2016

Thema Opdracht 6 Technische Informatica

CALGON

Joost Wagensveld-van Veen	1664713
Zehna van den Berg	1662506
Jessy Visch	1661709
Koen de Groot	1638079

Samenvatting

In dit document zullen alle onderdelen van het proces die wij als groep doorlopen hebben besproken worden. Er zal uitleg gegeven worden over de verschillende modellen die gebruikt zijn in het plannen en organiseren van het project. Er zal globaal ingegaan worden op de *Requirements architectuur*, de *solution architectuur* en de werking van de code. Deze documenten zullen als bijlagen bijgevoegd worden, ze zullen echter niet in detail behandeld worden in dit document.

De conclusie en de aanbeveling zullen het verslag afsluiten. In het hoofdstuk *Aanbevelingen* zal besproken worden welke punten belangrijk zijn voor een volgend team om verder te kunnen gaan. Dit zijn onderdelen die tijdens die project niet zijn gelukt en punten die tijdens het project aan het licht zijn gekomen. In de conclusie zal terug gekeken worden op het hele proces. Wat goed en wat fout ging en hoe het in het vervolg beter zou kunnen.

Inhoudsopgave

1. Inleiding

Dit project is begonnen vanuit de Hogeschool Utrecht. Het is een project dat zich richt op domotica. In het geval van dit project is het de bedoeling dat er een webinterface gemaakt wordt voor een wasmachine. Doelstelling is dan ook deze interface alle functionaliteit te laten hebben die normaal vanaf de wasmachine zelf beschikbaar zou zijn geweest.

Omdat er steeds meer mensen op smartphones en tablets werken zal hier ook rekening mee gehouden moeten worden. Bepaalde onderzoeken zullen dan ook beschreven worden en uitgevoerd. De verschillende diagrammen en tabellen die voor dit project zijn gemaakt zullen kort worden besproken en samengevat. Verder zullen de conclusies en aanbevelingen het verslag afsluiten.

2. De wasmachine

In het volgende hoofdstuk zal er gekeken worden naar de eigenschapen en onderdelen van de wasmachine.

2.1 De opdracht

De opdracht die afkomstig is van Swirl Industries is als volgt: “Ontwikkel software en een wasmachine.

Deze wasmachine moet op een voor de gebruiker eenvoudige manier bestuurbaar zijn over het internet. Hiervoor moest er gebruik gemaakt worden van een web browser. Vanaf deze browser moest het vervolgens mogelijk zijn om een wasprogramma te selecteren. Deze wasprogramma’s werden mee geleverd op de wasmachine. Wel moest het mogelijk zijn dat de gebruiker de tempratuur aan kon passen naar wat wenselijk was voor de was die zij wilde draaien. De gebruiker zou ook aan moeten kunnen geven dat de was om een bepaalde tijd moest starten.

De laatste optionele wens van Swirl Industries is voor deze wasmachine was dat er logs bijgehouden zou worden. Van deze logs zouden er 2 versies bestaan. Hier later meer over. Verder moet de wasmachine natuurlijk alles kunnen dat een gewone machine kan.

2.2 Bediening

De bediening van de wasmachine wordt gedaan via het internet. Hier voor is er met HTML, CSS en javascript een webpagina gemaakt. Deze pagina voorzien van een gebruiksvriendelijke vormgeving en makkelijk te gebruiken bediening, zorgt er voor dat de gebruiker op eenvoud wijze het gewenste wasprogramma kan starten. Wij hebben qua vormgeving gekozen voor grote knoppen en niet overdreven veel informatie in een scherm. Dit vooral met het oog op het gebruik van tablet en smartphones.

2.3 Communicatie van de web browser en wasmachine

Communicatie gaat zoals het woord al zegt 2 kanten op. De wasmachine kan dus niet alleen berichten ontvangen van af het web maar hij kan ze ook versturen. Het is hierdoor mogelijk om de gebruiker te voorzien van meer informatie dan bij een normale wasmachine. Op deze manier kan de gebruiker in het venster waar de was gestart wordt informatie zien. Informatie die bijvoorbeeld te zien is:

- Deur status (is de deur open of dicht);
- Deur vergrendeling (zit de deur op slot? ja/nee);
- Waterniveau (voor hoeveel procent is de trommel gevuld met water);
- Watertemperatuur (wat is de tempratuur van het water);
- Resterende tijd (wat de geschatte resterende tijd van het wasprogramma is).

