TAM / TAM
part of DEPT

Afstuderen Monitoring.

Lennart Timmers

1 juni 2017

Versiebeheer

Versie	Datum	Opmerkingen
0.1	9-2-17	Eerste opmaak document.
0.2	15-2-17	Bijwerken bestaande kopjes met nieuwe informatie.
0.3	24-2-17	Hoofdstuk 5 toegevoegd.
0.4	27-3-17	Hoofdstuk 6 toegevoegd.
0.5	31-3-17	Hoofdstuk 1, 2 en 3 bijgewerkt.
0.6	5-4-17	Hoofdstuk 5.1, 5.2, 5.3, 5.4 en 5.5 bijgewerkt.
0.7	10-4-17	Toevoegen van hoofdstuk 7 en 8.
0.8	24-4-17	Updaten document met Sprints en uitgebreidere uitleg van keuze ontwikkelmethodiek.
0.9	26-4-17	Toevoegen plan van aanpak hoofdstuk.
1.0	2-5-17	Bijwerken algehele document op verschillende plekken. Sprint reflecties toegevoegd. Start sprint 5 gemaakt.
1.1	3-5-17	Plan van aanpak herschreven.
1.2	8-5-17	Sprint 5 requirements hoofdstuk toegevoegd.
1.3	9-5-17	Document doorlopen en in het algemeen wijzigingen doorgevoerd.
1.4	19-5-17	Feedback Leon verwerkt.
1.5	22-5-17	Document doorlopen en in het algemeen wijzigingen doorgevoerd.
1.6	23-5-17	Sprint 5 ontwerp geschreven.
1.7	24-5-17	Sprint 4 en 5 wijzigingen in gemaakt.
1.8	25-5-17	Sprint 5 ontwerp bijgewerkt en bouwen geschreven.
1.9	26-5-17	Sprint 5 bouwen bijgewerkt.
2.0	29-5-17	Begonnen met de reflectie, spellingscontrole van Word uitgevoerd.
2.1	30-5-17	“Sprint … in het kort” toegevoegd bij de sprints. Hoofdstuk 12 afgemaakt.
2.2	31-5-17	Commentaar van Niek verwerkt.
2.3	1-6-17	Laatste check over het gehele document. Bijlagen toegevoegd en figuren goed verwezen. Kleine wijzigingen in het gehele document doorgevoerd.

Referaat

Dit is de afstudeerscriptie van Lennart Timmers geschreven in het kader van de opleiding Informatica aan De Haagse Hogeschool. In dit rapport vindt u een beschrijving van de processen die ik doorlopen heb, de nieuwe kennis die ik heb vergaard en de beslissingen die ik heb genomen en de daarbij horende motivatie.

Descriptoren:

- Afstuderen
- Plan van Aanpak
- Scrum
- .NET
- Nagios
- Monitoringtooling
- Architectuur
- Requirements
- Toolselectie

Voorwoord

Dit verslag is geschreven voor de afronding van mijn opleiding Informatica aan De Haagse Hogeschool. Deze opdracht is bij TamTam part of DEPT uitgevoerd. De opdracht die ik heb uitgevoerd ging over de monitoring binnen TamTam. Doel was het inzichtelijk maken van de monitoring processen, deze tegen de gestelde ambities met betrekking tot hosting aan te houden en op basis van conclusies en aanbevelingen nieuwe processen in kaart te brengen.

Dit document is bedoeld voor de examinatoren om een beeld te vormen van mijn werkzaamheden binnen TamTam. In dit document bevindt zich geen vertrouwelijke informatie.

Mijn afstuderen had niet goed kunnen verlopen zonder de ondersteuning van een aantal personen. Dit dankwoord is speciaal aan hen gericht voor alle moeite die zij voor mij gedaan hebben.

Ten eerste wil ik graag iedereen waarmee ik binnen TamTam gewerkt heb bedanken voor het bieden van een fijne stageomgeving. Een aantal van deze personen wil ik graag specifiek bedanken.

Als eerste Niek van Raaij. Dankzij hem ben ik in contact gekomen met TamTam en ben ik deze opdracht gaan doen. Verder was hij tijdens mijn stage mijn buddy, waar ik altijd vragen aan kon stellen als ik ergens niet uit kwam.

Verder wil ik graag Leon van Rees bedanken. Hij was degene die mij deze opdracht heeft aangeboden. Ook was hij mijn stagebegeleider gedurende dit traject. Hij was vaak druk, maar maakte altijd wel wat tijd voor mij vrij als ik hem nodig had.

Ook wil ik mijn begeleidend docent Nanny Jacobs bedanken. Niet alleen voor de feedback op dit rapport, maar ook voor alle hulp die zij mij heeft geboden gedurende de hele opleiding. In een opleiding waar veel problemen waren was Nanny altijd de redder in nood.

Inhoudsopgave.

Versiebeheer **2**

Referaat **3**

Voorwoord **4**

1. Inleiding **8**

2. Organisatiebeschrijving **9**

2.1 Ontstaan, Mentaliteit & Missie 9

2.2 Werkomgeving 10

3. De opdracht **11**

3.1 Probleemstelling 11

3.2 Doelstelling 12

3.3 Beoogd resultaat 12

4. Tools en methodes **13**

4.1 Ontwikkelmethodiek 13

4.2 Tools 17

5. Plan van aanpak **18**

5.1 Producten & aanpak 18

5.2 Aanvullende opdracht producten & aanpak 20

6. Sprint 1 ..... **21**

6.1 Requirements verzamelen 21

6.2 Intake huidige situatie 24

6.3 Reflectie sprint 1 27

6.4 Sprint 1 in het kort 27

7. Sprint 2 ..... **28**

7.1 Inventarisatie monitoringtools 28

7.2 Monitoring bij grote IT-bedrijven 29

7.3 Toolselectie nieuwe situatie 31

7.3.1 Monitoringselectie 32

7.3.2 Proof of concept Statuscake 34

7.3.3 Alertingselectie 35

7.3.4 Advies 36

7.4 Self-healing ... 37
7.5 Reflectie sprint 2 ... 38
7.6 Sprint 2 in het kort ... 38

8. Sprint 3 ... 39
8.1 Architectuur ... 39
8.2 Advies presentatie ... 41
8.3 Ontwerp klassendiagram TicketCreator ... 42
8.4 Reflectie sprint 3 ... 43
8.5 Sprint 3 in het kort ... 43

9. Sprint 4 ... 44
9.1 Nagios ... 44
9.2 Pingdom/UptimeRobot testen overzetten ... 46
9.3 Azure en AWS testen integreren ... 46
9.4 Nieuwe OpsGenie ... 47
9.5 Bouwen TicketCreator ... 48
9.5.1 Verkrijgen OpsGenie data ... 48
9.5.2 Aanmaken JIRA-ticket ... 49
9.6 Testen TicketCreator ... 51
9.6.1 Testen aanmaken tickets ... 51
9.6.2 Testen vergelijking openstaand ticket ... 53
9.7 Ontwerpen processen ... 54
9.7.1 Nieuw project ... 54
9.7.2 Out of serviceproject ... 56
9.7.3 Handelswijze OpsGenie ... 57
9.7.4 Presenteren processen ... 57
9.8 Eindproduct ... 58
9.9 Reflectie sprint 4 ... 60
9.10 Sprint 4 in het kort ... 60

10. Bepalen aanvullende opdracht ... 61
10.1 Probleemstelling ... 61
10.2 Doelstelling ... 61
10.3 Beoogd resultaat ... 61

11. Sprint 5 ... 63
11.1 Verzamelen requirements ... 63
11.2 Ontwerpen webapplicatie ... 64
11.3 Bouwen devops.tamtam.nl ... 68

11.4 Testen 70
11.5 Eindproduct devops.tamtam.nl 71
11.6 Reflectie sprint 5 73
11.7 Sprint 5 in het kort 73

12. Reflectie 74
12.1 Proces 74
12.2 Producten 74
12.2.1 Nieuwe monitoringsituatie 75
12.2.2 Devops.tamtam.nl 77
12.3 Beroepstaken 77

13. Bijlages 79
A Afstudeerplan 79
B Plan van aanpak 79
C Uitgewerkte interviews 79
D Requirementsdocument 79
E Toolselectie monitoring 79
F Statuscake proof of concept 79
G Architectuurdocument 79
H Testdocument 79
I Requirementsdocument devops.tamtam.nl 79

1. Inleiding

Ter afronding van mijn afstuderen bij TamTam is dit afstudeerverslag geschreven. Dit afstuderen heeft plaatsgevonden bij TamTam van 2 februari 2017 tot 3 Juni 2017. In dit document vertel ik over mijn werkzaamheden gedurende de afstudeerperiode bij TamTam.

Doelstelling
Dit verslag heeft als doel om de examinatoren inzicht te geven in het proces dat ik heb doorlopen gedurende mijn tijd bij TamTam. Aan de hand van dit rapport zullen zij een beeld kunnen vormen van mijn afstuderen en hierop gebaseerd een beoordeling geven voor mijn prestaties.

Leeswijzer
Dit verslag is onderverdeeld in de sprints die ik heb doorlopen bij TamTam. Per sprint leg ik uit wat ik in die sprint heb gedaan.

In de eerste sprint behandel ik het verzamelen van informatie en de intake van de huidige situatie. Deze opstartsprint was één week lang waardoor ik verder geen tijd had om andere werkzaamheden uit te voeren.

In de tweede sprint heb ik onderzoek gedaan naar de aanwezige monitoring tools op de markt en wat deze te bieden hebben. Vervolgens heb ik gekeken naar monitoring tools die bij bedrijven met een groot ICT-landschap worden gebruikt. Ik kijk of deze overeenkomen met de tools die ik had gevonden en kijk naar de functionaliteiten van de tools die ik nog niet kende.

In de derde sprint ben ik de nieuwe monitoring situatie gaan vormgeven. Ik heb een architectuur gemaakt voor hoe ik de nieuwe situatie voor mij zag en deze vervolgens gepresenteerd aan de stakeholders. Daarnaast heb ik ook een ontwerp gemaakt voor de TicketCreator die automatisch tickets kan aanmaken wanneer een OpsGenie alert langer dan een uur open heeft gestaan.

In de vierde sprint heb ik de nieuwe situatie geïmplementeerd. Om dit te doen moest ik eerst enige kennis over Nagios op doen. Ik heb toen een testserver opgesteld waar ik een aantal simpele test in Nagios heb leren aanmaken. Vervolgens heb ik een nieuwe OpsGenie account aangemaakt. Deze account heb ik geconfigureerd zodat deze op de juiste wijze alertvers uitstuurt naar de on-call personen. Ook heb ik de TicketCreator applicatie gebouwd in deze sprint.

In de vijfde en tevens laatste sprint heb ik gewerkt aan een aanvullende opdracht. De behoefte voor deze opdracht was ontstaan door de processen die ik had ontworpen (hoofdstuk 9.7). In deze sprint verzamel ik requirements, ontwerp ik een systeem en bouw ik dit systeem genaamd devops.tamtam.nl. Met dit systeem kunnen gebruikers automatisch nieuwe projecten en out of serviceprojecten aanmaken. De website maakt dan automatisch een taak met een aantal subtaken aan op het DevOps hosting bord in JIRA.

2. Organisatiebeschrijving

2.1 Ontstaan, Mentaliteit & Missie

TamTam is opgericht door Bart en Paul Manuel in 1996, voordat het woord “punt-com” was uitgevonden. Het doel was om een online catalogus te maken voor de technische groothandel. Dit plan is echter nooit afgemaakt en de plannen zijn volledig veranderd. TamTam begon met het maken van websites.

TamTam ontwerpt en bouwt deze websites voor haar klanten. Deze websites worden ontworpen door award winnende designers en gebouwd door enthousiaste programmeurs. Een deel van de klanten besteedt ook de hosting uit aan TamTam. Dit wordt vervolgens ingericht en beheerd door afdeling DevOps, waar ik gedurende mijn stage heb gewerkt.

De naam TamTam staat voor communicatie. TamTam onderscheidt zichzelf door goed te luisteren naar klanten, de juiste vragen te stellen, echte oplossingen te bedenken en deze ook goed te implementeren. En daar is TamTam heel trots op. We luisteren echter niet alleen naar onze klanten, maar ook naar elkaar. Hier wordt niet gelet op rangorde, de beste ideeën zijn ontstaan dankzij input van jongere collega’s.

TamTam gelooft in het geven van vrijheid. Ze gaan ervan uit dat wanneer je gezamenlijk besluiten neemt over doelstellingen, dat mensen hier enthousiast verantwoording voor nemen. Dit wordt binnen TamTam dan ook van haar werknemers verwacht.

De missie van TamTam is “we are here to create awesome digital experiences that make people smile”. Dit is ook terug te zien op de sites die TamTam voor de klanten maakt. Ga je bijvoorbeeld naar de site van Walibi dan komt groot in beeld “JAAAA!! IK WIL #HARDGAAN!!!”. Een slogan die je normaal niet zou verwachten van een professioneel bedrijf.

Ook op de eigen site van TamTam wordt gebruik gemaakt van humor. Onder contact valt bijvoorbeeld te lezen “Telefoon foetsie? Overkomt de besten. Geen zorgen. Wij bellen jou.”

Kortom TamTam is een energiek bedrijf met veel enthousiaste mensen. Loop je ergens mee vast dan zal je niet lang hoeven zoeken voordat je een collega hebt gevonden die je kan helpen en dan wordt het probleem ook echt opgelost. Dit is de kracht van TamTam.

2.2 Werkomgeving

TamTam heeft twee grote kantoren in Nederland, één in Rotterdam en één in Amsterdam. Tijdens mijn stage heb ik voornamelijk gewerkt op kantoor Rotterdam. Rotterdam was voor mij sneller met reizen en de mensen waarmee ik moest werken waren ook meestal op deze locatie. Af en toe heb ik echter ook in Amsterdam gewerkt. Dit gebeurde vooral wanneer mijn begeleider in Amsterdam werkte. TamTam werkt met flexplekken door het hele pand waardoor je overal je spullen kan neerploffen en aan de slag kan gaan. Er zijn echter wel onuitgesproken regels waar elke groep ongeveer zit in het pand. Aangezien ik onderdeel was van de DevOps afdeling heb ik de meeste tijd bij “hun flexplek” gewerkt.

Het DevOps team wordt geleid door Leon van Rees, mijn stagebegeleider. Leon werkte vier dagen in de week op locatie Rotterdam waardoor ik hem makkelijk kon bereiken als ik hem nodig had. Ook als hij in Amsterdam zat kon ik hem bereiken via communicatietools als HipChat en via de mail. Niek van Raaij, mijn “Buddy”, werkte meestal op locatie Rotterdam waardoor ik ook hier geen problemen ondervond in de vorm van begeleiding. Verder waren Friso en Jelmer onderdeel van het DevOps team. Ook aan hen kon ik om hulp vragen als iets niet lukte.

3. De opdracht

In dit hoofdstuk behandel ik de opdracht die ik heb uitgevoerd gedurende mijn afstudeerperiode. Ik behandel de probleemstelling waardoor deze opdracht is ontstaan, beschrijf de doelstelling van de opdracht en als laatste geef ik aan wat het beoogde resultaat van de opdracht is. Bijgevoegd in bijlage A is het originele afstudeerplan terug te vinden.

3.1 Probleemstelling

In de huidige situatie bij TamTam was er een gebrek aan overzicht op het gebied van monitoring. Dit gebrek was ontstaan doordat TamTam haar oudere applicaties liet hosten en monitoren door een hostingpartner genaamd Sentia.

Onder monitoring verstaat TamTam het controleren van de bereikbaarheid van de door haar gehoste websites. Dit wordt ook wel de uptime genoemd. Een site is up wanneer deze geen foutmeldingen weergeeft op de pagina en niet in het algemeen onbereikbaar is. Deze uptime moet 24/7 gecontroleerd worden omdat hierover afspraken zijn gemaakt met klanten, bijvoorbeeld het behalen van een uptime van 99% op de infrastructuur. TamTam wil niet dat er personen dag en nacht websites controleren. Daarom is er besloten om de monitoring met tooling te gaan doen.

Nu TamTam aan het groeien is en steeds grotere klanten zoals de Hypotheker, Oxxio en Walibi krijgt is besloten om over te stappen naar Cloud oplossingen Amazon Web Services en Microsoft Azure. Dit kwam deels uit eigen ambities en deels vanuit ambities en wensen van klanten. Voor deze hosting wilde TamTam ook eigen monitoring inrichten.

Tijdens de transitie naar de Cloud is de monitoring echter een beetje vergeten. Weliswaar waren er een aantal tools opgezet, maar er was geen proces wat voor de nieuwe monitoring gebruikt kon worden. Daarnaast hebben de personen die deze tools hadden ingericht ook ontslag genomen en de overdracht van de tooling was niet goed verlopen. Om deze reden stonden de monitoring checks verspreid over meerdere tools en was het niet duidelijk wie welke tool controleert.

Soms werden deze tools helemaal niet gecontroleerd en stuurden deze ook geen meldingen naar de eindverantwoordelijke. Daarnaast was het ook niet zeker of alle sites die gemonitord moesten worden wel in de monitoring stonden.

Doordat deze situatie niet duidelijk was werden sommige meldingen van een site die onbereikbaar is volkomen gemist en werd TamTam soms zelf op de hoogte gesteld door de klant van het aanwezige probleem.

Ook wanneer de melding wel binnenkwam werd deze niet efficiënt afgehandeld. Als noodsituatie waren er twee mensen die deze monitoring meldingen in ontvangst namen en hier wat mee deden. Deze mensen misten echter in 90% van de gevallen de kennis om het probleem op te lossen en moesten dan vervolgens achterhalen wie deze kennis wel heeft en deze persoon contacteren.
Dit koste veel tijd waardoor websites langer onbereikbaar waren.

3.2 Doelstelling

Het doel van de opdracht was om de monitoring van TamTam op orde te krijgen. Deze was in de oude situatie onoverzichtelijk. Ik heb dit gedaan door een voorstel te maken voor een nieuwe wijze van monitoring binnen TamTam. Nadat dit voorstel werd goedgekeurd ben ik dit ook daadwerkelijk gaan implementeren.

Deze nieuwe situatie moest ervoor zorgen dat er één standaard voor monitoring beschikbaar zou zijn. Ook moesten de benoemde problemen in paragraaf 3.1 zich niet meer voordoen. Elke check die af zou gaan moest in de nieuwe situatie opgemerkt worden en er moest duidelijkheid komen in wie deze checks in de gaten gaat houden.

Daarnaast was het ook de bedoeling dat wanneer een monitoringsalert afgaat eerst de server zelf dit probleem gaat proberen op te lossen. Dit heet self-healing. In de huidige situatie had TamTam nog geen enkele vorm van self-healing. De self-healing moest er uiteindelijk voor zorgen dat de developers niet belast werden met het oplossen van standaard problemen.

De nieuwe situatie moest TamTam ook in staat stellen om proactief te kunnen communiceren met de klanten. Wat moest leiden tot een hogere klanttevredenheid en snellere reactie- en oplostijden.

De nieuwe situatie moest dus een van de proposities van TamTam “Communicatie naar klanten” versterken.

Kortom het doel was het ontwerpen en implementeren van een nieuwe monitoringsituatie zodat er één manier ontstond waarop sites gemonitord konden worden.

3.3 Beoogd resultaat

Het beoogde resultaat was een nieuwe monitoring situatie vanuit een duidelijke visie, omgezet in een duidelijk proces. Deze situatie moest zo ontworpen zijn dat de monitoring naar behoren ging functioneren. De functionaliteiten van deze nieuwe situatie moesten worden getest met de requirements. Aan het eind van de afstudeerperiode zijn de volgende producten opgeleverd:

- Plan van Aanpak
- Requirementsdocument
- Toolselectie onderzoek
- Architectuur nieuwe situatie
- Implementatie nieuwe situatie
- Testdocument nieuwe situatie
- Overdrachtsdocumentatie

In hoofdstuk 5 bespreek ik de aanpak die ik heb gehanteerd voor het maken van bovenstaande producten.

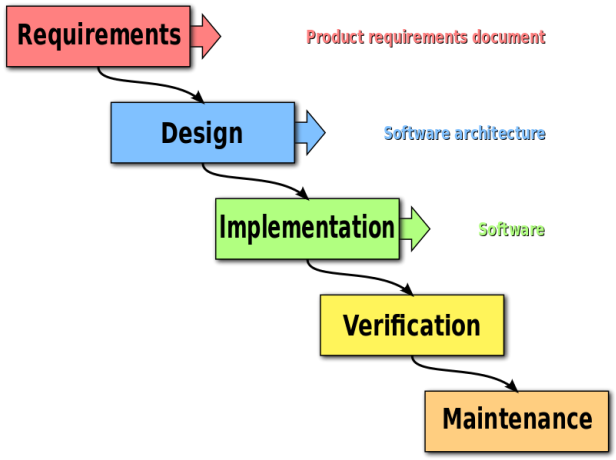
4. Tools en methodes

In dit hoofdstuk bespreek ik de methodes en de techniek die voor mijn afstuderen zijn gekozen en waarom deze zijn gekozen. Binnen het bedrijf wordt gebruik gemaakt van een vrije interpretatie van Scrum als ontwikkelmethodiek.

4.1 Ontwikkelmethodiek

Voordat ik begon met het project heb ik een ontwikkelmethodiek gekozen. Het gebruiken van een methodiek zorgt ervoor dat je efficiënter werkt omdat je niet zelf een methode van plannen moet gaan inrichten. Voor mijn beslissing heb ik gekeken naar de watervalmethode, Scrum en RUP. Dit zijn de drie methodes waar ik zelf het meest bekend mee was en om deze reden zijn deze geselecteerd.

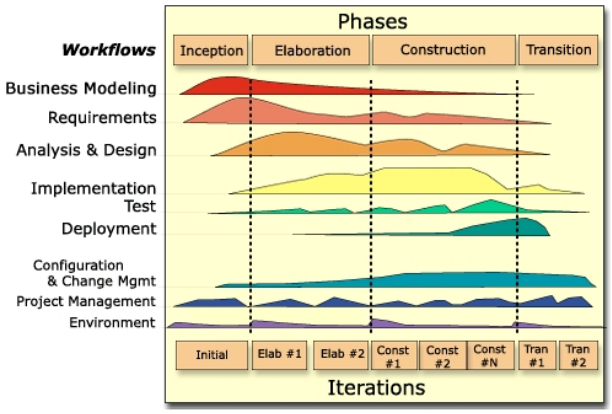
De watervalmethode (figuur 1) viel af voor dit project. In de watervalmethode maak je als eerste afspraken met de opdrachtgever en stel je de requirements vast. Vervolgens verdwijnt de opdrachtgever uit het proces en bouwt het team de applicatie. Deze wordt vervolgens pas aan het einde van het proces opgeleverd aan de opdrachtgever. De watervalmethodiek werkt met de volgende fases: vaststellen requirements, ontwerpen, implementeren (bouwen), verificatie (opdrachtgever komt terug in het proces) en onderhoud. Deze methode wordt echter af te raden om te gebruiken wanneer de opdrachtgever nog niet alle requirements kent, omdat de opdrachtgever alleen in de eerste fase inspraak heeft. Gaat er dus iets mis in de tweede fase dan wordt dit niet geconstateerd totdat het product is opgeleverd en is er veel fout werk gedaan.



Figuur 1 - Flow in de waterval ontwikkelmethodiek

Tijdens het bespreken van de opdracht in het eerste overleg met mijn opdrachtgever was onduidelijk wat voor eindproduct er moest gaan komen. Hij gaf aan dat de requirements waarschijnlijk nog gingen veranderen. Aangezien de watervalmethode het beste werkt wanneer de requirements allemaal gedefinieerd zijn, heb ik toen gekozen om deze methode af te laten vallen.

RUP (figuur 2) is een iteratieve methode die veel lijkt op de watervalmethode. Het verschil tussen RUP en de watervalmethode is echter dat RUP meer contactmoment met de klant heeft. RUP werkt met fases net zoals de watervalmethode. Deze fases zijn: de Inception, Elaboration, Construction en de Transition fase. Aan het eind van deze fase lever je een milestone product op aan de klant.

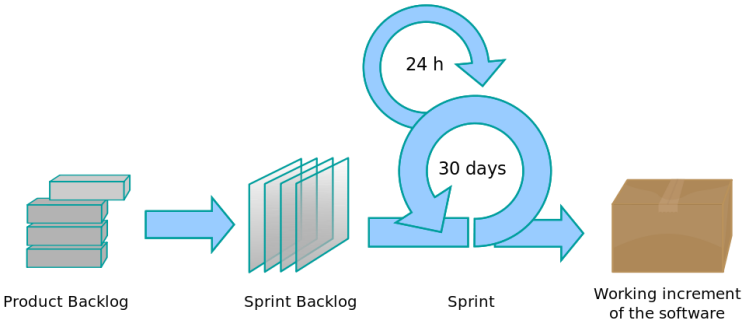


Figuur 2 - Fases en iteraties van RUP-ontwikkelmethodiek

De fases heten anders dan bij de watervalmethode, maar in essentie zijn de fases hetzelfde. In de inception fase worden er requirements opgesteld, in de elaboration wordt er ontworpen, in de construction wordt er gebouwd en getest en in de transitie fase wordt het product opgeleverd aan de klant.

Deze milestone producten geven de klant nog een kans om bij te sturen. Er kunnen bijvoorbeeld aan het eind van de inception en elaboration fase nog wijzigingen gemaakt worden. Als er echter na deze fases nog wijzigingen gemaakt moeten worden aan het product dan wordt dit een stuk lastiger met deze methode, omdat de bouwwerkzaamheden dan voor een deel of zelfs helemaal opnieuw moeten worden uitgevoerd. Dit gebeurd echter niet vaak dankzij de oplevermomenten aan het eind van de inception en elaboration fase waar fouten opgemerkt kunnen worden.

Scrum (figuur 3) is een Agile methode. In Scrum werk je met sprints die één tot vier weken kunnen duren. Aan het eind van deze sprints lever je een volledig functioneel product op aan de klant. Deze geeft dan aan of hij/zij hiermee tevreden is en vervolgens wordt de volgende sprint gepland. Om deze reden is Scrum een flexibele methode en kan deze goed toegepast worden wanneer de opdrachtgever nog geen beeld heeft van het product dat hij/zij wil.



Figuur 3 - Flow van ontwikkelmethodiek Scrum

In Scrum werk je met drie rollen. De product owner vertegenwoordigt de belangen van de stakeholders. Het development team voert elke sprint de taken in het sprint backlog uit. De Scrum master zorgt ervoor dat het development team correct met Scrum werkt.

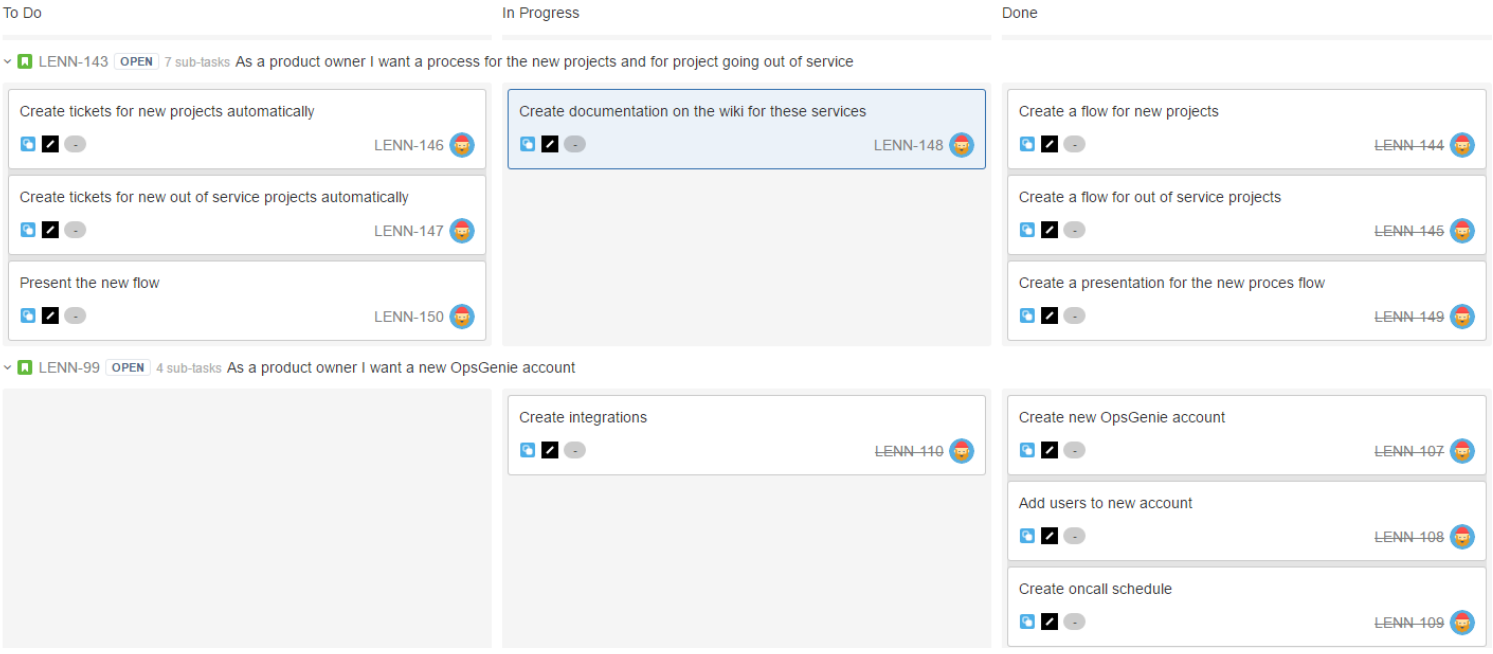
De product backlog wordt opgesteld in overleg met de product owner op volgorde van prioriteit gevuld. De prioriteit wordt per taak van hoog naar laag geïndiceerd. Dit gebeurt meestal door het toewijzen van storypoints. Het team geeft met storypoints aan hoe zwaar zij verwachten dat de taak is en de project owner geeft aan met storypoints hoe belangrijk die taak voor hem is. Vervolgens worden producten van de backlog toegevoegd aan de sprint backlog. Deze sprint wordt uitgevoerd en aan het eind van een vaste periode (een tot vier weken) wordt er een werkend product opgeleverd.

Het verschil tussen RUP en Scrum zit in de manier waarop de iteraties uitgevoerd worden. In Scrum voer je per iteratie (sprint) een gehele cyclus uit. Dit houdt in dat je aan het eind van de sprint een compleet product oplevert. Het ontwerpen, bouwen, testen en opleveren wordt allemaal in de sprint gedaan. In RUP wordt per fase een tussenproduct opgeleverd. Per fase wordt er op een punt van het proces gefocust. Het ontwerpen, bouwen en testen wordt per verschillende fase gedaan. Aan het eind van elke fase wordt het product vervolgens opgeleverd aan de klant. Deze fasen kunnen ook in lengte verschillen waar Scrum altijd een standaardlengte per sprint heeft.

Zoals al eerder vermeld viel de watervalmethode voor mij gelijk af. RUP en Scrum waren allebei goede keuzes voor het project. Met RUP had ik echter nog geen werkervaring binnen een bedrijf. Verder was mijn begeleider niet bekend met de methode RUP en wel met de methode Scrum. Ook wilde mijn begeleider nauw betrokken blijven bij het proces en hier leent Scrum zich ook beter voor dan RUP aangezien hier meer terugkoppelmomenten in zitten. Om deze redenen heb ik uiteindelijk gekozen voor Scrum. De opdrachtgever had in dit geval geen nieuwe kennis nodig en ikzelf had meer ervaring in Scrum.

Binnen TamTam is Scrum ook de standaard ontwikkelmethode. Projectteams maken hier gebruik van en een aantal grotere interne projecten worden ook met Scrum uitgevoerd. TamTam neemt hier een product voor af genaamd JIRA. In JIRA kunnen sprint borden en service desks aangemaakt worden.

Voor mij is er een eigen bord genaamd “Afstuderen Lennart” aangemaakt. Met dit bord heb ik gedurende mijn afstuderen gewerkt. In figuur 4 is te zien hoe het bord eruitzag.



Figuur 4 – JIRA-bord

Over het algemeen geef je wanneer je met Scrum werkt per taak aan hoeveel moeite hierin gestoken moet worden met behulp van story points. Binnen TamTam gebeurt dit niet. Ik heb mij hieraan aangepast en tijdens mijn afstuderen ook geen storypoints aan taken vastgehangen.

Aan het begin van mijn afstudeerperiode heb ik samen met mijn opdrachtgever de initiële backlog opgesteld. Deze backlog is gedurende de afstudeerperiode nog wel aangevuld aan de hand van de requirements en gesprekken met de opdrachtgever.

De doorlooptijd van de sprints is vastgesteld op drie weken. Dit is de standaard binnen TamTam en ik had hier geen bezwaar tegen aangezien we wel wekelijks een feedback moment hadden ingepland. Sprint 0 heeft één week geduurd en is gebruikt om kennis te maken met het bedrijf en de eerste processen op te zetten. Doordat er gekozen was om sprints te houden van 3 weken had ik geen afgerond getal aan sprints. Daarom heb ik besloten om eenmaal een sprint van 4 weken te houden wanneer er veel werk beschikbaar was. Dit is uiteindelijk in sprint 4 gebeurd.

Normaal is dit niet de manier waarop je hoort te werken met Scrum, maar doordat het aantal beschikbare weken zo uitkwam heb ik besloten om dit zo op te lossen.

Daarnaast heb ik in overleg met mijn opdrachtgever ervoor gekozen om tijdens mijn afstuderen geen daily standups te doen. Dit aangezien ik als enige aan het project werkte en mijn opdrachtgever niet de tijd had om elke dag een standup te doen. Wel hadden we de wekelijkse afspraken waarin we bespraken wat ik die week had gedaan. Hier kwamen dan ook weer nieuwe punten uit voort die dan aan het backlog toegevoegd werden en in sommige gevallen zelfs meteen aan de sprint. Deze wekelijkse afspraak viel ook op sprint deadlines dus tijdens deze afspraken werd dan ook meteen de sprint review gehouden.

De sprint retrospective heb ik deels met de opdrachtgever en deels zelf uitgevoerd. Tijdens de bespreking aan het eind van de sprint had ik het met de opdrachtgever over wat er goed en slecht ging. Daarnaast nam ik aan het einde van de sprint altijd even de tijd om te kijken waar ik meer aandacht aan moest besteden en als ik ergens op was vastgelopen hoe ik dat de volgende keer kon voorkomen. Als ik zelf niet het probleem kon oplossen dan zocht ik contact met mijn buddy Niek van Raaij die mij hier dan bij hielp of naar de juiste persoon doorverwees.

Zelf had ik nog niet eerder met zo'n losse houding met Scrum gewerkt. Het was voor mij dus wel even wennen dat ik geen tijd hoefde te besteden aan standups en het toewijzen van een aantal uren per taak. Hierdoor kreeg ik een stuk meer vrijheid in mijn werk. Aan deze extra vrijheid zat echter wel ook een stukje verantwoordelijkheid vast. Ik heb daarom in het begin voor mijn doen gedisciplineerd gewerkt, zodat ik niet te veel tijd zou besteden aan een taak. Ik had van tevoren in mijn hoofd al een planning gemaakt met de volgorde en snelheid waarmee ik de taken wilde gaan oppakken.

Naar mate ik wat verder in het project kwam ben ik minder streng geworden richting mijzelf. Ik wist dat ik de vrijheid aankon en stopte met het verzinnen van een schema. Dit is goed gegaan aangezien ik de sprints wel tot een succesvol einde bracht en de opdrachtgever tevreden was met de opgeleverde producten.

4.2 Tools

Tijdens het afstuderen heb ik ook een aantal tools gebruikt die binnen de organisatie gebruikt worden. In dit hoofdstuk bespreek ik welke tools ik heb gebruikt. De meeste van deze tools zijn eigendom van Atlassian. Hierdoor integreren de tools soepel met elkaar.

BitBucket & Git
TamTam gebruikt BitBucket om haar repositories te beheren. De tool is een implementatie van de open source technologie GIT. Binnen TamTam wordt GIT gebruikt voor het delen, mergen en branchen van source code. BitBucket is eigendom van Atlassian. Ik heb gebruik gemaakt van deze tool door hier mijn geschreven code in op te slaan en beschikbaar te stellen aan mijn collega's. Dit is standaardprocedure binnen TamTam zodat collega's van elkaars code gebruik kunnen maken als ze eenzelfde probleem tegenkomen.

JIRA
JIRA wordt gebruikt voor het werken met Scrum. Ook JIRA is onderdeel van Atlassian. Zie het voorgaande hoofdstuk voor meer informatie over hoe ik in JIRA heb gewerkt. Binnen het bedrijf wordt JIRA ook gebruikt voor het aanmaken van interne tickets. Zo heeft bijvoorbeeld afdeling DevOps een portal waarvandaan je een nieuw ticket kan aanmaken. Deze verschijnt dan in het ticket bord van DevOps.

SkyNet
SkyNet is een zelfgebouwde CMDB die nu vooral wordt gebruikt voor het opslaan van credentials, machine namen, Applicatie informatie en andere technische informatie. Daarnaast wordt Skynet ook gebruikt om software te deployen. Ik heb SkyNet gebruikt voor het opslaan van enkele user accounts en het verkrijgen van benodigde credentials.

Binnen de organisatie is er echter een discipline probleem, waardoor er in SkyNet veel gegevens missen. Niet elk project wordt door de teams goed in SkyNet toegevoegd. Dit probleem ben ik tijdens mijn afstuderen ook tegengekomen. Een groot deel van deze gegevens hebben mensen lokaal opgeslagen. Ik heb dit opgelost door deze gegevens te achterhalen en deze vervolgens toe te voegen.

HipChat
Dit is het interne chatsysteem van TamTam. De tool wordt bij TamTam gebruikt om snel en simpel contact te leggen met collega's. Wanneer je bijvoorbeeld een simpele vraag hebt of je bent op zoek naar iemand kan je gemakkelijk in contact komen via HipChat. Ook collega's die je niet persoonlijk kent kan je op deze manier makkelijk benaderen.

Projecttoolkit
Ook dit is een zelfgebouwde tool. Deze tool wordt gebruikt voor de administratie van projecten, mensen en uren. Ik heb deze tool gebruikt om mijn uren in te boeken. Ook kan je je via deze tool ziekmelden.

5. Plan van aanpak

In dit hoofdstuk behandel ik het plan van aanpak dat ik heb opgesteld aan het begin van mijn afstudeerperiode. Ik bespreek kort hoe ik het project van plan was uit te voeren. Het originele plan van aanpak is terug te vinden in bijlage B.

5.1 Producten & aanpak

De volgende Milestone producten zijn gedurende de stageperiode door mij gemaakt en opgeleverd. Elk product heeft goedkeuring van de begeleider ontvangen voordat ik verder ben gegaan met de volgende taken. Hieronder noem ik de producten die ik heb opgeleverd en welke aanpak ik hiervoor heb gehanteerd.

Het requirementsdocument
Voordat ik kon beginnen met het maken van een requirementsdocument heb ik eerst interviews gehouden met de stakeholders. Ik had hiervoor een document opgesteld. In dit document staat wie er geïnterviewd zijn en welke vragen aan hen zijn gesteld. Ook heb ik een introductie van de opdracht hierin naar voren laten komen, zodat de geïnterviewde personen een idee kregen van de werkzaamheden die ik ging verrichten. Aan de hand van dit document heb ik vervolgens de geselecteerde personen geïnterviewd.

