

Monitoringsapp

**Jongens van Techniek**

*Auteur : Necati Unal*

*Datum : 8 Oktober 2021*

Referaat

Descriptoren

Azure

Monitoring

Scrum

DevOps

Azure Function

Serverless

Timertrigger

Azure Alerts

Application Insights

Cloud-restrictions

Mei 2021, Den Haag

Dit afstudeerverslag beschrijft de totstandkoming van het project Monitoring app gedurende de afstudeerperiode bij Jongens van Techniek in Den Haag. De afstudeerperiode is van 10-05-2021 tot en met 08-10-2021.

Het project is ontwikkeld aan de hand van de Scrum projectmanagement methodiek. Eerst zijn er interviews gedaan om vervolgens requirements te kunnen verzamelen. Daarvanuit is de plan van aanpak en daarna de architectuur met C4 methode ontwikkeld.

De applicatie is ontwikkeld binnen Azure Functions. Deze is gekoppeld aan een trigger binnen Azure. Vervolgens is de Application Insights gekoppeld aan de applicatie om de prestaties visueel te kunnen weergeven.

Voorwoord

Dit afstudeerverslag is het resultaat van een afstudeerstage bij Jongens van Techniek tussen de periode 10 mei 2021 en 15 oktober 2021. Gedurende deze tijd heb ik een project uitgevoerd bij Jongens van Techniek genaamd de Monitoringsapp.

Het doel van dit verslag is het beschrijven van de werkzaamheden die ik heb uitgevoerd binnen de afstudeerperiode bij Jongens van Techniek (JvT). Deze beschrijvingen gaan over de beslissingen die gedurende het project genomen zijn of de code die is geleverd om tot het eindproject te komen.

Gedurende het afstudeerperiode ben ik geholpen met vragen door mijn collega’s binnen Jongens van Techniek en wil ze daarom bedanken voor de bijdrage die ze hebben gehad. Verder een speciaal bedank voor mijn bedrijfsmentor Jonathan Bosman voor zijn tijd en inzet om mij te kunnen begeleiden.

**Inhoudsopgave**

[1.0 Inleiding 7](#_Toc85157584)

[2.0 Jongens van Techniek 9](#_Toc85157585)

[3.0 Oriëntatie 10](#_Toc85157586)

[3.1 Aanleiding en probleemstelling 10](#_Toc85157587)

[3.2 Stakeholders 12](#_Toc85157588)

[3.3 Doelstelling 14](#_Toc85157589)

[4.0 Aanpak 15](#_Toc85157590)

[4.1 Projectmanagement 16](#_Toc85157591)

[4.2 Planning 17](#_Toc85157592)

[5.0 Sprint 0: Requirements verzamelen 18](#_Toc85157593)

[5.1 Werkzaamheden 19](#_Toc85157594)

[5.2 Requirementsanalyse 19](#_Toc85157595)

[5.3 Epics 20](#_Toc85157596)

[5.4 User stories 21](#_Toc85157597)

[5.5 User stories geprioriteerd 22](#_Toc85157598)

[5.6 Sprint review 23](#_Toc85157599)

[6.0 Sprint 1: Architectuur 24](#_Toc85157600)

[6.1 Architectuur ontwikkelmethode 24](#_Toc85157601)

[6.2 Keuze architectuur ontwikkelmodel 25](#_Toc85157602)

[6.3 C4 model 25](#_Toc85157603)

[6.4 Azure AZ-900 Fundamentals 33](#_Toc85157604)

[6.5 Sprint review 34](#_Toc85157605)

[7.0 Sprint 2: Onderzoek Serverless consumption & architecture 35](#_Toc85157606)

[7.1 Onderzoek Serverless 36](#_Toc85157607)

[7.2 Testpoject Serverless 36](#_Toc85157608)

[7.3 Architectuur 40](#_Toc85157609)

[7.4 Sprint review & retrospective 44](#_Toc85157610)

[8.0 Sprint 3: Pinger 45](#_Toc85157611)

[8.1 Aanpak 46](#_Toc85157612)

[8.2 Werkzaamheden 46](#_Toc85157613)

[8.3 Verwerken output PsPing 49](#_Toc85157614)

[8.4 Sprint review & retrospective 50](#_Toc85157615)

[9.0 Sprint 4: Testplan & alerts & Sending ping data to Application Insight 51](#_Toc85157616)

[9.1 Testplan 52](#_Toc85157617)

[9.2 Ping data sturen 53](#_Toc85157618)

[9.3 Alerts 55](#_Toc85157619)

[9.4 Sprint review & Sprint retrospective 57](#_Toc85157620)

[10.0 Sprint 5: Technical Debt, CertificateChecker & Application Insight Grafieken 58](#_Toc85157621)

[10.1 Nieuwe Function 59](#_Toc85157622)

[10.2 Technical Debt 61](#_Toc85157623)

[10.3 Application Insight Dashboard 65](#_Toc85157624)

[10.4 Sprint review & Retrospective 66](#_Toc85157625)

[11.0 Sprint 6: Certificaatcheck Alert & Application Insight 67](#_Toc85157626)

[11.1 CertificaatCheck Application insight 68](#_Toc85157627)

[11.2 Application Insight 69](#_Toc85157628)

[11.3 CertificaatCheck Alerts 70](#_Toc85157629)

[11.4 Testen 70](#_Toc85157630)

[11.4 Sprint review & Retrospective 73](#_Toc85157631)

[12.0 Sprint 7: Toevoegen Endpoints & Dashboard 74](#_Toc85157632)

[12.1 Pinger endpoints 75](#_Toc85157633)

[12.2 CertificateCheck endpoits 79](#_Toc85157634)

[12.3 Application Insight Dashboard 80](#_Toc85157635)

[12.4 Sprint review & Retrospective 81](#_Toc85157636)

[12.4.1 Evaluatie 81](#_Toc85157637)

[13.0 Evaluatie producten 82](#_Toc85157638)

[13.1 producten 82](#_Toc85157639)

[13.2 Beroepstaken 85](#_Toc85157640)

[Bibliografie 87](#_Toc85157641)

[Bijlages 89](#_Toc85157642)

[Bijlage 1: Interview David Mesman 90](#_Toc85157643)

[Bijlage 2: Interview Peter Hofman 91](#_Toc85157644)

[Bijlage 3: Plan van aanpak 93](#_Toc85157645)

[Bijlage 4: Requirements Rapport 101](#_Toc85157660)

[Bijlage 5: Onderzoek Architectuur Ontwikkelmethode 120](#_Toc85157676)

[Bijlage 6: Onderzoek Serverless 131](#_Toc85157683)

[Bijlage 7: Onderzoek Azure Application Insight 141](#_Toc85157692)

[Bijlage 8: Agile vs Waterval 157](#_Toc85157703)

[Bijlage 9: Chilkat certificate check 168](#_Toc85157712)

[Bijlage 10: Startup File 169](#_Toc85157713)

[Bijlage 11: Test Rapportage 170](#_Toc85157714)

[Bijlage 12: Grafieken Jongens van Techniek 190](#_Toc85157729)

[bijlage 13: Grafieken NSFO.nl 190](#_Toc85157730)

[Bijlage 13: Stakeholder Analyse 191](#_Toc85157731)

[Bijlage 14: Afstudeerplan 200](#_Toc85157732)

[Bijlage 15: Plan van aanpak 207](#_Toc85157733)

# 1.0 Inleiding

In dit document wordt de afstudeerperiode beschreven die is afgelegd tussen de periode 10-05-2021 tot en met 15-10-2021. Het project is gemaakt binnen Jongens van Techniek (JVT) in Den Haag. Het is bedoelt voor de examinatoren en geïnteresseerden in monitoringsmogelijkheden binnen Azure.

Jongens van Techniek is een bedrijf dat zich richt op het oplossen van problemen van haar klanten. JVT ontwikkeld op maat gemaakte software en biedt support voor deze Binnen JVT zijn er zes ontwikkelaars die zich dagelijks bezighouden met Software Engineering en support. Daarnaast is er een zusterbedrijf gekoppeld aan JVT genaamd Uptime Heroes. Daaronder werkt er één werknemer die zich bezighoudt met support gedeelte van het business.

Het project richt zich op de problemen die zich afspelen bij de support afdeling. Bij tickets die binnenkomen blijkt achteraf dat deze vroegtijdig gedetecteerd kunnen worden. Problemen zoals website die er uit ligt of dat de applicatie geen connectie meer heeft met de database. Doordat er geen monitoring beschikbaar is binnen JVT, zijn deze problemen niet vroegtijdig aan te pakken en komen terecht bij de klant. Dit zal resulteren in verstoringen aan bedrijfsprocessen en tijd/omzetverlies voor de klant. Deze problemen zijn motivatie geweest om een oplossing te vinden.

Hoofdstuk 2.0 wordt gestart met een introductie over JVT. Basisgegevens over het bedrijf zoals het aantal ontwikkelaars, de eigenaar en de bijbehorende zusterbedrijf en haar taken worden beschreven. Verder wordt ingegaan op wanneer JVT tot stand is gekomen en met welke ideeën. Verder wordt de klantenbestand van JVT beschreven en de producten die voor deze klanten zijn gemaakt. Als laatst wordt er afgesloten met welke waarde het project zal hebben op JVT en haar klanten.

Verder wordt de plan van aanpak behandeld. Het plan van aanpak is gemaakt als houvast voor het project. Hierin worden verschillende elementen onderbouwd zoals de projectmanagement methode en planning voor het project.   
  
In hoofdstuk 5 begint Sprint 0. Sprint 0 wordt voornamelijk gebruikt om het begin van een project aan te duiden. Hier wordt begonnen aan vergaring en analyse van requirements. Deze worden vervolgens gesplitst in Epics en user stories waarna ze worden geprioriteerd.

Aan de hand van de requirements wordt een architectuur ontwikkeld. Voordat de architectuur ontwikkeld kan worden zal een ontwikkelmethode gekozen worden. Uit een onderzoek is gebleken dat C4 het beste past bij het project en wordt gebruikt om de vier levels van het architectuur te ontwikkelen. Om informatie op te doen over de Azure omgeving wordt een cursus gevolgd. Hieruit zijn er nieuwe inzichten verkregen die in de volgende sprint worden behandeld. In hoofdstuk 6 worden de inzichten die gedaan zijn gedurende het cursus meegenomen. Deze inzichten vormen de basis voor een onderzoek naar Serverless Consumption. Uit een onderzoek en een test project is gebleken dat het gebruik van de Serverless meer toegevoegde waarde zal hebben. Hierdoor is de oude architectuur aangepast naar de nieuwe situatie die verder in 6.3 wordt behandeld.

Na het aanpassen van de architectuur wordt er een start gemaakt aan implementatie van de eerste Function. De userstory “Als supportmedewerker wil ik inzicht hebben op downtimes zodat ik weet of er een probleem zich afspeelt.” is binnen de sprint genomen. Door limitaties binnen de cloud is nodig geweest om het te herschrijven.

Na de meeste functionaliteiten van de ping Function af te hebben is er een stap gemaakt richting testen ervan. Binnen sprint 4 wordt het testplan opgezet. Herin wordt besproken wat getest zal worden op welke manier en welke methodes er gebruikt zal worden. Verder wordt gekeken naar de notificaties. Vervolgens wordt een stap genomen richting het opslaan van data door gebruik te maken van Application Insight.

Door een technical debt in de vorige sprint wordt sprint 5 gestart met het oplossen van deze technical debt. Deze debt komt door het gebruik van een obsolete object die vervangen moet worden. Vervolgens wordt er een stap gemaakt richting een nieuwe test genaamd de “CertificateCheck”. Als laatst worden de opgeslagen data gevisualiseerd in Application Insight.

Sprint 6 is een korte sprint geweest door feestdagen en vakanties. Desondanks zijn een aantal deliverables opgeleverd. De “CertificaatCheck” Function is gekoppeld aan Azure Application Insight voor opslag van data. Deze data is vervolgens gebruikt om grafieken mee te maken. Als laatst zijn er alerts toegevoegd aan de metric om mails te ontvangen wanneer grenswaardes worden overschreden.

De laatste sprint, oftewel sprint 7 gaat over het toevoegen van meerdere endpoints aan de tests. Door limitaties binnen Jongens van Techniek is besloten om deze endpoints lokaal op te slaan in plaats van een database. Verder is er extra metadata toegepast op de metric data dat verstuurd wordt naar Application Insight. Deze extra data, oftwel Custom Dimensions zijn nodig geweest om data op Application Insight te kunnen identificeren. Met deze identificatie is het mogelijk geweest om de grafieken voor elk endpoint aan te maken. Dit concludeert in een dashboard met twee grafieken per endpoint.

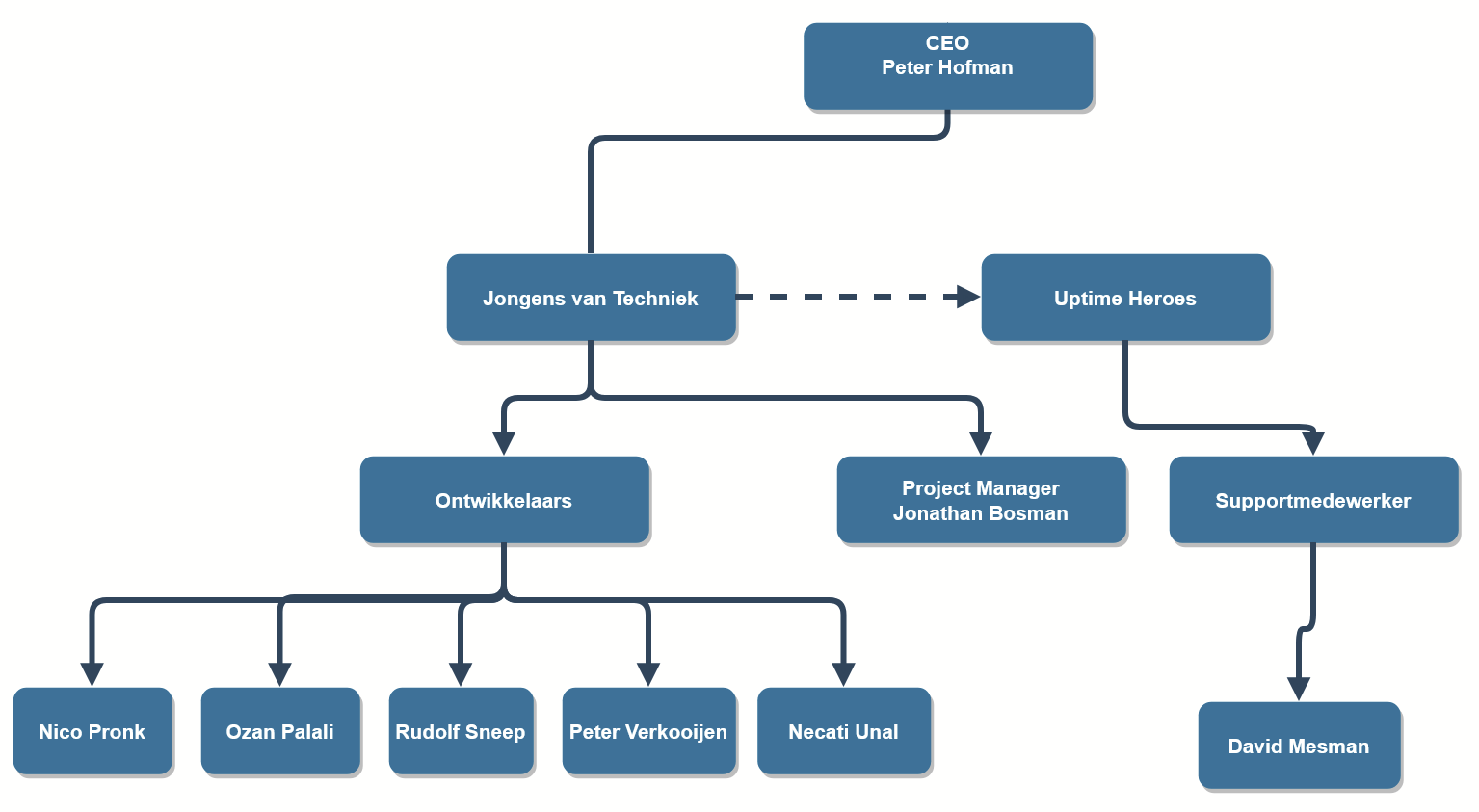
Als laatst wordt er geëvalueerd op de opgeleverde producten. In de evaluatie wordt gekeken naar de uiteindelijke product en wordt beschreven wat er is opgeleverd. Verder wordt bijzonderheden behandeld die gedurende productie van het tussenproduct is tegengekomen en de keuzes die gedurende het ontwikkeling zijn genomen. Gedurende de laatste hoofdstuk wordt geëvalueerd op de Plan van aanpak, requirementsanalyse, architectuur, onderzoek naar Serverless consumption en Application Insight, het testplan en testrapportages en het onderzoek naar Agile vs. Waterval.

# 2.0 Jongens van Techniek

In dit hoofdstuk zal inzicht gegeven worden in Jongens van techniek. Informatie zoals bedrijfsgegevens, aantal werknemers en de eigenaar wordt in dit hoofdstuk besproken. Deze informatie zal de lezer een indruk geven van de omgeving waarin de opdracht is gemaakt.

Jongens van techniek is een Software Engineering bedrijf dat op maat oplossingen levert. Opgericht in 2015 door Peter Hofman heeft Jongens van Techniek nu 5 tot 10 ontwikkelaars die zich dagelijks bezighouden met het ontwikkelen van oplossingen voor de klanten. Om het bedrijf uit te breiden is later Uptime Heroes opgericht als zusterbedrijf van Jongens van Techniek. Uptime Heroes is het bedrijf waar support plaatsvindt.

In de organogram (figuur 1) is te zien dat Dhr. Peter Hofman de oprichter is van JVT. Dhr. Hofman is naast CEO een ontwikkelaar en werkt aan deliverables. De sterke kant van JVT is dat het een volledig Nederlands bedrijf is zonder outsourcing. Dit zorgt voor een punt van communicatie mochten ze vragen hebben zonder verschillen in tijdzones en taalbarrières. Hiermee zorgt JVT dat er geen informatie verloren gaat tijdens translatie. Dit zorgt voor transparante communicatie en een hoge mate van bereikbaarheid mocht de klant meer wensen hebben.



Figuur 1 Organogram Jongens van Techniek

Het klantenbestand van Jongens van Techniek is divers. Op het moment van schrijven zijn Nederlandse Schapen- en Geitenfokkers Organisatie(NSFO) en Open universiteit(OU) de grootste projecten waaraan gewerkt wordt. Jongens van techniek heeft daarvoor projecten gehad van klanten zoals Siemens, Ministerie van Landbouw, Ziggo, Vodafone en SNS-bank.

De rol van het afstudeerproject is een ondersteuning voor alle oplossingen die geleverd zijn bij de klanten. De oplossingen die geleverd worden, zijn niet meer te beheersen. Hierdoor heeft Jongens van Techniek een stap gedaan richting support door Uptime Heroes op te richten. Om deze support uit te kunnen breiden heeft Jongens van Techniek daarom dit project opgezet om meer support te kunnen bieden aan hun klanten.

# 3.0 Oriëntatie

## 3.1 Aanleiding en probleemstelling

Dit hoofdstuk behandelt de oriëntatie van het project. Hier wordt de huidige situatie van de helpdesk beschreven. Daarna wordt de stakeholders beschreven. Wie deze zijn, waarom deze stakeholders gekozen zijn en om welke redenen. Er wordt uitgelegd hoe de klant zijn probleem bij de supportmedewerker krijgt en hoe deze verder behandeld wordt. Als laatst is er een rooster gemaakt van de producten die opgeleverd zullen worden.

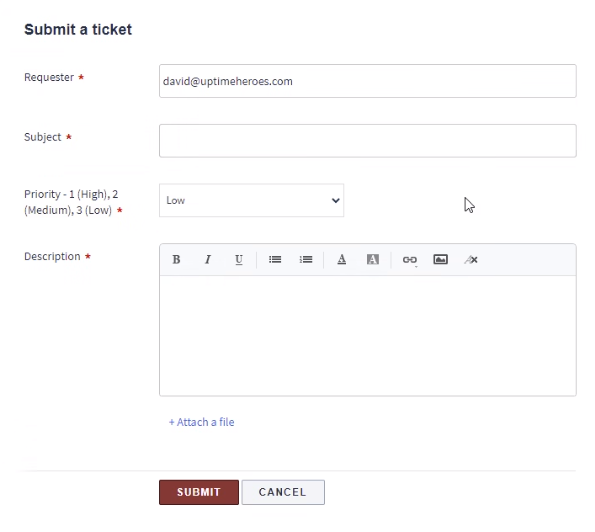
Oplossingen oftewel software die door Jongens van Techniek wordt geleverd, wordt geïmplementeerd in verschillende programmeertalen. Met focus op de Azure platform, ontwikkeld JVT software die zich richt op het oplossen van specifieke problemen.

Na oplevering van de oplossing heeft Jongens van Techniek geen controle meer over de applicatie en wordt beheer verplaatst naar de klant. Hierdoor kan Jongens van Techniek de problemen niet inzien. Gedurende levensduur van de applicatie kan de klant tegen fouten aanlopen die preventief oplosbaar zijn, zoals een volle database of een verlopen certificaat.

Voor dit soort problemen heeft Jongens van Techniek een stap gedaan richting support door het opzetten van het zusterbedrijf Uptime Heroes. De functie van Uptime Heroes is het leveren van support aan klanten die een onderhoudscontract zijn aangegaan. Klanten kunnen aan de hand van een formulier “Tickets” aanmaken die bij de supportmedewerkers komen. Mocht de medewerker genoeg informatie hebben en het probleem kunnen reproduceren, zal hij/zij poging toen tot het oplossen ervan. Hiermee wordt reactief gereageerd op fouten. Het doel van het project is om fouten die op kunnen treden preventief tegen te gaan door te monitoren.

### 3.1.1 Ticket aanmaken

Met een supportsysteem kunnen kanten een formulier invullen die tickets aanmaakt voor de supportmedewerker. Deze software heet Freshdesk. Freshdesk is het aanspreekpunt voor klanten zodra er een foutmelding optreed. De klant vult een formulier in waar het onderwerp, prioriteit en descriptie gevraagd wordt. Deze tickets komen dan terecht bij de supportmedewerker (figuur 2).



Figuur 2 Support ticket aanmaken

Als de supportmedewerker genoeg informatie heeft om het probleem op te lossen, zal hij/zij een poging wagen. Als dat niet mogelijk is kan diegene altijd hulp van het ontwikkelteam vragen om het probleem op te lossen.

Mocht de supportmedewerker niet genoeg informatie hebben om het probleem te kunnen reproduceren om het vervolgens op te lossen zal hij zijn/haar vraag stellen bij de desbetreffende klant. Dit concludeert een reactief proces waarbij de supportmedewerker zal reageren op het moment tickets binnenkomen.

## 3.2 Stakeholders

Om te kunnen achterhalen wie de juiste stakeholders zijn voor het project, is gebruik gemaakt van een stakeholderanalyse. De analyse is afkomstig van Handboek Requirements van Nicole de Swarts (Nicole de Swart, 2010) die beweert dat de vertegenwoordigers gekoppeld moeten worden aan één van de drie categorieën:

1. Stakeholders in het businessdoel
2. Stakeholders in het systeem
3. Stakeholders in het project

Door eerst belanghebbenden te plaatsen onder de juiste groep is een overzicht gemaakt van de vertegenwoordigers.

|  |  |  |
| --- | --- | --- |
| **Belanghebbenden Business** | **Belanghebbenden Systeem** | **Belanghebbenden Project** |
| Peter Hofman (Project eigenaar) | David Mesman (Supportmedewerker) | Jonathan Bosman (Project manager) |
|  |  | Peter Hofman |

Deze stakeholders zijn bevestigd op basis van een checklist. De checklist bevat 5 simpele vragen waar de stakeholders aan moeten voldoen.

Dit zijn:

1. De vertegenwoordiger weet wat er speelt bij de belanghebbenden
2. De belanghebbenden hebben vertrouwen in de vertegenwoordiger
3. De vertegenwoordiger is bevoegd om namens de belanghebbenden te spreken
4. De vertegenwoordiger heeft genoeg tijd
5. De vertegenwoordiger beschikt over goede communicatieve skills.

Door tijdsgebrek is gebleken dat Nico Pronk wegvalt als belanghebbenden van het systeem. Hetzelfde is ook toepasselijk op Peter hofman. Door drukke schema is het niet mogelijk om altijd bij de sprint reviews te kunnen zitten mocht dit nodig zijn. Desondanks is de beschikbaarheid van representatie binnen dit doelgroep klein waardoor hij toch de rol als stakeholders op zich heeft genomen.

Voor communicatiedoeleinden is vervolgens gebruik gemaakt van de “Influence vs. interest” matrix van Mendelow (1991). De stakeholders zijn vervolgens binnen het model geplaatst Vanuit het model kan achterhaald worden hoeveel invloed iemand zal hebben op het project en hoeveel focus er op de desbetreffende persoon gericht zal moeten worden. Hieruit is gebleken dat David en Jonathan de belangrijkste stakeholders zijn voor het project.

### 3.2.1 Supportmedewerker

Voor de stakeholder supportmedewerker is er alleen één persoon geselecteerd, genaamd David. Dit komt doordat er geen andere supportmedewerkers zijn binnen JVT. Door intern gebruik van de software kan er geen supportmedewerkers buiten JVT bij betrokken worden omdat ze niet de doelgroep supportmedewerker binnen JVT kunnen representeren. De beschikbaarheid voor dit rol is hierdoor klein. Supportmedewerkers van andere bedrijven is overwogen maar is niet mogelijk geweest doordat zij niet de dagelijkse werkzaamheden verrichten die David doet. Hierdoor is de representativiteit klein.

De ‘MonitoringApp zal intern gebruikt worden door de supportmedewerker. Hierdoor is het van belang dat de supportmedewerker mee kan kijken gedurende de ontwikkeling van het systeem. De supportmedewerker zal dagelijks te maken hebben met de monitoringstool en zal continue zich ermee bezig houden. Doordat deze medewerker een belangrijke stakeholder is wordt de focus gezet op de supportmedewerker om te kunnen zorgen welke tests het beste zijn. Deze supportmedewerker is representatief doordat hij voornamelijk werkt voor de supportafdeling en dagelijks werkt aan tickets. Hierdoor is zijn mening handig om vast te stellen welke problemen het meest voorkomen en welke tests handig zijn.

### 3.2.2 Projecteigenaar

De projecteigenaar is Peter Hofman, CEO van JVT. Door zijn titel, kan de projecteigenaar lange termijn oplossingen beoordelen, bijvoorbeeld gebruik van frameworks of programmeertalen. Verder is deze persoon de aanspreekpunt voor Azure gerelateerde vragen en kan helpen bij de architectuur van het project. De projecteigenaar zorgt voor de juiste prioriteiten van de user stories en de toekomst voor het project.

### 3.2.3 Projectmanager

Binnen JVT is jonathan de projectmanager. Hij zorgt ervoor dat projecten vloeiend verlopen en zorgt voor klantcontact. Hierdoor is Jonathan van belang binnen het project.

Jonathan zal gedurende het project de rol van de klant spelen. Hij zal aangeven wat hij liever wel en niet wilt zien qua tests. Daarnaast kan hij een evaluerende rol aannemen. Door sprints te reviewen en nieuwe sprints klaar te zetten kan hij de rol van productowner invullen. Jonathan is het aanspreekpunt voor vragen over klanten en over features.

## 3.3 Doelstelling

Het doel van dit project is het implementeren van een applicatie met monitoring functionaliteiten. Deze functionaliteiten zullen een applicatie testen met monitoring doeleinden. De resultaten van alle tests zullen in een publieke database worden opgeslagen. Er is gekozen om gebruik te maken van bestaande oplossingen voor een dashboard. De bestaande oplossingen kunnen worden gebruikt om een dashboard te kunnen maken voor de supportmedewerkers. Daarom is gekozen om bestaande oplossingen te gebruiken zodat de resterende tijd gebruikt kan worden voor implementatie van de applicatie.

Verder wordt een architectuur ontwikkeld om te kunnen zorgen dat er een fundament bestaat voor het project. De architectuur is ontwikkeld aan de hand van een ontwikkelmethode. In bijlage 5 worden de voordelen en nadelen besproken waaruit geconcludeerd wordt dat C4 gebruikt zal worden voor het project.

Vervolgens zal de kwaliteit van de applicatie worden bewaakt. Om ervoor te zorgen dat na elke iteratie de applicatie juist werkt zal een testplan worden geleverd. Om elke sprint te kunnen testen zal een testrapport uit komen met logische en fysieke testgevallen zodat de kwaliteit geverifieerd kan worden.

Opgesomd wordt de volgende producten in de loop van de afstudeerperiode geleverd:

* Plan van aanpak
* Requirement analyse
* Vooronderzoek monitoringsmogelijkheden
* Onderzoek architectuur/ architectuurontwerp
* Testplan/ Testrapport
* Monitoringsimplementatie en alle scripts die daarbij horen.
* En alle andere project gerelateerde producten die gemaakt wordt gedurende de afstudeerperiode

# 4.0 Aanpak

Voordat er begonnen kan worden aan de interviews is er een plan van aanpak gemaakt en gesprekken gevoerd met de stakeholders. In dit geval was het gesprek met de supportmedewerker. Daar is vastgesteld welke processen van start gaan zodra er een probleem zich voordoet bij de klant. Door een gesprek te houden met David was er voldoende informatie om de plan van aanpak verder aan te kunnen vullen. Aan de hand van de afstudeerplan en het interview, is er een requirementsrapport opgesteld.

Vervolgens wordt de architectuur van de applicatie ontwikkeld. Om een architectuur te ontwikkelen zal een methode voor gebruikt moeten worden. Uit het onderzoek naar ontwikkelmethodes is geconcludeerd dat C4 het beste past bij het project (bijlage 5).

C4 is licht van natuur waardoor er weinig overhead blijft bij een team van één. Daarnaast is C4 flexibel waardoor wijzigingen makkelijk doorgevoerd kunnen worden. Verder wordt het C4-model gevisualiseerd aan de hand van een “Ubiquitous Language”, oftewel een makkelijke taal, makkelijk te begrijpen door beide partijen (Business en Dev) (C4-model, 2021). Dit gaat hand in hand met de flexibiliteit van Scrum en vormen ze samen een goede connectie. Doordat Domain Driven Design (DDD) gericht is op projecten waarbij het domein onbekend is, is het niet toepasselijk op het project. Verder heeft Attribute Driven Design (ADD) alle requirements en user stories nodig om de architectuur mee te kunnen ontwikkelen. Dit is een toepasselijke scenario op de MonitoringsApp maar omdat de lange termijn gebruik van de MonitoringApp nog niet duidelijk is, is er besloten om deze niet te gebruiken vanwege het niet hebben van de requirements die nog later er bij zullen komen.

Na een vooronderzoek naar monitoringsmogelijkheden zijn afwegingen gedaan tussen de Azure Function oplossing en de ASP.Net framework. Hieruit is gebleken dat implementatie binnen Functions sneller en lichter opgelost kunnen worden in combinatie met Application Insight. Dit komt voort uit een term genaamd Serverless Consumption (Bijlage 6). Doordat de ASP.net framework een standaard aantal resources nodig heeft om draaiende te kunnen blijven op Azure zal deze ten alle tijden resources nodig hebben. De servers blijven draaien waardoor dit geld zal kosten. Dit kan makkelijker opgelost worden door de Serverless optie te gebruiken. In bijlage 6 wordt Serverless verder behandeld.

Verder willen we de kwaliteit behouden van de applicatie. Hiervoor is er gekeken naar welke testmethodieken passen bij de Monitoring App. Er wordt een testmethodiek gekozen die past bij Azure Functions. Hierin zal staan hoe er omgegaan wordt met de merge requests en static code analysis. Verder wordt een testplan opgesteld per sprint wanneer dat nodig is om het doel van de testen vast te stellen en de juiste test te implementeren.

## 4.1 Projectmanagement

Voor het project is gebruik gemaakt van een Agile projectmanagement methode(bijlage 8). Binnen het bedrijf werd een Scrum achtig methode gebruikt. Er werd gebruik gemaakt van Daily standup en Sprint planning maar sprint retrospective of sprint reviews waren absent.

Uptime Heroes en Jongens van techniek deden samen een daily standup voor verschillende projecten. Een daily standup was niet voor één project specifiek dus werd standup cross-project gedaan. Hierdoor werd verwacht dat de MonitoringApp project daarbij zal aansluiten.

Verder zijn er geen Scrum-achtige events binnen Jongens van Techniek. Doordat de reviews niet gezamenlijk plaatsvinden is besloten om dit zelf vorm te geven.

Voor het project is gebruik gemaakt van een methodiek gelijk aan Scrum. Zoals in bijlage 8 wordt beschreven heeft Scrum een aantal events. Hiervan is de Daily standup van gebruikt om aan het begin van de dag een samenvatting te geven aan het team wat de komende dag zal moeten gebeuren.

Van de Sprint review, sprint retrospective, sprint planning is geen gebruik van gemaakt. Dit komt doordat het ontwikkelteam klein was. Desondanks zal het project zich goed doen met retrospectives waardoor deze events zijn vervangen door een soort gelijke event. Verder is er gebruik gemaakt van sprints van twee weken.

Deze events zijn wekelijks gehouden. Gedurende deze sessies zijn reviews gehouden met de Supportmedewerker (David) of de projectmanager (Jonathan). Wanneer een deliverable opgeleverd was, is samen met David gekeken naar voortgang en is er gevraagd op feedback. Daarnaast zijn er aparte, wekelijkse sessies gehouden met Jonathan om te kunnen werken aan Sprint planning en sprint retrospective. Tijdens deze sessies is er gekeken naar wat er is gebeurd de afgelopen sprint. Wanneer er problemen zijn ondervonden in de afgelopen sprint, worden die besproken en verbeterend voor de volgende sprint. verder wordt er gekeken welke user Stories zijn verbeterd kunnen worden. Daarnaast wordt gedurende de sessie bekeken welke user story verbeterd of anders geprioriteerd moet worden. vervolgens wordt de user story binnen de sprint genomen.

## 4.2 Planning

Tijdens sprint 1 is er een plan van aanpak ontwikkeld die richting geeft aan het project. Binnen dit document is er een planning gemaakt voor het project. Een planning is gemaakt waarin de grote deliverables en fases een tijdslot gegeven worden om aan te geven wanneer verwacht wordt aan het product te kunnen beginnen en implementeren.

De planning:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sprint |  | |  | | 1 | | 2 | | 3 | | 4 | | | 5 | | | 6 | | | 7 | | | 8 | | | 9 | | | 10 | | | 11 | |
| Week | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | | 14 | 15 | | 16 | 17 | | 18 | 19 | | 20 | 21 | | 22 | 23 | | 24 | 25 | |
| Plan van aanpak |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |
| **Ontwerpen** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Onderzoek |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |
| Ontwikkelen |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |
| Testen |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |
| Architectuur ontwikkelen |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |
| **Requirements** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Requirements vergaren |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |
| Initiële backlog |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |
| Sprint backlog |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |
| **School** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Bedrijfsbezoek (online) |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |
| Afstudeerdossier |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |

# 5.0 Sprint 0: Requirements verzamelen

Sprint 0 staat in het teken van voorbereiding en oriëntatie (bijlage 1 & 2). Door een oriënterende sprint te doen is het mogelijk om meer inzichten te verzamelen voor het project, om de plan van aanpak mee te kunnen ontwikkelen. Hiermee is het mogelijk om te kunnen achterhalen of andere problemen zich afspelen buiten de problemen die in het afstudeerplan is beschreven. Gedurende sprint 0 worden geen user stories binnen de sprint gehaald omdat er nog requirements verzameld en geprioriteerd moeten worden. Dit betekend dat het doel voor de komende sprint zo veel mogelijk informatie verzamelen is.

Binnen sprint 0 zal de start van het plan van aanpak plaatsvinden. Er wordt besproken welke aanpak gekozen is voor het project en waarom. Verder wordt interviews afgenomen die in sprint 0 hebben plaatsgevonden. Aan de hand van deze interviews wordt een requirementanalyse gemaakt.

Om de requirements verder op te splitsen en te verduidelijken zijn Epics gemaakt om ze te kunnen categoriseren. Onder elke Epics worden de user stories gezet die uit de requirements zijn voortgekomen Daarna worden de user stories geprioriteerd aan de hand van de MoSCoW systeem die is aangepast om het makkelijker voor de stakeholders te maken.

## 5.1 Werkzaamheden

Voor het maken van het plan van aanpak is het afstudeerplan erbij gepakt. Hierbij is gekeken naar de problemen die zich binnen het domein voordoen. Deze probleemstelling is overgenomen en is er gedurende interviews gekeken naar hoe deze problemen tot stand komen. Hier is het inzicht van de stakeholders belangrijk doordat zij dagelijks te maken hebben met de problemen. David, oftewel de supportmedewerker van uptime Heroes gaf inzicht in het proces die hij uitvoert wanneer een ticket geopend wordt door een klant (Hoofdstuk 3.1.1). Hiervan uit is er kennis opgedaan op de huidige situatie van het applicatielandschap van Jongens van Techniek (Bijlage 1).

De vragen die daarna naar boven zijn gekomen zijn voorgelegd bij de bedrijfsmentor en de stakeholders. Dit is gedaan om zo veel mogelijk informatie te kunnen verzamelen voor het plan van aanpak.

## 5.2 Requirementsanalyse

Samen met het plan van aanpak, afstudeerplan en de interviews zijn de requirements uit opgesteld. Het afstudeerplan toonde een solide fundament voor het project maar het miste een aantal punten dat aangevuld worden. Deze punten zijn gedurende de interviews aangevuld door de interviews en het plan van aanpak aangevuld.

### 5.2.1 User requirements

De user requirements vormen de requirements die zichtbaar zijn voor een gebruiker van het systeem. Dit zijn tastbare requirements die zullen leiden tot functionaliteiten binnen de applicatie.

|  |  |
| --- | --- |
| Nr #. | User requirement |
| 1 | De supportmedewerker kan de uptime van de gemonitorde applicatie bekijken. |
| 2 | De supportmedewerker kan de certificaten van de gemonitorde applicatie bekijken. |
| 3 | De supportmedewerker kan een multi-step web test laten uitvoeren. |
| 4 | De klant krijgt een email wanneer er iets mis is met de software. |
| 5 | De klant kan aangeven of de benodigde werkzaamheden uitgevoerd wordt. |
| 6 | De supportmedewerker krijgt een email wanneer er iets mis is met de gemonitorde software. |
| 7 | De supportmedewerker kan handmatig de uptime-tests uit laten voeren. |
| 8 | De supportmedewerker kan handmatig de certificaat-check uit laten voeren. |
| 9 | De supportmedewerker kan handmatig de multi-step webtest check uit laten voeren. |
| 10 | De supportmedewerker ziet welke werkzaamheden voorgesteld zijn. |
| 11 | De klant kan zijn dashboard inzien met alle statussen |

Tabel 1: User requirements (functionele requirements)

### 5.2.3 Systeem requirements

Dit zijn functionaliteiten die niet vereisen door de gebruiker maar uitgevoerd worden door het systeem. Daarom zijn deze niet altijd even zichtbaar voor de gebruiker maar wordt op de achtergrond uitgevoerd. Verder wordt aan de systeemrequirements een kwaliteitseis toegevoegd zoals beschreven in de ISO 25010. Deze ISO is een standaard voor de industrie om deze iso is een verbetering op de ISO 9126 vanwege de nieuwe karakteristieken die bij komen kijken zoals beveiliging (security) en compatibiliteit (compatibility) (Monterail, 2019). Verder is het een internationaal geaccepteerde software testing certificaat. Daarom is er besloten om voor ISO 25010 te gaan in plaats van ISO 9126.

|  |  |  |
| --- | --- | --- |
| Nr #. | Systeem requirement | Kwaliteitseis |
| 1 | Het systeem mag de gemonitorde applicatie niet storen bij dagelijkse functionaliteiten | Resource utilization |

Tabel 2: Systeem requirements (niet-functionele requirements)

## 5.3 Epics

Vanuit de requirements ontstaan user stories. User stories kunnen onder een thema vallen genaamd een epic. Dit is een grotere functionaliteit waaronder meerdere user stories onder kunnen vallen. Deze zijn niet gesplitst in smallere en duidelijkere user stories.

In de tabel hieronder zijn de epics beschreven die van belang zijn tijdens het project.

|  |  |
| --- | --- |
| Nr # | Epics |
| 1 | Als supportmedewerker wil ik de resultaten van de toegevoegde tests weten zodat ik weet of er een probleem zich af speelt of heeft gespeeld. |
| 2 | Als supportmedewerker wil ik een testbeheer zodat ik alle testen kan verwijderen, wijzigen en toevoegen. |
| 3 | Als klant wil ik een dashboard, zodat ik een overzicht heb van de status van mijn software. |
| 4 | Als klant wil ik weten wanneer een probleem is, zodat ik up to date ben van de status van mijn website. |

Tabel 3: Epics

Binnen de epics zijn twee users te vinden waaronder de supportmedewerker en de klant. De taak van de supportmedewerker is om de monitoring bij te houden. Dit is waarom er besloten is om alle tests onder één grote epic te zetten. De tests die geïmplementeerd worden, worden later in de user stories bepaald.

Epic #2 zorgt ervoor dat alle tests beheert kunnen worden. Op deze manier kan er een beheer worden gemaakt voor de supportmedewerker waardoor hij test kan toevoegen aan de applicatie en de tests kan bewerken. Hiervoor is besloten om dit onder één epic te laten vallen.

Ten slotte zijn er twee epics gemaakt voor de klant. Epic #3 is gemaakt zodat er later gezorgd wordt voor een visueel dashboard voor de klant. Verder is het van belang dat de klant op de hoogte gehouden wordt van eventuele foutmeldingen binnen de applicatie voor de klant. Hiervoor is epic #4 voor geboren.

## 5.4 User stories

De user stories in dit hoofdstuk vormen de laag onder een Epic. Deze user stories zijn functionaliteiten die onder de Epic vallen en geïmplementeerd wordt aan de hand van de prioriteiten gesteld door de stakeholder. Per User story wordt de epic nummer bij genoteerd om duidelijk te maken onder welke epic deze behoort. Ten slotte zal het eerder benoemde systeem voor prioriteiten gebruikt om de user story te kunnen prioriteren.

|  |  |  |
| --- | --- | --- |
| Nr #. | Epics | User story |
| 1 | 1 | Als supportmedewerker wil ik inzicht hebben op downtimes zodat ik weet of er een probleem zich afspeelt. |
| 2 | 1 | Als supportmedewerker wil ik inzicht hebben op verlopen certificaten, zodat ik weet of er een probleem zich afspeelt. |
| 3 | 1 | Als supportmedewerker wil ik inzicht hebben op multi-step webtests, zodat ik de output kan valideren. |
| 4 | 1 | Als supportmedewerker wil ik een email ontvangen mocht er een downtime optreden. |
| 5 | 1 | Als supportmedewerker wil ik een email ontvangen mocht er een certificaat probleem optreden. |
| 6 | 2 | Als supportmedewerker wil ik endpoints wijzigen aan de lijst van de gemonitorde endpoints, zodat ik aanpassingen kan doen mochten de URL’s wijzigen. |
| 7 | 2 | Als supportmedewerker wil ik certificaten toevoegen aan de lijst van de gemonitorde certificaten, zodat ik bestaande en nieuwe certificaten kan aanpassen. |
| 8 | 3 | Als klant wil ik de uptime van mijn website inzien zodat ik weet op welke momenten mijn website niet beschikbaar is. |
| 9 | 3 | Als klant wil ik weten of mijn certificaat nog valide is zodat mijn website nog veilig en toegankelijk is. |
| 10 | 2 | Als supportmedewerker wil ik de testmomenten kunnen aanpassen, zodat ik zelf kan bepalen wanneer de tests uitgevoerd wordt. |
| 11 | 1 | Als supportmedewerker wil ik weten welke werkzaamheden voorgesteld zijn aan de klant, zodat ik kan zien dat er door ons al is gereageerd op een probleem. |
| 12 | 4 | Als klant wil ik een email ontvangen zodat ik weet dat de support op de hoogte is gesteld van de downtime. |
| 13 | 4 | Als klant wil ik een email ontvangen zodat ik weet dat de support op de hoogte is gesteld van een verlopen certificaat. |
| 14 | 4 | Als klant wil ik terugkoppelen, zodat ik weet dat de support het zal oplossen. |

Tabel 4: User stories

De totstandkoming van US1 & 2 is afkomstig van het gesprek met Peter waarbij hij aangeeft te willen beginnen met een paar simpele tests en later te willen uitbreiden (bijlage 2). US 4 en 5 zijn resultaten van het gesprek met Peter waarbij hij aangeeft dat sommige personen op de hoogte gehouden moeten worden van belangrijke events binnen de MonitoringApp.

## 5.5 User stories geprioriteerd

Gedurende het prioriteren is naar boven gekomen dat eerst het infrastructuur van het applicatie gemaakt zal moeten worden. Dit betekend dat alle tests eerst werkend moeten worden voordat er gewerkt zal worden aan het interface van de applicatie.

Dit is gedaan omdat de stakeholder niet zeker was van veranderingen die later zal plaatsvinden. De stakeholder wilde eerst de tests zien werken waarna er later beslist wordt op welke manier de data gepubliceerd zal worden bij de klant. Volgens de stakeholder kon dit op meerdere manieren waaronder Azure Monitor. Dit is een SaaS-oplossing binnen Azure dat monitoring kant en klaar mogelijk maakt.

Voor het prioriteren van de user stories is besloten de MoSCoW methode voor te gebruiken (Bijlage 4). Deze methode is gekozen na een vergelijking te doen met de HiPPO methode. De HiPPo methode staat voor dat de hoogst betaalde persoon het zwaarst wegende prioriteit kan opgeven (Alex Ponomarev, 2019). Dit creëert een sfeer waarbij het team dynamiek niet meer gelijk is terwijl binnen Agile dit wel de voorkeur voor is. Hierdoor zal dit niet overwogen worden. Gedurende MoSCoW methode wordt geprioriteerd aan de hand van vier categorieën. De valkuil hierbij is dat men al snel alle requirements op Must have zal zetten waardoor er geen overzicht meer is wat belangrijk is. Hiervoor is besloten om gebruik te maken van een cijfer methode waarbij er geprioriteerd zal worden aan de hand van cijfers.

De prioriteit die de user stories op dit moment krijgen zijn als volgt:

|  |  |  |
| --- | --- | --- |
| User story #1 | Prioriteit cijfer | Prioriteit volgens MoSCoW |
| 1 | 1 | Must have |
| 2 | 1 | Must have |
| 5 | 1 | Must have |
| 10 | 2 | Should have |
| 4 | 2 | Must have |
| 7 | 2 | Must have |
| 12 | 2 | Must have |
| 13 | 2 | Must have |
| 6 | 3 | Should have |
| 14 | 5 | Could have |
| 11 | 5 | Could have |
| 8 | 6 | Could have |
| 9 | 6 | Could have |
| 3 | 10 | Will be nice |

Tabel 5: Prioritering van de user stories

## 5.6 Sprint review

Het opstellen van de plan van aanpak is vermoeiend geweest. Hiervoor moest veel informatie verzameld worden over het project. Hier kwam de oriënterende gesprek met David goed van pas (H3). Daar vanuit is het probleemdomein verduidelijkt. Het was een handig gesprek omdat ik inzicht kreeg in de dagelijkse werkzaamheden van David. Na het interview met David ben ik iets wijzer geworden van de gang van zaken. Ik kreeg meer inzicht in de problemen die de supportmedewerker had waardoor ik al gelijk mee kon denken aan de mogelijke oplossingen.

De Interview van Peter kwam later bij kijken door drukke schema’s waardoor ik niet al gelijk de requirements kon opstellen. Het gebruik van de afstudeerplan voor de requirements is een goed begin geweest. Hierdoor zijn er een aantal requirements naar boven gehaald die later naar een hogere niveau zijn getrokken. Verder zijn deze samen met de stakeholder behandeld om een zo compleet mogelijk beeld te hebben. Daarnaast heeft de bedrijfsmentor, oftewel de projectmanager ernaar gekeken en zijn we samen door de requirements heen te gaan. Door deze review te doen, heb ik veel informatie kunnen verkrijgen om volgende fases van het project in te stappen.

Per requirement is er gekeken of dit anders verwoord kan zijn en of het duidelijk is voor beide partijen. Dit is een belangrijk moment geweest doordat het heeft geholpen bij de kwaliteit van de requirements. Daarnaast is het belangrijk geweest om vast te stellen of beide partijen elkaar goed hebben begrepen.

# 6.0 Sprint 1: Architectuur

Binnen sprint twee zal gewerkt worden aan het architectuur. Het doel van de sprint is het maken en reviewen van de architectuur. Het maken van de sprint is behaald alleen heeft de review ervan vertraging opgelopen door drukke schema van de bedrijfsmentor. Er worden geen user stories meegenomen in de sprint.

Alle informatie die tot nu toe verzameld zijn samen met de interviews wordt gebruikt om een architectuur te maken. De architectuur is geen permanent tekening van de uiteindelijke applicatie maar meer een dynamisch middel om het eindresultaat mee te kunnen visualiseren. Verder wordt binnen deze sprint een start gemaakt aan een Azure teaching programma waar de basiselementen van Azure uitgelegd wordt in een series video’s. Deze programma wordt gedurende het project gevolgd om zo veel mogelijk te kunnen leren over Azure.

Binnen deze sprint wordt gewerkt aan de beroepstaak A2 Informatie vergaren, analyseren & verwerken & C1 Ontwerpen software. A2 wordt bewezen door informatie te kunnen verzamelen die zal dienen tot het ontwikkelen van de architectuur. C1 wordt bewezen door een architectuur te ontwerpen en de feedback van de stakeholders in de architectuur te verwerken. Ten slotte wordt de beroepstaak Gf bewezen door een teaching program te volgen over Azure.

## 6.1 Architectuur ontwikkelmethode

In dit hoofdstuk wordt de totstandkoming van de architectuur gedocumenteerd. De architectuur is gemaakt met een architectuur ontwikkelmethode. Jongens van Techniek heeft geen ontwikkelmethode waar een verplichting aan zit en zal dus geen belemmerende factor zijn gedurende keuzeproces. De methodes, Attribute-Driven Design (ADD), Domain-Driven design (DDD) en C4 zijn bekend omdat ze eerder in het afstudeerprogramma zijn behandeld en zullen worden beoordeeld voor gebruik binnen het project.

Domain Driven design is gericht op grote projecten en op complexe projecten waarbij het domein niet bekend zijn. Hiervoor wordt samengewerkt met domein experts die het domein goed kennen (Kunal, 2020). Er wordt verwacht dat het domein lastig te begrijpen is voor ontwikkelaars waardoor experts erbij komen kijken om alle begrippen uit het domein te kunnen uitleggen die vervolgens door de ontwikkelaars geïmplementeerd kan worden.

Nadeel van DDD is dat het gebruikt zal worden voor projecten waarbij domeincomplexiteit hoog is. Daarnaast is het niet goed voor projecten die technisch lastig in elkaar zitten bijvoorbeeld systemen die met meerdere componenten gekoppeld zijn. Dit zal voor het monitoringsysteem niet handig zijn omdat meerdere externe systemen gekoppeld zal worden aan het monitoringsysteem. De complexiteit hiervoor zal vermeden worden.

ADD richt zich op het ontwikkelen van een architectuur op een grotere schaal en complexere technische limitatie. Doordat je nauw werkt met de klant is het makkelijk om op één lijn te zitten met de klantbehoeften.

Door alle requirements eerst te verzamelen zorgt ADD voor dat de volledige scope in architectuur om te zetten is. Hiervoor is er volledigheid nodig van de klant. Dit vergt veel inspanning vanuit de klant of de bedrijfsmentor die niet altijd beschikbaar kan zijn. Daarnaast kan er blijken bij ADD dat sommige technieken niet haalbaar zijn. Dit kan zorgen voor vertraging of kan achteraf blijken dat de requirement niet in het architectuur past. Verder eist ADD dat de requirements van te voren goed gedocumenteerd zijn. Dit kan niet het geval zijn doordat de lange termijn oplossingen voor de monitoring app nog niet bekend zijn.

Met C4 kan een abstracte level gemaakt worden tussen systemen. Doordat de Monitoringsapp andere applicaties zal monitoren die nog niet bekend zijn is het beste om deze abstracte level mee te kunnen nemen. De flexibiliteit om in te zoomen op componenten zorgt ervoor dat vanuit een groter perspectief naar het project gekeken kan worden voordat verder de technische opties overwogen kunnen worden. De Ubiquitous language die gebruikt kan worden binnen C4 kan ervoor zorgen dat de verschillende stakeholders de visuale grafieken zullen begrijpen die gedurende het ontwikkelproces gecreëerd zal worden.