Dankzij deze informatie kan de gebruiker een goed overzicht behouden wat de wasmachine doet.

2.4 Log systeem

Het log systeem bestaat uit 2 verschilden logs die bijgehouden worden.

Een systeem dat alleen maar de hoofd activiteiten van de wasmachine bijhoudt. Deze is bedoeld voor de gebruiker. De gebruiker kan bijvoorbeeld in dit systeem opzoeken wanneer welke was gedraaid was en hoe lang dat geduurd heeft.

Een ander systeem dat alle processen en (fout-)meldingen bijhoudt. Dit gedeelte is alleen bedoeld voor onderhoudsmonteurs die aan de wasmachine moeten werken. In deze log files worden alle stappen die de wasmachine doorloopt bijgehouden. Stap voor stap wordt er opgeslagen wat hoe laat gestart is en hoe lang dit geduurd heeft. Deze log files zijn niet bedoeld voor de gebruiker maar voor onderhoud/reparatie van het apparaat. Zo moet het bijvoorbeeld mogelijk zijn dat een reparateur kan zien dat het verwarmingselement niet goed werkt. Dit zou een monteur kunnen zien doordat uit de gegevens blijkt het verwarmen van het water steeds langer duurt.

3. Ontwikkelomgeving

Een ontwikkelomgeving is alles van de computersoftware en de hulpmiddelen die de ontwikkelaar ondersteunen bij het ontwikkelen van de software.

De aspecten van een ontwikkelomgeving hangen af van de taken die moeten worden uitgevoerd. Er kan een onderscheid gemaakt worden tussen het ontwerpen, programmeren en het testen van de software.

3.1 Ontwerpen

Bij het ontwerpen zal de omgeving vooral bestaan uit modelleringgereedschappen en programma's waarmee je modellen kan maken en aanpassen.

De modelleringgereedschappen bieden ondersteuning in het maken van bijvoorbeeld een Unified Modeling Language, klassendiagram en dergelijke. Deze worden ontworpen met behulp van het modellering programma *Software Ideas Modeler*.

3.2 Programmeren

Om te programmeren werken wij op verschillende operating systems, *Mac*, *Debian* en *Windows*. Hierdoor gebruiken we ook verschillende IDE's. De IDE's die worden gebruikt zijn *notepad++*, *sublime text* en *xCode*. Hiernaast wordt gebruik gemaakt van het programma *Make* om onze applicatie eenvoudig te builden.

3.3 Testen

Bij het testen wordt de ontwikkelde software getest. Voldoet de test aan de eisen dan wordt deze versie opgeslagen. Is dit niet het geval dan wordt de code aangepast en weer getest. Dit wordt herhaald totdat het geteste item naar behoren werkt.

Ook wordt de code door meerdere teamleden bekeken en getest. Dit om er voor te zorgen dat de code niet alleen goed werkt maar ook volgens de afspraken is opgemaakt.

4. Requirement Architecture

In dit hoofdstuk zal Requirement Architecture kort worden behandeld. Er zal dus ingegaan worden op wat de software moet gaan doen. Zoals de naam al doet vermoeden wordt het *Requirements Architecture* gemaakt om vast te kunnen stellen wat er nodig gaat zijn om een goed werkend systeem te maken. Hierbij word gebruik gemaakt van meerdere diagrammen namelijk:

- *Usecase Diagram*
- *Activity Diagram*
- *Constraints Diagram*.

Gezien de grootte van het Requirement Architecture document is er voor gekozen deze niet als bijlage toe te voegen. In plaats daarvan zal er worden verwezen naar het document (Wagensveld, van den Berg, Visch, & de Groot, 2016a).

4.1 Usecase diagram

De *Usecase* is bedoeld om een beeld te kunnen vormen over hoe de ideale gebruiker gebruik maakt van het product. We hebben het hier alleen over de ideale gebruiker omdat wanneer je een product maakt je nooit alles kan voorspellen in wat een gebruiker zou kunnen proberen. Natuurlijk moet er bij het schrijven van de code voor gezorgd worden dat onvoorzien gedrag van een gebruiker niet het hele systeem omgooit. Maar dit is niet het doel dat bereikt moet worden met een *Usecase Diagram*.