Ik heb deze gesprekken opgenomen en later uitgewerkt in een apart document. Vervolgens heb ik deze interviews samengevat in het document waarin de vragen waren opgesteld. Aan de hand van de volledig uitgewerkte interviews ben ik de requirements gaan achterhalen. Deze heb ik geprioriteerd met de methode MoSCoW. Toen ik dacht alle requirements achterhaald te hebben, heb ik het document voorgelegd aan mijn begeleider. Deze heeft hier feedback op gegeven die ik vervolgens heb verwerkt in het document.

Onderzoek naar tools op gebied van monitoring
In dit onderzoek heb ik meerdere documenten uitgewerkt en deze later samengevoegd. Als eerste ben ik de huidige situatie in kaart gaan brengen. In dit document heb ik alle tools die op dat moment bij TamTam gebruikt werden verzameld. Van deze tools heb ik vervolgens uitgelegd waar ze op dat moment voor gebruikt werden, wat deze tools kunnen en hoe duur deze tools zijn.

Daarna heb ik een document opgesteld met vergelijkbare tools. Per geselecteerde tool heb ik aangegeven welke functionaliteiten deze hebben en hoe duur ze zijn. Vervolgens heb ik een document opgesteld waarin ik de eisen waar de tool aan moest voldoen vergeleek met de uitgekozen tools. In dit document heb ik vervolgens een advies gegeven over de te gebruiken tools. Ook heb ik dit onderzoek gebruikt om kennis op te doen over monitoring.

Een self-healend monitoringsysteem met alerting functie ontwerpen en implementeren
In deze periode ben ik aan de slag gegaan met de geselecteerde tools. Ik heb eerst een architectuur document gemaakt van de gewenste situatie. Aan de hand van deze architectuur ben ik vervolgens een werkende oplossing gaan maken. Ik heb meerdere tools moeten inrichten namelijk OpsGenie, Nagios en AWS/Azure.

Ook moesten de servers met behulp van self-healing eerst proberen zichzelf te herstellen. Wanneer dit niet lukte moest er contact opgenomen worden met de eindverantwoordelijke. Deze self-healing is echter weggevallen dankzij de gekozen hosting oplossingen (zie paragraaf 7.4) Na het implementeren van de tools heb ik tests uitgevoerd om te kijken of deze correct werkte. Ik heb deze testen uitgewerkt in een testdocument. Uiteindelijk heb ik een werkende monitoringsoplossing opgeleverd die TamTam nu ook daadwerkelijk in gebruik heeft.

Naast het ontwerpen van een architectuur en het implementeren van de tools, heb ik ook nog een eigen tool geschreven. Deze tool maakt automatisch een ticket aan in het JIRA-hosting bord wanneer een probleem langer dan een uur in OpsGenie open heeft gestaan. Ik heb deze tool eerst ontworpen in Visio een tekenprogramma van Microsoft. In dit ontwerp heb ik de attributen van de OpsGenie data en de JIRA-ticket data nog niet toegevoegd, omdat ik niet precies wist wat deze API's aan informatie mee zouden sturen. Later heb ik de attributen die ik terugkreeg van de API in de code toegevoegd.

De code heb ik geschreven in C#. Deze taal wordt gebruikt binnen de organisatie en ik was hier zelf ook al mee bekend. Tijdens het schrijven van de code heb ik het ontwerp als basis gebruikt voor de uiteindelijke applicatie. Ook heb ik deze applicatie getest. Deze testen zijn ook terug te vinden in het testdocument. Ik heb voor het testen equivalentieklassen, logische- en fysieke testgevallen opgesteld. Aan de hand van deze gevallen heb ik vervolgens unittesten opgesteld en deze uitgevoerd.

Overdrachtsdocumentatie voor de DevOps afdeling
Deze documentatie heb ik op de TamTam Confluence geschreven. Dit is een plaats waar informatie binnen de organisatie wordt gedeeld. In deze documentatie staat informatie over de gekozen tool(s) en hoe hiermee gewerkt moet worden. Ook de aanvullende opdracht het ontwerpen en bouwen van Devops.tamtam.nl is op deze locatie gedocumenteerd.

5.2 Aanvullende opdracht producten & aanpak

Voor de aanvullende opdracht heb ik een aantal overeenkomende taken moeten uitvoeren. Het vaststellen van requirements, ontwerpen en vervolgens bouwen van een applicatie had ik al uitgevoerd voor de automatische ticket creator. Ik heb hier een vrij overeenkomende aanpak voor gehanteerd.

Requirements
Voor het vaststellen van de requirements heb ik gesprekken gevoerd met de stakeholders en aan de hand van deze gesprekken requirements vastgesteld. Deze heb ik geprioriteerd met dezelfde methode MoSCoW.

UML-ontwerp
Met deze requirements ben ik daarna een ontwerp gaan maken. Om dit ontwerp overzichtelijk te houden heb ik sommige associaties op een andere manier aangegeven. Ook heb ik het ontwerp in twee delen opgesplitst. Het deel wat data wijzigt en het deel wat tickets aanmaakt. Ik heb het ontwerp opgesteld met de kennis die ik vanuit mijn opleiding heb opgedaan over UML.

Applicatie
Ik had besloten om te gaan werken met het framework ASP.NET MVC. In deze taal had ik al de nodige kennis vanuit mijn stage en het C-blok. Hierdoor kon ik gelijk aan de slag met het bouwen en hoefde ik geen nieuwe taal te leren. Dit was van belang aangezien ik maar drie weken de tijd had.

Tijdens het programmeren heb ik het ontwerp redelijk een op een overgenomen. Er zijn enkele kleine wijzingen aangebracht om de applicatie te laten werken, maar er zijn geen extra klassen toegevoegd.

Overdrachtsdocumentatie
Ik heb de applicatie overgedragen aan het DevOps team in een vergadering. In deze vergadering heb ik toegelicht hoe de applicatie in elkaar zit en waar de documentatie (ontwerp en requirements) terug te vinden is. Ook heb ik in Confluence een proces geschreven zodat eindgebruikers weten hoe ze het intake en out of service formulier moeten gebruiken.

6. Sprint 1

In deze sprint heb ik een begin gemaakt met het verzamelen van informatie. Ik heb input verzameld voor de nieuwe situatie en met deze input een requirements document opgesteld. Daarnaast heb ik in gesprekken achterhaald hoe de huidige monitoring situatie eruitzag en dit uitgewerkt. Zoals eerder vermeld was dit een sprint van één week.

6.1 Requirements verzamelen

Om een nuttige oplossing aan TamTam op te leveren was het voor mij van belang dat de mensen die hier gebruik van gaan maken inspraak krijgen. Daarom zijn er interview afspraken gemaakt met mensen die in de organisatie betrokken zijn met de hosting. Het ging om de volgende personen:

- Niek van Raaij (DevOps engineer)
- Arthur Stobbelaar (Lead architect)
- Sandor Voordes (Lead architect)
- Leon van Rees (Teamlead OS & DevOps)
- Marlies van der Schoot (Teamlead OS)

Uit de eerste drie interviews werd meteen duidelijk dat er een probleem was met de verantwoordelijkheid. Wanneer er nu een melding binnenkwam was het onduidelijk welk team hier verantwoordelijk voor was. Hierdoor werd er traag gereageerd op het probleem. Ook de verantwoordelijke voor het toevoegen van de monitoringschecks was onduidelijk.

Daarnaast werd aangegeven dat het systeem simpel in gebruik moest zijn. Zodat alle werknemers binnen TamTam met minimale inspanning de tool konden leren te gebruiken. Arthur zei op de vraag waar moet het systeem aan voldoen? “Simplicity. Het moet niet te veel kunnen”. Dit requirement kwam in elk interview naar voren. Binnen de organisatie was het dus van groot belang dat de nieuwe situatie zonder al te veel moeite begrijpelijk zou zijn.

Verder ontdekte ik dat er meer soorten monitoring bestaan dan ik in eerste instantie dacht. Naast het monitoren van de uptime van een website kan je ook de infrastructuur en transacties monitoren. Een voorbeeld van een transactie is bijvoorbeeld het inloggen op een website. Dit zijn eigenlijk drie totaal verschillende onderdelen binnen monitoring. Omdat het veel werk is om deze drie soorten te implementeren is er besloten om een scope op te stellen. In deze scope is besloten dat er in eerste instantie gefocust ging worden op uptime en infrastructuur monitoring.

De afgenomen interviews heb ik allemaal opgenomen met een voice recorder zodat ik deze na afloop uit kon werken. Ik heb alle interviews volledig uitgewerkt en van de volledige uitwerking een samenvatting geschreven (bijlage C). Aan de hand van deze uitwerkingen ben ik vervolgens het requirementsdocument gaan opstellen. In dit document start ik met de scope voor de opdracht. Vervolgens behandel ik de betrokken stakeholders en de geconstateerde requirements.

Ik heb de requirements uit de tekst van de uitgewerkte interviews achterhaald. Het requirement in figuur 5 is bijvoorbeeld achterhaald door dit gedeelte van het interview:

"***Wat is je mening over de huidige monitoring?***
Compleet onoverzichtelijk. Ik weet dat we Nagios hebben maar ik zou niet weten waar dit is te vinden. Wat ik in het algeheel mis is dat ik het liefste naar een site zou gaan en dan meteen kan zien wat wij hebben draaien en dat ik dan veel groene lichtjes zie."

Aan de hand van deze en nog twee andere opmerkingen is vervolgens dit requirement (figuur 5) opgesteld.

Req	Requirement	Bron
1	Het systeem moet overzichten aanleveren waar de informatie uptime en reactietijd in staat.	Interview Niek, Leon, Sandor

Figuur 5 – voorbeeld van een functioneel requirement

Op deze wijze zijn ook de andere requirements in het document opgesteld.

In het requirementsdocument (bijlage D) wordt een onderscheid gemaakt in de requirements. Ik heb een onderscheid gemaakt tussen functionele requirements, requirements buiten de scope, niet-functionele requirements en business requirements.

De functionele requirements zijn de eisen waaraan het systeem moest voldoen. Dit waren er in totaal 21. Er waren echter wel twee requirements die minder belangrijk waren dan de andere. Deze heb ik als volgt aangetoond (figuur 6).

13	Het systeem moet actief pushen. Dit houdt in dat de tool contact blijft opnemen totdat de gecontacteerde reageert. (Could have)	Interview Leon

Figuur 6 - voorbeeld van een could have requirement

Zoals te zien heb ik achter deze requirements tussen haakjes "Could have" gezet. Dit is gebaseerd op de methode MoSCoW, wat staat voor "Must have, Should have, Could have en Won't have". Dit is een methode waarmee je de prioriteit van de requirements kan organiseren. Deze could have requirements heb ik uiteindelijk wel mee kunnen nemen in de oplossing, maar had de tooling dit niet gekund dan was alsnog voor die tooling gekozen (mits er alleen "Could haves" verdwenen).

Aangezien de andere requirements allemaal van belang waren is besloten om alleen bij de could have requirements een aanduiding te gebruiken. Dit zodat het document zelf overzichtelijker blijft.

De business requirements hebben vooral betrekking tot de processen die benodigd zijn voor de nieuwe monitoring situatie. Deze processen heb ik ook ingericht voor de nieuwe situatie.

De niet-functionele requirements stellen een aantal eisen waaraan de tools moeten voldoen. Hieronder is een voorbeeld gegeven (figuur 7).

NF2	Het liefst draaien de tools in de Cloud. Dit zodat deze niet ook in beheer van TamTam moeten komen.	Interview Sandor, Arthur

Figuur 7 - voorbeeld van een niet-functioneel requirement

Door dit requirement vielen tools die het bedrijf zelf zou moeten gaan beheren dus af.

Bij alle requirements heb ik een bron vermeld. Dit zodat wanneer iemand anders dit document inkijkt en vragen heeft over een requirement, zij het interview erbij kunnen pakken en deze vraag hiermee kunnen beantwoorden. Lukt dit nog niet dan kan diegene contact opnemen met het geïnterviewde persoon.

Nadat ik het requirementsdocument had opgesteld heb ik feedback gevraagd van de stakeholders op dit requirementsdocument. Nadat ik deze feedback ontvangen had heb ik nog enkele requirements aangepast om deze te verhelderen. Daarna heb ik goedkeuring ontvangen van mijn opdrachtgever en ben ik met dit document als basis aan de slag gegaan met mijn onderzoek.

6.2 Intake huidige situatie

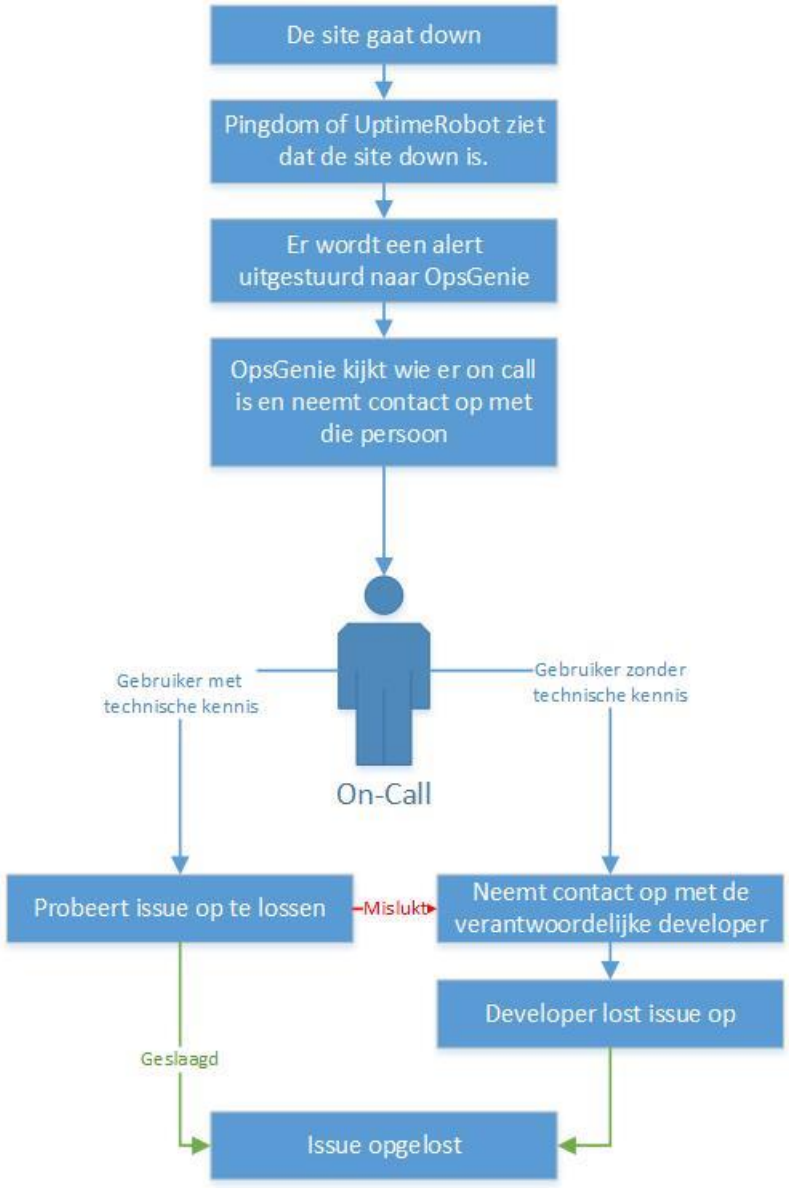
Toen ik begon bij TamTam heb ik eerst een gesprek gehad met Niek van Raaij. In dit gesprek heeft hij mij verteld over de hosting en de monitoring in de huidige situatie. TamTam host hun websites op verschillende manieren. Het grootste gedeelte wordt gehost in de EvoCloud. Dit is eigen hardware van TamTam die onderhouden wordt door hostingpartner Sentia. Verder host TamTam ook nog in de Cloud bij AWS en Azure. Deze hosting wordt gecontroleerd door TamTam zelf.

In dit gesprek gaf Niek een lijstje van tools die op dat moment werden gebruikt voor de monitoring. ook gaf hij aan wat de tool doet, hoeveel de tool op dat moment werd gebruikt en voor welke van deze tools hij een voorkeur had. Ook in de interviews met stakeholders heb ik gevraagd welke tools er op dat moment in gebruik waren, maar daar kwamen geen nieuwe tools in naar voren. In de oude situatie waren de volgende tools in gebruik:

- Nagios Core
- Pingdom
- UptimeRobot
- Azure portal
- CloudWatch
- OpsGenie
- New Relic

Ik heb toen aan de hand van het opgestelde lijstje een document (bijlage E hoofdstuk 2) opgesteld waar per tool toelichting wordt gegeven. In dit document staat wat de tool kan en waar de tool toen voor werd gebruikt bij TamTam. Verder wordt er ook per tool aangegeven wat deze per maand kost. Van bovenstaande tools heb ik alleen niet met New Relic gewerkt. Deze tool wordt gebruikt voor monitoring van code wat buiten mijn scope viel.

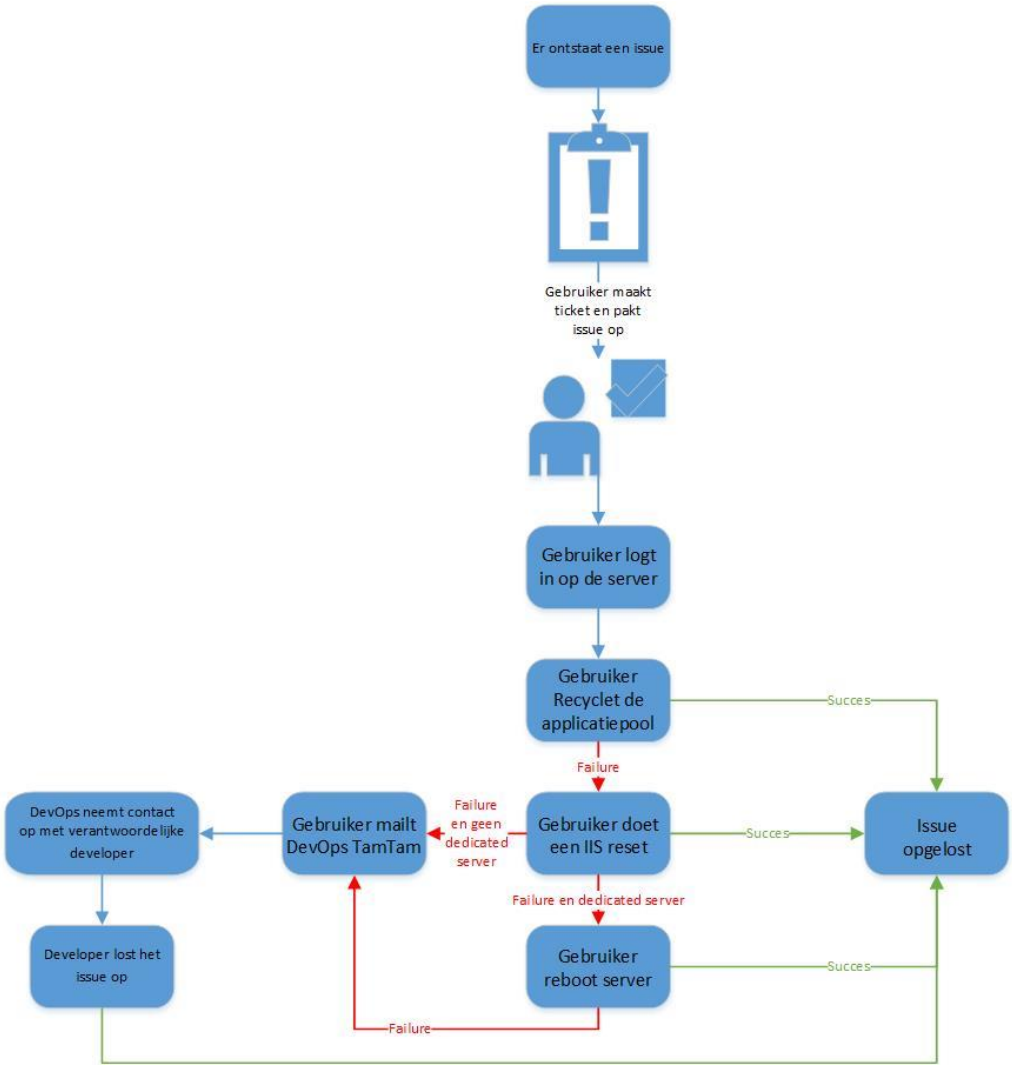
Om aan te tonen hoe de tools uit het lijstje gebruikt worden heb ik twee flows opgesteld. De eerste flow geeft aan wat er gebeurde wanneer er iets fout ging met een site die wordt gehost op AWS of Azure. De tweede flow geeft aan wat er gebeurde wanneer er iets fout ging met een applicatie gehost in EvoCloud.



Figuur 8 - In de huidige situatie zag de flow van TamTam er als volgt uit wanneer er iets fout ging met een website die gehost werd op Azure of Amazon AWS.

In figuur 8 is te zien dat er in de oude situatie twee verschillende tools werden gebruikt om een alert uit te sturen via OpsGenie. Wanneer dit alert werd uitgestuurd probeerde de on-call persoon dit op te lossen en als dit niet lukte nam diegene contact op met de verantwoordelijke developer.

Wat mij hier opviel was dat er twee tools werden gebruikt die beide keken of een site up of down is. Terwijl het veel gemakkelijker is om hier één tool voor te gebruiken. Verder waren er maar twee personen die on-call stonden in het oude schema. Een van deze twee personen had geen technische kennis. In 50% van de gevallen moest dus een developer gecontacteerd worden, waardoor het oplossen van het probleem langer duurde.



Figuur 9 - In het geval de applicatie bij Sentia op het platform EvoCloud werd gehost zag de flow er als volgt uit.

Uit figuur 9 blijkt dat wanneer er een issue ontstaat iemand bij Sentia aan de slag gaat met het probleem. Zij proberen dit op te lossen door een standaard flow te volgen. Lukt dit niet dan gaat deze persoon vervolgens contact opnemen met de DevOps afdeling van TamTam.

Wanneer de mail bij TamTam binnenkwam moest deze eerst gezien worden door DevOps. Als dit eenmaal was gebeurd dan ging DevOps kijken of zij het probleem konden oplossen. Lukte dit niet dan werd uiteindelijk door DevOps contact opgenomen met de verantwoordelijke developers.

Doordat deze flow erg uitgebreid was ontstond er enige ruis tussen de connectie van Sentia naar DevOps en van DevOps naar de developers. Hierdoor werd er door TamTam niet proactief gehandeld wanneer er een site down ging. Het gebeurde dan ook regelmatig dat een klant contact opnam met de hulpdesk en aan hen vertelde dat hun website niet meer werkte. Op dit moment werd het probleem dan ook pas binnen de organisatie bekend.

6.3 Reflectie sprint 1

Aan het einde van mijn eerste week (en sprint) bij TamTam had ik veel indrukken opgedaan van de mensen, het bedrijf en de vorm van mijn opdracht. De interviews had ik vrij vroeg in het traject afgenomen waardoor ik snel veel nieuwe kennis over monitoring opdeed. Dit kwam doordat ik weinig voorkennis hierover had en daardoor was bijna alles nieuw voor mij.

Ik had zelf wel het idee dat ik tijdens de interviews meer had kunnen doorvragen als ik meer kennis had gehad over monitoring en daardoor makkelijker requirements had kunnen achterhalen. Hiervan heb ik geleerd dat ik een afweging moet maken bij het inplannen van interviews. Als ik snel veel over het onderwerp wil leren (en de stakeholders hebben hier voldoende kennis van) dan kan ik het makkelijkste de interviews als eerste inplannen en tijdens de interviews nieuwe kennis opdoen. Wil ik echter beter kunnen doorvragen in het interview waardoor ik meer requirements kan ontdekken, dan is het verstandiger om mijzelf eerst in het onderwerp te verdiepen.

Ik ben verder wel tevreden met de door mij verzamelde requirements. Deze zijn na controle van de stakeholders en het verwerken van hun opmerkingen goedgekeurd waardoor ik in de volgende sprint aan de slag kan met het onderzoek naar tools die aan de requirements voldoen.

Ook over de intake ben ik tevreden dit is op kleine schaal gebeurd in overleg met Niek aangezien hij wist hoe de situatie in elkaar stak. Ik heb hierdoor snel in een gesprek de aanwezige tooling kunnen achterhalen waardoor ik een goed beeld kreeg van de huidige situatie. Dit idee heb ik vervolgens uitgewerkt en nog even laten controleren door Niek.

6.4 Sprint 1 in het kort

In dit hoofdstuk vat ik kort samen wat opgeleverd moest worden, of dit is gelukt en hoe dit is gebeurd.

Op te leveren	Opgeleverd?
Requirementsdocument	Ja, opgesteld aan de hand van vijf interviews gehouden met de stakeholders. Interviews zijn uitgewerkt en als bijlage terug te vinden. Goedgekeurd door stakeholders.
Intake huidige situatie	Ja, document met informatie over de huidige situatie opgeleverd. Opgesteld na een gesprek met Niek en door hem laten controleren en goedkeuren.

7. Sprint 2

In de tweede sprint ben ik aan de slag gegaan met de inventarisatie van tools op de markt. Naast de marktinventarisatie heb ik ook gekeken naar de aanpak van grote IT-bedrijven op gebied van monitoring. Ook heb ik een proof of concept opgeleverd in één van de geselecteerde tools. Gebaseerd op deze informatie heb ik vervolgens een advies opgesteld. Daarnaast heb ik mij verdiept in self-healing van servers.

7.1 Inventarisatie monitoringtools

Nadat ik de intake van de huidige situatie had gemaakt ben ik gaan kijken naar de andere opties aanwezig op de markt. Bij TamTam was namelijk niet bekend welke tools aangeboden werden op de markt. Dit had als reden dat de afdeling DevOps onderbezet was en er dus snel een paar tools in gebruik waren genomen zodat er in ieder geval wel gemonitord kon worden.

Ik heb de selectie van deze tools gemaakt aan de hand van de al aanwezige tools bij TamTam en de requirements. Op deze manier had ik een beeld gevormd waar de tools aan moesten voldoen. De volgende tools heb ik tijdens de inventarisatie bekeken:

- Uptrends
- Statuscake
- Dynatrace
- VictorOps
- BigPanda
- Pagerduty
- Xmatters
- Panopta

Deze tools zijn geselecteerd door te zoeken naar trefwoorden op Google. Vervolgens ben ik gaan kijken op de site van de tools. Meestal vond ik hier wel een kopje features waar de functionaliteiten van de tool werden aangegeven. Wanneer deze vervolgens overeenkwamen met de requirements en de opgestelde eisen dan besloot ik om deze tool te selecteren.

Nadat ik een aantal interessante tools verzameld had ben ik deze gaan uitwerken (bijlage E, hoofdstuk 3). Voor al deze tools heb ik gekeken naar de functionaliteiten die deze aanbieden en voor welke prijs. Deze uitwerking heb ik later gebruikt in de selectie van de nieuwe monitoring situatie.

7.2 Monitoring bij grote IT-bedrijven

Nadat ik een beeld had gevormd van de aanwezige tools binnen TamTam en op de markt, ben ik gaan kijken of deze tools overeenkwamen met tools die bedrijven met een groot IT-landschap gebruiken (bijlage E, hoofdstuk 4). Op de site Stackshare (Stackshare, 2017) heb ik achterhaald welke monitoring tools deze bedrijven gebruikten. Op deze site heb ik een selectie gemaakt van vijf grote bedrijven. Dit betreft de volgende:

- Spotify
- Dropbox
- Atlassian
- Stack overflow
- Vine

Deze bedrijven heb ik in een tabel tegenover de tools uit de intake en selectie uitgezet (figuur 10).

Tooling TamTam	Atlassian	Spotify	Dropbox	Stack overflow	Vine
Pingdom	X	X	X		X
Pagerduty	X			X	X
Nagios			X		X
Cloudwatch	X				
New Relic	X	X			

Figuur 10 – Vergelijking geselecteerde tooling tegenover tooling bij grote IT-bedrijven

Een paar tools die geselecteerd waren kwamen meerdere keren naar voren. Het viel mij op dat de tool Pingdom erg veel voorkwam. Van de vijf geselecteerde bedrijven wordt deze tool door vier bedrijven gebruikt. Ook viel het op dat Pagerduty door drie van de vijf bedrijven wordt gebruikt.

Verder zijn er twee bedrijven die gebruik maken van New Relic en Nagios. Ook is er een bedrijf dat gebruik maakt van Cloudwatch. Aan de hand van deze matrix zag ik dat de tools die ik geselecteerd had vaker voor kwamen bij grotere IT-bedrijven. Ik heb hieruit geconcludeerd dat ik op het goede pad was, aangezien deze bedrijven dankzij hun technische producten succesvol zijn geworden.

Er waren ook een aantal tools die ik zag terugkomen maar zelf nog niet was tegengekomen. Ik ben vervolgens gaan kijken wat deze tools konden doen op hun website. Ik heb net zoals bij de inventarisatie de tools uitgewerkt en hiervan de functionaliteiten en prijs genoteerd. Ook heb ik vervolgens aangegeven of deze tools interessant waren voor TamTam.

Tools	Atlassian	Spotify	Dropbox	Stack overflow	Vine
Datadog	X	X			X
Sentry			X		X
Airbrake				X	
Bosun				X	

Figuur 11 - Onbekende tools

Zoals te zien in figuur 11 wordt vooral Datadog door veel van de bedrijven gebruikt. Sentry wordt door twee van de bedrijven gebruikt en Stack overflow gebruikt twee andere tools genaamd Airbrake en Bosun. Airbrake wordt ook door andere bedrijven waar ik niet naar had gekeken gebruikt en is daarom ook meegenomen in de analyse. Bosun wordt maar door één ander bedrijf gebruikt en is daarom niet verder uitgewerkt.

Uit de analyse kwam naar voren dat de tooling niet interessant was voor TamTam. Datadog de tool die het meest voorkwam moest dan geïnstalleerd worden op alle aanwezige EvoCloud servers, aangezien hier geen standaard integratie voor was. Ook de integraties met Azure en AWS waren vrij ingewikkeld om op te zetten. Daarnaast was de tool te duur omdat je per server moet betalen.

Omdat de tool niet beter is of meer functionaliteiten heeft ten opzichte van de andere tooling, heb ik besloten om deze tool af te laten vallen. Een overstap zou namelijk veel tijd kosten en geen extra's toevoegen.

De tools Airbrake en Sentry vielen allebei af omdat deze functionaliteiten vervulden die buiten de scope van de opdracht vielen.

7.3 Toolselectie nieuwe situatie

Na deze inventarisatie ben ik gaan beslissen welke tools in de nieuwe situatie in gebruik genomen gingen worden (bijlage E, hoofdstuk 5). Ik besloot om alle tools die tijdens het onderzoek naar de huidige situatie en de inventarisatie van de markt naar voren waren gekomen, mee te nemen in de selectie. De nieuwe tools die bij de inventarisatie van grote marktspelers naar voren waren gekomen had ik besloten niet te gaan gebruiken (zie paragraaf 7.2).

Om de keuze overzichtelijk te houden heb ik een onderscheid gemaakt in twee soorten tools namelijk monitoring- en alertingtools.

De monitoringtools die ik heb meegenomen in de selectie zijn de volgende:

- Nagios Core
- Pingdom
- UptimeRobot
- Uptrends
- Statuscake
- Dynatrace
- BigPanda
- Xmatters
- Panopta

De alertingtools die ik heb meegenomen in de selectie zijn de volgende:

- OpsGenie
- VictorOps
- Pagerduty
- Panopta
- Xmatters

Panopta en Xmatters komen in beide selecties naar voren. Deze tools kunnen zowel monitoren als alerts uitsturen.

Als eerste heb ik een overzicht gemaakt met eisen. Deze eisen heb ik opgesteld aan de hand van het gemaakte requirementsdocument. Een eis was bijzonder belangrijk voor alle stakeholders en dit was de eis: ”*Het systeem moet simpel in de omgang zijn*”. Deze eis was belangrijk omdat TamTam als bedrijf niet veel tijd wilde besteden op het gebied van monitoring. Het systeem moest dus simpel zijn, zodat gebruikers de tool snel konden leren gebruiken.

Ook in de eisen is onderscheid gemaakt tussen monitoring en alerting tooling. Ik heb de eisen tegenover de tools uitgezet in een tabel.

7.3.1 Monitoringselectie

De volgende tabel (figuur 12) betreft de tools op het gebied van monitoring:

Eisen	Nagios	Pingdom	UptimeRobot	Uptrend	Statuscake	Dynatrace	BigPan	Panopt	Azure	AWS
Systeem moet overzichten aanleveren met informatie zoals uptime en reactietijd	X	X	X	X	X	X		X	X	X
Tool moet op infrastructuur niveau kunnen monitoren	X					X		X	X	X
Tool moet op functioneel niveau kunnen monitoren	X	X	X	X	X	X		X	X	X
Er moet een dashboard zijn waar van alle sites zichtbaar is of deze up of down zijn.	X	X	X	X	X			X		
Tool moet uptime checks kunnen toevoegen	X	X	X	X	X	X		X	X	X
Systeem moet schaalbaar zijn	X	X	X	X	X	X	X	X	X	X
Je moet een server kunnen muten als deze voor onderhoud down gaat	X	X	X		X	X		X		
Er moet een uptime report gegenereerd kunnen worden	X	X		X	X	X	X	X		
Het systeem moet simpel in omgang zijn		X		X	X				X	X
Tool moet meldingen kunnen geven met email	X	X	X	X	X	X	X	X	X	X
Tool moet meldingen kunnen geven met sms		X	X	X	X		X	X		
Er moet getest kunnen worden of bepaalde functionaliteiten werken	x	X		X	X	X		X	X	X
	x met custom plugins									

Figuur 12 – Vergelijking monitoring tooling

Zoals te zien in deze tabel voldoen de meeste tools aan de gestelde eisen. Er voldoen echter maar vijf tools op het gebied van de belangrijkste eis namelijk gebruikersgemak. Verder viel het ook op dat slechts de helft van de tools infrastructuur kunnen monitoren, terwijl negen van de tien uptime kunnen monitoren.

Uiteindelijk heb ik zes van de bovenstaande tools geselecteerd omdat deze aan de meeste eisen voldeden of omdat deze al vanuit TamTam in gebruik waren. Per tool bespreek ik kort waarom ik deze heb geselecteerd.

Nagios Core
Nagios heb ik geselecteerd omdat deze tool aan alle eisen behalve het gebruikersgemak en het sms'en voldeed. Het gebruikersgemak was de belangrijkste eis, maar omdat Nagios al bekend was binnen de organisatie verwachte ik hier geen problemen mee.

Daarnaast wordt Nagios gehost door Sentia die dit gebruikt om de EvoCloud servers van TamTam te monitoren. Ook stonden er eigen checks van TamTam in die keken naar uptime van websites. De tool was dus al ingericht met checks op de EvoCloud servers en een aantal checks op websites van TamTam. Het probleem wat in de huidige situatie vooral met Nagios werd ervaren was dat de alerts van deze checks nergens binnen kwamen en het portal waarin deze zichtbaar waren ook niet werd gecontroleerd.

Panopta
Panopta is een tool met veel functionaliteiten. Net zoals Nagios is Panopta ook niet simpel in gebruik. Wel heeft deze tool als voordeel dat deze naast de monitoring (infrastructuur en functioneel) ook een alertingfunctie aanbiedt.

Panopta miste als enige eis de gebruikersvriendelijkheid. Ook is Panopta de meest dure tool van deze selectie. Aangezien de prijs en het missen van gebruikersvriendelijkheid de enige minpunten aan deze tool waren heb ik besloten om deze tool te selecteren, omdat er verder geen enkele andere tool aanwezig was die aan alle eisen op gebied van monitoring voldeed.

Statuscake
Statuscake heb ik geselecteerd omdat deze tool aan alle eisen op de monitoring van infrastructuur na voldeed. Voorwaarde was dus wel dat er andere tools in de selectie moesten zitten die de infrastructuur konden monitoren. Deze tool zag er veelbelovend uit en hier is dan ook een proof of concept in gemaakt. Hier is meer over te lezen in het hoofdstuk "*Proof of concept Statuscake*".

Pingdom
Voor Pingdom geldt hetzelfde als voor Statuscake. Beide tools hebben overeenkomende functionaliteiten en voldoen aan dezelfde eisen. Het grootste verschil tussen Pingdom en Statuscake is het feit dat Pingdom al in gebruik was bij TamTam en Statuscake nog niet. Wel is Pingdom een stuk duurder dan Statuscake. Dit was ook de aanzet om een proof of concept in Statuscake op te gaan stellen.

AWS Cloudwatch
Voor Cloudwatch is gekozen omdat deze tool automatisch geïntegreerd is met de hosting die wordt afgenomen bij AWS. Verder voldoet het aan de eis van gebruikersgemak omdat er niets extra's geïnstalleerd hoeft te worden om de monitoring op te zetten. Er moeten alleen regels aangemaakt worden in de interface van Cloudwatch en de situatie werkt. Ook heeft Cloudwatch een functie voor uptime testing in Cloudwatch waarmee je kan kijken of je webpagina nog in de lucht is.

Azure
Ook voor Azure is gekozen omdat deze tool automatisch geïntegreerd is met de hosting die wordt afgenomen bij hen. Ook met deze tool hoef je geen extra programma op je server te installeren voordat je kan gaan monitoren. Je moet alleen regels aanmaken. Deze regels stel je in en vervolgens is je infrastructuur gemonitord. Ook in Azure is een functie voor uptime testing aanwezig.

Vervolgens heb ik een aantal opties genoteerd en uitgelegd waarom deze optie interessant was. Ook heb ik de prijs van deze optie aangegeven. Deze opties staan uitgewerkt in bijlage E.

7.3.2 Proof of concept Statuscake

Omdat er vanuit de stakeholders interesse was in Statuscake heb ik hier een proof of concept mee gedaan (bijlage F). De andere keuzes waren of al in gebruik of het maken van een proof of concept daarmee zou te veel tijd en geld kosten.

In het proof of concept heb ik een aantal gestelde eisen getest. Er is gekeken of deze eisen aanwezig waren binnen de tool en zo ja hoe deze in de tool te gebruiken waren. De volgende eisen waren in overleg met de opdrachtgever opgesteld.

- Statuscake moet een uptime-test kunnen aanmaken.
- Statuscake moet integreren met OpsGenie.
- Statuscake moet een test voor het inladen van de pagina kunnen aanmaken.
- Statuscake moet kunnen aangeven of een SSL-certificaat bijna verloopt.

De test voor de inlaad snelheid was hierbij nieuw. Hier was op dat moment een developer naar op zoek om inzicht te krijgen in de laadtijden van zijn pagina. Ik had in de documentatie van Statuscake gelezen dat dit een functionaliteit was in de tool en heb dit toen ook getest in het proof of concept.

In dit proof of concept heb ik per vraag een hoofdstuk geschreven waar ik mijn exacte stappen in uitleg. Dit zodat het concept voor anderen eventueel ook reproduceerbaar was. Nadat ik alle gestelde eisen had getest (die overigens allemaal slaagden) heb ik ook nog andere functionaliteiten die in de business versie aanwezig waren getest.