C4 is niet goed te gebruiken bij projecten waarbij het domein te lastig is. Externe partijen zullen zonder introductie de visuele aantekeningen niet snappen zonder instructies en introductie waardoor het lastiger zal worden voor nieuwe teammembers van het project. Als laatst is het lastig om de modellen die ontwikkeld zijn te hergebruiken omdat je dan sommige levels compleet opnieuw moet maken.

## 6.2 Keuze architectuur ontwikkelmodel

De uiteindelijke keuze voor het ontwikkelmodel voor de architectuur wordt C4. De beslissing is tot stand gekomen door de limitaties van het project mee te wegen. Dit is doordat het een klein project is en dat niet alle requirements duidelijk waren vanaf de start.

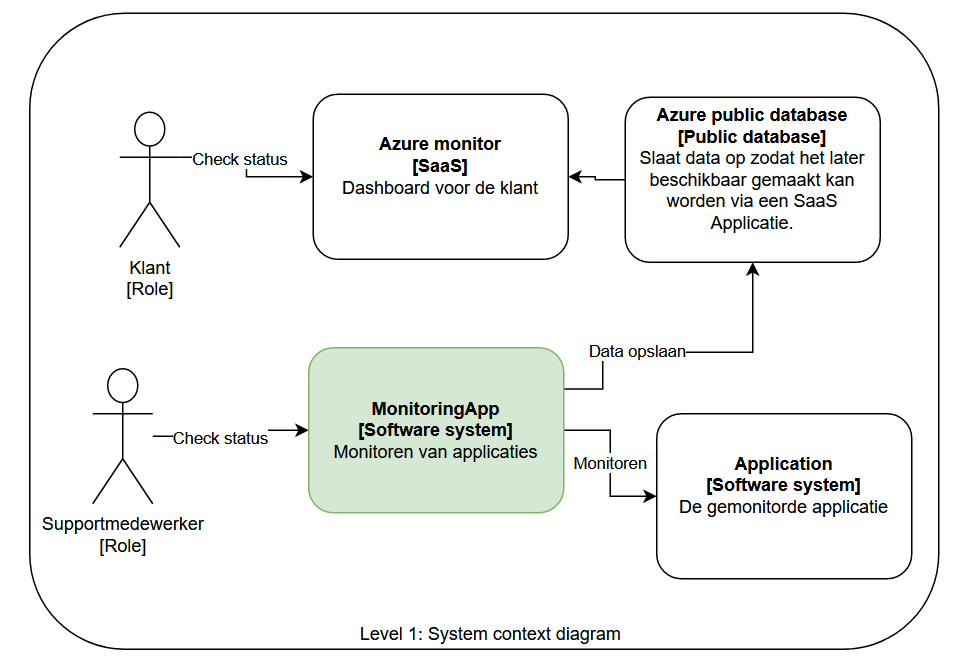
Het project eist een deel kennis over Azure en de componenten daarin. Doordat eerst een uitgezoomd perspectief getekend kan worden is C4 in dit geval handiger omdat niet gelijk alle technieken geïmplementeerd hoeft te worden. Dit betekent dat de architectuur in een latere sprint aangepast kan worden.

## 6.3 C4 model

Het C4 model bestaat uit 4 levels. System context, Container, Component en Code. Dit zijn de standaard levels die zich bevinden binnen het C4 model. Deze vier levels kunnen een architectuur creëren maar daarnaast zijn er nog ondersteunende levels dat geldt binnen het C4 model. Voorbeelden zijn een System landschape diagram, Dynamic diagram of een deployment diagram.

### 6.3.1 Level 1 Systeem Context Diagram

Startend vanuit de level 1, oftewel de ‘System context diagram’ wordt geanalyseerd welke systemen het project mee te maken zal hebben. In deze diagram worden naast het systeem, actoren gemodelleerd om vast te stellen welke stakeholders interactie hebben met de MonitoringsApp. Deze actoren zijn de primaire stakeholders van het systeem. Als laatst wordt de interactie tussen de systemen gevisualiseerd.



Figuur 3 Level 1 System context diagram

In figuur 3 is te zien dat de MonitoringApp groen getekend is, gekoppeld aan een applicatie. Deze applicatie is de gemonitorde software. Hiervan uit is op te maken dat er vanuit de monitoringsapp een verbinding is naar de applicatie. Dit betekent dat de gemonitorde applicatie niet van de hoogte is dat de MonitoringApp bestaat.

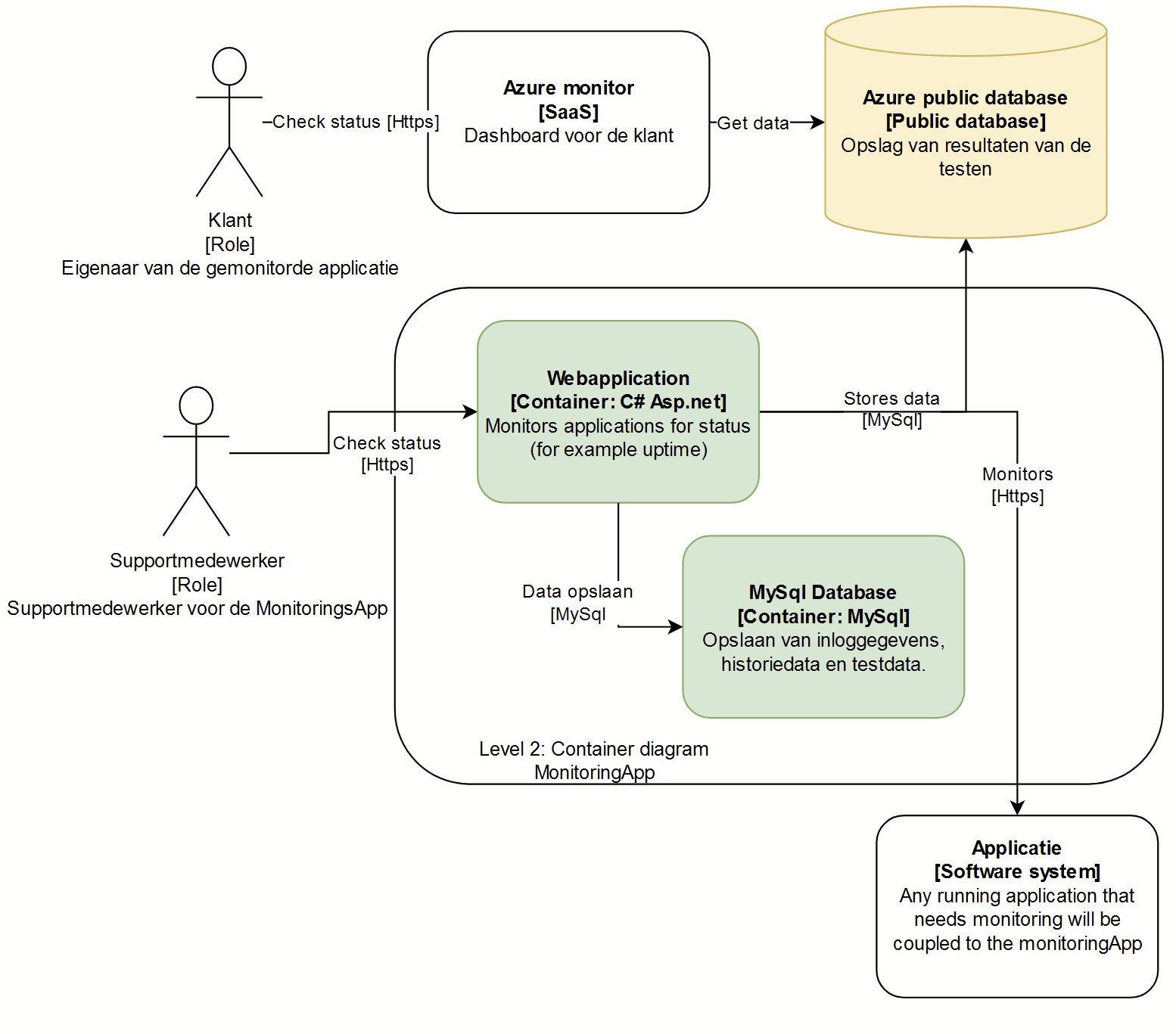
De twee actoren die hierboven zijn opgenomen zijn de klant (de eigenaar van de gemonitorde software) en de supportmedewerker. De klant kan via een dashboard allerlei informatie opvragen. De supportmedewerker kan dit doen via de 'MonitoringsApp' om ervoor te zorgen dat de status van de tests goed zijn.

|  |  |
| --- | --- |
| Component | Beschrijving |
| Klant | De eigenaar van de gemonitorde software die een onderhoudscontract is aangegaan bij JvT (Jongens van Techniek) |
| Supportmedewerker | De supportmedewerker bij UptimeHeroes. Deze persoon beheert de tests en kan de statussen inzien. |
| Application | De gemonitorde applicatie. De monitoringsapp zal deze applicatie monitoren om statussen te kunnen controleren. |
| MonitoringsApp | De applicatie dat ontwikkeld zal worden gedurende project. Deze 'MonitoringsApp’ monitort de applicatie die een onderhoudscontract is aangegaan. |
| Azure Public Database | Data vanuit de tests dat is verkregen zal opgeslagen worden in een public database. |
| Azure Monitor | De Azure Monitor is een dashboard voor de klant die een dashboard krijgt via de Azure Monitor applicatie. |

Tabel 6: Systeem context diagram beschrijving

### 6.3.2 Level 2 Container Diagram

Een stap verder naar level 2, oftewel de ‘Container diagram’. Binnen deze level wordt de focus gelegd op de groene componenten van Level 1 (figuur 3). De verschillende componenten worden gevisualiseerd en de onderlinge relaties worden blootgelegd. Een container kan zijn een systeem, database, mobiele applicatie of een resource binnen het systeem. Doordat er binnen het project meerdere containers zijn, zijn de relaties tussen deze belangrijk en zal zorgvuldig gevisualiseerd moeten worden.



Figuur Level 2 Container diagram

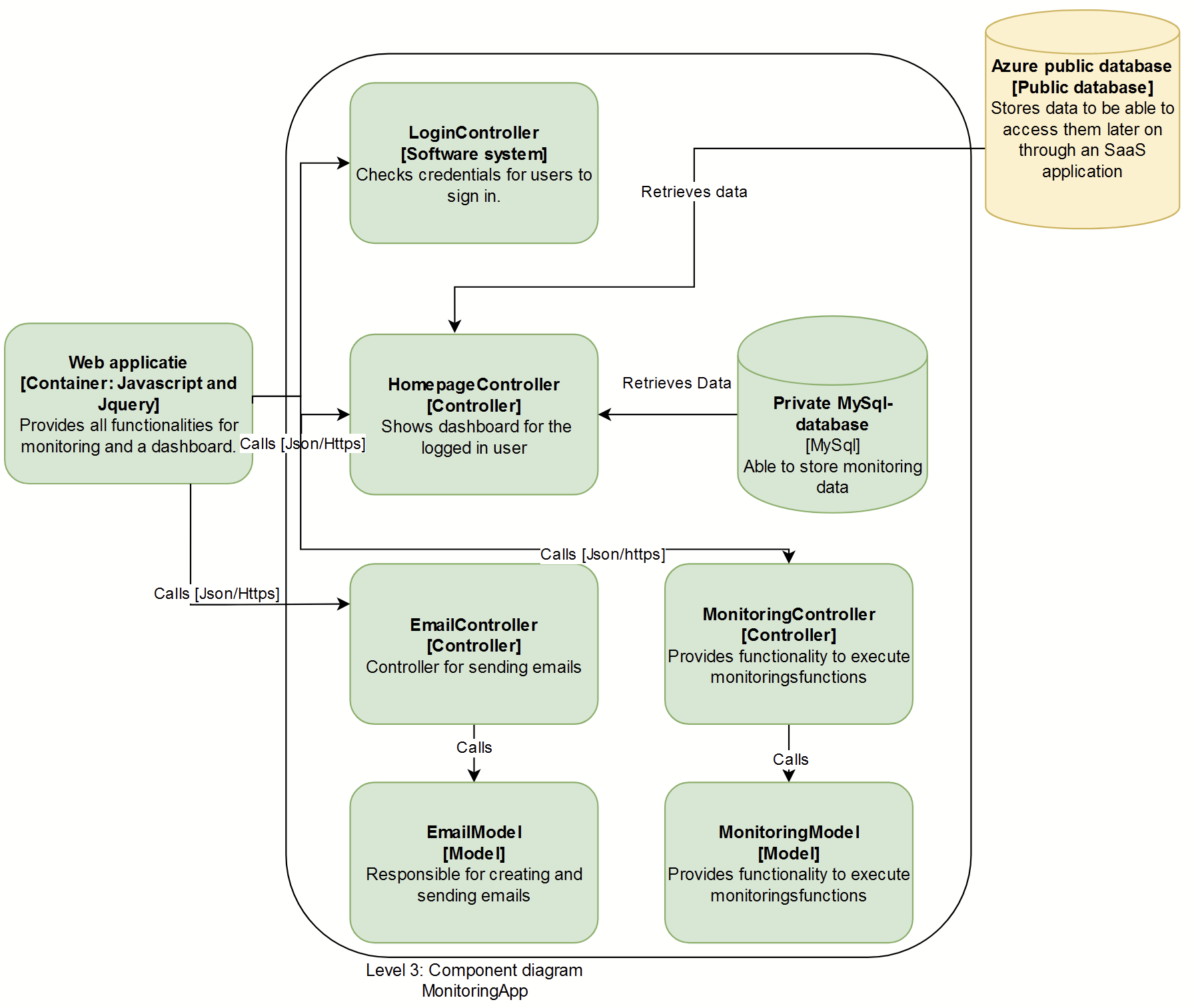
In figuur vier is te zien dat de webapplicatie verbonden is met twee databases. Het geel gemarkeerde element is een publieke database waarin de resultaten van de tests in opgeslagen wordt. Deze opslag gebeurt door de webapplicatie. De webapplicatie heeft daarna nog een privé database waarin andere gegevens in worden opgeslagen zoals inloggegevens, wachtwoorden en test gegevens. Deze gegevens mogen niet publiekelijk beschikbaar zijn waardoor deze beslissing voor scheiding is genomen. De publieke database heeft vervolgens nog een functie. Deze publieke database wordt gebruikt door een SaaS-applicatie. Deze applicatie is een dashboard voor de supportmedewerker om alle testresultaten gevisualiseerd te kunnen zien.

|  |  |
| --- | --- |
| Component | Beschrijving |
| Klant | De eigenaar van de gemonitorde software die een onderhoudscontract is aangegaan bij JvT (Jongens van Techniek) |
| Supportmedewerker | De supportmedewerker bij UptimeHeroes. Deze persoon beheert de tests en kan de statussen inzien. |
| Webapplicatie | De monitorings-applicatie zelf. Deze applicatie zal gebruikt worden om andere applicaties te kunnen monitoren. |
| MySql Database | De database dat gebruikt zal worden voor verschillende data dat niet publiekelijk beschikbaar mag zijn zoals inloggegevens en testdata. |
| Applicatie | De gemonitorde applicatie. |
| Azure public database | Resultaten van de tests worden opgeslagen in een public database. Deze wordt daarna gebruikt om een dashboard van te maken in een SaaS-applicatie. |
| Azure Monitor | De Azure Monitor is een dashboard voor de klant die een dashboard krijgt via de Azure Monitor applicatie. |

Tabel 7: Container diagram beschrijving

### 6.3.3 Level 3 Container Diagram

Daaropvolgend zal level 3 worden gevisualiseerd, oftewel de ‘Component diagram’. In deze level zal verder worden gegaan op de groene component in Level 2. Hier zijn twee groene componenten te vinden waaronder de database die in het diagram opgenomen zal worden. Verder zijn deze componenten de benodigde onderdelen om de applicatie te implementeren. Deze kunnen controllers zijn, database modellen of algoritmes die calculaties uitvoert. Deze componenten zijn in principe dichter bij de code en kunnen als een hoge level klassendiagram gezien worden. Wat deze level niet bevat zijn de functies waaruit een component bestaat.



Figuur Level 3 Component diagram

Links in figuur vijf is het startpunt van de webapplicatie. Deze webapplicatie is het front-end voor een gebruiker en zal gebruikt worden door de supportmedewerker. De login controller checkt voor authenticatie en zal de inloggegevens goed of afkeuren. Na inlog zal de gebruiker doorgestuurd worden naar de homepage controller waar het dashboard te vinden is.

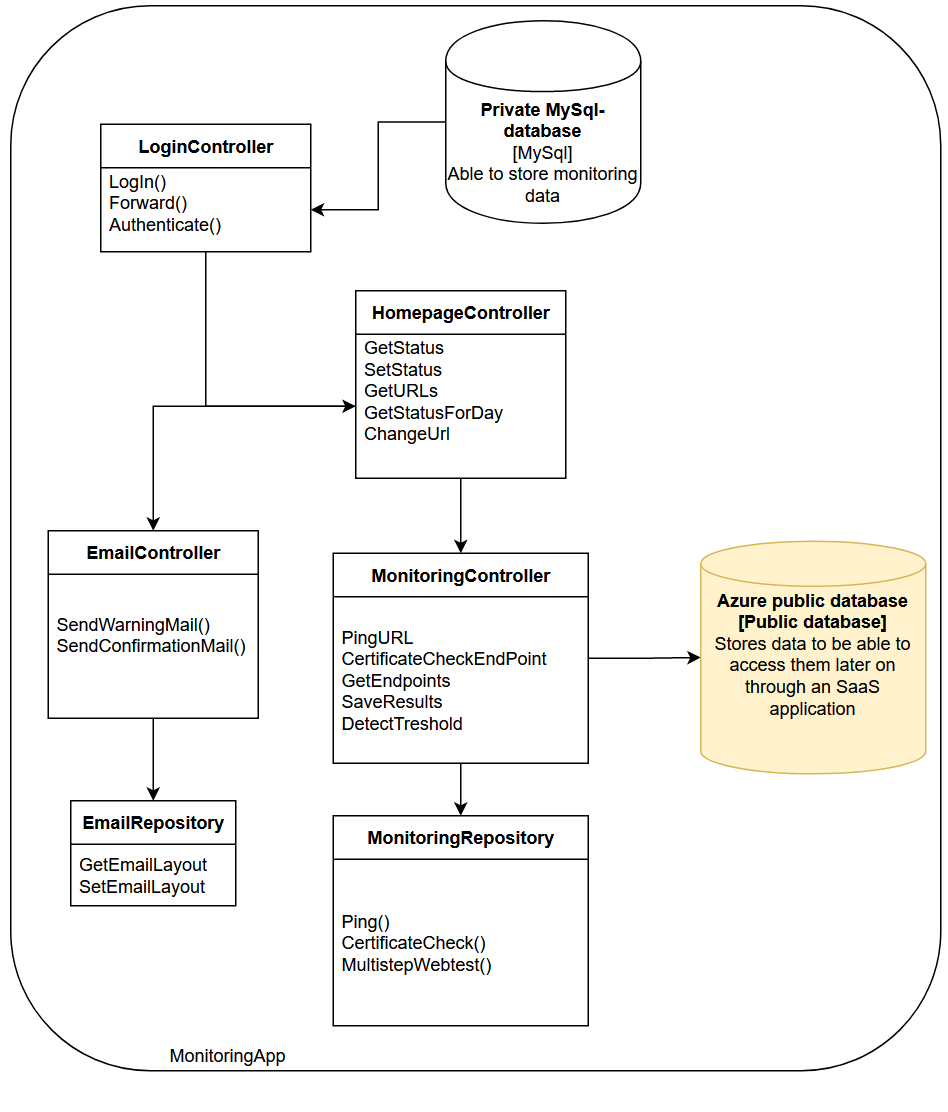
Op deze dashboard zijn de meest belangrijke gegevens te zien van alle gemonitorde applicaties. Mocht er fouten zijn opgetreden dan komt de email controller van pas. Hieruit worden de e-mails verstuurd naar de desbetreffende personen.

In de twee databasen worden de benodigde informatie opgeslagen zodat deze later opgevraagd kan worden. Deze informatie zal verder ook gebruikt worden voor het modeleren van een grafiek met testresultaten.

|  |  |
| --- | --- |
| Component | Beschrijving |
| Web Applicatie | De monitorings-applicatie zelf. Deze applicatie zal gebruikt worden om andere applicaties te kunnen monitoren. |
| HomepageController | De controller voor het dashboard |
| LoginController | De controller voor het controleren van de login gegevens |
| Private Mysql-Database | De database dat gebruikt zal worden voor verschillende data dat niet publiekelijk beschikbaar mag zijn zoals inloggegevens en testdata. |
| EmailController | De controller dat gebruikt zal worden om het versturen van e-mails te kunnen starten. |
| EmailModel | Het model waarin de e-mails gestructureerd zullen worden aan de hand van het doel van de e-mail. |
| MonitoringController | De controller voor het monitoren van de applicaties. Deze controller zal het monitoren aansturen. |
| MonitoringModel | Een model voor het uitvoeren van de tests. |
| Azure public database | Resultaten van de tests worden opgeslagen in een public database. Deze wordt daarna gebruikt om een dashboard van te maken in een SaaS-applicatie. |

### 6.3.4 Level 4 Code Diagram

De functies en voornamelijk code dat voorspeld wordt nodig te zijn om functionaliteiten te kunnen implementeren valt onder deze level. Deze level is daarom een klassendiagram van de componenten in level 3, ‘Code diagram’. Per component zal een klassendiagram uit voortkomen. Doordat de code continu kan veranderen zal deze klassendiagram een dynamisch leven hebben en leven gedurende het project.



## 6.4 Azure AZ-900 Fundamentals

Binnen Jongens van Techniek is er een mogelijkheid om certificaten te behalen. Eén van deze certificaten is de Azure AZ-900 certificaat. Deze programma is een examen die genomen kan worden om fundamentele kennis over Azure te kunnen beheersen.

Deze examen is niet technisch en bevat oppervlakkige vragen over Azure die een persoon zonder technische achtergrond kan beheersen. Dit is handig om te kunnen starten met Azure. Er zijn meerdere boeken en leermateriaal beschikbaar voor deze examen waaronder een video course beschikbaar binnen Jongens van Techniek.

Deze videocourse is een 113 delige course waarin alle fundamentele informatie over Azure wordt verteld. Informatie zoals Azure Blueprints, Azure policy, Access policy, Azure portal, Private cloud, hybrid cloud en nog veel meer. Alle video’s hebben een gemiddelde lengte van 15 minuten en kunnen rustig bekeken worden. Binnen deze video’s worden een onderdeel van Azure behandeld zoals eerder genoemd.

Binnen deze course zijn er een aantal video’s over monitoring die veel uitleggen over gebruik van Azure Monitor. Hierdoor is verder gezocht naar informatie over Azure Monitor, Azure Application insight en Serverless. Door dit voor te leggen bij de bedrijfsmentor is besloten om hierover een onderzoek uit te voeren, Deze onderzoek dient om vast te stellen welke voordelen het heeft voor het project en vervolgens hoe het te gebruiken is (bijlage7).

## 6.5 Sprint review

### 6.5.1 Evaluatie

Om de sprint af te sluiten is er besloten om wekelijks een vergadermoment te houden. Tijdens deze vergadermoment wordt de afgelopen sprint besproken met de bedrijfsmentor.

Gedurende deze sprint heb ik gemerkt dat ik nog informatie mis om de architectuur mee te maken. Vooral over implementatiemogelijkheden merk ik dat ik niet de juiste kennis heb die ik nodig heb voor het project. Ik zal eerst moeten onderzoeken naar mogelijkheden binnen Azure voor de volgende sprint voordat ik verder aan implementatie zal beginnen. Om meer informatie te vergaren heb ik geprobeerd om een vergadering in te plannen met een stakeholder, alleen dat ging door meerdere redenen niet door. Ik ben verdergegaan met de architectuur met de informatie die ik tot nu toe had.

De retrospective was redelijk kort omdat er geen input was van beide kanten dus kon het gesprek al snel afgesloten worden.

Doordat ik niet al teveel informatie had over mogelijke oplossingen binnen Azure ben ik verder in de cursus gaan duiken. Dit had ik eerder moeten doen maar doordat ik niet wist dat deze mogelijkheid binnen JVT beschikbaar was. De teaching program kwam pas naar boven na het gesprek met Peter Hofman.

De AZ-900 course gaf mij vele mogelijkheden. Het heeft mij meer informatie kunnen bieden over de Azure omgeving. Verder is het een handige tool geweest om mogelijkheden voor Monitoring te kunnen zoeken. Ik kwam tegen Azure Application insight binnen de programma en heb deze gevolgd. Dit is een mogelijkheid om de functionaliteiten te kunnen implementeren voor het project.

# 7.0 Sprint 2: Onderzoek Serverless consumption & architecture

Gedurende vorige sprint is er meer informatie opgedaan conform de oplossingen voor het project. Voor de komende sprint wordt er geen stories binnen de sprint genomen. De beroepstaak Gc Kritisch en methodisch werken en Gf Leren leren wordt bewezen door een onderzoek te doen naar Application Insight en Serverless consumption in de komende sprint.

Het sprint doel voor de komende sprint is het onderzoeken van mogelijkheden waarop Serverless toegepast kan worden en de voordelen ervan onderzoeken. De voor en nadelen wordt gebruikt om te kunnen concluderen of de Serverless functie voordelig zal zijn voor gebruik binnen het project. Om dit te kunnen bewijzen zal een testproject worden gemaakt om een basis te hebben voor de implementatie.

## 7.1 Onderzoek Serverless

Uit de onderzoek in bijlage 6 is geconcludeerd dat gebruik van de Serverless optie een voordeel zal zijn voor het project. Tijdens het onderzoek wordt gekeken naar traditionele Dotnet MVC framework en vergeleken met de voordelen die Serverless biedt.

Gebruik van Serverless levert veel voordelen op. Het past binnen de use-case waarvoor een Function app gebruikt kan worden, oftewel event based. De tests die uitgevoerd zullen worden binnen de Monitoringapp zijn events die als triggers kunnen dienen. Dit geeft ons een goede use-case voor de function app. F

Verder is gebruik van Functions een voordeel bij kostenmanagement. Door gebruik van de downtime tussen triggers door hoeft niet altijd een server standby te staan om de code uit te kunnen voeren. Er zal alleen betaald moeten worden op het moment er gebruik gemaakt wordt van de servers om code uit te voeren.

Desondanks moeten de nadelen niet door de vingers glippen. Het grootste probleem voor de monitoringsapp zijn de sessies die verdwijnen na elke uitvoering van de Function App. Hierdoor kan er niet gewerkt worden met sessies binnen de applicatie waardoor scheiding van rollen binnen de applicatie niet mogelijk is.

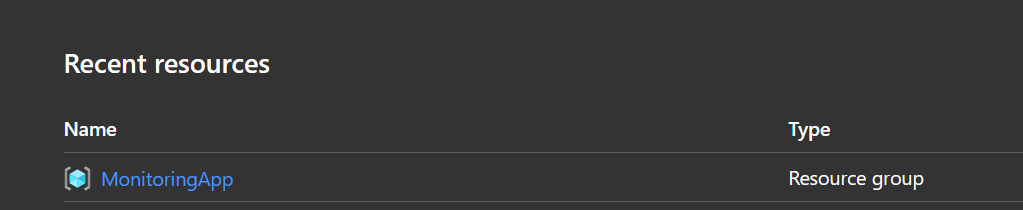
Alhoewel de nadelen impact hebben op de applicatie, zijn de voordelen in de meerderheid. Gebruik van Serverless zal het project ten goede doen waardoor is besloten om Serverless iin gebruik te nemen.

## 7.2 Testpoject Serverless

Om serverless te kunnen testen en ervaring op te doen is er een testproject gemaakt. Binnen deze testproject zijn er een aantal basis elementen getest zoals Code op de cloud krijgen en een debugger kunnen koppelen aan het project.

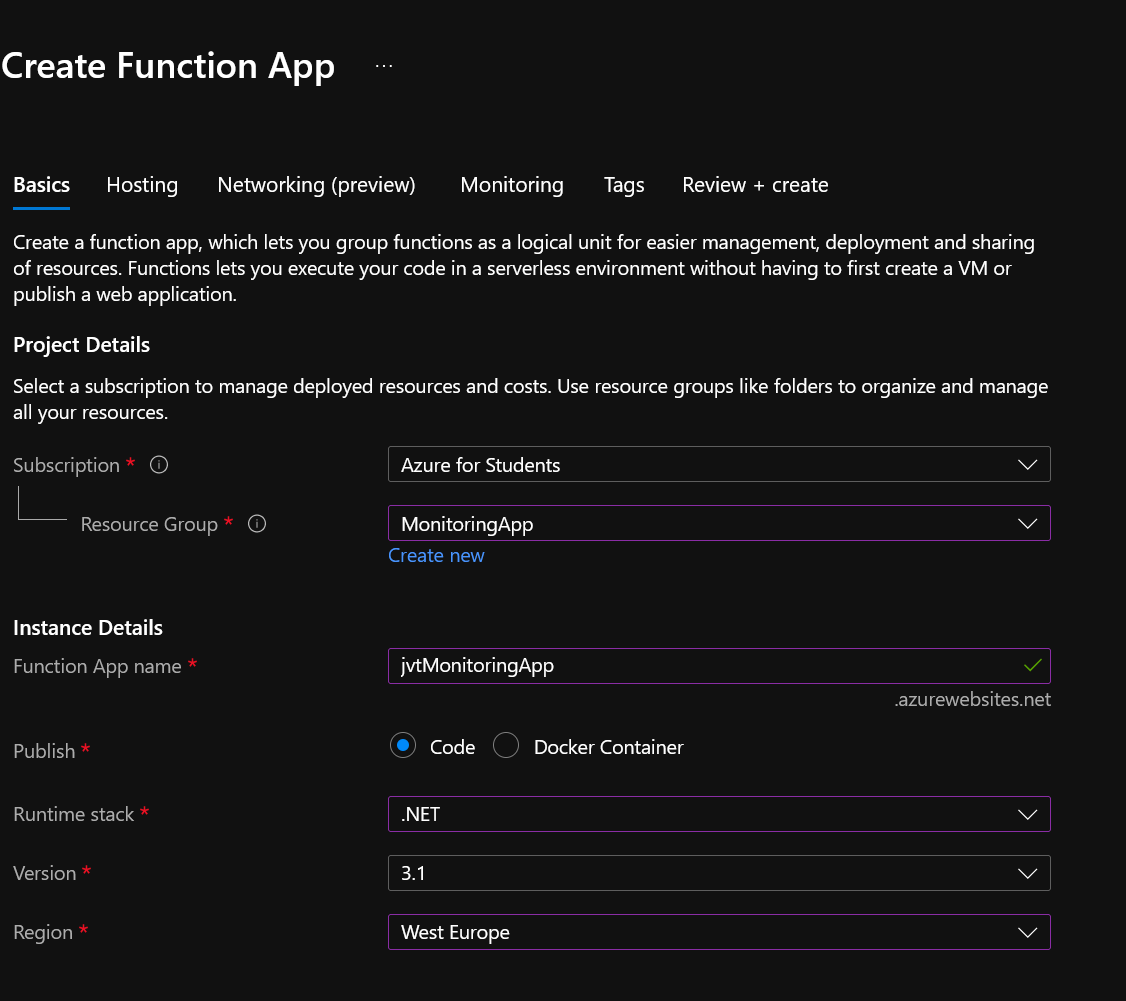
### 7.2.1 Aanmaken Azure functions resource

De Azure Functions resource moet aangemaakt worden in de Portal om een start te kunnen maken aan het test project. Hiervoor is een resourcegroup aangemaakt (figuur 6) binnen de Azure portal om vervolgens de Azure functions erin te plaatsen. Qua instellingen is niets verandert buiten dat de locatie is verplaatst naar West-Europa.



Figuur Resourcegroup

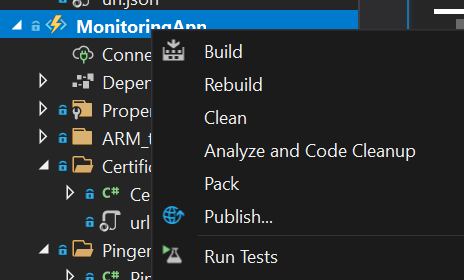
Binnen de resourcegroup is er een Function App aangemaakt. Bij het aanmaken van de Function app is gekozen om een .net runtime aan te maken met de meest recente non-preview versie. Doordat preview versies een experimentele versie is, is er gekozen om voor de meest recente niet-preview versie te gaan. In dit geval was dat versie 3.1.



Figuur Aanmaken Function App resurce

### 7.2.2 Publishen

De volgende stap is het uploaden van de testproject naar de cloud. Hiervoor wordt een publishing profile voor aangemaakt binnen Visual studio. Door op Publish (Figuur 8) te klikken wordt de gebruiker verstuurd naar de publishing profiel.



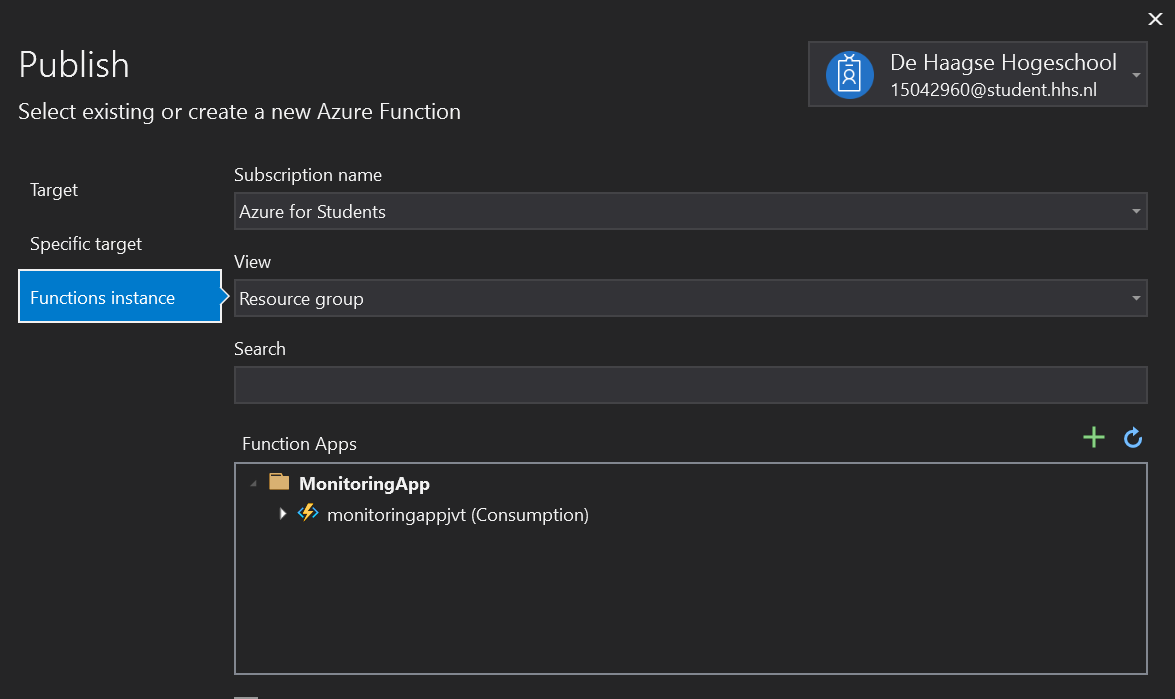
Figuur Publish project

Bij een leeg overzicht krijgt de gebruiker een wizard te zien waarin een publishing profile in aangemaakt kan worden.



Figuur Aanmaken Publishing profile

Hier wordt gekozen voor Azure en dan Function app (Windows) waarna een overzicht vertoond wordt van de ingelogde gebruiker. Deze ingelogde gebruiker moet hetzelfde gebruiker zijn als de account waarin de Azure Function in aangemaakt is. Wanneer dit het geval is, wordt er een overzicht vertoond met alle Function Apps binnen de portal. Hier wordt de Azure Function app vertoond die eerder in hoofdstuk 6.2.1 is aangemaakt.

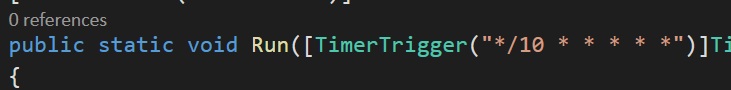


Figuur Koppelen function App resource

Na dit geselecteerd te hebben is het mogelijk om de App te publishen naar de cloud toe.

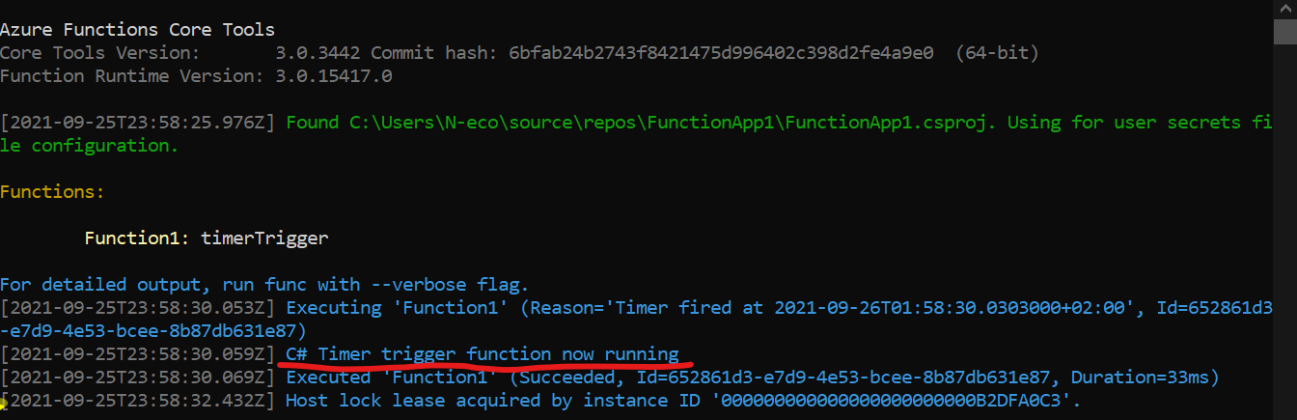
### 7.2.3 Runnen van de Function App

Het is mogelijk om de Function App lokaal uit te voeren (Microsoft, 2018). Visual studio heeft de mogelijkheid om de Functions lokaal te testen. Wel is het belangrijk dat de trigger geactiveerd moet worden. Doordat het testproject een timer trigger heeft, zal gewacht moeten worden, zodat de function via tijd geactiveerd kan worden. Hiervoor is er besloten om de timer op 10 seconden te zetten. Dit betekend dat de applicatie elke 10 seconden geactiveerd wordt (figuur 11).



Figuur Timer trigger

Na de timer aangepast te hebben, is het mogelijk om de applicatie te runnen. Binnen de run methode is er een log geplaatst om output te zien. Deze wordt vertoond in de output scherm van de applicatie.



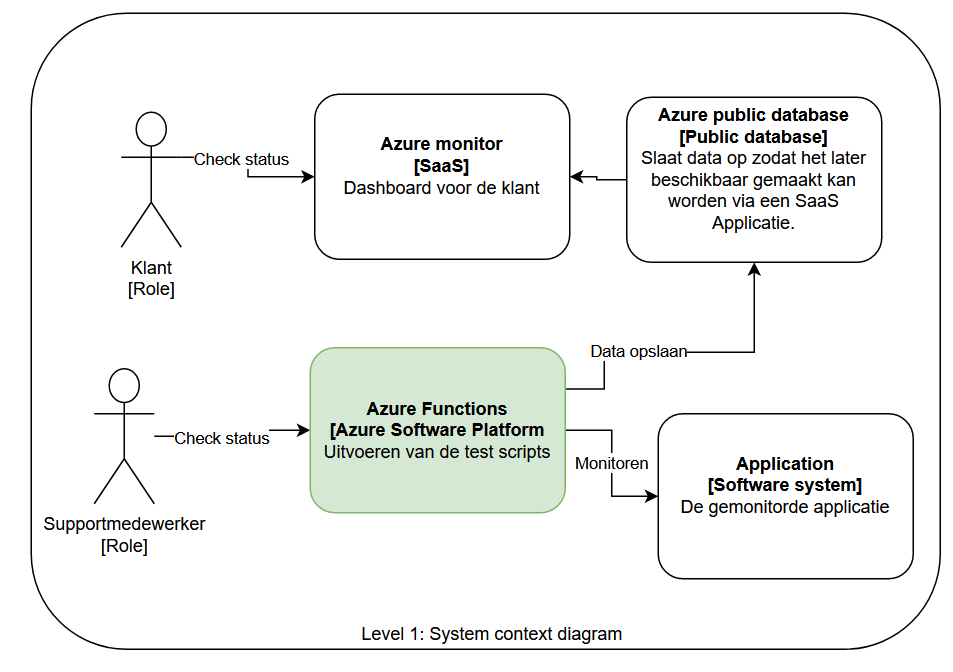
Figuur Debugging Function app

## 7.3 Architectuur

Door de wijziging aan de framework van de applicatie, is de architectuur niet meer up to date. Deze zal aangepast worden om te voldoen aan de nieuwe situatie. Hiervoor wordt de C4 opnieuw gebruikt en zullen de aanpassingen aangebracht worden. Deze aanpassingen worden vervolgens gereviewd door de stakeholder en projecteigenaar.

### 7.3.1 Level 1

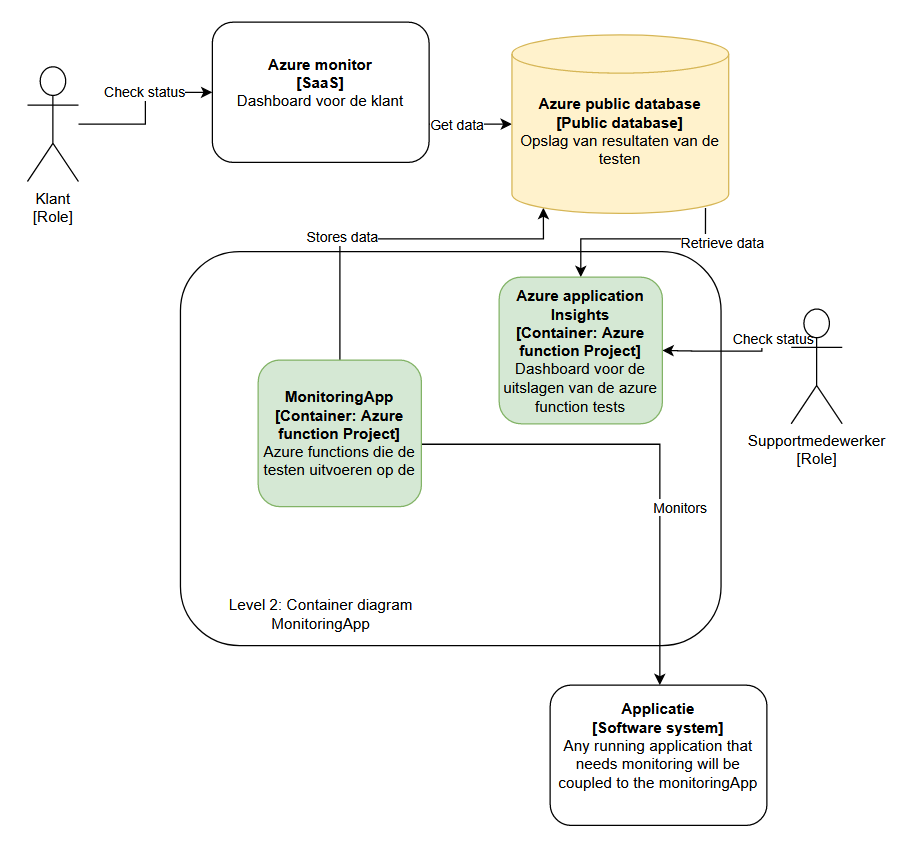
Aan de level 1 diagram is alleen de software platform gewijzigd. Het groene blok in figuur 13 was voorheen de Software systeem (H5.3.1). Deze is vervangen door Azure Functions die de monitorings functionaliteiten zal overnemen.



Figuur Level 1 System context Diagram

### 7.3.2 Level 2

Binnen level 2 zijn de groene blokken gewijzigd. Het groene blok MonitoringApp was gekenmerkt als C# ASP.net applicatie (H5.3.2). Deze is vervangen door een Function App, oftewel Azure Functions. Verder is hier Azure Application Insight aan toegevoegd om data heen te sturen.

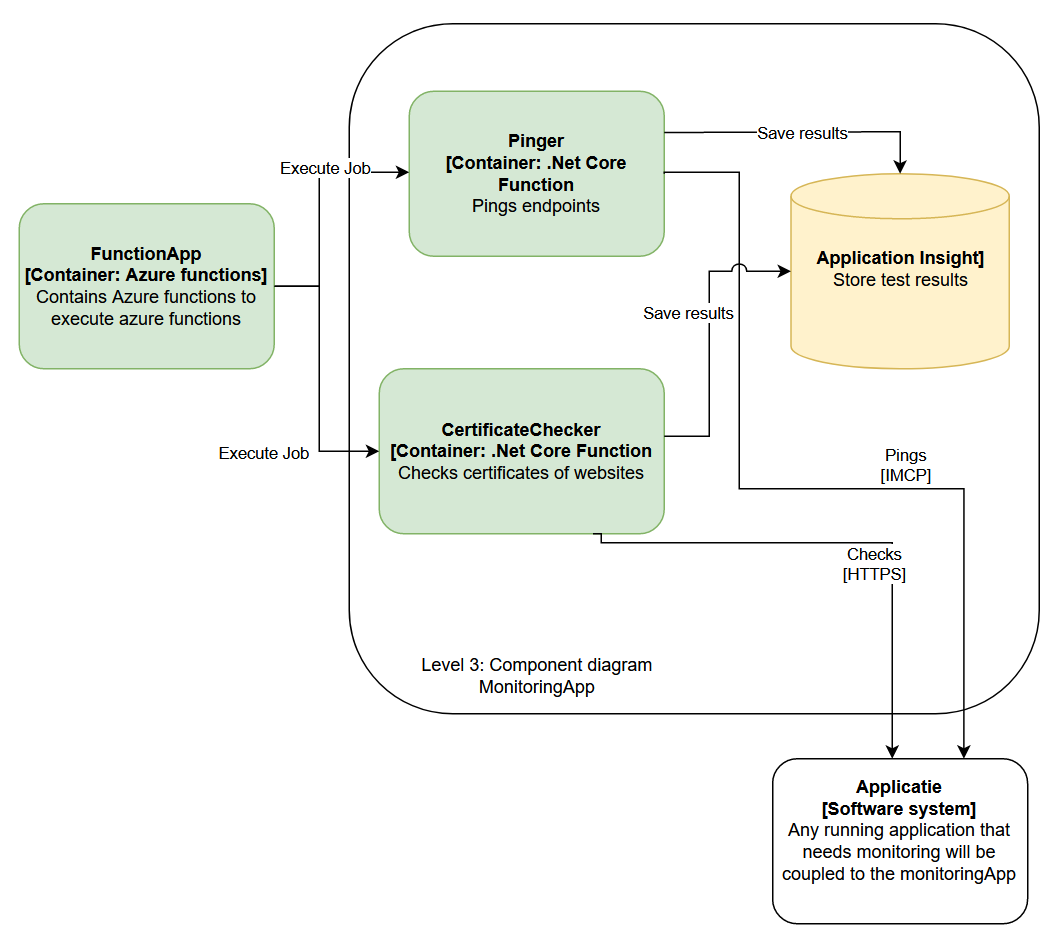


Figuur Level 2 Container diagram

### 7.3.3 Level 3

De grootste wijziging heeft plaatsgevonden in level 3. Dit komt doordat level 3 meer ingezoomd is op de applicatie dan het hele systeem. Alle controllers en modellen die te zien zijn in hoofdstuk 5.3.3 zijn verwijderd omdat deze obsolete zijn door de wijziging. Deze zijn verplaatst door een Function.

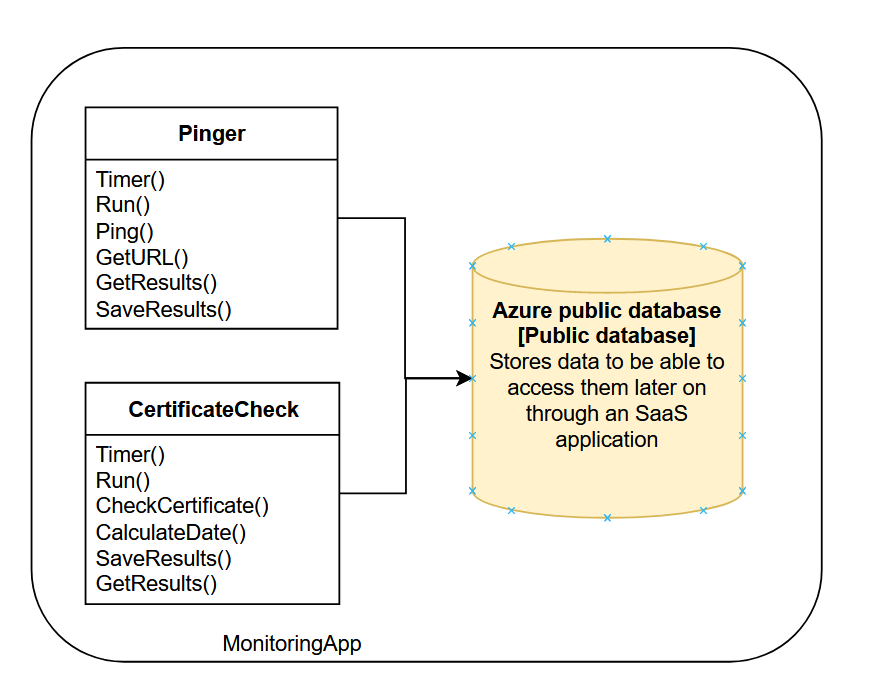
Per project zijn er meerder Functions aan te maken. Dit betekend dat binnen de Azure Function app resourcegroup die aangemaakt is in 6.2.1, meerdere functions ontwikkeld kunnen worden. Elk van deze functions kunnen individueel aangeroepen worden waardoor de indeling in figuur 15. Als laatst is het opslagpunt gewijzigd naar Application insight. De testresultaten worden in een publieke database opgeslagen zodat deze uitgelezen kan worden door Azure Application Insight.



Figuur Level 3 Component diagram

### 7.3.4 Level 4

Door grote wijzigingen in level 3, wordt in level 4 een aantal componenten gewijzigd en verwijderd. De voorspelling is dat per Function alleen een aantal functies nodig zijn om het af te kunnen krijgen. Door de twee tests die tot nu toe bekend zijn binnen het project zijn er voor de twee Functions een klassendiagram gemaakt. Deze classes zijn verbonden met de public database omdat de Functions direct data erheen versturen.



Figuur Level 4 Code diagram

## 7.4 Sprint review & retrospective

Voor sprint 2 zijn de sprint doelen behaald. Alleen heeft dit veel tijd gekost. Doordat ik niet alle informatie tot mijn beschikking had, was ik snel begonnen met een architectuur waarna ik later op een betere oplossing ben gekomen. Dit heeft geconcludeerd in vertraging in het project. Desondanks zijn de wijzigingen die worden aangebracht aan het project goed te beargumenteren. Dit is vooral het geval door de architectuur levels te vergelijken van oud en nieuw (oude architectuur H5.3, nieuwe architectuur 6.3). Na een klassendiagram gemaakt te hebben bleek veel minder code te vereisen om het uiteindelijke resultaat te kunnen behalen.

### 7.4.1 Evaluatie

Afgelopen sprint zijn geen user stories binnen de sprint genomen maar was er wel sprake van een sprint doel. Het doel was het onderzoeken naar Serverless consumption en de architectuur aanpassen mocht dat nodig zijn.

Uit het onderzoek naar Serverless Consumption is gebleken dat het beter is om Serverless te gebruiken voor de Applicatie. Hierdoor is de oude architectuur obsolete. Dit kon beter door eerst door eerst een beter vooronderzoek te doen naar mogelijkheden binnen Azure of de teaching program te volgen gevolgd door de interviews.

De architectuur moest aangepast worden omdat nieuwe informatie is gevonden die gedurende het project gebruikt kan worden. Ik wist wel dat Application Insight bestond maar doordat Functions een nieuw concept is geweest was het voor mij niet eerder te voorzien van de mogelijkheden die het te bieden heeft.