Usecases worden gebruikt om snel duidelijk te maken wat de gebruiker richting het systeem kan doen en wat het systeem richting de gebruiker moet doen. Voor de wasmachine hebben we het dan over: Wasprogramma uitvoeren, Wasprogramma Stoppen, Update aanbieden, Activiteiten logs weergeven, System logs weergeven.

Wasprogramma uitvoeren:

Het kunnen selecteren van het gewenste programma, wat vervolgens door het systeem uit gevoerd moet kunnen worden.

Wasprogramma stoppen:

Het stoppen van het huidige wasprogramma dat uitgevoerd wordt.

Update aanbieden:

Het aanbieden van een update voor het systeem en of de wasprogramma's die de gebruiker tot zijn beschikking heeft.

Activiteiten logs weergeven:

Toont een log met alle activiteiten die de wasmachine heeft uitgevoerd. Zoals eerder genoemd is dit log alleen bedoeld om de gebruiker te kunnen vertellen hoe veel wassen er gedraaid zijn en om hoe laat welke was gestart is.

Systeem logs weergeven:

Toont in groter detail wat het systeem gedaan heeft en hoe lang het wat heeft gedaan. Denk bijvoorbeeld aan het opwarmen van het water of het vullen van de trommel met water.

4.2 Activity diagram

Het *Activity Diagram* is bedoeld om een overzicht te geven van wat het systeem doet van begin tot einde. Deze "tekening" is nog niet een uitwerking van hoe het systeem (de wasmachine) echt gaat

werken. Het is meer een globaal overzicht van welke stappen het systeem moet doorlopen om tot het gewenste eindresultaat te komen. Er wordt hier dus gekeken naar het aanzetten van de machine en wat hij moet doorlopen om met succes een was te draaien. Hoe lang welke stap moet duren is daarentegen iets dat op dit moment minder belangrijk is.

4.3 Constraints diagram

Dit diagram wordt niet zo zeer gebruikt om vast te stellen wat het systeem moet doen. Dat wil niet zeggen dat het hier niet ook voor gebruikt kan worden. Wat dit diagram voornamelijk zo belangrijk maakt is dat het een beeld geeft aan welke eisen het systeem moet vol doen (hardware en software). Bijvoorbeeld het meten en weergeven van de temperatuur van het water in de trommel mag niet al te veel afwijken. Het is voor het draaien van een was natuurlijk ook niet nodig dat ik op een honderdste graad precies kan meten. Maar het is ook zeker niet de bedoeling dat de sensor 20 graden meet terwijl het eigenlijk 120 zou moeten zijn. Dit is natuurlijk niet het meest realistisch voorbeeld en zal in deze extremen niet snel voorkomen. Het zijn wel dingen waar voordat je begint over na gedacht moet worden. Zeker als het gaat over dingen die je alleen maar kunt weten wanneer je dit specifiek test. Zo is natuurlijk wel een probleem wanneer je geen rekening houdt met het feit dat je microprocessor te klein is voor de software die je geschreven hebt.

5. Solution Architectuur

Het Solution architecture wordt gemaakt om een goed en overzichtelijk beeld te geven over wat de beoogde werking is van de software. Hoe is deze ingedeeld en hoe communiceert die met de andere onderdelen. Ook wordt hier gekeken naar wat de prioriteit is van welke onderdelen. Om al deze vragen te beantwoorden wordt er gebruik gemaakt van een aantal diagrammen. Het zo genoemde klassendiagram, het *concurrency diagram* en het *state transition diagram*. Verder word er ook een taakstructurering en andere nodige protocollen gemaakt.

Gezien de grootte van het Solution Architecture document is er voor gekozen deze niet als bijlage toe te voegen. In plaats daarvan zal er worden verwezen naar het document (Wagensveld, van den Berg, Visch, & de Groot, 2016b).

5.1 Klassendiagram

Het *klassendiagram* wordt gebruik om een overzicht te geven van alle onderdelen van de software. De “klassen” zijn in dit geval de verschillende objecten die nodig zijn om de software te laten werken.

