calgon | Nijenoord 1

calgon

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

Thema Opdracht 6 Technische Informatica

2016

# Inhoudsopgave

[Inhoudsopgave 2](#_Toc440552111)

[1. Inleiding 4](#_Toc440552112)

[2. De wasmachine 5](#_Toc440552113)

[2.1 Algemeen 5](#_Toc440552114)

[2.1.1 De opdracht 5](#_Toc440552115)

[2.1.2 Bediening 5](#_Toc440552116)

[2.1.3 Communicatie van de web browser en wasmachine 5](#_Toc440552117)

[2.1.4 Uitvoering nodige stappen wasmachine 6](#_Toc440552118)

[2.1.5 Log systeem 6](#_Toc440552119)

[2.2 Screenshots van de webinterface 6](#_Toc440552120)

[3. Ontwikkelomgeving 7](#_Toc440552121)

[3.1 Ontwerpen 7](#_Toc440552122)

[3.2 Programmeren 7](#_Toc440552123)

[3.3 Testen 7](#_Toc440552124)

[4. klassendiagram 8](#_Toc440552125)

[5. Toelichting klassendiagram 9](#_Toc440552126)

[6. Beschrijving verantwoordelijkheid klassen 9](#_Toc440552127)

[6.1 Wasprogrammacontroller 9](#_Toc440552128)

[6.2 SensorHandler 9](#_Toc440552129)

[6.3 UpdatingSensor 9](#_Toc440552130)

[6.4 WaterniveauSensor 9](#_Toc440552131)

[6.5 TemperatuurSensor 9](#_Toc440552132)

[6.6 DeurvergrendelSensor 9](#_Toc440552133)

[6.7 WMStatusSensor 9](#_Toc440552134)

[6.8 Sensor 9](#_Toc440552135)

[6.9 SensorListener 9](#_Toc440552136)

[6.10 StatusWeergaveController 9](#_Toc440552137)

[6.11 NoodstopController 10](#_Toc440552138)

[6.12 UART 10](#_Toc440552139)

[6.13 Wasprogramma 10](#_Toc440552140)

[6.14 Fase 10](#_Toc440552141)

[6.15 LogController 10](#_Toc440552142)

[6.16 ActivityLogItem 10](#_Toc440552143)

[6.17 SystemLogItem 10](#_Toc440552144)

[6.18 MessageBuffer 10](#_Toc440552145)

[6.19 MessageBroadcaster 10](#_Toc440552146)

[6.20 Websocket 10](#_Toc440552147)

[6.21 WebsocketListener 10](#_Toc440552148)

[6.22 MessageQueue 10](#_Toc440552149)

[7. Requirment Architecture 11](#_Toc440552150)

[7.1 Usecase diagram 11](#_Toc440552151)

[7.1.1 Wasprogramma uitvoeren: 11](#_Toc440552152)

[7.1.2 Wasprogramma stoppen: 11](#_Toc440552153)

[7.1.3 Update aanbieden: 11](#_Toc440552154)

[7.1.4 Activiteiten logs weergeven: 11](#_Toc440552155)

[7.1.5 Systeem logs weergeven: 11](#_Toc440552156)

[7.2 Activity diagram 11](#_Toc440552157)

[7.3 Constraints diagram 12](#_Toc440552158)

[8. Testen 13](#_Toc440552159)

[9. Conclusie 14](#_Toc440552160)

[10. Evaluatie 14](#_Toc440552161)

[11. Suggesties en aanbevelingen 15](#_Toc440552162)

[12. Bijlagen 16](#_Toc440552163)

# Inleiding

We zullen het hier gaan hebben over alles dat van belang was voor het ontwikkelen van de over internet bestuurbare wasmachine. We zullen het hebben over de test die we hebben gedaan om tot bepaalde beslissingen te komen. Ook zullen we uitgebreid praten hoe de uiteindelijke bediening en werking van de wasmachine is geworden en de keuzes die we hier in gemaakt hebben. We hopen hier door een duidelijk beeld te geven over het proces dat wij als groep hebben doorlopen. Als afsluiting zullen we onze bevindingen als groep en wat wij denken dat de wasmachine nog beter kan maken.

