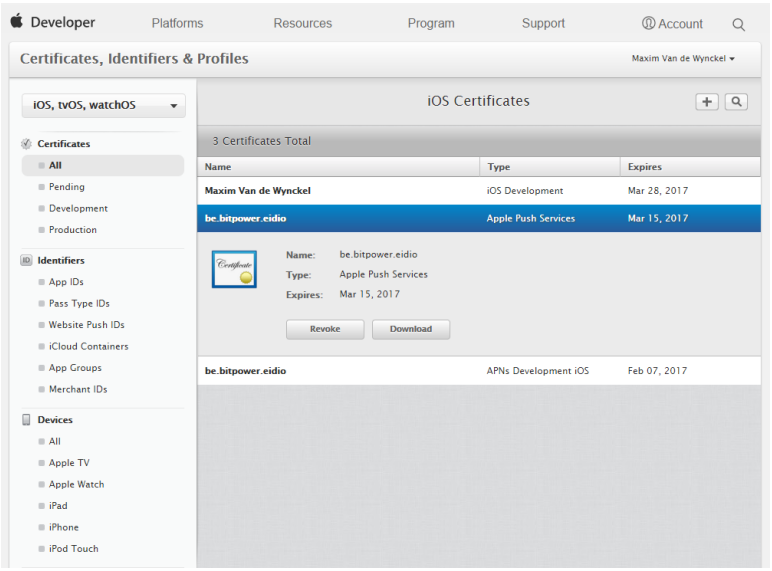


De keystore heb ik laten aanmaken in Android Studio, mits dit een makkelijke UI heeft om de keystore in te stellen.

5.4.1.1 *Artwork*

Als niet grafisch ontwerper heb ik dit gedeelte weg gelaten omdat het hoogstwaarschijnlijk veel beter kan. Ook is de kans groot dat er UI wijzigingen gebeuren aan de applicatie vooraleer deze in productie gaat.

5.4.2 iOS



Voor iOS dienen er iets meer zaken te gebeuren. Eerst en vooral moet je applicatie gecompileerd worden voor iOS 7 of hoger.

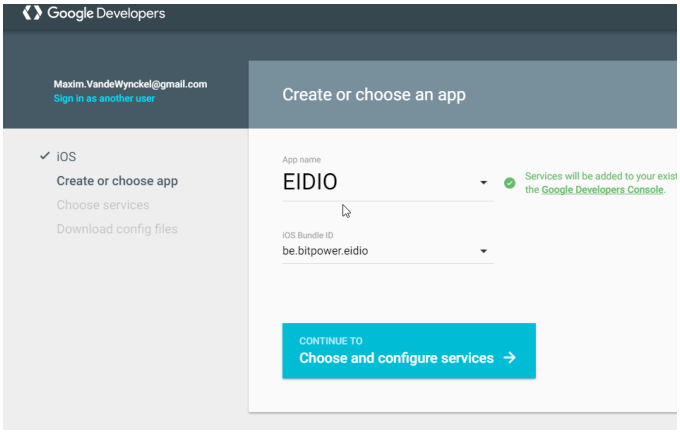
Het belangrijkste bij iOS is dat er certificaten moeten aangemaakt worden voor de push notificaties en iOS applicatie. Dit dient te gebeuren bij developer.apple.com, de SDK kost 99 euro per jaar.

Voor Bitpower is het belangrijk om te weten hoe hij deze certificaten zelf gaat moeten aanmaken, mits hij deze nodig zal hebben om de applicatie in de toekomst te onderhouden.

5.4.2.1 *Uploaden naar GCM*

Omdat we GCM gebruiken voor notificaties naar APNS te sturen dienen we het APNS-certificaat te uploaden naar Google.

https://developers.google.com/mobile/add?platform=ios



Op bovenstaande site kan men aan een bestaand Google App project met GCM support een APNS certificaat toevoegen.

Let wel op: GCM device ID's zijn niet gelijk aan iOS device ID's.

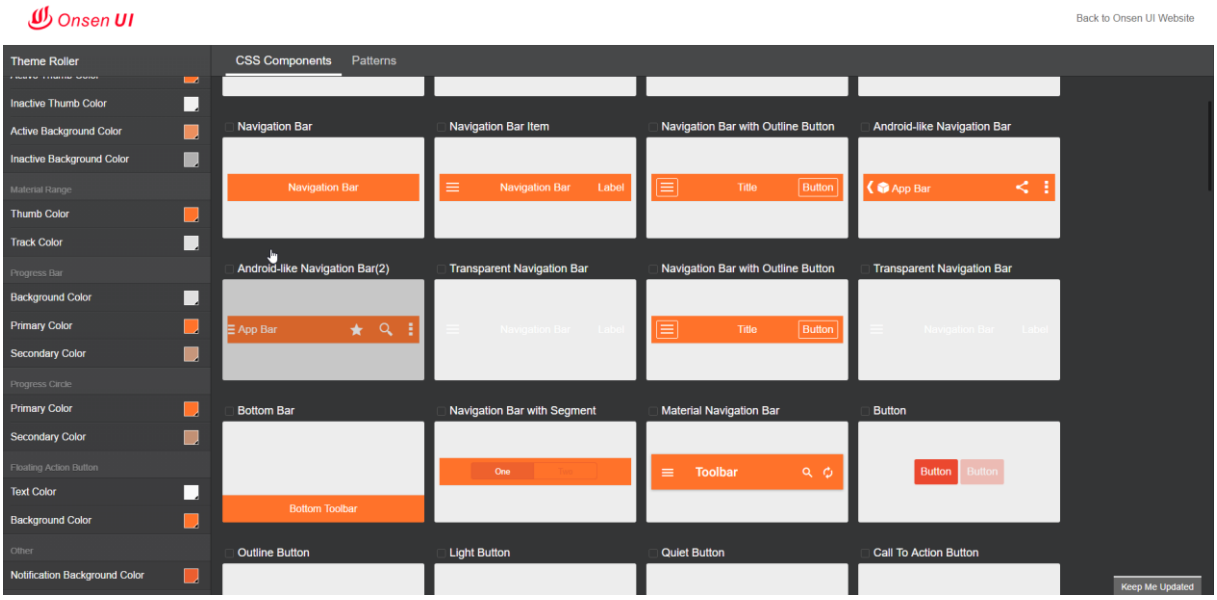
Update Mei 2016: Google raad aan om naar FCM over te schakelen als je iOS en Android wil ondersteunen.

5.5 UI & UX APPLICATIE

De UI van de applicatie was niet het allerbelangrijkste tijdens mijn final work maar was toch belangrijk.

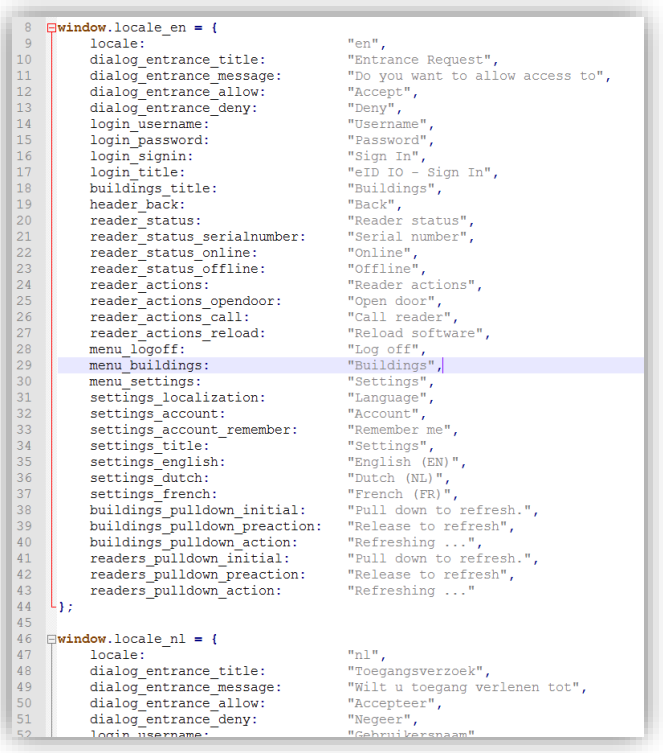
Ik heb een onderscheid gemaakt tussen de UI voor iOS en de UI voor Android. Dankzij het OnsenUI framework dat ik gebruikt heb, kon sinds Maart 2016 gebruik gemaakt worden van auto-modifiers om naar gelang het platform de UI 'material' of 'flat' eruit te laten zien.