In dit diagram is dus een overzicht te zien van hoe de verschillende objecten opgemaakt zijn, welke functionaliteit deze hebben en hoe ze met de andere objecten kunnen praten. Zo is hier ook in terug te zien dat niet alle klassen (objecten) met elkaar verbonden zijn.

Dit is voor de programmeurs handig omdat ze dan snel kunnen terug zien hoe de verschillende onderdelen zijn opgebouwd. Zeker als er meerdere programmeurs tegelijk aan onderdelen werken zorgt dit er voor dat ze van tevoren al weten wat er ongeveer verwacht kan worden betreffende andere klassen.

5.2 Concurrency diagram

Het *concurrency diagram* dient er voor om een beeld te geven over hoe bepaalde onderdelen berichten over en weer sturen naar mekaar. Dit zorgt er voor dat de programmeurs een duidelijk beeld kunnen vormen van hoe bepaalde onderdelen van de code de informatie waar zij mee moeten werken kunnen krijgen en zo nodig hoe zij deze terug sturen naar diegene die er om gevraagd heeft.

5.3 State Transition Diagram

Het *state transition diagram* is een diagram dat in detail afbeeldt in wat voor stadiums het programma terecht kan komen. Met behulp van dit diagram kan er dus goed uitgebeeld worden wat de weg is die het programma kan nemen om zijn nodige stappen door te lopen. Het geeft ook gelijk weer welke keuzen en/of afwegingen het programma maakt terwijl het van stadium naar stadium gaat.

5.4 Taalstructurering

De toolstructurering is een overzicht van alle taken die de verschillende objecten hebben. Hier wordt gelijk gekeken naar de prioriteit van deze taken. De prioriteit wordt beïnvloed door verschillende onderdelen. Het maakt namelijk een hoop uit hoe vaak een bepaalde taak uitgevoerd moet worden en wat de deadline is waar binnen deze taak klaar moet zijn met uitvoeren.

5.5 Communicatie protocol webserver websocket

Het communicatie protocol tussen deze onderdelen zorgt voor een structuur waar de programmeurs snel aan kunnen zien hoe of wat ze zouden moeten versturen/ontvangen wanneer ze iets van of naar

de webserver sturen. Door dit protocol op te stellen zorg je er voor dat op alle plekken in de code dit op soort gelijke manier gedaan kan/moet worden.

6. Werking van de code

De software is geschreven in C++ aan de hand van het klassendiagram. Dit klassendiagram is bijgevoegd in de bijlagen. De software is op te delen van verschillende onderdelen. De sensors, het wasprogramma uitvoeren en de webserver.

De sensoren:

Door middel van de SensorHandler wordt er gezorgd dat de sensoren beschikken over de meest recente data. Dit doet hij simpelweg door eens in de halve seconde alle sensoren de opdracht te geven nieuwe data op te vragen en deze te bewaren. Aan de sensoren kan vervolgens gevraagd worden wat de data is die zij hebben. Deze informatie kan dan verder in de software gebruikt worden om te bepalen wat de volgende stap moet zijn.

De webserver:

De webserver is verantwoordelijk voor de communicatie tussen het web en de gebruiker. Dit doet hij door middel van een websocket.

Wanneer de gebruiker via de browser naar de wasmachine gaat krijgt hij daar de keuze in wat hij wil starten en op welke manier. Deze gegevens worden door gegeven aan de wasmachine door middel van een string. Via bijna dezelfde weg wordt er ook data van af de wasmachine naar de web browser gestuurd. De gegevens die binnen komen worden vervolgens in de UI weergegeven en zo word de gebruiker up-to-date gehouden met wat de machine doet.

Het wasprogramma:

Dit stuk code is verantwoordelijk voor het uitvoeren van het juiste wasprogramma. Zoals hier boven gezegd komt van af de webserver een string, deze string wordt door de WasprogrammaControler ingelezen. Het inlezen van welk programma er is gekozen vertelt aan de WasprogrammaControler welk tekstbestand ingeladen moet worden. Door het inladen van het tekstbestand krijgt de controller de precieze gegevens die het wasprogramma vormen. Deze gegevens worden vervolgens in de juiste volgorde naar de wasmachine gestuurd die ze een voor een uitvoert.