Dit document heb ik in het Engels geschreven. Dit heb ik gedaan zodat deze informatie ook op de Confluence page van TamTam toegevoegd kon worden mocht er voor Statuscake gekozen worden. Op de Confluence van TamTam is de voertaal namelijk Engels.

7.3.3 Alertingselectie

De volgende tabel (figuur 13) betreft de tools op het gebied van alerting:

Eisen	OpsGenie	VictorOps	Pagerduty	Panopta	Xmatters
Tool moet contact kunnen opnemen met specifieke personen	X	X	X	X	X
Tool moet een mail sturen zodra een site down gaat	X	X	X	X	X
Tool moet met het opnemen van contact een duidelijke boodschap meesturen zodat de gebruiker op de hoogte is wat er is gedaan en welke foutmelding zich voortdoet	X	X	X	X	
Tool moet rekening houden met de prioriteit	X	X	X	X	
Een issue moet kunnen escaleren	X	X	X	X	
DevOps, Projectteams en OS moeten tegelijk op de hoogte worden gesteld van een probleem	X	X	X	X	X
Je moet een server kunnen muten als deze voor onderhoud down gaat	X	*		X	
Tool moet naast mail ook op andere manieren contact kunnen leggen	X	X	X	X	X
Je moet groepen kunnen managen in de tool	X	X	*	X	X
Tool moet contact blijven zoeken totdat de gebruiker op het probleem reageert	X	X		X	
Tool moet een on-call functie hebben	X	X	X	X	X
		* zit alleen in Enterprise versie	* zit alleen in Standard versie		

Figuur 13 - Vergelijking alerting tooling

In deze tabel werd snel duidelijk dat er eigenlijk drie tools aanwezig waren die in aanmerking kwamen namelijk OpsGenie, VictorOps en Panopta. Ik heb deze tools vervolgens ook geselecteerd en gebruikt om de opties te geven.

Voor deze drie tools geldt dat ze allemaal aan alle eisen voldoen die zijn gesteld. Dit is dan ook de reden dat deze tools zijn geselecteerd. Ik heb vervolgens van deze drie tools de verschillen onderling nog uitgelicht die niet in de matrix naar voren kwamen. Dit waren verschillen zoals OpsGenie is bij TamTam al bekend en VictorOps is een stuk duurder omdat de duurdere versie benodigd is om aan alle eisen te voldoen.

Ik heb vervolgens twee opties aangeboden en hiervan de prijs en de voor- en nadelen uitgelicht. Deze optie heb ik daarna ook weer voor het advies gebruikt. VictorOps is niet naar voren gekomen in deze opties aangezien deze in vergelijking met OpsGenie een stuk duurder was en hetzelfde aanbood.

Panopta was ook een stuk duurder met dezelfde functies, maar deze tool kon ook op het gebied van monitoring ingezet worden en kwam daarom wel in aanmerking.

7.3.4 Advies

Nadat ik alle tools met elkaar had vergeleken en een aantal opties had opgesteld voor de nieuwe situatie ben ik in overleg gegaan met de stakeholders. Ik heb hun input op deze opties gevraagd om een beeld te krijgen welke van deze opties het meest aansprak in het bedrijf. Hieruit kwam naar voren dat er interesse was in de tool Statuscake. Voor de stakeholders heb ik hier dan ook een proof of concept van gemaakt.

Toen ik dit proof of concept eenmaal had uitgewerkt was ik tot de conclusie gekomen dat Statuscake goed aansloot op de eisen en bovendien goedkoper was dan Pingdom. Om deze reden was ik dan ook van plan om Statuscake uiteindelijk te adviseren.

Terwijl ik met dit advies bezig was raakte ik in gesprek met een stakeholder. In dit gesprek kwam naar voren dat er sprake was van een nieuw samenwerkingscontract met Sentia. Hier zou dan ook de monitoringservice Nagios bij zitten. In dat gesprek ontstond toen de vraag waarom Statuscake überhaupt dan nog nodig was, aangezien Nagios ook uptime testen kan uitvoeren.

Met Nagios was er echter nog wel het probleem dat uiteindelijk de developers testen moesten kunnen toevoegen in de nieuwe situatie. De testen zijn simpel genoeg voor developers om zelf te kunnen schrijven, maar de toegang tot de configuratie file staat niet voor iedereen open. Hier is uiteindelijk een constructie voor bedacht met git. De developers kunnen bij de config file maar kunnen deze niet op de server implementeren. Het implementeren zal gedaan worden door DevOps nadat zij de nieuwe testen hebben nagekeken. Meer over deze constructie is in de paragraaf 8.1 te lezen.

Door deze oplossing kon ik een afweging maken tussen de extra functies van Statuscake die in het proof of concept naar voren waren gekomen en de simpele functionaliteiten van Nagios. De extra functies van Statuscake waren interessant, maar in een gesprek met de project owner gaf deze aan dat daar waarschijnlijk niemand gebruik van ging maken. Dit gaf de doorslag voor mij om Statuscake af te laten vallen en voor Nagios te kiezen.

Nu zag de situatie er zo uit dat Nagios en OpsGenie de nieuwe tools zouden gaan worden. Om Nagios echter te installeren op elke AWS en Azure omgeving zou veel tijd kosten. Daarom heb ik besloten om hier de ingebouwde monitoringtooling voor te gebruiken die Azure en AWS aanbieden. Alleen servers die in de EvoCloud staan worden gemonitord door Nagios. De infrastructuur hiervan wordt door Sentia gemonitord.

Ik heb toen in het document een nieuwe optie toegevoegd aan de opties op het gebied van monitoring en deze gebruikt in het uiteindelijke advies (bijlage E hoofdstuk 6).

Dit uiteindelijke advies zag er als volgt uit:

- Voor de uptimetesten raadde ik aan om te gaan werken met Nagios. Deze tool voldeed aan de gestelde eisen, was al bekend binnen het gebruik, wordt gehost door Sentia (geen eigen onderhoud) en is met de integratie met BitBucket (Hoofdstuk 8.1) bewerkbaar voor developers. Dit lost de huidige problemen

- Voor de infrastructuur monitoring raadde ik de ingebouwde tools in AWS en Azure aan. Deze tools brengen geen extra kosten met zich mee, hoeven niet onderhouden te worden en zijn simpel maar effectief in gebruik.
- Ik heb gekozen voor OpsGenie voor het alerting aspect van de oplossing. Deze tool integreert met de andere gekozen tooling, is persistent in het contacteren van de on-call users en is ook simpel in gebruik.

De nieuwe situatie bestond dus uit drie tools waar in de oude situatie in totaal zes tools in gebruik waren. Hiermee had ik aan het niet functionele requirement van minder tools in de voldaan. Daarnaast biedt de nieuwe situatie één wijze van monitoring aan. Hier werd om gevraagd in de doelstelling. De self-healing is besloten om niet mee te nemen in het advies. De aanleiding hiervoor is in het volgende hoofdstuk te lezen.

7.4 Self-healing

Een eis vanuit de organisatie was ook dat de monitoring een self-healend proces in gang kon zetten. Om dit te verwerkelijken had ik twee eisen voor mijzelf opgesteld. De eerste eis was “Vaststellen wat self-healing is” en de tweede “Vaststellen hoe dit te implementeren is”.

Om deze twee vragen beantwoord te krijgen had ik een meeting ingepland met mijn begeleider. Het doel hiervan was vast te stellen wat er precies verwacht werd vanuit het bedrijf op gebied van self-healing. In dit gesprek bleek dat de self-healing binnen TamTam vergelijkbaar was met wat Sentia op dat moment deed voor TamTam. Dit hield in het uitvoeren van een applicatiepool recycle en wanneer dit mislukte het uitvoeren van een IIS reset. Ook werd aangegeven dat net zoals bij Sentia er ook eventueel een server reboot uitgevoerd kon worden, maar dit had minder prioriteit.

Toen ik hier een beeld van had gekregen vroeg ik aan mijn begeleider of hij al enig idee had hoe dit te implementeren was. Hij gaf toen aan dat dit met Powershell waarschijnlijk wel zou gaan lukken.

Vervolgens ben ik met Powershell aan de slag gegaan. Ik heb geëxperimenteerd met het uitvoeren van commando’s op een server vanaf mijn eigen pc. Nadat ik lang bezig was geweest met een vreemde foutmelding oplossen lukte het om een connectie te maken met deze server en de applicatie pool te recyclen en een IIS reset uit te voeren.

Toen ik dit aantoonde aan mijn begeleider gaf deze aan dat dit in Azure al automatisch gebeurde. Ook in AWS kon je dit automatiseren door een Beanstalk bij hen af te nemen. We hebben toen overlegd of het nuttig was voor mij om een applicatie te schrijven die dit automatisch doet. In dit overleg bleek dat nieuwe klanten bijna altijd op AWS of Azure gehost zullen gaan worden. Mocht de klant toch op EvoCloud gehost worden dan voert Sentia deze stappen uit. Om deze reden is toen besloten om niet meer verder te werken aan de self-healing en dit door de Cloud providers te laten doen.

7.5 Reflectie sprint 2

Deze sprint is succesvol afgerond. Over ik ben tevreden met de manier waarop ik de sprint heb aangepakt. Ik had echter wel enkele andere stappen kunnen nemen. Ik had bijvoorbeeld eerst kunnen kijken naar de tools van bedrijven met een groot IT-landschap. De tools die deze bedrijven gebruiken zullen namelijk gegarandeerd goed hun functies vervullen anders zou een groot bedrijf deze niet gebruiken. Vervolgens had ik dan kunnen kijken of deze tool bruikbaar was voor TamTam, maar ik wist dan in elk geval al zeker dat de tool correct zou functioneren.

Daarnaast ben ik tevreden met de wijze waarop ik de selectie heb gemaakt. Ik heb wel nog een punt geconstateerd waar ik voortaan rekening mee zal houden. Ik had namelijk van de selectie nog een sub selectie gemaakt. Aan de ene kant denk ik dat dit nuttig is om een beter besluit te maken. Je kan dan namelijk nog een keer op een rijtje zetten waarom elke tool geschikt zou zijn en dan een keuze maken aan de hand hiervan. Aan de andere kant denk ik dat wanneer je meteen een keuze maakt veel tijd kan besparen.

Hier had ik een afweging tussen deze twee keuzes moeten maken maar dit heb ik niet gedaan. Achteraf denk ik wel dat het maken van een sub selectie de juiste keuze is geweest, maar ik heb hiervan wel geleerd om iets langer stil te staan bij het kiezen van de aanpak.

7.6 Sprint 2 in het kort

In dit hoofdstuk vat ik kort samen wat opgeleverd moest worden, of dit is gelukt en hoe dit is gebeurd.

Op te leveren	Opgeleverd?
Inventarisatie monitoringtools	Ja, document met informatie over beschikbare tools opgesteld. Vervolgens ook gekeken naar grote IT-bedrijven en hier een document voor opgesteld.
Toolselectie nieuwe situatie	Ja, document met selectiecriteria opgesteld met het requirementsdocument. Vervolgens een selectie en een passend advies gegeven.
Self-healing	Ja, geconcludeerd dat deze feature door gebruik van Cloud hosting niet zelf gebouwd hoeft te worden.

8. Sprint 3

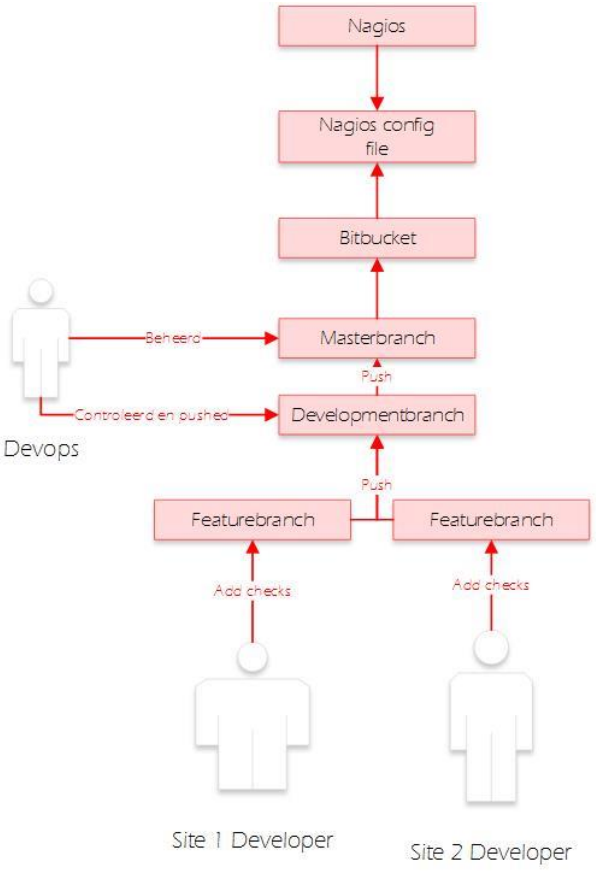
In deze sprint presenteer ik mijn advies voor de nieuwe situatie. Om dit een duidelijk advies te maken heb ik eerst een architectuur voor deze situatie opgezet. Vervolgens heb ik een presentatie met de gemaakte keuzes en architectuur gemaakt en deze gepresenteerd aan de stakeholders. Ook heb ik een tool gebouwd voor TamTam die automatisch een JIRA-ticket aanmaakt wanneer een OpsGenie alert meer dan een uur open heeft gestaan.

8.1 Architectuur

Toen ik eenmaal een advies had opgesteld voor de nieuwe situatie wilde ik aantonen hoe deze tools met elkaar gingen communiceren en wie waarvoor verantwoordelijk was. Dit was immers een eis die in de requirements voor kwam. Om dit duidelijk te maken besloot ik per tool een architectuurtekening te maken en in een architectuur document deze tekeningen uit te leggen (bijlage G).

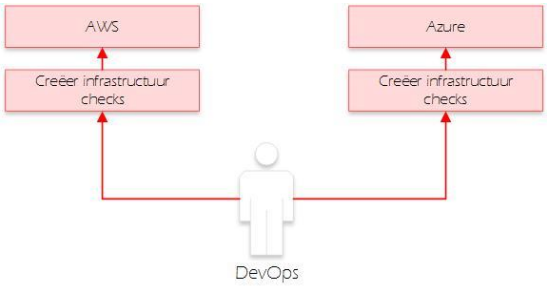
Ik zal in dit hoofdstuk kort de plaatjes behandelen zodat duidelijk wordt hoe de nieuwe situatie functioneert. Voor een uitgebreidere uitleg is het architectuur document bijgevoegd (bijlage G).

In figuur 14 is te zien hoe Nagios in de nieuwe situatie gebruikt gaat worden. Zoals te zien zal de config file van Nagios in BitBucket toegevoegd worden. Vanuit hier beheert DevOps de masterbranch. De config file die in de masterbranch staat zal door Nagios gebruikt worden. De developers kunnen een eigen branch van de development branch uitchecken, hier hun wijzigingen in doorbrengen en deze vervolgens weer mergen. Deze wijzigingen worden gecontroleerd en goedgekeurd door DevOps waarna deze de checks toevoegen aan de master branch en dus ook in Nagios.



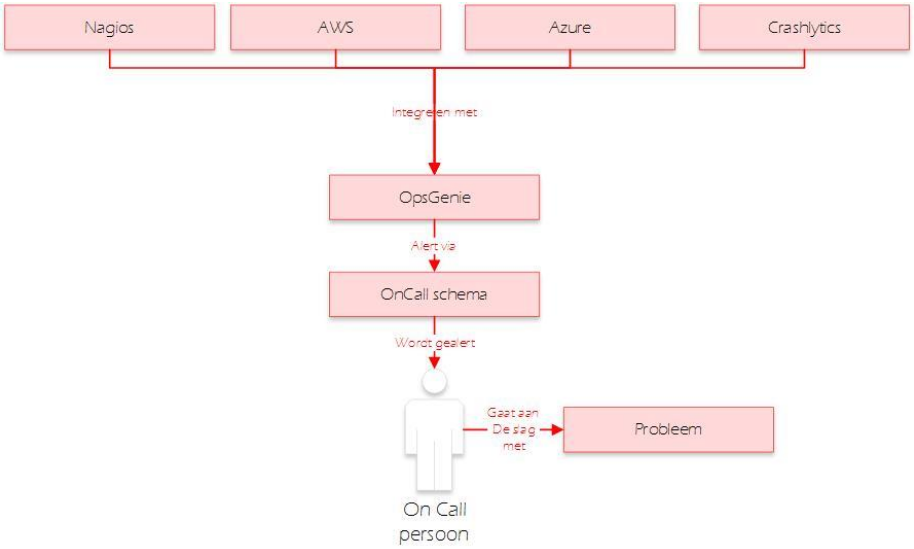
Figuur 14 – Architectuur van nieuwe Nagios situatie

Voor de infrastructuur zal er gebruik gemaakt gaan worden van AWS en Azures native monitoring. Dit zal ingericht gaan worden door afdeling DevOps. Zij richten namelijk de nieuwe omgevingen in waarop sites gehost gaan worden. Tijdens deze inrichting kunnen er ook gelijk checks toegevoegd gaan worden voor de infrastructuur. Dit zal eruit gaan zien zoals in figuur 15 is aangegeven.



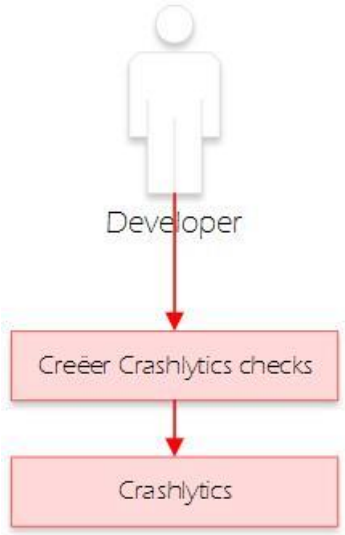
Figuur 15 – Infrastructuur monitoring architectuur

De nieuwe alerting situatie gaat als volgt werken (figuur 16). De monitoring tools integreren met OpsGenie. Wanneer er in een tool een fout wordt geconstateerd stuurt deze een bericht naar OpsGenie die hier vervolgens een alert van maakt. Deze kijkt dan vervolgens naar het on-call schema in OpsGenie en contacteert de persoon die on-call is. Deze lost dit probleem dan op of delegeert het naar de juiste persoon.



Figuur 16 - Infrastructuur monitoring OpsGenie

Mobile app monitoring



Figuur 17 – Crashlytics

De oplettende lezer heeft ongetwijfeld opgemerkt dat er een extra integratie in de tekening van OpsGenie staat. Deze monitoring tool kwam naar voren tijdens de advies presentatie. Een stakeholder was vergeten om deze tijdens zijn interview te vermelden. Gelukkig kon deze tool integreren met OpsGenie en meegenomen worden in de nieuwe situatie. Er is wel besloten om deze tool niet in de scope op te nemen omdat ik hier verder geen kennis van had opgedaan in mijn onderzoek. Ik heb de tool alleen laten integreren met OpsGenie. Verder is de verantwoordelijkheid volledig bij de developers neergelegd. Eventuele wijzigingen in de integratie met OpsGenie zullen zij ook verder regelen.

Doordat deze tool pas later aan het licht kwam was deze in eerste instantie niet meegenomen in het architectuurdocument. Na afloop van de presentatie heb ik deze vervolgens alsnog toegevoegd zoals te zien in figuur 17.

8.2 Advies presentatie

Nadat ik een advies had gevormd en hier een architectuur bij had gemaakt heb ik hier vervolgens een PowerPoint voor gecreëerd. In deze presentatie leg ik uit hoe de nieuwe situatie eruit gaat zien met de hulp van drie architectuurtekeningen. Voor elke tool heb ik een tekening gemaakt zodat ik kon uitleggen hoe deze tools ingezet gingen worden. Ook heb ik een architectuur document gemaakt waar deze plaatjes en de bijbehorende uitleg in te lezen zijn (zie bijlage G).

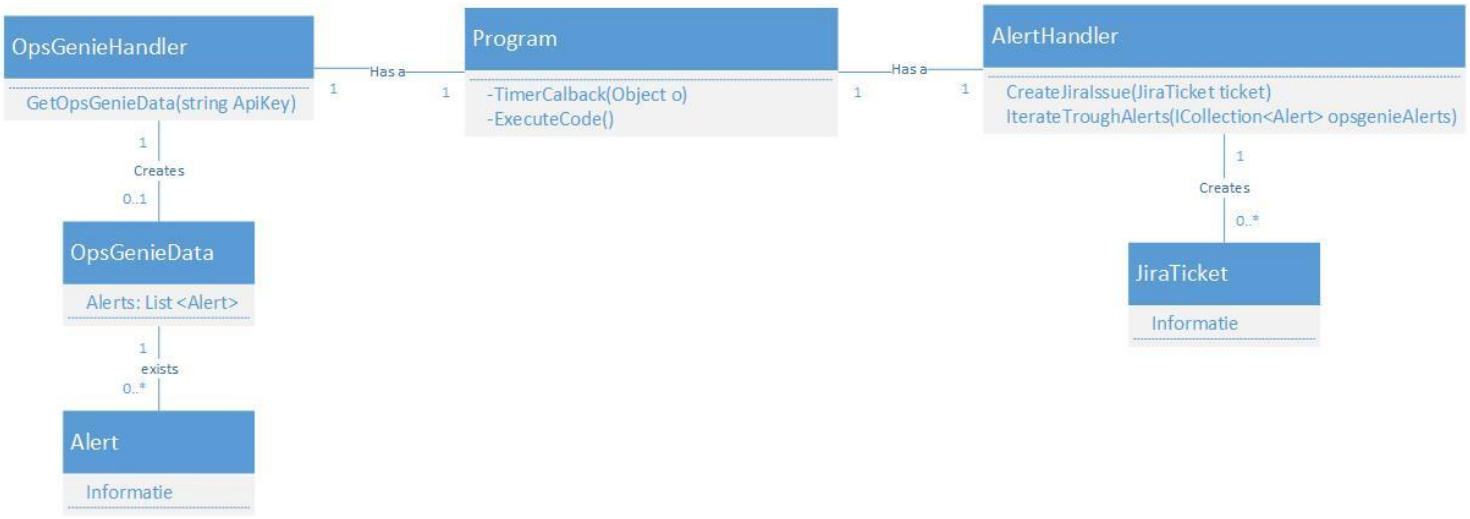
Na de presentatie kwam Arthur met de vraag of hij de tool Crashlytics ook kon meenemen in deze situatie. Zoals in het voorgaande hoofdstuk besproken is dit deels gebeurd. Nadat ik andere vragen had beantwoord heb ik vervolgens goedkeuring op mijn advies gekregen en ben ik aan de slag gegaan met het implementeren van de nieuwe situatie.

8.3 Ontwerp klassendiagram TicketCreator

Tijdens de opdracht kwam de wens vanuit de opdrachtgever om meldingen in OpsGenie slim te verwerken. Het was de bedoeling dat wanneer een site langer dan een uur down was geweest er automatisch een ticket in het DevOps bord aangemaakt werd. Als een site zo lang een probleem heeft gehad, dan moet hier altijd een onderzoek naar de aanleiding hiervan komen.

Daarvoor is besloten om een tool te bouwen die automatisch een ticket aanmaakt wanneer een alert langer dan een uur open heeft gestaan.

Om dit te realiseren ben ik in eerste instantie met een ontwerp hiervan begonnen (figuur 18). Dit ontwerp is als basis gebruikt voor de tool. In het ontwerp heb ik als basis het program genomen en verder een leeg JIRA-ticket en OpsGenieData object aangemaakt. Dit omdat ik van tevoren nog niet wist wat voor informatie ik hier precies aan moest meegeven. Hieronder is het klassendiagram weergegeven.



Figuur 18 - Klassendiagram automatische ticket creator

Ik besloot om gebruik te maken van een Timer. Deze timer roept elk uur de functie ExecuteCode aan. Wanneer dit gebeurt worden de laatste tien open alerts opgehaald door de OpsGenieHandler. Deze handler retourneert dan een lijst met alerts naar het program. Vervolgens wordt deze lijst naar de AlertHandler gestuurd. Deze kijkt vervolgens voor de gehele lijst of dat alert langer openstaat dan een uur. Is dit het geval dan wordt er automatisch een ticket aangemaakt op het JIRA-issue bord van DevOps.

Ik heb vervolgens deze applicatie gebouwd. Hoe dit is gedaan is te lezen in hoofdstuk 9.5.

8.4 Reflectie sprint 3

Ook deze sprint is goed verlopen. Wat wel minder handig was in is dat ik de adviespresentatie iets te laat had ingepland waardoor ik enkele dagen van tevoren het advies al klaar had liggen, maar door drukke agenda's deze pas op de ingeplande dag kon presenteren. Het presenteren van het advies is prima verlopen. Na een aantal kritische vragen waren de stakeholders tevreden over de voorgestelde oplossing. Ik heb ze niet over hoeven te halen er werd vertrouwd op mijn advies.

Ook het maken van de architectuur en het ontwerpen van het klassendiagram gingen goed. De architectuur is gebruikt om de nieuwe situatie te presenteren en dit zorgde voor meer duidelijkheid in mijn verhaal gedurende de presentatie.

Tijdens het ontwerpen van het klassendiagram heb ik expres de klassen alert en JIRA-ticket leeggelaten omdat ik niet wist wat voor informatie ik hierin zou gaan ontvangen. Achteraf gezien had ik deze informatie ook kunnen achterhalen door tijdens het ontwerpen in de documentatie van OpsGenie en JIRA te kijken. Het ontwerp was in dat geval completer geweest dan dat het nu is. Ik verwacht echter wel dat het ontwerp duidelijk genoeg is voor andere programmeurs.

8.5 Sprint 3 in het kort

In dit hoofdstuk vat ik kort samen wat opgeleverd moest worden, of dit is gelukt en hoe dit is gebeurd.

Op te leveren	Opgeleverd?
Architectuur	Ja, ontworpen per tool zodat het duidelijk is waar welke tool voor verantwoordelijk is.
Advies presentatie	Ja, succesvolle presentatie na afloop vragen beantwoord en vervolgens goedkeuring voor het advies ontvangen van de stakeholders.
Ontwerp klassendiagram TicketCreator	Ja, ontwerp gemaakt en later gebruikt voor het bouwen.

9. Sprint 4

In deze sprint ben ik aan de slag gegaan met het implementeren van de nieuwe situatie. Om dit meteen goed te doen ben ik niet gelijk met een live situatie begonnen. Ik heb eerst op een testserver Nagios geïnstalleerd en hierop geoefend met het maken van alerts. Daarna heb ik een nieuwe OpsGenie account aangemaakt en hier de gekozen personen en monitoringtools aan toegevoegd. Ook heb ik een eigen tool gemaakt die om het uur in OpsGenie kijkt. Als een alert langer dan een uur heeft open gestaan maakt deze vervolgens een ticket in JIRA aan. Dit heb ik ook getest.

Daarnaast heb ik ook nog processen ontworpen om te gaan gebruiken met de nieuwe monitoring. Deze processen zijn de handelswijze met monitoring wanneer er een nieuw project wordt aangemaakt of wanneer een project out of service gaat. Deze processen heb ik vervolgens ook weer aan de stakeholders gepresenteerd.

In deze sprint had ik in totaal 4 weken beschikbaar, omdat ik anders niet uit zou komen met het aantal weken tegenover het aantal sprints. Er was vooraf besloten (zie hoofdstuk 4.1) om één sprint van vier weken te houden als er veel werk aanwezig was.

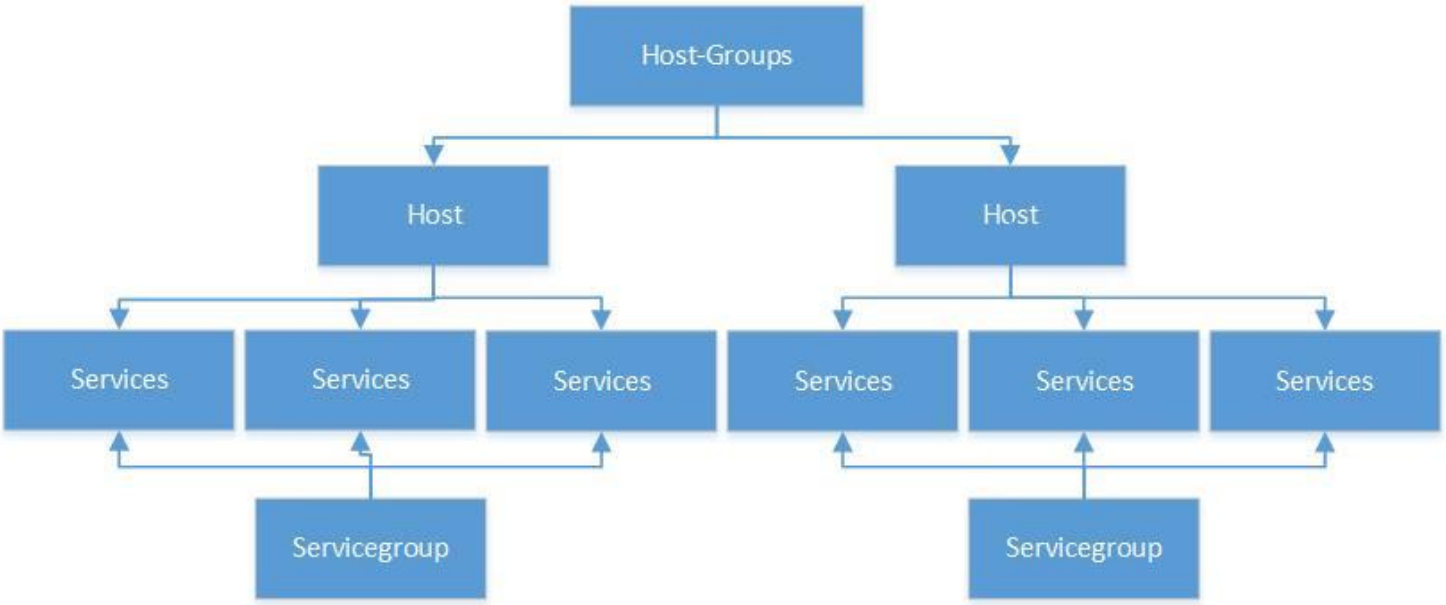
9.1 Nagios

Voor de uptime testing heb ik uiteindelijk Nagios gekozen. Omdat ik van alle tools de functionaliteiten had opgezocht en tegenover elkaar had uitgezet wist ik nog niet veel van de werking van Nagios. Op het moment dat ik begon met het inrichten van de nieuwe situatie had TamTam een Nagios server die onderhouden werd door Sentia. Deze server bevatte echter erg belangrijke checks die niet alleen door TamTam, maar ook door Sentia in de gaten werden gehouden. Op deze server kon ik dus niet gaan experimenteren.

Om deze reden heb ik besloten zelf een nieuwe Nagios server op te zetten. Deze server had als doel experimenteren met Nagios, ontdekken hoe Nagios functioneert en ontdekken wat ik hier allemaal mee kon op gebied van http-checks.

Ik heb vervolgens een Ubuntu server in AWS aangemaakt en hierop Nagios geïnstalleerd. Nadat dit gelukt was ben ik aan de slag gegaan met het configureren van test checks. In dit hoofdstuk behandel ik een test die ik heb gebruikt tijdens het werken met Nagios.

Het eerste wat ik leerde toen ik met Nagios begon te werken is dat er een classificatie in aanwezig is. Deze is niet heel ingewikkeld maar om dit snel duidelijk te maken heb ik een plaatje getekend (figuur 19).



Figuur 19 - Classificatie Nagios

Bovenaan staan de Host-Groups. Hierin worden groepen van Hosts oftewel servers gedefinieerd. Deze Hosts kunnen vervolgens services aan zich hebben hangen. De services voeren checks uit op zoals disk usage, CPU-load, memory usage, SSH en http-checks. Vervolgens kan je deze services ook nog aan een servicegroep toevoegen. Je kan op deze manier bijvoorbeeld al je CPU-checks onder een groep organiseren. Hierdoor is het gelijk overzichtelijk of het CPU van al je servers in orde is.

Het monitoren van Host-Groups en Hosts werd in de huidige situatie door hosting partner Sentia gedaan die de eigen servers van TamTam in beheer hebben. Ik heb mij beziggehouden met het maken van http-checks in de services.

Alle bovengenoemde waardes worden gedefinieerd in Nagios config files. Je kan dus niet een test toevoegen via het portal van Nagios. Onderstaande code (figuur 20) is een voorbeeld van een test die ik heb aangemaakt op mijn testserver.

```
1.  define service{
2.          use                     generic-service
3.          host_name               localhost
4.          service_description     www.superboek.nl
5.          check_command           check_http!www.superboek.nl!follow
6.          notifications_enabled   0
7.          contacts                Lennart
8.          notes                   Doe dit als de server down gaat
9.
10. }
```

Figuur 20 - Nagios check

In deze test wordt gekeken of de website www.superboek.nl goed reageert. Dit is een normale http-check die dankzij de waarde !follow een redirect kan volgen. Zoals is te zien kan je ook specifieke contacten toevoegen aan elke service.

Terwijl ik hiermee bezig was ging er ook een verzoek uit naar Sentia voor een nieuwe Nagios server. Het aanbod wat Sentia heeft gedaan is vervolgens door TamTam geaccepteerd. Er is echter wel afgesproken om de oude configuratie file te behouden. Ik kon dus in deze configuratie file wijzigingen maken zonder dat deze met de nieuwe server verloren gingen.

Om OpsGenie te laten integreren met Nagios is een nieuwe gebruiker genaamd OpsGenie aangemaakt. Deze gebruiker heeft een email adres van de nieuwe OpsGenie (hoofdstuk 7.4) waardoor alle mail die daarnaar gestuurd wordt automatisch een alert wordt. De OpsGenie gebruiker wordt standaard bij elke http-test toegevoegd zodat wanneer een van deze testen faalt een alert ontstaat in OpsGenie.

Verder heb ik documentatie geschreven voor het aanmaken van testen in Nagios. Met behulp van deze documentatie moeten developers testen in Nagios kunnen toevoegen. In deze documentatie leg ik uit wat er bij een test ingevuld moet worden om deze correct te laten werken. Dit zodat in de nieuwe situatie developers niet tegen problemen aan lopen wanneer zij een check moeten aanmaken. Deze documentatie heb ik op de Confluence van TamTam geschreven.

9.2 Pingdom/UptimeRobot testen overzetten

Nadat ik wist hoe ik http-checks moest aanmaken in Nagios kon ik beginnen met het overzetten van Pingdom/UptimeRobot. Deze keken in eerste instantie naar een aantal websites die ook al in Nagios stonden. Hierdoor hoefde ik maar weinig testen over te zetten naar Nagios. De testen die nog niet in Nagios stonden heb ik vervolgens toegevoegd in Nagios.

9.3 Azure en AWS testen integreren

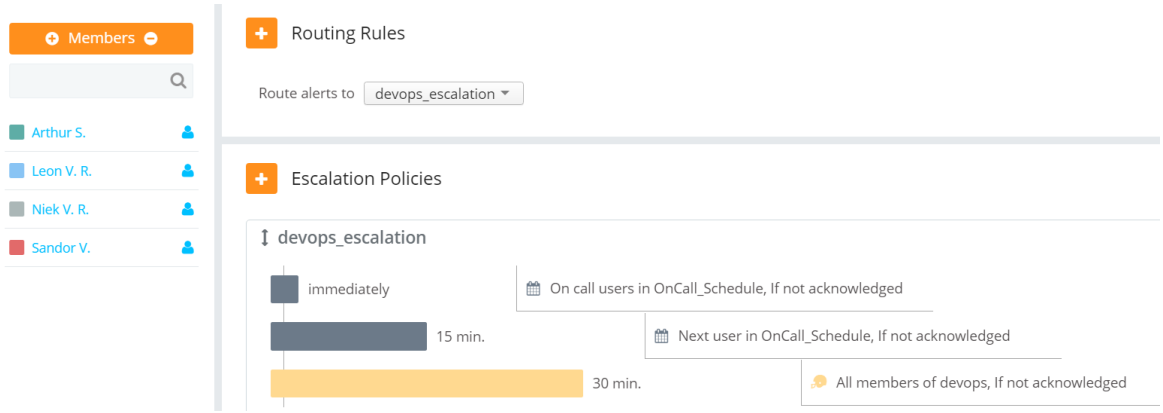
Voordat ik deze infrastructuur testen kon toevoegen moest ik eerst toegang krijgen tot deze platformen. Nadat ik toegang had gekregen heb ik met de stakeholders overlegt of deze checks wel of niet in OpsGenie moesten verschijnen. De stakeholders gaven aan dat wanneer een CPU hoger is dan 80% zij liever niet wakker gebeld willen worden. Wel moest er een mailtje uitgaan naar de algemene mailgroep van afdeling DevOps. Ook hoefde OpsGenie niet te bellen wanneer bijvoorbeeld de CPU de 100% aanraakte of de disk vol zat. Als de website daardoor onbereikbaar wordt dan gaat de uptime check in Nagios af. Door deze checks niet toe te voegen aan OpsGenie voorkom ik dus dat er dubbele alerts uitgestuurd worden.

9.4 Nieuwe OpsGenie

De OpsGenie die in de huidige situatie werd gebruikt was vrij vervuild geraakt. Er waren integraties aanwezig waarvan niemand wist wat deze deden, Het OS-team zat erin maar deed er niets mee en er stond een flink aantal oude alerts open. Om deze redenen is toen in overleg met de stakeholders besloten om deze OpsGenie uit te faseren en een nieuwe toe te voegen.

Ik heb de nieuwe versie van OpsGenie aangemaakt en hier de integraties met Nagios, AWS en Azure aan toegevoegd. Ook heb ik op verzoek van de stakeholders Pingdom en Crashlytics toegevoegd. Daarnaast heb ik vier gebruikers toegevoegd waarvan was afgesproken dat zij in de nieuwe OpsGenie zouden gaan deelnemen.

Vervolgens heb ik deze gebruikers samen toegevoegd aan een team waar een escalatieregel aan was verbonden (figuur 21).



Figuur 21 - Nieuwe OpsGenie

In deze situatie zijn Arthur, Leon, Sandor en Niek om de week on-call. Met de escalatieregel kunnen zij ook buiten het on-call zijn alerts ontvangen, maar dit gebeurt alleen wanneer degene die on-call is het alert niet erkend of closed.

Nadat ik alle gebruikers en integratie had toegevoegd heb ik de gebruikers uit de oude OpsGenie verwijderd en is de nieuwe OpsGenie in gebruik genomen. Hiermee was de nieuwe monitoring situatie officieel live.

Na een paar dagen live te zijn is er nog een kleine wijziging gemaakt in de setup. Alle alerts die Nagios aanmaakte werden op dat moment doorgestuurd naar OpsGenie. Overdag gaven de on-call personen aan dat dit prima was maar, s'nachts wilden zij alleen gebeld worden wanneer belangrijke sites onbereikbaar waren.

Om deze reden heb ik een prioriteitsniveau toegevoegd aan de belangrijke uptime checks. In deze checks heb ik het kenmerk TT1 toegevoegd. Vervolgens heb ik OpsGenie zo ingesteld dat wanneer s'nachts een alert binnenkomt er eerst wordt gekeken of deze TT1 in de naam heeft staan. Is dit zo dan gaat het alert af, anders gaat het alert de eerstvolgende dag om half negen af in OpsGenie.