De "Theme Roller" van OnsenUI laat toe om de kleuren van de componenten goed af te stemmen.



5.5.1 Taal

De taal kan worden ingesteld voor Nederlands, Engels of Frans. Het makkelijke systeem zorgt er echter voor dat men makkelijk meerdere talen kan toevoegen door de “localization.js” file aan te passen.



```
window.locale_en = {
    locale:                     "en",
    dialog_entrance_title:      "Entrance Request",
    dialog_entrance_message:    "Do you want to allow access to",
    dialog_entrance_allow:      "Accept",
    dialog_entrance_deny:       "Deny",
    login_username:             "Username",
    login_password:             "Password",
    login_signin:               "Sign In",
    login_title:                "eID IO - Sign In",
    buildings_title:            "Buildings",
    header_back:                "Back",
    reader_status:              "Reader status",
    reader_status_serialnumber: "Serial number",
    reader_status_online:       "Online",
    reader_status_offline:      "Offline",
    reader_actions:             "Reader actions",
    reader_actions_opendoor:    "Open door",
    reader_actions_call:        "Call reader",
    reader_actions_reload:      "Reload software",
    menu_logoff:                "Log off",
    menu_buildings:             "Buildings",
    menu_settings:              "Settings",
    settings_localization:      "Language",
    settings_account:           "Account",
    settings_account_remember:  "Remember me",
    settings_title:             "Settings",
    settings_english:           "English (EN)",
    settings_dutch:             "Dutch (NL)",
    settings_french:            "French (FR)",
    buildings_pulldown_initial: "Pull down to refresh.",
    buildings_pulldown_preaction: "Release to refresh",
    buildings_pulldown_action:  "Refreshing ...",
    readers_pulldown_initial:   "Pull down to refresh.",
    readers_pulldown_preaction: "Release to refresh",
    readers_pulldown_action:    "Refreshing ..."
};

window.locale_nl = {
    locale:                     "nl",
    dialog_entrance_title:      "Toegangsverzoek",
    dialog_entrance_message:    "Wilt u toegang verlenen tot",
    dialog_entrance_allow:      "Accepteer",
    dialog_entrance_deny:       "Negeer",
    login_username:             "Gebruikersnaam"
```

In de toekomst zou het beter zijn om de vertalingen in aparte “.json” files te steken in plaats van deze statisch in de javascript file te steken. Dit maakt het makkelijker om vertalingen aan te vragen bij vertaalbureaus.

5.5.2 Kleuren Bitpower

De kleuren van het logo en de site van Bitpower zijn wit en oranje. Deze kleuren zijn mee in de applicatie gebracht.

Klein wistjedatje: Bij het testen op over gesatureerde samsung toestellen heb ik het oranje “iets” donkerder gemaakt om niet een te “flashy” effect te krijgen.

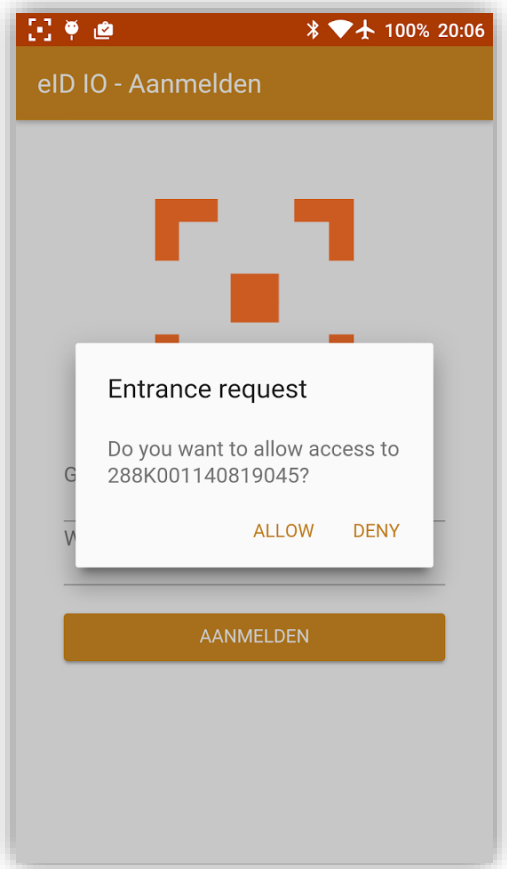
5.5.3 Logo ontwerp

Het logo van Bitpower is zeer simpel en is dus zeer makkelijk te gebruiken in iOS flat en Android Materialize design patterns. Toch heb ik enkele wijzigingen gemaakt aan het logo om het zowel te laten voldoen aan de iOS en Android style guidelines. Normaal wordt een “long shadow” in een rechte hoek gemaakt, maar ik heb bewust gekozen om dit wat naar buiten te laten schijnen om het effect te geven van een deur die open staat waar licht door schijnt.

5.5.4 Notificatie looks en acties

De notificatie die een gebruiker krijgt bij een belsignaal is het belangrijkste van de applicatie, het is ook normaal dat hier zeer veel tijd in is gestoken om dit stabiel en snel te krijgen op zowel iOS en Android.

5.5.4.1 *Voorstelling wanneer applicatie open is*



Omdat ik ervan uit ga dat eventueel in de toekomst de applicatie voor tablets gaat dienen en misschien vast in een muur gaan zitten heb ik hier wat onderzoek naar gedaan wat de meest bruikbare manier is om dit te doen.

Hiervoor heb ik gekeken hoe notificaties (alarm, brand, ...) werken op een touchscreen van Niko Home control (Linux).

Notificaties blijven hier in het midden staan boven alles, maar ze gaan niet weg totdat je op elk paneel dit wegdrukt.

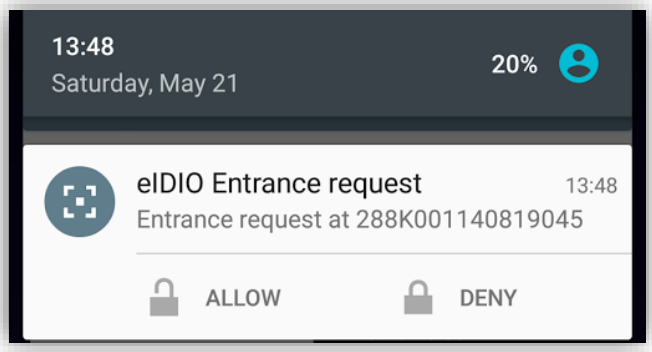
In mijn applicatie heb ik ervoor gezorgd dat een notificatie ook in het midden komt met een optie om de deur te openen of gesloten te houden, maar anders dan Niko Home control heb ik er een time-out op gezet dat de notificatie weg gaat na enkele minuten.

De notificatie acties werken enkel en alleen als de login gegevens bekend zijn.

Indien het toestel wel geregistreerd is, maar nog niet ingelogd is dan zal deze eerst moeten inloggen.