Door onderzoek te doen naar Serverless Consumption worden de beroepstaken Gf Leren leren & Gc Kritisch en methodisch werken. het onderzoek naar Serverless was lastig omdat het concept nieuw was (jaar of twee). Hierdoor moest ik zoeken naar betrouwbaar materiaal rondom Serverless Consumption. Hierbij ben ik meestal gestuit op bronnen van Microsoft zelf die informatie biedt op en hoog niveau. Desondanks was informatie over specifiekere elementen binnen Function wel schaars.

# 8.0 Sprint 3: Pinger

Na het opzetten van een het function App project is de volgende stap het implementeren van een functionaliteit. De user story 1 “Als supportmedewerker wil ik inzicht hebben op downtimes zodat ik weet of er een probleem zich afspeelt.” is binnen de sprint genomen.

Het doel voor de komende sprint is het opleveren van een Pinger functionaliteit die een URL zal pingen.

Om de pinger te kunnen implementeren is er gezocht naar oplossingen binnen C#. Samen met de bronnen is de pinger functionaliteit ontwikkeld die een URL zal pingen. Later is gebleken dat Azure functions het pingen limiteert door beveiligingsredenen waardoor de Ping op een andere manier uitgevoerd moest worden. Hiervoor is een vervanger voor gevonden en opnieuw geïmplementeerd. Gedurende de sprint wordt de beroepstaak D1 Realiseren software bewezen.

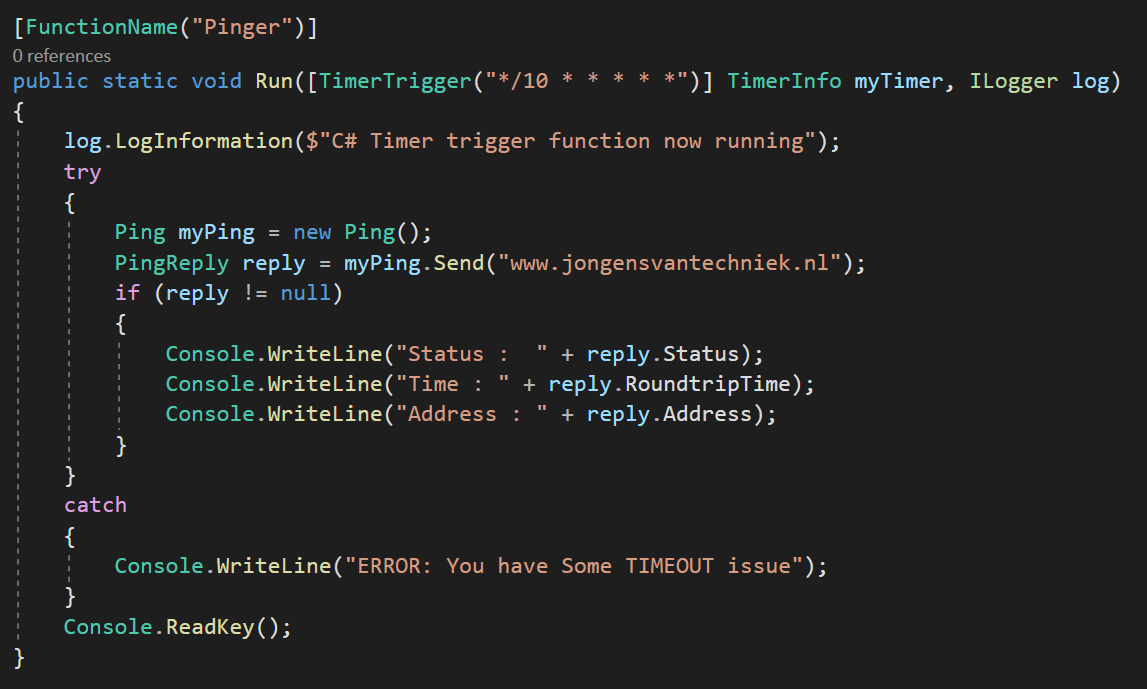
## 8.1 Aanpak

Te beginnen is er onderzocht welke opties beschikbaar zijn binnen C# om een ping uit te kunnen voeren. Vanuit meerdere bronnen is ervoor gekozen om een bron te gebruiken van Microsoft. Voor de pinger wordt gebriukt gemaakt van de ping class (Docs Microsoft, 2021).

De Ping class is een interne class binnen C# dat gebruikt kan worden om pings mee uit te voeren. Pings zijn kleine requests naar een server toe. Binnen deze request zit een kleine datapakket die zich gedraagt als een echt request. Wanneer de ping bij de server aankomt, wordt het behandeld en weer teruggestuurd naar de server. Hierdoor is het mogelijk om een aantal informatie uit te kunnen halen. Eén van de belangrijke informatie is de roundtriptime, oftewel de miliseconden het heeft geduurd voordat het datapakket heen en weer is geweest. Bij terugkomst kunnen de resultaten uitgelezen worden om deze verder te kunnen bestuderen.

## 8.2 Werkzaamheden

Het gebruik van de Ping class is simpel. Door de Ping class aan te maken en een endpoint (een url) te geven is het mogelijk om een ping uit te voeren. Voor de endpoint wordt een placeholder gebruikt omdat er geen gebruikers zijn van de monitoringApp. In dit geval wordt de homepagina van JVT gebruikt om het te kunnen testen.



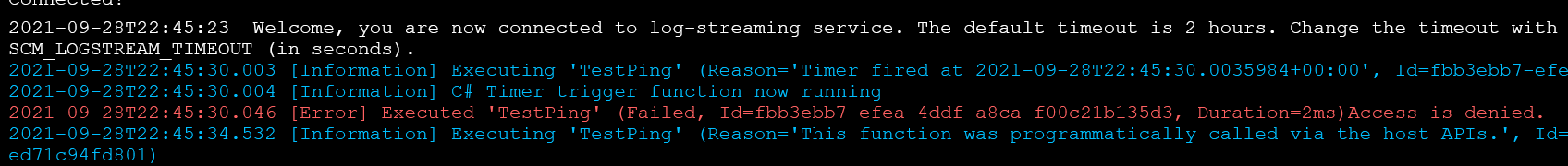
Figuur ping functionaliteit

### 8.2.1 Timeout

In de bron wordt gebruik gemaakt van een timeout van 1 seconden. Dit betekend dat de request na één seconde inactiviteit verloopt mocht de request verkeerd gaan. Hierdoor worden alle requests na 1 seconden, oftewel 1000 ms bij 1 seconden al direct stoppen. Alles na 1 seconden zal niet gerekend worden binnen de roundtriptime. Om een accurate meting te krijgen is ervoor besloten om geen gebruik te maken van de timeout. Hierdoor kan elke abnormaliteit gedetecteerd worden en zal niet afgesneden worden door de timeout.

### 8.2.2 Cloud restrictions

Na de code gerund te hebben op de cloud was het tijd om de code te kunnen testen via Azure Functions. Door gebruik te maken van de Portal is de code handmatig uitgevoerd. Het resultaat is niet wat er was verwacht. De foutmelding geeft aan dat toegang tot uitvoering van de testPing function is geweigerd.



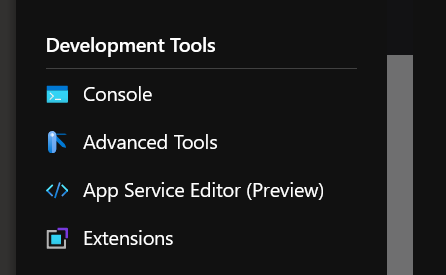
Figuur Denied ping

Het weigeren van de ping komt voort uit een restrictie die Azure gelegd heeft op Azure Functions (Keith, 2017). Er wordt aangegeven dat Azure Functions geen toegang heeft om ICMP protocol te gebruiken om pings te sturen naar een buitenstaande adres. De rede hierachter is doordat de Ping request een typisch DDOS aanval is waardoor het probleem is opgelost door de ICMP protocol compleet te blokkeren is.

### 8.2.3 Workaround TCMP

Uit een vraag binnen stackoverflow blijkt dat er een workaround beschikbaar is voor dit probleem (PedroC88, 2019). Het gebruik van een TCPPing is wel toegestaan binnen Azure Functions waardoor dezelfde functionaliteit behouden kan worden.

Een Ping tool is beschikbaar vanuit Microsoft die gebruikt kan worden voor installatie binnen Azure functions (Microsoft, 2016). Deze tool is te downloaden als zip bestand beschikbaar op de Microsoft documentatie website. Deze tool moet worden geüpload binnen Azure Functions. Dit is mogelijk via de Advanced Tools binnen de Function resource (Fiuur 19).

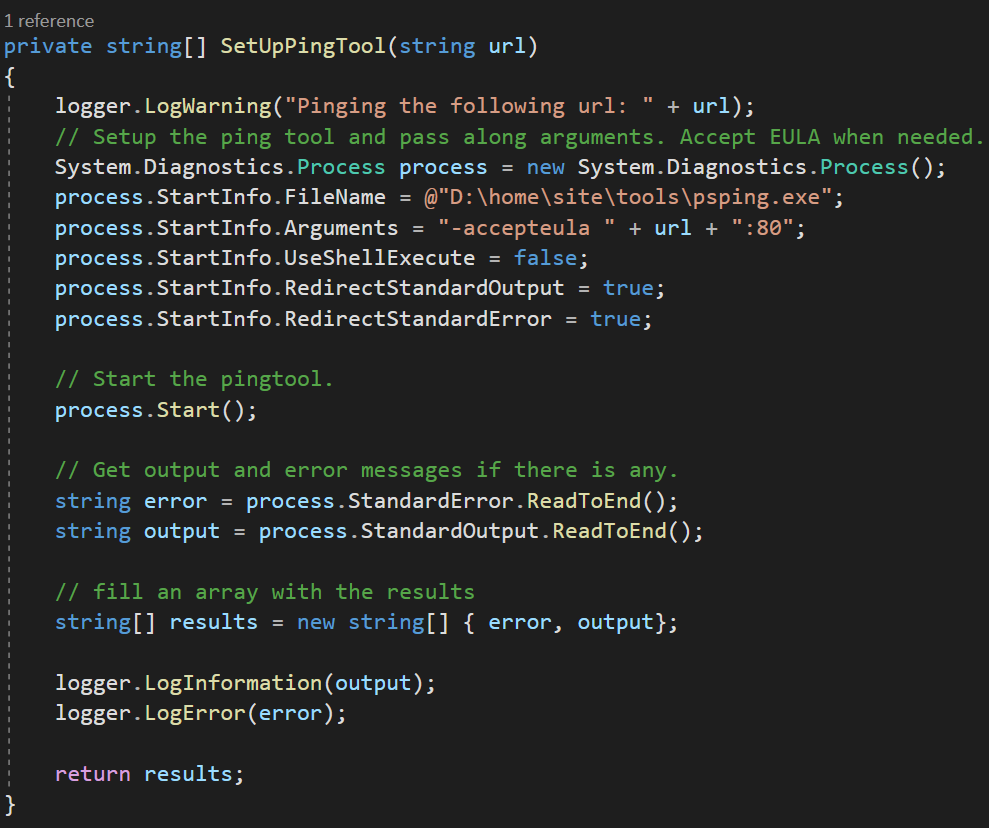


Figuur Development Tools Functions App

Binnen de Advanced Tools is een Debug Console te vinden met een File explorer. Hier wordt een Tools folder aangemaakt waarna de bestanden hierin geplaatst kunnen worden.

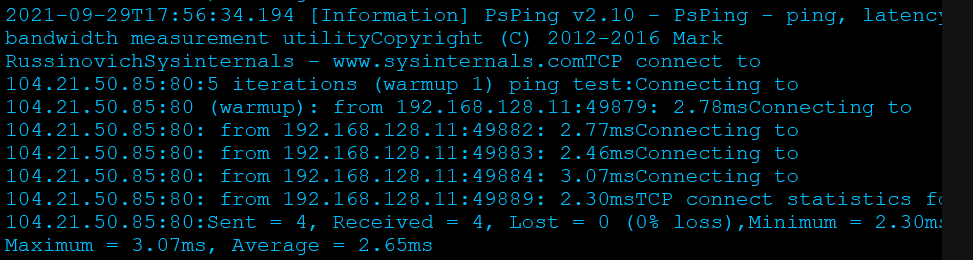
### 8.2.3 Gebruik van PsPing

Om gebruik te kunnen maken van de tool zal binnen de code de pad naar de tool gevonden moeten worden. De padnaam kan achterhaald worden door dezelfde padnaam te gebruiken die aangehouden wordt binnen de Developer tools file explorer. Hierin worden een aantal extra argumenten gegeven zoals -acceptEula om geen belemmeringen te ondervinden. Verder worden de outputs naar een veriabele verwezen die later gebruikt worden om foutmeldingen te kunnen beheren.



Figuur Code voor gebruik Psping

Hierna is nogmaals de Function handmatig uitgevoerd. De output van de Psping is een stuk string met informatie over de uitgevoerde ping zoals de minumum, maximum en gemiddelde miliseconden. De TCP ping voert standaard 4 pings uit per opdracht. Vanuit de vier pings wordt de gegevens opgeslagen en teruggestuurd naar de client als een string. Binnen deze string zijn er een aantal onnodige stukken tekst die niet gebruikt hoeven te worden.



Figuur Output Psping

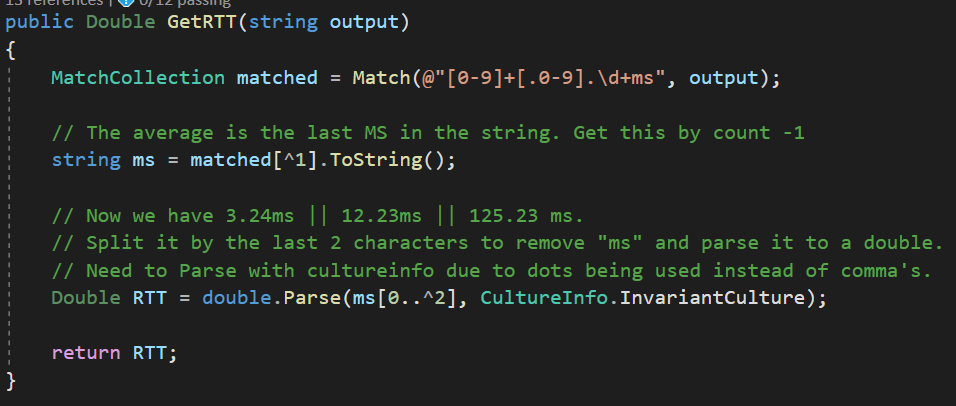
De laatste regel van figuur 21 is een overzicht van de ping. Informatie zoals aantal verzonden datapackets, lost packets, minumum MS, maximum MS en average MS zijn hier te vinden. Er is besloten om gebruik te maken van de Average MS omdat die de gemiddelde miliseconden aangeeft van alle vier pings.

## 8.3 Verwerken output PsPing

Na besluit om de gemiddelde miliseconden te gebruiken voor opslag zal de string verwerkt moeten worden om de data uit de blok string te kunnen halen. Om door de string heen te kunnen gaan is besloten gebruik te maken van Regex. Regex is een tool waarmee delen uit een tekst mee gedetecteerd kan worden op patronen. De volgende stap hierbij is het detecteren van de juiste stuk tekst binnen de string.

Om eerst vast te stellen of de data hetzelfde blijft is de Function voor een aantal minuten laten lopen. Elke 10 seconden wordt de output van de ping geprint. Er is opgemerkt dat elke keer de aantal miliseconden voor elke meting onderscheid wordt door een punt. Voor en na de punt bevinden zich getallen. Deze situatie is ook getest met getallen met een nul achter de komma om vast te stellen dat deze niet afgerond worden tot hele getallen. Getallen zoals “2” of “3” worden gerepresenteerd als “2.00” of “3.00”. In dit geval blijft de punt en de twee nullen staan.

Detectie zal plaatsvinden op basis van deze informatie. In figuur 22 is de Regex terug te vinden. Deze Regex detecteert alle decimaalgetallen waarin een punt te vinden is en eindigt op “MS”. Er wordt gezocht op getalen van alle lengtes voor de komma omdat er niet vastgesteld kan worden hoe lang de ping is. Dit kan zijn 700 milliseconden of één milliseconden.



Figuur Regex pinger

Alle gedetecteerde worden binnen een collection geplaatst. Doordat de volgorde hier belangrijk is, hoeft alleen de laatste van de array gehaald te worden. Vervolgens wordt deze gesplit op de letter “ms” en worden de decimalen geconverteerd naar een double variabel. De output is nu klaar om opgeslagen te worden. Gezien de Function de data niet in geheugen kan houden, wordt deze data voor nu vernietigd na elke uitvoering van de Functions.

## 8.4 Sprint review & retrospective

De user story 1 “Als supportmedewerker wil ik inzicht hebben op downtimes zodat ik weet of er een probleem zich afspeelt.” die binnen de sprint genomen is, halverwege afgerond. Daarbij hoort dat de sprintdoel ook halverwege is gebleven omdat opslag nog niet plaatsvindt.

Gedurende sprint review is gekeken naar de code om te zorgen dat kwaliteit hoog gehouden wordt. Mede-ontwikkelaars hebben hun mening gegeven over de code en is hierbij ook een static code analysis uitgevoerd. Hierbij is feedback geleverd op benamingen en gebruik van functies binnen de applicatie. Aan de hand van deze feedback zijn de desbetreffende benamingen aangepast. Dit zorgt ervoor dat medeontwikkelaars de code begrijpen en dat het over het algemeen te begrijpen is voor iedereen. Dit vormt ook de basis voor de volgende sprint waarin het testplan gemaakt wordt voor kwaliteitsbewaking.

### 8.4.1 Evaluatie

Bij het maken van de pinger functionaliteit is er een cloud restrictie over het hoofd gezien waardoor het uitvoeren van de ping niet mogelijk was. Dit kwam doordat er een protocol geblokkeerd werd waardoor toegang tot een ping niet mogelijk was. Dit resulteerde in vertraging van de User Story. Dit kon vermeden worden door eerst onderzoek te doen naar pingmogelijkheden binnen Azure Functions.

Binnen de sprint is wel uiteindelijk de user story geïmplementeerd nadat een vervanger voor is gevonden. De vervanger voor de ping was het gebruiken van een Ping tool dat pings op basis van de TCP protocol kon uitvoeren. Gelukkig kon ik goede bronnen vinden die mij hierbij konden helpen en heb ik een goed product kunnen leveren.

# 9.0 Sprint 4: Testplan & alerts & Sending ping data to Application Insight

Na een functionaliteit af hebben gemaakt is er besloten om zo zorgvuldig mogelijk voortgang te boeken zonder enige bugs te introduceren. Hiervoor is kwaliteitsbewaking nodig. Gedurende spint 4 zal het testplan gemaakt worden.

Er worden twee user stories binnen de sprint genomen. Gedurende sprint 4 zal gewerkt worden aan alert notificaties en data die ontvangen worden van de tests.

|  |  |
| --- | --- |
| # | User story |
| 1 | Als supportmedewerker wil ik inzicht hebben op downtimes zodat ik weet of er een probleem zich afspeelt. |
| 4 | Als supportmedewerker wil ik een email ontvangen mocht er een downtime optreden. |

Gedurende het ontwikkelen van US 4 en 1 wordt de beroepstaak D1 Realiseren software mee bewezen. Daarnaast wordt de beroepstaak D2 Testen en Evalueren bewezen bij het maken van een testplan.

Komende sprint is kwaliteitsbewaking en de Pinger Function aan de beurt. Gedurende het project zal kwaliteitsbewaking het project ten goede doen doordat er voortdurend aanpassingen gedaan worden aan de applicatie. Dit betekend dat er belemmeringen of bugs geïntroduceerd kunnen worden binnen de applicatie die te vermijden zijn.

Buiten dat wordt er verder gewerkt aan de pinger functionaliteit om opslag te implementeren zodat data niet kwijt geraakt wordt. Daarna wordt gewerkt aan een notificatiesysteem waarbij gebruikers op de hoogte gehouden kunnen worden bij bijzonderheden.

## 9.1 Testplan

Om een testplan te kunnen ontwikkelen voor het project is eerst gezocht naar voorbeelden van verschillende testplannen. Dit is gedaan doordat er voorheen geen ervaring of kennis opgedaan is op dit onderdeel. Met de voorbeeld die is gevonden is een lijst opgesteld met wat er in een testplan hoort.

Eerst werd het testplan gemaakt waar uitgegaan werd van een DotNetCore MVC applicatie. Na aanpassingen hierbij is het testplan aangepast naar de nieuwe situatie.

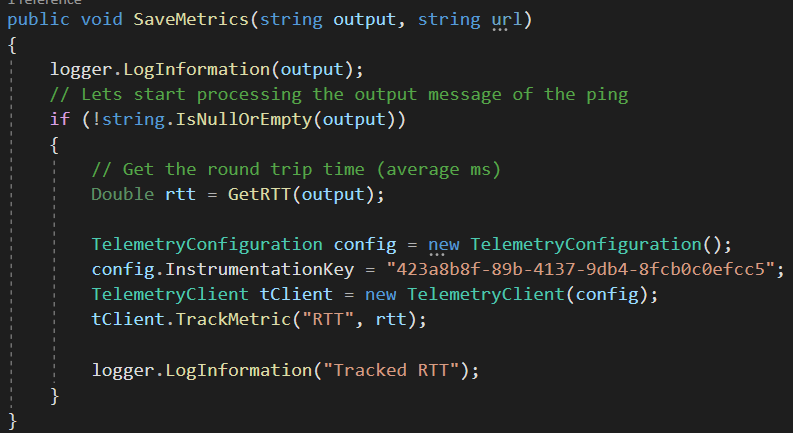
Vanuit de eerste perspectief was besloten om geen gebruik te maken van Unittesting doordat Azure Functions gebruik maakt van de cloud om te kunnen runnen. Door Serverless was eerst de gedachte dat de tests niet gerund kunnen worden. Desondanks is de advies van Microsoft om wel gebruik te maken van unittesting (Microsoft, 2019). Hierdoor is besloten om gebruik te maken van regressietests. Dit is gedaan om te kunnen ontwikkelen zonder dat oude gedrag van de applicatie niet zal veranderen en geen ongewenste bugs kunnen introduceren.  
  
Om de unittests te kunnen implementeren is gebruik gemaakt van tools. Er is een testproject aangemaakt met Xunit volgens de Microsoft bron waar gesuggereerd werd om Xunit te gebruiken (Microsoft, 2019). Verder is gebruik gemaakt van IntelliSense. Hiermee is mogelijk om code te verbeteren volgens guidelines die opgesteld zijn door de community en goedgekeurd zijn door Visual Studio.

## 9.2 Ping data sturen

In sprint 3 is de output van de pinger functionaliteit verwerkt tot een double object. Doordat de data niet opgeslagen wordt, wordt deze vernietigd zodra de Functions is afgelopen. Om deze data te kunnen opslaan wordt gedurende de Functions een koppeling gemaakt met Application Insight waarna de data heen verstuurd wordt voor lange termijn opslag.

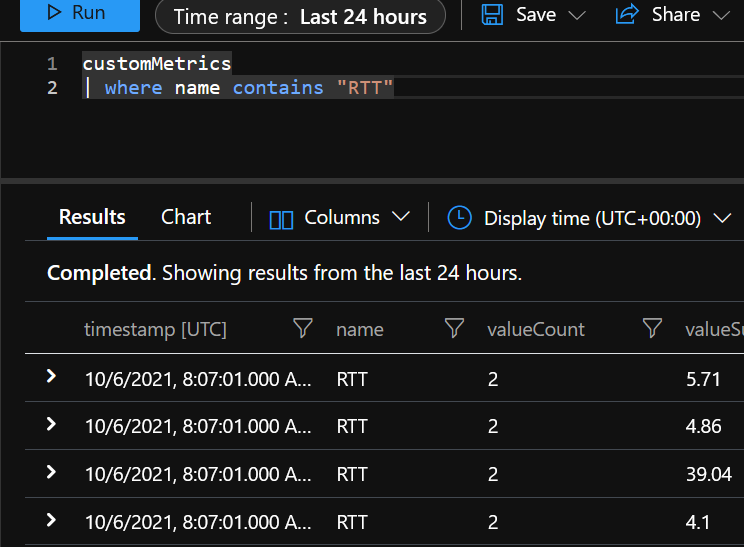
Te beginnen is gekeken naar bronnen die te maken hebben met het opslaan van data binnen Application Insight. Hiervoor is gebruik gemaakt van Microsoft documentaties. Uit de Application Insight metric documentatie (Microsoft, 2020) is gebleken dat gebruik gemaakt wordt van “TelemetryClient” object dat data verstuurt naar Application Insight.

Door een functie genaamd TrackMetric aan te roepen met twee parameters is het mogelijk om data te kunnen versturen naar Azure Application Insight. Om data te kunnen versturen is gebruik gemaakt van de voorbeeld die terug te vinden is in bijlage 6. Hier is te zien hoe de “TrackMetric” methode gebruikt wordt om data te kunnen versturen naar de database.



Figuur Codeopslag data

De “Instrument Key” vormt de koppeling tussen de applicatie en de Application Insight resource. Hiermee weet de applicatie naar welke resource data zal worden verstuurd. Na het runnen van de applicatie, zijn de logs terug te vinden onder de logs binnen Azure Application Insight. Dit geeft aan dat opslag plaatsvindt.



Figuur Logs data

Er is gekozen voor de naam RTT omdat dit deze waarde de “RoundTripTime” representeert. Dit is de tijd die de data packet nodig heeft om heen en weer te gaan. Een afkorting hiervan is gemaakt en wordt gebruikt om data van de pinger op te slaan.

### 9.2.1 Obsolete TelemetryClient

Gedurende ontwikkeling is erachter gekomen dat er een Obsolete melding wordt weergegeven. Deze waarschuwing is afkomstig door gebruik van de TelemetryClient object. Uit een bron van Github (Imolkova, 2019) blijkt dat het instantiëren van de TelemetryClient obsolete is en dat het moet werken via de Dependancy Injection. Dit komt doordat de Client alleen één keer geïnstantieerd moet worden via een Singleton. Op de achtergrond is deze verandering al doorgevoerd waardoor binnenkort gebruik van deze object niet meer mogelijk zal zijn. Het is voorlopig werkende maar er zal een vervanger gevonden moeten worden voor de TelemetryClient implementatie.

## 9.3 Alerts

Wanneer de ping functie gedurende het testen met een hogere waarde komt dan 150, zal een melding naar desbetreffende gestuurd moeten worden. Deze personen kunnen dan reageren op de melding en proberen het probleem op te lossen.

Om deze melding te kunnen versturen zal gebruik gemaakt worden van Azure. Binnen Application insight is er een optie om meldingen toe te voegen aan metrics. Deze meldingen zijn volledig aanpasbaar naar eigen beheer. Daardoor is besloten om gebruik te maken van Metrics. Het kan direct een mail sturen naar de behorenen in het geval een metric een waarde overschrijd. Er hoeft geen emailserver voor opgezet worden of een emailtemplate voor gemaakt worden. Daarnaast is het een onderdeel voor Application insight waar niet betaald voor hoeft te worden.

### 9.2.1 Alert toevoegen aan metric

Aangekomen bij Application Insight zijn alerts te vinden onder monitoring. Om een alert aan te maken zal eerst een “rule” aangemaakt moeten worden.

Een “Rule” bestaat uit vier delen:

1. Scope
2. Condition
3. Actions
4. Alert rule details

*Scope*

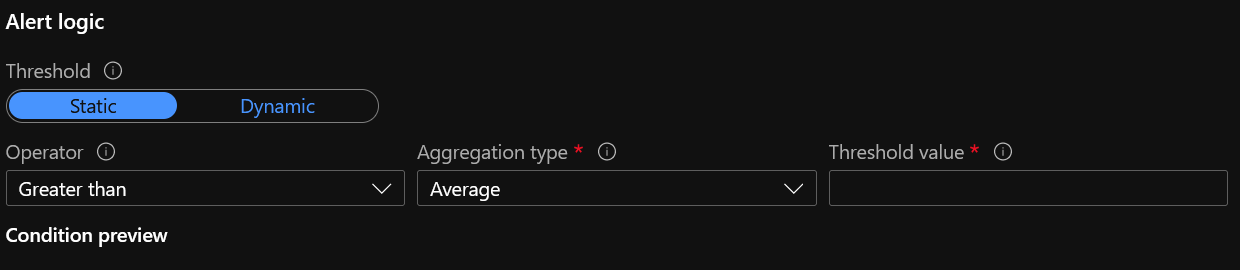
Binnen de scope wordt gekozen voor de resource Application Insight. Hierbinnen wordt de metric gekozen die vanuit Functions gestuurd wordt naar Application Insight (bijlage 7), Binnen de scope kan gezocht worden naar de desbetreffende metric waarop de alert geplaatst zal moeten worden.

Hier binnen wordt gekozen voor Application Insight waar de data heen gestuurd wordt.

*Condition*

Binnen de condition wordt meer gedetailleerd gekeken naar de gekozen metric. Hier wordt bepaald hoe de trigger eruit zal zien.

Aan de hand van “threshold values” kunnen de grenzen bepaald worden van de metric (figuur 23). Hiermee wordt de grens bepaald van de metric waarna de alert wordt getriggerd. Als “Threshold value” is 150 gebruikt. Dit is besloten gedurende gesprek met Peter die aangaf dat alle pings na 150 een waarschuwing moet zijn (bijlage 1).



Figuur Alert logic

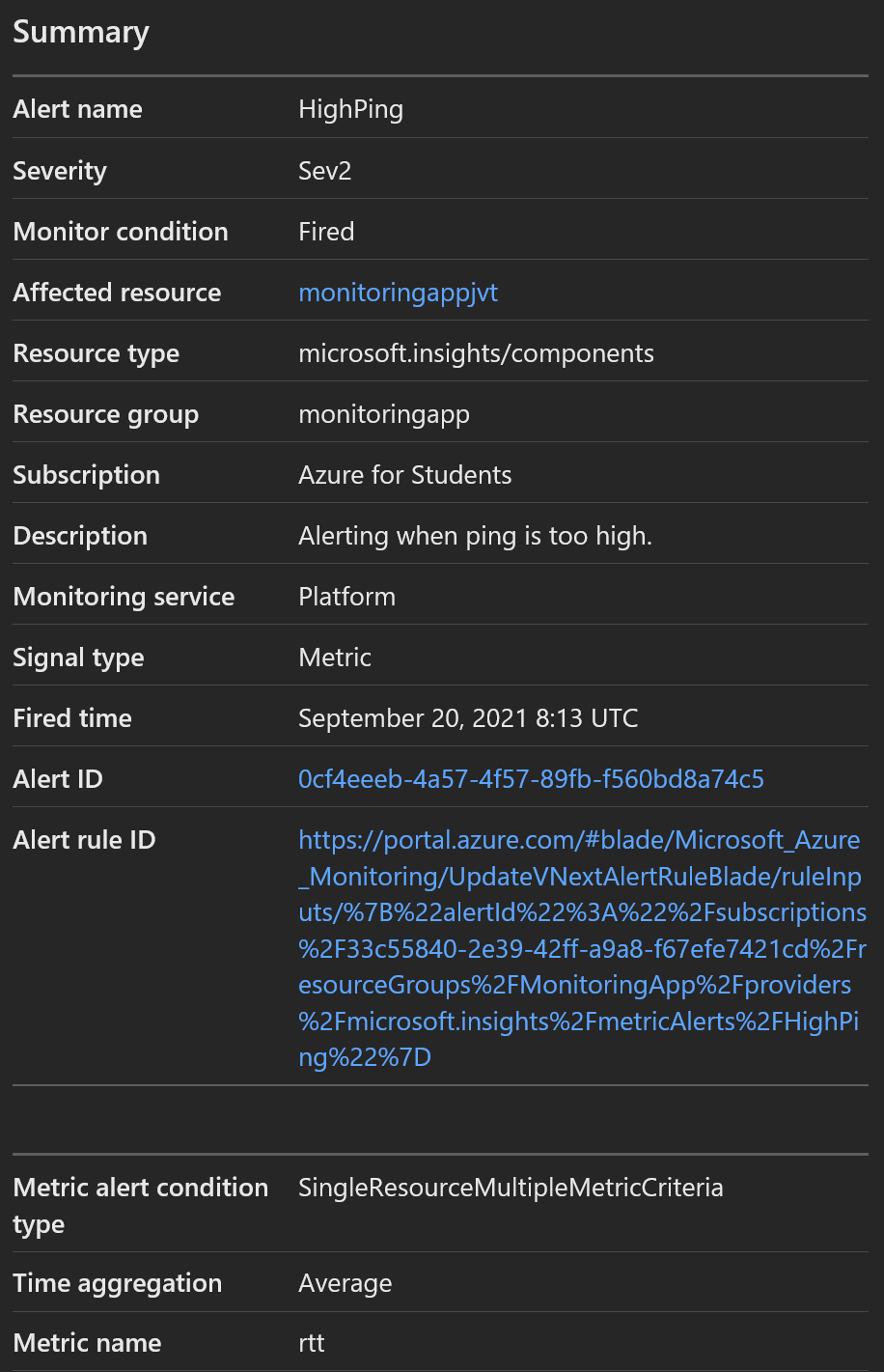
*Actions*

Ook wel Action group genoemd. Binnen de action wordt aangegeven welke acties uitgevoerd moeten worden zodra de grens is overschrijd. Wanneer de bedoeling is om meerdere personen te mailen over de overschrijding, kan dit gedaan worden binnen de actions. Voorlopig is gekozen om alleen een persoonlijke email te gebruiken waar meldingen heen verstuurd worden.

### 9.2.2 Email

Zodra de trigger wordt geactiveerd wordt er een mail gestuurd naar de personen die zijn toegevoegd aan de action group. De email bevat een aantal gegevens die aangeven waardoor ge alert is afgevuurd. Hieronder is een voorbeeld van de email.

Te beginnen wordt de severity aangegeven. Dit is de alert graad dat aan de alert is toegewezen. Vervolgens wordt de condition toegewezen aan de alert. Dit is altijd “fired” tenzij de metric daarna een waarde heeft ontvangen binnen de grenzen. Hierna zal de desbetreffende persoon een mail ontvangen waarin de condition op “Resolved” zal staan. Als laatst wordt de tijd weergegeven en de metric die het heeft veroorzaakt.



Figuur Alert summary

## 9.4 Sprint review & Sprint retrospective

*Review*

Gedurende sprint 4 zijn twee user stories afgemaakt. Dit waren:

|  |  |
| --- | --- |
| # | User story |
| 1 | Als supportmedewerker wil ik inzicht hebben op downtimes zodat ik weet of er een probleem zich afspeelt. |
| 4 | Als supportmedewerker wil ik een email ontvangen mocht er een downtime optreden. |

Beide functionaliteiten zijn afgemaakt in de afgelopen sprint. De Ping functie moest aangevuld worden met opslag die gedurende review op prijs gesteld werd. Nu is het mogelijk om meerdere testen aan Application Insights te koppelen met opslag zonder een SQL database nodig te hebben. Tijdens de review werd opgemerkt dat ook de kosten laag waren voor gebruik van een opslag waar mijn bedrijfsmentor blij mee was.

9.4.1 Evaluatie

Bij het koppelen van de Function met Application Insight kwam ik erachter dat ik een object gebruikt heb dat obsolete bleek te zijn. Na verder gezocht te hebben bleek dat dit een oude implementatie is, waardoor ik dit als technical debt heb behandeld. Om geen belemmeringen te hebben gedurende de sprint is de technical debt terug op de backlog geplaatst om er later terug op te komen.

Verder is er onderzocht om een notificatiesysteem te maken voor alerts binnen de Monitoringsysteem. Hier bleek een oplossing voor te zijn binnen Application Insight. Hierover is meer informatie verzameld om een zo compleet mogelijk beeld te verkrijgen van de mogelijkheden van Alerts (bijlage 7).

# 10.0 Sprint 5: Technical Debt, CertificateChecker & Application Insight Grafieken

Vorige sprint is de Pinger gekoppeld aan Application Insight. Nu is het mogelijk om data op te slaan die door de pinger geproduceerd wordt. Voor de komende sprint wordt er twee user stories binnen de sprint genomen. Naast de twee sprints, wordt de Technical Debt van de vorige sprint meegenomen om op te lossen.

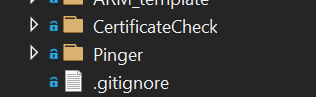
|  |  |
| --- | --- |
| # | User story |
| 2 | Als supportmedewerker wil ik inzicht hebben op verlopen certificaten, zodat ik weet of er een probleem zich afspeelt. |
| 4 | Als supportmedewerker wil ik een email ontvangen mocht er een downtime optreden. |
| - | Technical Debt |

Voor de komende sprint is de doel om de applicatie technical debt-free te krijgen en een certificaat test toe te voegen aan de applicatie zodat er meerdere tests beschikbaar zijn.

Voor de technical debt wordt naar een oplossing gezocht. Om dit op te lossen zijn gekeken naar beschikbare community resources die gebruikt kan worden om het probleem op te lossen.  
  
Daarnaast wordt gestart aan implementatie voor de certificatenchecker. Deze functionaliteit controleert de einddatum van certificaten. Het zal de einddatum vergelijken met de huidige datum en de verschil berekenen. Deze verschil (aantal dagen voor verloop certificaat) zal vervolgens opgeslagen worden in Application Insight.

## 10.1 Nieuwe Function

Om een nieuwe Function aan te maken in het project is besloten om een onderscheid tussen Functions te maken binnen de applicatie. Dit is gedaan doordat anders alle Functions in de root van de applicatie zal zitten. Hiervoor is besloten om alle Functions op te splitsen in een eigen Function folder. Voor de nieuwe “CertificateCheck” Function is een eigen folder voor aangemaakt.



Figuur Spplitsing van Functions

### 10.1.1 Chilkat Library

Om de certificaten te kunnen verkrijgen van een endpoint zal eerst een request gemaakt moeten worden naar de desbetreffende server. Om de certificaat te kunnen verkrijgen van een server zal eerst een aanvraag gedaan moeten worden, oftewel een HttpRequest.

Bij het zoeken naar bronnen naar certificaten is een library gevonden genaamd Chilkat. Door de library te gebruiken wordt de certificaat in een object omgezet binnen de library die uit te lezen is (bijlage 9).

Na het aanmaken van de request kan aan de socket naar de certificaat gevraagd worden. Deze variabele wordt omgezet naar een Chilkat certificaat object om vervolgens informatie uit te krijgen. Belangrijkste voor het project is het veld “bExpired”. Dit is de einddatum van het certificaat.

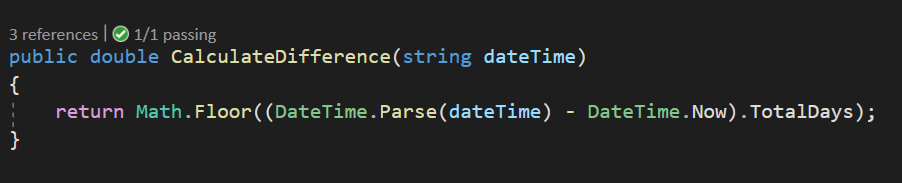
Na een aantal dagen werd er een notificatie vertoond tijdens het runnen. Deze notificatie gaf aan dat de Chilkat library een betaalde Library was (Chilkat, 2021). Na verder te kijken bleek dat er geen gratis versie van de library te bestaan. Na overlegd te hebben met de begeleider is besloten om geen gebruik te maken van de Chilkat Library. Hiervoor is een vervanger voor gezocht.

### 10.1.2 Vervangen Chilkat met HttpClient

Na verder te zoeken naar een vervanger is een HttpRequest oplossing gevonden. Uit documentatie van Microsoft is een bron gevonden waaruit de certificaat wordt ontleed vanuit een request (Microsoft, 2021). Uit de bron kan geconcludeerd worden dat een callback geplaatst kan worden zodra de request uitgevoerd is. Wanneer de request klaar is, wordt de callback uitgevoerd met certificaatgegevens. Hierin bevinden zich zelfde gegevens vergeleken met de output van de oude output met de Chilkat library, waardoor er geen grotere veranderingen plaats hoeven te vinden aan verdere verwerking.

### 10.1.4 Einddatum certificaat

Nadat de einddatum van het certificaat is gevonden kan een berekening plaatsvinden. Deze calculatie zal concluderen hoe veel tijd er nog is voordat het certificaat beëindigd. Doordat certificaten met maanden verlengt worden, is er besloten om aantal dagen te gebruiken voor de berekening. Bij het berekenen van de einddatum ontstaat er in dit geval nog een probleem. Er is een mogelijkheid dat een certificaat midden op de dag kan verlopen. Om zo nauwkeurig mogelijk te zijn, is besloten om alles naar beneden af te ronden (figuur 28). Dit wordt gedaan om vroegtijdig de verlopen certificaten te kunnen vinden.



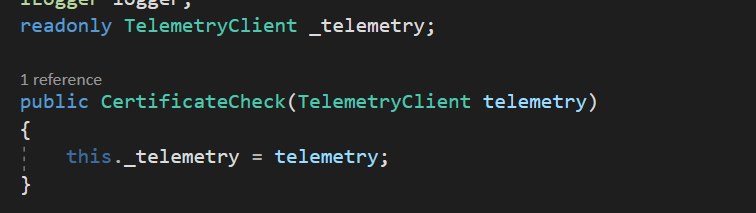
Figuur berekenen einddatum certificaat

## 10.2 Technical Debt

Tijdens vorige sprint is er Technical Debt overgebleven die opgelost moet worden. Dit bestaat uit een object van Application Insight die niet meer wordt gebruikt. In hoofdstuk 8.2.1 wordt verwezen naar een bron waarin de oplossing te vinden is. In de bron wordt een voorbeeld gegeven van een Singleton die gedurende startup wordt gecreëerd.

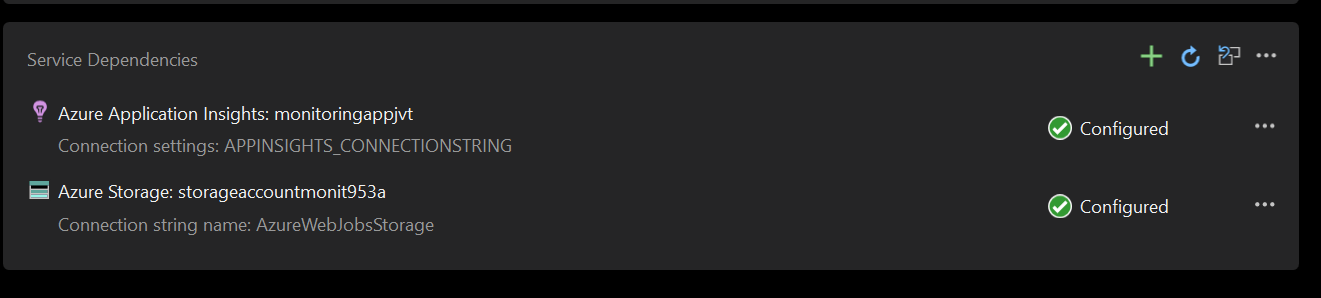
Hiervoor wordt gebruik gemaakt van een Startup file. Binnen de startup wordt aan de hand van dependancy injection een singleton geïnjecteerd in de applicatie (bijlage 10 startup file). Aan de hand van dependancy injection kan de object binnen de Function in gebruik genomen.

Hierdoor hoeft binnen de applicatie geen andere instanties van de Application Insight API aangemaakt worden.



Figuur Telemetryclient via de Constructor

De instrument key wordt via de service dependancies opgehaald. Dit betekend dat er geen instrument key opgegeven hoeft te worden. Zodra Application insight is toegevoegd aan service dependancies onder de Publishing Profile, wordt de instrument key automatisch toegevoegd aan de applicatie (figuur 30).



Figuur Service Dependancies

### 10.2.1 Testen

Na het oplossen van de Technical debt is focus gelegd op het testen van de applicatie. Voor het testen is een rapport opgesteld waarin alle testen in voorkomen (bijlage 11).

Voor de huidige sprint is de Pinger functionaliteit getest. De user story #1 “Als supportmedewerker wil ik inzicht hebben op downtimes zodat ik weet of er een probleem zich afspeelt.” Is getest aan de hand van Acceptatiecriteria.

Deze zijn:

1. AC1: ms boven de 150 is valide met alert
2. AC2: MS onder de 150 is valide
3. AC3: RTT moet een double zijn

Voor deze acceptatiecriteria zijn fysieke gevallen gemaakt:

|  |  |  |
| --- | --- | --- |
| **Attribuut** | **Geldig** | **Ongeldig** |
| Ms | Onder 150  Boven 150  2 Digit double  3 Digits double  4 Digits double  5 Digits double | Lege getal  String  3 digits integer  4 digits integer  5 digits integer |

Uit de geldige en ongeldige gevallen is te merken dat de tests meerdere digits gevallen bevatten. Deze gevallen representeren gevallen waarin de miliseconden in de honderden of duizenden zitten. De regex detecteert deze ondanks de lengte en zal het moeten matchen. Dit geldt voor beide gevallen.

Wanneer het niet kan matchen, wordt een string of een leeg getal teruggestuurd. Dit is een foutmelding waar rekening mee gehouden moet worden. Zodra dit het geval is wordt het opgevangen door een exception binnen het testen.

Logische testgevallen:

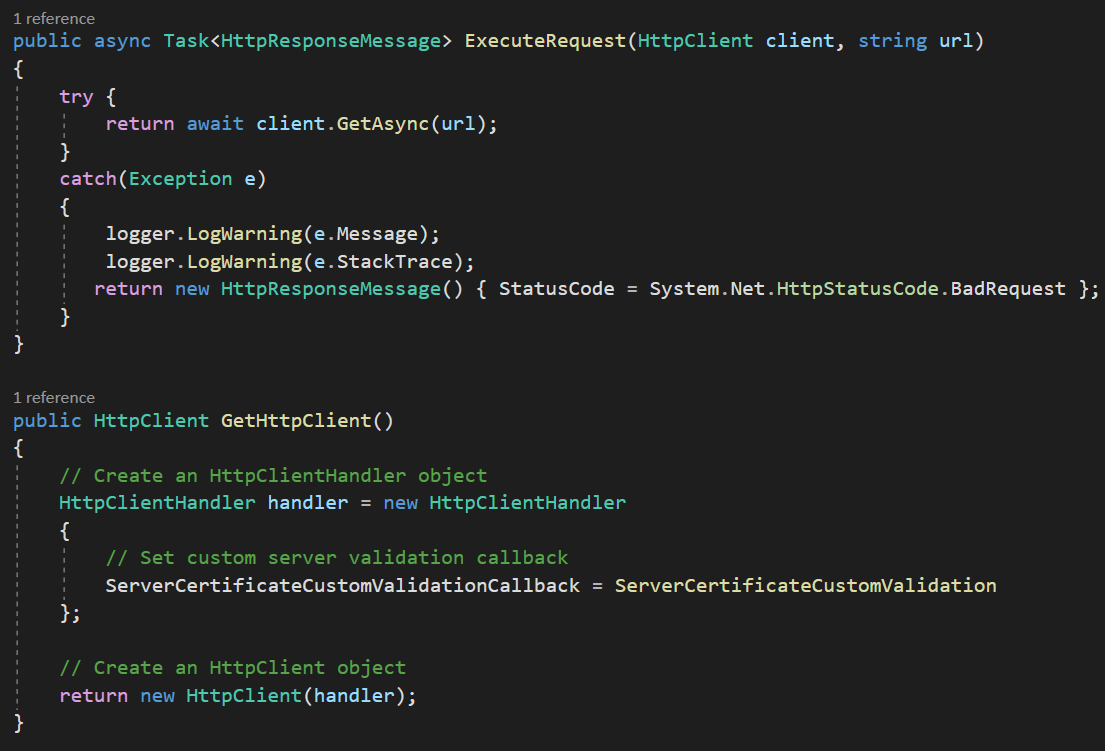
|  |  |
| --- | --- |
| **Geval** | **Resultaat** |
| Onder 150 | Valide |
| Boven 150 | Valide |
| 2 Digit double | Valide |
| 3 Digits double | Valide |
| 4 Digits double | Valide |
| 5 Digits double | Valide |
| Lege getal | Invalide (error) |
| String | Invalide (error) |
| 3 digits integer | Invalide (error) |
| 4 digits integer | Invalide (error) |
| 5 digits integer | Invalide (error) |

Fysieke testgevallen:

|  |  |
| --- | --- |
| **Geval** | **Resultaat** |
| Onder 150 | “RTT” |
| Boven 150 | “HighPingAlert” |
| 2 Digit double | “RTT” |
| 3 Digits double | “RTT” |
| 4 Digits double | “RTT” |
| 5 Digits double | Valide“RTT” |
| Lege getal | Exception |
| String | Exception |
| 3 digits integer | Exception |
| 4 digits integer | Exception |
| 5 digits integer | Exception |

Alle testen passen zodra ze gedraaid worden (bijlage 11).

Een connectie wordt gemaakt door de request naar de endpoint uit te voeren (figuur 28).



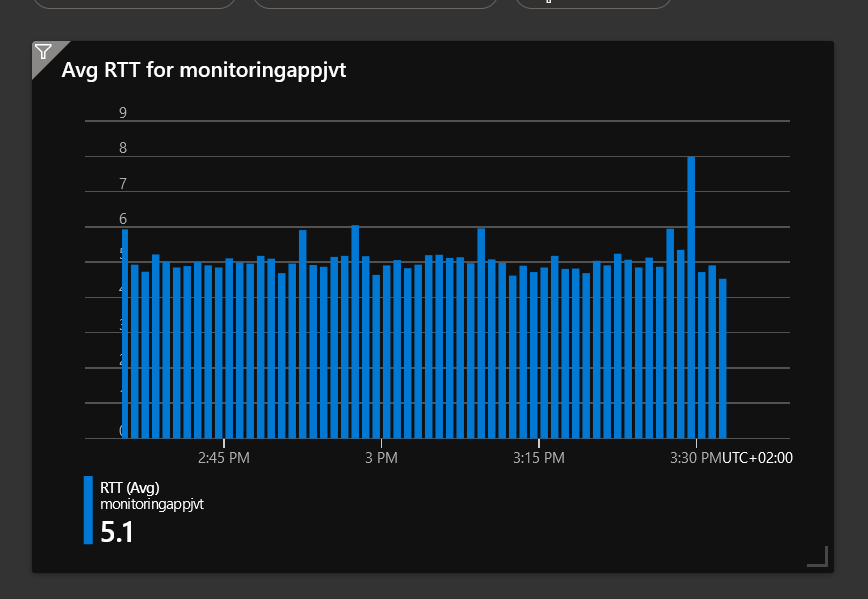
Figuur Request endpoint

In GetHttpClient functie wordt de Callback methode gezet om na de request naar de callback methode te gaan. Als laatst wordt de einddatum van de certificaat opgevraagd binnen de callback methode. Met deze data kan de berekening van eindtijd plaatsvinden (figuur 28).

## 10.3 Application Insight Dashboard

Nu al de data opgeslagen worden, zullen deze ergens gevisualiseerd moeten worden. Dit is mogelijk door grafieken te gebruiken op Application Insight. De supportmedewerker krijgt de mogelijkheid om op één oogopslag te controleren of er abnormaliteiten zijn.

Voor de grafieken is gebruik gemaakt van een metric chart zoals in bijlage 7. Om voor een beter overzicht te zorgen is gebruikt van een bar-chart over een langere tijd. Hierna is de “granularity” van de grafiek aangepast. Er is gebruik gemaakt van 5 minuten over een 4 uur tijdsperiode. Dit betekend dat elke bar 5 minuten representeert. Daarnaast is de X as verdeeld over vier uur. Dit komt goed uit doordat elke 5 minuten de ping test uitgevoerd wordt. Hierdoor worden testen niet samengevoegd.



Figuur MEtric Charts

## 10.4 Sprint review & Retrospective

*Review*

Feedback van de bedrijfsmentor op resultaten van sprint 5 zijn positief behalve op de grafieken. Deze waren voor de CertificaatCheck function nog niet af waardoor ik geen volledig dashboard kon vertonen. Daarbuiten is de bedrijfsmentor positief over hoe de grafieken eruitzagen en was hij blij met de lengte van de X-as waar tijd vertoond werd. Gedurende sprint 5 is de sprintdoel niet behaald. De taken binnen User story 2 zijn grotendeels af. Hiervoor hoeft alleen de Application Insight gedeelte voor geïmplementeerd worden om het af te kunnen sluiten.