9.5 Bouwen TicketCreator

Nadat ik een ontwerp had gemaakt voor de TicketCreator ben ik dit vervolgens gaan bouwen. Tijdens het bouwen ben ik tegen enkele dingen aangelopen. In dit hoofdstuk is te lezen wat ik ben tegengekomen en hoe ik dingen als het ophalen van data heb afgehandeld

9.5.1 Verkrijgen OpsGenie data

Ik had besloten om de laatste tien open alerts van OpsGenie op te halen. Deze limiet heb ik gespecificeerd zodat wanneer er alerts open blijven staan of wanneer er opeens heel veel alerts tegelijk aangemaakt worden, de applicatie niet ernstig vertraagd wordt en extreem veel tickets aan maakt. Met dit limiet zullen er maximaal tien tickets per uur aangemaakt worden. In werkelijkheid zal dit getal een stuk lager zijn, omdat het aantal alerts dat gemiddeld per dag wordt aangemaakt in OpsGenie tien is.

Vervolgens ben ik mij eerst gaan verdiepen in de API's van zowel OpsGenie als van JIRA. Uit de OpsGenie API haal ik informatie op en naar de JIRA API stuur ik data.

In de API van OpsGenie vond ik een definitie voor een get alert request. Dit request werkte met REST. Aangezien ik hier nog geen ervaring in had ben ik eerst gaan uitzoeken hoe REST werkt. Ik heb met behulp van google gezocht hoe ik een REST request in C# kon uitvoeren. Ik ben toen een libary genaamd RestSharp tegengekomen. Met deze libary kan je gemakkelijk een Rest request aanmaken. In figuur 22 is te zien hoe dit gebeurt.

```
// Retrieves the ten last created alerts in OpsGenie with Status = open and teams = DevOps and returns a list of these alerts.
// Data is retrieved using the OpsGenie API.
2 references | 0/1 passing | lennart timmers, 12 days ago | 1 author, 1 change
public ICollection<Alert> GetOpsGenieData(string ApiKey)
{
    RestClient client = new RestClient();
    client.BaseUrl = new Uri(ApiKey);
    RestRequest request = new RestRequest(Method.GET);
    IRestResponse response = client.Execute(request);
    string json = response.Content;
    OpsGenieData obj = JsonConvert.DeserializeObject<OpsGenieData>(json);
    List<Alert> alerts = obj.alerts;
    return alerts;
}
```

Figuur 22 - Voorbeeld van een Rest request met RestSharp en een voorbeeld van het converteren van json data in een object.

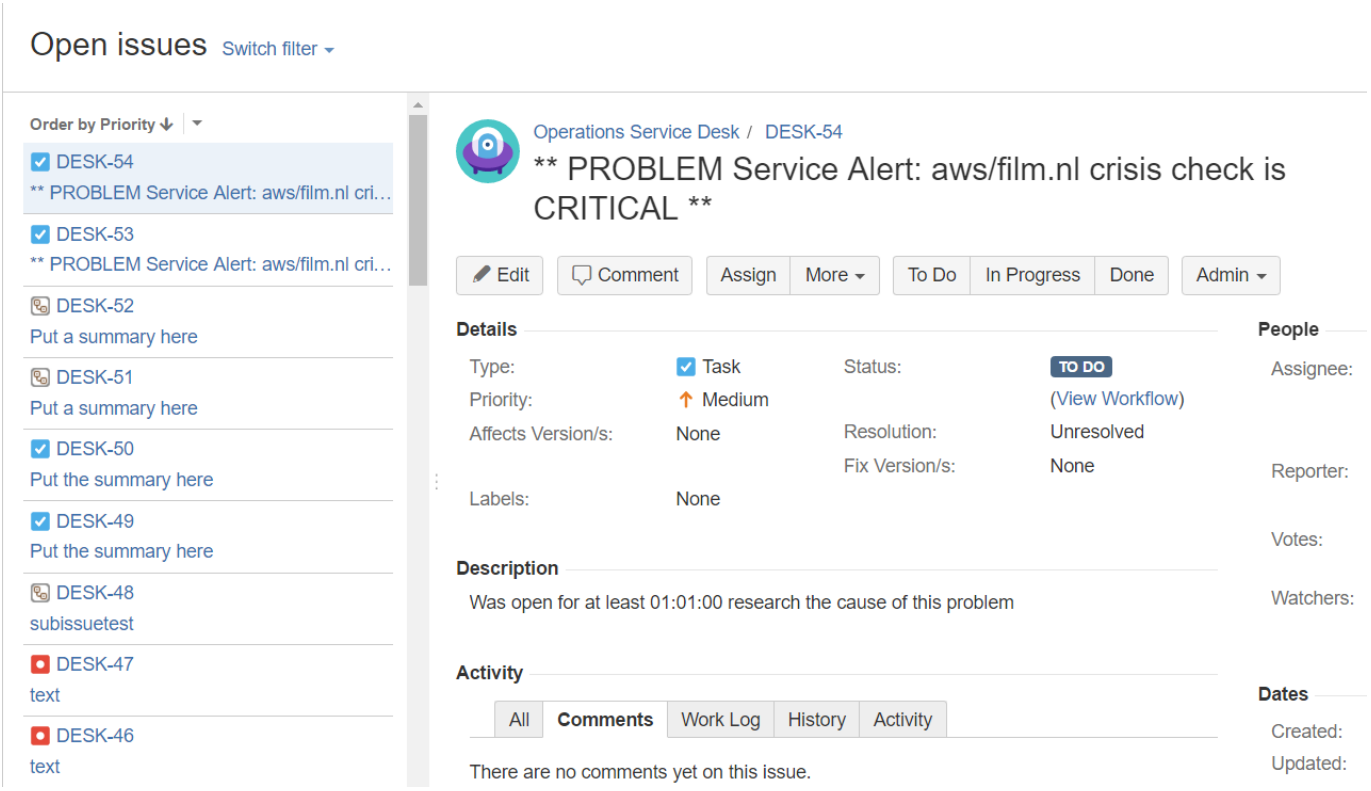
Zoals te zien maak je eerst een client aan waar je vervolgens de url van het API endpoint en het soort request kan invullen. In een post request kan je nog meer waardes invullen. Je kan bijvoorbeeld authenticatie en andere headers toevoegen.

In de documentatie van de API (OpsGenie, 2017) stond ook aangegeven dat de geretourneerde data in JSON-formaat zou zijn. Om dit gemakkelijk te kunnen gebruiken in code is er gebruik gemaakt van een andere libary namelijk Newtonsoft.Json. Deze libary zorgt ervoor dat je gemakkelijk de binnen gekregen data over kan zetten in een C# object.

Aan het eind van de methode wordt er een lijst met alerts teruggegeven. Deze alerts worden uit het object OpsGenieData gehaald. Dit gebeurt omdat de JSON in een formaat terugkomt waar naast een lijst met alerts ook een variabele took in staat. Aangezien de libary alleen de gehele JSON kan deserialiseren wordt dit in het object OpsGenie gestopt. Op deze manier kon ik makkelijk een lijst met data achterhalen.

9.5.2 Aanmaken JIRA-ticket

Om automatisch JIRA-tickets aan te maken heb ik in eerste instantie een eigen versie van JIRA geïnstalleerd. Deze versie heb ik lokaal op mijn eigen pc gedraaid en bevraagt. Doordat ik een bord wat vergelijkbaar was met de echte situatie had gemaakt was dit later ook makkelijker te implementeren. Dit bord is te zien in figuur 23.



Figuur 23 - Lokaal JIRA-bord

Zoals te zien heb ik aardig wat test tickets aangemaakt tijdens het ontwikkelen van de applicatie. De laatste twee tickets zijn gemaakt met echte input vanuit OpsGenie. In het ticket wordt in de titel de naam van het alert toegevoegd en in de beschrijving wordt aangegeven hoe lang het ticket minimaal heeft open gestaan.

In het ticket wordt een minimale tijd gegeven, omdat er wordt gecontroleerd of deze langer dan een uur en korter dan twee uur open heeft gestaan. Dit zodat er geen duplicaten worden toegevoegd in JIRA. Hierdoor kan alleen niet de exacte tijd dat dit ticket heeft open gestaan toegevoegd worden,

maar of het ticket nou één of anderhalf uur heeft opengestaan er moet sowieso onderzocht worden wat hiervan de aanleiding is geweest.

In de code ziet de bovengenoemde vergelijking er zo uit als hieronder in figuur 24.

```
if (DateTime.Today.DayOfYear == createdAt.DayOfYear) //Checks if alert was created on the same day.
{
    TimeSpan timeCreated = createdAt.TimeOfDay;
    TimeSpan now = DateTime.Now.TimeOfDay;
    TimeSpan timeDifference = now - timeCreated; // Calculates how long the alert has been open in hours.
    TimeSpan minTime = new TimeSpan(1, 0, 0);
    TimeSpan maxTime = new TimeSpan(2, 0, 0);

    if (timeDifference >= minTime && timeDifference <= maxTime) //Checks if alert is between 1 and 2 hours old. Method is executed each hour so no duplicates
    {
        JiraTicket JsonTicket = new JiraTicket(a.message, "Was open for at least " + timeDifference + " research the cause of this problem", "DESK", "Task");
        CreateJiraIssue( JsonTicket); //Creates an issue in Jira
    }
}
```

Figuur 24 – Vergelijking van de tijd

Hier wordt eerst gekeken of het alert op dezelfde dag is aangemaakt. Is dit het geval dan wordt er gekeken hoe lang het alert open staat. Als dit tussen de een en twee uur is dan wordt er een ticket aangemaakt op het JIRA-bord.

Dit gebeurt net zoals het ophalen van OpsGenie data met de RestSharp en Newtonsoft libary. Ook de API van JIRA (JIRA, 2017) verwacht een JSON-formaat. Het request ziet er echter wel anders uit omdat het dit keer om een POST gaat waar ook een authenticatie, header en parameter aan toegevoegd worden. In figuur 25 is te zien hoe dit precies gebeurt.

```
private void CreateJiraIssue(JiraTicket ticket)
{
    //Info for logging in into Jira.
    string jiraUrl = "http://localhost:8080/rest/api/2/";
    string user = "testaccount";
    string pass = "geheim123";

    string serialized = JsonConvert.SerializeObject(ticket);
    RestClient rest = new RestClient();
    rest.BaseUrl = new Uri(jiraUrl);
    rest.Authenticator = new HttpBasicAuthenticator(user, pass);
    RestRequest request = new RestRequest("issue", Method.POST);
    request.RequestFormat = DataFormat.Json;
    request.AddHeader("Content-Type", "application/json");
    request.AddParameter("Application/Json", serialized, ParameterType.RequestBody);
    IRestResponse response = rest.Execute(request);

}
```

Figuur 25 - Aanmaken van een JIRA-ticket

Nadat het request is uitgevoerd geeft deze een response terug. Uit dit response is te lezen of het aanmaken van een ticket is geslaagd.

Dit proces laat ik vervolgens elk uur uitvoeren zodat er geen openstaande alerts gemist worden. Dit wordt gedaan met behulp van een Timer die elk uur de code een keer uitvoert.

9.6 Testen TicketCreator

Nadat ik de TicketCreator applicatie gebouwd had wou ik ook graag testen of alles wel naar behoren werkte. Om dit te doen ben ik gaan kijken wat ik kon testen. De methodes die ik kon testen waren het ophalen van data vanuit OpsGenie, het aanmaken van een JIRA-ticket en de vergelijking die kijkt hoe lang het ticket open staat. Aangezien twee van deze methodes gebruik maakte van een API verwachte ik eigenlijk geen problemen hiermee, omdat deze API ook door het bedrijf zelf getest wordt. Om deze reden heb ik toen besloten om geen testen uit te voeren voor het ophalen van data vanuit OpsGenie.

Wel heb ik besloten om het aanmaken van JIRA-tickets te testen. Dit omdat JIRA een bedrijf kritisch onderdeel is voor TamTam. Mocht mijn tool bijvoorbeeld per ongeluk JIRA breken of allemaal tickets in verkeerde borden aanmaken dan zou dit niet gewaardeerd worden. Om deze reden heb ik besloten om dit ondanks dat het een API is alsnog te testen.

Ook test ik de vergelijking die kijkt hoe lang een ticket open staat. Deze vergelijking heb ik besloten te testen met unit testen.

9.6.1 Testen aanmaken tickets

Om te testen of het aanmaken van tickets correct verloopt heb ik eerst de attributen bepaald. Per attribuut geef ik vervolgens aan welke waardes geldig zijn. Ook heb ik een speciale opmerking hierbij gezet die een restrictie aangeeft die niet met de syntax te maken heeft. De equivalentieklassen zijn hieronder in te zien (figuur 26).

Attribuut	**Geldige klassen**	**Ongeldige klassen**
Summary	>=1 Tekstlengte <=255	1< Tekstlengte >255
IssueType	Aanwezige standard issuetypes	Niet bestaande types en sub-task issuetypes
Projectkey	DESK	Testboard waar gebruiker geen rechten heeft en ongeldige waardes

Figuur 26 - Equivalentieklassen

- *"Er wordt een speciaal account gebruikt voor het toevoegen van deze tickets. Deze account mag alleen op het project DESK nieuwe tickets toevoegen. Wanneer deze op een ander bestaand project een ticket toevoegt mag dit niet slagen."*

Vervolgens ben ik aan de slag gegaan met het opstellen van logische en fysieke testgevallen. Deze zijn gemaakt met behulp van de equivalentieklassen. Elke klasse komt een keer voor in de testgevallen. Ik combineer nooit twee ongeldige klassen omdat je dan niet zeker weet waardoor de test faalt (als dit gebeurt). Omdat de tabellen vrij groot zijn zal ik een van de zeven gevallen hier bijvoegen als voorbeeld. De andere gevallen zijn in te zien in de bijlage H.

Het eerste voorbeeld is een logisch testgeval en het tweede voorbeeld is een fysiek testgeval.

Testnummer	Summary	IssueType	Projeckey	Verwacht resultaat
1	>=1 Tekstlengte <=255	Aanwezige standard issuetype	DESK	Ticket aangemaakt in Desk met als type Task, summary “test” en description “test”

Figuur 27 - Logisch testgeval

Zoals in figuur 27 te zien wordt bij het logische testgeval aangegeven wat voor waarde er ingevuld moet worden en wat het verwachte resultaat is. Deze waardes komen uit de equivalentieklassen. Vervolgens geef ik aan wat het verwachte resultaat is van de test.

1	Test	Task	DESK	Ticket aangemaakt in Desk met als type Task, summary “test”

Figuur 28 - Fysiek testgeval

In het fysieke testgeval (figuur 28) geef ik vervolgens aan welke waardes er daadwerkelijk ingevoerd moeten worden. Aan de hand van de fysieke testgevallen heb ik vervolgens simpele unittesten geschreven. Deze testen zijn zichtbaar in figuur 29.

```
AlertHandler handler = new AlertHandler();
[TestMethod]
| 0 references | 0 changes | 0 authors, 0 changes
public void CreateTicket()
{
    JiraTicket ticket = new JiraTicket("Test","", "DESK", "Task");
    Assert.IsTrue(handler.CreateJiraIssue(ticket) == true);
}
[TestMethod]
| 0 references | 0 changes | 0 authors, 0 changes
public void CreateTicketEmptySummaryString()
{
    JiraTicket ticket = new JiraTicket("", "", "DESK", "Task");
    Assert.IsTrue(handler.CreateJiraIssue(ticket) == false);
}
[TestMethod]
| 0 references | 0 changes | 0 authors, 0 changes
public void CreateTicketSummaryStringToLong()
{
    JiraTicket ticket = new JiraTicket("asdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsd
    Assert.IsTrue(handler.CreateJiraIssue(ticket) == false);
}
[TestMethod]
| 0 references | 0 changes | 0 authors, 0 changes
public void CreateTicketIssueTypeFake()
{
    JiraTicket ticket = new JiraTicket("test ", "", "DESK", "NepIssueType");
    Assert.IsTrue(handler.CreateJiraIssue(ticket) == false);
}
[TestMethod]
| 0 references | 0 changes | 0 authors, 0 changes
public void CreateTicketIssueSubTask()
{
    JiraTicket ticket = new JiraTicket("test ", "", "DESK", "Sub-task");
    Assert.IsTrue(handler.CreateJiraIssue(ticket) == false);
}
[TestMethod]
| 0 references | 0 changes | 0 authors, 0 changes
public void CreateTicketKeyNoAcces()
{
    JiraTicket ticket = new JiraTicket("test ", "", "TES", "Task");
    Assert.IsTrue(handler.CreateJiraIssue(ticket) == false);
}
[TestMethod]
| 0 references | 0 changes | 0 authors, 0 changes
public void CreateTicketKeyDoesntExist()
{
    JiraTicket ticket = new JiraTicket("test ", "", "UNTA", "Task");
    Assert.IsTrue(handler.CreateJiraIssue(ticket) == false);
}
```

Figuur 29 - Unittesten gebaseerd op fysieke testgevallen

Zes van de zeven testen slaagden meteen. Eén test faalde echter. Dit was de test met een andere projectnaam te zien in het fysieke testgeval in figuur 30.

6	Test	Task	TES	Ticket is niet aangemaakt omdat gebruiker geen toegang tot TES heeft

Figuur 30 - Bijbehorend fysiek testgeval

Deze test verwachte als antwoord dat het aanmaken van het ticket mislukt was, omdat de gebruiker geen toegang heeft tot het bord TES. Het aanmaken van dit ticket lukte echter wel. Toen ik ging kijken hoe dit kwam ontdekte ik dat de permissies van de testgebruiker niet goed ingesteld stonden. Ik heb dit toen aangepast in de JIRA-instellingen en daarna slaagde de test (figuur 31).

```
[TestMethod]
0 references | 0 changes | 0 authors, 0 changes
public void CreateTicketKeyNoAcces()
{
    JiraTicket ticket = new JiraTicket("test ", "", "TES", "Task");
    Assert.IsTrue(handler.CreateJiraIssue(ticket) == false);
}
```

Figuur 31 - Geslaagde test na wijziging in JIRA

Achteraf bleek het dus inderdaad verstandig om hiervoor testen aan te maken ook al was de fout gemaakt door onoplettendheid en niet door foute code. Doordat dit met deze test werd geconstateerd is het wijzigen van de settings niet vergeten met de overgang naar de live situatie.

9.6.2 Testen vergelijking openstaand ticket

Naast het testen van het tickets aanmaken heb ik ook getest of de vergelijking van een openstaand ticket wel correct functioneerde. Deze functie was verantwoordelijk voor het besluit om een ticket wel of niet aan te maken. Het was dus van belang dat deze functie goed werkte omdat er anders onterecht tickets aangemaakt konden worden.

Dit heb ik getest met behulp van drie unittesten (figuur 32). Deze testen gaan stuk voor stuk een ander gedeelte van de vergelijking in. De eerste test krijgt een tijdswaarde mee van 1 uur en 1 minuut openstaan. In dit geval moet een Ticket object teruggegeven worden. De andere twee testen hebben als tijdswaarde dat ze 3 uur en 25 uur en 1 minuut openstaan. Deze testen moeten allebei als waarde null teruggeven. In figuur 32 is een voorbeeld gegeven van een van deze testen.

```
[TestMethod]
0 references | 0 changes | 0 authors, 0 changes
public void CheckAlertTestReturnTicket()
{
    AlertHandler handler = new AlertHandler();
    //same day between 1 and 2 hours
    DateTime dt = DateTime.UtcNow.AddMinutes(-61);
    //converts it to nanoseconds unix since this is the same value as OpsGenie supplies
    long unixTimestamp = (long)(dt.Subtract(new DateTime(1970, 1, 1))).TotalMilliseconds * 1000000;
    Alert a = new Alert()
    {
        acknowledged = true,
        owner = "",
        teams = new List<string>() { "devops" },
        count = 1,
        message = "** PROBLEM Service Alert: azure/www.tno.nl check op Contact via SSL en nl is CRITICAL **",
        isSeen = true,
        tags = null,
        createdAt = unixTimestamp,
        alias = "9f74fdc6-8718-48a0-bcf4-00e3912c9b52",
        id = "9f74fdc6-8718-48a0-bcf4-00e3912c9b52",
        status = "open",
        tinyId = "438",
        updatedAt = 1493209765552010606
    };
    JiraTicket ticket1 = handler.CheckAlert(a);

    //same day but not between 1 and 2 hours
    DateTime dt2 = DateTime.UtcNow.AddHours(-3);
    long unixTimestamp2 = (long)(dt2.Subtract(new DateTime(1970, 1, 1))).TotalMilliseconds * 1000000;
    a.createdAt = unixTimestamp2;
    JiraTicket ticket2 = handler.CheckAlert(a);


    Assert.IsTrue(ticket1 != null);
}
```

Figuur 32 - Uittest vergelijking openstaand ticket

Alle drie de testen slaagden meteen en ik concludeerde daaruit dat de functie correct werkt.

9.7 Ontwerpen processen

Om de nieuwe monitoring situatie ook daadwerkelijk te gaan gebruiken binnen het bedrijf, heb ik een aantal processen ontworpen die aantonen hoe hiermee om moet worden gegaan. Wanneer een nieuw project van start gaat dan moet dit ook opgenomen worden in de monitoring en als een project out of service gaat dan moet de monitoring hiervan ook weer verwijderd worden.

In deze processen wordt duidelijk aangegeven wie voor welk onderdeel verantwoordelijk is en wat diegene moet doen. Op de volgende pagina is het gehele proces voor het inrichten van de monitoring weergegeven. Ik zal hier kort bespreken wat er allemaal wordt gedaan.

9.7.1 Nieuw project

Voor een nieuw project is het volgende proces opgesteld (figuur 33).

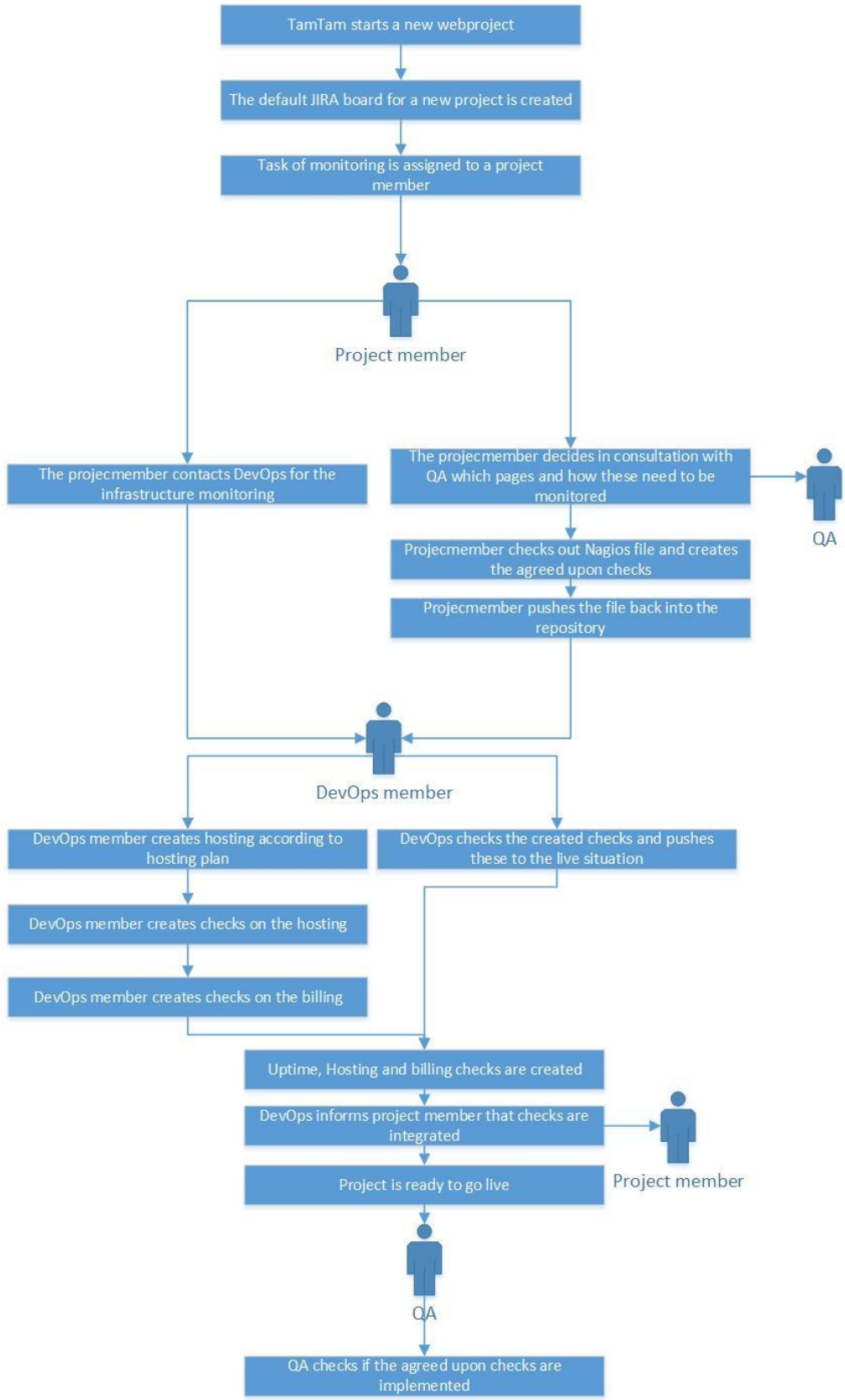
Wanneer een nieuw project wordt opgestart is er een standaard ingericht JIRA-bord wat gebruikt wordt. Dit bord heeft een task voor het inrichten van de monitoring. Deze task wordt aan iemand die deel is van het team toegewezen.

Vervolgens gaat die persoon twee dingen doen. Het eerste is contact opnemen met afdeling DevOps met het verzoek of zij de infrastructuur monitoring kunnen gaan inrichten. Vanaf dat moment wordt dit de taak van DevOps. Ook gaat het project lid in overleg met QA afstemmen wat er gemonitord moet gaan worden op de nieuwe website. Als zij tot een overeenstemming hiervoor zijn gekomen gaat het project lid aan de slag met het aanmaken van deze checks en wanneer hij/zij hiermee klaar is pushed diegene de veranderingen naar de repository.

Dan komen de taken bij DevOps te liggen. Deze creëren een hosting omgeving aan de hand van het afgesproken plan en besluiten aan de hand van dat plan hoe de infrastructuur checks eruit moeten zien. Ook worden er standaard checks op de rekeningen ingesteld als de hosting bij Azure of AWS wordt afgenomen. Daarnaast controleert DevOps ook de aangemaakte checks in Nagios. Wanneer ze geen fouten zien in de nieuwe checks en de site live is pushen zij dit door naar de live situatie van Nagios.

Wanneer die stap is uitgevoerd dan zijn alle benodigde checks aangemaakt. DevOps informeert dan het project lid dat de monitoring is ingericht. Wanneer het project vervolgens klaar is om live te gaan wordt eerst QA nog een keer geraadpleegd. Zij kijken dan of alles daadwerkelijk klaar staat voor de livegang. Hier zit onder andere een controle op de monitoring in. QA kijkt voor de livegang of de afgesproken checks ook daadwerkelijk zijn aangemaakt in Nagios en overlegt met DevOps of de infrastructuur monitoring is ingericht.

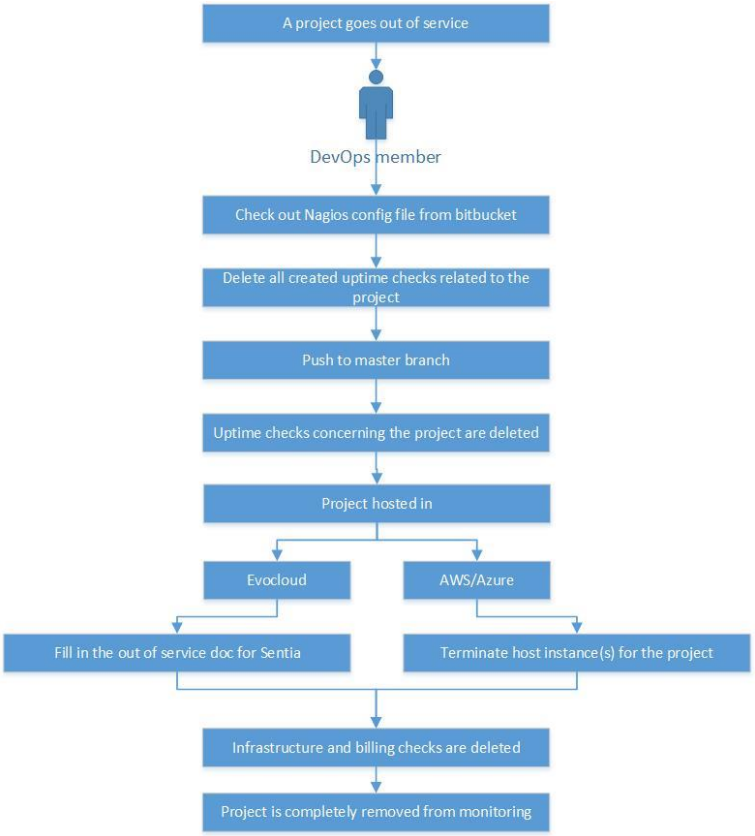
Is alles in orde dan wordt het project vervolgens live gezet en is het project af en de monitoring ingericht.



Figuur 33 - Proces nieuw project

9.7.2 Out of serviceproject

Wanneer een project out of service gaat moeten de monitoringchecks hiervoor verwijderd worden. Dit wordt afgehandeld door afdeling DevOps. Zij hebben namelijk de meeste kennis van alle aangemaakte checks en hebben ook toegang tot de locaties waar deze checks verwijderd moeten worden. Hoe dit wordt gedaan is zichtbaar in figuur 34.



Figuur 34 - Proces out of serviceproject

Een project gaat out of service. Wanneer dit gebeurt wordt afdeling DevOps ingelicht dat dit project niet meer gehost wordt. Het eerste wat wordt gedaan is het verwijderen van de Nagios checks. Dit omdat wanneer de infrastructuur eerst verwijderd wordt al deze checks zullen afgaan.

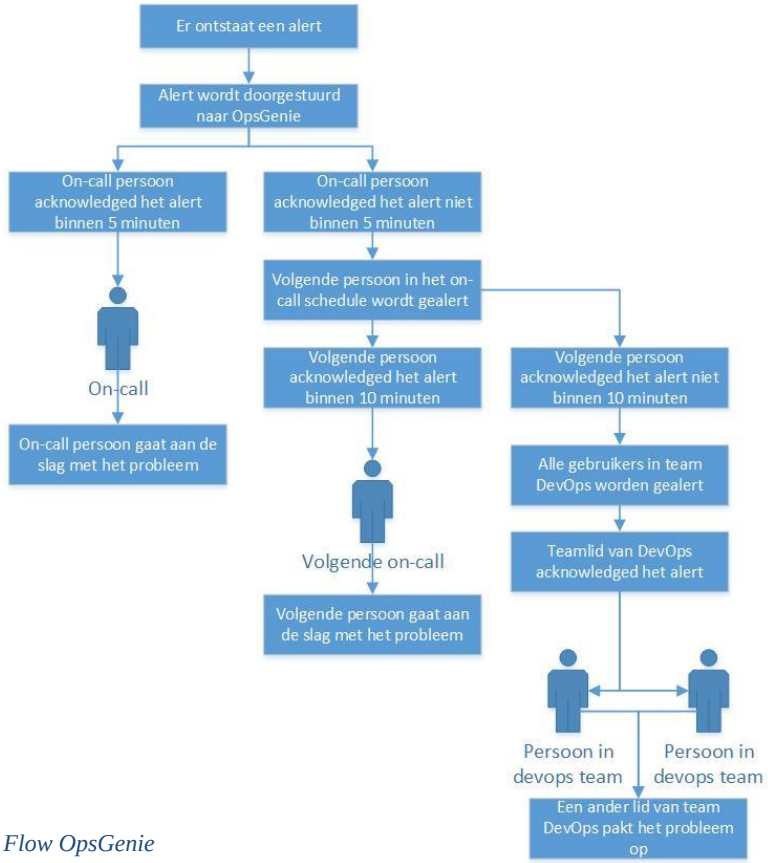
Nadat de checks allemaal verwijderd zijn wordt gekeken naar de hosting. Staat de omgeving in EvoCloud dan moet er een out of service formulier ingevuld worden voor Sentia. Zij zorgen dan dat de infrastructuur monitoring uit Nagios wordt verwijderd. Dit is onderdeel van het samenwerkingscontract met Sentia.

Wordt de site gehost op AWS of op Azure dan wordt de instantie waarop het project draait geselecteerd. Van deze instantie wordt dan een back-up gemaakt (standaardprocedure) en deze wordt vervolgens afgesloten. Dit zorgt er ook voor dat de monitoring regels automatisch verwijderd worden.

Er hoeven geen acties op de omgeving van OpsGenie ondernomen te worden. Dit omdat de andere tooling daarmee integreert en wanneer de checks daaruit weg zijn, deze automatisch ook uit OpsGenie verwijderd zijn.

9.7.3 Handelswijze OpsGenie

Voor de nieuwe OpsGenie situatie heb ik ook een flow gemaakt (figuur 35). Deze flow is gemaakt voor mensen die nieuw toegevoegd worden aan OpsGenie, zodat zij niet per ongeluk andere collega's storen. Wanneer een alert namelijk niet erkend (dit heet acknowledgen in OpsGenie) wordt, krijgt de volgende persoon in het schema ook een alert. Overdag is dit niet erg maar s'nachts kan dit erg vervelend zijn voor collega's.



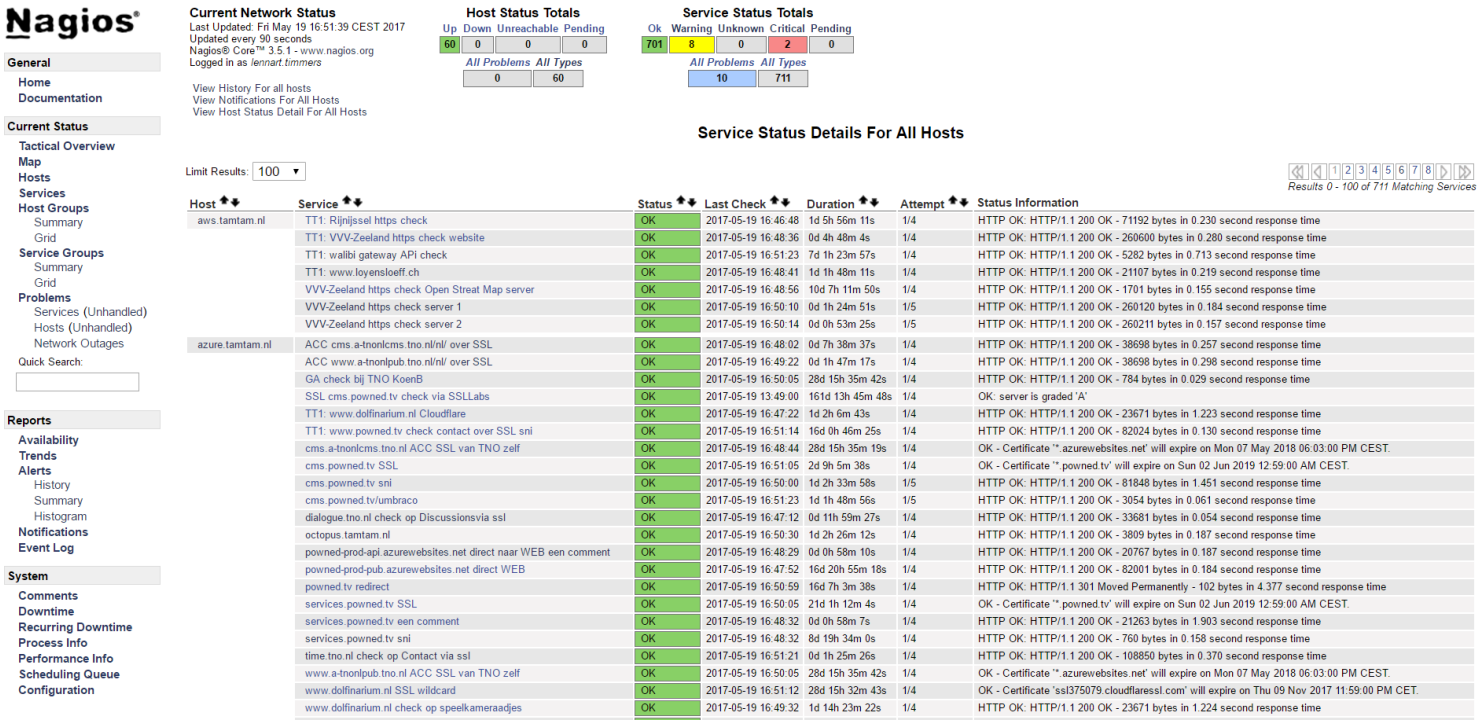
Figuur 35 – Flow OpsGenie

9.7.4 Presenteren processen

De nieuwe processen heb ik ook weer gepresenteerd aan de stakeholders. Uit dit overleg zijn toen een aantal actiepunten naar voren gekomen. Een paar van deze opmerkingen gingen over een extra stap toevoegen in de nieuwe processen. De andere gingen over OpsGenie. In deze vergadering is afgesproken dat personen die on-call zijn altijd het alert erkennen zodra ze deze zien. Daarnaast is ook afgesproken dat de telefoonnummers die OpsGenie gebruikt toegevoegd worden aan de favorieten van ieder on-call persoon. Dit zodat de telefoon ook s'nachts af kan gaan als diegene on-call is of als er een escalatieregel heeft plaats gevonden.

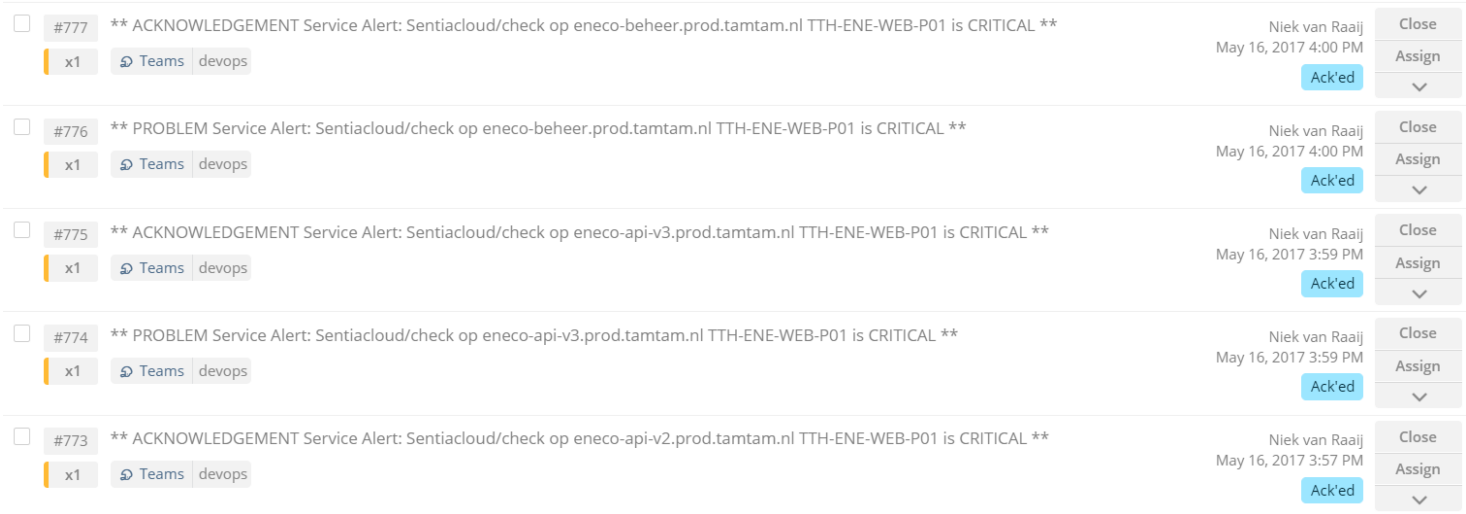
9.8 Eindproduct

Het eindproduct is in actief gebruik genomen door TamTam. Nagios integreert op een slimme manier met OpsGenie. Hieronder volgen een aantal screenshots met uitleg om duidelijk te maken hoe het eindresultaat precies werkt.