5.5.4.2 *Voorstelling wanneer applicatie gesloten is*

Wanneer de applicatie gesloten is zal de melding in de notification drawer terecht komen. Hier zullen ook knoppen voorzien zijn voor het openen/gesloten houden van de deur. Door problemen met de push plugin (zie inleiding hoofdstuk 5) kunnen de acties nog niet in de achtergrond uitgevoerd worden en zal de applicatie openen

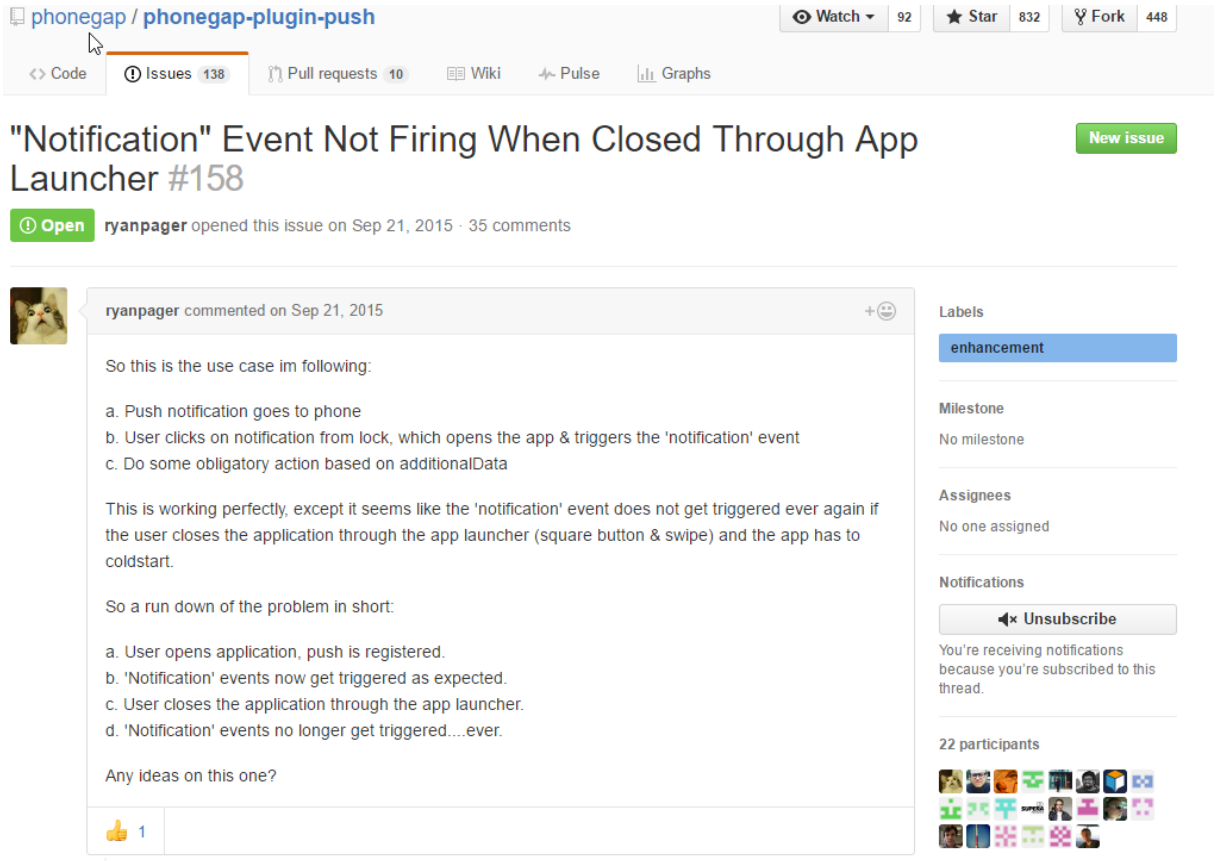


Voor iOS zal het kleuren icoontje getoond worden. Voor Android zal voor de meeste toestellen een wit icoontje worden weergegeven van het normale logo van Bitpower.

5.5.4.3 *Voorstelling op Android Wear/Pebble OS*

De push plugin die gebruikt wordt heeft ook de mogelijkheid om de notificaties en acties weer te geven op Android Wear en PebbleOS.

Echter doordat deze momenteel nog niet toelaat dat acties in de achtergrond worden gestart zal dit momenteel nog niet zo handig zijn totdat deze 'enhancement' is opgelost.

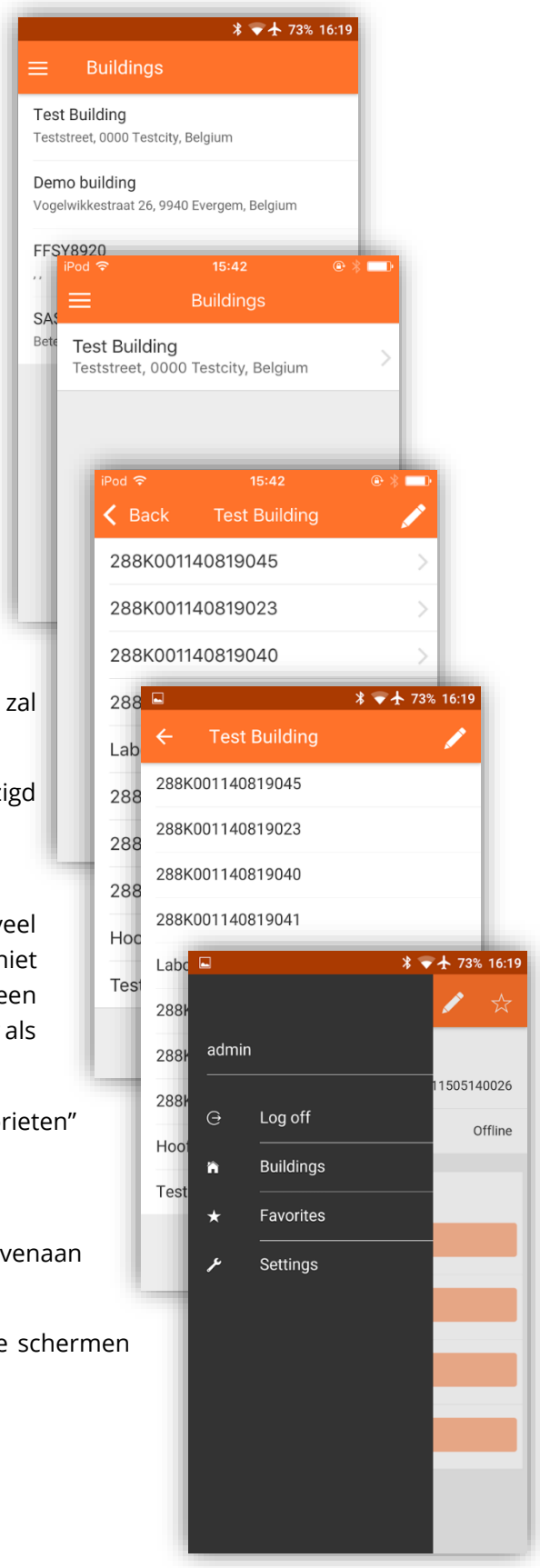


5.5.5 Oplijsten van gebouwen

Een klant kan meerdere gebouwen hebben, de applicatie moet een lijst van de gebouwen tonen waar de klant rechten toe heeft zodat men hier verder kon selecteren.

De UI van de lijst is niets bijzonder. Onderaan vind je twee screenshots waarvan de eerste gemaakt is met Android (Administrator account) en de tweede met iOS (Klant account).

In de lijst wordt alsook het adres getoond van het gebouw. De lijst zal een pijltje rechts krijgen wanneer dit op iOS wordt weergegeven.



5.5.6 Oplijsten kaartlezers

Het oplijsten van de kaartlezers gebeurt aan de hand van een API-call zodra een gebruiker een gebouw selecteert. Indien de gebruiker of installateur een alias gegeven heeft aan de kaartlezer (wat meestal het geval zal zijn, maar bij deze test kaartlezers niet) zal de naam weergegeven worden. Indien er geen alias ingegeven is zal het serienummer meegegeven worden.