Voor de twee user stories geldt de volgende:

|  |  |  |
| --- | --- | --- |
| # | User story | Resultaat |
| 2 | Als supportmedewerker wil ik inzicht hebben op verlopen certificaten, zodat ik weet of er een probleem zich afspeelt. | Wordt vervolgd. |
| 4 | Als supportmedewerker wil ik een email ontvangen mocht er een downtime optreden. | Afgerond |
| - | Technical Debt | Afgerond |

### 10.4.1 Evaluatie

Voor implementatie van de CertificateCheck functie is eerst gekeken naar mogelijkheden. Hiervoor is gekeken naar een library die later betaald bleek te zijn. Om later niet tegen problemen op te lopen is er een vervanger voor gezocht. Het gebruik van de betaalde library kon vermeden worden door beter de documentatie te lezen. Teruggekeken naar de library, is er geen teken van kosten voor het gebruik van de library waardoor het inschatten lastig blijft.

Verder is de functionaliteiten getest op fouten. Voor kwaliteitsbewaking is een testrapport opgesteld met daarin de resultaten van de test. Hiervoor is gebruik gemaakt van Fysieke en logische testgevallen om een testbasis op te zetten. Vervolgens zijn de fysieke testgevallen omgezet in code om de testen uit te kunnen voeren.

# 11.0 Sprint 6: Certificaatcheck Alert & Application Insight

Voor de komende sprint wordt de alert systeem opgezet voor de certificaatchecker. Hierbij wordt gekeken naar welke thresholds nodig zijn voor de metrics. Hiervoor moet eerst de data verstuurd worden naar Application Insight.   
  
Binnen sprint 6 wordt twee user stories in meegenomen:

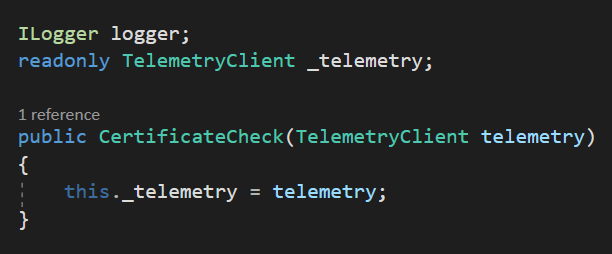
|  |  |
| --- | --- |
| # | User story |
| 13 | Als klant wil ik een email ontvangen zodat ik weet dat de support op de hoogte is gesteld van een verlopen certificaat. |
| 2 | Als supportmedewerker wil ik inzicht hebben op verlopen certificaten, zodat ik weet of er een probleem zich afspeelt. |

Het sprintdoel voor sprint 6 is het volledig afmaken van de Certificaatchecker tot en met Application Insight. Van code tot aan de grafieken op Application Insight om de data te kunnen visualiseren.

## 11.1 CertificaatCheck Application insight

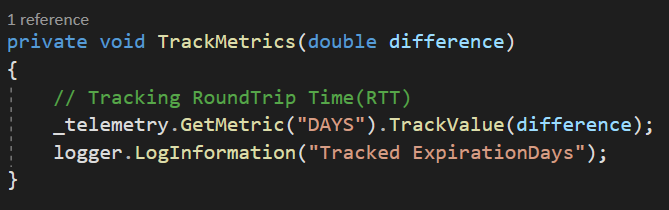
Voordat data gevisualiseerd kan worden, moet deze opgeslagen worden in Application Insight. Daardoor wordt gestart met het overnemen van de Dependancy injection implementatie toegepast op de Pinger functie in (h9.2).

Voor gebruik van de nieuwe TelemetryClient object binnen de CertificaatCheck methode is gebruik gemaakt van een constructor om de object binnen de class te kunnen injecteren.



Figuur Certificatecheck Dependancy Injection

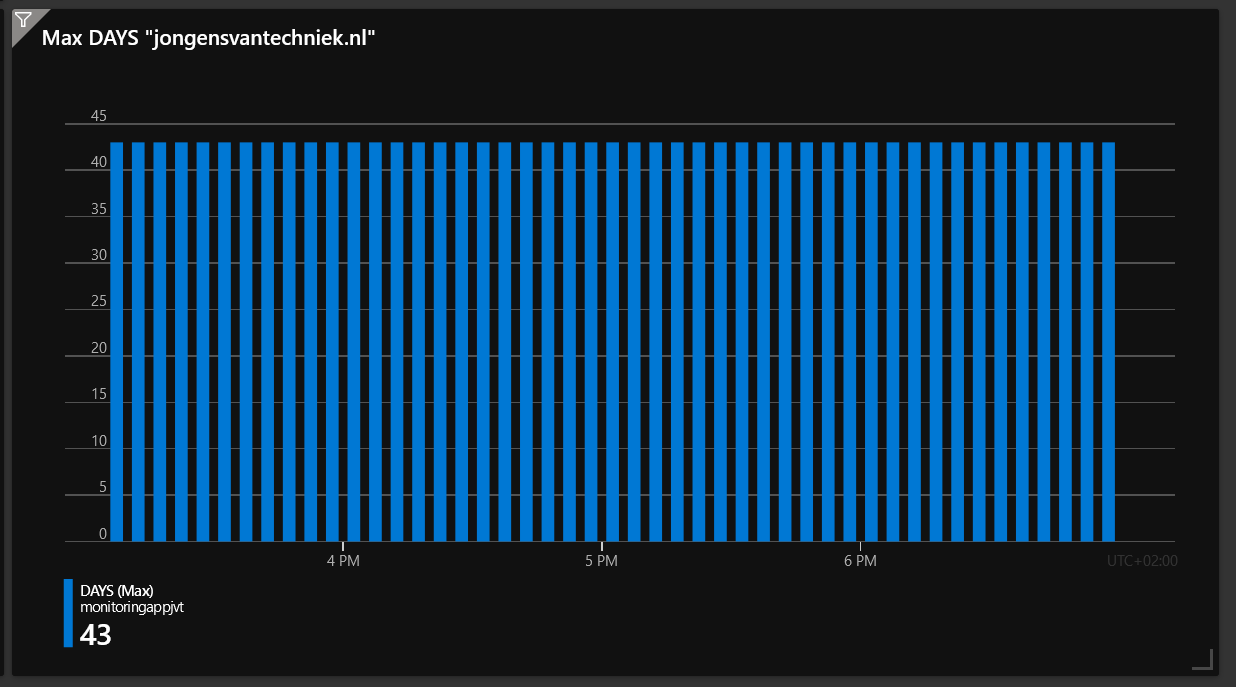
Vervolgens is de Client gebruikt om data te kunnen versturen naar Application Insight. Hiervoor is de naam van de Metric aangepast naar “DAYS” om aan te geven dat dit gaat om aantal dagen voor verloop certificaat (Figuur 34).



Figuur Tracking days

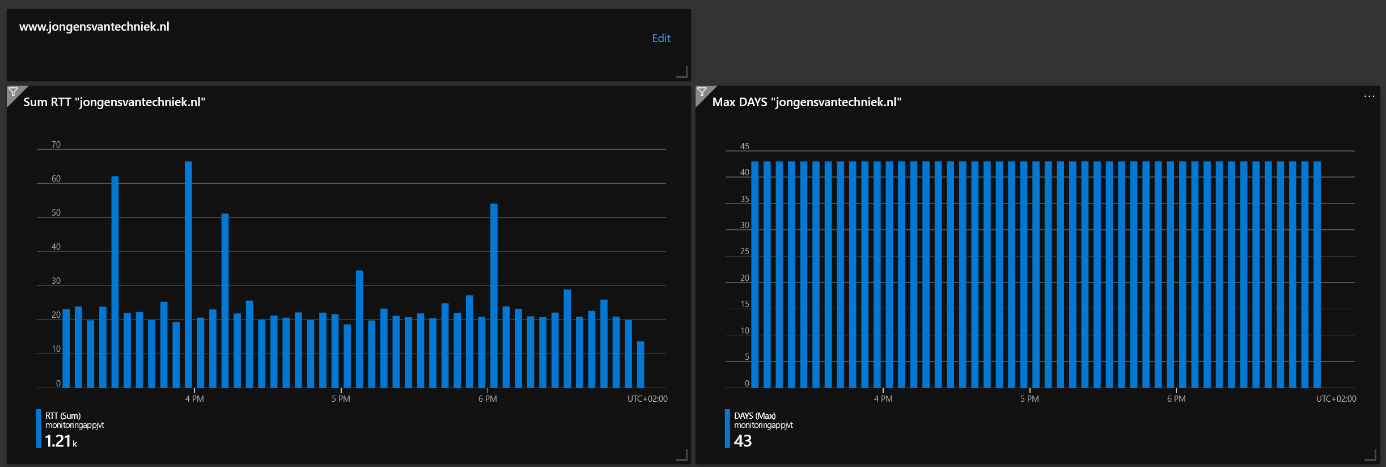
## 11.2 Application Insight

Nu certificaat data verstuurd wordt naar Application Insight kan deze gebruikt worden om de grafieken mee te kunnen maken. Op de dashboard van Application Insight is een nieuwe grafiek aan het dashboard toegevoegd zoals dit gebeurt voor de data vanuit de Ping Functions in hoofdstuk 9.3.



Figuur Days Chart

Tijdens een review sessie waren er een aantal opmerkingen op de grafieken. De eerste was dat de grafieken te klein waren op de dashboard. Hiervoor is zijn de grafieken iets vergroot qua breedte en lengte. Verder was het niet duidelijk voor welke endpoint de grafieken voor waren. Hiervoor is een label boven de grafieken geplaatst die aangeeft om welke endpoint het gaat (figuur 36).



Figuur Dashboard Application Insight

## 11.3 CertificaatCheck Alerts

Nu data verstuurd wordt naar Application Insight, kan de alert toegevoegd worden aan de metrics. Zoals dit in H8.2.1 besproken wordt, zijn er vier delen waar aan gedacht moet worden. Voor de Certificaatcheck zijn de meeste elementen hetzelfde behalve de alert condition.

Er is gekozen om de metric binnen de “CertificateCheck” function “DAYS” te noemen zoals dat in figuur 34 te zien is. Deze benaming is gekozen doordat de metric dagen representeert voordat een certificaat verloopt. Binnen de Condition wordt “DAYS” gekozen. Tijdens een gesprek is naar voren gekomen dat een alert verstuurd moet worden zodra een certificaat minder dn 30 dagen geldig is. Tijdens het aanmaken van de condition is daardoor 30 dagen als “Threshold value” gebruikt. Vervolgens wordt dezelfde actions gekozen als de pinger alert, oftewel mailen naar een persoonlijke email voorlopig.

Onder Alert rule details is er een naam en een beschrijving van de alert gegeven. De Alert naam is ExpiredCertificate met als beschrijving: “Alerting when certificate is close to expiry date”.

## 11.4 Testen

Na het afmaken van de certificate check functionaliteit, wordt focus geplaatst op het testen ervan. Hiervoor wordt de user story 2 “Als supportmedewerker wil ik inzicht hebben op verlopen certificaten, zodat ik weet of er een probleem zich afspeelt” voor getest.

Voor deze user story zijn de volgende acceptatiecriteria voor opgesteld.

De acceptatiecriteria gebaseerd op deze user story zijn:

1. AC1: resterende tijd wordt vertoond in dagen
2. AC2: Resterende tijd minder dan 30 dagen worden als fout behandeld
3. AC3: Resterende tijd meer dan 0 wordt verstuurd naar Application Insight
4. AC4: Endpoints zijn string in een array

Om voorspellingen vast te stellen worden de tests gesplit in geldige en ongeldige gevallen. De splitsing is als volgt:

|  |  |  |
| --- | --- | --- |
| **Attribuut** | **Geldig** | **Ongeldig** |
| DAYS | Boven 30  Onder 30 | Lege getal  String |

Doordat meerdere attributen getest worden zijn deze twee in aparte tabellen opgeslagen voor duidelijkheid. Hieronder volgt de geldige en ongeldige gevallen voor de URL.

|  |  |  |
| --- | --- | --- |
| **Attribuut** | **Geldig** | **Ongeldig** |
| URL | Strings  Begint altijd met een http | Lege string  Begint met www |

De gevallen worden vervolgens omgezet naar Logische testgevallen. Dit is gedaan om alle gevallen op een rij te kunnen hebben. Hierdoor is het mogelijk om te kunnen controleren of de grenzen zijn gedekt. Binnen de logische testgevallen is dit makkelijker te doen doordat deze in een leesbare taal worden gemaakt.

Logische testgevallen als volgt:

|  |  |  |
| --- | --- | --- |
| Geval # | Geval | Resultaatvoorspelling |
| 1 | DAYS boven 30 | valide |
| 2 | Days onder 30 | Valide |
| 3 | Days met leeg getal | Invalide |
| 4 | Days als string | Invalide |
| 5 | URL als string | Valide |
| 6 | URL begint met een http | Valide |
| 7 | URL als lege string | Invalide |
| 8 | URL begint met een www | Invalide |

Daarop volgend komen de fysieke testgevallen. Deze testgevallen worden beschreven aan de hand van de logische testgevallen. De fysieke gevallen zijn dichter bij het technische vorm van testen. Deze kunnen makkelijker omgezet worden in code.

Fysieke testgevallen als volgt:

|  |  |  |
| --- | --- | --- |
| Geval | Logische test geval # | Resultaat |
| Days = 31 | 1 | valide |
| Days = 29 | 2 | Valide |
| Days = ; | 3 | Invalide |
| Days = “2” | 4 | Invalide |
| URL = “https://jongensvantechniek.nl” | 5 | Valide |
| URL = “https://jongensvantechniek.nl” | 6 | Valide |
| URL = “” | 7 | Invalide |
| URL = “www.jongensvantechniek.nl” | 8 | Invalide |

## 11.4 Sprint review & Retrospective

*Review*

Binnen deze sprint zijn de twee user stories afgemaakt. Tijdens review kwamen er een aantal opmerkingen over de dashboard en de grafiek zichtbaarheid. Na deze aangepast te hebben was de supportmedewerker blij met de resultaten.

|  |  |  |
| --- | --- | --- |
| # | User story | Resultaat |
| 13 | Als klant wil ik een email ontvangen zodat ik weet dat de support op de hoogte is gesteld van een verlopen certificaat. | Afgerond |
| 2 | Als supportmedewerker wil ik inzicht hebben op verlopen certificaten, zodat ik weet of er een probleem zich afspeelt. | Afgerond |

*Retrospective*

Sprint 6 is door vakantie en feestdagen verkort. Desondanks heb ik nog een aantal deliverables die over gebleven zijn van de vorige sprint kunnen opleveren en een review kunnen krijgen. Door samen met David (Supportmedewerker) te zitten hebben we samen gekeken naar verbeteringen aan de grafieken. Hij gaf aan de X-as van 4 uur genoeg waardoor ik het zo heb gelaten.

### 11.4.1 Evaluatie

De afgelopen sprint is makkelijk geweest. Voor de koppeling tussen Application Insight en de CertificateCheck function is gebruik gemaakt van de al bestaande connectie in de Pinger function. Verder is de Alert toegevoegd aan de Metric dat de certiicateCheck function verstuurd naar Application insight. Dit was makkelijk doordat dit al eerder gedaan is voor de Pinger Functionaliteit.

Als laatst moest de dashboard opgezet worden met de grafieken voor de Metric. Hiervoor is gekozen om een grafiek op te zetten naast de bestaande grafiek voor de Pinger resultaten. Door een aantal instellingen binnen de granulariteit en de X-as te maken is de grafiek duidelijker gemaakt voor de gebruiker.

# 12.0 Sprint 7: Toevoegen Endpoints & Dashboard

Voor sprint 7 is het sprintdoel toevoegen van meerdere endpoints. Hiervoor wordt user story 6 en 7 binnen de sprint genomen.

|  |  |
| --- | --- |
| # | User story |
| 6 | Als supportmedewerker wil ik endpoints wijzigen aan de lijst van de gemonitorde endpoints, zodat ik aanpassingen kan doen mochten de URL’s wijzigen. |
| 7 | Als supportmedewerker wil ik certificaten toevoegen aan de lijst van de gemonitorde certificaten, zodat ik bestaande en nieuwe certificaten kan aanpassen. |

Voor de komende sprints wordt functionaliteit toegevoegd aan beide Functions waarbij de Supportmedewerker de kans krijgt meerdere URL’s toe te voegen aan de applicatie. Voor deze user stories wordt beslist op welke manier de functionaliteit geïmplementeerd wordt, omdat er geen database beschikbaar is. Met de beschikbare resources die het project heeft, zijn er tot oplossingen gekomen.

## 12.1 Pinger endpoints

Voor User Story 6 wordt een implementatie gemaakt waarbij meerdere endpoints gemonitord kunnen worden. Hiermee krijgt de supportmedewerker de bevoegdheden om meerdere endpoints te kunnen monitoren, waaronder toevoegen en verwijderen mocht dat nodig zijn.

Hiervoor is eerst gedacht om de endpoints, oftewel URL’s op te slaan in een database waar ze uitgehaald kunnen worden. Alhoewel dit een geschikt oplossing zou zijn, was er vanuit Jongens van Techniek een limitatie dat er nog geen gebruik van databases gebruikt mag worden omdat kostenbesparing prioriteit heeft. Daardoor is het voorlopig niet mogelijk om gebruik te maken van een database, omdat dit kosten met zich mee brengt.

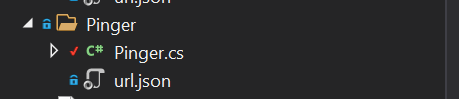
Een ander oplossing hiervoor is het gebruik van een lokaal opslag. Dit kan op meerdere manieren waaronder een Excel bestand, YAML-bestand of een JSON bestand. Voor lokaal opslaan van data is er geen best practice. Omdat er “name value pairs” opgeslagen zullen worden, is het handig om de JSON of een Yaml bestand voor te gebruiken. Dit komt doordat JSON en YAML in “name-value pairs” opgeslagen worden. De keuze tussen JSON en YAML is gemaakt op basis van ervaring waar JSON bovenuit is gekomen.

Om meerdere endpoints te verzamelen is beroep gedaan op de bedrijfsmentor. Deze heeft een aantal endpoints opgesteld die toegevoegd kunnen worden aan de MonitoringApp.

Deze endpoints zijn:

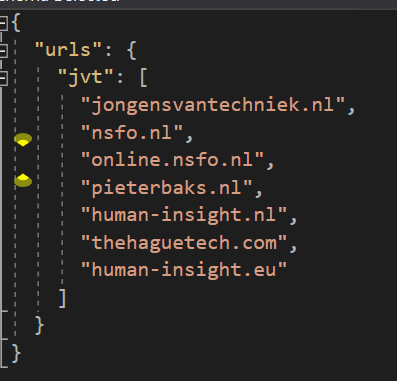
1. jongensvantechniek.nl
2. nsfo.nl
3. online.nsfo.nl
4. pieterbaks.nl
5. human-insight.nl
6. thehaguetech.com
7. human-insight.eu

Voor de endpoints is een JSON bestand aangemaakt in de desbetreffende Functions folder.



Figuur Pinger functions folder

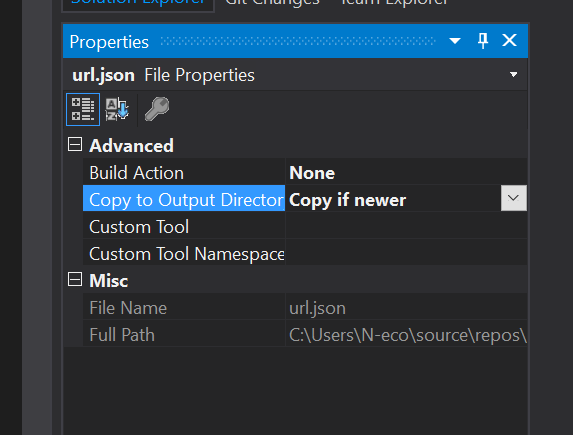
De urls zijn opgeslagen als een array. Hiervoor is gekozen om een “naam-value pair” aan te maken genaamd “url’s: jvt” om aan te geven dat de URL’s volgen voor JVT.



Figuur Endpoints json bestand

De endpoints zijn ingelezen aan de hand van de File object (Microsoft, 2021). Dit object wordt gebruikt om door bestanden binnen het project te kunnen bladeren en te openen.

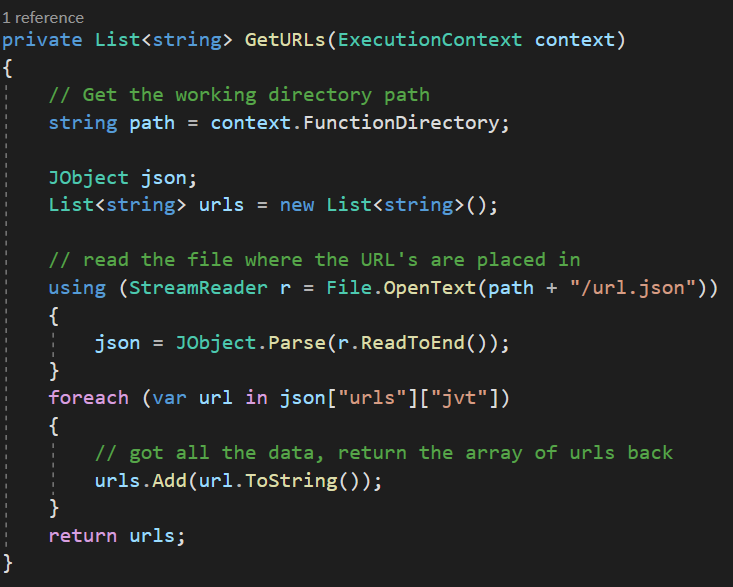
Het vinden van een bestand was nogal problematisch. Dit komt doordat het bestand beweegt tijdens het uitvoeren van de Function. Het verplaatst zich naar de debug van de applicatie. Om dit op te lossen zal de instellingen van het JSON bestand aangepast worden. Onder properties van het JSON bestand moet de “Copy to output directory” veranderd worden naar Copy if Newer. In dit geval wordt de JSON bestand gekopieërd naar de output directory wanneer deze is gewijzigd. De output directory is de locatie waar de executable bestand in komt te zitten zodra de applicatie gedraaid wordt (figuur 39).



Figuur Json bestand properties

### 12.1.1 Inlezen JSON

Na de URL bestand gekopieërd wordt naar de output directory, zal deze ingelezen moeten worden. Daarvoor zal eerst naar de root van de Function gegaan moeten worden. Dit wordt gedaan aan de hand van een Object genaamd “ExecutionContext”. Met dit object wordt de root path van de Function blootgesteld in code (Mikhail Shilkov, 2018). Het JSON bestand wordt ingeladen door de root path op te halen van het project. Deze variabele wordt vervolgens doorgegeven aan de File object om het Json bestand te kunnen vinden. Als laatst worden de urls geparst naar een string en opgeslagen in een array.



Figuur URL parser

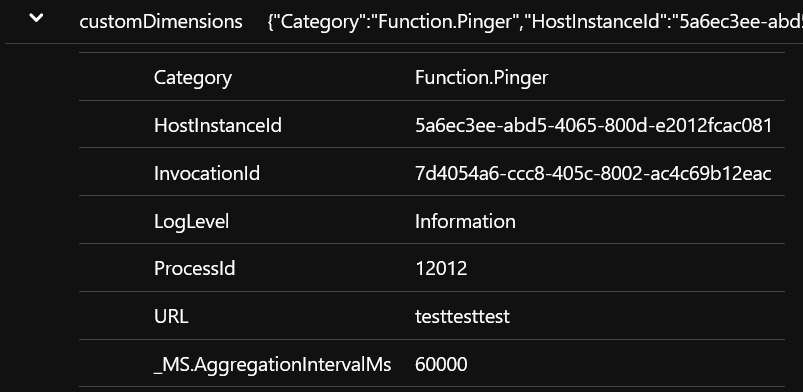
De endpoints kunnen nu getest worden door met een loop erover heen te gaan en ze één voor één te testen.

### 12.1.3 Application Insight

Een volgend probleem die naar boven komt is dat er nu meerdere Pings plaatsvinden elke keer de Function draait. Vervolgens wordt deze data verstuurd naar Application Insight onder de aanname dat JongensvanTechniek.nl getest wordt (h10.3). Hiervoor moet onderscheid voor gemaakt worden binnen Application Insight.

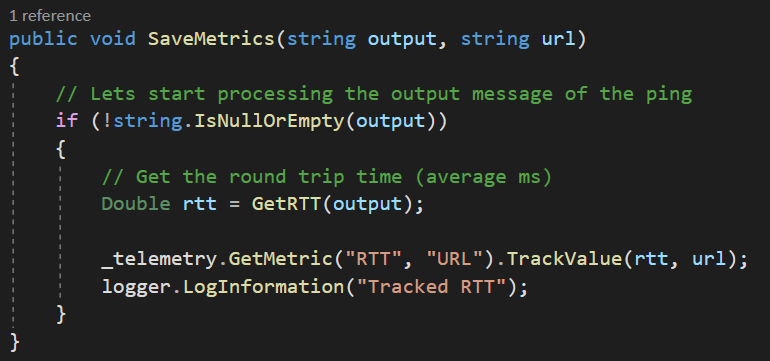
Uit de bron van Microsoft is gebleken dat de metrics de optie heeft om meerdere data op te kunnen slaan (Microsoft, 2020). De metric die nu opgeslagen wordt is een waarde voor Dag (“DAYS”) of tijd (“RTT”) waar de onderscheiding in de naam plaatsvindt. Uit de eerder genoemde bron is gebleken dat dit nog een level dieper kan, genaamd Custom Dimensions.

Per metric kan extra metadata aan meegegeven worden. Deze metadata wordt opgeslagen in de CustomDimensions onder de Logs. Om te kunnen testen is er een extra dimension toegevoegd aan de getMetric methode genaamd URL. Voor test doeleinden is eerst gekeken wat veranderd wordt. Hiervoor is terug gegaan naar de Azure portal en zijn de logs bekeken. Hier is gebleken dat extra data voor de metrics onder de kolom custom dimensions wordt geplaatst (figuur 41).



Figuur Logs testen custom dimensions

Er is gekozen om metadata te koppelen aan de verstuurde metrics. Dit geeft ons de mogelijkheid om de metric te kunnen identificeren aan de hand van de metadata. Hierin zal staan om welke URL (endpoint) het gaat. De test code is vervangen door de echte url. Nu wordt aan elke metric de url gekoppeld en verstuurd naar Application Insight.

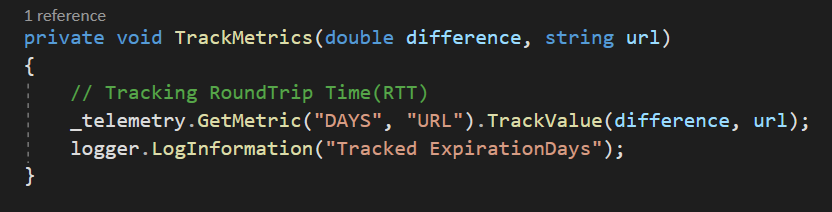


Figuur Save metrics met custom dimensions

## 12.2 CertificateCheck endpoits

Het Json bestand toegevoegd aan de pinger Function folder is gekopieërd en toegevoegd aan de certificateCheck function folder. Elke function heeft zijn eigen set aan endpoints die getest moet worden. Er is gekozen voor deze scheiding doordat om een scheiding te hebben voor endpoints voor verschillende Functions. De scheiding hiervoor is in stand gehouden voor toekomstige wijzigingen aan de endpoints.

Qua code zijn de wijzigingen hetzelfde. Een custom dimension genaamd “URL” is toegevoegd aan de “DAYS” metric.

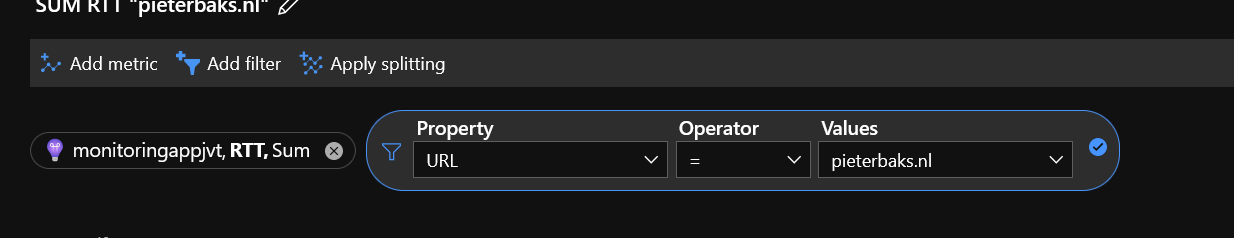


Figuur Custom dimensions voor Certificatecheck

## 12.3 Application Insight Dashboard

Na het toevoegen van de custom dimensions aan de metrics treedt er een volgende probleem op. Alle metrics worden opgeslagen maar worden vertoond onder de grafiek van Jongensvantechniek.nl (figuur 35). Hierdoor is het niet meer mogelijk om te achterhalen hoeveel miliseconden per endpoint het heeft gekost.   
  
Er zal een scheiding tussen de grafieken moeten komen. Hiervoor kan gebruik gemaakt worden van de filters (bijlage 7). Er is besloten om per endpoint een sectie aan te maken waar de grafieken onder kunnen vallen. Hiervoor is de label met de endpoint naam behouden waar vervolgens de grafieken onder komen. Voorlopig zijn dit twee grafieken maar kan later oplopen naar meerdere zodra er tests toegevoegd worden (bijlage 12, 13).

Per grafiek zal er een filter toegepast moeten worden. Deze filter is gebaseerd op de custom dimension die in hoofdstuk 12.1 en 12.2 is toegevoegd. Zodra Application Insight deze dimensions heeft gedetecteerd, zijn ze zichtbaar onder values (figuur 44).



Figuur Filter pieterbaks.nl

## 12.4 Sprint review & Retrospective

*Review*

Afgelopen sprint zijn er een aantal deliverables opgeleverd inclusief de twee user stories voor de Pinger en de Certificaatchecker. Het resultaat ervan was een dashboard waar de supportmedewerker veel baat bij heeft. Gedurende review gaf de supportmedewerker aan dat hij de grafieken handig vond om snel informatie uit te kunnen halen. De bedrijfsmentor gaf aan dat de dashboard te klein was voor de aantal grafieken die er bestaan maar dit is het gevolg van meerdere endpoints monitoren. Hiervoor kan later een oplossing voor gevonden worden door meer splitsing te gebruiken en gebruik te maken van meerdere dashboards binnen Application Insight. Voor sprint 7 geldt dat de twee user stories afgerond zijn.

|  |  |  |
| --- | --- | --- |
| # | User story | Resultaat |
| 6 | Als supportmedewerker wil ik endpoints wijzigen aan de lijst van de gemonitorde endpoints, zodat ik aanpassingen kan doen mochten de URL’s wijzigen. | Afgerond |
| 7 | Als supportmedewerker wil ik certificaten toevoegen aan de lijst van de gemonitorde certificaten, zodat ik bestaande en nieuwe certificaten kan aanpassen. | Afgerond |

## 12.4.1 Evaluatie

De afgelopen sprint was lastig geweest. Er is door documentatie heen gegaan worden om bronnen te vinden voor Custom dimensions. De meeste bronnen die beschikbaar zijn op Microsoft waren weer te licht en oppervlakkig waardoor niet voldoende gevonden kon worden. Desondanks heb ik wel de juiste bron gevonden die mij heeft geholpen bij het toevoegen van de Custom Dimensions. Dit is wel gebeurt aan de hand van Community resources in plaats van officiële documentatie van Microsoft. Hieruit is ontdekt dat er aan de hand van meerdere parameters, custom dimensions gebouwd kon worden. Dit stond niet in documentatie van Microsoft. Doordat de afgelopen sprint de laatste was, ben ik zo veel mogelijk gaan pushen om de laatste user story af te maken. Samen met de bedrijfsmentor was afgesproken om deze userstory als laatst nog af te maken voor het project. Op het laatste moment is dit gelukt, waardoor een goed fundament staat voor een monitoringsysteem. Dit kan later nog aangepast worden mocht dat nodig zijn. Voor anderen heb ik een handleiding gemaakt die gebruikt kan worden om het project te kunnen begrijpen.

# 13.0 Evaluatie producten

## 13.1 Producten

### 13.1.1 Plan van aanpak

Aan het start van het project heb ik een plan van aanpak opgesteld. Om dit te kunnen maken heb ik eerst een oriënterende gesprek gehad met de supportmedewerker, David. Samen zijn we gaan kijken naar welke stappen gezet worden om een ticket aan te maken.   
  
Dit heeft mij geholpen om het probleemdomein beter te begrijpen. Gedurende deze gesprek is vastgesteld welke problemen naar boven komen wanneer het gaat om support en de support tickets die klanten aanmaken.

Aan de hand van de probleemdomein en problemen zijn doelstellingen opgesteld. Hierin heb ik beschreven welke producten worden opgeleverd. Hierna heb ik de oplossingen als doelstelling genoteerd. Daarna is beschreven welke problemen het project zal oplossen.

Verder is gekeken naar risico’s binnen het project. Er is een risicoanalyse gemaakt op basis van technische, organisatorische en functionele problemen. Deze problemen zijn vooraf opgestelde problemen die naar voren kan halen welke problemen impact kunnen hebben mocht het probleem optreden gedurende het project.

Voor het project is er een aanpak opgesteld. Binnen de aanpak wordt beschreven welke projectmanagementmethode gebruikt zal worden om het project te kunnen implementeren. Hiervoor is een onderzoek gedaan naar Agile vs waterval (te vinden in bijlage 8). Hierin zijn de voordelen en nadelen tegenover gezet en is er een conclusie uit getrokken.

Als laatst wordt er een rooster voor de producten ontwikkeld. Binnen deze rooster wordt voorspelt wanneer een product af is. Deze producten zijn te vinden in Hoofdstuk 3.3. De uiteindelijke rooster is terug te vinden in hoofdstuk 4.2.

### 13.1.2 Requirementanalyse

De requirementsanalyse is opgesteld aan de hand van de oriëntatie gesprek met David. Dit is het eerste gesprek met david waarna de plan van aanpak is gevormd. Om meer informatie te kunnen verkrijgen, heeft er nog een gesprek plaatsgevonden met David om meer de detail in te gaan.

Verder is de afstudeerplan ook een bron geweest voor de requirements. Hieruit is de problemen uit gehaald die later omgezet is in oplossingen voor implementatie.

Verder is uit gesprek met peter voort gekomen dat Jongens van Techniek een cursus te bieden heeft die ik kon volgen. Uit het gesprek met peter is gebleken dat er ook andere oplossingsmogelijkheden zijn waardoor het project een andere richting heeft kunnen nemen.

### 13.1.3 Architectuur

Het architectuur ontwerp was eerst een systeem gebaseerd op een Dotnet Core applicatie. Na meer informatie verkregen te hebben via de course is besloten om een andere technologie te gebruiken voor het project. Hiervoor moest het project aangepast worden. Hiervoor is het eerdere ontwerp hergebruikt omdat het niet compleet verandert hoeft te worden. Alleen de levels dieper moesten verandert worden waardoor weer uit geconcludeerd wordt dat het gebruik van de C4 methode goed is geweest qua flexibiliteit. In bijlage 5 wordt deze beslissing uitgelegd.

### 13.1.4 Onderzoek Serverless Consumption

Het onderzoek naar Serverless Consumption heeft plaatsgevonden nadat dit opgekomen is gedurende een course. Tijdens de course heb ik gemerkt dat de Serverless mij een voordeel kan bieden in meerdere gebieden waardoor ik meer in ben gaan duiken. Hiervoor heb ik een onderzoek voor gestart om achter te komen welke voordelen dit zijn. Deze heb ik vergeleken met de voordelen die een dotnet core applicatie te bieden heeft. Dit is gedaan doordat er vanuit eerste oogpunt een dotnet core applicatie gebouwd zou worden. Het onderzoek naar Serverless heeft geleid naar een onderzoek naar Application Insight. Deze twee hebben een basis gevormd voor het project. Gedurende sprint 4, 5 en 6 heb ik gebruik gemaakt van het onderzoek om data te kunnen versturen, grafieken aan te maken en koppeling te kunnen vastzetten.

### 13.1.5 Onderzoek Application Insight

Na het onderzoek naar Serverless Consumption ben ik achter gekomen dat er meer beschikbaar is qua monitoring binnen Azure. Hiervoor ben ik in Application Insight gaan duiken.

Ik ben een onderzoek gestart naar de mogelijkheden die Application Insight te bieden heeft. Hierbij heb ik gekeken naar de verschillende metrics en de custom metrics die te volgens zijn binnen Application Insight. Om deze elementen te kunnen testen heb ik een testproject gemaakt gekoppeld aan een Application Insight. Door deze test project heb ik ervaring opgedaan die mij ten goede heeft gedaan gedurende sprint 4 en 6 waar connectie tussen de Function app en Application Insight gemaakt werd.

Daarnaast heb ik deze metrics gebruikt om data te kunnen visualiseren. In het onderzoek ben ik gaan kijken welke mogelijkheden er zijn voor data visualisatie. Hiervoor heb ik een dashboard aangemaakt waar ik test grafieken in heb gezet. Deze dashboard is later gebruikt voor echte data.

Desondanks de goede bronnen van Microsoft ben ik vaak vast gezeten. Dit kwam doordat er geen goede community resources beschikbaar zijn. De resources op Microsoft documentation zijn te oppervlakkig en gaan niet te diep op specifieke gevallen. Hiervoor ben ik vaker vastgezeten en moest ik terug naar documentatie van Microsoft om informatie bij elkaar te puzzelen voor het onderzoek.

### 13.1.6 Testplan & Testrapportage

De testplan is opgesteld om vast te stellen welke tests nodig zijn en welke tools nodig zijn voor het project. Gedurende project zijn deze tools gebruikt om de tests te maken. De testplan is opgesteld voor kwaliteitsbewaking-doeleinden.

Binnen de testplan is eerst gekeken naar welke tests handig zou zijn. Door wijzigingen aan de applicatie is gekozen om regressietests op te stellen gedurende het project aan de hand van unittests.   
  
Deze tests worden aan de hand van Fysieke en logische testgevallen opgezet. Eerst wordt gekeken welke user story getest zal worden waaruit acceptatiecriteria uit komen. Vervolgens heb ik gezorgd dat deze acceptatiecriteria alle grenzen dekken zodat geen grenswaardes over het hoofd gezien worden. Bijvoorbeeld dat de ping functionaliteit alle ms na 150 als waarschuwing en alles onder 149 als goed beschouwd.

### 13.1.7 Onderzoek Agile vs. Waterval

Het onderzoek tussen Agile en Waterval komt voort uit de zoektocht naar de juiste projectmanagementmethode. Hierin wordt gekeken naar welke voordelen en nadelen beide methodes hebben. Verschillende bronnen wordt gebruikt om te zoeken naar de voordelen van Scrum en waterval. Na deze gevonden te hebben, heb ik de voordelen en nadelen tegenover elkaar gezet. Sommige voordelen hebben zwaarder gewogen voor het onderzoek omdat ik ervoor wilde zorgen dat de bedrijfsmentor de mogelijkheid kreeg om de requirements aan te passen gedurende het project. Toch is heeft deze mogelijkheid zwaarder gewogen bij de beslissing waardoor de voorkeur sneller is uitgegaan naar gebruik van Agile.

## 13.2 Beroepstaken

### 13.2.1 A1 Analyseren probleemdomein & opstellen probleemstelling

Aan beroepstaak A1 is voldaan door een plan van aanpak te maken (H4.0). Gedurende het maken van de plan van aanpak is gewerkt aan oriëntatie en vervolgens opstellen van het probleemdomein. Binnen de plan van aanpak is gekeken naar verschillende onderdelen zoals projectmanagement methodes, testplan en testrapportages, DevOps mogelijkheden en het maken van een planning.

Verder is er een huidige situatie gecreëerd om vast te stellen wat het probleemdomein is. Door oriënterende gesprekken te houden met de juiste stakeholders bewijst de beroepstaak.

### 13.2.2 Gc Kritisch en methodisch werken

Gedurende het project zijn meerdere bronnen gebruikt om aan de juiste implementaties te voldoen. Bronnen zijn gecontroleerd op betrouwbaarheid en wanneer het bron is geplaatst om vast te stellen of het om een recente document gaat. Desondanks is dit één keer fout gegaan gedurende implementatie van de Chilkat library die een betaalde library bleek te zijn. Dit wilden we vermijden binnen het project doordat er geen budget beschikbaar is voor dit soort doeleinden. Desondanks heb ik geleerd om dit soort library’s te vermijden door te kijken naar het inhoud van de website waar de library op staat. Door te zoeken naar keywords zoals “Payment services” of “payment” is ervoor te zorgen dat dit soort problemen niet opnieuw zich kunnen voordoen.

Verder wordt dit beroepstaak bewezen aan de hand van de onderzoeken die zijn gedaan voor het project. Er zijn meerdere projecten uitgevoerd waarin gebruikt wordt gemaakt van betrouwbare bronnen en materiaal. Door deze te onderzoeken en kritisch te zijn over de kwaliteit hiervan is mogelijk geweest om componenten te vergelijken zoals Agile vs Waterval of onderzoek naar Serverless Consumption.

### 13.2.3 Gf Leren leren

Gedurende het project heb ik veel kunnen leren over Azure. Door resources te onderzoeken die ik nodig heb, heb ik mijzelf veel kunnen leren. Ik moest mijzelf verdiepen in Serverless Consumption en kennis opdoen op Application Insight om werking ervan te kunnen leren. Door tutorials op te zoeken naar deze resources heb ik onderzocht hoe deze resources werken. Om te kunnen testen voor werking heb ik testprojecten gemaakt ( bijlage 6, 7) waarin ik informatie die ik geleerd heb, in het praktijk breng.

### 13.2.4 C1 Ontwerpen software

Het ontwerpen van software is bewezen aan de hand van opzetten van een architectuur (H 7.3). Door een architectuur te ontwikkelen is bewezen dat ik de kennis heb om een fundament te leggen voor een project. Verder wordt deze beroepstaak bewezen door het kiezen van de juiste ontwikkel methodiek zoals dit is geleerd in blok 12.

Het architectuur is ontwikkeld aan de hand van verschillende levels. Hiervoor is gekeken naar welke koppelingen belangrijk zijn per level. Dit bewijst dat ik de mogelijkheid heb om op business level me te richten op het ontwikkelen van een architectuur die duidelijk is voor verschillende partijen.

### 13.2.5 A2 Informatie vergaren, analyseren & verwerken

Aan deze beroepstaak is voldaan door het opstellen van de requirements (H 5). Deze zijn opgesteld aan de hand van de meerdere gesprekken die zich gedurende het afstudeerperiode hebben plaatsgevonden. Door deze data te kunnen analyseren en verwerken zijn de requirements boven water gekomen.

### 13.2.6 D1 Realiseren software

In hoofdstuk 8, 9, 10, 11 en 12 wordt beschreven hoe de applicatie is geïmplementeerd. Hier wordt gekeken naar de bronnen die zijn verzameld om de applicatie mee te kunnen ontwikkelen. Door best practices te gebruiken voor het coderen worden bad smells vermeden. Verder is gebruik gemaakt van reviews om code schoon te houden.

Verder wordt deze beroepstaak bewezen door nieuwe techologiëen te gebruiken en te implementeren. Aan de hand van het onderzoek is Azure Functions gebruikt die niet eerder voor mij duidelijk was. Hiervoor is er een werkwijze bij gevonden doordat het lastig was om lokaal te runnen waardoor het runnen op de cloud meer aannemelijker was (bijlage 6).

### 13.2.7 D2 Testen en evalueren

In hoofdstuk 9 wordt er een testplan opgesteld. Binnen de testplan wordt beschreven welke tests er uitgevoerd zullen worden gedurende het project. Daarnaast wordt beschreven welke tools gebruikt zullen worden voor het project.

Als laatst wordt de beroepstaak bewezen door de testrapportages die zijn opgesteld voor de Azure Functions. Voor de Pinger en voor de Certificatecheck zijn rapportages opgesteld. In deze rapportages worden de user stories meegenomen om vast te kunnen stellen wat de acceptatiecriteria voor zijn. Deze worden vervolgens in een logische en vervolgens een fysieke testgeval gezet.

Deze gevallen worden vervolgens gebruikt om een unittest mee te maken. Met de tool die in de testplan staat, Xunit, worden de gevallen in code omgezet om vast te stellen of de verwachtte resultaat klopt.

# Bibliografie

Adam Marczak. (2020, 02 04). *Youtube*. Opgehaald van Azure application Insights tutorial: https://www.youtube.com/watch?v=A0jAeGf2zUQ

Amazon. (2021, 01 10). *Amazon Documentation*. Opgehaald van AWS Documentation: https://docs.aws.amazon.com/

Baarda, B. (2014). *Dit is onderzoek.* Nederland: Noordhoff.

Bass, L. (2012). *Software Architecture in Practice.* Pearson Education.

C4-model. (2021, 06 07). *C4-model*. Opgehaald van C4-model: https://c4model.com/

Chilkat. (2021, 01 01). *Licensing explained*. Opgehaald van ChilkatSoft: https://www.chilkatsoft.com/licensingExplained.asp

C-sharpCorner. (2019, 06 19). *Working with timer-triggered azure functions*. Opgehaald van C-sharpCorner: https://www.c-sharpcorner.com/article/working-with-timer-trigger-azure-functions/

C-shartcorner. (2020, 07 28). *Azure Application Insight resources*. Opgehaald van Azure Application insight: https://www.c-sharpcorner.com/article/azure-application-insights-resources-classic-and-workspace-based/

DataShieldprotect. (2021, 05 20). *Whatis Azure Monitor*. Opgehaald van Datashieldprotect: https://www.datashieldprotect.com/blog/microsoft-azure-monitor

Docs Microsoft. (2021, 06 01). *Ping Class*. Opgehaald van Ping: https://docs.microsoft.com/en-us/dotnet/api/system.net.networkinformation.ping?view=net-5.0

Imolkova. (2019, 06 08). *Depreciating TelemetryConfigutation.active*. Opgehaald van Github: https://github.com/microsoft/ApplicationInsights-dotnet/issues/1152

Jim Highsmith. (2001, 01 01). *The Agile manifesto*. Opgehaald van TheAgileManifesto: https://agilemanifesto.org/history.html

Keith. (2017, 06 13). *Ping restrictions*. Opgehaald van Stackoverflow: https://stackoverflow.com/questions/44532923/azure-functions-ping-restrictions

Kunal. (2020, 11 16). *Codez Up*. Opgehaald van What is Domain-driven design (DDD): https://codezup.com/what-is-domain-driven-design-ddd-pros-cons/

Microsoft. (2016, 06 16). *Psping v2.1*. Opgehaald van Docs microsoft: https://docs.microsoft.com/zh-cn/sysinternals/downloads/psping#installation

Microsoft. (2018, 04 09). *Docs Microsoft*. Opgehaald van Code and test Azure Functions Locally: https://docs.microsoft.com/en-us/azure/azure-functions/functions-develop-local

Microsoft. (2019, 02 18). *Azure Function triggers en bindingsconcepten*. Opgehaald van Docs Microsoft: https://docs.microsoft.com/nl-nl/azure/azure-functions/functions-triggers-bindings?tabs=csharp#supported-bindings

Microsoft. (2019, 03 25). *Strategies for testing your code in Azure Functions*. Opgehaald van Microsoft Docs: https://docs.microsoft.com/en-us/azure/azure-functions/functions-test-a-function

Microsoft. (2020, 11 05). *Application Insights API for custom events and metrics*. Opgehaald van Microsoft Docs: https://docs.microsoft.com/en-us/azure/azure-monitor/app/api-custom-events-metrics#trackmetric

Microsoft. (2020, 05 11). *Application Insights API for custom events and metrics*. Opgehaald van docs.microsoft: Application Insights API for custom events and metrics

Microsoft. (2020, 04 28). *Custom metric collection in .NET and .NET Core*. Opgehaald van Docs Microsoft: https://docs.microsoft.com/nl-nl/azure/azure-monitor/app/get-metric#dimension-and-time-series-capping

Microsoft. (2021, 01 01). *File Class*. Opgehaald van Microsoft Docs: https://docs.microsoft.com/en-us/dotnet/api/system.io.file?view=net-5.0

Microsoft. (2021, 01 01). *HttpClientHandler.ServerCertificateCustomValidationCallback Property*. Opgehaald van Microsoft Docs: HttpClientHandler.ServerCertificateCustomValidationCallback Property

Microsoft. (2021, 01 10). *Microsoft Azure*. Opgehaald van Microsoft Docs: https://docs.microsoft.com/nl-nl/learn/azure/

Mikhail Shilkov. (2018, 04 1). *Is it possible to read File from same folder where Azure function exists*. Opgehaald van StackOverflow: https://stackoverflow.com/questions/49597721/is-it-possible-to-read-file-from-same-folder-where-azure-function-exists

Monterail. (2019, 8 19). *The differences between Iso 25010, ISO 9126 and ISTQB*. Opgehaald van Monterail: https://www.monterail.com/blog/software-qa-standards-iso-25010

PedroC88. (2019, 08 22). *Ping server from Azure Functions*. Opgehaald van Stackoverflow: https://stackoverflow.com/questions/57600698/ping-server-from-azure-function

ProductPlan. (2021, 12 05). *Moscow Prioritization*. Opgehaald van ProductPlan: https://www.productplan.com/glossary/moscow-prioritization/

Randen, H. J. (2015). *Aan de slag met Scrum.* Nederland: Academic service.

Rijksoverheid. (2019, 09 27). *Advies opslag Medische data in de cloud*. Opgehaald van Rijksoverheid Documenten: https://www.rijksoverheid.nl/documenten/rapporten/2019/10/08/advies-opslag-medische-data-in-de-cloud

TechRepublic. (2006, 09 22). *Understanding the pros and cons of the waterfall model of software development*. Opgehaald van Techrepublic.com: https://www.techrepublic.com/article/understanding-the-pros-and-cons-of-the-waterfall-model-of-software-development/

VitalityChicago. (2018, 04 01). *Agile Projects are more successful compared to traditional projects*. Opgehaald van https://vitalitychicago.com/blog/agile-projects-are-more-successful-traditional-projects/: https://vitalitychicago.com/blog/agile-projects-are-more-successful-traditional-projects/

# Bijlages

## Bijlage 1: Interview David Mesman

**David Mesman**

**Welke problemen komen de vaakste voor?**

Downtime, Feature bugs, waarschuwingen zoals packages tijdens developen.

**Heb je vaak Uptime problemen gehad?**

Nee niet altijd. Als er een uptime probleem is, dan zijn wij bezig met de server om bijvoorbeeld de server aan te passen. Desondanks hebben we wel paar keer problemen gehad met Cloudflare.

**Hebben SSL certificaten een einddatum?**

Ja. Deze staan er inderdaad bij en beëindigen op een bepaalde datum. En je wilt toch wel deze certificaten hebben waardoor je liever die certificaten wel hebt om het veilig te houden.

**Wanneer zou je willen weten dat zo een certificaat verloopt?**

Ongeveer een maand vantevoren zou je liever willen weten of een certificaat zal verlopen

Wanneer is een ping schadelijk en wil je op de hoogte van gehouden worden?

Ongeveer rond de 100 tot 150 ms. Dan beginnen de package losses en kan je aannemen dat er iets gaande is.

**Watvoor elementen zal je gemonitord willen zien?**

Uptime zou handig zijn. Database connectie? Aantal requests?

Voor uptime:

* Statuscodes
* Weekgrafiek uptime
* Weekgrafiek responstime

De responstimes deed bijvoorbeeld een website heel erg traag omdat een inspecteur veel informatie te zien kreeg. Doordat er geen indexes waren in de database deed de query er heel lang over. Dit is dus handig om meerdere elementen tegelijk te controleren.

**Zal het handig zijn voor je om zelf sub-url’s toe te voegen om te kunnen monitoren?**

Dat zou handig zijn inderdaad. Alleen is het lastig om een bepaalde URL’s te monitoren omdat je een account voor nodig hebt. Misschien kan je dit vastzetten in je contract om zo een specifieke dummy account voor te hebben.

**Komen cross-browser problemen voor?**

Niet altijd. Ik heb wel af en toe CSS verschillen tussen browsers maar dat is geen ticket waardige dingen.

**Welke limitaties zijn er bij monitoring?**   
Toegang tot persoonsgegevens is belangrijk. Gevoelig informatie mag niet publiekelijk.

## Bijlage 2: Interview Peter Hofman

**Peter hofman**

**Is David de enige die de support tickets oppakt?**

Ja, Officieel werkt David voor Uptime Heroes. Dit is een ander bedrijf maar hetzelfde eigenaar. Dit hebben we gedaan voor financiële redenen en om scheiding in te houden op papier maar we werken samen.