Figuur 36 - Nieuwe situatie monitoring via Nagios

De monitoring van de sites wordt uitgevoerd door Nagios. In figuur 36 is te zien welke checks worden uitgevoerd door Nagios. Ook is te zien dat bij sommige checks TT1 in de beschrijving staat en bij andere niet. Voor alle checks in figuur 36 geld dat wanneer deze rood worden en in een HARD-state critical status komen te staan er een mail naar OpsGenie verstuurd wordt. De services staan in een HARD-state als het opvragen x aantal keer is mislukt (dit verschilt per check).



Figuur 37 - Nieuwe situatie van OpsGenie

In OpsGenie komen deze alerts vervolgens binnen. Deze alerts kijken via het on-call schema (figuur 21) welke persoon gecontacteerd moet worden en doen dit vervolgens door middel van mail, sms, push-notificaties en door te bellen. Wanneer deze contactpersoon het alert heeft erkend zorgt hij/zij ervoor dat het probleem wordt opgelost.

Komt het alert binnen buiten kantoortijden (08:00 – 18:00) dan wordt er eerst gekeken of er TT1 in de naam van het alert staat. Is dit het geval dan geeft het alert meteen een melding. Anders dan wordt dit uitgesteld tot de eerstvolgende dag om half negen s'ochtends.

9.9 Reflectie sprint 4

Tijdens deze sprint heb ik veel nieuwe kennis opgedaan op het gebied van Nagios. Daarnaast heb ik ook kennis opgedaan van het werken met Rest request en het ontwerpen van processen.

Deze sprint had ik een extra week tot beschikking vanwege het aantal beschikbare weken vanuit school. Door deze extra tijd heb ik meer dan normaal weten op te leveren. Het implementeren van de nieuwe situatie is vrij goed verlopen. Ik was in eerste instantie alleen vergeten om de uit de oude OpsGenie de gebruikers te verwijderen die in de nieuwe waren geplaatst. Hierdoor zijn dubbele kosten in rekening gebracht. De volgende keer moet ik bij dit soort migraties een reminder instellen zodat ik dit niet meer vergeet.

Het bouwen van de TicketCreator ging ook goed. Doordat ik een eigen testomgeving had aangemaakt op mijn lokale computer kon ik vrijuit testen zonder dat ik iets kapot maakte in JIRA. Het enige lastige was de overgang naar de JIRA-omgeving van TamTam. Ik moest hier een ander account voor regelen. Het bleek echter vrij lastig om een JIRA-admin te vinden die tijd had om mij te helpen. Nadat dit eenmaal was gelukt heb ik nog enkele keren de code getest op mijn afstudeerbord in de TamTam JIRA.

Het ontwerpen van de processen ging ook prima. Tijdens de presentatie kwamen er nog enkele verbeterpunten naar voren. Deze heb ik na afloop in de processen opgenomen waardoor deze vervolgens ook naar de wensen van de stakeholders waren.

Aan het eind van deze sprint had ik ook het product opgeleverd wat vanuit TamTam van mij werd verwacht. Omdat ik echter pas iets over de helft was van mijn afstudeertraject, is toen besloten dat ik door ging werken aan nieuwe inzichten die door mijn project aangetoond werden. Ik ben hiermee begonnen in sprint 5 en in het volgende hoofdstuk valt hier meer over te lezen.

9.10 Sprint 4 in het kort

In dit hoofdstuk vat ik kort samen wat opgeleverd moest worden, of dit is gelukt en hoe dit is gebeurd.

Op te leveren	Opgeleverd?
Nieuwe situatie	Ja, eerst leren omgaan met Nagios door hier een testserver voor op te zetten. Vervolgens testen overgezet en laten integreren met OpsGenie als dit voor desbetreffende test vereist was.
Bouwen en testen TicketCreator	Ja, TicketCreator gebouwd aan de hand van het ontwerp uit de vorige sprint. Vervolgens dit getest met behulp van een testdocument.
Ontwerpen processen	Ja, processen voor de nieuwe monitoringsituatie ontworpen deze gepresenteerd en kleine wijzigingen doorgevoerd. Vervolgens de processen gedocumenteerd op Confluence.

10. Bepalen aanvullende opdracht

Nadat sprint 4 was afgelopen was er een werkend product opgeleverd en was mijn opdracht succesvol afgerond. Ik ben toen in overleg gegaan met mijn Opdrachtgever Leon en buddy Niek en hierin hebben we besproken welke andere werkzaamheden ik ging verrichten. Tijdens deze bespreking kwamen er een aantal punten naar voren waaraan de aanvullende opdracht moest voldoen.

- Te voltooien binnen de resterende tijd.
- Moet aansluiting hebben op vorige opdracht.
- Moet passen bij de geselecteerde beroepstaken.

Aan de hand van deze eisen is de volgende opdracht ontstaan.

10.1 Probleemstelling

Voor de monitoring waren aan het eind van mijn opdracht door mij processen gemaakt (hoofdstuk 9.7). Om deze echter na te streven moest voor elk nieuw of out of serviceproject een ticket met subtaken aangemaakt worden in JIRA.

Naast de standaard subtaken voor monitoring zijn er nog een aantal taken die altijd uitgevoerd moeten worden in het geval van een nieuw of out of serviceproject. Ook zijn er taken die bij specifieke oplossingen altijd uitgevoerd moeten worden. Het aanmaken van deze taken kost echter tijd en soms wordt een taak vergeten. Deze wordt dan later wel toegevoegd, maar dit gooit de planning door de war.

10.2 Doelstelling

Het doel van de opdracht was om de intake van nieuwe projecten en het afsluiten van oude projecten te automatiseren. Hierdoor moest de afdeling DevOps meer tijd krijgen om gelijk aan de slag te gaan met het project in plaats van zelf eerst een JIRA-ticket met subtaken in te moeten richten.

Dit moest mogelijk worden door de servicedesk een formulier in te laten vullen en insturen. Wanneer dit formulier wordt opgestuurd moesten er automatisch taken van een out of serviceproject aangemaakt worden in het JIRA-bord van DevOps.

Ook moesten project leads in een intake formulier een aanvraag kunnen doen voor de hosting van een nieuw project. Dit formulier moest verschillende pakketten bevatten zodat de lead een keuze kan maken tussen de verschillende soorten hosting (Azure, AWS, EvoCloud).

10.3 Beoogd resultaat

De applicatie moest bestaan uit twee belangrijke functionaliteiten. De eerste functionaliteit is het aanmaken van de tickets op het goede JIRA-bord inclusief sub tickets. Dit moest kunnen door middel van twee verschillende formulieren. Het eerste formulier is voor nieuwe projecten en het tweede is voor de out of serviceprojecten.

Deze formulieren zijn intern (op IP-adressen van TamTam) zichtbaar voor de medewerkers op devops.tamtam.nl. Zij moeten in het formulier voor nieuwe projecten een keuze kunnen maken uit een aantal pakketten. Deze pakketten moeten aangemaakt kunnen worden door admins (afdeling DevOps). Zij moeten hier producten en JIRA-tickets aan kunnen toevoegen. De producten worden toegevoegd om collega's inzicht te geven in het pakket. De tickets worden naast de standaard tickets toegevoegd aan het JIRA-bord.

Het out of service formulier is iets simpeler. In dit formulier moet een werknemer een project kunnen aanduiden dat out of service gaat. Voor out of serviceprojecten zijn een aantal standaard stappen die doorlopen moeten worden. Deze stappen moesten automatisch aangemaakt worden in een JIRA-bord wanneer de gebruiker het out of service formulier invult. Ook deze standaard tickets kunnen door admins toegevoegd of verwijderd worden.

De tweede functionaliteit was het beheren van standaard packages met bijbehorende producten en tickets. Dit moesten admins doen in een beheerpaneel waar ze packages kunnen aanmaken, verwijderen en bewerken.

Voor deze opdracht was afgesproken dat de volgende producten opgeleverd zouden worden:

- Requirementsdocument
- Ontwerp nieuwe situatie
- Implementatie nieuwe situatie
- Testdocument nieuwe situatie
- Overdrachtsdocument

11. Sprint 5

In sprint 5 ben ik begonnen met de aanvullende opdracht namelijk, het bouwen van Devops.TamTam.nl. Ik ben de requirements gaan achterhalen bij afdeling DevOps. Deze implementatie wordt voor hen geschreven en zij zijn dus de stakeholders. Nadat ik de requirements had heb ik een ontwerp gemaakt voor de nieuwe applicatie. Dit ben ik vervolgens gaan bouwen in het ASP.NET framework. Toen ik hiermee klaar was heb ik de functionaliteiten in de website getest.

11.1 Verzamelen requirements

Voor het verzamelen van de requirements heb ik een soortgelijke aanpak gehanteerd als bij het opstellen van het eerste requirementsdocument. Ik heb hiervoor met de volgende mensen gesprekken gevoerd.

- Niek van Raaij (DevOps engineer)
- Leon van Rees (Teamlead OS & DevOps)

Deze personen zullen beide uiteindelijk deze applicatie zelf gaan gebruiken of hiernaar doorverwijzen. Ik heb in totaal drie gesprekken gehouden over devops.tamtam.nl. Een gesprek met Leon van Rees en Niek van Raaij. In dit gesprek zijn de meeste requirements vastgesteld. De andere twee gesprekken waren samen met Niek van Raaij. Hierin werden nog een aantal aanvullende requirements naar voren gebracht.

Ook in dit requirementsdocument heb ik gebruik gemaakt van de prioritering methode MoSCoW. Deze requirements waren eveneens vooral must have requirements. Er zaten echter ook een paar should have's bij. Per requirement is tussen haakjes met dikgedrukte letters aangegeven wat voor prioriteit deze had (figuur 38).

Req	Requirement	Bron
1	Het systeem moet tickets kunnen aanmaken in het HOSTING bord van TamTam. **(must have)**	Gesprek Niek, Leon

Figuur 38 - Voorbeeld van devops.tamtam.nl requirement

De meeste geconstateerde requirements bestonden uit CRUD-functionaliteiten. Deze afkorting staat voor Create, Read, Update en Delete. Hieronder is een voorbeeld gegeven van een onderdeel waar CRUD-functionaliteit voor gemaakt moest worden (figuur 39).

Req	Requirement	Bron
6	Een admin moet pakketten kunnen aanmaken. **(must have)**	Gesprek Niek
7	Een admin moet pakketten kunnen bewerken. **(should have)**	Gesprek Niek
8	Een admin moet pakketten kunnen verwijderen. **(must have)**	Gesprek Niek
9	Pakketten moeten getoond kunnen worden. **(must have)**	Gesprek Niek

Figuur 39 - Voorbeeld van CRUD requirements

Naast de functionele requirements zijn er ook nog een niet-functioneel en een business requirement opgesteld. Het niet functionele requirement gaf aan dat de formulieren niet te ingewikkeld mochten zijn. Het ging hier met name om de intake en het out of service formulier. Deze formulieren moesten ook ingevuld kunnen worden door de servicedesk en deze personen hebben niet allemaal technische kennis.

Het business requirement gaf aan dat de te ontwikkelen applicatie ook daadwerkelijk in gebruik moest worden genomen. Dit zodat de intake van nieuwe projecten en out of serviceprojecten minder tijd in beslag nemen waardoor afdeling DevOps meer tijd overhoudt voor andere zaken.

Alle requirements en de bron hiervan zijn terug te vinden in het requirements document (bijlage I).

11.2 Ontwerpen webapplicatie

Nadat ik de requirements had vastgesteld en dit akkoord was vanuit de stakeholders ben ik begonnen met een ontwerp maken voor de webapplicatie. Dit ontwerp is gebruikt als basis voor het bouwen van de applicatie. Aangezien er echter weinig tijd was is dit ontwerp niet meer aangepast wanneer er tijdens het programmeren andere keuzes werden gemaakt. Dit zijn echter niet grote veranderingen in keuzes geweest, maar bijvoorbeeld het toevoegen van een extra methode of parameter.

De webapplicatie heb ik ontworpen met het MVC framework. In het MVC framework heb je een Model, View en een Controller. Deze drie componenten hebben alle drie hun eigen functies. Het Model geeft de logica voor de businessdata aan, de View bevat de logica voor het userinterface en in de Controller wordt de input van de gebruikers verwerkt en de view aan de hand hiervan vervolgens geüpdatet.

Om het diagram overzichtelijker te houden heb ik besloten om de views niet te tekenen. Deze zijn ook minder interessant dan de overige klassen waarin de data zich voortbeweegt. De views worden alleen gebruikt door de gebruiker om de data in te voeren en door de controller om ingevoerde data aan de gebruiker te tonen.

Zoals te zien in het ontwerp (figuur 39) heb ik gebruik gemaakt van services. In deze services programeer ik de business logica van de applicatie. Hierdoor blijven de controllers “dom” en voeren deze geen functionaliteiten uit die niet bij hun taakomschrijving passen.

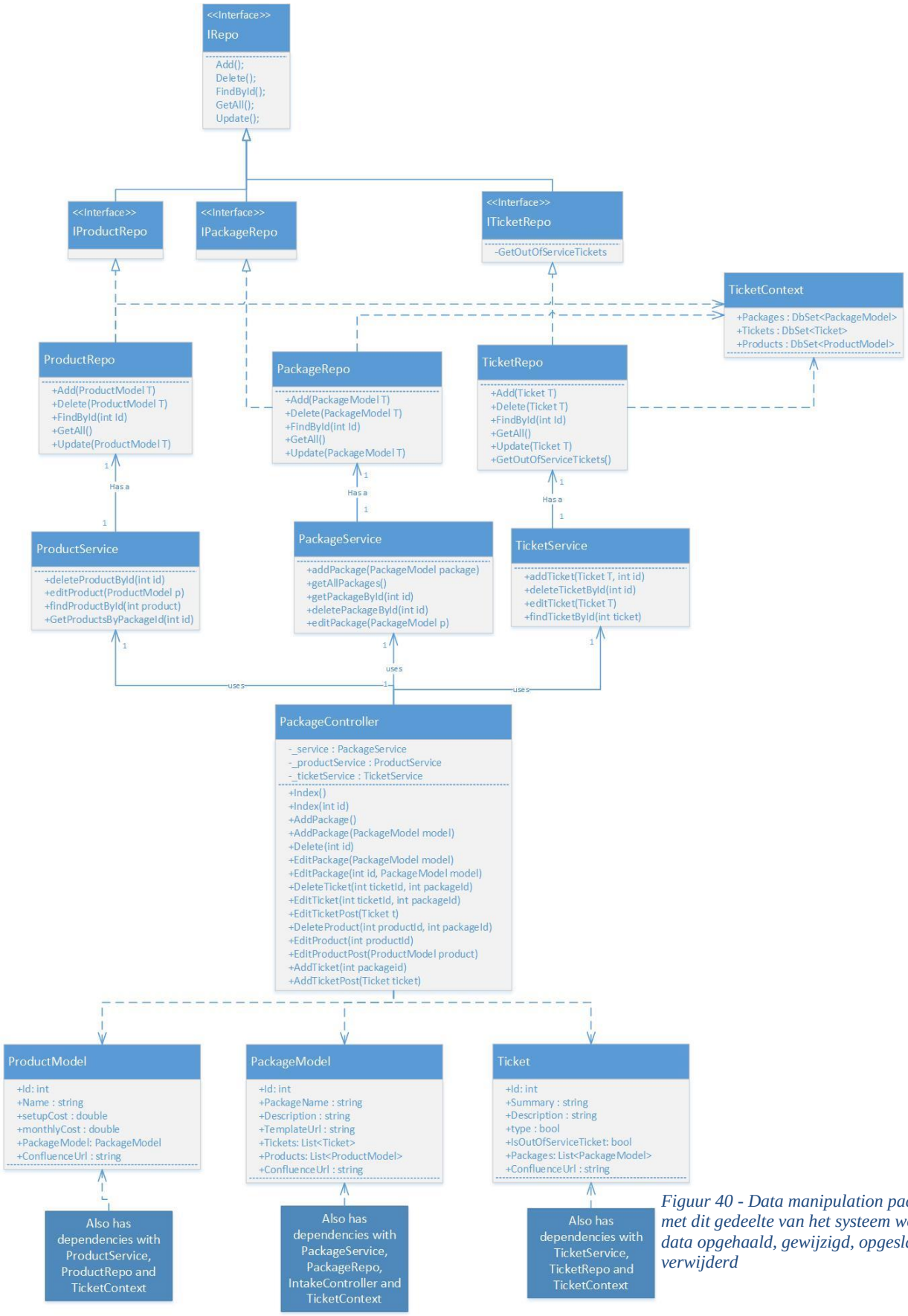
Daarnaast heb ik het Entity framework gebruikt om de database te bouwen en bevragen. Om het Entity framework te implementeren moet je een klasse hebben die overerft van DBcontext, een klasse uit het Entity framework. De klasse die in mijn diagram deze geval deze rol vervuld is de TicketContext klasse.

Op de volgende pagina is het ontwerp van de data access laag te zien (figuur 40). Dit ontwerp geeft aan hoe de CRUD-functionaliteiten in de applicatie verwerkelijkt zijn.

Zoals te zien wordt er gebruik gemaakt van drie verschillende modellen. Deze modellen worden door de controller gebruikt om informatie naar de services te sturen. De services verwerken vervolgens de ontvangen informatie en sturen dit naar de bijbehorende repo.

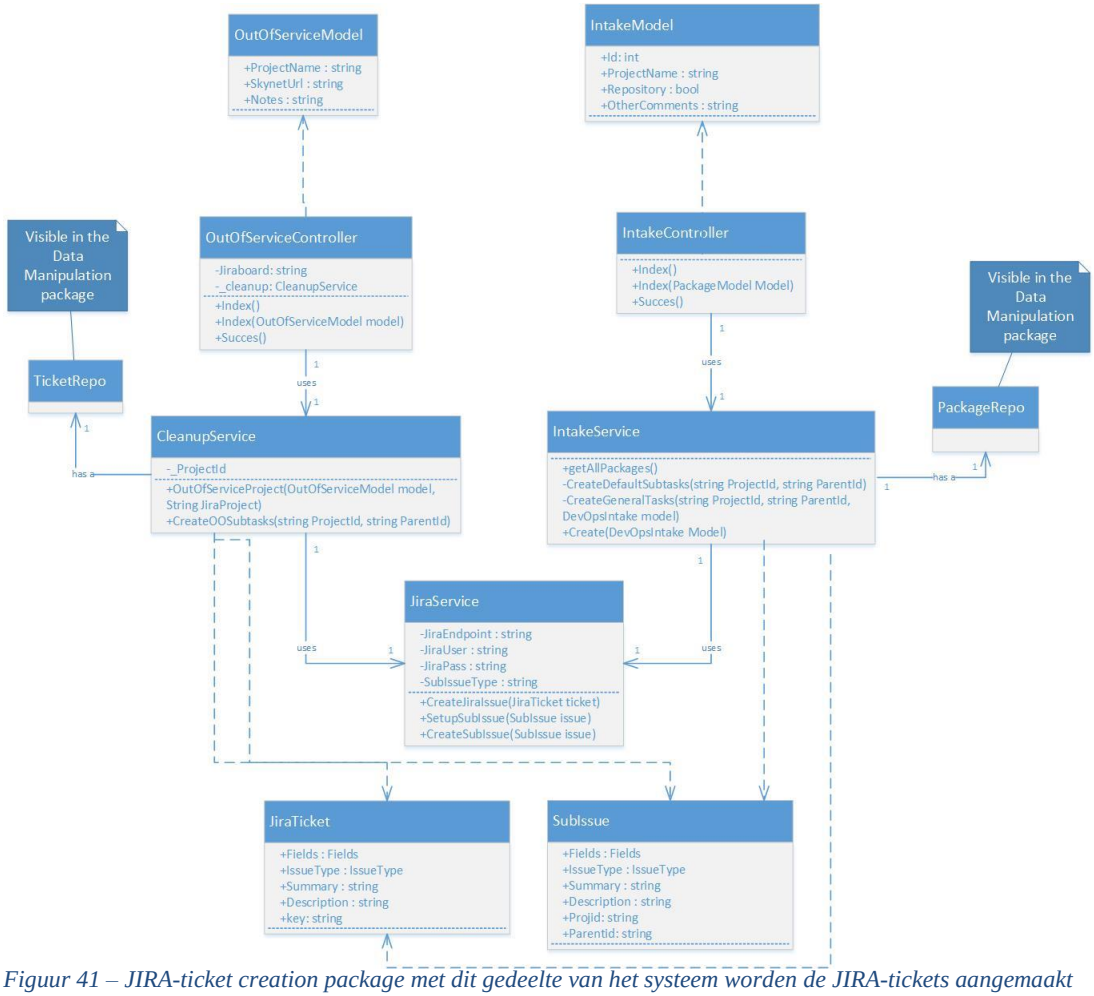
Deze repos implementeren interfaces die ook weer een interface implementeren. Met de interface IRepo dwing ik af dat elke repo een aantal standaardfunctionaliteiten implementeert. Omdat repos soms ook nog unieke functionaliteiten hebben heb ik per repo een eigen interface aangemaakt. Door met interfaces te werken dwing ik af dat de klassen een standaard opbouw hanteren. Dit zorgt ervoor dat wanneer later uitbreidingen in deze code worden gemaakt de structuur van de klasse hetzelfde blijft.

De services geven vervolgens aan de repos een ticket, package of een product mee waar een CRUD-functionaliteit mee moet gebeuren. Deze repo’s spreken vervolgens de TicketContext aan die de DBcontext van het Entity framework implementeert. De TicketContext voert vervolgens de wijzigingen door in de database.



Figuur 40 - Data manipulation package met dit gedeelte van het systeem wordt de data opgehaald, gewijzigd, opgeslagen en verwijderd

Naast de data access laag moest er ook nog een gedeelte van de applicatie ontworpen worden waarmee de tickets in het JIRA-bord aangemaakt konden worden. Hiervoor heb ik gebruik gemaakt van de JIRA-TicketCreator applicatie als voorbeeld, aangezien ik hier al eerder JIRA-tickets mee had aangemaakt. Het ontwerp is zichtbaar in figuur 41.



Figuur 41 – JIRA-ticket creation package met dit gedeelte van het systeem worden de JIRA-tickets aangemaakt op het hosting bord

Er zijn twee verschillende soorten tickets die aangemaakt kunnen worden op het hosting bord. Het out of service ticket en het nieuwe project ticket. Beide ticketsoorten moesten uit een hoofdticket met subtaken bestaan. Hiervoor zijn twee modellen aangemaakt namelijk het JIRA-ticket en het Sub Issue. Ook heb ik een model aangemaakt voor de out of serviceprojecten en voor de project intake.

Twee controllers ontvangen de user input via het OutOfService- en het IntakeModel. Deze services hebben een relatie met een Repo zodat ze de informatie uit de database kunnen ophalen. Deze repos zijn terug te zien in het Data manipulation package. Vervolgens sturen de services de benodigde informatie door naar de JIRA-Service. In deze service wordt de input en bijbehorende data verwerkt met behulp van het JIRA-ticket model en het Sub Issue model. Wanneer dit is gebeurd wordt deze input en data doorgestuurd naar de JIRA-Service. Deze service verstuurd dit met behulp van de RestSharp en Newtonsoft libary naar het JIRA-bord waarin de tickets dan aangemaakt worden.

11.3 Bouwen devops.tamtam.nl

Nadat ik het ontwerp had gemaakt ben ik aan de slag gegaan met het bouwen van de webapp. Deze webapplicatie heb ik gebouwd met het MVC framework. Door de scheiding van lagen in MVC (View, Controller, Model) is het gemakkelijk om te werken aan één aspect van de webapplicatie. Zo kun je bijvoorbeeld aan de controller werken zonder dat het model hierdoor wordt beïnvloed.

Verder heb ik met het Entity framework gewerkt om de database te maken. Met het Entity framework kan je models in je code aanmaken en vervolgens aan de hand van deze models automatisch een database laten genereren. Deze methode heet de *"model first approach"*. De database wordt gegenereerd aan de hand van een context klasse die overerft van DBcontext een klasse uit het Entity framework. In deze context klasse geef je aan welke models in de database aangemaakt moeten worden door middel van een DbSet. Vervolgens maak je connectionstrings aan in de web.config file. Daarna run je in de package manager console de commando's Add-Migration en Update-Database en dan wordt je database gegenereerd.

Het manipuleren van de database wordt gedaan door de DbSet van het model waar je informatie van wil updaten/bewerken/verwijderen/uitlezen aan te spreken.

Tijdens het programmeren liep ik tegen een probleem aan met de TicketContext. Ik maakte deze elke keer aan wanneer ik commando's naar de database wilde sturen. Dit werkte prima totdat ik tijdens het debuggen tegen een vreemde foutmelding aan liep. Deze foutmelding was als volgt *"An entity object cannot be referenced by multiple instances of IEntityChangeTracker"*. Ik ben toen gaan zoeken naar een oplossing voor dit probleem en kwam een stackoverflow artikel tegen (Slauma, 2012). Het antwoord wat in het artikel werd gegeven was vrij uitgebreid. Samengevat was dit als volgt: *"You can fix this by creating a context outside of the service classes and injecting and using it in both services:"*. Ik leidde hieruit af dat er maar één context aangemaakt mocht worden. Het voorstel in het antwoord stelde voor om de context als een parameter mee te geven. Ik besloot echter om het probleem met het Singleton pattern op te lossen.

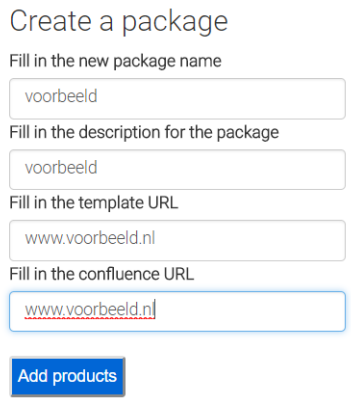
```
26 references | lennart timmers, 9 days ago | 1 author, 3 changes
public class TicketContext:DbContext
{
    private static TicketContext context;
    1 reference | lennart timmers, 27 days ago | 1 author, 1 change
    public TicketContext()
        :base("name=CustomDb") {}


    5 references | lennart timmers, 27 days ago | 1 author, 1 change
    public DbSet<PackageModel> Packages { get; set; }
    7 references | lennart timmers, 27 days ago | 1 author, 1 change
    public DbSet<Ticket> Tickets { get; set; }

    8 references | lennart timmers, 14 days ago | 1 author, 1 change
    public DbSet<ProductModel> Products { get; set; }

    19 references | lennart timmers, 9 days ago | 1 author, 1 change
    public static TicketContext GetContext
    {
        get
        {
            if (context == null)
            {
                context = new TicketContext();
            }
            return context;
        }
    }
}
```

Figuur 42 - Probleem opgelost door implementatie van het singleton patroon.

Het Singleton pattern (figuur 42) zorgt ervoor dat er van een klasse maar één instantie in totaal aangemaakt mag worden. Dit was een effectieve oplossing want de foutmelding kwam daarna niet meer voor. Daarnaast werd de koppeling tussen de klassen ook nog eens lager. Eerst was dit een associatie en in de nieuwe situatie werd dit een dependecy. Door losser te koppelen is de code beter onderhoudbaar omdat klassen minder van elkaar afhankelijk zijn. Ik heb dit geïmplementeerd met behulp van een Microsoft guide (Microsoft, 2017).

Tijdens het aanmaken van packages zijn het aantal tickets en producten variabel. Dit houdt in dat per package het aantal tickets en producten verschilt. Ik heb gezocht of hiervoor een standaard handelswijze is maar kon deze niet vinden. Daarom heb ik hier zelf een oplossing voor bedacht. Daarnaast wou ik ook dat er geen postbacks uitgevoerd worden bij elke stap omdat de site dan trager wordt en dit niet fijn werkt. Ik heb dit opgelost door gebruik te maken van javascript. Hieronder doorloop ik wat ik heb gedaan om dit probleem op te lossen.

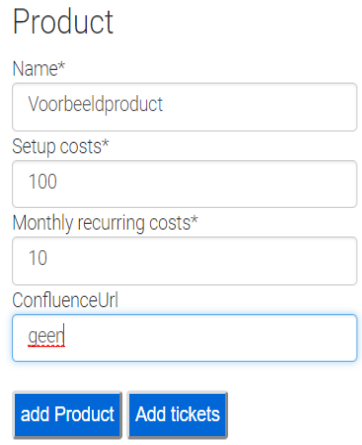
Wanneer een admin een package aanmaakt moet hij/zij eerst een aantal gegevens voor het package zelf invullen (figuur 43). Daarna navigeert de admin door naar het toevoegen van producten scherm.



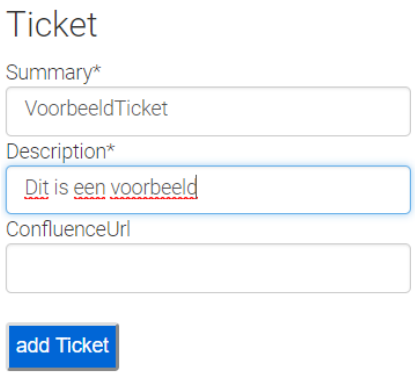
Figuur 43 - Package informatieinvulveld

Wanneer dit gebeurd wordt er javascript aangeroepen die de invulvelden van het package aanmaken verbergt en voor een product maakt deze methode nieuwe velden in de html aan. Vervolgens kan de admin zo vaak als hij/zij wil producten blijven toevoegen (figuur 45). Zolang dit gebeurt worden de productinvulvelden verborgen en weer nieuwe aangemaakt. Is de admin klaar met het toevoegen van producten dan klikt hij/zij door naar de add tickets pagina. (Er kan dus ook gekozen worden om geen producten toe te voegen)

Voor de tickets geld exact hetzelfde als voor de producten (figuur 44). De velden worden aangemaakt en verborgen met behulp van javascript.



Figuur 45 – Productinformatie invulveld



Figuur 44 – Ticketinformatieinvulveld

Wanneer de admin vervolgens op submit klikt wordt het ticket ingestuurd en komt deze in de backend van de code terecht.

```
string hiddenfieldvalue = Request.Form["hidden"];
int value = int.Parse(hiddenfieldvalue);
List<Ticket> ticketList = new List<Ticket>();
for (int i = 0; i < value; i++)
{
    int temp = 1 + i;
    Ticket ticket = new Models.Meta.Ticket();
    try
    {
        ticket.ConfluenceUrl = Request.Form["confluenceUrl" + temp];
    }
    catch
    {
        ticket.ConfluenceUrl = "";
    }

    ticket.Summary = Request.Form["TicketSummary" + temp];
    ticket.Description = Request.Form["TicketDescription" + temp];
    ticket.type = false;
    if(ticket.Summary != "" && ticket.Description != "") {
    ticketList.Add(ticket);
    }
}
model.Tickets = ticketList;
_service.addPackage(model);
return RedirectToAction("index");
```

Figuur 46 - Code die de input voor tickets ophaalt.

In deze code (figuur 46) worden alle aangemaakte ticketvelden uitgelezen en aan een lijst van tickets toegevoegd. Dit gebeurt als volgt.

1. Wanneer de javascript methode een nieuwe reeks velden toevoegt hoogt deze een waarde in een hiddenfield op. Dit getal haal ik vervolgens op door Request.Form te gebruiken. Dit geeft je de waarde van een html attribuut.
2. In de loop vul ik vervolgens de waardes van het ticket ook weer met behulp van Request.Form. Deze velden hebben allemaal dezelfde naam maar, onderscheiden zich met de waarde die steeds wordt opgehoogd. De temp waarde wordt gebruikt om alle velden op te halen.
3. Het type van het ticket geeft aan of het een out of service ticket of een normaal ticket is. Een package ticket hoort nooit bij een out of service ticket vandaar dat hier altijd false wordt meegegeven.
4. Als de Summary en de Description niet leeg is wordt het ticket vervolgens aan de lijst toegevoegd
5. Wanneer de for loop afgelopen is worden de tickets toegevoegd aan het packagemodel en wordt dit aan de service aangeboden.

Ditzelfde gebeurt ook voor producten. Door op deze manier de tickets uit te lezen kan ik zonder postbacks een variabel aantal tickets en producten toevoegen.

11.4 Testen

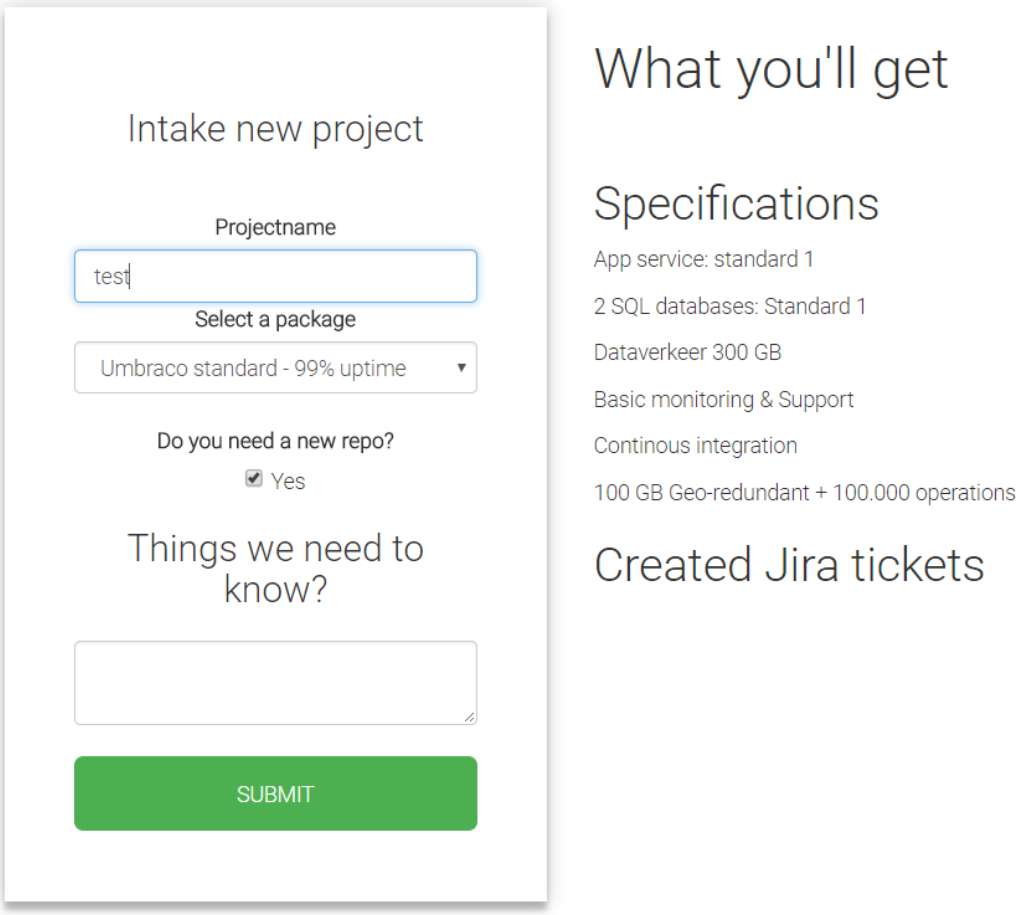
Ik had helaas niet genoeg tijd meer om de applicatie uitgebreid te testen. De functionaliteiten zijn echter wel getest tijdens het schrijven van de code. Dit is onder andere gebeurd door gebruik te maken van de debugger functie in Visual studio.

Ook was er voor mij een speciale test omgeving devops.test.tamtam.nl opgezet. Deze omgeving maakte gebruik van een testdatabase en een test JIRA-bord. Wanneer ik wijzigingen had gemaakt in de code en deze naar de live versie pushte dan kwamen deze veranderingen ook in devops.test.tamtam.nl te staan.

Toen de applicatie af was heb ik deze ook nog laten goedkeuren door de stakeholders. Zij hebben op de testomgeving door alle aanwezige functionaliteiten geklikt en gekeken of dit naar behoren functioneerde. Toen zij tevreden waren hebben zij mij goedkeuring gegeven en vervolgens is de website beschikbaar gemaakt op devops.tamtam.nl.

11.5 Eindproduct devops.tamtam.nl

Het eindresultaat is een werkende website waar mensen nieuwe projecten kunnen aanvragen en out of serviceprojecten kunnen insturen. Dit kunnen gebruikers doen door naar devops.tamtam.nl/intake te gaan of naar devops.tamtam.nl/outofservice. Hieronder volgen een aantal screenshots met uitleg om duidelijk te maken hoe het eindresultaat eruitziet.



Figuur 47 - Intakeformulier voor een nieuw project beschikbaar op devops.tamtam.nl/intake

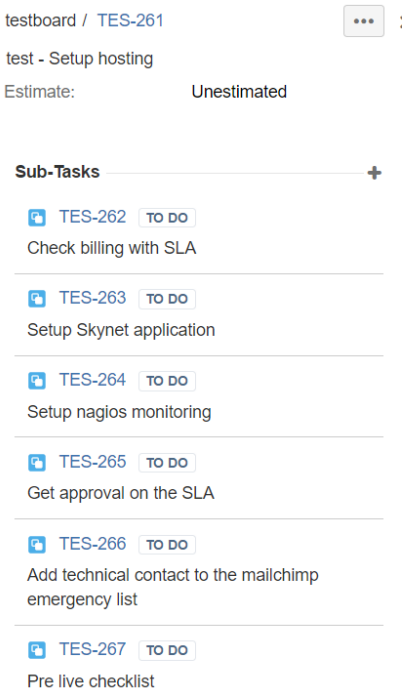
In het formulier in figuur 47 is te zien hoe een gebruiker nieuwe projecten kan aanvragen. De gebruiker geeft de naam van het nieuwe project aan, kiest een package uit de dropdownlist en geeft aan of er een nieuwe repository voor aangemaakt moet worden. Ook is er de optie om extra opmerkingen bij te voegen. Deze komen dan in het hoofdticket te staan.

Rechts in het scherm is te zien welke producten en tickets in het gekozen package aanwezig zijn. In dit package zitten toevallig geen tickets maar wel een aantal producten. De standaard tickets die altijd worden aangemaakt worden niet weergeven in deze lijst.