De naam van het gebouw kan in de applicatie gewijzigd worden door het drukken op het pennetje.

5.5.7 Favorieten

Al snel merkte ik bij het testen dat een klant zoveel gebouwen/kaartlezers kan hebben dat hij/zij misschien niet altijd wil zoeken naar de juiste kaartlezer. Daarom heb ik een “favorieten” functie gemaakt die toelaat om een kaartlezer als favoriet aan te duiden.

Deze komen dan allemaal in een apart menu item “Favorieten” terecht.

5.5.8 Menu

Voor het menu heb ik een donkergrijze kleur gekozen. Bovenaan staat de username van de ingelogde gebruiker.

Vanaf het menu kan men zich afmelden en naar andere schermen gaan zoals:

- Gebouwen
- Favorieten
- Instellingen

5.5.9 Kaartlezer detail

Het belangrijkste scherm in de applicatie is de kaartlezer in detail. Op dit scherm kan je zien of een kaartlezer online is en vervolgens enkele acties uitvoeren.

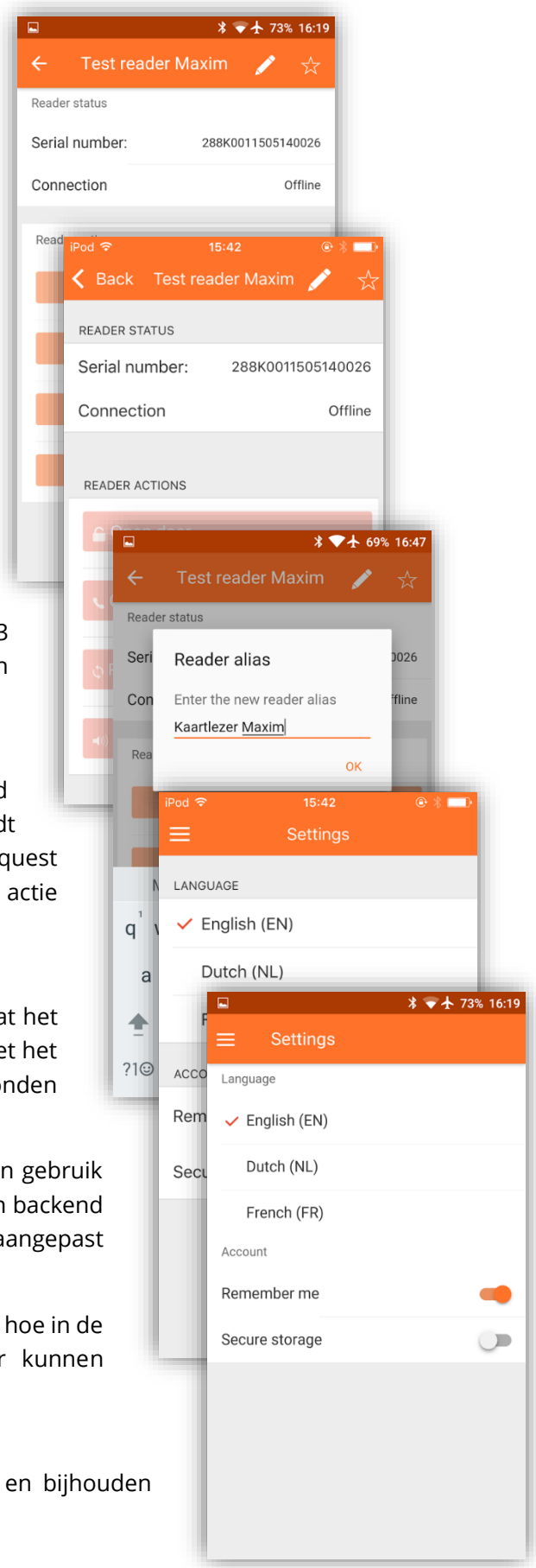
Volgende acties zijn geïmplementeerd:

- Openen van de deur
- Herladen van de software (Proof of concept)
- Het laten opbellen: *De kaartlezer zal een skype call starten met een ingesteld nummer*
- Geluid afspelen (Proof of concept)

Als de kaartlezer offline is zullen de knoppen niet kunnen worden ingedrukt.

In de toekomst kan dit scherm eventueel uitgebreid worden met meer acties voor de kaartlezer of configuratie instellingen.

Het maken van een nieuwe actie neemt hooguit 2-3 minuten in beslag om dit in de applicatie, kaartlezer en server te implementeren.



5.5.9.1 *Online/Offline weergave*

De online status van de kaartlezers gebeurt aan de hand van de online status met de websocket server. Er wordt elke 5 seconden (wanneer de pagina open staat) een request gestuurd naar de REST API welke een “connected” actie uitvoert met de websocket servlet.

5.5.9.2 *Aanpassen van de alias*

Iedereen die de applicatie begon te testen merkte op dat het zeer lastig was dat een kaartlezer werd weergegeven met het serienummer. Hierop zei ik steeds dat ze dit konden aanpassen in het online web paneel.

Ik vond dit echter omslachtig omdat aliassen pas echt in gebruik worden genomen bij deze applicatie. Daarom heb ik een backend functie gemaakt die toelaat dat de naam kan worden aangepast vanuit de applicatie.

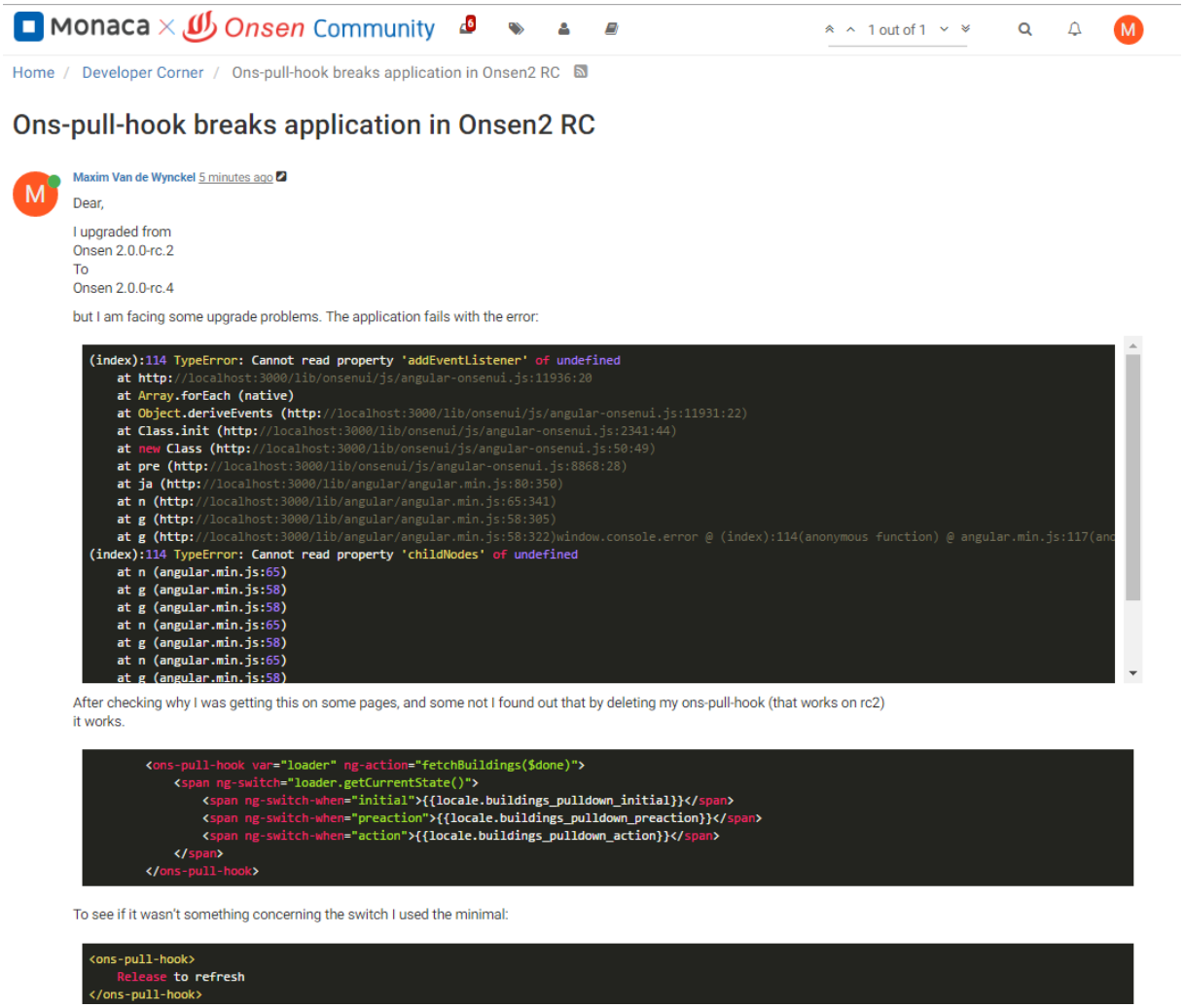
Naast een handige functie is dit ook een goed voorbeeld hoe in de toekomst gegevens van een kaartlezer of gebruiker kunnen worden aangepast.