**Waarom wordt er geen gebruik gemaakt van Azure application insights**Azure application insight bied inderdaad een monitoringservice aan. Dit was een goed systeem alleen is er een functionaliteit verwijderd waarbij multi-step testing is verwijderd. Deze functionaliteit is in principe wat wij nodig hadden om te kunnen monitoren voor klanten.

**Waarom is er voor Azure gekozen in plaats van Amazon?**

Wij werken met Azure omdat zij hun certificaten in orde hebben. Daarnaast zijn er in EU bepaalde regelgevingen waar Amazon niet aan voldoet. Wij hebben banken als klant en soms zelfs medische data. Financiële en medische data mag land van oorsprong niet uit. Azure doet dit door middel van regions en daarmee kan de data opgeslagen worden binnen de datacenter in Amsterdam. Amazon doet dit wel binnen de EU maar dan is het de land uit. Google is nog erger. Ze zeggen dat het in principe binnen het land van oorsprong blijft maar ze vallen onder de Amerikaanse wet. Zij hoeven niet de Europese wetgeving te volgen en is dus gelijk een no-go.

**Welke type testen zijn er nodig?**

In principe moeten wij externe testen doen. Dit zijn testen waarbij de codebase van de gemonitorde applicatie niet bereikbaar is en dus moeten we het op een andere manier doen. We kunnen bijvoorbeeld pingtesten en kijken of er een uptime is. Het testen van de certificaten kunnen ook van belang zijn binnen de MonitoringApp. En vooral latern kunnen we een soort multi-step testing maken. Dit zat in een oudere versie van Application insight maar niet meer in de nieuwere. Dit is een manier om de content van de website te controleren waar wijzelf de handeling kunnen bepalen.

**Wat is het uiteindelijke doel van de monitoringssysteem?**

Technisch gezien is het verkrijgen van inzichten. We onderhouden systemen en vanuit technische oogpunt wil je weten waarom iets niet werkt.

Zakelijk gezien is het een manier voor ons om geld te verdienen. Nu doen we reactief onderhoud. Op die manier laten we David die oppakken. Waar we naar toe willen is door de monitoring proactief te doen, punten naar voren halen die onderhoudt kunnen worden.

**Voor nu hebben we twee monitoringselementen. Uptime en de multistep testing. Welke soort testen denk je in de toekomst nog hieraan toe te voegen?**

Om het klein te houden beginnen we met een uptime test. Later kunnen we meerdere soorten testen uitvoeren. Voorbeelden zijn multi-step web tests en Certificaatcheck (Certificaten hebben een bepaalde levensduur. Meestal 1 of twee jaar)

**Willen we ook verschillende browsers testen?**

Ja, dit is toekomst. Hiermee bouw je test-automatisering mee op. Maar dit is voor iets laters. Dit is voor na het multistep-testen.

Uptime:

**Wanneer er een beschikbaarheidsprobleem voordoet, wie brengen we op de hoogte daarvoor?**

De klant en wijzelf

**Wanneer wordt de klant op de hoogte gebracht?**

Zodra het probleem zichtbaar is zal iedereen op de hoogte gebracht moeten worden. Deze problemen moeten ook gerapporteerd kunnen worden op een dashboard. Hiervoor is Azure Monitor wel goed voor. We kunnen de resultaten hiernaar uploaden via een API.

**Is alle communicatie vanuit de monitoringssoftware over email?**

Ja

**Wanneer worden de testen uitgevoerd?**

Uptime monitor: Elke vijf minuten default

De rest van de tests zal configureerbaar moeten blijven.

**Is het mogelijk om extra accounts te regelen voor monitoring voor zoals de inspecteur om sub-url's mee te monitoren?**

Dit lossen we op aan de hand van de Multistep testing door variabele mee te geven aan de test. Hiermee kunnen we de bot laten inloggen aan de hand van de opgegeven account.

## Bijlage 3: Plan van aanpak

Plan van Aanpak

Auteur: Necati Unal

Datum 21-04-2021

Inhoudsopgave

[Aanleiding 95](#_Toc85156013)

[Probleemstelling 95](#_Toc85156014)

[Doelstelling van het project 96](#_Toc85156015)

[Risicoanalyse 96](#_Toc85156016)

[Technische risico 96](#_Toc85156017)

[Organisatorisch 96](#_Toc85156018)

[Functioneel 97](#_Toc85156019)

[Aanpak 98](#_Toc85156020)

[Architectuur 98](#_Toc85156021)

[Testplan 99](#_Toc85156022)

[Ontwikkeling 99](#_Toc85156023)

[Requirements 99](#_Toc85156024)

[DevOps 99](#_Toc85156025)

[Planning 100](#_Toc85156026)

# Aanleiding

Jongens van Techniek is opgericht door Dhr. Peter Hofman. Het is een commercieel software engineering bedrijf die zich begeeft op het ontwikkelen van oplossingen voor klanten met specifieke wensen. Daarnaast bieden ze ondersteuning aan voor deze oplossingen. Het klantenbestand van Jongens van Techniek bevat onder andere Ministerie van Economische zaken, Vodafone en Open University.

Na levering van de oplossing kan de klant ervoor kiezen om een ondersteuningspakket aan te gaan. Hiermee is het mogelijk voor de klant om contact op te nemen met de supportdesk en om hulp te vragen bij problemen. Vervolgens wordt het probleem onderzocht door de support desk en wordt er een poging gedaan tot oplossen

# Probleemstelling

Het probleem start zodra een (gedeelte) oplossing is opgeleverd aan de klant. De software wordt in gebruik genomen bij de klant en heeft Jongens van Techniek geen zicht meer over de software. De klant kan dan tegen problemen aanlopen. Deze problemen zijn zoals het niet beschikbaar zijn van de applicatie (downtime) of een volle database (capaciteitsproblemen). Klanten die software in gebruik nemen zijn niet op de hoogte van problemen die optreden tijdens de levensduur van de applicatie. De eerder benoemde problemen kunnen ertoe doen dat klanten de opgeleverde oplossing niet meer kunnen gebruiken.

Jongens van Techniek heeft een supportdesk waar klanten terecht kunnen. Het probleem wordt bij de support desk behandeld en zo een oplossing voor gemaakt. Hiermee wordt er reactief gereageerd bij een probleem. Dit betekent dat er alleen gereageerd wordt, zodra een probleem heeft afgespeeld en is gemeld bij de support desk. Het probleem kan groot zijn, waardoor een systeem plat komt te liggen en daarbij ook de workflow. Om dit op te lossen kost dit de klant geld, tijd en energie. Daarnaast zal er mankracht gereserveerd moeten worden om het probleem op te lossen. Alhoewel het fijn is voor de klant om ergens heen te kunnen voor hulp wanneer een probleem optreedt, is het beter dat er überhaupt geen problemen optreden. Nu het applicatielandschap is vergroot, is het een grote taak voor de support desk om al deze problemen na te lopen en op te lossen.

Jongens van Techniek wil zorgen dat er preventief gemonitord en geacteerd wordt. Een voorbeeld is een volle database. Dit kan tegengegaan worden door het geheugen van de database te monitoren en aan te vullen zodra deze een grens heeft bereikt. Op deze manier wordt de uptime gehandhaafd en heeft de klant controle en kan vroegtijdig inspelen op problemen die zullen optreden.

Daarom wil Jongens van Techniek bij hun onderhoudsabonnement een monitoringsservice aanbieden aan hun klanten. Dit is een preventieve oplossing voor de eerder benoemde problemen die voorkomen. Qua complexiteit kan dit vrij snel oplopen doordat er meerdere applicaties gemonitord moeten worden. Dit resulteert in de wens dat de monitoring flexibel ingezet kan worden.

Alle bovengenoemde problemen opgesomd:

1. Geen zicht meer op de software wanneer deze online gaat.
2. Problemen kunnen niet tegengehouden worden. Oplossing vindt pas plaats nadat een probleem is afgespeeld.
3. Meerdere projecten betekend meer problemen die kunnen voordoen.

# Doelstelling van het project

Het doel van het project is om binnen de afstudeerperiode een onderzoek te doen naar de manier van monitoring binnen de Cloud wereld en daarna een Minimal Viable Product (MVP) op te leveren. De oplossing zal moeten voldoen aan de vooraf gedefinieerde hoog geprioriteerde requirements en als er tijd voor over is de lager geprioriteerde requirements. Daarnaast is het doel om de monitoringsapplicatie te koppelen aan een applicatie waarop de monitoring is toegepast.

# Risicoanalyse

Tijdens het project zijn er een aantal risicofactoren die invloed kunnen hebben tot het project. Met behulp van een risicoanalyse kunnen we de vooraf gesignaleerde risico’s beschrijven met de daarbij de maatregel die genomen moeten worden tegen het probleem.

## Technische risico

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nr #. | Bedreiging | Kans (1 – 5) | Impact (1-5) | Risico (K \* I) |
| 1 | Beperkte cloud kennis | 2 | 5 | 10 |
| 2 | Verlies van code door slechte code versioning. | 1 | 5 | 3 |
| 3 | Geen toegang hebben tot een andere applicatie voor monitoring | 3 | 5 | 15 |

**Tegenmaatregel #1:**

Bronnen opzoeken die te maken hebben met Cloudoplossingen voor Monitoringssystemen.

**Tegenmaatregel #2:**

Gebruik maken van Git zodat code versioning te volgen is. Dit zal ook goed bijgehouden moeten worden door de ontwikkelaar zelf zodat code local en in de cloud up to date is.

**Tegenmaatregel #3:**

Het monitoringssysteem zal getest moeten worden voor functionaliteit door deze te koppelen op een andere applicatie. Dit kan belemmerd worden door geen toegang hebben tot een andere applicatie. Het is mogelijk om gebruik te maken van een dummy applicatie zodat de monitoring op gebouwd kan worden.

## Organisatorisch

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nr #. | Bedreiging | Kans (1 – 5) | Impact (1-5) | Risico (K \* I) |
| 1 | Beperking op het gebied van Business logic | 5 | 5 | 25 |

**Tegenmaatregel #1:**

Deze bedreiging kan tegengegaan worden door het interviewen van de beschikbare stakeholders binnen Jongens van Techniek. Door interviews af te leggen en de juiste informatie te vergaren is het mogelijk om de correcte requirements op te stellen.

## Functioneel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nr #. | Bedreiging | Kans (1 – 5) | Impact (1-5) | Risico (K \* I) |
| 1 | Onderschatten benodigde tijd voor stories | 2 | 3 | 6 |
| 2 | Storen van de gemonitorde applicatie | 2 | 5 | 10 |

**Tegenmaatregel #1:**

Stories klein houden en volgens de User story regels. Deze goed formuleren zodat er geen miscommunicatie ontstaat en de benodigde werk ervoor duidelijk in taken verdeeld zijn.

**Tegenmaatregel #2:**

Zorgvuldig testen zodat er geen belemmering ontstaat tussen de monitoringsysteem en de desbetreffende applicatie.

# Aanpak

Tijdens het project zal gebruik gemaakt worden van een ontwikkelingsmethode. Hier kan de Waterval of de Agile methode voor gekozen worden.

Waterval-methode bestaat al sinds 1970. Waterval-methode is een framework waarin ontwikkeling in stappen volgt naar het eindproduct. De stappen van de waterval methode (TechRepublic, 2006) zijn als volgt:

1. Requirements analysis
2. Design
3. Implementation
4. Testing
5. Installation
6. Maintenance

De voordelen van watervalmethodiek is dat het discipline eist. Er is een start en een eind voor elke stap. Start en eind van een fase is duidelijk aangegeven. Door focus op requirements en design te leggen voordat ontwikkeling start zorgt voor minimale verspilling van tijd en moeite. Hier worden alle requirements met de klant besproken om zo een goed mogelijk requirementsroverzicht te kunnen creëren waaruit verder ontwikkeld kan worden.

Doordat waterval vereist dat er in stappen gewerkt wordt is het lastig om terug te gaan naar de requirements in het geval deze later veranderen. Binnen waterval is het lastig om terug te gaan naar de vorige stap. Het is in dit geval lastig om requirement-wijzigingen toe te passen als de klant zijn gedachte heeft veranderd.

Om deze probleem op te kunnen lossen is het Agile framework in het leven geroepen. Onder Agile vallen meerdere soorten ontwikkelmethodes waaronder Scrum. Uit een onderzoek (VitalityChicago, 2018) is gebleken dat traditionele ontwikkelingsmethodes zoals waterval twee keer minder succesvol waren in het slagen van het project vergeleken met een Agile ontwikkelmethode. Scrum is één van de vele Agile ontwikkelmethodes die ingezet kan worden binnen een team. Het zorgt ervoor dat de ontwikkeling van software op een effectieve en flexibele manier verloopt. Scrum heeft feedback loops binnen elke stap van het project en er kan terug op gekomen worden zodra wijzigingen voordoen. Dit zorgt voor dat aanpassingen vanuit de klant binnen het ontwikkelproces genomen kan worden. Een ander voordeel van Scrum is dat de klant volledig betrokken raakt bij het ontwikkelproces en kan hij het ontwikkelproces meemaken en feedback leveren wanneer dat nodig is.

# Architectuur

Om de architectuur te kunnen ontwikkelen voor het monitoringssysteem is eerst van belang dat kennis opgedaan zal worden op de bestaande architectuur die gehandhaafd wordt binnen Jongens van Techniek. Vervolgens zal er aan de hand van de requirements onderzocht worden welke architectuur ontwikkelmethode het beste past bij het project.

Met de ontwikkelmethode zal de architectuur worden ontwikkeld. Deze architectuur zal de basis vormen voor het project en zal rest van de afstudeerperiode ontwikkeld worden.

# Testplan

Om de kwaliteit van de code en het product te kunnen garanderen zal er vooraf een testplan worden ontwikkeld. In dit document zal je maatregelen treffen die gebruikt worden om de kwaliteit van het product te waarborgen. Na ontwikkeling van een user story zal deze getest moeten worden aan de hand van de testmethode dat is beschreven.

# Ontwikkeling

Ontwikkeling met scrum betekend dat progressie incrementeel zal vorderen. Iedere twee weken, oftewel een sprint, wordt een aantal user stories binnen een sprint genomen voor ontwikkeling. Deze stories zijn gekozen aan de hand van de prioriteringsmethode die zal gehanteerd worden tijdens het project.

Ontwikkelen van het platform wordt gedaan op Cloud basis. Hiervoor zal Azure gebruikt worden doordat het bedrijf daar een partnerschap mee heeft. Als laatst wordt het project ontwikkeld in C#. Doordat niet alle functionaliteiten van de cloud-platform duidelijk zijn, zal onderzocht moeten worden welke functies beschikbaar zijn. Mocht budget een rol spelen binnen deze beslissing dan zal hierover ook beslist moeten worden door de bedrijfsmentor.

# Requirements

Een ander kritiek onderdeel is het vaststellen van de requirements. Deze worden geleverd door mijn mentor (en als het nodig is andere collega’s). Samen met de context en opdrachtomschrijving die is geleverd wordt er een prioriteitenlijst gemaakt. Daarvan uit kan een huidige situatie, de wens en eisen van de stakeholders boven water worden gehaald. De prio-lijst zal opgesteld worden op basis van de MoSCoW methode. Om te voorkomen dat alle user stories op de hoogste prio beland, zal samen met de stakeholder gereviewd worden om de belangrijke items hoger te prioriteren en de nice-to-haves een minder belangrijke prioriteit krijgen. De stories worden beschreven aan de hand van de User Story regels (As a < type of user >, I want < some goal > so that < some reason >).

# DevOps

Voor versiebeheer van de applicatie zal DevOps gebruikt worden. DevOps geeft ons de mogelijkheid om de deployment en de infrastructuur te kunnen realiseren. DevOps geeft daarnaast ons de mogelijkheid om de best practices te kunnen gebruiken zoals Continious integration en Continious delivery. Voor Devops zal gebruik gemaakt worden van Azure. Dit is een organisatorisch limitatie doordat Jongens van Techniek in het verleden projecten voor banken en gemeenten heeft gerealiseerd. Dit zijn gevoelige gegevens die men in Nederland wilt houden. Door Azure te gebruiken is ervoor te zorgen dat deze gegevens Europa niet verlaat.

# Planning

In de bijlage vind je de planning voor de komende werkzaamheden. Deze zullen een plaats krijgen in de rooster volgens de indruk die ik heb gekregen van de moeilijkheidsgraad die het project heeft. Voor het project is er gebruik gemaakt van een Sprint 0 om een zo goed mogelijk beeld te krijgen van het systeem. Verder ook Sprint 0 gebruiken om de requirements vast te stellen voor het project. Elke sprint zal ervoor gezorgd worden dat de stories goed beschreven zijn en dat er tickets aangemaakt zijn voor de desbetreffende stories. Voor mezelf zal ik wel gebruik maken van de standaard user story structuur omdat dit voor mij duidelijker is. Alhoewel daarnaast zal ook dat de omschrijving voor de user story van pas komen om de user story te kunnen beschrijven.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sprint |  | |  | | 1 | | 2 | | 3 | | 4 | | | 5 | | | 6 | | | 7 | | | 8 | | | 9 | | | 10 | | | 11 | |
| Week | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | | 14 | 15 | | 16 | 17 | | 18 | 19 | | 20 | 21 | | 22 | 23 | | 24 | 25 | |
| Plan van aanpak |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |
| **Ontwerpen** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Onderzoek |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |
| Ontwikkelen |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |
| Testen |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |
| Architectuur ontwikkelen |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |
| **Requirements** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Requirements vergaren |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |
| Initiële backlog |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |
| Sprint backlog |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |
| **School** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Bedrijfsbezoek (online) |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |
| Afstudeerdossier |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |

## Bijlage 4: Requirements Rapport

Requirements rapport

Jongens van Techniek

Auteur : Necati Unal

Datum : 05-05-2021

Opdrachtgever : Jongens van Techniek

Inhoudsopgave

[Inleiding 103](#_Toc85156253)

[Scope van de opdracht 104](#_Toc85156254)

[Requirements 104](#_Toc85156255)

[MoSCoW 104](#_Toc85156256)

[Must have 104](#_Toc85156257)

[Should have 105](#_Toc85156258)

[Could have 105](#_Toc85156259)

[Will not have 105](#_Toc85156260)

[Problemen bij prioriteren 105](#_Toc85156261)

[User requirements 106](#_Toc85156262)

[Systeem requirements 106](#_Toc85156263)

[Epics 107](#_Toc85156264)

[User stories 108](#_Toc85156265)

[Prioriteiten 109](#_Toc85156266)

# Inleiding

In dit document zal stappen gezet worden richting het verzamelen van de requirements van het project. Bij het verzamelen van de requirements komen de interviews en categorisatie hiervan bij kijken. In dit document worden deze requirements behandeld.

Het document zal beginnen met de scope van het project. Door de scope vast te stellen kan bepaald worden waar het project zal stoppen. Door grenzen te stellen is het mogelijk om vast te stellen wanneer het project als af bestempeld kan worden.

Door alle requirements vast te stellen zullen interviews plaatsvinden. In de komende weken zullen er een aantal interviews plaatsvinden. Aan de hand van deze interviews en het afstudeerplan is het mogelijk om de requirements boven water te halen. Vervolgens worden de requirements gecategoriseerd aan de hand van functioneel en niet-functionele requirements. De niet-functionele requirements worden aan de hand van de User story methode genoteerd.

# Scope van de opdracht

De opdracht is het ontwikkelen van een monitoringssoftware. Deze software is de oplossing op de problemen die zich voordoen na levering van de oplossing aan de klant. De monitoringssoftware moet preventief problemen opzoeken en de desbetreffende personen op de hoogte brengen.

Door tijdsgebrek zal grenzen getrokken moeten worden voor het project. Deze grenzen geven aan voor welke elementen binnen het project vallen en welke daar buiten.

De volgende elementen die binnen het project vallen:

* Het ontwikkelen van de monitoringssoftware
* Testen van het monitoringssoftware
* Koppeling tussen de gemonitorde software en de monitoringssoftware

De volgende elementen die buiten het project vallen:

* De gemonitorde software
* Accounts voor het inloggen op de gemonitorde software.

# Requirements

De requirements zijn voortgekomen uit de meerdere gesprekken met stakeholders, het afstudeerplan en de plan van aanpak. Vanuit deze bronnen zijn functionele en niet-functionele requirements uit gekomen evenals technische en business requirements.

Vanuit de requirements zullen Epics uit komen. Deze epics zijn op te splitsen in meerdere user stories. Epics en user stories worden in de User story formaat beschreven. Dit is als volgt:

Als een <rol> wil ik <een functionaliteit>, odat ik <reden voor de functionaliteit>. Met deze formulering kan de oorzaak van de functionaliteit beschreven worden. Dit is belangrijk omdat het doel van de functionaliteit verduidelijkt kan worden.

Vervolgens zullen de user stories geprioriteerd worden aan de hand van een prioriteringsmethode. Deze volgorde wordt gebruikt om een volgorde te kunnen maken voor implementatie van functionaliteiten. Dit zal gedaan worden aan de hand van de MoSCoW methode.

## MoSCoW

Tijdens het project zal MoSCoW aangehouden worden om de user stories te kunnen prioriteren. MoSCoW is een framework dat simpele termen gebruikt. Deze termen kunnen makkelijk begrepen worden gezien ze gebruikt zullen worden door de klant om een prioriteit te stellen. De methodology is een handige oplossing voor conflicten binnen stakeholders.

Met MoSCoW is het mogelijk om 4 levels van prioriteiten toe te wijzen aan een requirement (ProductPlan, 2021). Van hoge naar lage prioriteit als volgt:

1. Must have
2. Should have
3. Could have
4. Will not have (this time)

### Must have

“Must have” requirements zijn belangrijk voor de team. Deze requirements wordt als een hoge prioriteit beoordeeld door de klant en zal dus voorgang moeten hebben tijdens implementatie.

### Should have

Een stap lager zijn de “Should have” requirements. Deze requirements zijn belangrijk voor het project maar niet vitaal. Ze vormen geen belemmering en de klant zal zich niet druk om maken als dit later afgemaakt zal worden.

### Could have

Als derde prioriteit is de “Could have”. Deze prioriteit wordt ook de Nice-to-have genoemd. Alles wat onder deze prioriteit valt zal geen essentieel belang hebben tot het eindproduct. Ze hebben een smalle impact op het project.

### Will not have

Als laatste komt de “Will not have” prioriteit. Onder deze prioriteit vallen de requirements die voorlopig niet geïmplementeerd zullen worden. Deze requirements zijn dan ook niet relevant en hebben weinig tot geen impact op het project. Om ervoor te zorgen dat de requirements die hieronder vallen niet altijd verdwijnen worden ze wel.

## Problemen bij prioriteren

Men zal gauw streven naar requirements dat vallen onder “Must haves”. Desondanks is het onmogelijk om alles in een sprint te nemen voor implementatie. Dit is de zwakte van de MoSCoW, het feit dat alles onder “Must-have” kan vallen.

Hiervoor is het mogelijk om een numeriek systeem te gebruiken. Deze systeem zal ons helpen om zonder deze termen de requirements te prioriteren. Het is makkelijker om namens een cijfersysteem te kunnen prioriteren. Hiervoor worden cijfers gebruikt van 1 tot 10 waarbij 1 het hoogste en 10 de laagste prioriteit. Aan de hand van deze getallen wordt de eerder benoemde termen eraan gelabeld. Aan de hand van een tabel kan daarna de bijbehorende prioriteit eraan gekoppeld worden.

De tabel als volgt:

|  |  |  |
| --- | --- | --- |
| Nr #. | MoSCoW Prioriteit |  |
| 1 t/m 2 | Must have |  |
| 3 t/m 5 | Should have |  |
| 5 t/m 7 | Could have |  |
| 8 t/m 10 | Will not have (this time) |  |

# User requirements

De user requirements vormen de requirements voor een gebruiker. Hierin kan achterhaalt worden welke functionaliteit geimplementeerd moeten worden.

|  |  |
| --- | --- |
| Nr #. | User requirement |
| 1 | De supportmedewerker kan de uptime van de gemonitorde applicatie bekijken. |
| 2 | De supportmedewerker kan de certificaten van de gemonitorde applicatie bekijken. |
| 3 | De supportmedewerker kan een multi-step web test laten uitvoeren. |
| 4 | De klant krijgt een email wanneer er iets mis is met de software. |
| 5 | De klant kan aangeven of de benodigde werkzaamheden uitgevoerd kunnen worden. |
| 6 | De supportmedewerker krijgt een email wanneer er iets mis is met de gemonitorde software. |
| 7 | De supportmedewerker kan handmatig de uptime-tests uit laten voeren. |
| 8 | De supportmedewerker kan handmatig de certificaat-check uit laten voeren. |
| 9 | De supportmedewerker kan handmatig de multi-step webtest check uit laten voeren. |
| 10 | De supportmedewerker ziet welke werkzaamheden voorgesteld zijn. |
| 11 | De klant kan zijn dashboard inzien met alle statussen |

# Systeem requirements

In dit hoofdstuk worden de systeemrequirements oftewel de niet-functionele requirements behandeld. Dit zijn functionaliteiten die niet vereist worden door de gebruiker maar uitgevoerd zullen worden door het systeem. Daarom zijn deze niet altijd even zichtbaar voor de gebruiker maar wordt op de achtergrond uitgevoerd. Verder wordt aan de systeemrequirements een kwaliteitseis toegevoegd zoals beschreven in de ISO 25010. Deze ISO is een standaard voor de industrie om deze iso is een verbetering op de ISO 9126 vanwege de nieuwe karakteristieken die bij komen kijken zoals beveiliging (security) en compatibiliteit (compatibility) (Monterail, 2019). Verder is het een internationaal geaccepteerde software testing certificaat. Daarom is er besloten om voor ISO 25010 te gaan in plaats van ISO 9126.

|  |  |  |
| --- | --- | --- |
| Nr #. | Systeem requirement | Kwaliteitseis |
| 1 | Het systeem mag de gemonitorde applicatie niet storen bij dagelijkse functionaliteiten | Resource utilization |

# Epics

Vanuit de requirements ontstaan user stories. User stories kunnen onder een thema vallen genaamd een epic. Dit is een grotere functionaliteit waaronder meerdere user stories onder kunnen vallen. Deze is namelijk nog niet gesplitst in smallere en duidelijkere user stories. Dat wordt later in hoofdstuk 7 gedaan.

In de tabel hieronder zijn de epics beschreven die van belang zijn tijdens het project.

|  |  |
| --- | --- |
| Nr # | Epics |
| 1 | Als supportmedewerker wil ik de resultaten van de toegevoegde tests weten zodat ik weet of er een probleem zich af speelt of heeft gespeeld. |
| 2 | Als supportmedewerker wil ik een testbeheer zodat ik alle testen kan verwijderen, wijzigen en toevoegen. |
| 3 | Als klant wil ik een dashboard, zodat ik een overzicht heb van de status van mijn software. |
| 4 | Als klant wil ik weten wanneer een probleem is, zodat ik up to date ben van de status van mijn website. |

# User stories

De user stories in dit hoofdstuk vormen de laag onder een Epic. Deze user stories zijn functionaliteiten die onder de Epic vallen en geïmplementeerd worden aan de hand van de prioriteiten gesteld door de stakeholder. Per User story wordt de epic nummer bij genoteerd om duidelijk te maken onder welke epic deze behoort. Ten slotte zal de eerder benoemde systeem voor prioriteiten gebruikt om de user story te kunnen prioriteren.

|  |  |  |
| --- | --- | --- |
| Nr #. | Epics | User story |
| 1 | 1 | Als supportmedewerker wil ik inzicht hebben op downtimes zodat ik weet of er een probleem zich afspeelt. |
| 2 | 1 | Als supportmedewerker wil ik inzicht hebben op verlopen certificaten, zodat ik weet of er een probleem zich afspeelt. |
| 3 | 1 | Als supportmedewerker wil ik inzicht hebben op multi-step webtests, zodat ik de output kan valideren. |
| 4 | 1 | Als supportmedewerker wil ik een email ontvangen mocht er een downtime optreden. |
| 5 | 1 | Als supportmedewerker wil ik een email ontvangen mocht er een certificaat probleem optreden. |
| 6 | 2 | Als supportmedewerker wil ik endpoints wijzigen aan de lijst van de gemonitorde endpoints, zodat ik aanpassingen kan doen mochten de URL’s wijzigen. |
| 7 | 2 | Als supportmedewerker wil ik certificaten toevoegen aan de lijst van de gemonitorde certificaten, zodat ik bestaande en nieuwe certificaten kan aanpassen. |
| 8 | 3 | Als klant wil ik de uptime van mijn website inzien zodat ik weet op welke momenten mijn website niet beschikbaar is. |
| 9 | 3 | Als klant wil ik weten of mijn certificaat nog valide is zodat mijn website nog veilig en toegankelijk is. |
| 10 | 2 | Als supportmedewerker wil ik de testmomenten kunnen aanpassen, zodat ik zelf kan bepalen wanneer de tests uitgevoerd wordt. |
| 11 | 1 | Als supportmedewerker wil ik weten welke werkzaamheden voorgesteld zijn aan de klant, zodat ik kan zien dat er door ons al is gereageerd op een probleem. |
| 12 | 4 | Als klant wil ik een email ontvangen zodat ik weet dat de support op de hoogte is gesteld van de downtime. |
| 13 | 4 | Als klant wil ik een email ontvangen zodat ik weet dat de support op de hoogte is gesteld van een verlopen certificaat. |
| 14 | 4 | Als klant wil ik terugkoppelen, zodat ik weet dat de support het zal oplossen. |

# Prioriteiten

De ontwikkeling van de applicatie zal verlopen aan de hand van de prioriteiten die de stakeholders hieraan aangegeven hebben. Deze prioriteiten zijn van te voren bepaald aan de hand van zogenaamde “Chain” van epics om het zo logisch mogelijk te laten verlopen. Epic #1 zal voorrang hebben voor epic nummer #2. Desondanks kunnen deze prioriteiten wijzigen tijdens backlog refinement sessies. Daarnaast kan de volgorde wijzigen mocht een user story invloed hebben op een andere user story.   
  
De prioriteit die de user stories op dit moment krijgen zijn als volgt:

|  |  |  |
| --- | --- | --- |
| User story #1 | Prioriteit cijfer | Prioriteit volgens MoSCoW |
| 1 | 1 | Must have |
| 2 | 1 | Must have |
| 5 | 1 | Must have |
| 10 | 2 | Should have |
| 4 | 2 | Must have |
| 7 | 2 | Must have |
| 12 | 2 | Must have |
| 13 | 2 | Must have |
| 6 | 3 | Should have |
| 14 | 5 | Could have |
| 11 | 5 | Could have |
| 8 | 6 | Could have |
| 9 | 6 | Could have |
| 3 | 10 | Will be nice |

Tabel 6: Prioritering van de user stories

Architectuur Design

Auteur: Necati Unal

Datum: 26-05-2021

[Inleiding 112](#_Toc85156418)

[C4-Model 113](#_Toc85156419)

[Level 1: System context diagram 114](#_Toc85156420)

[Level 2: Container diagram 116](#_Toc85156421)

[Level 3 Component diagram 118](#_Toc85156422)

# **Inleiding**

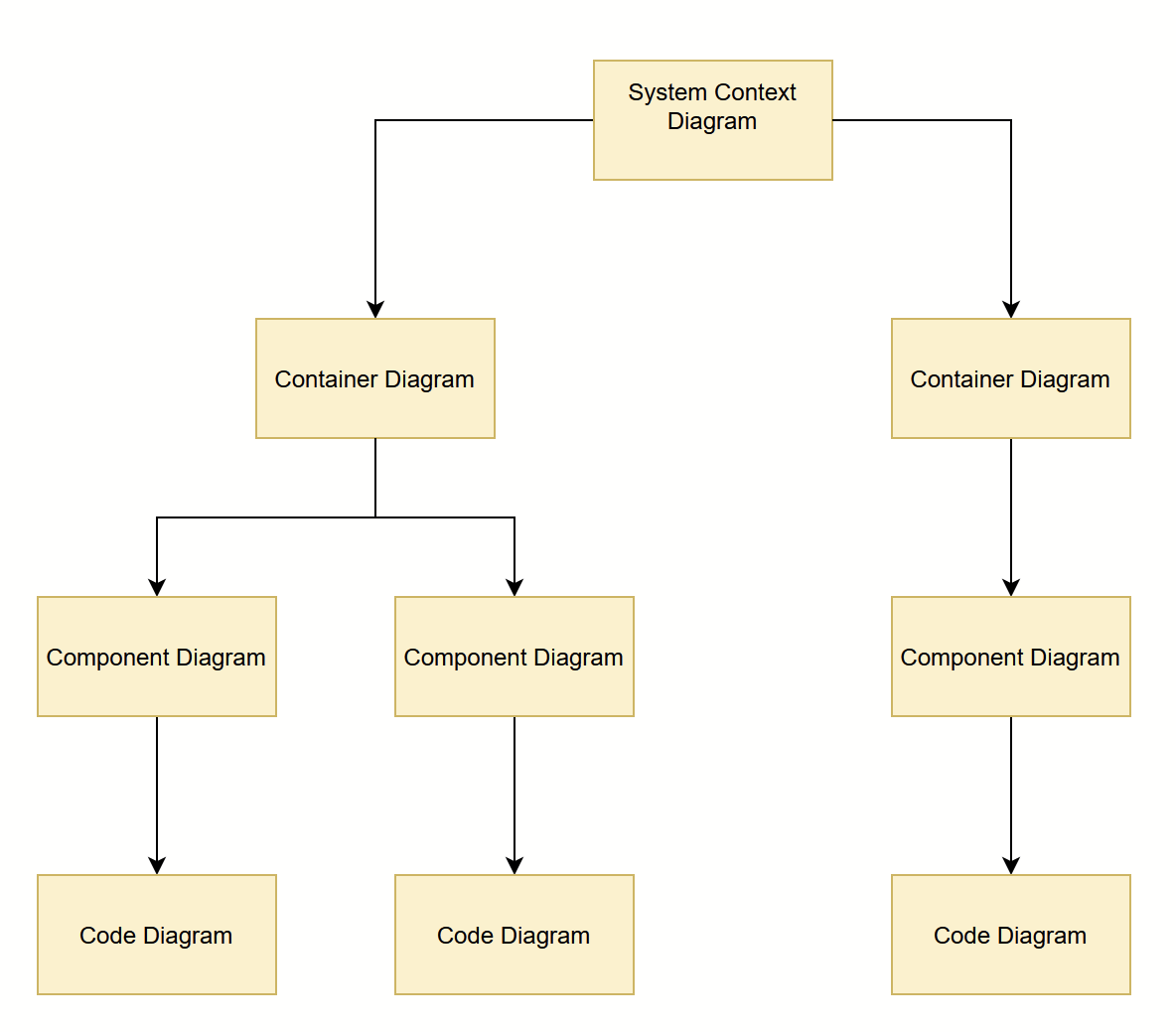
In dit document zal de totstandkoming van de architectuur gedocumenteerd worden. Dit doen we aan de hand van een architectuur ontwerpmethode. Deze methodiek is geselecteerd door de voordelen en nadelen tegen elkaar te zetten. Jongens van Techniek heeft geen vaste methodiek voor ontwikkelmethode van een architectuur. Daardoor zal de keuzen zelf gemaakt moeten worden.

Eerst zal een introductie gegeven worden aan het C4 model. Daarbij wordt uitgelegd hoe C4 in elkaar zit. Verder wordt ingelicht welke delen het C4-model inhoudt.

In de daarna komende alinea's zal de verschillende levels van het C4-model uitgevoerd worden. Het begint met Level 1 System Context Diagram. Het diagram dat gemaakt is wordt vervolgens vertoond en besproken. Per element binnen het systeem wordt een beschrijving gegeven

# C4-Model

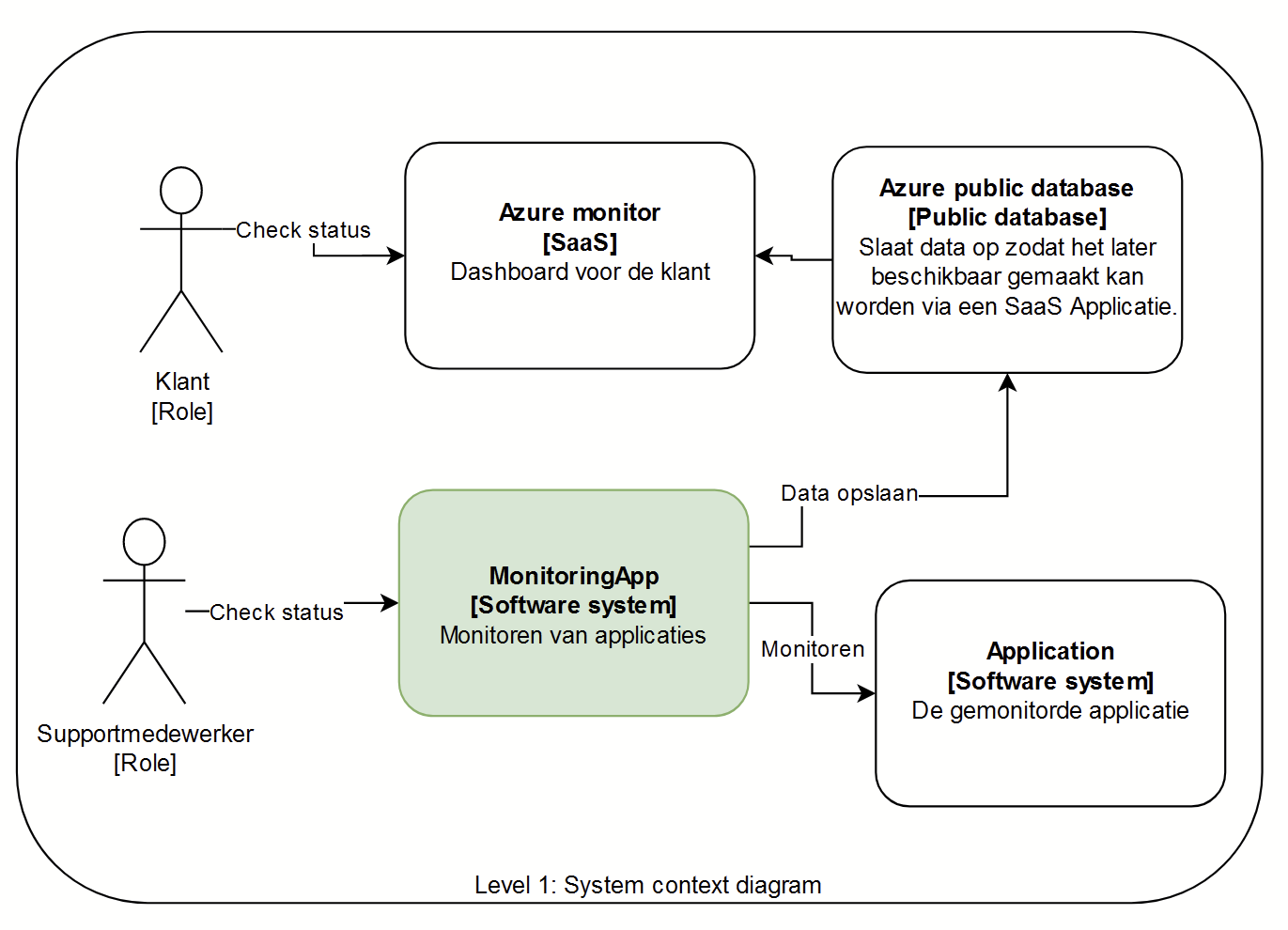
Voor de ontwikkeling van het Monitoringssysteem wordt gebruik gemaakt van het C4-model. Volgens Simon (C4-model, 2021) bestaat C4 uit vier verschillende levels. Deze zijn de System Context, Container, Component en Code diagram. Daarbij horen nog drie additionele diagrammen bij waaronder System Landschape, Dynamic en Deployment diagram als ondersteunende diagrammen. C4 is bedoeld om gebruik te kunnen maken van een algemene taal om het architectuur duidelijk te maken voor stakeholders. Daarvoor is gekozen om UML voor te gebruiken om de technische termen zo min mogelijk te houden. Wanneer er dieper in de levels gegaan worden zullen de technische termen terugkomen doordat de doelgroep van deze diagram wijzigen. De bovenste levels hebben de stakeholders en programmeurs als doelgroep. Onderaan schuift de doelgroep naar alleen de software developers.



Figuur Indeling C4-model architectuur ontwikkelmodel

## Level 1: System context diagram

In dit hoofdstuk zal het level 1 van het project worden gemodelleerd. Onder deze level vallen alle systemen die te in contact zullen komen met het ‘MonitoringsApp’. Hieronder vallen systemen zoals api's of mobiele applicaties en zelfs personen die gebruik van het systeem zullen maken. Deze personen worden als actoren of stakeholders beschouwd.



Figuur Systeem context diagram

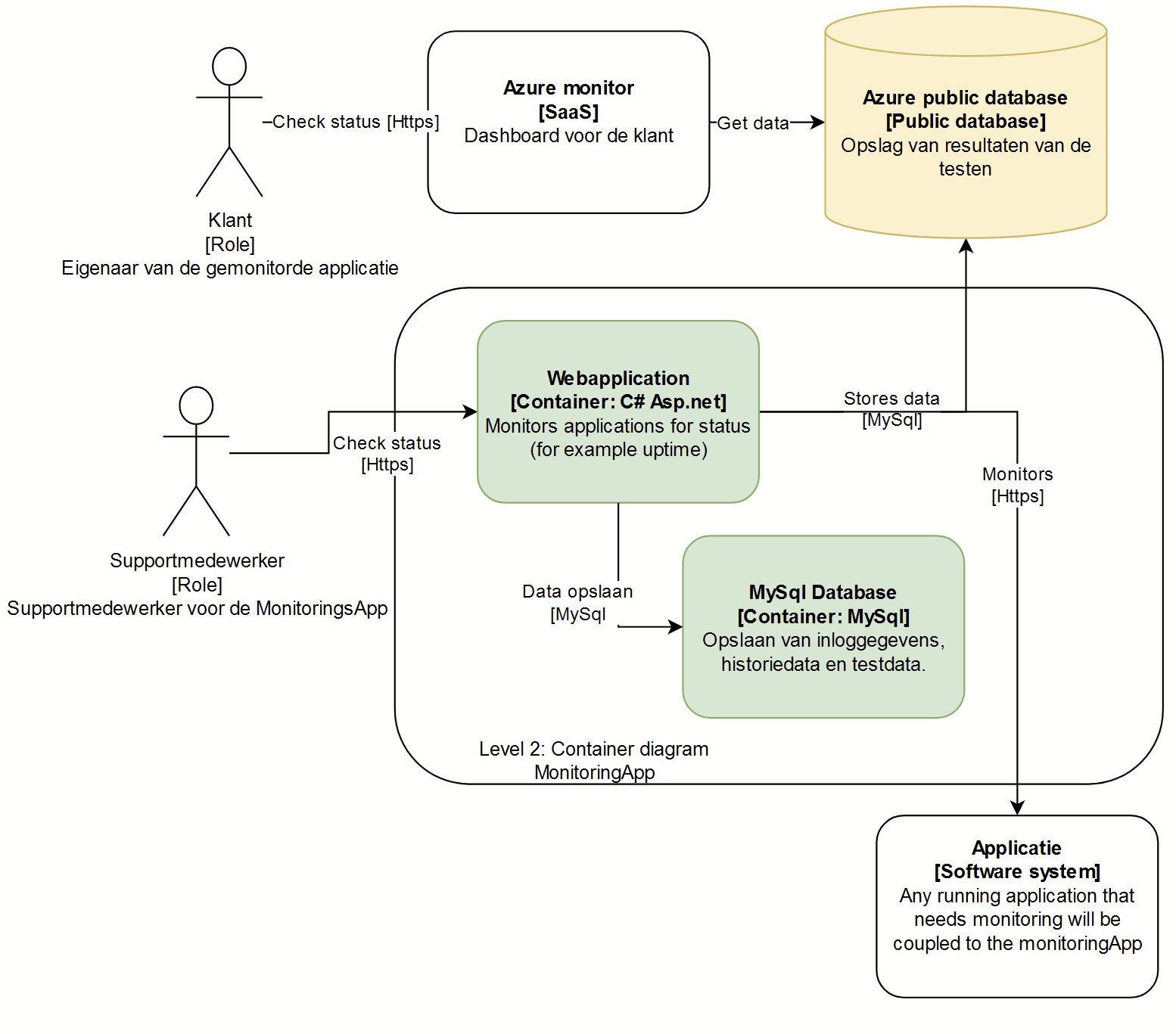
In figuur twee is te zien dat het systeem bekend is met meerdere systemen. Deze systemen horen niet bij de 'MonitoringsApp' en heeft dus buiten publiekelijk beschikbare data geen andere kennis over de desbetreffende systemen. De twee actoren die hierboven zijn opgenomen zijn de klant (de eigenaar van de gemonitorde software) en de supportmedewerker. De klant kan via een dashboard allerlei informatie opvragen om zo zelf te kunnen monitoren en in de gaten houden of alles goed gaat. De supportmedewerker kan dit doen via de 'MonitoringsApp' om ervoor te zorgen dat de status van de tests goed zijn. Hij kan administratieve handelingen in uitvoeren en zo de testen aanpassen mocht dat gewenst zijn.

|  |  |
| --- | --- |
| Component | Beschrijving |
| Klant | De eigenaar van de gemonitorde software die een onderhoudscontract is aangegaan bij JvT (Jongens van Techniek) |
| Supportmedewerker | De supportmedewerker bij UptimeHeroes. Deze persoon beheert de tests en kan de statussen inzien. |
| Application | De gemonitorde applicatie. De monitoringsapp zal deze applicatie monitoren om statussen te kunnen controleren. |
| MonitoringsApp | De applicatie dat ontwikkeld zal worden gedurende project. Deze 'MonitoringsApp’ monitort de applicatie die een onderhoudscontract is aangegaan. |
| Azure Public Database | Data vanuit de tests dat is verkregen zal opgeslagen worden in een public database. |
| Azure Monitor | De Azure Monitor is een dashboard voor de klant die een dashboard krijgt via de Azure Monitor applicatie. |

Tabel 1: Systeem context diagram beschrijving

## Level 2: Container diagram

Aangekomen op level twee zal de container diagram gemodelleerd worden. Doordat er meerdere diagrammen kunnen ontstaan is de focus gelegd op de component dat groen is gemarkeerd in Figuur één.



Figuur Level 2 Container diagram

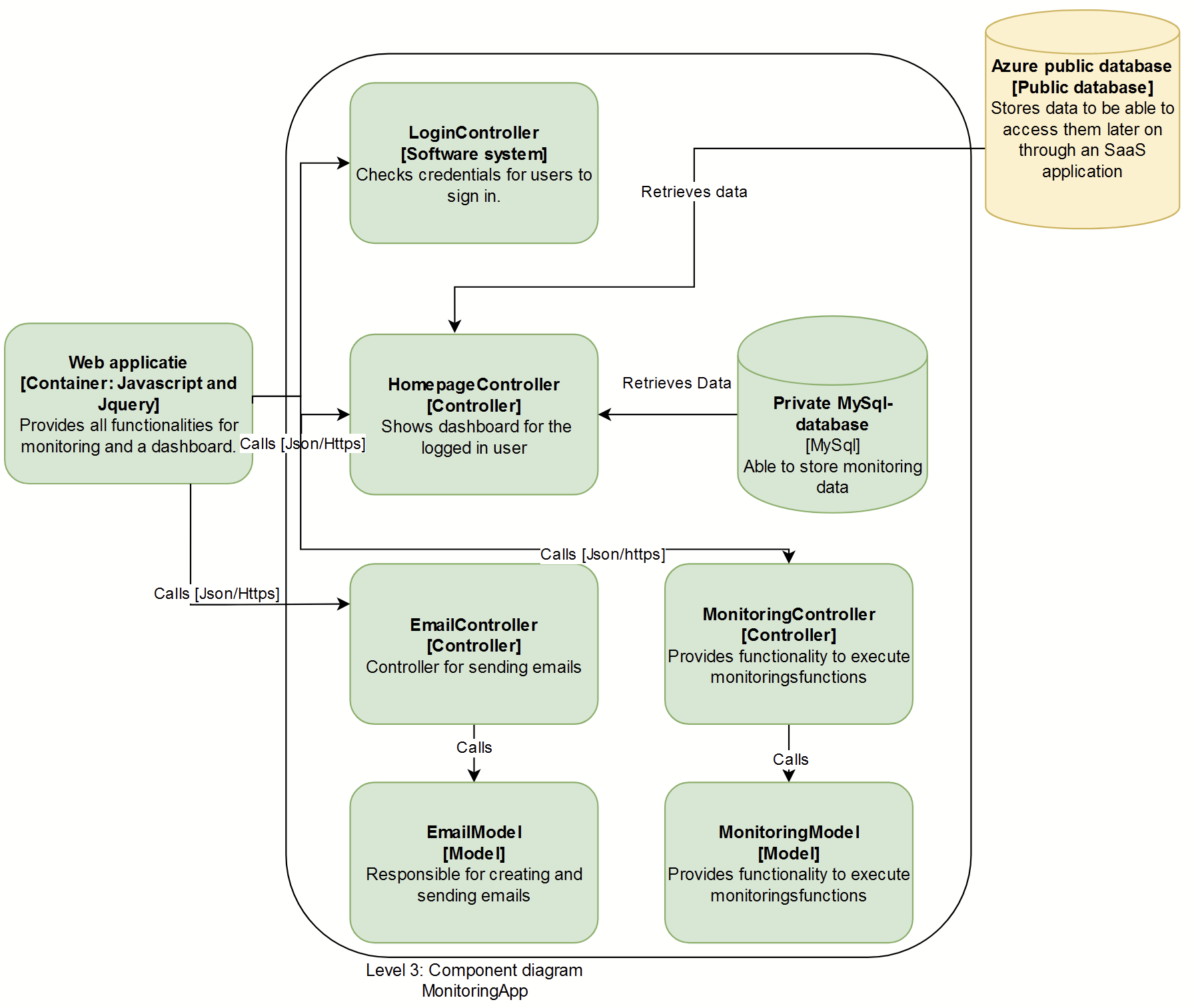
In figuur drie is te zien dat de webapplicatie verbonden is met twee databasen. Het geel gemarkeerde element is een publieke database waarin de resultaten van de tests in opgeslagen wordt. Deze opslag gebeurt door de webapplicatie. De webapplicatie heeft daarna nog een privé database waarin allerlei andere gegevens in worden opgeslagen zoals inloggegevens, wachtwoorden en test gegevens. Deze gegevens mogen niet publiekelijk beschikbaar zijn waardoor deze beslissing voor scheiding is genomen. De publieke database heeft vervolgens nog een functie daarnaast. Deze publieke database wordt gebruikt door een SaaS-applicatie. Deze applicatie is een dashboard voor de klant om te kunnen zorgen dat er een kant en klaar dashboard beschikbaar is voor het dashboard.

|  |  |
| --- | --- |
| Component | Beschrijving |
| Klant | De eigenaar van de gemonitorde software die een onderhoudscontract is aangegaan bij JvT (Jongens van Techniek) |
| Supportmedewerker | De supportmedewerker bij UptimeHeroes. Deze persoon beheert de tests en kan de statussen inzien. |
| Webapplicatie | De monitorings-applicatie zelf. Deze applicatie zal gebruikt worden om andere applicaties te kunnen monitoren. |
| MySql Database | De database dat gebruikt zal worden voor verschillende data dat niet publiekelijk beschikbaar mag zijn zoals inloggegevens en testdata. |
| Applicatie | De gemonitorde applicatie. |
| Azure public database | Resultaten van de tests worden opgeslagen in een public database. Deze wordt daarna gebruikt om een dashboard van te maken in een SaaS-applicatie. |
| Azure Monitor | De Azure Monitor is een dashboard voor de klant die een dashboard krijgt via de Azure Monitor applicatie. |

Tabel 2: Container diagram beschrijving

## Level 3 Component diagram

In level twee zal het groen gemarkeerde gedeelte in focus komen. In deze level is besloten om alle groen gemarkeerde delen op te nemen in één diagram omdat alle componenten dichtbij gerelateerd zijn. Hierdoor kunnen ze in één diagram gemodelleerd worden.



Figuur Component diagram

Links in figuur vier is het startpunt de webapplicatie. Deze webapplicatie is het front-end voor de gebruiker en zal gebruikt worden door de supportmedewerker. De login controller checkt voor authenticatie en zal de inloggegevens dat ingevoerd is door de gebruiker goedkeuren of afkeuren. Na inlog zal de gebruiker doorgestuurd worden naar de homepage controller waar het dashboard te vinden is.