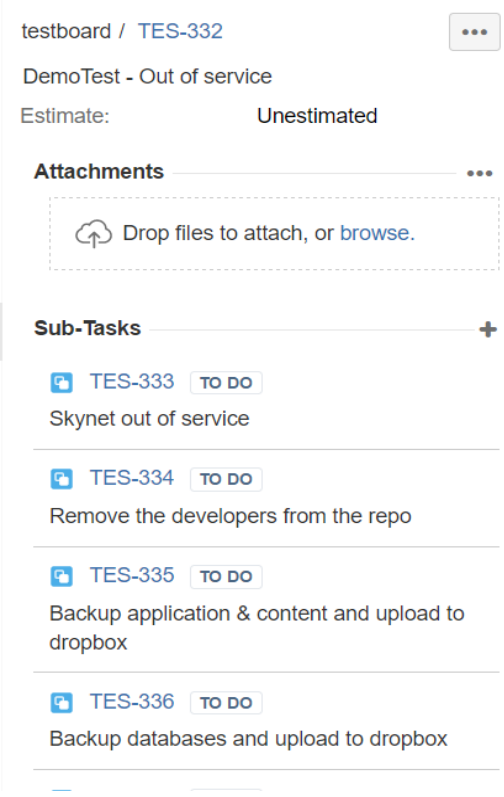
Wanneer de gebruiker op submit klikt krijgt hij/zij een confirmatie scherm te zien en wordt het ticket in het hosting bord aangemaakt. Dit ticket ziet er in het bord uit zoals in figuur 48.

Zoals te zien is in het hoofdticket (linksboven) de naam verschenen die de gebruiker in figuur 47 heeft aangegeven. Ook is te zien dat onder het Sub-Tasks kopje de standaard taken zijn aangemaakt. Het ticket komt in de backlog van het bord te staan. Vanaf hier kan het aan een sprint toegevoegd worden en kan een team gelijk aan de slag gaan met alle uit te voeren taken.

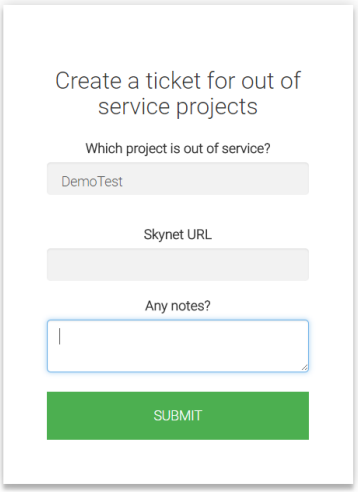
Het aanmaken van een out of serviceproject gebeurd ook met behulp van een formulier. In figuur 50 is dit formulier te zien. Ook wanneer dit formulier wordt ingestuurd worden taken aangemaakt op het JIRA-bord. Dit resultaat is te zien in figuur 49.



Figuur 48 - Eindresultaat in JIRA na het verzenden van het intakeformulier



Figuur 49 - Eindresultaat in JIRA na het verzenden van het out of service formulier



Figuur 50 - Formulier voor out of serviceprojecten beschikbaar op devops.tamtam.nl/outofservice

11.6 Reflectie sprint 5

Ik ben tevreden over het opgeleverde product in deze sprint. Ondanks dat ik weinig tijd beschikbaar had heb ik een werkend product opgeleverd. Ik ben alleen niet toegekomen aan het uitgebreid testen van deze applicatie. Ik had dit misschien wel kunnen doen als ik minder functionaliteiten in de website had ingebouwd maar, ik heb hier geen spijt van. Als ik namelijk een aantal functionaliteiten niet had geïmplementeerd dan kon de webapplicatie niet in gebruik worden genomen wat nu wel is gebeurd.

Het verzamelen van de requirements is vlot verlopen omdat de stakeholders een duidelijk beeld hadden van de op te leveren applicatie. Door gebruik te maken van MoSCoW was ook duidelijk welke requirements het belangrijkste waren.

Tijdens het bouwen van de applicatie heb ik het ontwerp als basis gebruikt. Ik denk dat dit een slimme aanpak was aangezien je tijdens het bouwen altijd nog wel tegen andere problemen aanloopt. Er zijn geen opmerkelijke verschillen tussen het ontwerp en de code. Er zijn af en toe alleen extra parameters aan een methodes meegegeven. Het aantal klassen en de associaties/implementaties/dependencies zijn allemaal hetzelfde gebleven.

11.7 Sprint 5 in het kort

Op te leveren	Opgeleverd?
Requirementsdocument	Ja, opgesteld aan de hand van drie gesprekken twee met Leon en Niek en een met Niek.
Ontwerpen devops.tamtam.nl	Ja, ontwerp opgeleverd en dit gebruikt als basis tijdens het bouwen
Bouwen devops. tamtam.nl	Ja, Gebouwd met het Entity en het MVC framework. Ontwerp gebruikt als basis. Kleine wijzigingen in methodeparameters
Testen devops.tamtam.nl	Deels, wel getest of de functionaliteiten werken, maar niet uitgebreid getest met een testdocument. Wel goedkeuring ontvangen van de stakeholders.

12. Reflectie

In dit hoofdstuk reflecteer ik op afstudeerperiode bij TamTam. Ik geef aan wat ik heb geleerd gedurende deze periode en wat ik van mijn eigen handelen vind. Ik bespreek dit voor het proces en de producten. Ook bespreek ik hoe ik aan de beroepstaken heb voldaan.

12.1 Proces

Over het algemeen ben ik tevreden met de wijze waarop ik het proces heb uitgevoerd. Scrum was een goede keuze omdat tijdens het project de requirements nog zijn veranderd. Ik heb echter niet helemaal volgens het traditionele Scrum proces gewerkt. Dit vereist een standaardlengte van sprints. Dit is niet gebeurd omdat ik sprints van drie weken heb gehanteerd. Hierdoor kwam ik niet precies uit met de weken die ik beschikbaar had voor de opdracht.

Ik heb dit opgelost door een opstartsprint van een week te houden en een sprint van vier weken. De volgende keer kan ik beter eerst kijken naar het aantal weken wat ik beschikbaar heb voor een opdracht en vervolgens een keuze maken voor de sprintduratie. Dit zodat ik een vaste doorlooptijd heb voor elke sprint.

Verder zijn de sprint reviews goed verlopen. In deze reviews besprak ik met Leon en Niek de opgeleverde producten. Aan de hand van deze besprekingen ontstonden soms ook nog nieuwe backlog items. Alleen als deze binnen de opdracht vielen en haalbaar waren werden deze toegevoegd aan de product backlog.

Vervolgens werd aan het eind van de sprint gekeken welke items van de backlog in de nieuwe sprint meegenomen gingen worden en werd de nieuwe sprint gestart. Ook werd het retrospective uitgevoerd. In dit retrospective besprak ik met mijn begeleider wat er beter kon en wat goed was gegaan. Ik nam ook zelf aan het einde van een sprint altijd even de tijd om mijn werk te evalueren zoals te zien in de reflectie kopjes bij elke sprint.

Ik heb tijdens mijn afstudeerperiode geleerd om op een flexibelere manier om te gaan met Scrum. Deze wijze van werken geeft je meer vrijheid aangezien je geen story points toewijst per taak. Hierdoor kan je zelf bepalen in welke volgorde je te werk gaat met de sprint backlog. Persoonlijk vond ik dit een fijne manier van werken.

12.2 Producten

Tijdens het afstuderen heb ik twee complete producten opgeleverd. Het eerste product is de nieuwe monitoringsituatie binnen TamTam. Het tweede product is de interne website devops.tamtam.nl. Beide opgeleverde producten zijn inmiddels door TamTam in actief gebruik genomen en bevallen goed.

Zelf ben ik tevreden over deze producten. De stakeholders waren blij met de eindproducten die ik aan hen heb opgeleverd. Daarnaast zijn de opgeleverde producten ook duidelijk gedocumenteerd op de Confluence van TamTam.

12.2.1 Nieuwe monitoringsituatie

Ik ben erg tevreden over de nieuw opgeleverde monitoringsituatie. In de doelstelling werd aangegeven dat TamTam proactief wilde gaan handelen, omdat in de oude situatie klanten af en toe aangaven dat een website onbereikbaar was zonder dat TamTam dit wist. Daarnaast moest er een vaste wijze van monitoren zijn en moest er duidelijk worden wie waarvoor verantwoordelijk is.

Sinds de door mij ontworpen situatie live is gegaan is TamTam altijd als eerste op de hoogte zodra er een website onbereikbaar is. Daarnaast wordt er nu ook één standaard van monitoring gehanteerd met een on-call schema. Hierdoor is ook duidelijk wie waarvoor verantwoordelijk is. De doelstelling van de opdracht is dus succesvol behaald.

- **Intake huidige situatie**
  Ik vind dat ik dit op de juiste manier heb aangepakt. Door te praten met mensen die hier kennis van hebben kon ik mij een duidelijk beeld vormen van deze situatie en dit vervolgens uitwerken in een document.
- **Requirementsdocument**
  Ik had aan de hand van interviews een requirementsdocument opgesteld. Dit requirementsdocument heb ik na feedback van mijn examinatoren nog wel enigszins aangepast zodat deze goed gedefinieerd waren. Ik had deze requirements iets te “losjes” opgesteld waardoor deze verkeerd geïnterpreteerd konden worden. De volgende keer moet ik de requirements meteen duidelijk opstellen zodat deze niet verkeerd begrepen kunnen worden.
- **Onderzoek nieuwe situatie**
  Over dit onderzoek ben ik erg tevreden. Het onderzoek kijkt niet alleen naar de oude en nieuwe tools, maar ook naar wat tooling bij grote IT-bedrijven. Hierdoor was de selectie van tooling erg uitgebreid maar naar mijn idee ook erg nauwkeurig.
- **Architectuur & presentatie nieuwe situatie**
  Over deze architectuur ben ik ook tevreden. Hierin wordt per tool aangegeven wat deze gaat doen in de nieuwe situatie en wie waarvoor verantwoordelijk is. Hierdoor is het duidelijk hoe de nieuwe situatie zou gaan functioneren. Ik heb deze architectuur dan ook gebruikt tijdens de presentatie van de nieuwe situatie waar ik meteen goedkeuring voor ontving.
- **Klassendiagram TicketCreator**
  Het klassendiagram voor de TicketCreator had ik duidelijker kunnen maken. Ik had namelijk in de klassen waarin ik data ging ontvangen nog geen variabelen toegevoegd, omdat ik niet wist welke waardes ik terug zou krijgen van de API’s. Achteraf gezien had ik dit tijdens het ontwerpen gewoon kunnen opzoeken in de bijbehorende documentatie van de API’s. Volgende keer moet ik hier dus netter mee omgaan zodat het ontwerp duidelijker is. Het ontwerp is verder wel gebruikt voor het bouwen van de TicketCreator
- **Implementeren nieuwe situatie**
  Door eerst een eigen Nagios server op te zetten en hierin te oefenen met het aanmaken van testen kon ik ervaring opdoen zonder de actieve Nagios server van TamTam te laten

crashen. Ook begreep ik hierdoor de classificatie die Nagios hanteert. Ik vind dat ik hier een verstandige aanpak heb gehanteerd met de minste risico's.

Het overzetten van testen naar de Nagios server vanuit Pingdom en UptimeRobot is dankzij het oefenen met een test Nagios server soepel verlopen. De Nagios server is niet down gegaan tijdens het overzetten van deze testen waardoor de monitoring niet is uitgevallen.

Het aanmaken van alerts in AWS en Azure is ook goed uitgevoerd. Dit is niet ingewikkeld omdat je alleen maar een grenswaarde en een email adres hoeft aan te geven. Ik heb hier wel duidelijke afspraken voor gemaakt met de stakeholders zodat zij niet onnodig wakker gebeld worden.

Vervolgens heb ik een nieuwe OpsGenie omgeving ingericht zodat de nieuwe situatie geen oude monitoringmeldingen met zich meebracht. Er zijn wel nog wat wijzigingen uitgevoerd om de situatie te optimaliseren. Er is later een prioritering toegevoegd zodat alerts alleen s'nachts binnenkwamen wanneer deze het prioriteitsniveau TT1 had. Ik had hiervan tevoren rekening mee kunnen houden zodat de stakeholders niet wakker gebeld waren voor minder belangrijke alerts.

- **Bouwen en testen TicketCreator**
  Ik vind dat ik dit goed heb aangepakt. De code wordt duidelijk uitgelegd in de comments en is niet overbodig complex. Door gebruik te maken van twee libaries hoefde ik ook zelf minder code te schrijven. Verder heb ik het ontwerp gebruikt als basis voor het bouwen van deze applicatie.

  Ook het testen hiervan is goed verlopen. Ik heb dit gedaan door een testdocument op te stellen en deze testen vervolgens uit te voeren. Hierdoor werd ook nog een fout ontdekt in de permissies van JIRA.

  Ik heb tijdens het maken van de nieuwe monitoringsituatie heel veel nieuwe kennis opgedaan. Ik heb geleerd wat monitoring is, hoe je kan monitoren, wat voor tools hiervoor op de markt aanwezig zijn een hoe je deze tools met elkaar kan laten integreren. Ook heb ik geleerd hoe je met API's kan werken. Ik had dit nog niet eerder gedaan.

12.2.2 Devops.tamtam.nl

Devops.tamtam.nl wordt ook actief gebruikt door TamTam. Ondanks dat ik maar drie weken beschikbaar had voor deze opdracht vind ik dat ik een redelijk compleet product heb opgeleverd. Door tijdsgebrek is er alleen niet uitgebreid getest. Ook van deze opdracht is de doelstelling bereikt. Het aanmaken van nieuwe projecten en out of serviceprojecten gebeurt nu volledig automatisch in JIRA.

- **Verzamelen requirements**
  Ik heb dit gedaan door middel van gesprekken met Leon en Niek. De requirements die ik uit de gesprekken heb achterhaald heb ik vervolgens door hen laten controleren en goedkeuren. Ook voor dit requirementsdocument geld dat ik de requirements achteraf nog duidelijker heb gedefinieerd.
- **Ontwerpen devops.tamtam.nl**
  Ik heb dit ontwerp met de UML-modeleertaal gemaakt. Door een onderscheid te maken in twee packages heb ik het ontwerp overzichtelijker kunnen houden waardoor dit makkelijker te begrijpen is. Tijdens het bouwen heb ik het ontwerp constant gebruikt. Af en toe heb ik in methodes andere parameters meegegeven dan in het ontwerp, maar verder heb ik geen andere veranderingen gemaakt. Alle klassen en associaties zijn dus op de ontworpen wijze in de code terug te vinden.
- **Bouwen devops.tamtam.nl**
  Zoals hierboven vermeld is het ontwerp gebruikt om de code te schrijven. Tijdens het programmeren zijn er af en toe kleine wijzigingen gemaakt, maar deze zijn niet noemenswaardig. Verder functioneert de code naar behoren en maakt deze de juiste tickets aan in het JIRA-bord.

Tijdens het bouwen van dit product heb ik leren werken met het Entity framework. Ik had hier nog nooit eerder mee gewerkt en dit was dus een nieuwe ervaring. Daarnaast heb ik ook voor het eerst het Singleton pattern geïmplementeerd in code. Ik heb dus vooral veel nieuwe kennis opgedaan op het gebied van coderen tijdens het bouwen van devops.tamtam.nl.

12.3 Beroepstaken

1.1 Selecteren methoden, technieken en tools

Ik heb dit gedaan door een onderzoek te doen naar de aanwezige tools binnen TamTam, op de markt en bij grote IT-bedrijven. Vervolgens heb ik een aantal tools geselecteerd aan de hand van het voldoen aan de eisen. Met Nagios heb ik leren werken voordat ik aan de slag ging met dit implementeren in de nieuwe situatie. Dit zodat in de huidige omgeving van Nagios geen fouten zou maken door mijn onervarenheid in de tool.

Ook heb ik een ontwikkelmethodiek geselecteerd uit drie verschillende opties. Deze opties bespreek ik en ik geef aan waarom ik uiteindelijk voor Scrum kies.

Bijbehorende hoofdstukken voor deze beroepstaak: 4, 6.2, 7, 9.1

1.4 Uitvoeren analyse door definitie van requirements

Ik heb requirements opgesteld door interviews te houden met de vijf verschillende stakeholders. Deze interviews heb ik opgenomen met een voice recorder en vervolgens uitgewerkt. Uit deze uitwerkingen heb ik daarna een requirmentsdocument opgesteld met als prioritering methode MoSCoW.

Voor de tweede opdracht heb ik gesprekken gehouden en de requirements uit deze gesprekken direct opgeschreven. Ook hier heb ik als prioritering methode MoSCoW gebruikt.

Bijbehorende hoofdstukken voor deze beroepstaak: 6.1, 11.1

3.2 Ontwerpen systeemdeel

Ik heb gedurende mijn afstudeerperiode meerdere ontwerpen gemaakt. Dit zijn naast ontwerpen voor systeemdelen ook de architectuur van de geselecteerde tools. Ik heb voor het ontwerpen van de systeemdelen gebruik gemaakt van de UML-modeleertechniek. Voor de architectuur heb ik een simpeler plaatje getekend waarmee duidelijk wordt gemaakt wie verantwoordelijk is voor welke componenten en hoe deze met elkaar praten.

Bijbehorende hoofdstukken voor deze beroepstaak: 8.1, 8.3, 11.2

3.3 Bouwen applicatie

Ik heb twee applicaties gebouwd tijdens mijn afstuderen. De eerste applicatie is nog niet in gebruik genomen, maar is wel volledig afgebouwd namelijk de TicketCreator. Dit is een kleine applicatie die met een timer elk uur een methode uitvoert. Dit is een console applicatie geschreven in C#.

De andere applicatie is devops.tamtam.nl. Deze applicatie is inmiddels in actief gebruik door TamTam. Deze applicatie is geschreven in C# met behulp van het Entity en het MVC framework.

Voor beide applicaties is er gebruik gemaakt van een gemaakt ontwerp als basis.

Ook heb ik de geselecteerde tools met elkaar laten integreren. In Nagios heb ik de configuratie zo aangepast dat alle testen de juiste prioriteit hebben en ik heb de testen van Pingdom en UptimeRobot overgeplaatst.

Bijbehorende hoofdstukken voor deze beroepstaak: 9.4, 9.5, 11.3

3.5 Uitvoeren van en rapporteren over het testproces

Ik heb om de TicketCreator applicatie te testen een testrapport opgesteld. In dit rapport bepaal ik de equivalentieklassen en aan de hand hiervan de logische en fysieke testgevallen. Deze testgevallen test ik vervolgens met unit-testen. Daarnaast heb ik ook nog een aantal unit-testen geschreven voor een vergelijking die in de code wordt gemaakt.

Ik had geen tijd meer om devops.tamtam.nl uitgebreid te testen. De functionaliteiten heb ik uiteindelijk getest door tijdens het programmeren te kijken of alles goed werkte. Ook heb ik bij het opleveren de stakeholders door de applicatie laten klikken en daarna hun goedkeuring ontvangen.

Bijbehorende hoofdstukken voor deze beroepstaak: 9.6, 11.4

13. Bijlages

A Afstudeerplan

B Plan van aanpak

C Uitgewerkte interviews

D Requirementsdocument

E Toolselectie monitoring

F Statuscake proof of concept

G Architectuurdocument

H Testdocument

I Requirementsdocument devops.tamtam.nl

A Afstudeerplan

Afstudeerplan

Informatie afstudeerder en gastbedrijf (*structuur niet wijzigen*)

Afstudeerblok: 2017-1.1 (start uiterlijk 6 februari 2017)
Startdatum uitvoering afstudeeropdracht: 6 februari
Inleverdatum afstudeerdossier volgens jaarrooster: 2 juni 2017

Studentnummer: 13093193
Achternaam: dhr Timmers
Voorletters: L.H.
Roepnaam: Lennart

Opleiding: Informatica
Locatie: Zoetermeer
Variant: voltijd

Naam studieloopbaanbegeleider: Nanny Jacobs
Naam begeleidend examinator: Nanny Jacobs
Naam tweede examinator: Hani Al-Ers

Naam bedrijf: TamTam
Afdeling bedrijf: DevOps
Bezoekadres bedrijf: Delftseplein 27N
Postcode bezoekadres: 3013AA
Postbusnummer:
Postcode postbusnummer:
Plaats: Rotterdam
Telefoon bedrijf (algemeen):
Telefax bedrijf: +31 (0)88-0400811
Internetsite bedrijf: www.tamtam.nl

Achternaam opdrachtgever: dhr van Rees
Voorletters opdrachtgever: L.
Titulatuur opdrachtgever:
Functie opdrachtgever: Teamlead OS & DevOps
Doorkiesnummer opdrachtgever:
Email opdrachtgever: Leon@tamtam.nl

Achternaam bedrijfsmentor: van Rees
Voorletters bedrijfsmentor: L
Titulatuur bedrijfsmentor:

Functie bedrijfsmentor: Teamlead OS
Doorkiesnummer bedrijfsmentor:
Email bedrijfsmentor: leon@tamtam.nl

NB: bedrijfsmentor mag dezelfde zijn als de opdrachtgever

Doorkiesnummer afstudeerder:
Functie afstudeerder (deeltijd/duaal):

Titel afstudeeropdracht:

Servers self healend maken en wanneer de self healing stap niet lukt automatisch een melding naar de eindverantwoordelijke sturen

Opdrachtomschrijving

1. Bedrijf

Het Bedrijf TamTam is een van de top 3 digital agencies in Nederland, het levert diensten omtrent online aanwezigheid van bedrijven. Hieronder vallen onder andere het bouwen van websites en mobiele apps en het bepalen van online marketingstrategieën.

Het bedrijf bestaat uit diverse projectteams en hiernaast ondersteunende afdelingen zoals Finance, Operational services en DevOps. De probleembeschrijving is voornamelijk geconcentreerd op het gebied van DevOps en Development. De afdeling waar ik ga werken tijdens mijn stageperiode is DevOps.

2. Probleemstelling

Binnen het bedrijf TamTam lopen er veel verschillende projecten met verschillende omgevingen. Voor ieder nieuw project wordt er automatisch een of meerdere serveromgevingen ingeregeld en hier een CI/CD tool aan gekoppeld. Dit kan zijn; Octopus, Bamboo of TamTam Skynet.

Deze omgevingen kunnen door fouten met de server of de applicatie plat komen te liggen. Wanneer dit gebeurt krijgt Sentia, hosting partner van TamTam een melding dat er wat aan de hand is met de server of applicatie. Dit bedrijf gaat dan eerst een standaardprotocol doorlopen om het probleem op te lossen. Wanneer dit niet lukt wordt er contact opgenomen met een contactpersoon (Leon van Rees) bij TamTam zodat zij dit probleem kunnen oplossen.

Door de afhankelijkheid van een derde partij blijkt het lastig om de juiste mensen te bereiken in geval van nood. Doordat Leon de bottleneck is moet hij eerst uitzoeken wie verantwoordelijk is voor het project en vervolgens met deze persoon contact opnemen wat tot gevolg heeft dat de downtime op de applicatie hoger wordt en de klanttevredenheid daalt.

In sommige gevallen wordt zelfs door de klant aangegeven dat de website down is zonder dat dit bij TamTam bekend is. Ook dit maakt geen goede indruk op de klant waardoor de klanttevredenheid daalt.

3. Doelstelling van de afstudeeropdracht

TamTam maakt een transitie door van fysieke eigen servers op het Sentia evo-platform naar cloudoplossingen zoals; Amazon Web Services, Microsoft Azure en Sentia cloud. Daarbij is het mogelijk om de cloud-infrastructuur proactief te monitoren en self healing te maken. Hierdoor wordt dure support op de infrastructuur praktisch overbodig en kunnen we een betere “service experience” leveren aan de klant.

Het doel is dus om de supportdiensten van Sentia overbodig te maken door de nieuwe servers in de cloud self healend te maken. Dit houdt in dat de server zelf een standaardprotocol doorloopt als de server een foutmelding geeft. Wanneer de server zichzelf niet weet te herstellen of wanneer de fout zich in de applicatie zelf voordoet, moet er automatisch een melding worden verstuurd. Deze melding gaat naar de eindverantwoordelijke van het project zodat deze hier direct mee aan de slag kan.

4. Resultaat

Wanneer deze opdracht succesvol wordt afgesloten zal TamTam veel kosten kunnen besparen op het gebied van tijd, denk hier aan de tussenpersoon bij TamTam die nu nog de juiste persoon op de hoogte moet brengen van dit probleem. Ook op het gebied van geld zullen de kosten die nu gemaakt worden door het monitoringbedrijf bespaard worden.

Daarnaast zal er veel sneller geschakeld kunnen worden waardoor de klanten veel beter geholpen
worden. Deze proactiviteit moet leiden tot een betere klanttevredenheid en een hogere uptime van de applicaties.

5. Uit te voeren werkzaamheden, inclusief een globale fasering, mijlpalen en bijbehorende activiteiten

Bij TamTam wordt de ontwikkelmethode SCRUM gebruikt. Deze methode zal ik dus ook gaan gebruiken gedurende mijn stage. In SCRUM wordt er met sprints gewerkt. Aan het eind van de sprint wordt er een functionerend product opgeleverd.

Tijdens mijn periode bij TamTam zal ik meerdere fases doorlopen. De volgende fases zal ik doorlopen.

- Research en informatieverzameling
- Ontwerpen
- Implementatie
- Testen
- Documenteren

6. Op te leveren (tussen)producten

Ik zal de volgende producten opleveren:

- Een plan van aanpak
- Een requirements document
- Een onderzoek naar bestaande oplossingen op gebied van self healing servers en monitoring wat aantoont of en hoe dit toepasbaar is bij TamTam

- Een self healend serversysteem dat meldingen stuurt naar de eindverantwoordelijke wanneer self healing mislukt
- Een overdracht dossier voor de dev-ops afdeling

7. Te demonstreren competenties en wijze waarop

De volgende competenties zal ik gedurende mijn afstuderen aantonen.

1.1 Selecteren methoden, technieken en tools
Ik zal moeten bepalen welke programmeertaal en welke eventueel welke tools er gebruikt gaan worden bij dit project

1.4 Uitvoeren analyse door definitie van requirements
Het binnen de organisatie verzamelen van requirements en deze op een traceerbare manier documenteren.

3.2 Ontwerpen systeemdeel
Aan de hand van de requirements het systeem ontwerpen wat hierna gebouwd kan worden. Dit zal in de vorm van een architectuur plaatje gedaan worden.

3.3 Bouwen applicatie
Het bouwen van de eerder ontworpen applicatie.

3.5 Uitvoeren van en rapporteren over het testproces
Het testen van de gebouwde applicatie om te garanderen dat deze goed functioneert.

B Plan van aanpak

Plan van aanpak.

Server Monitoring

1 juni 2017

Inhoudsopgave.

Informatie begeleiders en student 86
Student
.....8
6
Begeleiders 86

Over TamTam 87

1. Introductie opdracht 88
1.1 Huidige situatie 88
1.2 Doelstelling 88
1.3 Beoogd resultaat 88
1.4 Projectorganisatie 89
1.5 Randvoorwaarden opdracht 89
1.6 Afbakening opdracht 90
1.7 Risico's opdracht 90
1.8 Bewaking van de voortgang 90

2. Mijlpaalproducten 91

3. Planning 92

4. Aanpak 93

Informatie begeleiders en student

Student

Naam: Lennart Timmers
Studentnummer: 13093193
Opleiding: Informatica
Email: 13093193@student.hhs.nl l.h.timmers@hotmail.com Lennart.timmers@tamtam.nl
Telefoon: 0681080325

Begeleiders

TamTam

Naam: Leon van Rees
Rol: Bedrijfsmentor
Contactgegevens: Leon@tamtam.nl

De Haagse Hogeschool

Naam: Nanny Jacobs
Rol: Begeleidend examinator
Contactgegevens: A.A.A.M.Jacobs@hhs.nl

Naam: Hani Al-Ers
Rol: Expert examinator
Contactgegevens: H.Al-Ers@hhs.nl

Over TamTam

TamTam is opgericht door Bart en Paul Manuel in 1996, voordat het woord “punt-com” was uitgevonden. Het doel was om een online catalogus te maken voor de technische groothandel. Dit is er echter nooit van gekomen en TamTam is nu één van de grootste digital agencies.

De naam TamTam staat voor communicatie. TamTam onderscheid zichzelf door goed te luisteren naar klanten, de juiste vragen te stellen, echte oplossingen te bedenken en deze ook goed te implementeren. En daar is TamTam heel trots op. We luisteren echter niet alleen naar onze klanten maar ook naar elkaar. Hier wordt niet gelet op rangorde, de beste ideëen zijn ontstaan dankzij input van jongere collegas.

TamTam gelooft in het geven van vrijheid. Ze gaan ervan uit dat wanneer je gezamenlijk besluiten neemt over doelstellingen, dat mensen hier enthousiast verantwoording voor nemen. Dit wordt binnen TamTam dan ook van haar werknemers verwacht.

TamTam is niet echt fan van procedures en regels. De aanpak die TamTam prefereert is het doen van nodige dingen. Niet het doen van dingen die moeten. Dit leidt af en toe wel tot verwarring in bepaalde processen, maar uiteindelijk loopt alles goed af en zelfs nog beter dan wanneer een “correcte procedure” gevolgd zou zijn.

1. Introductie opdracht

Gedurende mijn stage zal ik mij bezig houden met het verbeteren van de monitoring van de servers. Dit gebeurt nu door verschillende oplossingen die draaien bij TamTam. Enkele voorbeelden hiervan zijn OpsGenie, Uptimerobot en Pingdom. Dit zijn tools die allemaal een deel van de servers monitoren op dit moment. Het is de bedoeling om te kijken welke tools er nu al aanwezig zijn bij TamTam en of deze ingezet kunnen worden voor onder andere het self-healend maken en monitoren van servers.

1.1 Huidige situatie

De servers draaien op verschillende platformen namelijk, Amazon webservices, Microsoft Azure, Sentia Evo platform en Sentia cloud. Van deze servers wordt de monitoring nu deels gedaan door Sentia namelijk de Sentia Cloud servers en een deel van de azure servers. De Amazon servers worden op dit moment door DevOps gemonitord. Het doel is om uiteindelijke de servers van het Sentia Evo platform te verplaatsen naar een cloudoplossing.

1.2 Doelstelling

Tijdens de transitie van fysieke servers naar cloudoplossingen is het mogelijk om servers proactief te monitoren en self healend te maken. Hierdoor wordt dure support op de infrastructuur praktisch overbodig en kan er een betere “Service experience” geleverd worden aan de klant.

Het doel is om de supportdiensten van Sentia overbodig te maken door de nieuwe cloudoplossingen self-healend te maken. Dit houdt in dat de server zelf een standaardprotocol doorloopt wanneer er een fout op de server ontstaat. Wanneer dit de fout niet oplost moet er automatisch een melding naar de projectowner van dat project gestuurd worden. Door de tussenpersoon te elimineren in het monitoring process zal dit uiteindelijk een betere klanttevredenheid moet resulteren.

1.3 Beoogd resultaat

Wanneer deze opdracht succesvol wordt afgesloten zal TamTam veel kosten kunnen besparen op het gebied van tijd, denk hier aan de tussenpersoon bij TamTam die nu nog de juiste persoon op de hoogte moet brengen van dit probleem. Ook op het gebied van geld zullen de kosten die nu gemaakt worden door het monitoringbedrijf bespaard worden. Daarnaast zal er veel sneller geschakeld kunnen worden waardoor de klanten veel beter geholpen worden. Deze proactiviteit moet leiden tot een betere klanttevredenheid en een hogere uptime van de applicaties.

1.4 Projectorganisatie

Onder dit kopje staan de gemaakte afspraken en verantwoordelijkheden tussen de opdrachtgever en de opdrachtnemer.

De opdrachtnemer is verantwoordelijk voor:

- Maken van de documentatie
- Realiseren functionaliteiten
- Testen functionaliteiten
- Implementeren functionaliteiten
- Overdragen kennis
- Afgesproken aantal uren voldoen
- Aanleveren laptop

De rol van opdrachtnemer zal vervuld worden door de afstudeerder Lennart Timmers.

De opdrachtgever is verantwoordelijk voor:

- Aanleveren materiaal
- Aanleveren werkplek
- Verantwoordelijk voor beslissingen over het product
- Onderhoud, bugfixes en aanpassingen na afloop van de afstudeer periode
- Begeleiding van de student

De rol van opdrachtgever zal vervuld worden door Leon van Rees teamlead DevOps en OS.

1.5 Randvoorwaarden opdracht

Er zijn een aantal randvoorwaarden om de afstudeeropdracht succesvol af te ronden, namelijk:

- Stakeholders moeten beschikbaar zijn gedurende de opdracht. Zonder stakeholders kunnen er geen stappen ondernomen worden om de opdracht te starten en te verwerken.
- De afstudeerder moet toegang krijgen tot de benodigde systemen. Dit zijn systemen zoals BitBucket, JIRA en andere systemen die nodig zijn om de opdracht te voltooien.
- De bedrijfsmentor moet beschikbaar zijn voor feedback en vragen.

Als er niet aan deze voorwaarden voldaan kan worden zullen er nieuwe afspraken moeten worden gemaakt. Mocht dit gebeuren dan zal dit vermeld worden in het afstudeerverslag.

1.6 Afbakening opdracht

Tijdens dit project zullen alle producten uit hoofdstuk 2 opgeleverd worden.

De onderdelen die buiten de scope van de opdracht vallen zijn als volgt:

- Het in kaart brengen van de eindverantwoordelijken van elk project binnen TamTam.
- Het actief monitoren van de servers en applicaties bij TamTam.

1.7 Risico's opdracht

Tijdens dit project zijn er enkele risico's. De risico's zijn als volgt:

- **Gebrek aan Kennis**
  De afstudeerder heeft nog totaal geen kennis op het gebied van monitoring en self healing. Dit zal tot gevolg hebben dat er vertraging ontstaat bij het werken aan nieuwe onderwerpen.
- **Wijzigende requirements**
  Nadat de Researchfase voorbij is kunnen de requirements door omstandigheden wijzigen. Wanneer dit gebeurt is er een kans dat de oplossing (deels) niet meer voldoet aan de requirements

1.8 Bewaking van de voortgang

Om te verzekeren dat de opdrachtnemer de afspraken die zijn gemaakt nakomt zijn er wekelijkse reflectie momenten ingepland. Tijdens deze momenten wordt er met de afstudeerder gereflecteerd op wat er die week is gedaan en of dit voldoende was. Zo nee dan worden er verbeterpunten bedacht zodat deze problemen zich niet nogmaals voordoen.

2. Mijlpaalproducten

De volgende mijlpaalproducten zullen gedurende de stageperiode gemaakt worden door de student.

Requirements document
Dit document heeft als doel de eisen van de opdracht duidelijk vast te stellen. Om dit document op te stellen zullen er enkele interviews afgenomen worden met TamTammers die nuttige input voor dit document kunnen verzorgen. Dit interview zal in de eerste periode van de stage worden gehouden omdat er anders geen progressie gemaakt kan worden op andere gebieden.

Onderzoek bestaande oplossingen op gebied van self healing en monitoring
Dit onderzoek zal kijken naar de mogelijkheden op het gebied van self healing en monitoring binnen TamTam met bestaande tools. Er zal onder andere gekeken worden welke tools nu al binnen TamTam gebruikt worden en of deze aan de eisen voldoen van het requirements document. Zo ja dan wordt er gekeken hoe deze tool optimaal in gebruik kan worden genomen, zo nee dan wordt er gekeken naar andere tools die dit wel kunnen.

Een self healend server systeem met notificaties
Dit kan de implementatie van een bestaande tool zijn of eventueel het bouwen van een eigen tool.

Een overdracht dossier voor de DevOps afdeling
Dit dossier zal alle benodigde informatie bevatten over de uiteindelijke monitoring oplossing die is gekozen. Met dit dossier zal elke DevOpser van TamTam de benodigde kennis hebben om de oplossing te gebruiken.

Al deze producten zullen wanneer ze klaar zijn goedgekeurd moeten worden. Voordat de producten goedgekeurd zijn door de opdrachtgever kan er niet verder gewerkt worden.

3. Planning

Voor dit project heb ik in eerste instantie een aantal fases gemaakt. Het is namelijk nog niet duidelijk hoe de planning er exact uit zal zien omdat er gewerkt gaat worden met scrum.

- **Research en informatieverzameling**
  In deze fase zal het requirementsdocument opgesteld worden. Dit zal gebeuren door interviews te houden. Ook zal er een inventarisatie van de huidige situatie gemaakt worden en zal er research worden gedaan naar andere tools die aanwezig zijn op de markt.
- **Ontwerpen**
  In deze fase zal een architectuur gemaakt worden voor hoe de nieuwe situatie eruit moet gaan zien.
- **Implementatie**
  In deze fase zal de architectuur verwerkelijkt gaan worden.
- **Testen**
  In deze fase zal de opgeleverde oplossing getest worden.
- **Documenteren**
  In deze fase zal de documentatie benodigd voor het onderhouden van de nieuwe oplossing opgeleverd worden.

Er zal gewerkt worden met de ontwikkelmethode scrum. Een methode waarmee je wekelijks een planning maakt genaamd een sprint.

De sprints die ik ga uitvoeren zullen over het algemeen twee weken duren. Als er echter aanleiding is om een sprint korter of langer te maken dan kan dit in overleg gebeuren. In totaal ben ik 17 weken aan het afstuderen bij TamTam. Ervan uitgaande dat elke sprint twee weken duurt zal ik in totaal acht sprints uitvoeren van twee weken en één sprint van een week.

Om een inzicht te geven in de globale planning is hieronder een tabel met het aantal sprints per fase aangemaakt.

Fase	**Datum**	**Sprints**
Research	7 februari tot 28 februari	1.5 (3 weken)
Ontwerpen	28 februari tot 7 maart	1 (2 weken)
Implementatie & testen	7 maart tot 30 mei	5.5 (11 weken)
Documenteren	30 mei tot 2 juni	0.5 (1 week)

4. Aanpak

Tijdens het afstuderen zal er gebruik gemaakt worden van scrum. Aangezien scrum sprints altijd worden ingepland vlak voordat deze begint is er nog geen concrete planning aanwezig. Wel is al duidelijk welke producten er opgeleverd gaan worden. In dit hoofdstuk staat beschreven hoe de op te leveren producten tot stand zullen komen.

Het requirementsdocument
Voordat ik kan beginnen met het maken van een requirementsdocument zullen er eerst interviews gehouden moeten worden. Ik zal allereerst een document opstellen voor de interviews. Hierin zal staan wie er geinterviewd gaan worden en welke vragen aan hen gesteld gaan worden. Ook zal er een introductie van de opdracht in naar voren komen zodat de interviewees een idee hebben wat ik precies ga doen. Aan de hand van dit document zal ik vervolgens de geselecteerde personen gaan interviewen. Ik zal deze gesprekken opnemen en later uitwerken in een apart document. Vervolgens zal ik deze interviews netjes samenvatten in het document waarin de vragen waren opgesteld. Aan de hand van de volledig uitgewerkte interviews zal ik daarna requirements gaan constateren. Wanneer ik denk alle requirements achterhaald te hebben zal ik dit voorleggen aan mijn begeleider en de feedback die ik hierop ontvang dan verwerken.