5.5.10 Instellingen

Bij het instellingen scherm kan men de taal instellen en bijhouden of/hoe de gebruiker bewaard dient te worden.

5.6 Upgrade van Onsen 1 naar 2

De upgrade van OnsenUI 1 naar OnsenUI verliep zeer stroef, ik heb vaak bug reports moeten maken voor problemen in het framework tijdens de release candidates en beta.



https://community.onsen.io/topic/404/ons-pull-hook-breaks-application-in-onsen2-rc
https://github.com/OnsenUI/OnsenUI/issues/1402

Wat ik lastig vond tijdens het rapporteren was om 'genoeg' maar niet te veel informatie te geven. De meeste van mijn vrije projecten zijn open source, dus is het makkelijk bij foutrapportering om naar een stuk code te verwijzen. Met deze applicatie moet ik vaak proof-of-concept applicaties/functies maken om met minimale code een probleem aan te kunnen tonen.

Bij het posten van screenshots moet ik ook erg letten op niet te veel informatie te tonen zoals API endpoints en debug messages die verwijderd zouden worden.

5.7 COMPATIBILITEIT

Compatibiliteit in hybrid applicaties is zeer groot, maar door het gebruik van plugins wordt dit steeds kleiner.

iOS apps onder iOS 7 worden niet goedgekeurd door Apple, dus compatibiliteit voor iPhone en iPad begint bij iOS 7 tot momenteel iOS 9.X.

Na het testen op een 30-40 tal toestellen heb ik volgende minimum specificaties vastgelegd op basis van problemen en snelheid.

Android wordt ondersteund vanaf 4.4+
Alles apparaten onder 4.4 zijn niet snel genoeg
iOS wordt ondersteund vanaf 7.2+
Mogelijk kunnen er CSS fouten optreden doordat Safari niet up-to-date is.

5.7.1 Niet officieel

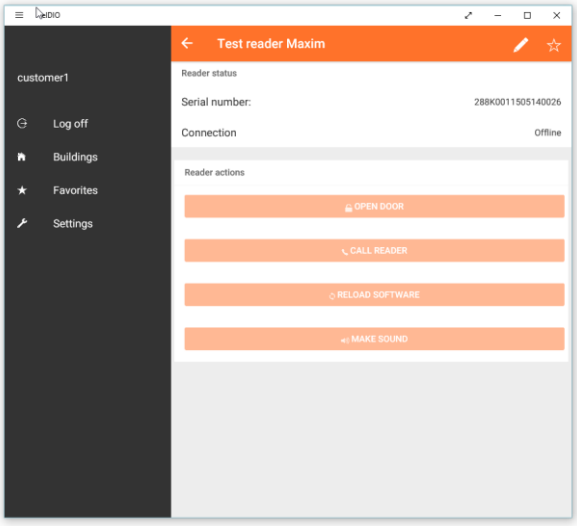
Cordova is in staat om een zeer uitgebreid aantal operating systems te ondersteunen. Het gebruik van plug-ins en bepaalde UI-elementen zorgt dat dit al snel minder uitgebreid wordt.

5.7.1.1 Windows Phone

Windows Phone wordt niet ondersteund omdat er geen notificaties worden getoond en ook omdat de lay out bij windows phone er niet goed uit ziet.

Van zodra OnsenUI uit de release candidates zijn vermoed ik dat de lay out problemen opgelost zullen zijn binnen 3-4 maanden.

5.7.1.2 Windows Desktop



De applicatie heeft de mogelijkheid om te draaien op Windows Desktop als een app. Dit wordt echter sterk afgeraden door Windows en is niet volgens het design patterns van Windows store apps.

Officieel ondersteund mijn applicatie dit dus niet, maar met enkel aanpassingen zou het in theorie ook werken op Windows.

Notificaties zou met Google's nieuwe FCM-technologie ook voor Windows apps kunnen worden geïmplementeerd.

5.7.2 Tablet

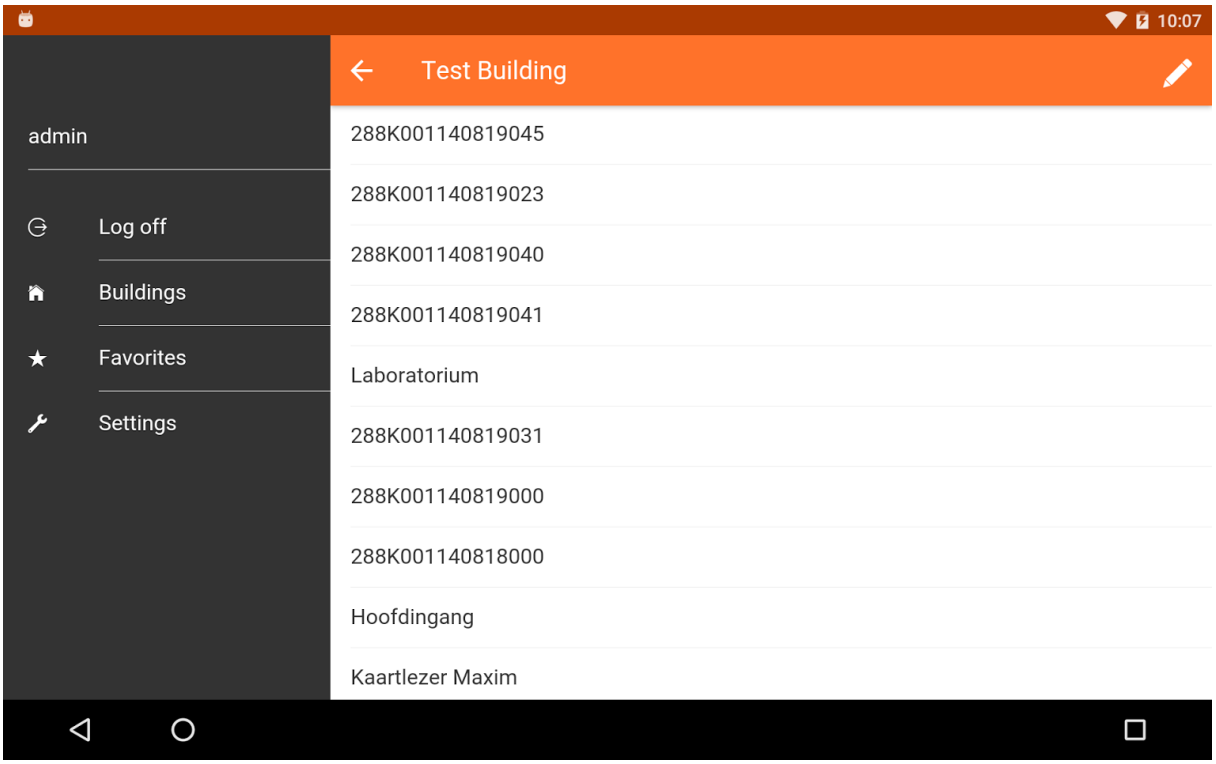
Bij OnsenUI 1 was het nog zeer lastig om de applicatie compatibel te maken voor tablets. De applicatie zoals ze nu is werkt volledig op tablets, enkel is de resolutie niet geoptimaliseerd op tablets.