# Management samenvatting.

In dit document zullen we nader in gaan op alle onderdelen van het proces dat wij als groep hebben doorlopen om tot het eind product te komen. Er zal uitleg gegeven worden over de verschillende modellen die gebruikt zijn in het plannen en organiseren van het project. Er zal dus globaal ingegaan worden op de requirments, de requirments architectuur, de solution architectuur en de daadwerkelijke werking van de code. Deze documenten zullen als bijlagen bijgevoegd worden ze zullen echter niet in detail behandeld worden in dit document.

Aan het einden van het verslag zal het team nog even een moment nemen om kort wat te vertellen over hun eigen bevindingen, wat ging goed want niet. Welke onderdelen waren uitdagender dan verwacht. Ook zullen we kort wat zegen over hoe de samenwerking gegaan is.

De conclusie en de aanbeveling zullen het verslag afsluiten. Bij de aanbeveling zullen voornamelijk kijken naar wat wij denken dat de wasmachine nog nodig heeft of wel wat wij er graag bij zouden willen maken als we hier de tijd voor zouden hebben. In de conclusie zal kort samengevat worden wat wij in onze persoonlijke bevindingen waren. Natuurlijk word hier ook nog even terug gekeken naar het gehele proces en hoe wij denken dat het in het vervolg beter kan gaan.

***\*\*\*De student maakt een voorblad paginanummering, hoofdstuk- en paragraafnummering, index, referenties volgens APA, titels en kopjes.\*\*\****

***\*\*\* De student gebruikt verwijzingen en bronvermeldingen volgens APA.\*\*\****

***\*\*\*De student maakt een managementsamenvatting van het gehele document\*\*\****

*\*\*\*De student schrijft een inleiding met opening, doel en vooruitblik\*\*\**

***\*\*\*De student beschrijft op heldere wijze de requirements, de requirements architecture, de solution architecture, de werking van de code en de uitgevoerde tests. \*\*\****

***\*\*\*De student schrijft een afsluitend hoofdstuk met conclusie(s) en aanbevelingen.\*\*\****

***\*\*\*De student maakt gebruik van bijlagen\*\*\****

# De wasmachine

## Algemeen

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

### De opdracht

De opdracht die wij van Swurl Industries hebben gekregen. Was het ontwikkelen van software en een wasmachine.

Deze wasmachine moest op een voor de gebruiker eenvoudige manier bestuurbaar zijn over het internet. Hier voor moest er gebruik gemaakt worden van een web browser. Vanaf deze browser moest het vervolgens mogelijk zijn om een was programma 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 Unique wens die Swurl Industries had 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.

### Bediening

De bediening van de was machine word 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 groten knoppen en niet overdreven veel informatie in een scherm. Dit vooral met het oog op tablet en smartphones.

### Communicatie van de web browser en wasmachine

Gezien het mogelijk is om vanaf het web opdrachten te geven aan de wasmachine is het natuurlijk ook mogelijk om de wasmachine informatie te laten sturen naar het web. Zo is het dus mogelijk om op de zelfde plek als waar de gebruiker het wasprogramma heeft gestart te laten zien waar de machine mee bezig. Maar ook informatie als de huidige water tempratuur. De tijd die hij nog nodig verwacht te hebben.

### Uitvoering nodige stappen wasmachine

### Log systeem

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

Een die alleen maar de hoofd activiteiten van de wasmachine bijhoudt. Deze is bedoeld voor de gebruiker om bijvoorbeeld terug te kunnen kijken wanneer welke was gedraaid was en hoe lang dat heeft geduurd.