Op deze dashboard zijn de meest belangrijke gegevens te zien van alle gemonitorde applicaties. Mocht er fouten zijn opgetreden dan komt de email controller van pas. Hiervan uit worden de e-mails verstuurd naar de desbetreffende personen mochten ze op de hoogte gehouden moeten worden.

|  |  |
| --- | --- |
| Component | Beschrijving |
| Web Applicatie | De monitorings-applicatie zelf. Deze applicatie zal gebruikt worden om andere applicaties te kunnen monitoren. |
| HomepageController | De controller voor het dashboard |
| LoginController | De controller voor het controleren van de login gegevens |
| Private Mysql-Database | De database dat gebruikt zal worden voor verschillende data dat niet publiekelijk beschikbaar mag zijn zoals inloggegevens en testdata. |
| EmailController | De controller dat gebruikt zal worden om het versturen van e-mails te kunnen starten. |
| EmailModel | Het model waarin de e-mails gestructureerd zullen worden aan de hand van het doel van de e-mail. |
| MonitoringController | De controller voor het monitoren van de applicaties. Deze controller zal het monitoren aansturen. |
| MonitoringModel | Een model voor het uitvoeren van de tests. |
| Azure public database | Resultaten van de tests worden opgeslagen in een public database. Deze wordt daarna gebruikt om een dashboard van te maken in een SaaS-applicatie. |

## Bijlage 5: Onderzoek Architectuur Ontwikkelmethode

Architectuur ontwikkelemethode

Auteur : Necati Unal

Opdrachtgever : Jongens van Techniek

Inhoudsopgave

[Inleiding 122](#_Toc85156761)

[Architectuur ontwikkelmethode 123](#_Toc85156762)

[Domain Driven Design (DDD) 123](#_Toc85156763)

[ADD 124](#_Toc85156764)

[Stappen van ADD 124](#_Toc85156765)

[C4 128](#_Toc85156766)

[Level 1: System Context Diagram 128](#_Toc85156767)

[1.3.2 Level 2: Container diagram 129](#_Toc85156768)

[Level 3: Component diagram 129](#_Toc85156769)

[Level 4: Code 129](#_Toc85156770)

[Keuze 130](#_Toc85156771)

# Inleiding

Dit document dient voor het beschrijven van de besluitvormsing voor de architectuurontwikkelmethode. In dit document komt de architectuur voor het project in te staan en de beslissingen die daarbij komen kijken. Gedurende het project wordt een architectuur gemaakt die als fundament zal plaatsnemen binnen het project. De architectuur is geen statisch document/grafiek maar meer een dynamisch document die gedurende het project aangepast zal worden mocht er wijzigingen nodig zijn aan de architectuur.

De methodieken, Attribute-Driven Design (ADD), Domain-Driven design(DDD) en C4 zijn bekend omdat ze eerder in het afstudeerprogramma zijn behandeld en zullen beoordeeld worden voor gebruik binnen het project.

Bij elk van deze methodes zal een korte omschrijving bij komen kijken met daaropvolgend een lijst met voor en nadelen. Dit zal belangrijk zijn omdat deze invloed heeft op de eindbesluit voor gebruik van een methodiek. Indien van toepassing zal daarna informatie gegeven worden over het werkproces die iedere ontwikkelmethode voorlegt.

# Architectuur ontwikkelmethode

## Domain Driven Design (DDD)

DDD is één van de ontwikkelmethodes voor software architectuur. DDD vereist robuuste domeinkennis en moedigt iteratieve werkwijze (agile) werkwijze aan wat tegenwoordig al door velen wordt gebruikt. Het doel van Domain Driven Design is het opstellen van een model die gelijk staat aan de werkelijkheid. Hierbij werk je samen met een domain expert. Deze persoon is een expert in het proces die de applicatie over zal nemen. Bij contact met de expert wordt gebruik gemaakt van “Ubiquitous Language” als taal. Deze taal is van tevoren afgesproken met de domain expert om er zeker van te zijn dat zowel de architect als de expert de begrippen begrijpen. Vervolgens kan de functionaliteit van het systeem genoteerd worden in deze ‘taal’ zodat de expert kan verifiëren dat dit correct is ten opzichte van de werkelijkheid. De concepten beschreven zullen dan als basis kunnen dienen voor een object-oriënted ontwerp later in de ontwikkelfase.

Voordelen van DDD zit bij dat een expert meekijkt tijdens het developen van de architectuur. Deze expert kent het domein waarvoor de software voor gemaakt zal worden. Door gebruik van een expert zorg je ervoor dat je onduidelijkheden kunt vermijden in je project.   
Daarnaast is een expert handig wanneer vaktermen gebruikt worden. Deze termen kunnen dan uitgelegd worden voor men die het domein niet kennen/begrijpen.

Voordelen DDD:

1. Vergemakkelijkt communicatie tussen tech en business
2. DDD is flexibel. Het creëert componenten die modulair en geëncapsuleerd zijn.
3. Benadrukt domein over interface

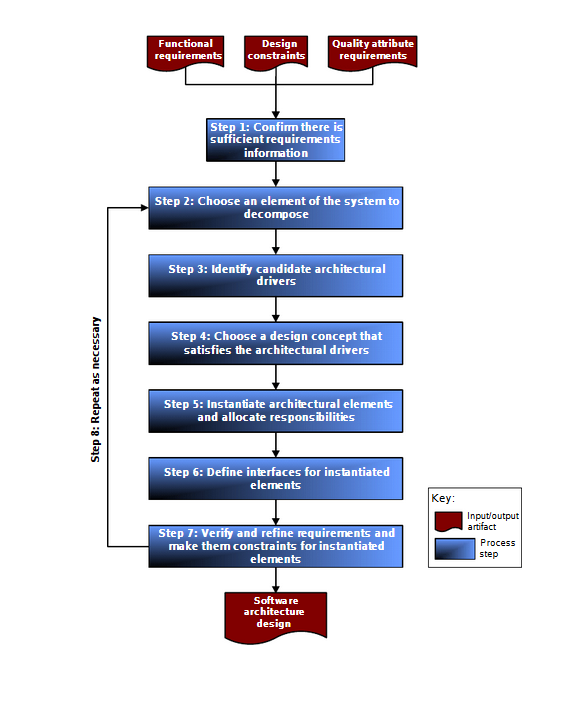
Nadelen DDD:

1. Slecht geschikt voor technische projecten. Het is wel ideaal voor projecten met grote mate van domeincomplexiteit (waarbij bedrijfslogica complex is).
2. Domeinexpert speelt een belangrijke rol in het ontwikkelproces omdat hij de go-to is. Wanneer dit te complex wordt, kan er problemen op latere termijn ontstaan omdat niet alle termen werden begrepen door teamleden.
3. Onhandig bij kleine projecten. DDD zorgt voor veel overhead.

## ADD

ADD is een ontwikkelmethode die zich focussed op het decomposen van een process waarbij stapsgewijs refinement plaatsvindt (Rob Wojcik, Felix Bechmann, Len Bass, Paul Clements, Paulo Merson, Robert Nord, Bill Wood, 2006). Daarnaast houdt ADD ook rekening mee met het vervullen van attribute qualities die het systeem nodig heeft. Het is een recursieve process waarbij elke ronde een tactiek, architectuur of een design pattern gekozen wordt om aan een quality attribute te voldoen.

Elke iteratie of ronde representeert een sprint, wat het werken met ADD Agile maakt. Tijdens deze rondes kunnen een serie aan iteraties uitgevoerd worden.



### Stappen van ADD

In het figuur hierboven wordt een pijltje vanuit stap 7 terug naar stap 2 getrokken. Dit is het iteratieve gedeelte van ADD, voordat een uiteindelijke software architectuur is gemaakt. Per stap heeft een software architect bepaalde verantwoordelijkheden die uitgevoerd zal moeten worden om een uiteindelijke architectuur te kunnen ontwerpen.

**Stap 1**

Stap 1 is het nagaan of de verzamelde requirements duidelijk zijn. Daarna moeten deze requirements geprioriteerd zijn volgens de business en de wensen van de klant. Vervolgens eist stap 1 dat er genoeg requirements zijn die goede informatie bevatten. Als dit niet het geval is, kan het iteratieve proces van ADD nog altijd ruimte bieden om deze requirements later mee te nemen.

**Stap 2**

Stap 2 zal meerdere keren uitgevoerd worden. Eerste keer wordt het systeem in zijn geheel genomen om te kunnen decomposen. Daarbij hoort standaard alle requirements dus zal alles meegenomen worden. Is het tweede keer dat stap 3 uitgevoerd zal worden, dan is de keuze een gedeelte van het systeem dat tijdens de huidige ronde in behandeling genomen zal worden.

**Stap 3**

In dit stap is een element gekozen vanuit het systeem en stakeholders hebben de requirements een prioriteit gegeven. De software architect zal verder moeten prioriteren aan de hand van de volgende elementen: “belang van de requirement voor de stakeholders” en “impact die de requirement zal hebben op de architectuur”. Vanuit deze paar worden de hoogst geprioriteerde requirement gekozen die gebruikt zullen worden in de volgende stap.

**Stap 4**

Hier worden de grote componenten gekozen die in de architectuur terechtkomen. Aan de hand van design constraints en quality attribute requirements worden het type element, relatie tussen componenten en de interactie vastgesteld.

Daarbij horen zes substappen bij

1. Identificeer de design concerns die komen kijken bij de kandidaat pattern.
2. Voor elke design concern, maak een lijst van alternatieve patronen die de problemen op kunnen lossen.
3. Selecteer een pattern die het beste bij past vanuit de lijst van design patterns die in de vorige substap is gemaakt.
4. Beoordeel de gekozen patterns en kijk hoe ze functioneren als ze bij elkaar komen. Kijk voor overlappingen en probeer overbodige delen uit te halen. Delen die niet overlappen kunnen weg. En als laatste identificeer nieuwe elementen die uit deze combinatie van patterns komen.
5. Omschrijf de pattern die je gekozen hebt en zet het om in een architectuur zoals module, Component-and-Connector en Allocation views. Er hoeft geen volledige documentatie gemaakt worden. Documenteer alleen informatie waarvan zeker is dat die een belangrijk functie hebben binnen de architectuur.
6. Evalueer en los inconsistenties op in het designconcept.

**Stap 5**

De gekozen patterns wordt in deze stap geinstantieerd. Dat wil zeggen dat ze getekend worden in de architectuur en verantwoordelijkheden krijgen. Bijvoorbeeld bij een ping-echo pattern heeft een ping element verantwoordelijkheden zoals het versturen van een ping message en het echo element een echo verantwoordelijkheid die de message terugkaatst.

Hierbij horen zes substappen bij:

1. Instantieer één instantie van elke type element die je gekozen hebt in stap 4.
2. Ken verantwoordelijkheden toe aan deze instanties. Als we terugkijken naar de ping-echo voorbeeld, moeten we vaststellen naar waar de ping gestuurd worden, waar komt de ping vandaan en wat de ping inhoudt.
3. Neem de parent verantwoordelijkheden (oftewel functies) over in de Child elements. Zorg ervoor dat de juiste functies bij het juiste element horen.
4. Maak additionele element types aan als deze zorgen voor een reactie op de quality attributes zoals modifiability.

**Stap 6**

In stap 6 zorg je ervoor dat elke element die in de vorige stap zijn gemaakt, een attribuut of een service krijgt.

**Stap 7**

In de laatste stap verifieer je of de elementen tot nu toe voldoen aan de functionele requirements, quality attribute requirements en design constraint.

Daarnaast bereid je ook volgende child elementen voor voor verdere verwerking.

In deze stap horen drie substappen:

1. Verifieer dat alle requirements die in deze ronde in behandeling zijn geweest, door de elements die tot nu toe gemaakt zijn opgelost zijn.
2. Vertaal alle verantwoordelijkheden naar functionele requirements voor de individuele elements.
3. Zorg dat de quality attribute requirements verder refined wordt voor child elements.

**Stap 8**

Wanneer de stappen 1 t/m 7 zijn doorlopen concludeer of het systeem volledig hebt gedekt. Kijk na of elke requirement terug is gekomen en zorg dat deze is verwerkt in een element. Daarna per element afgaan of ze de juiste functies en de attributen hebben. Is het systeem af? Dan heb je een architectuur die je kunt realiseren en implementeren. Anders ga terug naar stap 2 om een nieuwe iteratie te starten.

Voordelen ADD:

1. Afwegingen vroeg in het ontwerpprocess kunnen makkelijk ontdekt en genomen worden.
2. Je maakt geen compleet architectuur. Per iteratie maak je een deel van de architectuur. Dit maakt een architectuur flexibel en is het makkelijk aan te passen aangezien je een klein deel hoeft aan te passen.
3. Zit op één lijn met de klantenbehoeften.

Nadelen ADD:

1. Vereist volledigheid van de klant requirements. Dit vergt veel inspanning vanuit de klant.
2. Inzichten kunnen technisch gezien niet haalbaar zijn

## C4

Het C4 model is een procedure voor software developers om een software architectuur te kunnen documenteren en designen. Doel van C4 is om door middel van 4 diagrammen, een uiteindelijke architectuur design te kunnen realiseren die te begrijpen is door de business en door de developers.

Binnen het process hebben we 4 levels. Elke level representeert een laag van het systeem waarbij level 1 het hoogste laag is, genaamd “System Context Diagram”. Dit volgt door level2: “Container diagram”, level 3: “Component diagram” en als laatste level 4: “Code”.



### Level 1: System Context Diagram

Level 1 is het startingspunt van C4. Hier wordt volledig uitgezoomd en modelleer je het systeem. Dit is een mogelijkheid om het volledige systeem in te zien. In het diagram zullen alleen acties beschreven worden die een gebruiker uit kan voeren op het systeem. Daarom zal er in dit level geen technische details in voorkomen. Het diagram moet dusdanig gemaakt worden dat het gelezen kan worden door iedere belanghebbenden (de technische, niet-technische en business staff). Detail is hier niet belangrijk omdat gefocussed moet worden op mensen (actors, rollen en personas) en externe softwaresystemen waar het project mee in contact zal komen.

### 1.3.2 Level 2: Container diagram

Als eenmaal begrepen wordt hoe het systeem in verbinding staat met externe systemen, kan de volgende level gestart worden. Een nuttige stap is het inzoomen tot het systeem container. Een container hier wordt gezien als een apart unit dat online gezet kan worden en dat code kan uitvoeren of data kan opslaan. Voorbeelden zijn: single-page applicatie, server-side webapplicatie, mobiele app of een database schema.   
  
Gedurende level 2 worden de high level structuur en verantwoordelijkheden van het systeem beschreven. Vervolgens worden technische beslissingen hier genomen zoals welke programmeertaal, framework en welke database van pas zullen komen.   
  
Doelgroep van dit level zijn de technische mensen binnen en buiten het softwareteam. Deze kunnen zijn software architects, developers en operation of supportmedewerkers.

### Level 3: Component diagram

Aangekomen bij level 3 zal verder in de container gezoomd worden. In level 3 zoom je verder in de container. Hier zullen de verschillende componenten worden getekend. Daarnaast worden de verbindingen aan het diagram toegevoegd en worden ze beschreven met bijbehorende interactiesoort. Elke component krijgt als volgt een kort beschrijven van zijn verantwoordelijkheden binnen het systeem en wordt de technische details per component eraan toegevoegd. Deze technische details zijn dan: een controller, een model, een view of een database model.

De doelgroep van deze level zijn de software architecten en developers.

### Level 4: Code

Als laatst zal level 4 getekend worden. Level 4 wordt gerepresenteerd door een UML-diagram, oftewel een klassendiagram. Binnen een klassendiagram wordt elke component gemodelleerd als een entiteit met de attributen en de functies daaronder.

Level 4 is een optioneel, maar een complex level binnen het C4 ontwikkelmethode. Het is veeleisend om in vroege stadium vast te stellen welke functies/code je nodig zal hebben om een gewenst resultaat te kunnen bereiken. Om deze redenen wordt een klassendiagram meerdere malen opnieuw aangepast/gemaakt.

Een mogelijkheid is tools te gebruiken die automatisch een klassendiagram maakt van de code. Doelgroep van deze level zijn de software architecten en de developers.

Voordelen C4:

1. Uiteindelijke product van C4 is een klassendiagram die gebruikt kan worden tijdens het realiseren van het product.
2. Business is makkelijker bij te betrekken door de ambiguitious language die door beide partijen

Nadelen C4:

1. C4 is lightweight. Het is niet geschikt voor projecten waar uitgebreide domeinkennis nodig is.
2. Een externe partij zal zonder introductie het niet snappen.
3. Met C4 is het lastig om producten te evalueren. Wanneer je moet aanpassen is het soms nodig om het level compleet opnieuw te maken.

# Keuze

Zoals bij de introductie aangegeven is de rede dat voor de eerdergenoemde drie ontwikkelmethodes gekozen is voor beschrijving dat het eerder behandeld is. Door de complexe aard en de korte tijd is er gekozen om niet verder te zoeken naar andere ontwikkelmethodes en daar een beschrijving bij te maken.

Het project eist een deel kennis over vliegdomein. Daarnaast is het project meer technisch complex. Aangezien DDD niet goed werkt met technisch complexe projecten is het afgeraden om DDD te gebruiken. Daarnaast wordt DDD aangeraden bij projectduur langer dan zes maanden. Het project is niet groot genoeg om gebruik te maken van zo een investering.

Vervolgens blijven alleen C4 en ADD over. Doordat je met ADD-iteratief werkt, kan dat goed passen binnen een scrum omgeving. Twijfels liggen bij overhead die aanwezig is per iteratie doordat er meerdere elementen binnen ADD gedocumenteerd moeten zullen worden voordat het een succes is. Verder wordt de wensen van de klant binnen ADD behandeld. Dit is in het huidige project niet van toepassing.

Als laatst hebben we C4. Het project is meer technisch complex project waar C4 goed mee om kan gaan. Verder is C4 lightweight wat in ons voordeel kan spelen. Vooral omdat er een deadline is voor het project van 17 weken. Als laatste is de “ubiquitous language” een handige manier om de tekeningen leesbaar te maken voor externe partijen.

Ter conclusie, gezien de context van het project en de beschikbare informatie waar over beschikking is, is C4 de juiste methodiek om gebruik van te maken tijdens het project.

## Bijlage 6: Onderzoek Serverless

Onderzoek Serverless

**Jongens van Techniek**

*Auteur : Necati Unal*

*Datum : 8 Oktober 2021*

**Inhoudsopgave**

[Inleiding 133](#_Toc85156940)

[Wat is Serverless 134](#_Toc85156941)

[Azure functions 134](#_Toc85156942)

[Functions debugging 137](#_Toc85156943)

[Voordelen 138](#_Toc85156944)

[Minder server kosten 138](#_Toc85156945)

[Scalable 138](#_Toc85156946)

[Personeelskosten 138](#_Toc85156947)

[Focussen op de eindgebruiker 138](#_Toc85156948)

[Nadelen 139](#_Toc85156949)

[Gebruik van cloud-platforms 139](#_Toc85156950)

[Learning curve 139](#_Toc85156951)

[Long-term taken & sessies 139](#_Toc85156952)

[Verschillen Serverless vs traditionele Dotnet Core MVC oplossing 139](#_Toc85156953)

[Conclusie 140](#_Toc85156954)

# Inleiding

In dit document wordt er een onderzoek gedaan naar de mogelijkheden om Azure Application Insight te gebruiken. Voor het project was eerst een oplossing gedacht binnen de .net framework met een MVC oplossing maar na de cursus EZ-900 gevolgd te hebben is er een andere oplossing naar voren gekomen die beter kan zijn voor het project.

De verschillen tussen een MVC oplossing en een serverless oplossing wordt in dit document behandeld. Er wordt onderzocht welke voordelen Serverless heeft tegenover een traditionele Dotnet core MVC applicatie tegenover een serverless toepassing.

Hiervoor wordt eerst uitgelegd wat serverless is en wat de toepassingen ervan zijn. Verder wordt uitgelegd welke voordelen deze hebben vanuit een traditionele Dotnet core MVC applicatie waarbij de verschillen tegenover elkaar gezet worden.

Uiteindelijk wordt er een testapplicatie ontwikkeld waar bij Azure Application Insight in geïmplementeerd wordt om de eindapplicatie mee te kunnen representeren.

# Wat is Serverless

Serverless computing, ook wel Serverless zorgt ervoor dat ontwikkelaars sneller kunnen bouwen door infrastructuur management te elimineren binnen het ontwikkelproces. Met Serverless applicaties is het mogelijk om automatisch server resources toe te wijzen aan de applicatie dat gedraaid zal moeten worden.

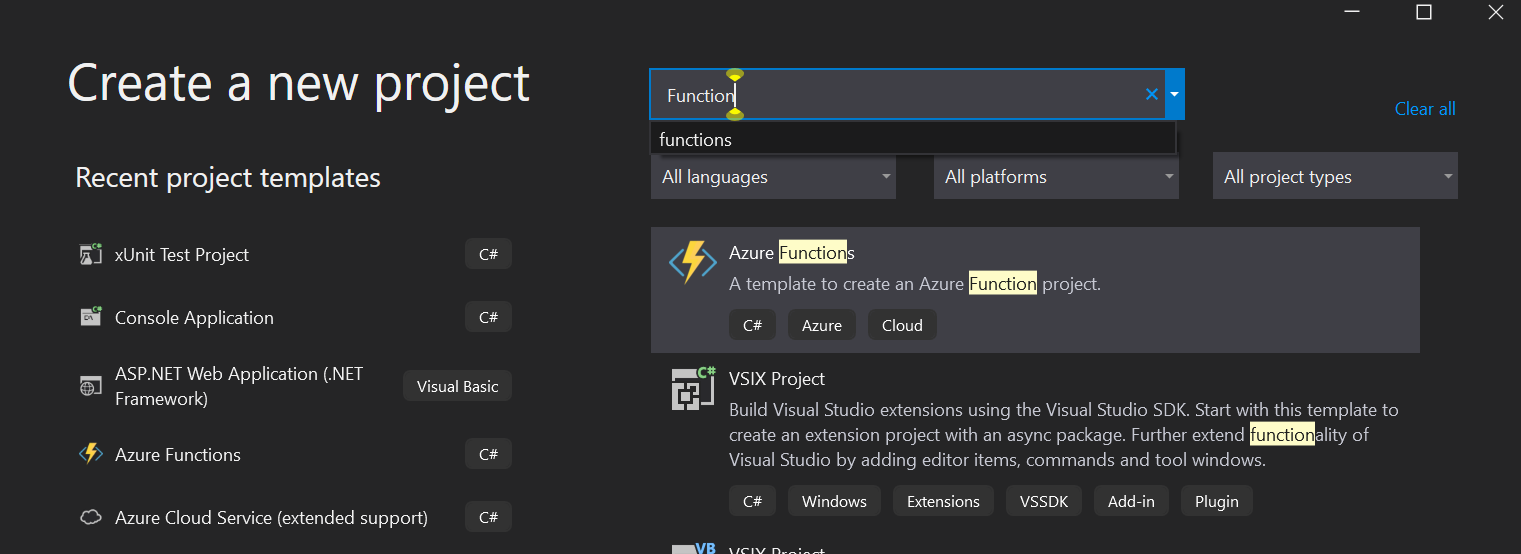
De cloud zorgt voor vele oplossingen waaronder kostenbeheer. Het gebruiken van Serverless consumption is daar een onderdeel van. Serverless zorgt ervoor dat niet continu een server draaiende is voor de applicatie. Wanneer een trigger voor de Serverless applicatie getriggerd wordt, zal de code op een server geplaatst en gedraaid worden.

Het is hier belangrijk om te snappen dat Serverless computing niet betekent dat de servers overbodig zijn. De servers en een omgeving waarop de code gedraaid zijn van belang op het moment dat de code gedraaid moet worden. De serverless naam is afkomstig van het feit dat de taken die nodig zijn voor infrastructuur en servermanagement niet meer zichtbaar zijn voor de ontwikkelaar. De ontwikkelaar hoeft alleen de code schrijven en deployen naar de cloud, waarna de code gedraaid kan worden. Dit zorgt ervoor dat de ontwikkelaar meer tijd heeft voor het ontwikkelen van het product. Het managen van hardware wordt niet meer belangrijk en zal niet meer het werkproces belemmeren.

## Azure functions

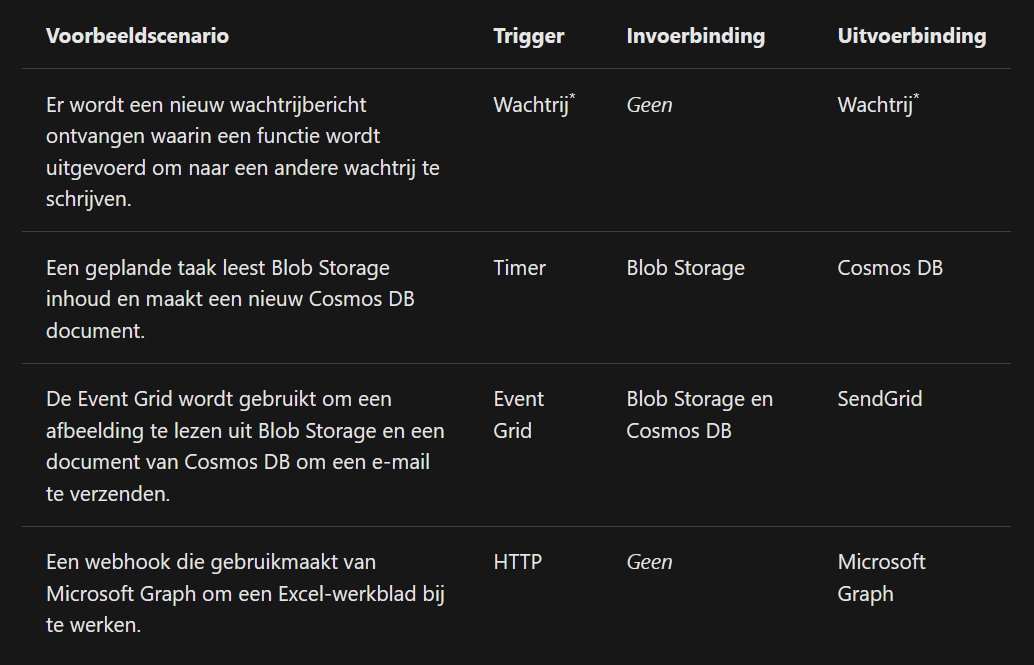
De toepassing van Serverless op Azure is de Azure functions resource. Azure geeft ons de mogelijkheid om een eigen serverless applicatie te kunnen hosten door deze aan te maken binnen een resource group.

Een Azure functions project is aan te maken via Visual studio code. Dit is mogelijk door via Visual studio een nieuw project aan te maken. Op het “Create a new project” scherm van Azure kunnen templates gekozen worden. Hier kan gezocht worden op Function app (figuur 1).



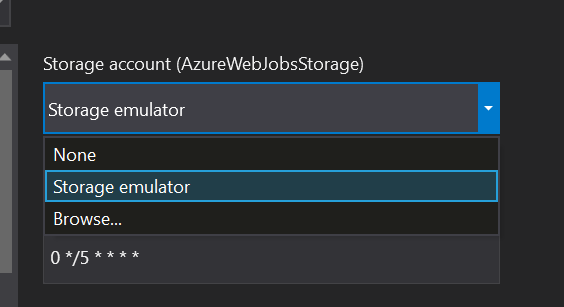
Figuur Nieuwe functions project wizard

Na een naam gegeven te hebben is er een lijst aan triggers te vinden. Volgens Microsoft zijn er een aantal triggers waar uit gekozen kunnen worden (Microsoft, 2019). In figuur twee zijn er een aantal triggers te vinden waar uit gekozen kunnen worden. De simpelste is een timertrigger of een httptrigger. Uit de interviews kwam naar boven dat de tests om de x aantal minuten uitgevoerd moeten worden. Hierdoor is besloten om voor een timertrigger te gaan.



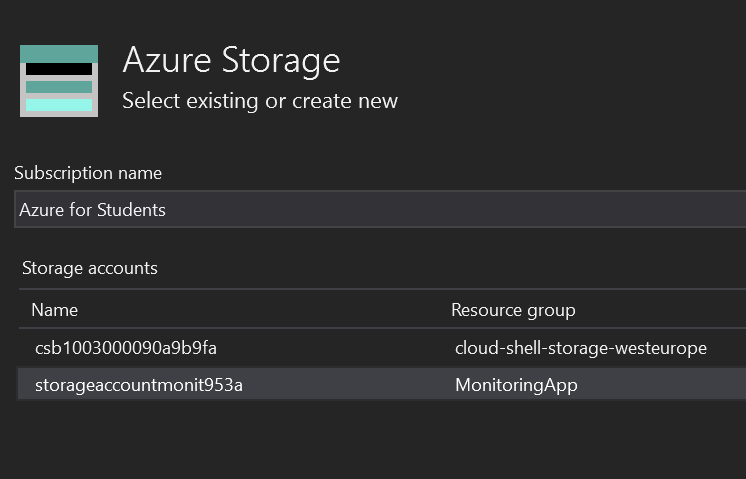
Figuur Triggers

Bij het selecteren van de timer trigger wordt een storage account gekoppeld worden aan de Azure Functions resource. In de storage account wordt de timer opgeslagen van de Function zodat de applicatie weer wanneer het getriggerd zal moeten worden. Deze zal in dezelfde resourcegroup van de azure functions moeten zitten. Rechts van de triggers zijn een aantal opties geselecteerd waaronder “None” en “Emulator”. None geeft aan geen storage te verbinden aan de Azure function en de Emulator creëert een mock storage account aan (figuur 3).



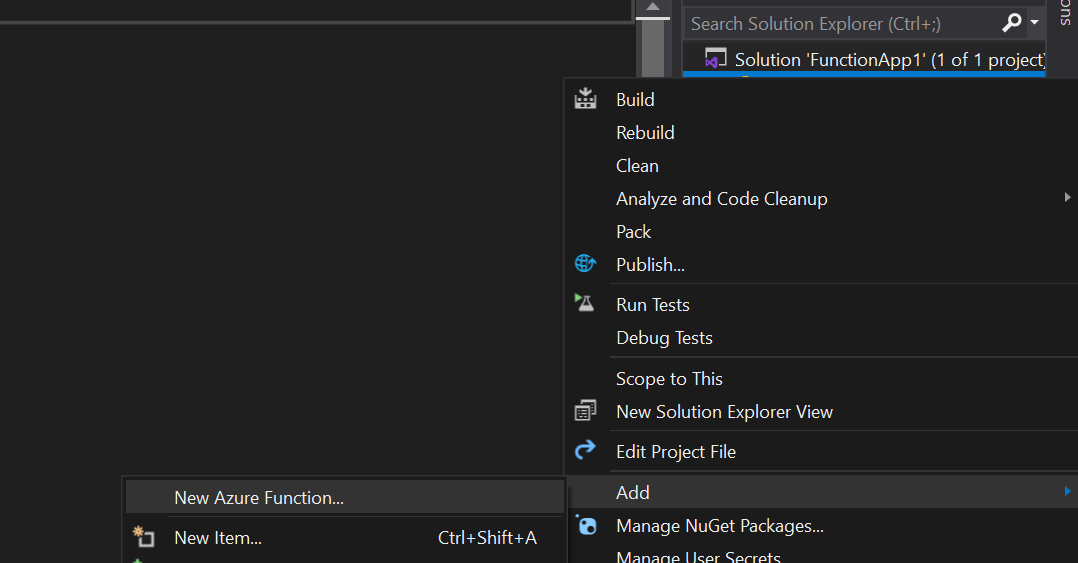
Figuur Dropdown menu storage emulator

Om direct alles goed werkend te krijgen is ervoor gekozen om een storage account aan te maken. Door Browse te klikken, wordt een overzicht vertoond van alle aangemaakte storage accounts. Hier wordt gekozen voor de storage account gemaakt binnen de resourcegroup voor de monitoringApp (figuur 4).



Figuur Kiezen storage account wizard

Hierna kan een schedule gekozen worden. Het schema is een Cronjob genoteerde (C-sharpCorner, 2019) notatie die meegegeven kan worden aan de timer (figuur 3). Deze timer wordt gebruikt om de functie te triggeren.  
  
Er is nu een timer trigger Function app gecreëerd door Visual studio die gebruikt kan worden voor het project. Verder is er mogelijkheden om meerdere functions aan te maken. Per function kunnen verschillende triggers aan gekoppeld worden zodat er functions zijn met verschillende triggers mocht dat nodig zijn (figuur 5).



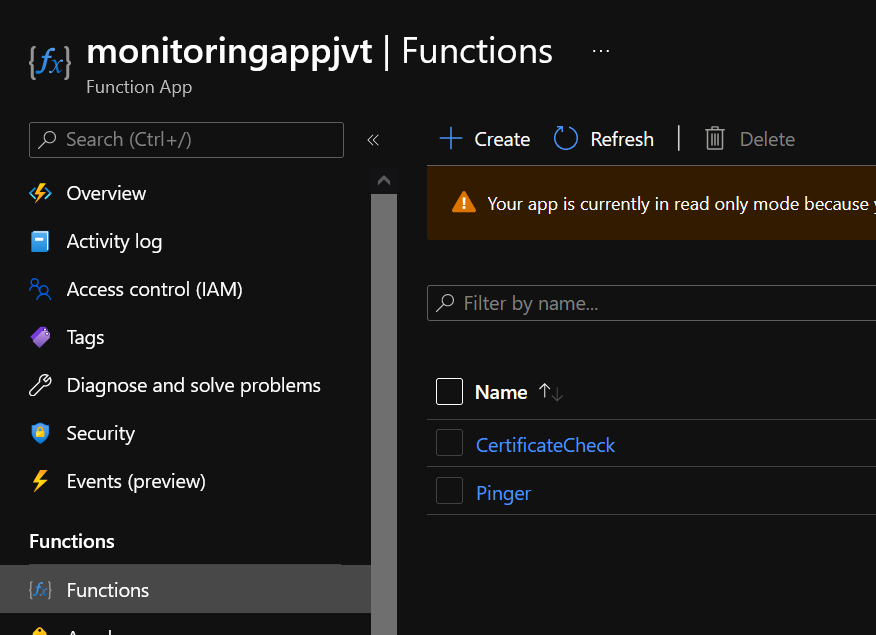
Figuur Toevoegen nieuwe Functions

## Functions debugging

Doordat Functions cloud gebonden is, is het lastiger om code te debuggen en zal er verschil zijn in omgevingen waarin de applicatie gerund zal worden. Lokaal zal de Functions anders gedragen vergeleken met wanneer de applicatie in de cloud gerund wordt. Dit komt doordat lokaal een emulator draait van Functions die niet alle functionaliteiten van Serverless kan repliceren.

Hiervoor is er een mogelijkheid om de applicatie handmatig te runnen in de cloud.

Binnen de Functions resource is een overzicht te vinden van alle gepublished en beschikbare Functions (figuur 6). Hier zijn er een aantal developer opties beschikbaar waaronder “Code + Test”.



Figuur Azure functions resource overzicht

Dit is een Simpele UI waar de Function handmatig op de server gerund kan worden (figuur 7). Dit geeft ons de mogelijkheid om de code op Azure portal te testen op fouten.

Afbeelding met tekst, monitor, schermafbeelding, scherm

Automatisch gegenereerde beschrijving

Figuur Code + Test omgeving

## Voordelen

Het gebruik van functions neemt grote voordelen mee. Voordelen bestaan uit het vermijden van servermanagement en infrastructuur aan automatisering zodat daar niet gedacht hoeft te worden. Wanneer dieper gekeken wordt naar de voordelen zijn er nog een aantal die uitsteken. Deze zijn hieronder te vinden:

1. Minder server kosten
2. Scalable
3. Geen tijd kwijt aan personeelskosten
4. Het kunnen focussen op iets tastbaars voor de eindgebruiker i.p.v. infrastructuur.

### Minder server kosten

Door gebruik te maken van serverless consumption wordt er geen specifiek server voor het project gereserveerd. Dit betekend dat er niet betaald hoeft te worden voor tijd wanneer de applicatie niet gebruikt wordt. Wanneer de applicatie uit staat en dus niet getriggered is, wordt er ook geen server voor toegewezen waardoor uiteindelijk de kosten minder zijn.

### Scalable

Het is makkelijk om te kunnen schalen en meer resources toe te wijzen aan de applicatie. Wanneer een trigger op een hogere frequentie getriggerd wordt, kan er meer serverruimte aan toegewezen worden om het draaiende te houden.

### Personeelskosten

Naar personeelskosten kan vanuit meerdere perspectieven gekeken worden. De belangrijkste is een onderscheid tussen on-premise applicaties en cloud applicaties.

Wanneer on-premise applicaties overstappen naar function apps, is er geen behoefte aan onderhoud van de servers. Deze verantwoordelijkheid kan verplaatst worden naar de cloud waardoor onderhoudsmedewerkers obsolete worden. Verder wordt winst geboekt op het hardware gebied, maar deze kosten zullen zich verplaatsen naar de cloud toe om de kosten daar te kunnen dekken

De cloud-applicaties die gebruik zullen maken van Function apps zullen voordeel boeken op het gebied van implementaties. De inftrastructuur vereist geen onderhoud meer en kan gelaten worden aan de cloud.

### Focussen op de eindgebruiker

Deze voordeel is indirect gekoppeld aan 2.2.3 personeelskosten. Mankracht blijft over omdat ze zich niet meer focussen op infrastructuur. Hierdoor is er mogelijkheid om deze focus te verschuiven. Mankracht dat overblijft kan zich focussen op de klant. Meer requirements kunnen geïmplementeerd worden waardoor de klant iets tastbaars krijgt. Dit betekend meer deliverables en minder tijd nodig voor het gehele project.

## Nadelen

Al in al klinkt het alsof Function app perfectie is. Verminderde kosten, meer focus op de klant en het project is sneller compleet. Desondanks is er een grote nadeel bij het gebruik van de Functions.

1. Gebruik van cloud-platforms
2. Learning curve
3. Niet toepasselijk voor long-term taken & sessies

### Gebruik van cloud-platforms

Gebruik van Functions betekend automatisering. Azure heeft zijn Function Apps en Amazon heeft Lambda. Helaas zijn dit platform afhankelijke oplossingen en is het overzetten van een function van één naar de andere platform lastig te doen. Dit betekend dat men zich moet houden aan de regels van de platform waarop ze zitten. Dit resulteert in een vendor lock.

### Learning curve

Desondanks de vele documentatie en de community resources, zal men zich snel realiseren dat Functions vrij nieuw is en daardoor niet altijd goed gedocumenteerd of documentatie voor kan vinden. Om bestaande applicaties te kunnen niet altijd omgezet worden door de use-case van de applicaties. Hierover kan in 2.3.3 meer gelezen worden.

### Long-term taken & sessies

Zoals in het begin van dit hoofdstuk aangegeven wordt, klinkt het te goed om waar te zijn en dat is het ook. De enige punt waarom men overstap naar Functions is de use-case van de applicatie. Het is belangrijk te begrijpen dat de applicatie op de server vernietigd wordt op het moment dat de code uitvoering af is. Wanneer de code uitgevoerd is, zal de server zich zichzelf beschikbaar te stellen voor een andere Functions. Dit betekend dat er geen sessions bijgehouden worden. Verder is het niet mogelijk om een log-in te implementeren door de hele applicatie doordat de ingelogde persoon niet opgeslagen worden in sessies. Per trigger zal de persoon moeten inloggen wat lijdt tot belemmering van de user experience.

Functions is meer geschikt voor event-based applicaties. Voorbeelden hiervan zijn Internet of Things (IoT) applicaties, chatbots, of “korte-levensduur” applicaties. Dit zijn applicaties die een duidelijke begin en een eind hebben. Een voorbeeld van een lange-termijn applicatie is een webapplicatie. De gebruiker komt op een website terecht en zal daarop blijven voor een onbepaalde tijd. Het is voor de programmeur niet bekend wanneer de gebruiker “klaar” is met de website. Een voorbeeld van een “korte-levensduur” applicatie gebaseerd op events. Voorbeeld hiervan zijn applicaties die een input nodig hebben om daarna een output te kunnen produceren. Er is een duidelijke beginpunt en een eindpunt van de applicatie waarna de applicatie zich in een “slaapstand” zal bevinden.

## Verschillen Serverless vs traditionele Dotnet Core MVC oplossing

De belangrijkste verschil tussen traditionele applicaties en Serverless applicaties is de levensduur van de applicatie dat wordt gedraaid. Zoals in 2.3.3 aangegeven wordt is het belangrijk dat de functie van de applicatie duidelijk is. Er is een event nodig voor de Function App om getriggerd te worden. Na de trigger krijgt de functions app ruimte op de server om uitgevoerd te worden. Hier tegenover staat een traditionele Dotnet Core applicatie dat een vooraf gedefinieerde ruimte heeft op de on-premise server of de cloud-server waarop het continu gedraaid kan worden. Het zal geen downtime hebben en gedurende het draaien kan de sessie bijgehouden worden van de gebruiker.

# Conclusie

Gebruik van Functions levert veel voordelen op. Het past binnen de use-case waarvoor een Function app gebruikt zal moeten worden. De tests die uitgevoerd zullen worden binnen de Monitoringapp zijn events die als triggers kunnen dienen. Dit geeft ons een goede use-case voor de function app.

Verder is gebruik van Functions een voordeel bij kostenmanagement. Door gebruik van de downtime tussen triggers door hoeft niet altijd een server standby te staan om de code uit te kunnen voeren. Er zal alleen betaald moeten worden op het moment er gebruik gemaakt wordt van de servers om code uit te voeren.

Desondanks moeten de nadelen niet door de vingers glippen. Het grootste probleem voor de monitoringsapp zijn de sessies die verdwijnen na elke uitvoering van de Function App. Hierdoor kan er niet gewerkt worden met sessies binnen de applicatie waardoor scheiding van rollen binnen de applicatie niet mogelijk is.

Alhoewel de nadelen impact hebben op de applicatie, zijn dit geen brekende nadelen. De voordelen van de FunctionsApp wegen zwaarder dan de nadelen waardoor besloten is om de Function App te gebruiken voor de MonitoringsApp.

## Bijlage 7: Onderzoek Azure Application Insight

Onderzoek Azure Monitor & Application Insight

Auteur: Necati Unal

Datum: 26-05-2021

**Inhoudsopgave**

[Inleiding 148](#_Toc85157138)

[Wat is Azure Application Insight 149](#_Toc85157139)

[Wat kan Azure Application insight tracken? 149](#_Toc85157140)

[Wat zijn metrics? 149](#_Toc85157141)

[Custom Metrics 151](#_Toc85157142)

[Wat is Azure monitoring 152](#_Toc85157143)

[Hoe kan Azure Application Insight & Monitoring gebruikt worden binnen het project 152](#_Toc85157144)

[Opzetten Application Insight 153](#_Toc85157145)

[Opzetten test project 154](#_Toc85157146)

[Dashboard Application Insights 157](#_Toc85157147)

[Conclusie 160](#_Toc85157148)

[Bibliografie 161](#_Toc85157149)

# Inleiding

Dit document is een vooronderzoek naar Azure Application insight en Azure monitor. Deze onderzoek is tot stand gekomen doordat het potentie heeft om binnen het project gebruikt te worden. De component is nog onbekend en zal onderzocht worden of het toegevoegde waarde heeft. De hoofdvraag hier is “Wat zijn de Voor en Nadelen van Azure Application Insight en Azure Monitoring voor het project?”

In het onderzoek wordt eerst onderzocht wat Azure Application insight is. Hier wordt bekeken naar de verschillende functionaliteiten die Application Insight te bieden heeft.

Daarna wordt dieper ingegaan op de metrics die Application Insight bijhoudt. Deze metrics zijn prestatie waardes die per applicatie bijgehouden worden. Naast de standaard metrics biedt Application insight meer aanpassingen. De custom metrics worden uitgelegd en wordt besproken waarvoor deze gebruikt kunnen worden.

Vervolgens wordt er een testapplicatie gemaakt om al deze elementen van Application Insight te kunnen testen. Door een Application Insight resource aan te maken, te koppelen aan een test project wordt een start gemaakt. Vervolgens wordt er een test metric verstuurd naar Application Insight die terug te vinden is binnen de logs.

# Wat is Azure Application Insight

Het monitoring van je applicatie is belangrijk om inzicht te krijgen in de gebruikers en de prestatie van een applicatie. Op deze manier kan je inschatten wanneer je extra resources nodig zal hebben voor je applicatie indien het verkeer te veel wordt. Je kunt aan de hand van deze piekmomenten beslissen om extra resources op een bepaald tijdstip beschikbaar te maken. Dit voorkomt vertraging binnen de applicatie en zal de gebruikers ervan niet merken dat er een piekmoment is. Piekmomenten kunnen ervoor zorgen dat de applicatie uit kan vallen.

Application insight is en SaaS oplossing van Azure om prestatiemanagement te doen (Pragmatic Works, 2018). Het is ontwikkeld om te zorgen dat je zo optimaal mogelijk prestatie kan leveren aan je gebruikers door onder andere meer resources toe te wijzen aan de applicatie.

Application Insight kan gebruikt worden door meerdere platformen zoals een webapplicatie of een console applicatie. De werking ervan is vrij simpel. Door een package te installeren in de applicatie is Azure Application insight beschikbaar voor gebruik. Hij zal dan de benodigde gegevens naar de Azure Application sturen. Application Insight heeft dan de mogelijkheid om standaard metrics te tracken van de gekoppelde applicatie. Deze gegevens zijn gekoppeld aan de soort applicatie die eraan verbonden is.

# Wat kan Azure Application insight tracken?

Application Insights heeft de mogelijkheid om een breed scala aan informatie bij te houden over de gemonitorde applicatie. Standaard gegevens worden geaggregeerd tijdens het verzamelen. Dit betekend dat ze verzameld worden niet per tijd maar over een tijdsbestek . Hoelang dit is, is afhankelijk van de metric dat wordt verzameld.

## Wat zijn metrics?

Metrics zijn waardes die gedurende de applicatie bijgehouden worden. Deze waardes kunnen van alles zijn. Binnen Application Insight is er een standaard overzicht aan metrics dat bijgehouden kan worden (Microsoft, 2021). Informatie zoals CPU gebruik en geheugen. Desondanks kan soms dit niet genoeg zijn.

Hieronder is een tabel te vinden van de standaard metrics die Application Insight van te voren standaard verzameld.

|  |  |
| --- | --- |
| Naam Metric | Betekenis |
| Beschikbaarheid | Toont de beschikbaarheidspercentage van de webtests dat geen fouten heeft vertoond. |
| Beschikbaarheid test lengte | Toont de aantal tijd dat heeft gekost om te test te kunnen runnen. |
| Beschikbaarheid test | Toont de aantal keer dat de tests zijn gerund. |
| Website laadtijden | Toont de website laadtijden. |
| Client processing time | Toont de browser laadtijden. |
| Page load network connection time | Toont de tijd die de netwerk nodig heeft om de pagina terug te sturen. |
| Receiving response time | Toont de tijd dat nodig is om terug te reageren op een request. |
| Browser Exceptions | Toont de excepties dat de website heeft vertoond. |
| Dependency call failures | Toont de aantal dependancy call failures. |
| Exceptions | Toont de aantal foutmeldingen. |
| Failed requests | Toont de aantal foute requests naar de website toe. |
| Server Exceptions | Toont de aantal server exceptions |
| Available memory | Toont de beschikbare geheugen van de server |
| Exception rate | Toont de frequentie van exceptions |
| HTTP request execution time | Toont de execution tijd voor een http request |
| HTTP request rate | Toont de frequentie voor een request rate dat binnenkomt |
| HTTP request in application queue | Toont de lengte van de queue aan requests |
| Proces CPU | Toont hoeveel CPU capaciteit wordt gebruikt |
| Proces IO Rate | Toont de lees/schrijf snelheid |
| Proces Private Bytes | Toont de non-shared memory dat de proces gebruikt voor zijn data (geheugen dat alleen door eigen applicatie wordt gebruikt en niet door anderen). |
| Processor time | Toont CPU consumptie in tijd. |
| Dependancy calls | Toont aantal dependancy calls |
| Dependancy duration | Toont lengte van de Dependancy |
| Server request rate | Toont frequentie van inkomende requests bij de server |
| Server requests | Toont aantal inkomende requests |
| Server response time | Toont tijd die de server nodig heeft om te reageren op een request. |
| Page view load time | Toont pagina laadtijd |
| Page views | Toont aantal pagina views |
| Traces | Toont aantal trace logs |

## Custom Metrics

Er is een heel waslijst aan metrics die gebruikt kan worden. Desondanks kan er nog meer met Application Insights. Binnen de applicatie is er mogelijkheid om custom metrics te volgen (Microsoft, 2021). Dit zijn door de gebruiker geregistreerde metrics die bijgehouden kunnen worden door Application Insight. Als voorbeeld kan aantal verkochte producten bijgehouden worden of aantal keer dat een knop gedrukt wordt. Aan de hand van deze metrics kan de website beter in de gate gehouden worden zodat men bijvoorbeeld de user experience kan verbeteren.

Binnen custom metrics zijn er meer mogelijk. Er is een lijst aan functies opgesomd binnen de documentatie van Application insight waarmee custom metrics mee verstuurd kan worden (Microsoft, 2020).

# Wat is Azure monitoring

Microsoft Azure Monitor is een hulpmiddel voor het verzamelen en analyseren van gegevens die zijn verzameld uit Azure-omgevingen en on-premise systemen. Met Azure Monitor is een uniforme oplossing voor het opslaan en analyseren van telemetry van een applicatielandschap.

Azure Monitor is een schaalbaar en geoptimaliseerd voor prestaties en kosten. Het bied een geavanceerde analytische mogelijkheden voor de gebruiker met een interactieve query taal waar de logs uit gehaald kan worden. Azure monitor kan worden geïntegreerd met meerdere programma’s in meerdere talen en is dus niet gelimiteerd tot C# of tot een Azure software programma.

Azure monitor is ontwikkeld om telemetry data te kunnen visualiseren aan de hand van de meerdere tools en kant en klare grafieken. Het helpt met identificeren van problemen die de prestaties van een applicatie kan beïnvloeden (DataShieldprotect, 2021).

Sommige use-cases van Azure Monitor zijn:

1. Detectie van problemen over de gehele applicatielandschap.
2. Monitoren van het infrastructuur
3. Analyseren van custom data die door de gebruiker gecreëerd kan worden.
4. Ondersteunt Alerts
5. Genereren van visuele elementen binnen de Monitoring voor gemakkelijk detectie van problemen door de gebruiker.

# Hoe kan Azure Application Insight & Monitoring gebruikt worden binnen het project

De verschillende mogelijkheden binnen Azure Application Insight is groot. Er is een groot scala aan metrics die standaard opgeslagen kan worden met optie om custom metrics erbij toe te voegen.