7. Testen

Er zijn tijdens het project meerdere tests/onderzoeken gedaan om tot sommige beslissingen te kunnen komen of om kennis op te doen over de werking van onderdelen.

Zo zijn de volgende onderzoeken gedaan:

- Welke webserver software pakket het meeste pasten bij onze wens.
- Hoe de Linux queue functioneert?
- Hoe een wasprogramma er uit ziet?
- Hoe de Raspberry pi zijn IP-adres bekend kan maken?
- Hoe RTOS functioneert?
- Hoe de wasmachine emulator werkt?

In dit hoofdstuk zullen al deze onderzoeken aanbod komen. Er zal per onderzoek een korte samenvatting gegeven worden wat de eindconclusie is en waarom dit is besloten.

7.1 Welk webserver software pakker gaan we gebruiken?

In dit onderzoek hebben we gekeken naar 2 verschillende software pakketten voor webservers Apache en Nginx. Beiden software pakketten heeft zo zijn voordelen en nadelen. Zo is Nginx over het algemeen sneller. Er wordt namelijk niet voor iedere verbinding een nieuwe proces gestart. Dit zorgt er ook voor dat hij minder geheugen gebruikt van de Raspberry pi. Daarentegen is de mogelijkheden en documentatie van Nginx minder uitgebreid.

Hoewel Nginx sneller is en minder geheugen verbruikt is er als team toch gekozen voor apache. Deze keuze is genomen omdat de snelheid van Nginx voor onze doeleinden niet van belang was. Je zou namelijk echt pas wat van deze snelheid merken wanneer er meerdere verbindingen tegelijk open staan. Dit is bij de wasmachine niet het geval, er zal namelijk altijd alleen maar een verbinding zijn van de server naar het web zijn en van de server naar de wasmachine. Ook de uitgebreidere documentatie en de al bestaande kennis binnen het team maakte apache de beste keuze voor ons(Ellingwood, 2015; The_Apache_Software_Foundation, 2015).

7.2 Hoe functioneert de Linux queue?

Tijdens het uitvoeren van dit onderzoek kon al snel worden geconcludeerd dat de Linux queue veel ingewikkelder was dan voor ons nodig. Hierom veranderde de richting van het onderzoek naar wat een betere optie was om ons doel te bereiken. We zijn toen uitgekomen bij de STD::queue. Deze queue is onderdeel van de STD library. Omdat we deze library toch al gebruikte was het natuurlijk een logische keuze. De queue werkt op het First in First out principe. De andere bewerkingen vallen in de zelfde vorm en logica als andere onderdelen van de library. Dit zorgde ervoor dat het makkelijk te begrijpen was en snel te implementeren(die.net, 2015; Unknown, 2015).

7.3 Hoe ziet een wasprogramma er uit?

Het doel van dit onderzoek was om een duidelijk beeld te krijgen van wat er kwam kijken tijdens het uitvoeren van een wasprogramma. Welke stappen loopt hij door en hoe lang duren deze stappen ongeveer. Dit moest er ook voor zorgen dat de verschillen tussen de wasprogramma's duidelijk werden(Diepstraten, 2016).

7.4 Hoe kan de Raspberry pi zijn IP-adres bekend maken?

Er zijn verschillende manieren om dit te kunnen doen. De simpelste manier zou zijn om een seriële kabel aan te sluiten en via die verbinding het aan de Raspberry pi te vragen. Maar dat moet je elke keer weer met die kabel en aansluiting rommelen. Na wat onderzoek op internet vonden we een Python script die als deze in het opstarten van de pi wordt uit gevoerd een e-mail stuurt met daar in zijn IP-adres. Dit was voor ons de beste oplossing geen gerommel met kabels en andere verbindingen. De Raspberry pi vertelde het ons gewoon zelf terwijl wij ons op andere onderdelen richten(Raspberry_Pi_Foundation, 2015).