OnsenUI 2 brengt handige functies om de applicatie voor tablets te maken, echter waren deze op het moment van schrijven tijdens de Bèta versie nog niet stabiel:

https://github.com/OnsenUI/OnsenUI/issues/1392

Bovendien, zoals reeds vermeld in de inleiding ben ik geen designer. Mogelijk heeft een designer een ander en beter inzicht in hoe het er zou moeten uitzien voor tablet weergave.

EDIT 28/05: De problemen met de splitter zijn inmiddels opgelost en ik heb het menu geoptimaliseerd voor tablets. Door het tijdsgebrek heb ik de lijsten van gebouwen en kaartlezers niet verder kunnen optimaliseren voor tablets.



De enige aanpassing tegenover de gewone applicatie is dat het menu niet ‘swipeable’ is en deze steeds in beeld staat.

Ook is het icoontje om de menu te tonen niet zichtbaar op de tabletversie. Indien er ooit een tablet gaat ingebouwd worden met de applicatie zou ik het volgende voorstellen:

- Applicatie in full screen (config.xml van project aanpassen)
- Auto startup on boot
- Avatars voor gebouwen of kaartlezers (icoontjes)

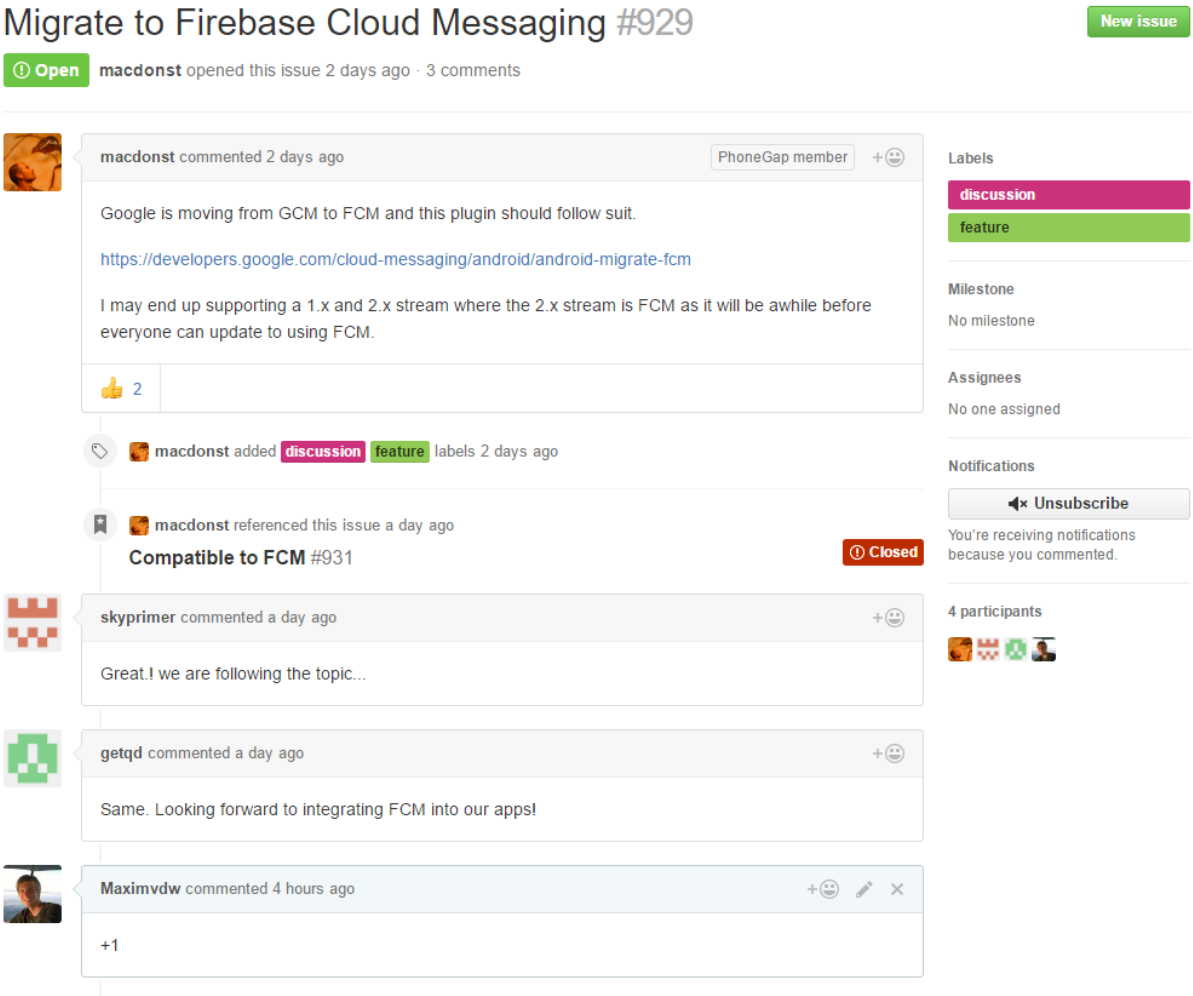
5.8 ONDERHOUDEN

Bij het kiezen van het framework wist ik dan OnsenUI een upgrade naar 2.0 ging krijgen (branch). OnsenUI 2.0 support zowel: Javascript, AngularJS1, Angular2 en ReactJS wat het dus zeer makkelijk maakt om in de toekomst te updaten. Angular2 is zeker iets om naar uit te kijken door zijn efficiëntie, al is het nog niet iets waar veel webdevelopers mee vertrouwd zijn mits het in een tussentaal wordt geschreven (typescript).

Verder is het enkel een kwestie van te zorgen dat alle plugins en Cordova up-to-date blijven. Zei zorgen er wel voor dat alles werkt op de laatste Android of iOS versie.

Het is voor Bitpower aangeraden om OnsenUI goed in het oog te houden, omdat deze nog steeds in “Release Candidate” was toen ik mijn eindwerk beëindigde.

Er is ook aangekondigd dat de push plugin een update zal krijgen naar FCM in versie 2.X.X en dit is zeker iets om naar uit te kijken want het gaat alles een stuk makkelijker maken.



6 OWASP

Onderaan vindt u de implementaties en veiligheidsmechanismen die ik gebruikt heb om rekening te houden met de OWASP Top 10 guidelines.

Let wel op dat dit 'guidelines' zijn en geen TODO lijst van zaken die moeten beveiligd worden.

Niet alle OWASP punten zijn hieronder vermeld omdat ze soms uit de scope van het project vallen of niet van toepassing zijn.