Het anderen log dat bijgehouden zou worden is dat van het systeem. In dit log worden dus alle stappen die de wasmachine doorloopt bijgehouden stap voor stap wordt er dus vertelt wat hoe laat gestart is en hoe lang dit geduurd heeft. Dit log was niet bedoeld voor de gebruiker maar voor onderhoud/reparatie van het apparaat. Zo zou het dankzij dit log mogelijk moeten zijn dat een reparateur goed kan zien dat het verwarmingselement niet goed meer werkt omdat het verwarmen van het water steeds langer duurt.

## Screenshots van de webinterface

# 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 onderscheidt gemaakt worden tussen het ontwerpen, programmeren en het testen van de software.

## Ontwerpen

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

De modelleringsgereedschappen bieden ondersteuning in het maken van bijvoorbeeld een UML, klassendiagram en dergelijke. Deze worden ontworpen met behulp van het modellering programma Software Ideas Modeler.

## Programmeren

Notepad ++

## 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.

# Solution architectuur

Solution architecturen worden 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 anderen onderdelen. Ook word hier gekeken naar wat de prioriteit is van welke onderdelen. Om all deze vragen te beantwoorden word er gebruik gemaakt van een aantal diagrammen. Het zo genoemde klassendiagram, het concurrencty diagram en het state transision diagram. Verder word er ook een taakstructurering en anderen nodige protocollen gemaakt.

## Klassendiagram

het klassendiagram word 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 word is dus een overzicht te zien van hoe de verschillende objecten opgemaakt zijn, welke functionaliteit deze hebben en hoe ze met de anderen objecten kunnen praten. Zo is hier ook in terug te zien dat niet elke 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 anderen klassen.

## 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 de genen die er om gevraagd heeft

## state transision diagram

het state transision diagram is een diagram dat in detail afbeeld 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.

## Taalstructurering

De toolstructurering is een overzicht van alle taken die de verschillende objecten hebben. Hier word gelijk gekeken naar de prioriteit van deze taken. De prioriteit word 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.

## communicatie protocol webserver websocket

Het coummunivatie 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 stelen zorg je er voor dat op alle plekken in de code dit op soort gelijke manier gedaan kan/moet worden.

# Requirment Architecture

De naam zegt het al een beetje. De *Requirments Architecture* wordt gemaakt om voordat er geprogrammeerd word vast te kunnen stellen wat er ongeveer nodig gaat zijn om een goed werkten systeem te maken.

Hier bij word gebruik gemaakt van meerdere diagrammen namelijk: *Usecase Diagram*, *Activity Diagram* en *Constraints Diagram*.

## Usecase diagram

De *Usecase* is bedoeld om een beeld te kunnen vromen 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 voor spellen in wat een gebruiker zou kunnen proberen. Natuurlijk moet er bij er schrijven van de code voor gezorgd worden dat onvoorzien gedrag van een gebruiker niet het hele systeem om gooit. Maar dit is niet het doel dan 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 die vervolgens door het systeem uit gevoed moet kunnen worden.

### Wasprogramma stoppen:

Het stoppen van het huidige wasprogramma dat uitgevoerd word.

### Update aanbieden:

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

### Activiteiten logs weergeven:

Toont een log met alle activiteiten die de wasmachine door 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 gestart is.

### Systeem logs weergeven:

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

## Activity diagram

Het *Activity Diagram* is bedoeld om een over zicht 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 over zicht van welke stappen het systeem moet door lopen om tot het gewenste eind resultaat te komen. Er wordt hier dus gekeken naar het aanzetten van de machine en wat hij moet door lopen om met succes een was te draaien. Hoe lang welke stap moet duren is daar aan tegen iets dat op dit moment minder belangrijk is.

## Constraints diagram

Dit diagram wordt niet zo zeer gebruikt om vast te stellen wat het systeem moet doen. Dat wil niet zegen dat het hier niet ook voor gebruikt kan worden maar 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 houd met het feit dat je microprocessor te klein is voor de software die je geschreven hebt.

# Testen

# Conclusie

# Evaluatie

# Suggesties en aanbevelingen

# Bijlagen