7.5 Hoe functioneert RTOS en hoe functioneert de wasmachine emulator?

De wasmachine emulator en het RTOS zijn allebei zaken die door de Hogeschool Utrecht zijn verschaft. De Hogeschool verschaft een hoop documentatie hierover(Wensink et al., 2015). Hierdoor is er niet echt onderzoek nodig hoe dit werkt. Het is echter wel goed dat al deze informatie goed bekend is, daarom zijn deze twee vragen toch opgenomen als literatuuronderzoek.

8. Conclusie

In dit hoofdstuk worden alle onderdelen van het project kort besproken. Er wordt een korte samenvatting gegeven van de testen/onderzoeken die gedaan zijn. Verder wordt er een conclusie getrokken over het behalen van de doelstellingen van het project.

De opdracht die tot dit project geleid heeft is het verzoek vanuit Swirl Industries voor het maken van een wasmachine-webinterface. In de eerste weken van het project is een interview afgenomen bij Jan Swirl van Swirl Industries. Uit dit interview zijn alle eisen aan het systeem duidelijk geworden. In dit document wordt uitleg gegeven over de verschillende keuzes die genomen zijn bij het maken van de verschillende diagrammen. Deze keuzes hebben geleid tot het formuleren verschillende *must- & should-haves*. Aan de hand daarvan zijn de *Requirement Architecture* en *Solution Architecture* gemaakt. In het Solution Architecture is ook het klassendiagram opgenomen. Dit Klassendiagram is op zijn beurt verantwoordelijk voor de opbouw van de klassen.

Een onderdeel van het project is het doen van onderzoeken. Zo is er onder andere onderzoek gedaan naar welke softwarepakketten het beste gebruikt kunnen worden. Er is onderzoek gedaan naar het verschil tussen de Linux queue en STD::queue. Hieruit is gebleken dat in ons geval de STD::queue het beste werkt. Een ander onderzoek is gedaan naar hoe een wasprogramma er uit ziet. Ook zijn een aantal literatuuronderzoeken gedaan om zaken als RTOS, de wasmachine emulator en het IP-adres van de Raspberry pi duidelijk te krijgen.

Doelstelling van dit project was het creëren van een werkende webinterface voor een wasmachine. Dit doel is behaald, hoewel het uiterlijk van de webpagina nog wat te wensen overlaat is aan alle functionele eisen voldaan.

9. Evaluatie

Over het algemeen is het project goed verlopen. In het begin was het even lastig om goed een duidelijk beeld te krijgen bij wat de opdracht nu precies inhield. Dit maakte het voorbereiden van het interview wat lastiger en het inplannen van de taken. Na het interview kwam hier wat meer helderheid voor ons in en konden we de planning invullen.

De samenwerking van het team begon even wat lastig maar dat hoort een beetje bij het begin van een project iedereen moet even een plekje zoeken. Dit loste zichzelf dan ook snel op. Het enige dat het soms een beetje lastig maakte was de communicatie. Het was zeker niet het geval dat deze slecht was. Het ging alleen op sommige momenten wat slechter. Zeker in de project week wilde iedereen gelijk aan de slag gaan en kostte het soms even wat moeite om een vergadering gaande te krijgen. De vergaderingen waren soms dan ook kort en bondig en niet uitgebreid genoeg.

Zeker in de tweede week toen alles richting een einde gebracht moest worden zorgde dit voor wat verwarring. Het was altijd wel snel op te helderen maar iets meer tijd in de vergadering zou het kunnen hebben voorkomen. Dit is dan ook zeker iets dat mee genomen zal worden voor volgende projecten.

Gelukkig is de communicatie voor zover het echt een probleem was ook het ergste dat fout ging geweest. Zo was eigenlijk iedereen elke dag op de afgesproken tijd aanwezig en was dit niet het geval dan werd dat ruim op tijd gemeld.

Zo kan dus met een goede blik terug gekeken worden. Want ook al waren er soms wat problemen, dit heeft nooit geleid tot een onprettige samenwerking. We konden het dan ook altijd op een goede manier oplossen.

10. Suggesties en aanbevelingen

In dit hoofdstuk zullen de suggesties en aanbevelingen aan bod komen. Deze aanbevelingen zijn bedoeld als overdracht naar een team dat op dit project verder gaat.