6.1 A1 Injection

In de backend heb ik managers gemaakt die gegevens in/uit de database halen. Deze worden beveiligd met prepared statements.

```php
include_once('DatabaseUtils.php');

class ReaderManager{
    // Database
    private $db = null;

    function __construct($host,$login,$pass,$db){
        $this->db = new mysqli($host,$login,$pass,$db);
        if ($this->db->connect_errno) {
            echo "Failed to connect to MySQL: (" . $this->db->connect_errno . ") " . $this->db
        }
        $this->db->set_charset("utf8");
    }

    /**
     * Get the reader by SN
     *
     * @param sn Serial Number
     **/
    function getReaderBySN($sn){
        $sql = "SELECT * FROM sensors WHERE deviceSN=?";
        $stmt = $this->db->prepare($sql);
        $stmt->bind_param('s', $sn);
        $stmt->execute();
        $result = get_result( $stmt );

        if (sizeof($result) == 0)
            return null;
        return $result[0];
    }

    /**
     * Get the reader by SN filtered
```

Voor mijn project moesten enkel volgende gegevens bewaard worden:

- Device UID (Notificaties)
- Alias van gebouw en kaartlezer

Deze vereiste echter geen validatie bijkomstige informatie omdat het device UID zeer hard kan verschillen van lengte en de alias in principe alles mag zijn dat kan weergegeven worden in UTF-8.

6.2 A2 Weak authentication and session management

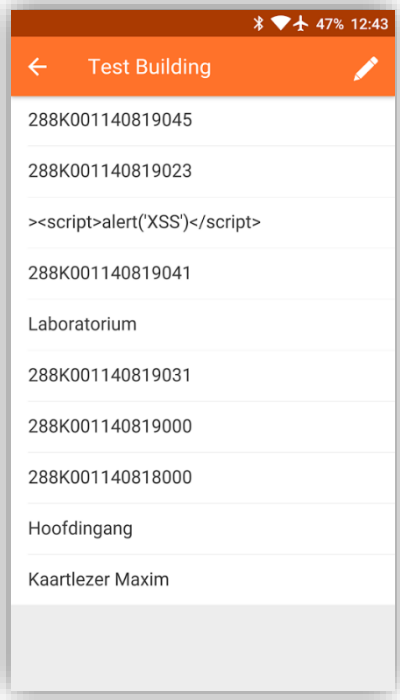
Voor elke request naar de API voor functies voor de applicatie zal de gebruiker via HTML Basic Authenticatie worden gevalideerd over SSL.

Daarna zal worden gecontroleerd of het gebouw/kaartlezer waarop een actie of zoekopdracht wordt uitgevoerd wel degelijk van de geauthentiseerde gebruiker is.

```
if ($action == "getreader"){
    // Get reader(s)
    if (isset($_GET['building_id'])){
        // Get readers by building id
        validate_user();
        $buildingID = $_GET['building_id'];
        validate_user_owns_building($buildingID);
        $output['readers'] = $readerManager->getReadersByBuilding($buildingID);
    }
```

6.3 A3 XSS

AngularJS zal steeds zorgen dat de HTML-applicatie tegen XSS kan door het gebruik van placeholders die intern door Angular sanitised worden.



6.4 A6 Sensitive Data Exposure

De API-calls zenden enkel data terug dat relevant is voor de gebruiker en dit steeds over een SSL-connectie. Intern zal voor de applicatie ook een optie voorzien zijn om de gebruikersdata te bewaren in een secure storage die deze met AES 256 zal encrypteren.

De private keys van de kaartlezers en gehashte wachtwoorden zullen nooit in een response zitten.

6.5 A8 Using components with known vulnerabilities

Voor alle plugins en libraries die gebruikt zijn in de scope van mijn project heb ik onderzocht naar beschikbare exploits.

Alle 3de partij componenten worden goed geüpdatet en worden ook onderhouden door respectabele ontwikkelaars of communities.

7 TODO'S VOOR BITPOWER

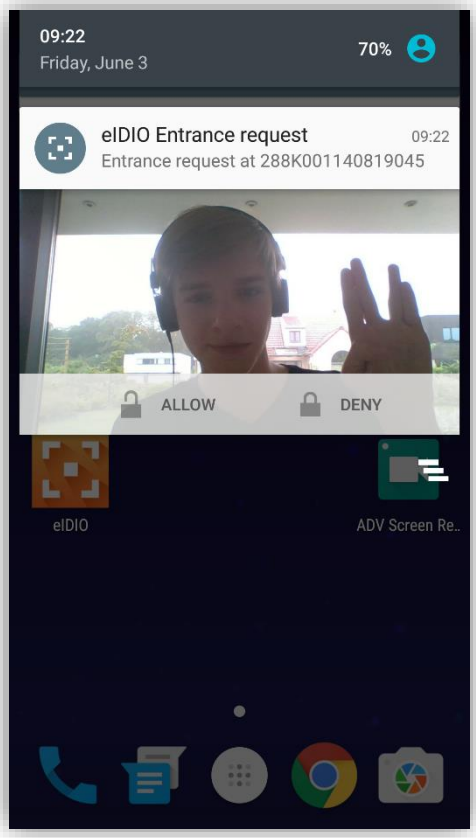
Dit is een lijst met Todo's voor Bitpower. Omdat ik voor mijn final work geen toegang tot de configuratie van servers had zijn deze voor het project op eigen servers geplaatst.

- Eigen linux server voor de websocket server. Momenteel draait de websocket server op http://eidio.mvdw-software.com/readeraction
  Dit moet naar een server van Bitpower verplaatst worden
  - De URL dient aangepast te worden in reader.php van de /api/ folder op de backend
  - De URL dient aangepast te worden in de kaartlezer op EIDIOWebsocket.c eventueel ook andere poorten. Momenteel is de URL voor de websocket server ingesteld op srv6.mvdw-software.com op poorten 2515 (Plain) en 2516 (SSL)
  - Wildly 9 (of 10) met 2GB RAM en een medium grade CPU zou volstaan voor de load die momenteel nodig is.
- Certificaten aanmaken voor eigen server en eigen SDK licenses
  - Apple certificaten dienen aangemaakt te worden voor testtoestellen en APS (99 euro voor developer license)
  - Er dient een nieuw JKS keystore met certificaten aangemaakt te worden voor de websocket server (gekocht SSL-certificaat aanbevolen maar niet noodzakelijk)
  - De servlet op de websocket server moet ook een certificaat krijgen
- Eigen Google App aanmaken op http://console.developers.google.com met GCM/FCM toegang.
  - Aanpassen van de GCM key in de backend op: /lib/NotificationUtils.php
  - Uploaden van eigen APS-certificaten naar GCM (zie 5.4.2.1)
- Kaartlezer werkende krijgen en code mergen
  - Zorgen dat de socket connecties goed verlopen
  - Eventueel het "pollen" van config updates langzaam beginnen te verplaatsen naar de websocket connectie (Update push).
- REST API verplaatsen naar een dedicated server (liefst een linux box waar de wildfly server mee op kan draaien)
- Werkstation installeren
  - NodeJS installeren: https://nodejs.org/en/
    OnsenUI, Phonegap en CSS tools zitten in deze package manager
  - Phonegap CLI installeren: http://docs.phonegap.com/references/phonegap-cli/install/
    Nodig voor het compilen en toevoegen van plugins
  - Ionic CLI installeren: http://ionicframework.com/docs/cli/
    Nodig voor het genereren van afbeeldingen met "ionic resources"
  - Android Studio installeren
    Nodig voor het compilen voor Android en debuggen
  - XCode op Mac (OSX)
    Nodig voor het compilen voor iOS en debuggen (Safari)
  - Installeren van Maven en een Java IDE
    Nodig voor het compilen en aanpassen van de websocket server
  - Installeren van Wildfly 10
    Lokaal testen van de websocket server

- Werkende krijgen van de applicatie op eigen Android toestellen.
  Een eerste stap na het installeren van het werkstation is om de huidige applicatie te compilen en runnen op een eigen android toestel
- Werkende krijgen van de applicatie op eigen iOS toestellen.
  Na het aanmaken van certificaten in Apple Developer Center is het mogelijk om te testen op eigen iOS toestellen.
  Testen voor iOS is moeilijker dan Android omdat problemen met certificaten vaak lastig te vinden zijn.
- Controleren op updates voor plug-ins en OnsenUI
  - Het is zeer belangrijk om na het final work te controleren voor updates. OnsenUI update regelmatig omdat ze in “Release candidates” zitten.
- Overschakelen naar Firebase Cloud Messaging zodra de push plugin dit toelaat
  Dit heeft momenteel nog geen prioriteit maar kan wijzigingen aan notificaties in de toekomst vergemakkelijken

7.1 Coole Ideeën

Dit zijn enkele ‘coole’ ideeën die ik had die momenteel zeer makkelijk te implementeren zijn. Ik heb ze niet geïmplementeerd omdat ik meer tijd wou steken in stabiliteit.