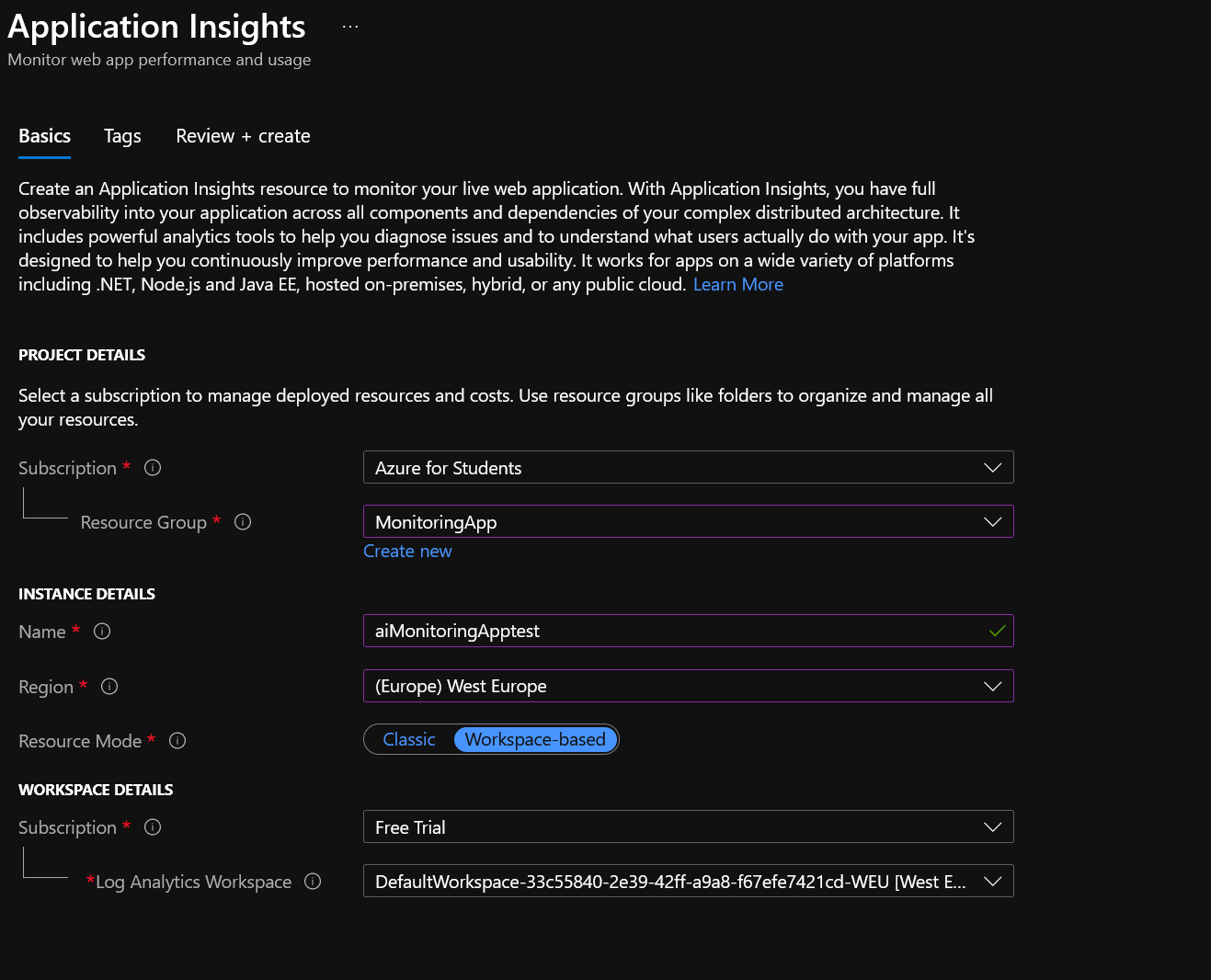
Verder is er gekeken naar implementatiemogelijkheden voor Azure Application Insight. Door een package te installeren binnen de code is het mogelijk om metrics te sturen naar Application insight en deze op te slaan.

Om dit te kunnen testen is de Application Insight package toegevoegd aan een test project. Aan de hand van een tutorial is er een testopstelling opgezet om een test te versturen naar Application Insight. Hiervoor is een tutorial gevolgd om de resource aan te maken (C-shartcorner, 2020).

## Opzetten Application Insight

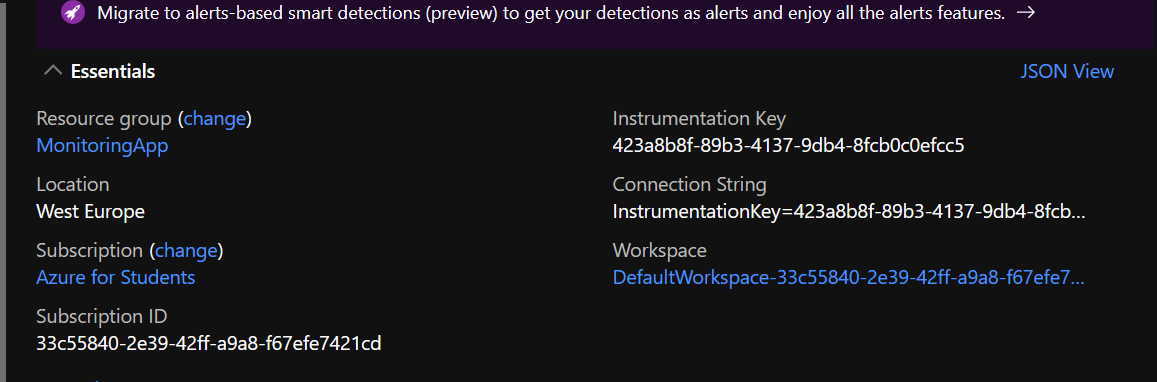
Om dit voor elkaar te krijgen zal er een omgeving eerst opgezet worden. Door een subscription van Jongens van Techniek te gebruiken heb ik toegang verkregen tot een testomgeving waar dit uitgevoerd kan worden.   
  
Eerst zal er een Azure Application insight aangemaakt moeten worden via de Azure Portal. Wanneer de knop Create gedrukt wordt, wordt een klein formulier getoond. Dit is volgens figuur 1 ingevuld.

Na preview & create te drukken wordt de resource aangemaakt in de resource group die al eerder is aangemaakt. Na nogmaals op create te drukken wordt de resource aangemaakt en wordt de dashboard van de resource vertoond.

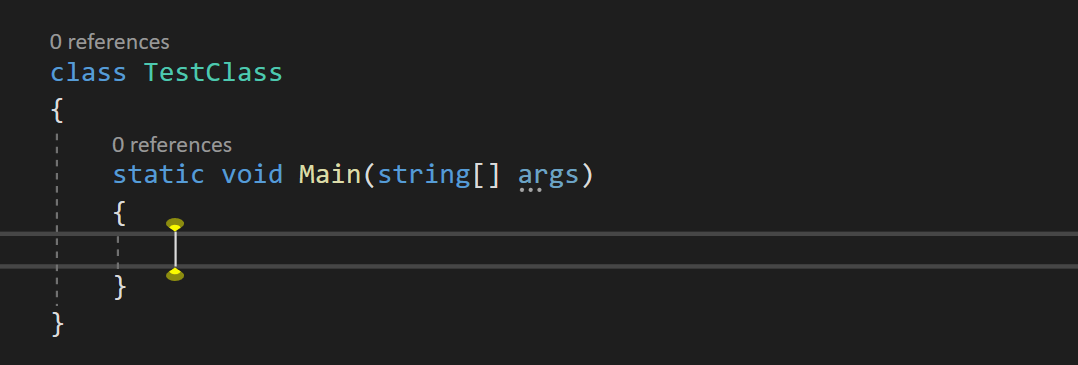


Figuur Aanmaken Resource Application Insight

De koppeling tussen de Application Insight gebeurt op basis van de Instrument\_key dat is angemaak tijdens het aanmaken van de resource. Deze instrument\_key is weer terug te vinden op de dashboard van Application Insight.



Figuur Gegevens Application Insight

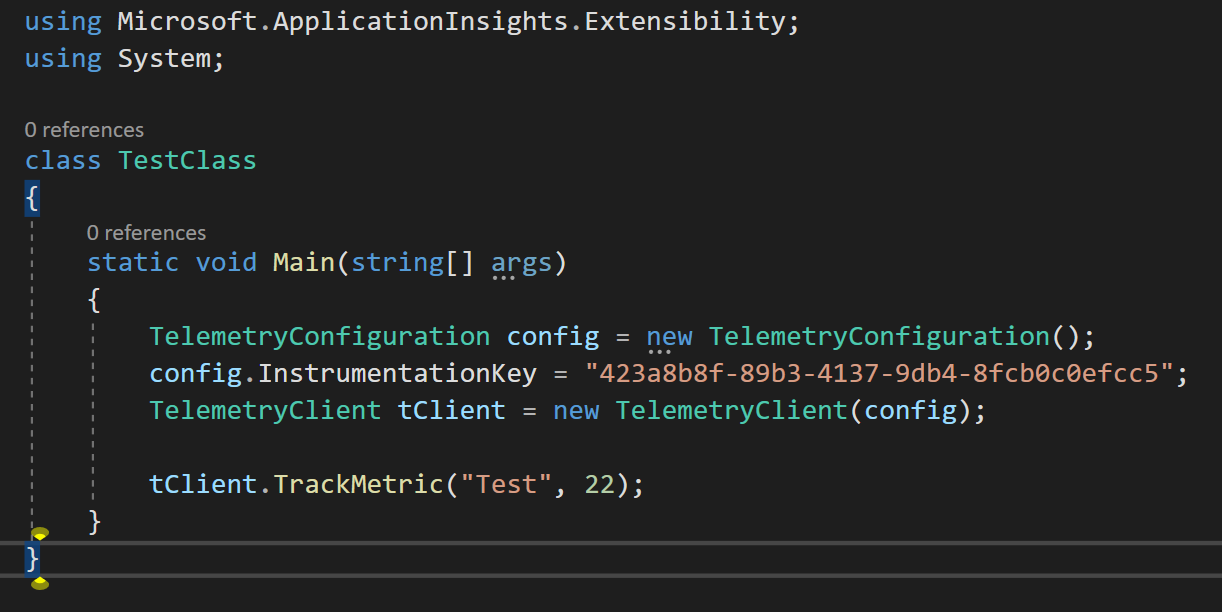


Figuur .net core console applicatie

### Opzetten test project

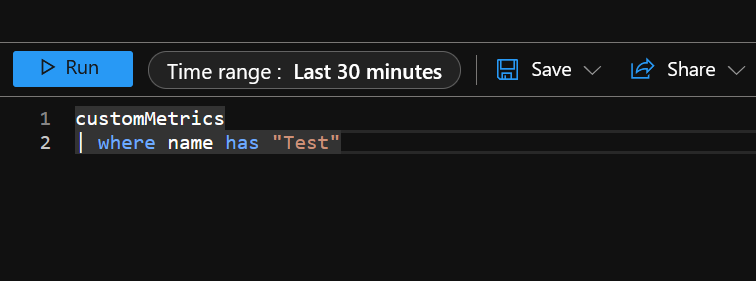
Nu zal een testapplicatie ontwikkeld moeten worden waaruit metrics verstuurd wordt. Hiervoor is een andere tutorial voor gevolgd om ervoor te zorgen dat de juiste applicatie aangemaakt kan worden. Binnen deze voorbeeld wordt een .net MVC applicatie aangemaakt die gebruikt wordt als test (Adam Marczak, 2020). Om het simpel te houden is er besloten om een asp.net core console applicatie te houden (figuur 3).

Eerst moest de object voor de telemetryConfiguration aangemaakt worden. Hierna wordt de connectie gelegd tussen de Application Insight en de applicatie door de instrument key door te geven aan het object en vervolgens het object aan te maken voor de telemetry. Met deze object is het mogelijk om de trackmetric methode aan te roepen. Aan deze functie moet de tracking gegevens meegegeven worden. Als test wordt “Test” als naam gebruikt en 22 als waarde zoals in de figuur hieronder te zien is.



Figuur Code voor test metric sturen naar Application Insight

Als we terug gaan naar Azure portal is het mogelijk om binnen de logs te kijken naar entries. De query gedeelte is hiervoor handig.



Figuur Query logs

Door binnen de query te zoeken naar de naam “Test” is het mogelijk om alle andere ruis weg te sorteren en alleen de zelfgemaakte custom metric op te zoeken. In figuur 6 is het resultaat terug te vinden van de stuk code die is geschreven. Hieruit kan geconcludeerd worden dat er custom metrics bij de logs toegevoegd kan worden aan de hand van de trackmetric methode.

Afbeelding met tekst, monitor, zwart, scherm

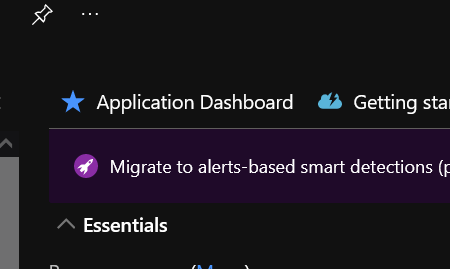
Automatisch gegenereerde beschrijving

Figuur Logs test metric

### Dashboard Application Insights

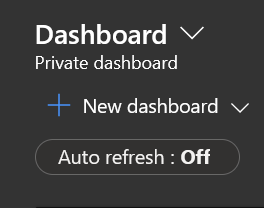
Nu alle data opgeslagen kunnen worden, kan deze data gebruikt worden voor verschillende doeleinden. Binnen Application Insight zijn er mogelijkheden om data te kunnen visualiseren. Dit wordt ookwel Azure monitor genoemd waar alle binnenkomende data live afgebeeld kunnen worden. Hiervoor kan de Application insights Dashboard gebruikt worden.

Aangekomen bij de Application Insights resource is de Application Dashboard link beschikbaar om naar de dashboard te gaan (Figuur 7).



Figuur Application Insight Dashboard

Dashboard is een oppervlakte waar allerlei op gezet kan worden, standaard bevat het een aantal grafieken van de server. Deze grafieken zijn niet van belangrijk dus kan deze verwijderd worden of een nieuwe dashboard kan aangemaakt worden.



Figuur Nieuw dashboard aanmaken

Hier kunnen meerdere grafieken aan toegevoegd worden mocht dat nodig zijn. Een grafiek kan toegevoegd worden na rechtermuisknop > Edit (figuur 9).



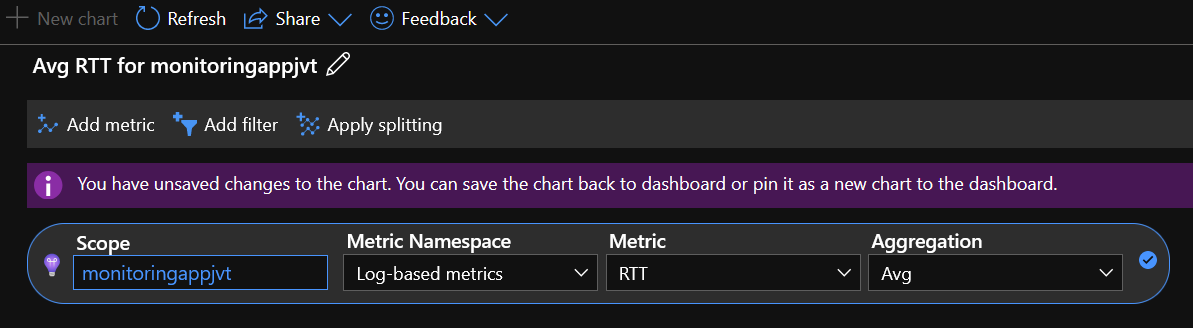
Figuur Edit Dashboard

Na het editen wordt een menu aan de rechterkant geopend waar allerlei componenten vertoond worden. De eerste is een metric chart. Door deze naar de dashboard te slepen kan gebruik van gemaakt worden. Klik nu op save. Er is nu een lege grafiek beschikbaar. Aan deze grafiek moet eerst gegevens toegekend worden voordat de visualisatie kan starten. Klik op de Edit in Metrics knop op de grafiek om de gegevens toe te voegen.

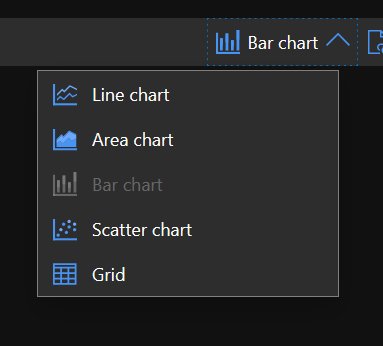


Figuur Edit grafiek

Links boven de grafiek is de Add metric knop. Door deze te klikken kan er metrics toegevoegd worden aan de grafiek om gevisualiseerd te worden. De Scope van de grafiek is de locatie waar de metric heen verstuurd wordt, oftewel de Application Insight. Vervolgens wordt gekozen voor Log based metrics om alle logs te kunnen inzien in plaats van applicatie gegevens. Onder de metric wordt gekozen voor RTT. RTT is de naam van de metric die gegeven is aan data via de Pinger functie. Hiervan wordt alleen de gemiddelde geaggregeerd (.

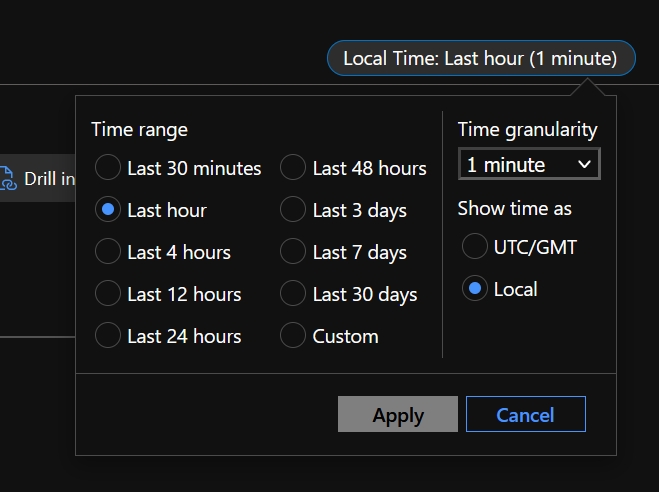


Figuur gegevens toevoegen grafiek

Na gegevens toegevoegd te hebben zijn er ook visualisatiemogelijkheden binnen Application Insight. Er kunnen verschillende grafieken gebruikt worden voor visualisaties. Dit hangt af van persoonlijke voorkeur of welke grafiek het beste de waardes kan representeren. In dit geval is het beste als een Bar grafiek gebruikt wordt voor visualisatie.

Figuur Soorten grafiek

Application Insight gaat nog een stap verder. Aan de grafiek kan ook toegevoegd worden op welke schaal van tijd de grafiek vertoond zal moeten worden. “Granularity” zoals ze het noemen, kan aangepast worden zodat de X as, oftewel de tijd, veranderd kan worden (figuur 13).



Figuur Tijd granulatie

# Conclusie

Application insight heeft veel opties die gebruikt kunnen worden binnen het project. Het heeft zelfs al een kant en klare oplossing voor de requirement #1: “De supportmedewerker kan de uptime van de gemonitorde applicatie bekijken”.

Door Application Insight toe te voegen aan het project is er een mogelijkheid om meerdere metrics te volgen voor meerdere applicaties. Door pagina’s eraan toe te voegen en custom metrics kan ervoor gezorgd worden dat de applicatie meerdere endpoints kan monitoren.

Desondanks zijn er veel overhead dat bij komen kijken. Niet alle metrics zijn nodig voor het project. Hiervoor zal een overweging gemaakt worden samen met de bedrijfsmentor of dit de overhead een probleem zal vormen vooral omdat Logs bijhouden geldt kost binnen Application Insight. Achteraf is gebleken dat af en toe logs sturen geen probleem zal zijn qua kosten. Vooral omdat het project op dit moment op een free trial abonnement gedraaid kan worden. Desondanks is het ook mogelijk om alerts aan te maken die aan of uitgezet kunnen worden. Hiermee kan kosten gecontroleerd worden door alles uit te zetten gedurende ontwikkeling van het project.

# Bibliografie

Adam Marczak. (2020, 02 04). *Youtube*. Opgehaald van Azure application Insights tutorial: https://www.youtube.com/watch?v=A0jAeGf2zUQ

C-shartcorner. (2020, 07 28). *Azure Application Insight resources*. Opgehaald van Azure Application insight: https://www.c-sharpcorner.com/article/azure-application-insights-resources-classic-and-workspace-based/

DataShieldprotect. (2021, 05 20). *Whatis Azure Monitor*. Opgehaald van Datashieldprotect: https://www.datashieldprotect.com/blog/microsoft-azure-monitor

Microsoft. (2020, 05 11). *Application Insights API for custom events and metrics*. Opgehaald van docs.microsoft: Application Insights API for custom events and metrics

## Bijlage 8: Agile vs Waterval

Scrum vs Waterval

Auteur: Necati Unal

Datum: 26-05-2021

**Inhoudsopgave**

[Inleiding 159](#_Toc85157345)

[Agile 160](#_Toc85157346)

[Waterval 161](#_Toc85157347)

[Conclusie Agile vs Waterval 162](#_Toc85157348)

[Agile Frameworks 163](#_Toc85157349)

[Scrum 163](#_Toc85157350)

[Backlog refinement 164](#_Toc85157351)

[Sprint planning 164](#_Toc85157352)

[Sprint review 165](#_Toc85157353)

[Retrospective 165](#_Toc85157354)

[Verschillen Agile vs. Waterval 165](#_Toc85157355)

[Toepassing Scrum 166](#_Toc85157356)

# Inleiding

In dit document zal dieper gekeken worden naar de verschillen tussen Waterval en Agile Projectmanagementmethode.

Eerst zal het document ingeleid door de geschiedenis van het ontstaan van Agile methode. Hier zal besproken worden hoe de Agile methode is ontstaan.   
  
Vervolgens worden de verschillen genoemd tussen de Agile en de Waterval methodiek waarna de verschillen worden vergeleken.  
  
Als laatst worden de belangrijkste details van Scrum uitgelegd en ten slotte wordt het afgesloten met het aangepaste vorm van Scrum dat binnen het project wordt gebruikt.

# Agile

Het eerste Scrum project was uitgevoerd in 1993. Een aantal jaar daarna is de Agile manifesto ontwikkeld met meerdere pioneers uit het veld die met Scrum gewerkt hebben waarna het Agile manifesto is ontwikkeld (Jim Highsmith, 2001). Dit vormt tegenwoordig het fundament voor meerdere Agile frameworks die gebruikt worden zoals Kanban, Scrum, Extreme Programming en meerdere andere frameworks.

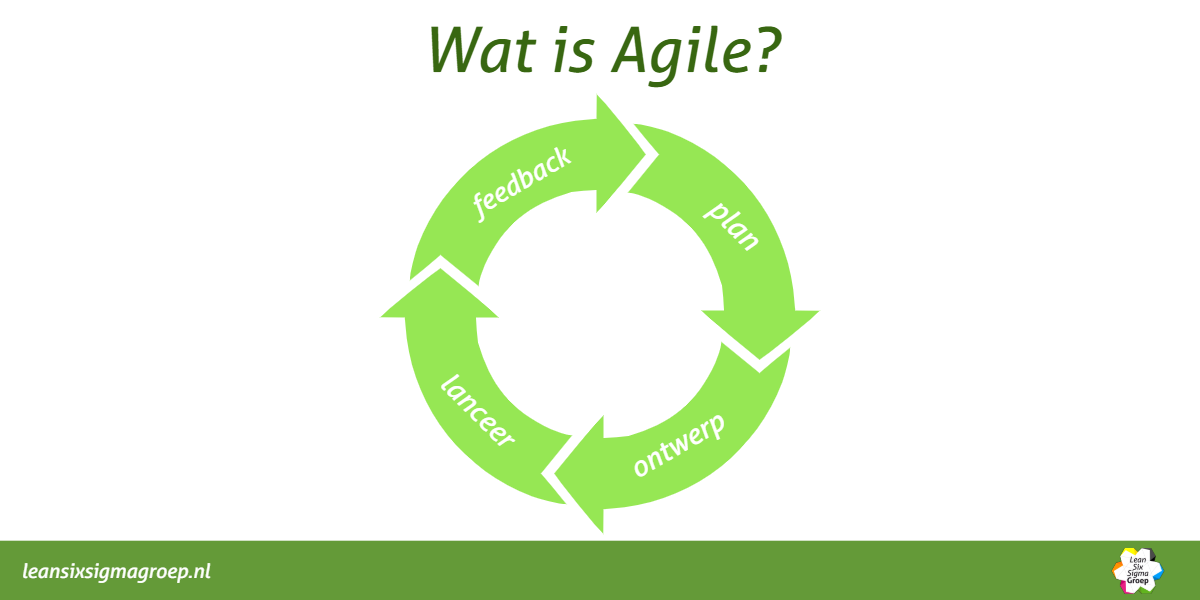
Van origine was het een standaard om eerst een probleem te identificeren, een product ontwikkelen die deze probleem zal oplossen, het product bouwen, testen, bugs oplossen en het product op de markt brengen. Dit betekend dat er soms meerdere jaren duurde voordat een product de markt op kwam waardoor soms wel het probleem was veranderd wat de requirements niet meer valide maakte.  
  
Voor de ontwikkelaars betekende dat ze een product hadden ontwikkeld die geen plek meer had op de markt.

In meeste gevallen werd het project terug in de kast gelegd wat weer kon stof happen.

Meerdere ontwikkelteams veranderde hun aanpak rond de 1990 waardoor er meerdere ontwikkelingsmethodieken ontstonden gedurende deze tijd. Door een kritische vergadering in 2001 is de beroemde Agile manifesto ontstaan die tot de tijd van vandaag nog steeds gebruikt wordt tijdens ontwikkeling.

Agile is een iteratief framework die in een team verband verloopt. Deze aanpak van een ontwikkelingsmethode legt de aandacht op het snel leveren van tussenproducten. Dit is waarom dit een incrementeel aanpak is in plaats van een iteratief aanpak (Agility, 2021). Wat Agile sterk maakt, zijn de korte loops binnen het ontwikkelingsperiode.

Agile methode zorgt voor een iteratieve ontwikkeling. Dit betekend dat componenten niet in één keer af zijn maar in delen gemaakt worden. Door kleine delen toe te voegen per iteratie aan het compleet deel, kan een eindproduct ontwikkeld worden



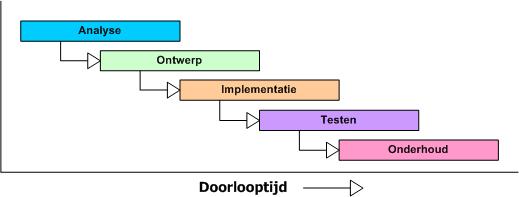
Figuur Agile loop

Elke deel wordt binnen een iteratie gemaakt. Gedurende een iteratie wordt gekeken naar welke requirements het hoogst geprioriteerd zijn. De hoogst geprioriteerde requirements of user stories worden binnen een iteratie geïmplementeerd genomen (Toolsqa, 2021).

Aan het eind van elke iteratie zal we feedback momenten plaatsvinden. Deze momenten worden samen met de klant gedaan. De klant kan hier feedback geven op het gedeelte dat is geïmplementeerd. Samen met de feedback kan het ontwikkelteam de volgende sprint in om de volgende functionaliteiten te kunnen implementeren. Dit zal door gaan tot dat de scope is afgerond of de klant geen budget meer beschikbaar maakt voor het project.

# Waterval

Waterval biedt een lineaire ontwikkeling aan voor een project waarbij de fases doorlopen worden. Het hoeft niet direct als de slechte methodiek geschetst te worden omdat daaruit goed gedefinieerde projecten uit kunnen ontstaan (Seguetech.com, 2018). Het biedt een efficiënt en een gestroomlijnd proces die makkelijk te volgen is. Dit is mogelijk doordat de volledige scope van het project vanaf het begin al duidelijk is voor de ontwikkelaars.



Figuur Waterval fases

Doordat de project scope vanaf het begin al duidelijk is, is het voor de ontwikkelteam en de product owner duidelijk welke producten ontwikkeld zal worden (Seguetech.com, 2018). Wanneer alle requirements duidelijk zijn voor beide partijen is de volledige scope duidelijk en zal de planning en ontwikkeling soepeler gaan. Verder is het volgen van de ontwikkeling makkelijker omdat de ontwikkeling volgens een tijdlijn zal gaan en er zijn vooraf gedefinieerde mijlpalen (Seguetech.com, 2018).

Doordat alle requirements al vaststaan, is het mogelijk voor de ontwikkelteam hoe het product ontwikkeld zal worden. Voor het project geldt dat alle requirements vanaf het begin al duidelijk zijn voor beide partijen, zowel de ontwikkelaar en de stakeholders. Beide partijen zullen overeenkomen wat er verwacht worden en zal vanaf het begin alles duidelijk zijn voor de implementatie van start zal gaan.

Desondanks deze voordelen, wat waterval minder effectief maakt zijn de requirements. Tijdens ontwikkeling kunnen er veranderingen voorkomen die ervoor zorgen dat de requirements anders geformuleerd moeten worden. Wanneer de requirements op de oude manier geïmplementeerd zal worden, kan het eindproduct niet meer het probleem oplossen voor wat het gemaakt wordt.

Dit betekend uiteindelijk dat de klant niet genoegen zal hebben aan het eindproduct omdat de probleem ondertussen is veranderd. Dit kan betekenen dat de product dat in de tussentijd is ontwikkeld niet meer voldoende is voor het probleem.

## Conclusie Agile vs Waterval

Het belangrijkste binnen het project is dat gedurende het project requirements kunnen veranderen. Om dit te kunnen faciliteren is hetDeze flexibiliteit zal het project ten goede doen doordat de kans groot is de requirements tussen door het project verandert kunnen worden.

Binnen waterval zal het niet mogelijk zijn om terug naar de requirements te gaan om de requirements aan te passen. Dit geldt niet alleen voor de requirements. Binnen waterval kan binnen de fases terug gekeken worden en aanpassingen gedaan worden. Na een fase afgesloten wordt, kan men niet terug naar de vorige fase.

Niet kunnen aanpassen aan veranderingen betekent dat men niet kan aanpassen aan het probleem. Gedurende het project kan het probleem een andere vorm aannemen waardoor het oplossing ervoor niet meer voldoende is. Daarentegen biedt Agile de mogelijkheid om deze veranderingen in mee te nemen. Dit betekend het project up-to-date blijft bij veranderingen.

# Agile Frameworks

Na besluit genomen te hebben om Agile te gebruiken zal onderzocht worden welke framework het beste past binnen het project. Hierin was de keuze gelimiteerd door het bedrijf. Doordat Jongens van Techniek en Uptime Heroes gebruik maken van Scrum, was de verwachting dat ik daarin mee zal doen.

Binnen Jongens van Techniek zijn er verschillende projecten gedurende tijd van schrijven. Tijdens een daily standup zijn er verschillende werkzaamheden cross-project. Een daily standup was niet gefocussed op één specifiek project. De MonitoringApp was gedurende het project ook een onderdeel daarvan.

## Scrum

Binnen Agile zijn verschillende frameworks beschikbaar die gebruikt kunnen worden. Gedurende project zal Scrum gebruikt worden doordat deze goed werkt in een klein team. Kanban is een andere soort Agile framework maar deze is meer gericht op het verbeteren met het ontwikkelproces en niet het ontwikkelen van producten als eerste prioriteit. Met Scrum is het mogelijk om gebruik te maken van de Agile voordelen en ook producten te kunnen leveren.

Scrum is een Agile framework die de voordelen van Agile gebruikt om snel producten te leveren aan de klant. Het is een effectieve en flexiebele manier van ontwikkelen van producten doordat er gebruik gemaakt wordt van korte “loops”.

Deze loops worden sprints genoemd. Binnen deze voor-gedefinieerde sprints worden een aantal artefacten uitgevoerd. Dit zijn werkzaamheden die binnen de sprint uitgevoerd worden. Wanneer een sprint is afgerond, wordt deze beoordeeld door de stakeholders en wordt een nieuwe sprint gestart. Door elke sprint kort te houden en daarna deze te laten beoordelen door de stakeholder, geeft de stakeholder de indruk dat er iets geproduceerd wordt. Bij waterval zal het product pas na een jaar ontwikkeling bij de stakeholder terecht komen. Dan pas kunnen ze een product zien en beoordelen.

Scrum heeft een aantal rituelen:

1. Daily standup
2. Backlog refinement
3. Sprint planning
4. Sprint review
5. Retrospective

Daarnaast zijn er een aantal rollen binnen Scrum:

1. Product eigenaar (product owner)
2. Scrum Master
3. Ontwikkelteam

**Daily standup**  
De daily standup vindt dagelijks plaats aan het begin van een dag. Dit is een vaste tijdstip aan het begin van de dag waarbij elke teammember vertelt wat hij/zij zal gaan doen. De daily standup is altijd op een vaste tijdstip en zal desondanks afwezigen van start gaan. Verder wordt de standup kort gehouden, binnen 15 minuten zal de standup afgelopen moeten zijn. Tijdens een standup kan verteld worden wat men zal gaan doen en of er impedimenten zijn, (uitdagingen, problemen of uitzonderingen).

### Backlog refinement

Tijdens een refinement sessie wordt samen met de product eigenaar (productowner) de user stories gerefined. Dit betekend dat ze worden besproken en aangevuld worden mocht dit nodig zijn. Tijdens refinement sessies kan de team vragen stellen aan de product eigenaar als deze de story niet begrijpt. Daarnaast wordt tijdens deze sessie de stories opnieuw geprioriteerd om vast te stellen welke stories de productowner geïmplementeerd wilt zien.

### Sprint planning

Aan het begin van de sprint is er een sprint planningsoverleg. Binnen deze overleg wordt vastgesteld welke userstories van de backlog in de sprint worden genomen. Het is belangrijk dat de ontwikkelteam deze beslissing maakt omdat zij diegene zijn die de onderliggende taken moeten maken. Daarom bepaalt het team hoeveel werk opgenomen kan worden in de sprint en is het team verantwoordelijk voor de inschatting van de hoeveelheid werk per userstory. Dit wordt gedaan door punten toe te wijzen aan de user stories.

### Sprint review

Een sprint wordt afgesloten met een sprint review. Tijdens de sprint review wordt het product getoond die gedurende sprint is ontwikkeld. Hierin is de gehele team bij aanwezig inclusief de product owner en andere geïnteresseerden.

### Retrospective

Bij Scrum wordt aangemoedigd dat men van elkaar leert. Daarom is het terug kijken naar de afgelopen sprint een handige moment om aan te geven wat er goed is gegaan en wat er minder goed is gegaan. Hiermee kan men aangeven wat hij/zij moeilijk vond, wat opviel. Het is niet de bedoeling om mensen de schuld te geven bij een vertraging. De bedoeling is om deze vertraging te kunnen identificeren zodat men eraan kan werken voor de volgende sprint.

## Verschillen Agile vs. Waterval

Er zijn een naar belangrijke verschillen tussen Agile en Waterval. Deze verschillen zijn te categoriseren zodat de belangrijke verschillen naar voren komen.

De categorieën zijn:

1. Klant
2. Scope
3. Prioriteit
4. Team
5. Kosten

In de tabel hieronder worden de verschillen per methode uitgelegd.

|  |  |  |
| --- | --- | --- |
|  | Agile | Waterval |
| 1. Klant | Klant betrokkenheid aan het einde van alle sprints gewenst. | Klant betrokkenheid alleen gewenst tussen faseovergang |
| 1. Scope | Scope veranderingen worden verwelkomd maar zal ten koste gaan van het rooster en kosten. | Scope is bekend en zal niet wijzigen gedurende het project |
| 1. Prioriteit | Prioriteiten zullen voordurend veranderen. De klant zal op prijs stellen dat zijn hoogste prioriteit zo snel mogelijk geïmplementeerd zal worden. | De ontwikkeling zal gaan volgens de planning. De klant zal de volgorde van de tussenproduct niet merken omdat het product iteratief ontwikkeld zal worden en dus niet gelijk een werkend product zal ontvangen. |
| 1. Team | Kleine teams dat bestaan uit verschillende specialiteiten. | Grote teams zijn acceptabel. |
| 1. Kosten | Kosten van het project kan verschillen. Dit komt doordat de scope kan veranderen waardoor extra tijd/funding nodig is voor het project. | Kosten wordt vooraf bepaald. Het project wordt met de bestaande kosten ontwikkeld. |

# Toepassing Scrum

Door de grootte van het team en gelimiteerde resources kunnen niet alle onderdelen van Scrum meegenomen worden. Hierdoor is besloten om Scrum te gebruiken maar sommige elementen van Scrum weg te laten.

Samen met de product eigenaar zal de ontwikkelaar een refinement sessie in gaan om zo de user stories die nog niet duidelijk genoeg zijn te kunnen verbeteren. Daarna zal aan de hand daarvan de sprint opgezet worden door een thema te geven aan de sprint. Deze thema zal bepaald worden door de ontwikkelaar. Hierin zal een story in meegenomen worden aan de hand van de prioriteiten die gegeven zijn aan de user stories. De ontwikkelaar heeft daarna de ruimte om een review te doen van de opgeleverde functionaliteiten en zal daarna feedback ontvangen van de product eigenaar. Mocht deze goed zijn, dan kan de sprint succesvol afgesloten worden. Als er nog afwijkingen zijn, kan daarvoor een nieuwe user story voor aangemaakt worden die de volgende vergadering gerefined kan zijn.

Bibliografie

Jim Highsmith. (2001, 01 01). *The Agile manifesto*. Opgehaald van TheAgileManifesto: https://agilemanifesto.org/history.html

## Bijlage 9: Chilkat certificate check

Afbeelding met tekst

Automatisch gegenereerde beschrijving

## Bijlage 10: Startup File

Afbeelding met tekst

Automatisch gegenereerde beschrijving

## Bijlage 11: Test Rapportage

Test rapport

**Jongens van Techniek**

*Auteur : Necati Unal*

*Datum : 15 Oktober 2021*

Inhoudsopgave

[Inleiding 178](#_Toc85157566)

[Pinger test 179](#_Toc85157567)

[Logische test gevallen 180](#_Toc85157568)

[Fysieke testgevallen 181](#_Toc85157569)

[Resultaat unittests 182](#_Toc85157570)

[Melding HighPing 183](#_Toc85157571)

[CertificaatCheck test 184](#_Toc85157572)

[Logische test gevallen 185](#_Toc85157573)

[Fysieke test gevallen 185](#_Toc85157574)

[Uitslagen 186](#_Toc85157575)

[Alert 187](#_Toc85157576)

[Bijlages 189](#_Toc85157577)

[Certificate Check Tests 189](#_Toc85157578)

[Pinger Tests 193](#_Toc85157579)

# Inleiding

In dit document worden de testresultaten beschreven. Eerst wordt de pinger function getest. Hiervoor zijn de acceptatiecriteria opgesteld die ervoor zorgen dat alle gevallen gedekt kunnen worden. Vervolgens wordt de CertificateCheck getest. Hiervoor wordt ook alle gevallen gedekt door eerst voorwaardes op te stellen die de function moet valideren.

Voor elke function worden logische en fysieke testgevallen gemaakt. Deze vormen de testgevallen waarop getest moet worden binnen Unittests. Deze tests volgen met een screenshot van de tests.

# Pinger test

De Pinger test wordt gebaseerd op de volgende user story:

* Als supportmedewerker wil ik inzicht hebben op downtimes zodat ik weet of er een probleem zich afspeelt.

Voor deze user story gelden er een aantal randwaardes die als acceptatiecriteria dienen. Gedurende het interview met Peter is besloten dat alle pings hoger dan 150 een error melding moeten zijn. Alles boven 150 is een foutmelding. Alle pings onder de 150 kunnen als acceptabel worden geaccepteerd.

Hiervoor geldt de volgende acceptatiecriteria:

1. AC1: MS boven de 150
2. AC2: MS onder de 150

Om het resultaat van de tests te kunnen voorspellen worden de tests gesplit in geldige en ongeldige gevallen. Hiermee is het later mogelijk te achterhalen of de juiste test is geslaagd en of het de juiste gedrag vertoond.

|  |  |  |
| --- | --- | --- |
| **Attribuut** | **Geldig** | **Ongeldig** |
| RTT | Onder 150  Boven 150  2 Digit double  3 Digits double  4 Digits double  5 Digits double | Lege getal  String  3 digits integer  4 digits integer  5 digits integer |

## Logische test gevallen

De gevallen worden uit de geldig/ongeldig kolom gehaald in tabel 1. Hieronder zijn de logische testgevallen waar uitgegaan van wordt tijdens testen.

|  |  |  |
| --- | --- | --- |
| Geval # | Geval | Resultaatvoorspelling |
| 1 | Onder 150 | Valide |
| 2 | Boven 150 | Valide |
| 3 | 2 Digit double | Valide |
| 4 | 3 Digits double | Valide |
| 5 | 4 Digits double | Valide |
| 6 | 5 Digits double | Valide |
| 7 | Lege getal | Invalide (error) |
| 8 | String | Invalide (error) |
| 9 | 3 digits integer | Invalide (error) |
| 10 | 4 digits integer | Invalide (error) |
| 11 | 5 digits integer | Invalide (error) |

## Fysieke testgevallen

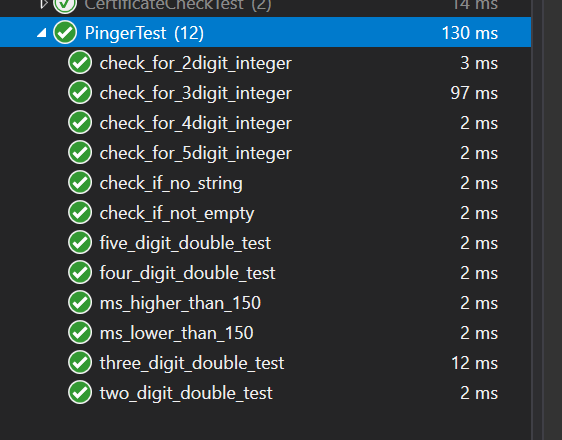
In het tabel hieronder is te zien welke resultaten in code voortkomen. De miliseconden die als resultaat verkregen worden, is een double object waar op gecontroleerd wordt. Wanneer dit niet geconverteerd kan worden, wordt er een string gegeven die opgevangen wordt als een foutmelding.

Als laatst wordt alle miliseconden boven de 150 gemarkeerd als een HighPingAlert. Dit betekend dat wanneer het resultaat bestaat uit een Double object maar boven de 150. Er een alert van gemaakt moet worden.

|  |  |  |
| --- | --- | --- |
| Geval | Logische test geval # | Resultaat |
| RTT = 4 | 1 | “RTT” |
| RTT = 170 | 2 | “HighPingAlert” |
| RTT = 17.22 | 3 | “RTT” |
| RTT = 107.22 | 4 | “RTT” |
| RTT = 1700.22 | 5 | “RTT” |
| RTT = 17000.22 | 6 | “RTT” |
| Int RTT = ; | 7 | Exception |
| RTT string | 8 | Exception |
| INT RTT = 170 | 9 | Exception |
| INT RTT = 1700 | 10 | Exception |
| INT RTT = 17000 | 11 | Exception |

## Resultaat unittests

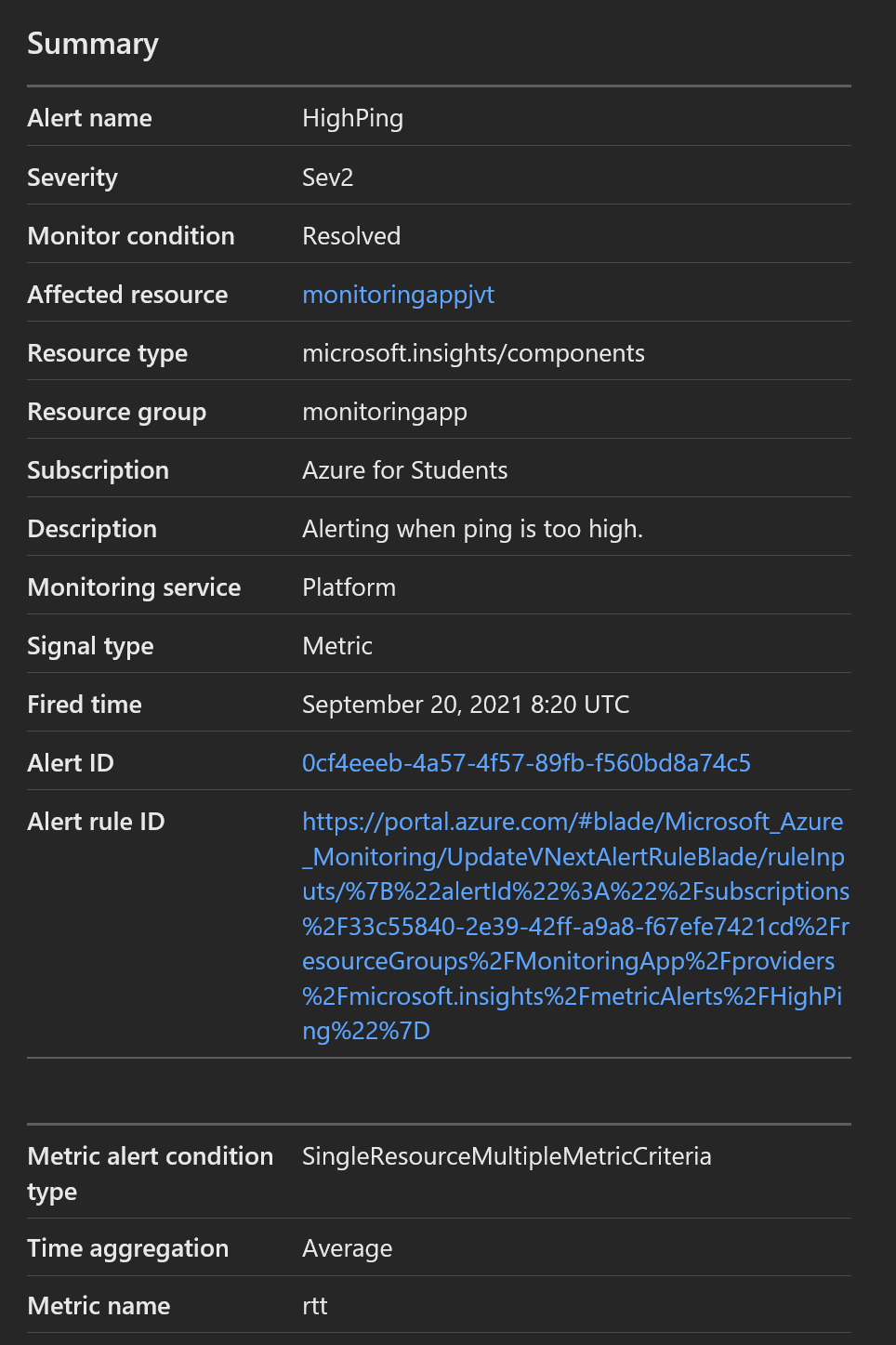
Door de tests in Visual studio te runnen is het mogelijk om de resultaten van de tests op te vragen. Hieronder is een overzicht te zien van de resultaten van alle tests.



Figuur Resultaten unittests

## Melding HighPing

De situatie bij een HighPingAlert is iets anders. Dit is hardcoded te testen maar het resultaat daarvan kan niet getest worden. Het resultaat van een HighPingAlert is dat er een email verstuurd wordt naar de desbetreffenden. In deze email wordt aangegeven dat er een ping is gedaan waaruit geconcludeerd is dat de ms hoger dan 150 is.



Figuur Alert High Ping

Een voorbeeld hiervan is te vinden in figuur 1. Er zijn een aantal velden die belangrijk zijn. De monitoringCondition staat op resolved omdat er een mechanisme eraan vast zit die dit bepaald. De condition zal staan op Fired zodra deze eerste keer is getriggered. Zodra de metric de volgende keer met een waarde minder dan 150 komt, zal de mechanisme de condition op resolved zetten doordat deze niet meer wordt getriggered.

# CertificaatCheck test

Het testen van Certificaatcheck is gebaseerd op de volgende user story: “Als supportmedewerker wil ik inzicht hebben op verlopen certificaten, zodat ik weet of er een probleem zich afspeelt.”.

Voor de user story zijn er acceptatiecriteria opgesteld. Gedurende gesprekken met David (bijlage 1) is vastgesteld dat alle certificaten minder dan 30 dagen geldig, als een error behandeld wordt.

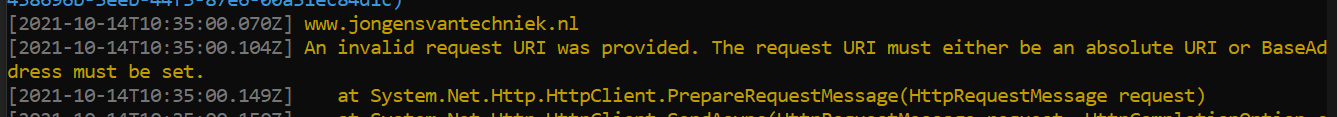
De acceptatiecriteria gebaseerd op deze user story zijn:

1. AC1: resterende tijd wordt vertoond in dagen
2. AC2: Resterende tijd minder dan 30 dagen worden als fout behandeld
3. AC3: Resterende tijd meer dan 0 wordt verstuurd naar Application Insight
4. AC4: Endpoints zijn string in een array

Om het resultaat van de tests te kunnen voorspellen worden de tests gesplit in geldige en ongeldige gevallen. Hiermee is het later mogelijk te achterhalen of de juiste test is geslaagd en of het de juiste gedrag vertoond.

|  |  |  |
| --- | --- | --- |
| **Attribuut** | **Geldig** | **Ongeldig** |
| DAYS | Boven 30  Onder 30 | Lege getal  String |

Voor de endpoints zijn er een aantal elementen belangrijk. Om een certificaat binnen te halen moet de endpoint aangesproken worden met een http. De URI object vraagt om een volledige adres voordat het een aanvraag wilt gaan doen (Figuur 3). Hierdoor moet begonnen worden met een “http://” en niet met een [www.](http://www.).



Figuur URL error

|  |  |  |
| --- | --- | --- |
| **Attribuut** | **Geldig** | **Ongeldig** |
| URL | Strings  Begint altijd met een http | Lege string  Begint met www |

## Logische test gevallen

Hieronder is een samenstelling van alle attributen die getest zullen worden. Achter de gevallen wordt er aangegeven wat het resultaat van de test zal moeten worden en hoe het behandeld moet worden. In dit geval wordt er uitgegaan van valide en invalide.

|  |  |  |
| --- | --- | --- |
| Geval # | Geval | Resultaatvoorspelling |
| 1 | DAYS boven 30 | valide |
| 2 | Days onder 30 | Valide |
| 3 | Days met leeg getal | Invalide |
| 4 | Days als string | Invalide |
| 5 | URL als string | Valide |
| 6 | URL begint met een http | Valide |
| 7 | URL als lege string | Invalide |
| 8 | URL begint met een www | Invalide |

## Fysieke test gevallen

Nadat de logische gevallen duidelijk gemaakt zijn, worden de ze omgezet in fysieke testgevallen.

In het tabel hieronder is te zien welke deze zijn. Beide attributen worden getest in de fysieke testgevallen. De grenswaardes worden getest om vast te stellen of de alert goed geplaatst zijn waardoor email trigger wordt geactiveerd.

|  |  |  |
| --- | --- | --- |
| Geval | Logische test geval # | Resultaat |
| Days = 31 | 1 | valide |
| Days = 29 | 2 | Valide |
| Days = ; | 3 | Invalide |
| Days = “2” | 4 | Invalide |
| URL = “https://jongensvantechniek.nl” | 5 | Valide |
| URL = “https://jongensvantechniek.nl” | 6 | Valide |
| URL = “” | 7 | Invalide |
| URL = “www.jongensvantechniek.nl” | 8 | Invalide |

## Uitslagen

Hieronder zijn de uitslagen van de tests te vinden.

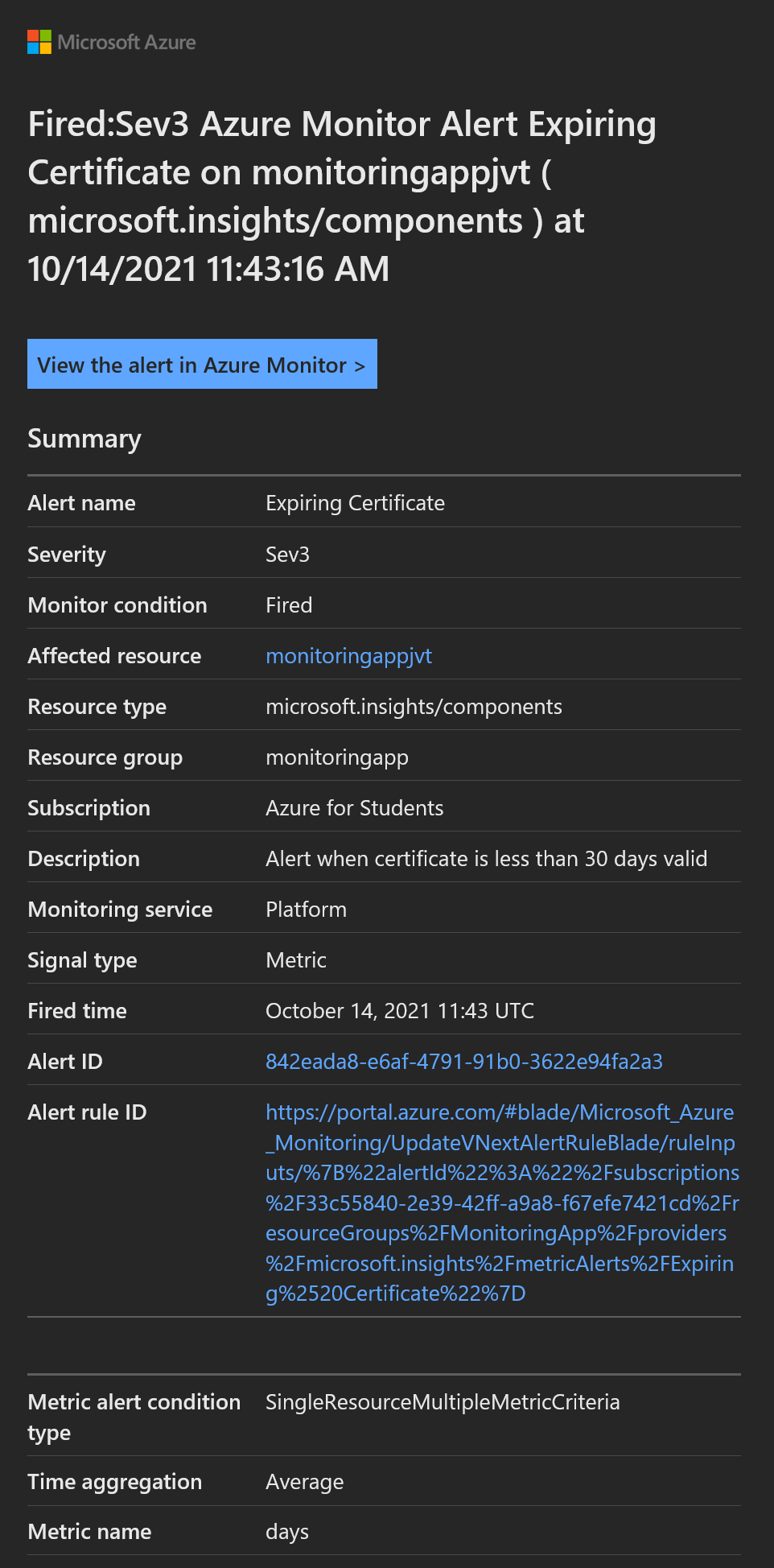
Afbeelding met tekst

Automatisch gegenereerde beschrijving

# Alert

Om verlopen certificaten te kunnen testen wordt binnen de tests gekeken naar DAYS lager dan 30. Wanneer dit het geval is wordt er een notificatie verstuurd naar de supportmedewerker om diegene op de hoogte te houden van een certificaat die binnenkort zal verlopen. Het ontvangen van deze notificatie email wordt manueel getest. Hiervoor is gezorgd dat de DAYS waarde die naar Application Insight verstuurd wordt lager is dan 30 om handmatig de condition te triggeren van de Alert.