Onderzoek tools op gebied van self healing en monitoring
In dit onderzoek zullen meerdere documenten uitgewerkt gaan worden. Als eerste zal ik de huidige situatie in kaart gaan brengen. In dit document zal ik alle tools die op dat moment bij TamTam gebruikt worden verzamelen. Over deze tools zal ik dan uitleg geven waar ze op dat moment voor gebruikt worden, wat de tools allemaal kunnen en hoe duur deze tools zijn. Daarna zal ik een document opstellen met (waarschijnlijk vergelijkbare) tools die functionaliteiten aanbieden waar TamTam naar op zoek is (herleid uit de requirements). Van deze tools zal aangegeven worden welke functionaliteiten ze hebben en hoe duur ze zijn. Vervolgens zal ik een document opstellen waarin ik de eisen waar de tool aan moet voldoen vergelijk met de uitgekozen tools. In dit document zal ik een conclusie trekken welke tool (of tools) het beste bij TamTams wensen aansluiten. Ook zal dit onderzoek gebruikt worden om de kennis van de afstudeerder over het onderwerp te verbreden.

Een self-healend monitoring systeem met alerting functie
In deze periode zal ik aan de slag gaan met de geselecteerde tools. Ik zal eerst een architectuur maken van de gewenste situatie. Aan de hand van deze architectuur zal ik vervolgens een werkende oplossing maken. Ik zal een tool of meerdere tools moeten inrichten zodat de monitoring bij TamTam beter gaat verlopen. Ook moeten de servers met behulp van self healing eerst proberen zichzelf te herstellen. Lukt dit niet dan zal er door de tool contact opgenomen worden met de eindverantwoordelijke van het project. Tijdens het ontwikkelen hiervan zullen er tests uitgevoerd worden om te kijken of alles werkt. Deze test zullen opgesteld worden in een document en vervolgens uitgevoerd worden door de afstudeerder. De resultaten zullen vervolgens opgenomen worden in het document. Uiteindelijk zal er een werkende monitoringsoplossing opgeleverd worden die TamTam in gebruik kan nemen voor hun servers.

Een overdracht dossier voor de DevOps afdeling

Dit dossier zal waarschijnlijk op de TamTam Confluence geschreven worden. Dit is een plaats waar informatie binnen de organisatie wordt gedeeld. In dit dossier zal informatie staan over de gekozen tool(s) en hoe hiermee gewerkt moet worden.

C Uitgewerkte interviews

Uitgewerkte interviews.

Monitoring

1 juni 2017

Inhoudsopgave.

1. Interview Marlies 97

2. Interview Niek 99

3. Interview Arthur 100

4. Interview Leon 102

5. Interview Sandor 106

1. Interview Marlies

Wat voor rol heb je binnen TamTam?
Ik ben Teamleader Operational Services dit houdt in dat ik de schakel ben tussen de klantentickets en de ticketverwerkers.

Wat is je mening over de huidige monitoring?
Ik kom hier eigenlijk alleen mee in aanraking wanneer er een server down is. Ik weet natuurlijk wel dat we weggaan bij Sentia omdat we naar de Cloud willen en dat hier ook al nieuwe projecten naartoe gezet worden. Het is dan dus ook wel spannend hoe de monitoring hiervoor behandeld wordt. Wie gaat dit s 'nachts bijhouden en wat doen we als er iets kapot gaat? Dat zijn vraagstukken die ik wel belangrijk vind want ik bel zelf natuurlijk liever de klant op het moment dat er iets kapot is. Op dit moment zitten wij wel echt in een fase dat wij door de klant worden gebeld en dat komt niet heel professioneel over. Ik zou dus graag concrete afspraken zien over wat er van elkaar verwacht wordt. Dit ook zodat je de klant correcte feedback kan geven wanneer zijn probleem is opgelost.

Vind je dan dat de servers te vaak down zijn?
Nou dat valt best mee vind ik maar, soms wordt het structureel bij bepaalde klanten. Sommige klanten geven we bijvoorbeeld uptime garantie en daar komen we dan meestal wel uit. Op het moment dat er echter te veel downtime is dan zit je onder de performance en dan krijg je een onderzoek. Iets meer pro activiteit op dit gebied zou dan wel fijn zijn. Of de servers echter te veel down zijn nee dat vind ik niet. Het zijn over het algemeen dan wel dezelfde klanten maar daar speelt dan meestal ook wat.

Waar moet het nieuwe systeem volgens jou aan voldoen?
Een systeem moet helpen, het moet je leven makkelijker maken. Stel dat er bijvoorbeeld een klant bij het aws down gaat. Ik wil dan dat de informatie in Skynet ook matched met de informatie die Amazon aangeeft. Dan kan ik namelijk gelijk zien welke klanten er bij Amazon zitten en om welke klanten het bij dat probleem gaat. Dat zijn voor ons valide dingen ik weet nu namelijk niet welke klant ik moet informeren wanneer er iets mis gaat. Vooral tijdens de overstap naar de Cloud is juist nu het moment om dit professioneel aan te pakken. Dat het duidelijk is welke klant welke dienst afneemt en dat deze informatie ook matched met de informatie in Skynet.

Wanneer TamTam zelf gaat monitoren hoe zit dan voor jou de ideale situatie eruit?
Daar heeft Leon vrij specifieke ideeën over waar ik het wel mee eens ben. Er zijn een paar vaste teams binnen TamTam, gaat er bijvoorbeeld met Oxxio of de Hypotheker iets mis dan kan ik daaraan gaan sleutelen maar dat heeft totaal geen zin want wij weten van die applicatie gewoon te weinig af. Dan moet het dus gelijk duidelijk zijn wie daarvoor gebeld moet worden. Dan moeten er ook afspraken komen dat mensen af en toe in de avond of in het weekend gebeld zullen worden. En dan moet er ook rekening gehouden worden met het prioriteit niveau van de website. Wanneer er bijvoorbeeld een informatieve website midden in de nacht een hikje heeft hoeft hier natuurlijk niet voor gebeld te worden. Maar er moeten dus wel duidelijke afspraken gemaakt worden hoe er gehandeld wordt wanneer een website down gaat. En ook de administratie van wie waarvoor verantwoordelijk is, is hier van belang. TamTam groeit snel waardoor er veel nieuwe gezichten bij komen en uiteindelijk verlies je het overzicht.

Wie denk je dat hier een rol in moet gaan spelen? En wie dan in de zin van een specifiek persoon of een team.

In ieder geval geen specifiek persoon. Leon of ik kunnen onze schouders hier wel onder zetten maar, het zijn uiteindelijk de developers die dit als normaal moeten gaan accepteren. Dat zal nog wel even hoofdpijn opleveren maar ik denk wel dat daar een leerschool in gaat zitten want dan zorgen de programmeurs er tenminste voor dat hun werk op orde is voordat ze naar huis gaan. Nu wordt er best wel makkelijk gedacht dat het toch wel wordt afgevangen maar, wanneer het spanningsveld verplaatst zal dit waarschijnlijk wel veranderen.

2. Interview Niek

Wat voor rol heb je binnen TamTam?
Ik ben DevOps engineer.

Wat is je mening over de huidige monitoring?
Er zijn op dit moment te veel verschillende tools. Ook oudere tools die aan persoonlijke accounts zijn gekoppeld en dus niet meer goed worden bekeken. Het is vervelend dat Sentia Nagios beheert want die machine werkt niet helemaal goed en het is heel erg onoverzichtelijk er staan nu 9000 regels aan configuratie in één bestand waardoor dit bijhouden bijna niet te doen is. We worden geacht van half 8 tot half 6 ofzo dit zelf te monitoren maar dit gebeurd eigenlijk nooit. Wanneer er nu een site down gaat gaan er pas na 3 kwartier pagers piepen bij Sentia en dat duurt dus te lang.

Heb je al een voorkeur voor een tool?
Een nieuwe Nagios server, Pingdom en dit dan netjes integreren met een tool als OpsGenie. Of iets dergelijks.

Waar moet het nieuwe systeem volgens jou aan voldoen?
Het nieuwe systeem moet makkelijk in omgang zijn. Het liefst zo makkelijk dat iedereen het kan gebruiken. Ook zou het fijn zijn als we het kunnen automatiseren. De overzichten die het systeem aanlevert moeten helder zijn. Je moet de uptime garantie uit kunnen lezen. Ook mogelijkheden om custom checks, dus niet alleen maar pingen, aan te maken wil ik graag hebben. Servers op infrastructuur niveau en op functioneel niveau monitoren. Op dit moment doen wij nu eigenlijk alleen maar functionele monitoring en Sentia monitort nu de infrastructuur. Maar het doel is wel om zelf meer de infrastructuur te kunnen monitoren. Dit zodat we eerder problemen kunnen detecteren en kunnen kijken of we de servers zelf de problemen kunnen laten oplossen met een script.

Als TamTam zelf gaat monitoren hoe ziet de ideale situatie er dan voor jou uit?
Als eerste zou ik graag willen dat servers eerst zelf kijken of ze het probleem kunnen oplossen. Stel er is een issue met de website dan start de server eerst de applicatie opnieuw op en kijkt die of dit geholpen heeft. Zo nee dan wordt er een reset van IIS uitgevoerd. Helpt dat niet dan geef je de server een reboot (als het een dedicated server is). En daarna kijk je via een call schema wie ga ik hiervoor bellen. En dan stuur je die persoon de informatie “Hé dit is er aan de hand, ik zie deze errormelding terugkomen, deze acties zijn al doorlopen, los dit probleem op”.

Wie denk je dat in het monitoren een rol moet gaan spelen?
Ik denk dat projectteams het zelf moeten gaan inrichten. De teams weten ook beter wat er gemonitord moet worden. DevOps kan wel pagen maar daar houdt het ook mee op. Lead developers gaan hier een rol in spelen, DevOps zal een rol vervullen, Leon zal hierbij betrokken zijn en de servicedesk voor het contact naar de klanten. En dan komen er misschien ook nog een paar escalatiepunten bij zoals Joep en Bart.

3. Interview Arthur

Wat voor rol heb je binnen TamTam?
Ik ben architect

Wat is je mening over de huidige monitoring?
Ik heb nog meegemaakt dat het helemaal niet gebeurde dus wat dat betreft vind ik het een stuk beter. Ik denk alleen dat de rol van wie de monitoring moet regelen nu niet goed duidelijk is. Als dit niet gebeurt dan schiet je niets met dit project op. Het minimale wat eigenlijk moet gebeuren is dat je www.klantnaam.nl ergens hebt geregistreerd, met daarachter een e-mail of een sms naar een bepaald nummer. Ik geloof niet dat je met automatisering ermee komt, omdat ik niet het gevoel heb dat je eromheen komt om het handmatig te doen. De simpelheid en het ordenen daarvan moeten echter wel on point zijn. In mijn ogen zijn er eigenlijk drie niveaus van monitoren.

De meest simpele is, is de site up of down? Daar moet één tool voor komen die dit kan. En ik hoor Pingdom en Nagios maar volgens mij doen deze tools hetzelfde. De reden waarom Nagios een rotzooitje is en soms niet wordt opgepakt om monitoring in te richten is omdat Nagios complexer is. En wanneer iemand er dan iets simpels mee wil doen lukt dit niet omdat het te complex is. Bij Pingdom bijvoorbeeld voeg je een URL toe en hij staat in het systeem. Bij Nagios moet je veel meer informatie verschaffen. Dat is tenminste wat ik van afstand heb begrepen.

De tweede vorm van monitoring is werken bepaalde functionaliteiten wel of niet. Dit is wat Nagios dus voor een deel doet met een post. Je kan het zien als het unit testen van URL's.

De derde vorm van monitoring zijn de achterliggende systemen waar onze systemen gebruik van maken om uiteindelijke een website te laten zien. Bijvoorbeeld bij de Alliantie die hebben een hele grote infra backend waar ons systeem tegen aan praat. Ons systeem is weer de basis voor de website. De website kan functionaliteiten missen doordat er een bepaalde database bij de Alliantie uit ligt.

In die zin zijn er dus drie vormen van monitoring. Eigenlijk moet je deze drie vormen gedekt hebben om goed te kunnen monitoren. Dit zodat als ergens een signaal afgaat er ook sneller naartoe gewerkt kan worden naar één probleem. Een site die down is kan verschillende dingen betekenen. Is de site echter down en Azure is down ja dan weet je waar het probleem zit. En Azure of AWS is dan dus ook weer een derde vorm die ook gemonitord zal moeten worden.

Heb je al een voorkeur voor een tool?
Nee niet echt. Wat ik wel belangrijk vind is dat de tool simpel in gebruik is. Als het kan wil ik het liefst dat Marlies van OS het kan inrichten
Waar moet het nieuwe systeem volgens jou aan voldoen?
Mag zelf nooit down gaan (grappend), nee maar simplicity. Het moet niet te veel kunnen. En het liefste heb ik een Cloud service waar we geld aan betalen in plaats van een Nagios server die zelf gehost wordt. Je ziet wat er nu met Nagios gebeurt dat komt doordat wij het moeten onderhouden en niemand deze taak krijgt toegewezen. Neem je een service in de Cloud dan heb je hier geen last van.

Als TamTam zelf gaat monitoren hoe ziet de ideale situatie er dan voor jou uit?
Op het moment dat de nieuwe situatie live is. Dan moeten er bepaalde tickets in JIRA staan die jouw verplichten om bepaalde acties te ondernemen. Denk hier dan bijvoorbeeld aan tickets zoals “URL in Pingdom zetten”, “Bepalen welke calls in Nagios kunnen” en “Die calls in Nagios zetten”. En totdat je dat hebt mag je niet live.

Als het dan eenmaal live staat wil ik graag dat er een duidelijke chain of command is. Dus eerst wordt die ge-sms’t en vervolgens deze.

Wie denk je dat hier een rol in moet gaan spelen?
DevOps en de projectteams. De projectteams zijn ervoor verantwoordelijk dat er een omgeving ingericht is en DevOps is verantwoordelijk voor het juist acteren als er een probleem is.
En OS hebben die hier nog wat mee te maken?
In de zin van het verzorgen van klantcontact wel maar, het oplossen dat ligt niet bij hen.
Wat mij wel ook heel belangrijk lijkt is dat alle drie de kanalen weten dat de andere op de hoogte zijn gebracht en dat zij ook allemaal tegelijk op de hoogte zijn gebracht. Dit zodat zij zelf hun eigen acties kunnen verrichten en dat ze weten van elkaar dat ze dat bericht hebben gekregen.
Heb jij nog eisen waar per se aan voldaan moet worden omdat het project anders in jouw ogen mislukt is?
Wat het project wel moet doen is bedrijf kritisch zijn. Dat jij weet of een site up of down is, is nuttig maar, als een bepaalde salesflow niet werkt dan kan de site gewoon up zijn zonder dat je merkt dat er iets mis is. Bijvoorbeeld als de functie ticket kopen niet werkt bij Walibi. Eigenlijk is dit dan nog de vierde vorm. Bij het functioneel testen moet je kijken of er dagelijks een test scriptje af kan gaan. En of dit meerdere keren per dag moet dat verschilt per service. Dit leidt wel tot het probleem dat een klant testdata moet afvangen. In het geval van Walibi moeten ze registreren dat er een ticket is verkocht, maar eigenlijk is er geen ticket verkocht. En deze systemen zijn hier niet op gemaakt. Ik zeg niet dat dit binnen de scope van TamTam moet zijn maar, benoem wel dat daar een oplossing voor is.

4. Interview Leon

Wat voor rol heb je binnen TamTam?
Ik ben lead OS en DevOps. Zal ik daar iets over vertellen i.v.m. je verslag?
Misschien is dat wel handig ja.
Ik ben hier nu 2 jaar en ben toen van een eigen bedrijfje hier binnengekomen als programma manager. Als programma manager was ik met name bezig met klanten. Ik werkte dan voor klanten zoals De Haagse Hogeschool, het Florensis team de lang lopende projecten niet de losse dingetjes. En toen merkten we dat er in de organisatie behoefte was aan iemand die het DevOps en OS team intern naar de volgende stappen ging brengen. Deze rol werd echter door niemand opgepakt en omdat ik al ervaring had met het bouwen van een eigen bedrijf ben ik op die manier in de functie gevallen. Toen ik in deze functie begon was Martijn hoofd DevOps en de verwachting was dat ik met hem zou optrekken, omdat ik hier nog niet veel kennis van had. Op dat moment zei Martijn echter zijn baan op waardoor ik op een afdeling belande waar degene met de meeste ervaring weg ging. Ook Dennis ging weg en inmiddels hebben we ook afscheid genomen van Kevin (oud DevOps). We zijn eigenlijk dus opnieuw begonnen met bouwen van DevOps. Het lastige daaraan is dat ik eigenlijk geen technische kennis heb en alles dus benader vanaf de proces kant. Dat is dus eigenlijk mijn rol hier, Service, DevOps en Quality insurance.
En Friso hoe lang werkt hij hier dan nu?
Friso is eigenlijk de enige engineer die nog bij DevOps werkt en voor mijn tijd hier ook al werkte. Ik ben ook heel blij met hem en later ook Niek. Het lastige hieraan was dat je aan de ene kant een nieuw team wil inwerken, maar je hebt ook ambities als afdeling en je wil deze ook waarmaken. Bijvoorbeeld het werken met Amazon en daar moet je dan wel de juiste mensen voor weten te vinden.
Dus DevOps had ook al een rol voordat dit gebeurde?
Jazeker, maar het veranderd. Onze visie nu heeft te maken met de 65 developers die we hier in dienst hebben. Dat zijn eigenlijk onze klanten. Zij moeten zo goed mogelijk gefaciliteerd worden als het gaat om hosting, deploying, kortom ondersteuning bij technische vraagstukken en dat is echt ons doel.
Wat is je mening over de huidige monitoring?
Dat het niet goed in elkaar zit. Dat is ook de reden waarom ik jou heb aangenomen om dit op te lossen. En dan wil je zeker ook weten waarom ik dat vind?
Ja.
Het belangrijkste is dat we op dit moment met Sentia samenwerken dat is ook logisch.
Ja want Sentia was een oudere samenwerkingspartner van jullie toch?
Wij hebben jaren geleden geïnvesteerd in een eigen stukje hardware bij TamTam Evo. Sentia is degene die daar altijd de inrichting voor had gedaan. Ook het beheer stuk ligt bij Sentia. Daar zit onder andere ook een stukje monitoring in. Dat is allemaal vrij duidelijk. Nu gaan wij echter de shift naar Cloud maken waarop Sentia zegt dat is leuk maar wij gaan niet zomaar sites van jullie monitoren die jullie gebouwd hebben maar ergens op een andere omgeving neerzetten. Want als iets fout gaat hebben zij geen flauw idee hoe dat in elkaar zit. Dus vandaar dat er nu een gat begint te ontstaan. Dat is de ene kant van het verhaal dus hoe accuraat is de monitoring en wat doen we met de nieuwe smaken die we in gaan zetten? De andere kant van het verhaal is dat de sites die wel door Sentia gemonitord worden en waar vervolgens een actie uit komt dat er twee mogelijke aanleidingen zijn. Er zit een fout in de infra of er zit een fout in de applicatie. Verreweg de meeste gevallen ligt het aan de infra kant en dan kan Sentia een aantal stappen doorlopen om te kijken of ze het weer up en running

krijgen. Ligt het in de applicatie dan is het de vraag “en nu”? En we zijn nog zoekende hoe we hiermee moeten handelen. Op dit moment ga ik van persoon naar persoon. Kortom het is niet duidelijk wie waarvoor verantwoordelijk is.

Ja Joey heeft hiervoor een lijst vanuit Skynet gegenereerd maar hij gaf al aan dat die informatie outdated was.

Ja dat kan ik alleen maar beamen. Dit is misschien ook wel interessant om te weten. Joey is op dit moment eigenlijk bezig met zoeken naar een CMDB, dus een systeem waarin je gegevens kunt onderbrengen, wat hier een goede oplossing voor moet zijn en misschien komt hier wel gewoon als antwoord Skynet uit. Maar dan zal deze informatie wel up to date moeten worden gebracht. Dus dat is aan de data kant dat moet zuiver zijn. Daar moet in ieder geval actie op worden ondernomen. En dan is de vraag van oké stel dat er iets down is. Dan gaat er vanuit Sentia naar TamTam en vanuit TamTam naar een developer die route. En die straat moet helpen om die route duidelijk te zien. Stel dat ik nu word gebeld en Oxxio ligt plat dan kan ik naar dat team lopen, maar dan is het hopen dat daar iemand zit.

Ja precies, en wanneer het midden in de nacht is dan moet het eigenlijk doorgeschakeld worden naar het Oxxio team.

Ja maar die gaan daar niet hun bed voor uitkomen want die hebben geen afspraken richting de klant. En dat is eigenlijk nog een beetje een vraagstuk waar we mee zitten. We bieden namelijk geen 24/7 support aan op de applicatie wel op de infrastructuur dat wordt door Sentia gewaarborgd. Dus in het slechtste geval gaat er iets down om één uur en dan ligt iedereen nog in bed. Dan wordt het om 8 uur voor het eerst gelezen door iemand en dat zou ik dan bijvoorbeeld kunnen zijn. Ik neem dan contact op met het Oxxio team eer ik dan iemand te pakken heb is het 9 uur. Dan gaat iemand het een keertje oplossen en dat duur ook nog even en tegen de tijd dat het opgelost is heeft die site er 12 uur uitgelegen. Dat is natuurlijk wel het doomscenario en dat is eigenlijk veel te lang, maar het is wel conform de afspraken die we met de klant hebben. Op de applicatiesupport hebben wij tot 6 uur s ’avonds. Er is dus een risico voor zo’n klant dat als er een fout zit in de applicatie dat de site een halve dag down is. En wanneer het in een weekend is kan dit nog veel langer zijn, want we bieden alleen tijdens kantooruren support. En de vraag is of we dat willen veranderen. We kijken nu of we dept breed iets kunnen organiseren omdat we dan veel meer schaal hebben. We zitten immers in Duitsland maar ook bijvoorbeeld in Boston en dan heb je verschillende tijdzones die misschien elkaar kunnen coveren. Het lastige hiervan is dat je te maken hebt met verschillende partijen die verschillende cmsen gebruiken en op verschillende manieren zijn opgebouwd. Je kan dus ook niet 1 persoon in Boston verantwoordelijk maken voor alles wat in Nederland wordt gemaakt. Dat is dus ook nog een lastige maar we gaan dat dus ook niet in jouw onderzoek beantwoorden.

Heb je al een voorkeur voor een tool?

Ja ik heb wel een voorkeur. We moeten er zo min mogelijk werk aan hebben. Wanneer wij als DevOps voor postbode gaan spelen dan kunnen we eigenlijk nooit op vakantie of ziek zijn. We hebben immers maar een klein team zitten. We kunnen dus moeilijk verantwoordelijk zijn voor alle projecten binnen TamTam. Tot nu toe zijn we dit wel altijd geweest en dat komt doordat Martijn die hiervoor bij DevOps werkte de afgelopen 10 jaar dat “gewoon gedaan heeft”. Het allerlekkerste zou zijn dat een externe partij voor de monitoring verantwoordelijk is en wanneer zij er echt niet uitkomen dat zij contact met ons opnemen. Dan krijg je echter weer twee vraagstukken. De externe partij (in dit geval Sentia) hoe benaderen zij die mensen? En staan zij daarvoor open? De tweede vraag is de

applicaties die we nu in de Cloud stoppen die niet bij Sentia staan. Hoe gaan die uiteindelijk gemonitord worden?

Waar moet het nieuwe systeem volgens jou aan voldoen?

Het moet nou ja sowieso zuiver zijn. De juiste persoon moet gecontacteerd worden. Het moet makkelijk op te schalen zijn, een duidelijk overzicht van wat er op dit moment speelt, makkelijk kunnen uitzetten wanneer je een server voor onderhoud zelf down brengt dus makkelijk muten, het moet een aantal mensen kunnen bereiken. En het liefste ook dat je voor sommige applicaties specifieke mensen kan pagen.

Oké laten we dan even zeggen dat er een server down gaat bij Azure bijvoorbeeld de server van Powned. Azure stuurt dan een melding naar OpsGenie en die moet dit dan kunnen herkennen. En dan kan je inderdaad een specifiek persoon aan de hand van het subject contacteer die en die zeggen.

Ja top dat moet het eigenlijk wel een beetje kunnen. En ja ik wil niet te veel vanuit OpsGenie denken maar, wat ik er wel fijn aan vind is dat ik berichtjes krijg en dat ik ook nagebeld wordt wanneer ik deze niet lees. Dus actief pushen is wel fijn. Ook wanneer een telefoon gemute staat. Mijn telefoon staat zo ingesteld dat als ik meerdere keren word gebeld dat het door de mute functie heen gaat. Dat soort functie zijn erg fijn.

Als TamTam zelf gaat monitoren hoe ziet de ideale situatie er dan voor jou uit?

Voor mij is de constatering dat wij als klein DevOps groepje niet de hele dag moeten gaan monitoren gedurende de kantooruren. Het kan helpen maar het moet ook los kunnen functioneren. Het moet niet uitmaken waar je zit er moet nog steeds gemonitord kunnen worden. En eigenlijk zie ik DevOps nu als een postbode omdat we vaak niets kunnen doen aan de problemen. Met cloudoplossingen moeten we uiteindelijk ervoor zorgen dat ze niet down gaan door middel van self healing. En op het moment dat er dan iets aan de hand is zal het voor 90% aan de applicatie liggen. Dus het liefst dat dan direct het lijntje met de lead developer wordt gelegd. Daar is Joey nu ook mee bezig om die lijntjes duidelijker te maken, maar er zit nog een stukje van hoe we die afspraken gaan maken en hoe we diegene gaan contacteren en of ze dag en nacht beschikbaar zijn. En nogmaals we hebben daar geen verplichting toe. Vanuit TamTam willen we wel graag mensen helpen dus met onze best effort gaan we het heus wel proberen op te lossen. Wanneer Powned plat ligt gaan we echt niet wachten tot maandag om dat op te lossen. Dus hoe zie ik dit het liefste ingericht? Alles wat we extern kunnen beleggen dit extern beleggen en alles wat dat niet kan dat dat zo min mogelijk acties vereist. Dat stukje self healing bijvoorbeeld. En op het moment dat het echt fout gaat dat de juiste personen gelijk gecontacteerd worden. Wat er ook nog in ontbreekt, ik weet niet of je dat al eerder hebt gehoord, is dat wij niet heel goed zijn in het terugkoppelen naar klanten toe. We moeten klanten laten weten wat voor problemen we hadden en hoe we het hebben opgelost. Dat willen klanten graag weten.

Wie denk je dat hier een rol in moet gaan spelen?

Degene die hier het best op ingericht zijn is de supportdesk. Maar ook Marlies haar kennis houdt snel op. Dan moet eigenlijk de Lead developer snel aangesproken worden. Je hebt twee dingen het proces de communicatie en de inhoud en dat zijn de developers. En hoe meer extra schakels je hiertussen kan weghalen hoe beter.

Dus aan de ene kant moet OS er veel mee te maken krijgen omdat zij de klant gerust moeten stellen en aan de andere kant moet de lead architect ook weten dat zijn project down is.

Ja precies.

Heb jij nog eisen waar per se aan voldaan moet worden omdat het project anders in jouw ogen mislukt is?
Er moet gewoon een duidelijk plan zijn omdat we zoveel technieken door elkaar gebruiken. We hebben custom projecten we hebben .net cmsen we hebben PHP cmsen. En voor mij is het belangrijk dat wanneer er een nieuw project wordt gehost dat het gelijk duidelijk is hoe je dit gaat monitoren. Dus bijvoorbeeld dat de lead developer in the loop zit. Kortom het landschap goed in kaart hebben.

De tweede voorwaarde is dat we de juiste tooling beschikbaar hebben en daar ook harde keuzes in maken op het gebied van kosten en features dat dat goed met elkaar matched. Een antwoord zou echter ook kunnen zijn dat je dit niet zelf wilt monitoren omdat het dan te complex wordt. Dat hebben we zo aangetoond en laten we zorgen dat er 1 partij is die de monitoring oppakt. Ik wil dat de juiste keuzes gemaakt kunnen gaan worden. En dan heb ik dan nog het beter voorkomen dan genezen stukje en dat is de self healing waar we het over hadden. Dat zouden we kunnen starten op het punt waar Niek gestopt is met de testomgeving en daar dan tastbare output uit produceren. Wat doen we bijvoorbeeld met een testomgeving? Deze worden nu niet gemonitord. En je hebt verschillende soorten van monitoring bijvoorbeeld is de site up of down, maar ook we hebben zoveel testomgevingen in de lucht staan en deze worden al heel lang niet gebruikt kunnen deze weg? Het gaat mij meer om het stuk overzicht hebben in alles wat we doen en wat voor keuzes hierin gemaakt gaan worden. Ja ik besef me dat dit een extra component is wat ik toevoeg, maar dat stukje overzicht is denk ik een beetje de rode draad in dit verhaal. Dit is wat we gaan doen en dit gaan we ermee bereiken.
Is er een budget voor deze tool? En zo ja hoeveel?
Nee er is niet een budget maar er is wel het simpele beredeneren van monitoring is nodig en als we verschillende smaken hebben dan gaan we vergelijken wat de beste oplossing is. En dat kan zijn bij Sentia waar we nu al een ton per jaar neerleggen voor de monitoring. Of dat je de applicaties self healend maakt en een paar OpsGenie accounts afneemt. Ik noem maar wat. Als je zo'n overweging kan maken dan kun je een business case maken. Prijs is hier dus een pro of een con in. En ja dat zal geld kosten maar we willen hier ook weer geen tonnen aan uitgeven.
Hoe belangrijk is het voor jou om een functie voor een on call schedule te hebben?
Als in wie heeft de dienst? Ja dat zou ik wel een interessante optie vinden. Nu hebben we een aantal mensen die vanuit verantwoordelijkheidsgevoel snel acteren, maar dan kom ik er toch op terug mensen moeten ook op vakantie kunnen. En dan zou ik het fijn vinden als andere hier de verantwoording voor oppakken.
Wil je uptime report kunnen genereren?
Ja graag. Nu moeten we aan klanten gaan vertellen dat we het niet weten en daar hebben we dan nu zelf wat voor ingesteld. En het zou wel fijn zijn als we dat aan onze klanten kunnen bieden. We komen ook bij grotere klanten en dit soort klanten willen dit gewoon weten.
Gebeurd er iets wanneer Sentia een issue oplost voor TamTam?
Ze zullen altijd een ticket aanmaken en aangeven wat ze gedaan hebben. Als het opgelost is dan geven ze dat aan en anders geven ze aan dat het aan de applicatie ligt en dan beland het bij onze developers. Ze houden ons dus goed op de hoogte via die tickets.

5. Interview Sandor

Wat voor rol heb je binnen TamTam?
Ik heb eigenlijk dezelfde rol als Arthur. Lead architect. Ik zit in de techniek ik ben developer, een soort technical director dus ik hou me bezig met het technische vlak van TamTam
Wat is je mening over de huidige monitoring?
Compleet onoverzichtelijk. Ik weet dat we Nagios hebben maar ik zou niet weten waar dit is te vinden. Wat ik in het algeheel mis is dat ik het liefste naar een site zou gaan en dan meteen kan zien wat wij hebben draaien en dat ik dan veel groene lichtjes zie.
Dus eigenlijk gewoon een dashboard waarop je bijvoorbeeld ziet Powned is up?
Ja per server, wat mij betreft niet allemaal op één dashboard maar onderverdeeld in de TAP van OTAP. Misschien hebben we dit in Nagios al, maar ik heb geen idee waar dit zit. Als er iets is wie krijgt hier dan een signaal van wat gebeurd er? Wat ik ook mis is hoe de monitoring is ingericht. Is dat nu gewoon een homepage en krijgen we 200 OK's dan is het goed? Want ik kan een groen lichte zien branden terwijl de site niet werkt. Dus kortom erg onduidelijk geen beeld missende sites.
Heb je al een voorkeur voor een tool?
Nee niet specifiek volgens mij is Nagios een beetje een tractor onder de monitoringsystemen en tractors zijn praktische apparaten. Pingdom biedt volgens mij wat meer visualisatie en dergelijken. New Relic gaat denk ik wat verder dan puur een stukje monitoring. Er zijn er genoeg. Ik kan mij voorstellen dat we er wel meerdere hebben dus dat Nagios bijvoorbeeld ons instap model is omdat dit goedkoop is en dan runt dit voor elke site. Dan kunnen we daarbij nog Pingdom verkopen als een klant zelf inzichten wil hebben. Daar kan je dus variaties op maken. Ik heb in ieder geval voorkeur voor een tool die in de Cloud opereert. En het zou ook mooi zijn als die tool met nieuwe functionaliteiten komt. En ik vind het belangrijk dat de tool goed kan integreren met Amazon en Azure. Dat deze tool geautomatiseerd acties kan ondernemen in het geval er iets mis is. Dus niet alleen alerting maar ook restarts van je applicatie. Als er bijvoorbeeld 10 minuten iets mis is dan zou je dit kunnen laten doen. Zelfs als het op een server van ons een app kan recyclen doordat je daar een cliënt op zet.
Waar moet het nieuwe systeem volgens jou aan voldoen?
Inzicht mogelijkheid, berichtgeving mogelijkheid en een hoger niveau dan e-mail bijvoorbeeld sms of WhatsApp. Ja of op basis van een app. Het liefst een open medium maar als het daarmee kan ook prima. En dus geautomatiseerde healing. Bij Nagios kan dit ook, maar dan moet je dit zelf helemaal gaan aansturen met powershell scriptjes en dan werkt dat niet en dan moet dat weer getest worden. Ik heb het dus echt over full fledged integratie met Amazon en Azure om dat soort dingen te doen. In de tool kan je dan dus zelf de app selecteren en dan wordt deze gemonitord. Dan zet je ergens een vinkje en wanneer de applicatie dan een paar keer niet op een ping reageert dan wordt er een restart gedaan. En verder denk ik dat het handig zou zijn als je een goed groepen/manager systeem in de tool hebt.
Wie denk je dat er een rol moeten gaan spelen in de nieuwe monitoring?
Joey vanuit QA. Het heeft wel wat weg van DevOps dus dat is Arthur en verder Leon vanuit Operaties en Marlies vanuit OS. En ik.

D Requirementsdocument

Requirementsdocument.

Monitoring

1 juni 2017

Versiebeheer

Datum	Samenvatting
8-2-17	Initiële opzet document. Eerste geconstateerde requirements uit het interview van Niek en Marlies toegevoegd.
9-2-17	Verwerken van interview met Arthur
13-2-17	Verwerken van interview met Leon
14-2-17	Verwerken van interview met Sandor en nogmaals doorlopen van de andere interviews.

Inhoudsopgave.

Versiebeheer **108**

1. Inleiding **110**

2. Scope en Stakeholders **111**

2.1 Scope 111

2.2 Stakeholders 112

3. Functionele requirements **113**

3.1 Requirements buiten scope 114

4. Andere requirements **115**

4.1 Niet functionele requirements 115

4.2 Business requirements 115

1. Inleiding

Dit rapport bevat de wensen en eisen die door TamTam gesteld worden aan het monitoring project. De requirements zijn gebaseerd op interviews met de stakeholders. In dit document zal eerst een scope opgesteld worden en de stakeholders zullen geïdentificeerd worden. Vervolgens zullen de requirements gedefinieerd worden met prioritiseringsmethode MOSCOW.

2. Scope en Stakeholders

2.1 Scope

Voor TamTam moet er een systeem gemaakt worden waarmee de servers gemonitord kunnen worden. De servers zullen eerst een standaardprotocol voor herstel uitvoeren wanneer er is geconstateerd dat er iets op die server kapot is. Wanneer dit het probleem niet oplost zal er door het systeem contact worden gezocht met de verantwoordelijke van het project dat daarop draait.

Om dit project te verwerkelijken zullen de volgende producten worden opgeleverd:

- Plan van Aanpak
- Requirementsdocument
- Een onderzoek naar bestaande oplossingen op gebied van self healing servers en of dit toepasbaar is bij TamTam
- Een self healend serversysteem dat meldingen stuurt naar de eindverantwoordelijke wanneer self healing mislukt
- Een overdracht dossier voor de DevOps afdeling met aanbevelingen over processen.

Er is een kans dat er requirements naar voren komen die wel belangrijk zijn maar niet binnen de scope vallen. Deze requirements zullen wel opgeschreven worden, maar er zal wel bij vermeld worden dat deze buiten de scope vallen.

Alle requirements waar geen MOSCOW vorm bij vermeld staat (could have, should have, must have en won’t have but would like) is automatisch een must have requirement. Alleen als er een ander type achter het requirement staat vermeld geldt dit type.

2.2 Stakeholders

De volgende stakeholders zijn betrokken bij deze oplossing

- **Klanten van TamTam**
  Voor de klanten van TamTam geldt dat hun website door dit systeem gemonitord zal worden.

- **Medewerkers van TamTam**
  Het eindproduct van dit project zal door de medewerkers van TamTam gebruikt gaan worden voor de monitoring van hun servers. De volgende mensen zullen hier veel mee gaan werken en daarom ook geïnterviewd worden.

- **Leon van Rees**
  Opdrachtgever van deze opdracht, Lead DevOps en Operational Services

- **Niek van Raaij**
  DevOps engineer

- **Marlies van der Schoot**
  Lead Operational Services

- **Arthur Stobbelaar**
  Architect

- **Sandor Voordes**
  Architect

3. Functionele requirements

Req	Requirement	Bron
1	Het systeem moet overzichten aanleveren waar de informatie uptime en reactietijd in staat.	Interview Niek, Leon, Sandor
2	Het systeem moet op functioneel en op infrastructuur niveau kunnen monitoren.	Interview Niek
3	Servers moeten een standaardprocedure met als doel het probleem oplossen doorlopen voordat ze een waarschuwing geven.	Interview Niek, Leon, Sandor
4	Het systeem moet contact kunnen opnemen met de juiste project owner wanneer de automatische procedure geen effect heeft gehad.	Interview Niek, Leon, Arthur
5	Het systeem moeten wanneer ze contact opnemen met de eindverantwoordelijke informatie opsturen over wat er al gedaan is door de server en welke foutmelding zich voordoet.	Interview Niek
6	De servicedesk moet een mail ontvangen zodra een site/server down is.	Interview Marlies
7	Het systeem moet rekening houden met het prioriteit niveau van de website	Interview Marlies
8	In het systeem moet er één locatie zijn waar wordt aangegeven welke sites up en welke down zijn.	Interview Arthur, Sandor
9	Het systeem moet een issue kunnen escaleren	Interview Arthur, Niek, Leon
10	Een gebruiker moet uptime checks kunnen toevoegen.	Interview Arthur
11	Het systeem moet testen of de site bereikbaar is en of deze de juiste pagina vertoont.	Interview Arthur, Sandor
12	DevOps, Projectteams en OS moeten tegelijk op de hoogte gesteld worden van een probleem	Interview Arthur
13	Het systeem moet actief pushen. Dit houdt in dat de tool contact blijft opnemen totdat je reageert. (**Could have)**	Interview Leon
14	Het systeem moet gemakkelijk te schalen zijn.	Interview Leon
15	Een gebruiker moet een check kunnen muten wanneer deze een server voor onderhoud down brengt.	Interview Leon
16	Het systeem moet een on-call functie heben. **(Could have)**	Interview Leon
17	Het systeem moet een uptime report kunnen genereren voor klanten	Interview Leon
18	Het systeem moet meerdere wijzes van contacteren hebben naast email	Interview Sandor
19	De self healing die door het systeem in werking wordt gezet moet volledig automatisch zijn.	Interview Sandor
20	Het systeem moet groepen kunnen managen.	Interview Sandor
21	Toevoegen van monitoring aan een project moet “1 druk op de knop” zijn	Gesprek Leon

3.1 Requirements buiten scope

Zoals in het hoofdstuk scope aangegeven was er een kans dat geconstateerde requirements wel belangrijk waren, maar buiten de scope vielen. De volgende requirements zijn wel van belang voor TamTam, maar vallen buiten de scope:

Req	Requirement	Bron
1	Het systeem moet custom checks kunnen toevoegen.	Interview Niek
2	Externe systemen waar systemen van TamTam afhankelijk van zijn moeten gemonitord worden	Interview Arthur
3	Als de nieuwe situatie live is dan moeten er tickets in JIRA geplaatst worden m.b.t. monitoring	Interview Arthur
4	De metadata in het CMDB moet correct en up to date zijn	Interview Leon

4. Andere requirements

4.1 Niet functionele requirements

Req	Requirement	Bron
NF1	Het systeem moet makkelijk in omgang zijn.	Interview Niek, Arthur, Leon, Marlies
NF2	Het liefst draaien de tools in de Cloud. Dit zodat deze niet ook in beheer van TamTam moeten komen.	Interview Sandor, Arthur
NF3	Projectteams moeten de monitoring kunnen inrichten	Interview Marlies, Niek, Arthur
NF4	DevOps, Projectteams en OS moeten van elkaar weten dat ze op de hoogte zijn van het probleem	Interview Arthur

4.2 Business requirements

Req	Requirement	Bron
BR1	Er moeten duidelijke afspraken worden gemaakt over de handelswijze als een server down gaat.	Interview Marlies, Leon
BR2	De verantwoordelijkheid moet duidelijk bij een rol neergelegd worden	Interview Marlies, Arthur
BR3	Er moeten minder tools gebruikt gaan worden.	Interview Niek
BR4	Wanneer er een project gestart wordt moet gelijk duidelijk zijn hoe de monitoring hiervoor gedaan gaat worden	Interview Leon, Sandor

Toolselectie Monitoring.