- Foto nemen van de persoon die binnen wil komen
  Het is mogelijk om de foto te tonen wanneer iemand op de deurbel drukt.
  - Op de kaartlezer dient een camera aangesproken te worden om een foto te nemen.
    Bitpower liet weten dat er ooit een camera ging worden geïmplementeerd. Deze zou hiervoor gebruikt kunnen worden.
  - De foto mee doorsturen met de HTTP request die wordt gebruikt om een belsignaal te sturen.
    - De foto mag niet te groot zijn (~100 Kb) om de latency zo klein mogelijk te houden
  - In de backend bij “reader-call.php” optioneel controleren op een image upload.
  - Vervolgens de foto op een publieke URL zetten (liefst met een unieke hash als naam en tijdelijk)
  - De foto URL mee doorsturen in het push bericht voor Android
    - Voor iOS of in-app notificaties kan de url als extra payload worden meegestuurd zodat deze kan weergegeven worden in de applicatie zelf.
- In de applicatie kunnen bekijken wie op welk moment binnengekomen is
- Bijhouden welke “entrance requests” er geweest zijn in de applicatie
- Optie om notificaties wel/niet te krijgen bij specifieke kaartlezers

8 BESLUIT EN EINDWOORD

Mijn doel was om uit te zoeken wat de snelste, stabielste en veiligste manier is om de bouwheer te laten weten dat iemand binnen wil komen om deze vervolgens toegang te geven.

Het grootste deel van mijn final work heb ik gespendeerd aan het zoeken van een manier om live met de kaartlezer te communiceren. Niet omdat het moeilijk was om een veilige oplossing te vinden, maar wel omdat het moeilijk was om een oplossing te vinden die stabiel was zowel bij de server kant als de kaartlezer kant.

Veiligheid was een belangrijke factor in het begin, ik was hier zeer hard op gefocust voordat ik de code van de backend in handen kreeg. Na het bekijken van de backend code besefte ik dat hier nog veel werk was en dus heb ik een grotere portie tijd dan voorzien gespendeerd om de API zo veilig mogelijk te krijgen. De beveiliging tussen de kaartlezer en socket server ging redelijk vlot, maar hier heb ik vooral aandacht gestoken om het zo performant mogelijk te krijgen met weinig resourceverbruik.

Wat ik vooral heb geleerd is dat hybride applicaties zeker een toekomst hebben, maar naar gelang het doel van de applicatie zou het soms beter zijn om een native applicatie te maken. Zolang je bezig bent met UI en simpele HTTP requests is een hybride applicatie perfect, maar ga je ook maar een klein beetje de kant van native functionaliteit op (zoals een simpele notificatie) dan word het ineens veel ingewikkelder om een hybride oplossing te vinden dan een native solution te ontwikkelen.

Mark Zuckerberg heeft ooit gezegd: "*Our biggest mistake was betting too much on HTML5 for mobile applications*"
source: http://techcrunch.com/2012/09/11/mark-zuckerberg-our-biggest-mistake-with-mobile-was-betting-too-much-on-html5/

Hij heeft gelijk, als je echt iets wil maken dat geoptimaliseerd is voor het toestel waarop het draait en je een budget hebt zoals Facebook dan is het inderdaad slim om native te werken. In een wereld waar we 100de of zelfs 1000de toestellen moeten ondersteunen is het niet altijd haalbaar om native te werken en daarvoor zijn hybride applicaties perfect.

CTO van Facebook Bret Taylor zei een jaar eerder: "*The popularity of mobile devices will change," he said implying that the dominant devices today might not be so dominant in the future. And if that's the case, why should Facebook dump resources into them? Wouldn't it be easier if they just focused on HTML5 — something which will work on an increasing number of devices going forward? Of course*."

Ik ben er zeker van dat ik in de toekomst nog hybride applicaties ga maken: het is goedkoper, het is simpel en je kan webdesigners gebruiken om de applicatie naar je noden te zetten.

Meer en meer bedrijven beginnen frameworks te maken die cross platform zijn. Zo heeft Google aan het einde van mijn final work FCM voorgesteld (Firebase Cloud Messaging) dat toelaat om makkelijk en 'unified' push notificaties te implementeren in zowel iOS als Android.

8.1 ZELFEVALUATIE

Tijdens mijn final work heb ik met veel interesse naar oplossingen voor mijn probleemstellingen gezocht. Al snel had ik vóór de deadline van Januari een groot deel van de applicatie af voor Android.

Mijn onderzoek heb ik begonnen bij hybride frameworks voor mobiele applicaties omdat dit een basis vormde van heel de scope van de opdracht. De argumenteringen die ik heb kunnen maken bij de keuze van OnsenUI zijn zeer goed. Ook het aantal frameworks dat ik getest heb is zeer uitgebreid. Ondanks de positieve commentaar tijdens mijn tussentijdse evaluatie vind ik toch dat ik iets te veel tijd aan dit onderzoek heb gespendeerd omdat ik vaak twijfelde of ik wel de juiste keuze heb gemaakt.

De opmerking bij mijn tussentijdse evaluatie was dat ik te zelfstandig werkte. Ik heb dit proberen op te lossen door meer advies te vragen bij kennissen die goed zijn in wat ze doen.

Waarschijnlijk zou ik dit in de toekomst nog kunnen verbeteren, maar ik vond dat voor dit type opdracht er niet meer hulp nodig was dan dat ik momenteel vroeg.

Ondanks mijn stage die in van Februari tot eind Mei duurde, heb ik nog vaak aan de applicatie kunnen werken. OnsenUI 2.0 was uitgekomen wat ervoor zorgde dat ik per 'beta' release of 'release candidate' de nieuwe features/bugfixes kon implementeren.

In het begin van het jaar moesten we een inschatting van alle features maken, hoewel ik hier goed op schema was heb ik toch zeer veel tijd verloren met het implementeren van de notificaties in iOS. Maar het feit dat dit mijn deadlines niet in gedrang bracht bewijst dat mijn planning zeer goed was.

Wel moet ik mijn limiet in het aantal werk dat ik om mij neem beter proberen inschatten in de toekomst. Vaak zak ik tot diep in de nacht bezig terwijl ik al vroeg om 6 uur moest wakker worden. Het zorgde ervoor dat mijn bachelor proef goed vooruitging maar was niet gezond.

Technisch vind ik dat de opdracht zeer goed gelukt is. De doelstellingen zijn behaald en het resultaat is een Android en iOS applicatie die release waardig zijn.

Sterke punten:

- Zelfstandig tot een eindresultaat komen
- Doorgronde analyse van alle mogelijkheden
- Technisch inzicht in complexe systemen
- Onderzoeken zeer uitgebreid met goede argumenteringen

Zwakke punten:

- Ik ben niet iemand die snel aan iedereen advies gaat vragen
- Limiteren hoelang ik aan een opdracht werk per dag