De email die daarbij hoort is te vinden in figuur 4.



Figuur Notificatie expired certificate

# Bijlages

## Afbeelding met tekst Automatisch gegenereerde beschrijvingCertificate Check Tests

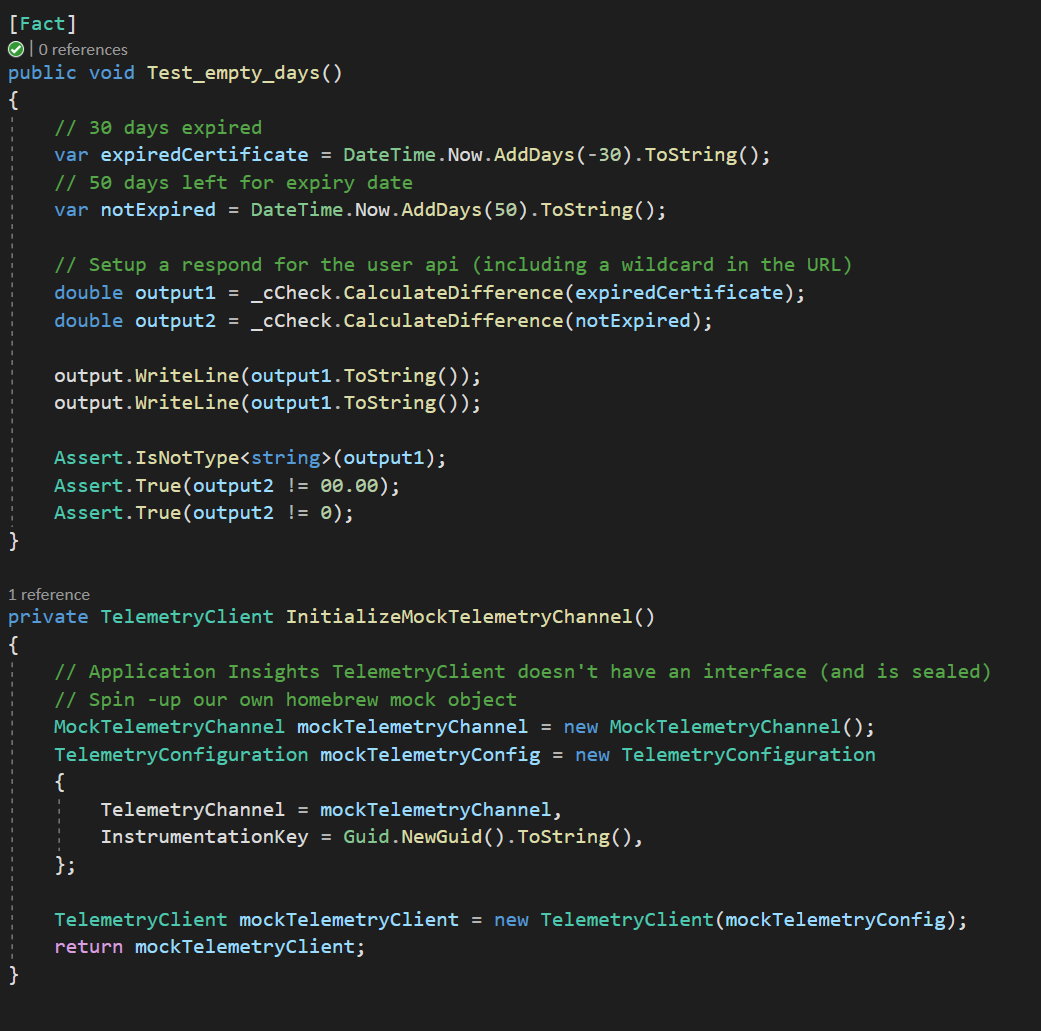
Figuur Testcode deel 1



Figuur Testcode deel 2

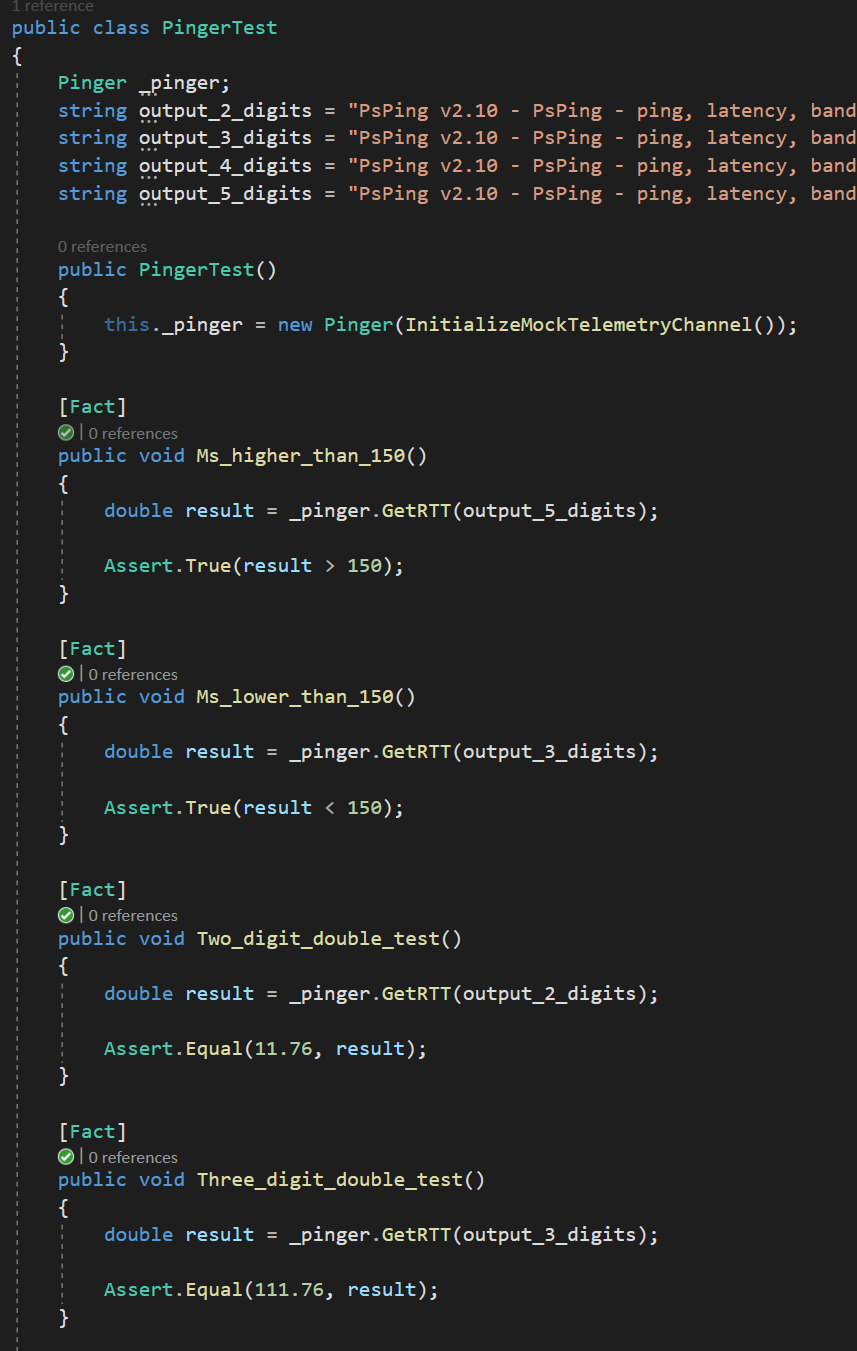


Figuur Test code deel 3

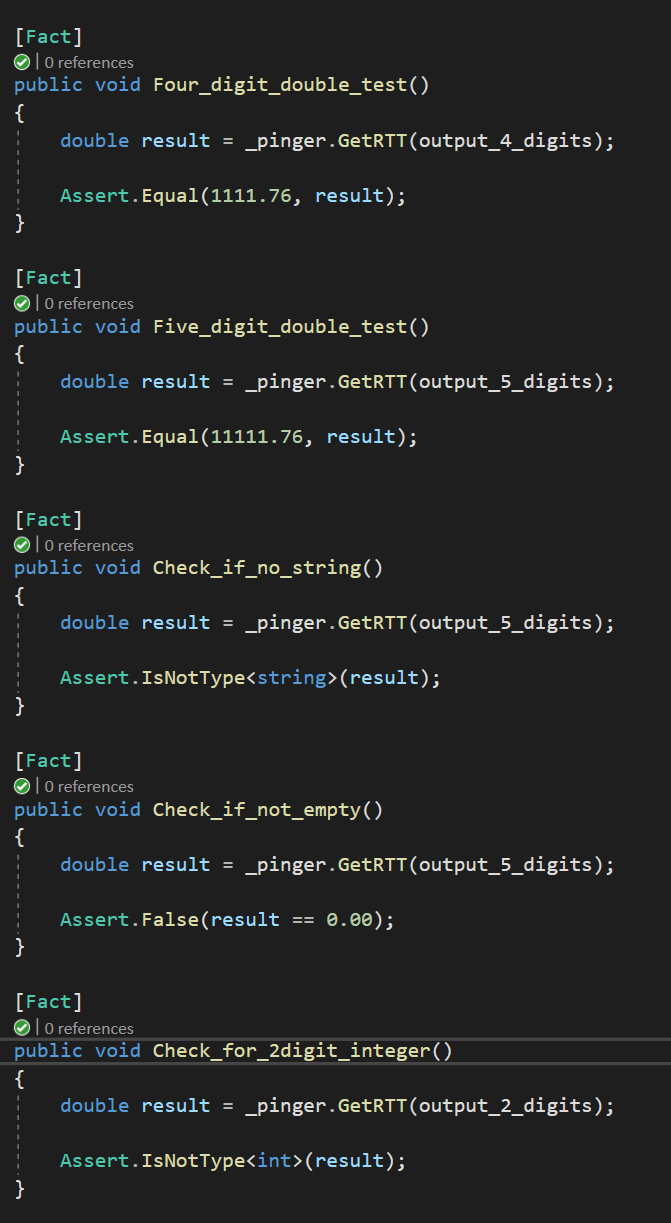


Figuur test code deel 4

## Pinger Tests



Figuur Test code deel 1



Figuur Test code deel 2

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Figuur Test code deel 3

## Bijlage 12: Grafieken Jongens van Techniek

## bijlage 13: Grafieken NSFO.nl

## Bijlage 13: Stakeholder Analyse

Stakeholderanalyse

Auteur: Necati unal

Datum: 15-04-2021

Inhoudsopgave

[1.0 Inleiding 193](#_Toc85153889)

[2.0 Analyse 194](#_Toc85153890)

[2.1 Stakeholders in het businessdoel 194](#_Toc85153891)

[2.2 Stakeholders bij het systeem 194](#_Toc85153892)

[2.3 Stakeholders bij het project 194](#_Toc85153893)

[3.0 Vertegenwoordigers 195](#_Toc85153894)

[3.1 Selectie 195](#_Toc85153895)

[3.1.1 Belanghebbenden in het business 195](#_Toc85153896)

[3.1.2 Belanghebbenden in het Systeem 195](#_Toc85153897)

[3.1.3 Belanghebbenden bij het project 19](#_Toc85153898)6

[3.2 Overzicht vertegenwoordigers 196](#_Toc85153899)

[3.3 Stakeholders informeren 198](#_Toc85153900)

[4.0 Conclusie 198](#_Toc85153901)

[Bibliografie 199](#_Toc85153902)

**Inleiding**

In dit document wordt een stap gezet richting de stakeholderanalyse. Deze analyse wordt gebruikt om te kunnen beslissen op welke stakeholders we ons moeten richten.  
  
Om de juiste stakeholders te kunnen analyseren wordt gebruik gemaakt van een analyse. Hiermee wordt verduidelijkt op wie gedurende het project de focus op gelegd moet worden om de juiste requirements naar boven te halen.

Te beginnen wordt gekeken naar de drie categorieën waarin de stakeholders in vallen. Deze categorieën vallen onder de stakeholder analyse zoals benoemd in Handboek Requirements (Nicole de Swart, 2010). Per categorie wordt behandeld welke stakeholders daarin vallen.  
  
Vervolgens worden de beschikbare stakeholders die geselecteerd zijn voor het project onder een categorie geplaatst. Deze personen worden daarna aan de hand van een checklist beoordeeld of ze respresentatief genoeg zijn voor de rol. Hieruit is gebleken dat één persoon hier niet onder past waardoor deze persoon niet als stakeholder gebruikt zal worden.

Als laatst worden deze stakeholders beoordeeld op belang door de Mendelow influence vs interest model. Dit wordt gebruikt om vast te stellen hoeveel communicatie nodig is voor deze personen gedurende het project. Bi

**Analyse**

Voor de analyse wordt gerefereerd naar de analyse gebruikt door Nicole de Swart (Nicole de Swart, 2010). Voor het analyse wordt gebruik gemaakt van drie categorieën.

Belanghebbenden worden ingedeeld in drie groepen:

1. Stakeholders in het businessdoel
2. Stakeholders in het systeem
3. Stakeholders in het project

**Stakeholders in het businessdoel**

Stakeholders in het businessdoel hebben belang bij het oplossen van een probleem of het behalen van een businessdoel. Om dit te kunnen bereiken hebben deze personen een stap gezet richting het oplossen van het probleem. Om dit op te lossen willen zij een software ontwikkeld hebben die het probleem oplost.

**Stakeholders bij het systeem**

Stakeholders bij het systeem zijn mensen die dagelijks in contact komen met het systeem. Hieronder valt iedereen van beheerders, klanten tot medewerkers.   
  
Voor dit doelgroep betekent het niet altijd dat de verandering een voordeel is. Dit kunnen ook negatieve veranderingen zijn waardoor men vast zit (Swart, 2010). Dit komt door de limitaties van het opgeleverde systeem of andere problemen die met het systeem komen.

**Stakeholders bij het project**

Deze stakeholders kunnen invloed hebben op de werking van het systeem. Deze hebben in vele gevallen ook invloed tot na het uitleveren van het project. Hieronder vallen meestal projectleden zelf, andere ICT-medewerkers of ICT-leveranciers die te maken hebben met het project.

**Vertegenwoordigers**

De volgende stap is het beslissen van de vertegenwoordigers van de doelgroep. Hierbij moet gekeken worden naar de representativiteit van de desbetreffende persoon. Hiervoor is een wordt gebruik gemaakt van een checklist (Nicole de Swart, 2010).

De checklist houdt in:

1. De vertegenwoordiger weet wat er speelt bij de belanghebbenden
2. De belanghebbenden hebben vertrouwen in de vertegenwoordiger
3. De vertegenwoordiger is bevoegd om namens de belanghebbenden te spreken
4. De vertegenwoordiger heeft genoeg tijd
5. De vertegenwoordiger beschikt over goede communicatieve skills.

**Selectie**

Eerst zal een selectie van stakeholders gekozen worden voor de desbetreffende categorieën. Hiervoor wordt gebruik gemaakt van de checklist in h 3.0.

**Belanghebbenden in het business**

Bij het kiezen voor belanghebbenden in de business wordt gekeken wie er invloed heeft op het project. Uit eerste instantie is dit iedereen die te maken zullen hebben met het project. Diegene die het meest invloed heeft is de projectleider, oftewel Peter Hofman. Hij heeft het project geïnitialiseerd. Hij heeft belang bij het implementatie van de MonitoringApp doordat hij hierop verder zijn business kan uitbreiden.

Bij het kiezen valt op dat het lastig is contact te leggen met deze persoon. Voor het project zal deze persoon tijdig aangesproken moeten worden om gesprekken en interviews af te kunnen nemen. Verder is er geen punt e maken qua vertrouwen doordat deze persoon zelf het project opgebouwd. Deze persoon heeft belang bij het implementatie van het software.

**Belanghebbenden in het Systeem**

Voor belanghebbenden in het systeem zijn meerdere personen beschikbaar. Onder deze categorie vallen mensen die dagelijks contact zullen hebben met het systeem. De belangrijkste hier zijn de klanten en werknemers.   
  
Doordat er nog geen klanten zijn voor het monitoringssysteem is besloten nog geen stakeholders uit de klanten groep te nemen. Hiervoor is besloten de Supportmedewerker van Uptime Heroes het woord te geven. Deze persoon heeft belang in de werking van het systeem. Hij zal dagelijks ermee te maken hebben en zal ermee moeten werken. De Supportmedewerker zal dus een bron zijn van feedback binnen het project.   
  
Verder is er ook belangrijk dat ontwikkelaars opgenomen worden die met het systeem te maken krijgen. Hiervoor is de Senior ontwikkelaar, Nico Pronk voor gekozen als stakeholder. Na het aflopen van deze stakeholder is opgevallen hierbij dat deze persoon weinig tijd zal hebben voor het project doordat er andere projecten lopende zijn waar hij zich bezig mee houdt. Hierdoor valt deze persoon er weer van af.

**Belanghebbenden bij het project**

Voor belanghebbenden in het project is gekozen voor Jonathan Bosman. Deze persoon is beschikbaar voor het project en zal ook invloed hebben in de werking van het systeem. Doordat deze persoon een rol heeft als projectmanager, zal hij na het project invloed hebben op het project. Hij is de tussenpersoon tussen klanten en de ontwikkelteam waardoor gekozen is voor Jonathan.

Daarnaast valt Peter Hofman ook onder de belanghebbenden voor het project. Hij heeft invloed op de werking van het project. Hij heeft invloed op de tests en welke oplossingen gebruikt kunnen worden. Verder heeft hij ook invloed op de omgeving waarop de MonitoringApp op zal draaien. Door zijn titel heeft hij ook invloed op het budget van het project waardoor hij een belangrijke stakeholder voor dit groep vormt.

**Overzicht vertegenwoordigers**

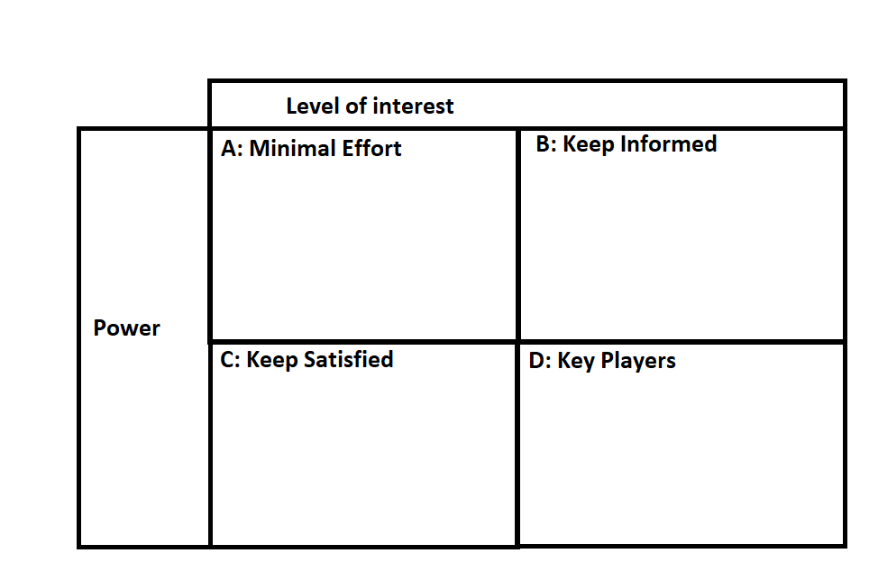
Ter overzicht worden de geselecteerde vertegenwoordigers hieronder opgesomd.

|  |  |  |
| --- | --- | --- |
| **Belanghebbenden Business** | **Belanghebbenden Systeem** | **Belanghebbenden Project** |
| Peter Hofman | David Mesman | Jonathan Bosman |
|  |  | Peter Hofman |

3.3 Interesse vs. invloed

Stakeholders kunnen invloed hebben in het systeem maar zullen niet altijd op de hoogte gehouden hoeven worden. Het verschil tussen interesse en invloed kan gemodelleerd worden om vast te leggen welke persoon vaker bij het project betrokken moeten worden. Hiervoor wordt een model gebruikt om de desbetreffende gebruikers in de juiste box te kunnen plaatsen.

Hiervoor wordt gebruik gemaakt van de Mendelow interest vs influence model (Oxford, 2021) als volgt:



Figuur Influence model

Het model is ingedeeld in twee waardes. Power en Level of interest.  
  
Power staat gelijk aan de hoeveelheid invloed die deze persoon zal hebben op het project. Level of interest staat gelijk aan de interesse die deze persoon zal hebben in het project. Hiermee kan een schatting gedaan worden waar de desbetreffende stakeholder in geplaatst kan worden.

Dit is als volgt:

Afbeelding met tafel

Automatisch gegenereerde beschrijving

**Stakeholders informeren**

Om het project naar een succes te kunnen tillen zullen deze stakeholders volgens het model geïnformeerd te worden. Zodra ze geïnformeerd kunnen worden volgens het model is het mogelijk om meer feedback te ontvangen zodat het project een succes kan worden.

Dit wordt gedaan aan de hand van de volgende categorieën

1. Coalitiepartners: Deze groep staat achter de inhoud van het project maar hebben weinig vertrouwen in de relatie. Vertrouwen vergroten door afspraken te maken en duidelijk te communiceren.
2. Bondgenoten: Deze groep zijn overeen met de ontwikkelaars. Zij zullen snel accepteren omdat ze vertrouwen hebben in het project en de ontwikkelaars.
3. Tegenstanders: Deze groep heeft geen vertrouwen in het project en zullen dit ook heel duidelijk maken. Met deze groep weet je waar je aan toe bent en kan je haalbaarheid geven door grenzen te zetten binnen je project.
4. Vijanden: Deze groep is het oneens met de inhoud van het project en zullen daarnaast ook weinig vertrouwen hebben in de ontwikkelaars. Besteed niet veel energie aan deze doelgroep. Het is soms handiger om een bondgenoot in te schakelen om communicatie te kunnen voeren met dit groep.
5. Opportunisten: Deze groep willen in op het project alleen hebben geen vertrouwen Ze blijven kijken tot het laatste moment en zullen dan pas actie ondernemen.
6. Twijfelaars: Deze groep is vaak nog niet voldoende geïnformeerd. Dit resulteert in dat ze hun standpunt maken op foutieve informatie. Door informatie te verschaffen kan deze groep betrokken worden.

Ter opsomming vallen de stakeholders in de volgende categorieën

1. David – Bondsgenoot
2. Jonathan – Opportunisten
3. Peter – Coalitiepartner

**Conclusie**

Ter conclusie zijn er in principe drie stakeholders die gedurende het project van belang zijn.

1. David
2. Peter
3. Jonathan

Als laatste is Nico Pronk ook beschikbaar als senior ontwikkelaar alleen door tijdslimiet zal deze persoon niet altijd beschikbaar zijn. Voor dit persoon geldt dat hij op de hoogte gehouden kan worden van de veranderingen gedurende het project zodat hij geïnteresseerd blijft. Doordat sommige calamiteiten bij Nico kunnen komen via David, is het belangrijk dat deze persoon op de hoogte gehouden zal worden van veranderingen.

Verder is David een belangrijke stakeholder. Hij is de enige supportmedewerker binnen Uptime Heroes. Dit betekend dat de beschikbaarheid voor dit groep klein is. Hierdoor is er geen mogelijkheid om te kiezen voor andere vertegenwoordigers. Desondanks wilt dit niet zeggen dat David een slechte vertegenwoordiger is. Hij wordt als eerst benaderd bij calamiteiten en kan dit oplossen.

**Bibliografie**

Mirjam Sluis. (2006). *Projecten op de rails.* Amsterdam: Boom uitgevers.

Nicole de Swart. (2010). *Handboek Requirements.* Delft: Eburon Business.

Oxford. (2021, 01 01). *What Is Mendelow’s Matrix And How Is It Useful?* Opgehaald van Oxford college of marketing: https://blog.oxfordcollegeofmarketing.com/2018/04/23/what-is-mendelows-matrix-and-how-is-it-useful/

Swart, N. d. (2010). Belanghebbenden bij het systeem. In N. d. Swart, *Handboek Requirement* (p. 173). Delft: Eburon Business.

## Bijlage 14: Afstudeerplan

|  |  |
| --- | --- |
| **Afstudeerblok** | 2021-2.2 |
| **Startdatum uitvoering afstudeeropdracht** | **10-05-2021** |
| **Inleverdatum afstudeerdossier volgens jaarrooster** | **8-10-2021** |

|  |  |
| --- | --- |
| **Studentnummer** | 15042960 |
| **Achternaam** | Unal |
| **Voorletters** | N |
| **Roepnaam** | Necati |
| **Adres** | Veluweplein 32 |
| **Postcode** | 2571GB |
| **Woonplaats** | Den Haag |
| **Telefoonnummer** | 0646396212 |
| **Mobiel nummer** |  |
| **Privé emailadres** | Necati.unal01@gmail.com |
|  |  |
| **Differentiatie** | SE |
| **Afstudeerprogramma** | SE |
| **Domein\*** | SE |
| **Locatie** | **Den Haag** |
| **Variant** | Voltijd |

\**geef hier aan binnen welk domein jouw opdracht valt, m.a.w. wat voor type opdracht het is*

|  |  |
| --- | --- |
| **Naam studieloopbaanbegeleider** | B.M. Derks |
| **Naam begeleidend examinator** | A. Wieman |
| **Naam expert examinator** | J.J. van der Hoek |

**Bedrijf**

|  |  |
| --- | --- |
| **Naam** | Jongens van Techniek |
| **Afdeling** | Engineering |
| **Bezoekadres** | Wilhemina van Pruisenweg 35 |
| **Postcode** | 2595 AN |
| **Plaats** | Den Haag |
| **Telefoonnummer** | +31 6 39 21 25 04 |
| **URL** | Jongensvantechniek.nl |

**Opdrachtgever**

|  |  |
| --- | --- |
| **Achternaam** | Dhr Peter |
| **Voorletters/ Voornaam** | P |
| **Titel/ Opleiding** | CEO |
| **Functie** | Founder |
| **Telefoonnummer (werk)** | 06 14345853 |
| **Emailadres (werk)** | peter@jongensvantechniek.nl |
| *is de opdrachtgever ook de bedrijfsmentor?* | Nee |

**Bedrijfsmentor**

|  |  |
| --- | --- |
| **Achternaam** | Bosman |
| **Voorletters/ Voornaam** | Jonathan |
| **Titel/ Opleiding** | MSc |
| **Functie** | Projectmanager |
| **Telefoonnummer (werk)** | 070 2060166 |
| **Emailadres (werk)** | jonathan@jongensvantechniek.nl |

**Titel afstudeeropdracht**

Onderzoeken naar en implementeren van monitoring van opgeleverde applicaties.

**Opdrachtomschrijving**

**Bedrijf & Probleemdomein**

Jongens van Techniek is opgericht door Dhr. Peter Hofman. Jongens van Techniek is een commercieel software engineering bedrijf. Zij ontwikkelen oplossingen voor klanten met specifieke wensen en doen ondersteuning van deze oplossingen. Hun klantenbestand bevat onder andere Ministerie van Economische zaken, Vodafone. Ze zijn partner van Microsoft, Amazon webservices en Transip.

Jongens van techniek is een bedrijf met 15 tot 20 medewerkers waaronder 12 engineers waarmee ik dagelijks mee in contact zal komen. Dagelijkse activiteit houdt in ontwikkelen van software-, cloud- en dataoplossingen en het bieden van support aan deze klanten.

**Probleemdomein**

Het probleem start zodra een (gedeelte) oplossing is opgeleverd aan de klant. De software wordt in gebruik genomen bij de klant en heeft Jongens van Techniek geen controle meer over de software. De klant kan dan tegen problemen aanlopen. Deze problemen zijn zoals het niet beschikbaar zijn van de applicatie (downtime) of een volle database.

Jongens van Techniek heeft een supportdesk waar de klanten terecht kunnen. Het probleem wordt bij de support desk behandeld en zo een oplossing voor gemaakt. Hiermee wordt er reactief gereageerd bij een probleem. Dit betekent dat er alleen gereageerd wordt, zodra een probleem heeft afgespeeld. Voor de klant is dit vervelend. Het probleem kan groot zijn, waardoor een volledig systeem plat komt te staan en daarbij ook de workflow. Dit kan in vele gevallen de klant geld, moeite en energie kosten. Ze zullen tijd reserveren om contact op te nemen met de supportdesk om vervolgens het probleem op te lossen.  
  
Jongens van Techniek wil ervoor zorgen dat er preventief gemonitord en geacteerd kan worden. Een voorbeeld is een volle database of een server die niet bereikbaar is. Dit kan tegengegaan worden door het geheugen van de database te monitoren en aan te vullen zodra deze een grens heeft bereikt. Op deze manier wordt de uptime gehandhaafd, heeft de klant controle over de software en kan vroegtijdig ingespeeld worden op problemen die kunnen optreden.

**Aanleiding & probleemstelling**

Klanten die een software in gebruik nemen zijn niet op de hoogte van problemen die optreden tijdens de levensduur van de applicatie. Problemen zoals volle servers, beschikbaarheidsproblemen, nieuwe software en nieuwe updates/versies kunnen ertoe doen dat klanten de opgeleverde oplossing niet meer kunnen gebruiken. Al deze dingen kunnen, als er geen onderhoud gepleegd wordt, tot problemen leiden zoals het (deels) niet werken van de opgeleverde oplossing.

Jongens van Techniek heeft hiervoor de support desk in het leven geroepen. Alhoewel het fijn is voor de klant om ergens heen te kunnen voor hulp wanneer een probleem optreedt, is het beter dat er überhaupt geen problemen op zullen treden. Nu het applicatielandschap is vergroot is het een grote taak voor de support desk om al deze problemen na te lopen en kunnen er wachttijden ontstaan.   
  
Daarom wil Jongens van Techniek bij hun onderhoudsabonnement een monitoringsservice aanbieden aan hun klanten. Het project draagt aan bij het dagelijks gebruik van de software van de klant. Het project zal een preventieve oplossing moeten vormen voor de problemen die kunnen ontstaan bij het in gebruik nemen van de applicatie. Qua complexiteit kan dit vrij snel oplopen doordat er meerdere applicaties gemonitord moeten worden. Doordat abonnementen afgesloten en afgezegd kunnen worden, zal de monitoring flexibel moeten kunnen zijn.

**Doelstelling**

Het doel van het project is een onderzoek doen naar de mogelijkheden om binnen een Cloud omgeving monitoring toe te passen. De wens van Jongens van Techniek is om een cloudapplicatie te kunnen leveren welk makkelijk te configureren is om applicaties toe te voegen en los te koppelen.

Verder zal tijdens het project een implementatie voor een monitoringssoftware worden gemaakt. Hier wordt gekeken welke elementen van een website of applicatie gemonitord moet worden. De problemen die bij klanten voordoen zullen eerst geïnventariseerd worden. Vervolgens zullen deze problemen aan de hand van een prioriteringslijst opgelost worden binnen de applicatie. Ook worden er meldingen ingebouwd aan de hand van vooraf gedefinieerde stappen om automatisch een bericht te sturen of een automatische handeling waarbij problemen worden vermeden.

Aan het eind van het project doel ik op het leveren van een ‘minimal viable product’ (MVP). Dit product zal een start zijn voor een monitoringssoftware gebouwd op basis het onderzoek en requirements. Hoofdvraag binnen het onderzoek is ‘Welke monitoringsmogelijkheden zijn er beschikbaar voor het monitoren van verschillende applicaties binnen Jongens van Techniek.’

**Concrete werkzaamheden en producten.**

1. Plan van aanpak (PVA) maken. In het plan van aanpak komen de volgende informatie terug: bedrijfsomschrijving, probleemstelling, doelstelling, resultaat, scope van de opdracht en op te leveren (tussen)producten. (5 dagen)
2. De business cases verzamelen door interviews te houden en requirements op te stellen (3 dagen)
3. Een onderzoek doen naar de mogelijkheden binnen de monitoringswereld en een eventuele cloudoplossing. De vraag: “Welke cloudoplossing kunnen we gebruiken om een monitoringssysteem toe te passen op de reeds afgemaakte oplossingen voor klanten”. (10 dagen)
4. Aan de hand van de requirements en het onderzoek een oplossing implementeren (30 dagen)
5. De kwaliteit van de geïmplementeerde functionaliteiten behouden door deze te testen (8 dagen)
6. Eindverslag opstellen van de afstudeerperiode (15 dagen)

**Beroepstaken**

De beroepstaken zijn gekoppeld aan de werkzaamheden opgesomd in hoofdstuk 4 Concrete werkzaamheden en producten

**A1 Analyseren probleemdomein & opstellen probleemstelling**

**Omschrijving**:

De student brengt helder in kaart waar, wanneer, voor wie, waarom en welke problemen zich voordoen, wat de uitgangssituatie is, welke risico’s er mogelijk zijn en welke (f)actoren daarbij een rol spelen.

**Manier van aantonen**:

Er zal eerst een analyse plaatsvinden aan de hand van een plan van aanpak (PVA). Dit zal helpen om een fundament te vormen voor het project. Hier kunnen we de problemen vaststellen die in de huidige situatie voordoen en een gewenste situatie schetsen. In het PVA komen vervolgens een rooster van looptijden voor elke tussenproduct dat gedurende afstudeerperiode ontwikkeld zal worden. Dit zijn alle producten die door de afstudeerder zal worden gemaakt zoals onderzoekdocumentatie, interviews, testrapportages en cases en het geïmplementeerde product.

**Gc Kritisch en methodisch werken**

**Omschrijving**:

De student hanteert hoge kwaliteitsnormen voor anderen en zichzelf en stelt zichzelf vragen. Op zoek naar antwoorden hanteert de student gestandaardiseerde methoden/technieken/processen, en relevante en adequate argumentatie.

**Manier van aantonen:**

Gedurende afstudeerperiode zal nieuwe technieken geleerd moeten worden om zo bij het eindproduct te komen. Dit betekend goede technieken gebruiken om te zorgen dat het probleem zo efficiënt mogelijk opgelost zal worden. Hiervoor zal kritische keuzes gemaakt moeten worden om ervoor te zorgen dat een oplossing goed past. Deze keuzes zullen methodisch opgebouwd worden door bijvoorbeeld een onderzoek of een interview.

**Gf Leren leren**

**Omschrijving**:

De student is in staat het eigen leerproces vorm te geven waarbij voorbereid wordt op life-long-learning.

**Manier van aantonen:**

Tijdens de stage zal ik gebruik maken van nieuwe technieken. Verder is voor mij het Cloud omgeving nog onbekend buiten dit theoretisch behandeld is. Om deze omgeving te kunnen leren zal gebruik gemaakt worden van goede bronnen om tot een eindproduct te komen. Dit zal ik doen door middel van bronnen en hulp van collega’s te raadplegen.

**C1 Ontwerpen software**

**Omschrijving**:

De student maakt op basis van analyses en best practices gemotiveerde keuzes voor de opbouw van en softwareproject. Zowel op micro (binnen modules), meso (tussen modules) als macro (module selectie) niveau.

**Manier van aantonen:**

Het product die tijdens de stage geleverd zal worden moet eerst ontworpen worden. De software die ontwikkeld gaat worden moet eerst ontworpen worden. Aan de hand van juiste richtlijnen zal ik eerst een architectuur ontwerpen. De architectuur zal ik maken aan de hand van een architectuur ontwikkelmethode die het beste past bij het project.

**A2 Informatie vergaren, analyseren & verwerken**

**Omschrijving:**

De student kan methodisch informatie en/of data verzamelen waarbij gemotiveerde keuzes worden gemaakt op basis van relevantie en betrouwbaarheid. De informatie wordt vervolgens geanalyseerd en verwerkt tot (onderdeel van) een eindproduct in de vorm van een literatuurreview, requirementsoverzicht of data-analyse.

**Manier van aantonen:**

Ik zal tijdens mijn project eerst boven water halen welke functionaliteiten de stakeholders willen hebben. Door middel van interviews met collega’s en stakeholders de juiste requirements achterhalen en prioriteren middels een prioriteringsmethode.

**D1 Realiseren software**

**Omschrijving:**

De student ontwikkelt een onderbouwd softwareproduct op basis van een ontwerp. Hierbij wordt gebruik gemaakt van programmeer-, opmaak- en scripttalen. Er kan gebruik gemaakt worden van bestaande libraries en frameworks of begonnen worden 'from scratch'.

**Manier van aantonen:**

Deze competentie zal bewezen worden door implementatie van de software die gemaakt zal worden door de afstudeerder. Onder deze competentie valt code waarin geen “bad smells” in zitten.

**D2 Testen en evalueren**

**Omschrijving:**

De student toetst of een ontwikkeld ICT-product of beleid voldoet aan de opgestelde (Niet-)functionele requirements. Hierbij wordt planmatig en volgens best practices gewerkt en waar mogelijk wordt het proces geautomatiseerd. Er wordt aandacht besteed aan het formaliseren van testcases en evaluatiecriteria.

**Manier van aantonen:**

Tijdens implementatie van het product willen we continu weten of we niks anders stuk gemaakt hebben. Hiervoor willen we gaan testen zodat de applicatie geen onverwachte output geeft. De applicatie moet inmiddels integer zijn doordat het een support rol invult voor andere applicaties. Door het testen willen we de kwaliteit behouden van de applicatie. Verder zal aan de hand van peer reviews gecontroleerd worden of de code goed is.

**Bibliografie**

Amazon. (2021, 01 10). *Amazon Documentation*. Opgehaald van AWS Documentation: https://docs.aws.amazon.com/

Baarda, B. (2014). *Dit is onderzoek.* Nederland: Noordhoff.

Bass, L. (2012). *Software Architecture in Practice.* Pearson Education.

Microsoft. (2021, 01 10). *Microsoft Azure*. Opgehaald van Microsoft Docs: https://docs.microsoft.com/nl-nl/learn/azure/

Randen, H. J. (2015). *Aan de slag met Scrum.* Nederland: Academic service.

*.*

## Bijlage 15: Plan van aanpak

MonitoringApp

**Plan van aanpak**

Auteur: Necati Unal

Datum 21-04-2021

Inhoudsopgave

[*2* Aanleiding 209](#_Toc85155081)

[3 Probleemstelling 209](#_Toc85155082)

[4 Doelstelling van het project 210](#_Toc85155083)

[5 Risicoanalyse 210](#_Toc85155084)

[5.1 Technische risico 210](#_Toc85155085)

[5.2 Organisatorisch 210](#_Toc85155086)

[5.3 Functioneel 211](#_Toc85155087)

[6 Aanpak 212](#_Toc85155088)

[7 Architectuur 212](#_Toc85155089)

[8 Testplan 213](#_Toc85155090)

[9 Ontwikkeling 213](#_Toc85155091)

[10 Requirements 214](#_Toc85155092)

[11 DevOps 214](#_Toc85155093)

[12 Planning 214](#_Toc85155094)

# Aanleiding

Jongens van Techniek is opgericht door Dhr. Peter Hofman. Het is een commercieel software engineering bedrijf die zich begeeft op het ontwikkelen van oplossingen voor klanten met specifieke wensen. Daarnaast bieden ze ondersteuning aan voor deze oplossingen. Het klantenbestand van Jongens van Techniek bevat onder andere Ministerie van Economische zaken, Vodafone en Open University.

Na levering van de oplossing kan de klant ervoor kiezen om een ondersteuningspakket aan te gaan. Hiermee is het mogelijk voor de klant om contact op te nemen met de supportdesk en om hulp te vragen bij problemen. Vervolgens wordt het probleem onderzocht door de support desk en wordt er een poging gedaan tot oplossen

# Probleemstelling

Het probleem start zodra een (gedeelte) oplossing is opgeleverd aan de klant. De software wordt in gebruik genomen bij de klant en heeft Jongens van Techniek geen zicht meer over de software. De klant kan dan tegen problemen aanlopen. Deze problemen zijn zoals het niet beschikbaar zijn van de applicatie (downtime) of een volle database (capaciteitsproblemen). Klanten die software in gebruik nemen zijn niet op de hoogte van problemen die optreden tijdens de levensduur van de applicatie. De eerder benoemde problemen kunnen ertoe doen dat klanten de opgeleverde oplossing niet meer kunnen gebruiken.

Jongens van Techniek heeft een supportdesk waar klanten terecht kunnen. Het probleem wordt bij de support desk behandeld en zo een oplossing voor gemaakt. Hiermee wordt er reactief gereageerd bij een probleem. Dit betekent dat er alleen gereageerd wordt, zodra een probleem heeft afgespeeld en is gemeld bij de support desk. Het probleem kan groot zijn, waardoor een systeem plat komt te liggen en daarbij ook de workflow. Om dit op te lossen kost dit de klant geld, tijd en energie. Daarnaast zal er mankracht gereserveerd moeten worden om het probleem op te lossen. Alhoewel het fijn is voor de klant om ergens heen te kunnen voor hulp wanneer een probleem optreedt, is het beter dat er überhaupt geen problemen optreden. Nu het applicatielandschap is vergroot, is het een grote taak voor de support desk om al deze problemen na te lopen en op te lossen.

Jongens van Techniek wil zorgen dat er preventief gemonitord en geacteerd wordt. Een voorbeeld is een volle database. Dit kan tegengegaan worden door het geheugen van de database te monitoren en aan te vullen zodra deze een grens heeft bereikt. Op deze manier wordt de uptime gehandhaafd en heeft de klant controle en kan vroegtijdig inspelen op problemen die zullen optreden.

Daarom wil Jongens van Techniek bij hun onderhoudsabonnement een monitoringsservice aanbieden aan hun klanten. Dit is een preventieve oplossing voor de eerder benoemde problemen die voorkomen. Qua complexiteit kan dit vrij snel oplopen doordat er meerdere applicaties gemonitord moeten worden. Dit resulteert in de wens dat de monitoring flexibel ingezet kan worden.

Alle bovengenoemde problemen opgesomd:

1. Geen zicht meer op de software wanneer deze online gaat.
2. Problemen kunnen niet tegengehouden worden. Oplossing vindt pas plaats nadat een probleem is afgespeeld.
3. Meerdere projecten betekend meer problemen die kunnen voordoen.

# Doelstelling van het project

Het doel van het project is om binnen de afstudeerperiode een onderzoek te doen naar de manier van monitoring binnen de Cloud wereld en daarna een Minimal Viable Product (MVP) op te leveren. De oplossing zal moeten voldoen aan de vooraf gedefinieerde hoog geprioriteerde requirements en als er tijd voor over is de lager geprioriteerde requirements. Daarnaast is het doel om de monitoringsapplicatie te koppelen aan een applicatie waarop de monitoring is toegepast.

# Risicoanalyse

Tijdens het project zijn er een aantal risicofactoren die invloed kunnen hebben tot het project. Met behulp van een risicoanalyse kunnen we de vooraf gesignaleerde risico’s beschrijven met de daarbij de maatregel die genomen moeten worden tegen het probleem.

## Technische risico

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nr #. | Bedreiging | Kans (1 – 5) | Impact (1-5) | Risico (K \* I) |
| 1 | Beperkte cloud kennis | 2 | 5 | 10 |
| 2 | Verlies van code door slechte code versioning. | 1 | 5 | 3 |
| 3 | Geen toegang hebben tot een andere applicatie voor monitoring | 3 | 5 | 15 |

**Tegenmaatregel #1:**

Bronnen opzoeken die te maken hebben met Cloudoplossingen voor Monitoringssystemen.

**Tegenmaatregel #2:**

Gebruik maken van Git zodat code versioning te volgen is. Dit zal ook goed bijgehouden moeten worden door de ontwikkelaar zelf zodat code local en in de cloud up to date is.

**Tegenmaatregel #3:**

Het monitoringssysteem zal getest moeten worden voor functionaliteit door deze te koppelen op een andere applicatie. Dit kan belemmerd worden door geen toegang hebben tot een andere applicatie. Het is mogelijk om gebruik te maken van een dummy applicatie zodat de monitoring op gebouwd kan worden.

## Organisatorisch

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nr #. | Bedreiging | Kans (1 – 5) | Impact (1-5) | Risico (K \* I) |
| 1 | Beperking op het gebied van Business logic | 5 | 5 | 25 |

**Tegenmaatregel #1:**

Deze bedreiging kan tegengegaan worden door het interviewen van de beschikbare stakeholders binnen Jongens van Techniek. Door interviews af te leggen en de juiste informatie te vergaren is het mogelijk om de correcte requirements op te stellen.

## Functioneel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nr #. | Bedreiging | Kans (1 – 5) | Impact (1-5) | Risico (K \* I) |
| 1 | Onderschatten benodigde tijd voor stories | 2 | 3 | 6 |
| 2 | Storen van de gemonitorde applicatie | 2 | 5 | 10 |

**Tegenmaatregel #1:**

Stories klein houden en volgens de User story regels. Deze goed formuleren zodat er geen miscommunicatie ontstaat en de benodigde werk ervoor duidelijk in taken verdeeld zijn.

**Tegenmaatregel #2:**

Zorgvuldig testen zodat er geen belemmering ontstaat tussen de monitoringsysteem en de desbetreffende applicatie.

# Aanpak

Tijdens het project zal gebruik gemaakt worden van een ontwikkelingsmethode. Hier kan de Waterval of de Agile methode voor gekozen worden.

Waterval-methode bestaat al sinds 1970. Waterval-methode is een framework waarin ontwikkeling in stappen volgt naar het eindproduct. De stappen van de waterval methode (TechRepublic, 2006) zijn als volgt:

1. Requirements analysis
2. Design
3. Implementation
4. Testing
5. Installation
6. Maintenance

De voordelen van watervalmethodiek is dat het discipline eist. Er is een start en een eind voor elke stap. Start en eind van een fase is duidelijk aangegeven. Door focus op requirements en design te leggen voordat ontwikkeling start zorgt voor minimale verspilling van tijd en moeite. Hier worden alle requirements met de klant besproken om zo een goed mogelijk requirementsroverzicht te kunnen creëren waaruit verder ontwikkeld kan worden.

Doordat waterval vereist dat er in stappen gewerkt wordt is het lastig om terug te gaan naar de requirements in het geval deze later veranderen. Binnen waterval is het lastig om terug te gaan naar de vorige stap. Het is in dit geval lastig om requirement-wijzigingen toe te passen als de klant zijn gedachte heeft veranderd.

Om deze probleem op te kunnen lossen is het Agile framework in het leven geroepen. Onder Agile vallen meerdere soorten ontwikkelmethodes waaronder Scrum. Uit een onderzoek (VitalityChicago, 2018) is gebleken dat traditionele ontwikkelingsmethodes zoals waterval twee keer minder succesvol waren in het slagen van het project vergeleken met een Agile ontwikkelmethode. Scrum is één van de vele Agile ontwikkelmethodes die ingezet kan worden binnen een team. Het zorgt ervoor dat de ontwikkeling van software op een effectieve en flexibele manier verloopt. Scrum heeft feedback loops binnen elke stap van het project en er kan terug op gekomen worden zodra wijzigingen voordoen. Dit zorgt voor dat aanpassingen vanuit de klant binnen het ontwikkelproces genomen kan worden. Een ander voordeel van Scrum is dat de klant volledig betrokken raakt bij het ontwikkelproces en kan hij het ontwikkelproces meemaken en feedback leveren wanneer dat nodig is.

# Architectuur

Om de architectuur te kunnen ontwikkelen voor het monitoringssysteem is eerst van belang dat kennis opgedaan zal worden op de bestaande architectuur die gehandhaafd wordt binnen Jongens van Techniek. Vervolgens zal er aan de hand van de requirements onderzocht worden welke architectuur ontwikkelmethode het beste past bij het project.

Met de ontwikkelmethode zal de architectuur worden ontwikkeld. Deze architectuur zal de basis vormen voor het project en zal rest van de afstudeerperiode ontwikkeld worden.

# Testplan

Om de kwaliteit van de code en het product te kunnen garanderen zal er vooraf een testplan worden ontwikkeld. In dit document zal je maatregelen treffen die gebruikt worden om de kwaliteit van het product te waarborgen. Na ontwikkeling van een user story zal deze getest moeten worden aan de hand van de testmethode dat is beschreven.

# Ontwikkeling

Ontwikkeling met scrum betekend dat progressie incrementeel zal vorderen. Iedere twee weken, oftewel een sprint, wordt een aantal user stories binnen een sprint genomen voor ontwikkeling. Deze stories zijn gekozen aan de hand van de prioriteringsmethode die zal gehanteerd worden tijdens het project.

Ontwikkelen van het platform wordt gedaan op Cloud basis. Hiervoor zal Azure gebruikt worden doordat het bedrijf daar een partnerschap mee heeft. Als laatst wordt het project ontwikkeld in C#. Doordat niet alle functionaliteiten van de cloud-platform duidelijk zijn, zal onderzocht moeten worden welke functies beschikbaar zijn. Mocht budget een rol spelen binnen deze beslissing dan zal hierover ook beslist moeten worden door de bedrijfsmentor.

# Requirements

Een ander kritiek onderdeel is het vaststellen van de requirements. Deze worden geleverd door mijn mentor (en als het nodig is andere collega’s). Samen met de context en opdrachtomschrijving die is geleverd wordt er een prioriteitenlijst gemaakt. Daarvan uit kan een huidige situatie, de wens en eisen van de stakeholders boven water worden gehaald. De prio-lijst zal opgesteld worden op basis van de MoSCoW methode. Om te voorkomen dat alle user stories op de hoogste prio beland, zal samen met de stakeholder gereviewd worden om de belangrijke items hoger te prioriteren en de nice-to-haves een minder belangrijke prioriteit krijgen. De stories worden beschreven aan de hand van de User Story regels (As a < type of user >, I want < some goal > so that < some reason >).

# DevOps

Voor versiebeheer van de applicatie zal DevOps gebruikt worden. DevOps geeft ons de mogelijkheid om de deployment en de infrastructuur te kunnen realiseren. DevOps geeft daarnaast ons de mogelijkheid om de best practices te kunnen gebruiken zoals Continious integration en Continious delivery. Voor Devops zal gebruik gemaakt worden van Azure. Dit is een organisatorisch limitatie doordat Jongens van Techniek in het verleden projecten voor banken en gemeenten heeft gerealiseerd. Dit zijn gevoelige gegevens die men in Nederland wilt houden. Door Azure te gebruiken is ervoor te zorgen dat deze gegevens Europa niet verlaat.

# Planning

In de bijlage vind je de planning voor de komende werkzaamheden. Deze zullen een plaats krijgen in de rooster volgens de indruk die ik heb gekregen van de moeilijkheidsgraad die het project heeft. Voor het project is er gebruik gemaakt van een Sprint 0 om een zo goed mogelijk beeld te krijgen van het systeem. Verder ook Sprint 0 gebruiken om de requirements vast te stellen voor het project. Elke sprint zal ervoor gezorgd worden dat de stories goed beschreven zijn en dat er tickets aangemaakt zijn voor de desbetreffende stories. Voor mezelf zal ik wel gebruik maken van de standaard user story structuur omdat dit voor mij duidelijker is. Alhoewel daarnaast zal ook dat de omschrijving voor de user story van pas komen om de user story te kunnen beschrijven.