Na het succesvol afronden van het proof of concept zijn er altijd verbeterpunten die het geheel nog kan verbeteren. Hier zijn aan het begin van het project al enkelen *should haves* en *could haves* voor bedacht. Als team zijn wij hier niet aan toe gekomen. Wel denken we nog dat deze punten de wasmachine en gebruiksvriendelijkheid kunnen verbeteren.

De should haves die wij bedacht hebben maar geen tijd meer voor hadden zijn:

- Kleine knop “extra opties”
  De knop zou ergens onder in het scherm geplaatst kunnen worden. Wanneer de gebruiker op de knop drukt kan er een nieuwe pagina of nieuw scherm geopend kunnen worden. Dit scherm kan dan alle niet essentiële informatie beschikbaar stellen voor de gebruiker.
- Info over gebruiksgeschiedenis
  Dit onderdeel zou ook zichtbaar gemaakt kunnen worden door middel van de “extra opties” knop. Hier zou dan globaal weergegeven kunnen worden wat het gebruik is van de wasmachine.
- Logs bijhouden
  De logs die hier bijgehouden zouden worden zijn gedetailleerder dan de gebruiksgeschiedenis. Het doel van deze logs is de monteur van nauwkeurige informatie voorzien over het gebruik van de wasmachine. Op deze manier kan bij onderhoud makkelijker bepaald worden hoe veel en hoe lang een onderdeel gebruikt is.
- Stoppen/pauzeren wasprogramma
  In het interview met de opdrachtgever is dit als mogelijkheid naar voren gekomen. Hoewel het stoppen van de was handig kan zijn is het pauzeren een stuk minder praktisch. Het is niet altijd goed voor de was als een wasprogramma gepauzeerd wordt. Zo zou de was in een plas stilstaand water kunnen blijven liggen en het water langzaam afkoelen. Wanneer het programma dan weer hervat wordt is lastig te voorspellen in welke staat de was zich bevindt. In dat geval is het beter het programma af te breken en weer van af het begin te laten beginnen. Om deze reden denken wij dat het beter is gewoon de was geheel te stoppen en opnieuw te starten wanneer dat wenselijk is.

Deze 4 punten hadden wij er graag ingebracht als hier tijd voor was geweest. Natuurlijk zijn er nog meer dingen die onderzocht zouden kunnen worden. Zo zou een *user test* opzetten kunnen worden om te kijken welke onderdelen van de UI wel en niet goed werken voor de doelgroep. Of welke onderdelen de doelgroep er nog graag bij zou willen zien.

Het is echter wel van belang dat het geheel niet te ingewikkeld wordt gemaakt voor de gebruiker en ontwikkelaar. Het is goed en leuk met de mogelijkheid van het internet te spelen. Meer mogelijkheid hebben tot betekent niet dat het ook allemaal handig is om te doen.

11.Appendices

I. Referenties

die.net. (2015). Linux Documentation. from http://linux.die.net/

Diepstraten, Z. (2016). Hoe werkt de wasmachine. from http://www.zdiepstraten.nl/index.php/keuze-in-witgoedmenu-hoe/hoe-werkt-de-wasmachine

Ellingwood, J. (2015). Apache vs Nginx: practical Considerations. from https://www.digitalocean.com/community/tutorials/apache-vs-nginx-practical-considerations

Raspberry_Pi_Foundation. (2015). Raspberry Pi Documentation. Retrieved 26-11-2015, from https://www.raspberrypi.org/documentation/

The_Apache_Software_Foundation. (2015). The Apache Software Foundation. from http://www.apache.org/

Unknown. (2015). queue. 2016, from http://www.cplusplus.com/reference/queue/queue/

Wagensveld, J., van den Berg, Z., Visch, J., & de Groot, K. (2016a). Requirement Architecture (pp. 7). Utrecht: University of Applied Sciences Utrecht.

Wagensveld, J., van den Berg, Z., Visch, J., & de Groot, K. (2016b). Solution Architecture (pp. 8). Utrecht: University of Applied Sciences Utrecht.

Wensink, M., Zuurbier, J., van Ooijen, W., Ovink, G., Schalken-Pinkster, J., & van Doesburg, A. (2015). SharePoint. Retrieved 9-11-2015, from https://cursussen.sharepoint.hu.nl/fnt/35/TCTI-V2THO6-14/

II. Klassendiagram