1 juni 2017

Inhoudsopgave.

1. Inleiding 119

2. Huidige situatie 120
2.1 Nagios Core 121
2.2 Pingdom & UptimeRobot 123
2.2.1 Pingdom 123
2.2.2 UptimeRobot 124
2.3 Amazon CloudWatch & Azure portal 125
2.3.1 CloudWatch 125
2.3.2 Azure portal 126
2.3.3 Handelswijze bij fouten 127
2.4 OpsGenie 128
2.5 New Relic 130
2.6 Sentia 132

3. Toolinventarisatie 134
3.1 Inventarisatie 134
3.2 Uptrends 135
3.3 Statuscake 137
3.4 Dynatrace 139
3.5 VictorOps 141
3.6 BigPanda 143
3.7 Pagerduty 144
3.8 Xmatters 146
3.9 Panopta 148

4. Monitoring bij grote bedrijven 150
4.1 Matrix bekende tools 151
4.2 Matrix onbekende tools 152
4.3 Datadog 153
4.3.1 Interessant voor TamTam? 154
4.4 Sentry 155
4.4.1 Interessant voor TamTam 156
4.5 Airbrake 157
4.5.1 Interessant voor TamTam? 158

5. Toolselectie nieuwe situatie 159

5.1 Functionaliteiten 160
5.2 Monitoring tool 161
5.2.1 Geschikte tools 162
5.2.2 Conclusie & opties 164
5.3 Alerting tool 166
5.3.1 Geschikte tools 167
5.3.2 Conclusie & opties 168

6. Advies 168

7. Verwijzingen 170

1. Inleiding

In dit document wordt gekeken naar het onderwerp monitoring binnen TamTam. Specifiek zal de focus liggen op de tooling. Er zal worden gekeken welke tools in de huidige situatie bij TamTam gebruikt worden en wat deze tools doen, kunnen doen en kosten.

Vervolgens zal er een inventarisatie worden gedaan waar tools die overeenkomen met de huidige tools geselecteerd worden. Ook van deze tools zal gespecificeerd worden wat ze kunnen en kosten.

Ook zal er gekeken worden hoe grote bedrijven hun monitoring aanpakken. Er zal gekeken worden of de tools uit de hierboven vernoemde stappen bij hen gebruikt worden. Ook wordt er gekeken welke tools vaker voorkomen maar nog niet in de eerste inventarisatie naar voren zijn gekomen. Van deze tools wordt vervolgens ook een opmaak gemaakt van functies en de kosten.

Daarna zullen er aan de hand van een requirementsdocument, wat tegen die tijd is opgesteld met behulp van interviews, eisen opgesteld worden waaraan de nieuwe monitoringsituatie moet voldoen. Deze eisen zullen tegenover de tools uitgezet worden. Hier zal vervolgens een advies voor de nieuwe situatie ontstaan.

2. Huidige situatie

In dit hoofdstuk is te lezen hoe de kijk op de huidige situatie is van de stakeholders en welke tools TamTam op het moment van schrijven (8-2-17) gebruikt om hun servers en applicaties te monitoren. Dit hoofdstuk wordt opgesteld met als doel inzicht te geven over de huidige situatie, de tools die TamTam op dit moment in bezit heeft en hoe intensief deze gebruikt worden. Ook wordt over de tools verteld wat deze kunnen en hoe duur ze zijn.

In de huidige situatie heeft TamTam meerdere situaties met betrekking tot hosting. De applicaties die zij nu levert worden op dit moment gehost bij Sentia Cloud, Amazon en Azure. De monitoring van deze hosts verschilt. De applicaties die bij Sentia staan worden op infrastructuur niveau gemonitord door Sentia. Wanneer er een fout optreedt in de applicatie dan wordt er contact opgenomen door Sentia met TamTam.

Azure en Amazon worden niet gemonitord door Sentia. Deze servers worden nu in de gaten gehouden via tooling zoals Nagios, Pingdom, UptimeRobot en OpsGenie. Ook hebben deze hosts zelf een portal waaruit meldingen gestuurd kunnen worden (Cloudwatch en Azure portal). Wanneer een server van Azure of Amazon down gaat dan gaat er een melding naar TamTam dat er problemen zijn op de server. Dit kan dan aan de infrastructuur liggen en opgelost zijn wanneer de applicatie opnieuw wordt opgestart. Het kan ook aan de applicatie zelf liggen. Wanneer dit het geval is zal het probleem door een programmeur die hier kennis van heeft opgepakt moeten worden.

Tijdens de interviews met de stakeholders heb ik hen gevraagd wat zij van de huidige situatie vonden. Ze reageerden hier allemaal op met de mededeling dat de huidige situatie niet voldoende was. Eén specifiek punt sprong naar voren tijdens de gesprekken. Dit punt was de onduidelijkheid over verantwoordelijkheid. Wanneer op dit moment iets kapot gaat en hier een melding over binnenkomt pakt of Leon of Niek dit op. Meestal hebben zij echter niet de kennis die nodig is om dit probleem op te lossen. Wanneer dit niet lukt nemen zij contact op met een developer en kijken ze de volgende dag of deze dit heeft gezien. Het is echter niet altijd bekend welke developer er verantwoordelijk is voor dat project.

Dit is overegins wel in overeenstemming met de SLA. TamTam reageert op applicatieproblemen binnen kantoortijden binnen zes uur.

Het is nu dus niet duidelijk wie waarvoor verantwoordelijk is. Leon zegt hier letterlijk over *“Ligt het probleem in de applicatie dan is het de vraag “en nu”? En we zijn nog zoekende hoe we hiermee moeten handelen. Op dit moment ga ik van persoon naar persoon. Kortom het is niet duidelijk wie waarvoor verantwoordelijk is.”*

2.1 Nagios Core

Nagios is een tool die op dit moment door Sentia beheerd wordt.
Met Nagios kan er zowel op functioneel als op infrastructuur niveau gemonitord worden.

Hieronder staan alle functionaliteiten die Nagios Core aanbiedt:

- Comprehensive monitoring
  Geeft een duidelijk overzicht van de kritieke punten van alle servers die gemonitord worden.

- Visibility
  Geeft een totaaloverzicht van alles wat draait met gedetailleerde status informatie.

- Awareness
  Geeft een melding via email of SMS wanneer een probleem zich voordoet zodat dit zo snel mogelijk opgelost kan worden.

- Problem remediation
  Event handlers kunnen automatisch applicaties, services, servers en apparaten opnieuw starten wanneer er een probleem wordt gevonden.

- Proactive planning
  Zorgt ervoor dat je upgrades kan inplannen.

- Reporting
  Kan gebruikt worden om aan te tonen dat SLA-overeenkomsten zijn behaald. Ook staat hier informatie over de kritieke problemen die zich voor hebben gedaan.

- Multi-tenant capabilities
  Meerdere gebruikers kunnen gebruik maken van Nagios met hun eigen specifieke views. Op deze manier kan je dus eventueel ook de klant toegang geven tot de informatie van alleen zijn website.

- Extendable architecture
  Meerdere API's kunnen gebruikt worden waardoor je de diensten van Nagios kan gebruiken in je eigen applicaties.

Nagios wordt op dit moment voor de volgende dingen gebruikt.

Op functioneel niveau wordt Nagios op de volgende manier gebruikt:

- Nagios kijkt naar de aanwezige URL's en vraagt aan de hand hiervan op of een Http/Https pagina werkt.
- Er is een custom check in ingebouwd die kijkt of een SSL nog geldig is.

Op infrastructuur niveau wordt Nagios op de volgende manier gebruikt:

- Is er genoeg geheugen beschikbaar?
- Is de disk al vol?
- Is er genoeg CPU vrij?

Deze controle kan alleen gedaan worden wanneer er een programma van Nagios is geïnstalleerd op de server die gecontroleerd moet worden. Op dit moment wordt de infrastructuur onderhouden door Sentia. Zij installeren het monitoring programma volledig automatisch op de server.

Kosten
Nagios is open source software en kost dus geen geld. Het is echter nu in beheer bij Sentia. Hierover valt meer te lezen in het hoofdstuk Sentia.

(Nagios, sd)

2.2 Pingdom & UptimeRobot

Pingdom en UptimeRobot zijn twee tools die in vergelijking met Nagios vrij weinig doen. Het enige wat deze tools doen is het pingen van een website. Wanneer dit mislukt dan sturen ze een melding in de vorm van een mail, sms of belletje. Van deze twee tools wordt Pingdom het meeste gebruikt. In UptimeRobot draaien een paar websites maar, deze tool moet uiteindelijk gaan verdwijnen.

2.2.1 Pingdom

Hieronder staan de functionaliteiten die Pingdom aanbiedt:

- Uptime monitoring
  Elke minuut test Pingdom of hij een antwoord krijgt van de door jouw gespecificeerde website.

- Real user monitoring
  Geeft inzicht in de performance van je website van echte bezoekers.

- Transaction monitoring
  Test of belangrijke acties in de site nog werken zoals een signup of zoekveld en het downloaden van files.

- Page speed monitoring
  Test de laadtijd van je website zodat je bottlenecks kunt vinden waarop je kan verbeteren.

- Reliability
  Als er een probleem is wordt dit met een second opinion getest zodat false alerts uit de meldingen worden gefilterd.

- Root cause analysis
  Constateert de reden waarom een website of server is uitgevallen zodat dit probleem opgelost kan worden.

- Incident management
  Verbeterd de workflow wanneer er een incident plaats vindt, waardoor de juiste persoon op de juiste tijd gecontacteerd wordt.

- API
  De API kan gebruikt worden om de functionaliteiten die je in de interface vindt te kunnen gebruiken in eigen applicaties.

Van deze functionaliteiten worden op dit moment alleen de uptime monitoring gebruikt en de alert functie. Deze functie verstuurd een melding naar OpsGenie die dit vervolgens verder afhandelt.

Kosten
Pingdom kost een vast bedrag per maand voor een vast aantal checks.
TamTam betaald hier €439,45 per maand voor.

(Pingdom, sd)

2.2.2 UptimeRobot

Hieronder staan de functionaliteiten van UptimeRobot:

- Uptime monitoring
  Met deze functionaliteit kan je testen of HTTP(S), ping, port en sleutelwoorden nog goed functioneren.

- Alerting
  Meld wanneer er een incident is via e-mail, SMS, Twitter, HipChat en nog veel meer.

- Statistics
  Laat je zien wat je uptime, downtime en response time is.

- Verification
  Test de downtime van meerdere locaties.

- Advanced verification
  Zorgt ervoor dat je zelf een profiel kan maken hoe UptimeRobot moet handelen wanneer een site down gaat.

- Maintenence windows
  Je kan tijdperiodes aangeven wanneer je niet gemonitord wilt worden.

- API
  De API kan gebruikt worden om de meeste functionaliteiten te kunnen gebruiken in eigen applicaties.

- Public status pages
  Een publieke pagina die je kan delen met je bedrijf en/of klant zodat zij inzicht hebben in de uptime van de site(s).

Ook UptimeRobot wordt op dit moment alleen gebruikt voor de uptime monitoring en alerts. Ook deze alerts worden via OpsGenie verstuurd.

Kosten
UptimeRobot is een tool die een gratis versie aanbiedt. Deze versie wordt op dit moment gebruikt.

(UptimeRobot, sd)

2.3 Amazon CloudWatch & Azure portal

Tijdens de transitie van lokale servers naar de Cloud zijn twee partijen gekozen waar TamTam klanten bij gaat hosten. Deze partijen zijn Amazon en Azure. Beide bieden een monitoring optie aan op het niveau van de infrastructuur die veel op elkaar lijkt. Overegins worden de Azure servers voor een deel ook door Sentia gemonitord.

2.3.1 CloudWatch

Hieronder staan de functionaliteiten van CloudWatch:

- Monitor Amazon EC2
  Deze functie zorgt ervoor dat je een overzicht krijgt van de metingen die op de server geconstateerd worden. Bijvoorbeeld het CPU-gebruik en de disk usage. Eventueel kan je het pakket detailed monitoring aanschaffen wat een specifieker overzicht geeft met meer data aanwezig.

- Monitor other AWS Resources
  Kan gebruikt worden om overzicht te krijgen over andere diensten die je hebt draaien bij AWS. Bijvoorbeeld de metingen op een Elastic load balancer of een Amazon RDS DB.

- Monitor Custom Metrics
  Geeft je de optie om metingen die je zelf uitvoert op je applicatie te monitoren via Amazon CloudWatch. Dit kan gebruikt worden om trends te spotten en om je applicatie te troubleshooten.

- Monitor and Store Logs
  Kan gebruikt worden om je logfiles naar CloudWatch te sturen en deze in CloudWatch te monitoren. Je kan deze informatie ook tegen betaling opslaan.

- Set Alarms
  Deze functie laat je een actie triggeren wanneer een server een bepaalde waarde overschrijdt. Je kan dan bijvoorbeeld een notificatie sturen naar de verantwoordelijke van deze server of je kan de server automatisch laten schalen.

- View Graphs en Statistics
  Toont de gegevens van je server over een periode van twee weken. Kan gebruikt worden om trends en dergelijken te constateren.

- Monitor and React to Resource Changes

CloudWatch geeft je de mogelijkheid om een workflow te maken die automatisch handelt op de manier die jij aangeeft.

Van de aanwezige functies in CloudWatch wordt nu vooral gebruik gemaakt van de Monitor EC2 functie. Voor film.nl wordt ook gebruik gemaakt van de Set Alarmfunctie. Deze verstuurt een melding naar OpsGenie die dit vervolgens afhandelt.

Kosten
CloudWatch heeft twee variaties een gratis versie en een uitgebreide versie. TamTam maakt gebruik van de gratis versie.

(Amazon, sd)

2.3.2 Azure portal

Hieronder staan de functionaliteiten van Azure portal:

- Quota's
  Deze functie geeft je een overzicht van hoeveel CPU, Memory, Bandwidth, etc.. tot je beschikking hebt.

- Metrics
  Met deze functie kan je de informatie van de server(s) zoals de gemiddelde responsetijd, het gemiddelde geheugen dat wordt verbruikt, en nog veel meer. Deze functie is geeft dus een overzicht van de infrastructuur van de server. Deze metingen worden opgeslagen zodat je ze later nog terug kan kijken. Metingen per minuut worden 48 uur bewaard, metingen per uur worden voor 30 dagen bewaard, en metingen per dag woorden 90 dagen bewaard.

- Alerts and Autoscaling
  Wanneer er iets met je server gebeurd kan je deze een alert laten versturen. Je kan zelf instellen wanneer een alert verstuurd moet worden bijvoorbeeld, wanneer de applicatie down is. Ook is er een optie aanwezig om automatisch te scalen wanneer je te weinig ruimte beschikbaar hebt of wanneer er te veel ruimte over is.

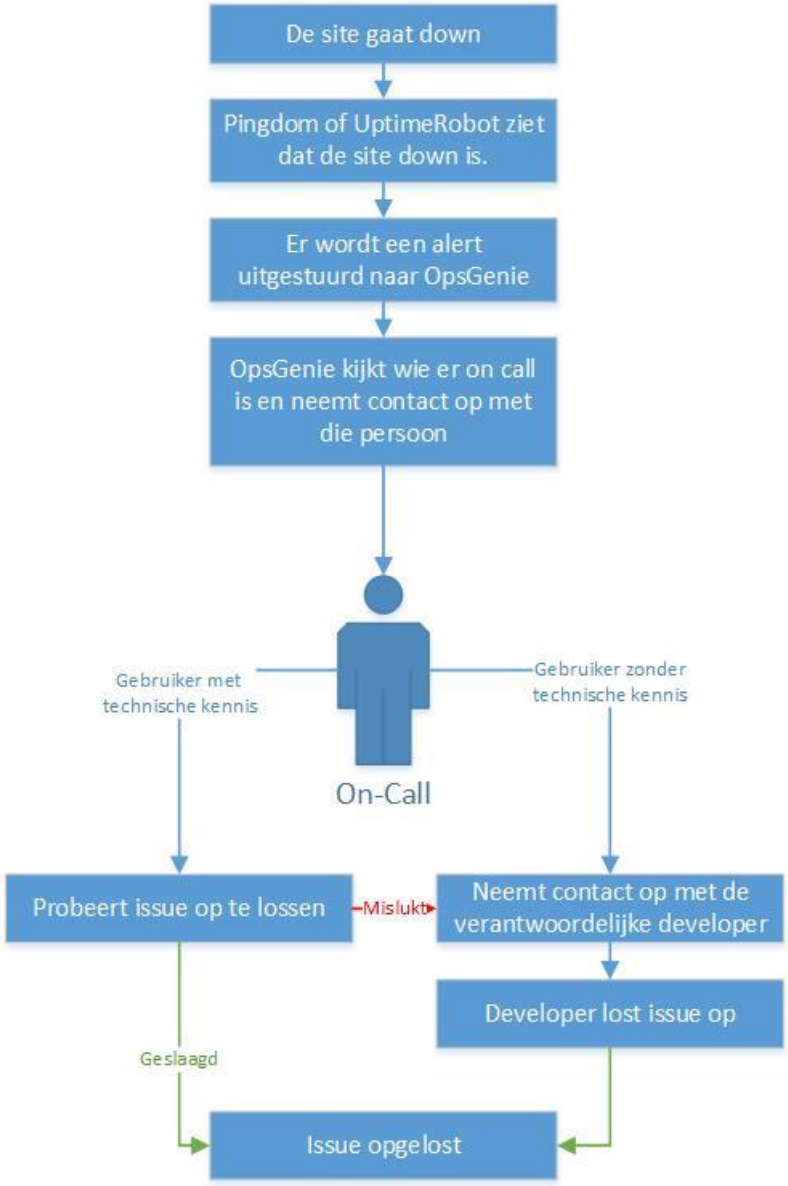
In het portal wordt nu gebruik gemaakt van de functie om de Quota's en Metrics in te zien. Ook is de Alert functie zo ingesteld dat deze tegen OpsGenie praat. Deze stuurt een melding naar OpsGenie die dit vervolgens afhandelt.

Kosten
Azures monitoring is gratis bij de azureproducten. Ze bieden zelf wel extra opties aan tegen betaling, maar dit wordt niet door TamTam afgenomen.

(Microsoft, sd)

2.3.3 Handelswijze bij fouten

De servers bij Amazon en een deel van de servers bij Azure worden gemonitord door TamTam zelf. In de huidige situatie wordt dit gedaan met behulp van de tools Pingdom en OpsGenie. De handelswijze ziet er als volgt uit:



Wanneer dit protocol wordt doorlopen is er kans dat de persoon On-Call geen technische kennis heeft. In dit geval neemt hij/zij contact op met de verantwoordelijke developer. Heeft de persoon On-Call wel kennis maar lukt het niet om de issue op te lossen dan zal ook hij/zij contact opnemen met de developer.

2.4 OpsGenie

Opsgenie wordt nu gebruikt om de meldingen van alle aanwezige monitoring tools te verwerken. Bij OpsGenie neem je gebruikersabbonementen af. Op dit moment neemt TamTam twee users af bij OpsGenie namelijk Leon van Rees en Niek van Raaij. Zij hebben nu een roterend schema. Om de week staat een van hen on call. Er staan nog niet veel sites in OpsGenie maar wel hele belangrijke sites. Wanneer er dus iets mis gaat wordt Leon of Niek hier gelijk van op de hoogte gesteld. Ervaring toont aan dat meestal als er iets fout gaat er gelijk meerdere dingen fout gaan. Waardoor er meerdere meldingen achter elkaar binnenkomen.

Hieronder staan de functionaliteiten van OpsGenie:

- Multiple Communication Chanels
  OpsGenie bied notificaties aan via mail, SMS, push alerts en een telefoon belletje.

- Rich Notifications
  De waarschuwingsnotificatie kan heel veel informatie bevatten. Er zit geen limiet op het aantal karakters. Ook kunnen er hele files bijgevoegd worden.

- On call Schedules
  Met deze functie kun je op een simpele manier een on call rooster maken.

- Detailed Tracking
  Gebruikers kunnen met deze functie zien wie op de hoogte is gesteld van het alert en of deze persoon al een actie heeft uitgevoerd.

- Escalations
  Er is ook een optie aanwezig om een escalatie in te stellen. Deze functie notificeert automatisch de volgende persoon als de eerste gecontacteerde niet reageert.

- Phone Call Routing
  Je kan binnenkomende support calls automatisch doorverwijzen naar de juiste persoon. Is er niemand aanwezig, dan zorgt OpsGenie ervoor dat de boodschap wordt opgenomen en als alert wordt doorgestuurd.

- Seamless Integration
  Naast de OpsGenie app kan OpsGenie ook integreren met teamtools zoals HipChat.

- Alert Actions
  Deze functionaliteit biedt gebruikers de keuze om te reageren op alerts op andere manier dan de standaard acties “Add Note” en “Close”. Deze acties moeten wel gespecificeerd worden.

- Heartbeat Monitoring
  Elke keer als een gespecificeerde actie wordt uitgevoerd ontvangt OpsGenie een melding. Wordt deze melding niet ontvangen dan geeft OpsGenie een melding.

In OpsGenie wordt nu gebruik gemaakt van de Multiple Communication Chanels, De On call Schedules, Escalations, Detailed Tracking, Seamless Integration, Alert Actions en Rich Notifications.

Kosten
OpsGenie is een tool die een pay per user betalingsplan heeft. TamTam heeft op het moment zes gebruikers en betaald nu aan OpsGenie in totaal $91,- per maand.

(OpsGenie, sd)

2.5 New Relic

New Relic is een monitoringtool op het gebied van applicaties. Je kan deze tool dus gebruiken om inzichten te krijgen in de prestaties en knelpunten van je applicaties. New Relic ondersteund de programeertalen Ruby, Java, Node, PHP, .NET, Python en Go. Aangezien TamTam PHP en .NET gebruikt voor haar applicaties is de ondersteuning op het gebied van talen voldoende.

Verder is er ook al besloten dat dit onderdeel van monitoring buiten de scope valt voor deze opdracht.

Hieronder staan de functionaliteiten die New Relic aanbiedt:

- **Application Monitoring**
  Deze functie geeft inzicht in de prestaties van je applicaties. Je kan zien hoe snel de response tijd is van je pagina, Het aantal verzoeken dat per minuut door de applicatie wordt gestuurd en je kan het percentage van foutmeldingen over een in te stellen periode van tijd monitoren. Deze informatie kan je analyseren door middel van histogrammen. Je kan een overzicht krijgen van je transacties. Wat deze doen, welke lang duren en aan welk proces in de transactie dit ligt.

  Verder zitten er in de PRO versie nog features zoals het monitoren van belangrijke business transacties. Ook Transaction breakdowns zitten in de PRO versie. Hiermee kan je precies inzien wat er in de transactie gebeurd en hoeveel tijd dit kost. In de PRO versie heb je ook toegang tot X-Ray Sessions. Hiermee kan je nog uitgebreider je transacties analyseren door middel van grafieken.Verder worden er nog een paar minder interessante features in de PRO editie aangeboden.

- **Database Monitoring**
  Met deze functie kan je inzicht krijgen in de prestaties van je database. Je kan zien hoeveel tijd er in database calls wordt doorgebracht. Ook kan je van die call zien wat de reactie tijd en doorvoer tijd van deze call is. Verder kan je SQL rapportages uitdraaien over de traagste SQL transacties. In deze rapportage staat vervolgens informatie die je kan gebruiken om de transacties sneller te laten verlopen. Ook kan je een overzicht krijgen van alle aanwezige databases die je monitord. Je hebt dan direct een overzicht van al je databases op één dashboard.

- **Availability & Error Monitoring**
  Deze functie is eigenlijk te vergelijken met de standaardfunctie van Pingdom. De functie pingt een website en wanneer deze geen antwoord krijgt verstuurd hij een alert. Deze alerts kun je vervolgens net zoals bij Pingdom laten versturen naar een tool zoals OpsGenie zodat de juiste mensen op de hoogte worden gesteld van het probleem.

- **Reports**
  Met deze functie kan je eigen dashboards ontwikkelen. Je kan alle data die je via New Relic tracked terug laten komen op dit dashboard. Het dashboard is dus volledig naar eigen voorkeur in te stellen. Verder geeft deze functie inzichten in de availibility, scalability, capacity en host-usage door middel van rapportages. Ook is er in de PRO versie een optie om SLA rapportages te genereren.

- **Team Collaboration**
  Met deze functie kan je bij reports notities toevoegen voor je collega's. Verder kan je hiermee in de PRO versie gebruik maken van Multi-Accountmanagement en Integrated Collaboration Tools. Multi-Accountmanagement kan je gebruiken om gebruikers alleen bepaalde rechten te geven en met Integrated Collaboration Tools kan je een ticketsysteem zoals JIRA aan New Relic vast maken.

- **Security**
  Verder garandeert New Relic dat je gegevens veilig zijn. Je kan bijvoorbeeld zelf bepalen welke gevoelige data wel en welke niet naar New Relic verstuurd wordt. In de PRO versie kan je eventueel zelfs gebruik maken van Enterprise Mode. In deze modus kan je bepaalde veiligheidsmaatregelen vaststellen zodat er niet per ongeluk gevoelige data verstuurd wordt.

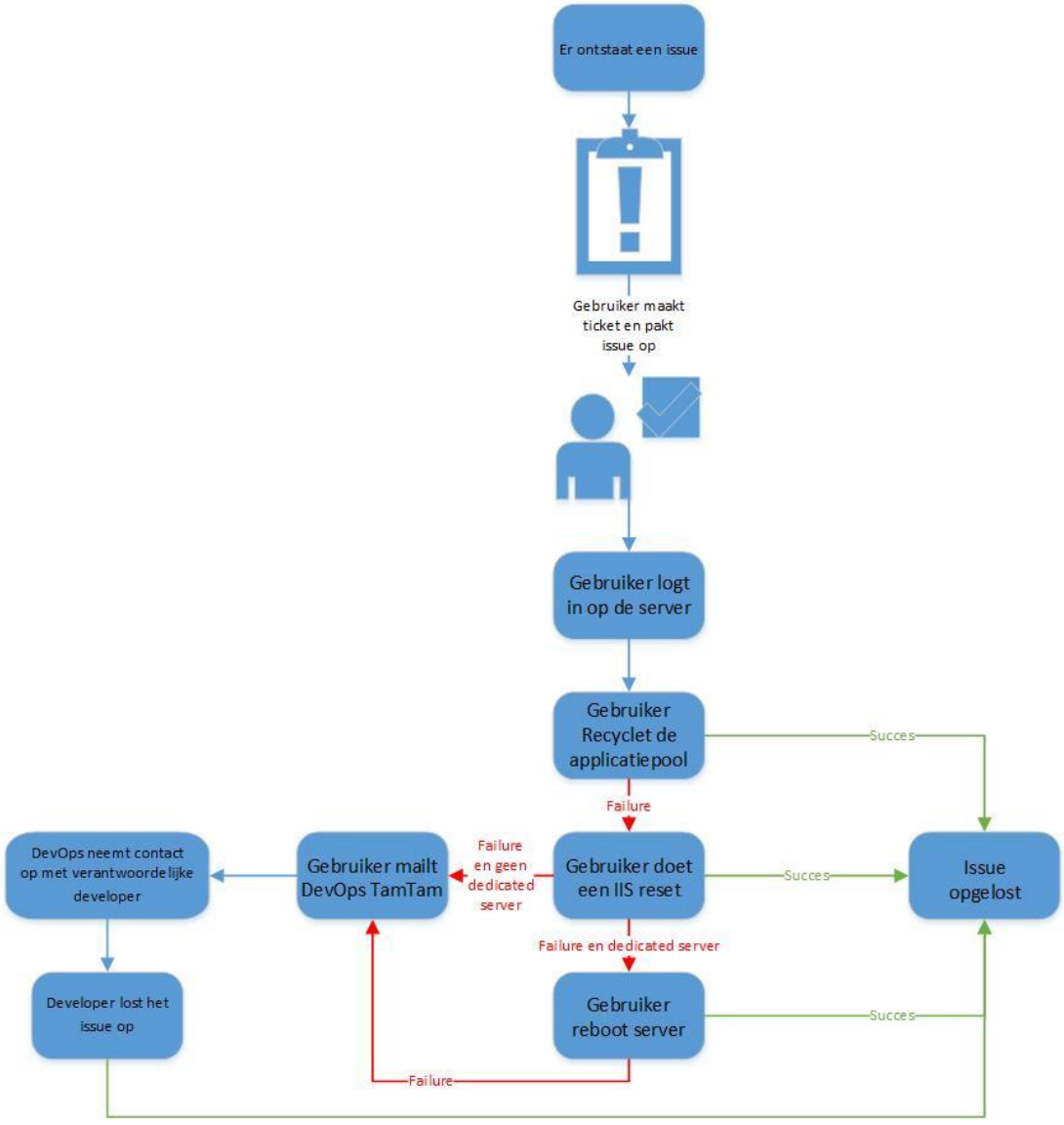
Kosten
In de huidige situatie neemt TamTam bij New Relic een abbonement van $199,- per maand af.

(New Relic, sd)

2.6 Sentia

Er zijn ook nog servers die nu door Sentia gehost en gemonitord worden met behulp van Nagios. Dit zijn servers die in Sentia Cloud staan en een deel van de servers bij Azure. De monitoring van Sentia houdt in dat ze een standaardprotocol doorlopen wanneer er iets fout gaat met een applicatie.

Het protocol wat nu door Sentia doorlopen wordt ziet er als volgt uit:



Wanneer dit protocol doorlopen wordt zijn er twee mogelijke uitkomsten. Het issue is opgelost en de applicatie werkt weer. Als dit het geval is dan wordt het aangemaakte ticket in Sentia bijgewerkt met

de handelingen die zijn uitgevoerd om het issue op te lossen. De andere uitkomst is dat er nog steeds een probleem is met de applicatie. In dit geval neemt de gebruiker van Sentia contact op met de DevOps afdeling van TamTam.

Kosten

Sentia brengt hun kosten in rekening. Deze rekening wordt aan TamTam aangeboden. Dit bedraagt zomaar een ton per jaar. Het is lastig in te schatten hoeveel geld er precies per maand wordt betaald aan Sentia. Dit komt doordat Sentia de kosten van het verwerken van issues in rekening brengt. Als er dus een maand niets gebeurd met de servers kost dit niets. Vallen alle servers dagelijks weg dan kost dit veel geld.

3. Toolinventarisatie

In dit hoofdstuk wordt er gekeken naar tools aanwezig op de markt voor monitoring. Op dit moment heeft TamTam al enkele monitoring tools in gebruik (zie Huidige situatie monitoring). Het doel van dit hoofdstuk is om soortgelijke tools te bekijken. Er zal gekeken worden naar specificaties, gebruikersgemak en kosten. Uiteindelijk zal aan de hand van dit hoofdstuk een keuze kunnen worden gemaakt tussen de huidige tools en de aanwezige tools op de markt.

3.1 Inventarisatie

Voor dit document is eerst een inventarisatie uitgevoerd van aanwezig tools op de markt. Door eerst tools te verzamelen die overeenkomsten hadden met de huidige tools en voldeden aan de eisen uit het requirementsdocument, kon er daarna uitgebreider onderzoek gedaan worden naar de geselecteerde tools. Uit deze selectie zijn de volgende tools naar voren gekomen:

- Uptrends
- Statuscake
- Dynatrace
- VictorOps
- BigPanda
- Pagerduty
- Xmatters
- Panopta

In dit hoofdstuk worden alle bovegenoemde tools verder uitgewerkt.

3.2 Uptrends

Uptrends is een functionele tool die de klikpaden op je website kan monitoren. Je kan hiermee bijvoorbeeld login en zoekfuncties testen. Uptrends is als tool te vergelijken met Pingdom aangezien je er uptime mee kan monitoren. Uptrends is echter slimmer dan Pingdom omdat je hier ook klikpaden mee kan monitoren. Terwijl Pingdom alleen de site pinged.

Hieronder staan de functionaliteiten die Uptrends aanbiedt:

- Locatiemonitoring
  Zorgt ervoor dat je zelf kan bepalen waarvandaan de bezoekers komen. Uptrends biedt 162 locaties aan.

- Multi browser monitoring
  Uptrends test de data in verschillende sites en laadt de site in zoals de gebruikers dit zien.

- Deel en exporteer data
  Laat je de monitoring data delen via pdf, Excel of e-mail.

- Alert escalaties
  Zorgt ervoor dat je escalaties kan instellen zodat de juiste mensen worden gewaarschuwd bij problemen

- 1 minuut monitoring
  Zorgt ervoor dat je de uptime kan monitoren met intervallen van 1 minuut.

- Root cause analyse
  Levert automatische traceroutes aan en screenshots van fouten.

- Publieke status pagina
  Een pagina waarmee je de uptime en performance van je websites publiekelijk toegankelijk kan maken.

- API
  Zorgt ervoor dat je data vanuit je account kan versturen zoals de huidige status, performance statistieken en meer.

Kosten

Uptrends heeft twee abonnementen voor geavanceerde website monitoring. Het business en het Enterprise plan. Het businessplan biedt het volgende aan:
Volledig aan te passen Ongelimiteerde Operators, Geavanceerde Dashboards, Publieke Status Pagina, Multi Browser Monitoring, Mobile Website Monitoring, Transacties, 600 sms alerts, E-mail en Tel. Support hier betaal je eerst met 20% korting €19,73 per maand voor en als de korting stopt €24,67 per maand.

De Enterprise versie biedt exact hetzelfde aan met de volgende opties als extra: Transactie Waterval, Transactie Screenshots en 1200 sms alerts. Hier betaal je eerst met 20% korting €35,71 per maand voor en als de korting stopt €44,63 per maand.

(Uptrends, sd)

3.3 Statuscake

Statuscake is een alternatief voor Pingdom. Zelf maken ze op hun site reclame dat ze het beste alternatief zijn voor Pingdom. Statuscake kijkt of de website die jij specificeert nog in de lucht is.

Hieronder staan de functionaliteiten die Statuscake aanbiedt:

- Tests toevoegen
  Deze functionaliteit spreekt voor zich. Je kan hier test toevoegen. Je kan kiezen tussen de types HTTP, TCP, DNS, SMTP, SSH, PING en PUSH. Vervolgens geef je de test een naam en kan je een contactgroep toevoegen. Als je de premium functie hebt kan je ook test locaties toevoegen. Vervolgens kan je de treshold instellen. Dit bepaald wanneer er een alert wordt gestuurd en hoeveel confirmatie servers er nog tussen zitten. Vervolgens kan je instellen per hoeveel minuten er wordt gekeken naar de site en hoe lang er wordt gewacht voordat je website als down wordt geclassificeerd. Er is ook een optie voor bulk insert. Dit laat je meerdere URL's tegelijk toevoegen.

- SSL monitoring
  Een functie om je SSL certificaten te monitoren. Wanneer een certificaat bijna verloopt krijg je hier een melding van.

- Page Speed
  Hiermee kan je de performance van je site in de gaten houden. Je kan alerts instellen wanneer je site onder een performance komt.

- Domain monitoring
  Hiermee kan je je domein monitoren. Je kan bijvoorbeeld zien wanneer je domein verloopt.

- Virus Checking
  Statuscake checkt de site die wordt geladen voor bekende malware.

- Content Match
  Dit laat je testen of een bepaalde string wordt teruggevonden op de site.

- Real Browser Testing
  Geeft een testresultaat van de performance met de laatste Chromebrowser builds.

- Custom Status Codes
  Hiermee kan je een alert sturen naar een statuscake van jouw keuze.

- Email Templating
  Laat je de email alerts zo ontwerpen als jij zelf wil.

- Alerting
  Deze functie laat je een alert sturen wanneer een site down is door middel van een sms'je of een email.

- Integrations
  Statuscake integreert met een paar third party services. Je kan je alerts dan via de gekozen third party laten binnenkomen. Hier zitten onder andere VictorOps, OpsGenie en Pagerduty als opties tussen.

Kosten
Statuscake heeft drie verschillende pakketten die zij aanbieden.

Het eerste pakket is gratis. Hier kan je testen of je pagina live is en dit kan elke 5 minuten. Deze test wordt gedaan vanaf een willekeurige locatie. Ook kan je contacten en groepen voor de alerts instellen en gebruik maken van integraties.

Het tweede pakket heet Superior en kost €24,19 per maand. Met dit pakket kan je elke minuut checken. Verder heb je toegang tot de volgende functies: alle functies van de gratis versie, 25 sms credits, keuze uit acht locaties, Bulk Insert, Virus checking, Content Matching, Real Browser Testing, Custom Status Codes, Brand Free Publishing, Domain Monitoring, Page Speed Monitoring en SSL Monitoring.

Het derde pakket heet Business en kost €84,69 per maand. Met dit pakket kan je elke 30 seconden checken. Ook heb je 400 sms credits tot je beschikking en keuze uit alle aanwezige locaties. Ook heb je alle opties uit het Superior en Free pakket en krijg je daarnaast nog de optie van Email Templating.

(Statuscake, sd)

3.4 Dynatrace

Dynatrace is een tool waar je heel veel mee kan op het gebied van monitoring. De tool kan dingen zoals uptime maar ook de applicatie in de gaten houden. Verder integreert het met veel andere monitoringsystemen zodat je de configuratie zelf kan optimaliseren.

Hieronder staan de functionaliteiten die Dynatrace aanbiedt:

- Business and performance analytics
  Deze functie biedt je een uitgebreid inzicht in hoe de klanten met de applicatie interacteren. Je kan inzien met welke browser je klanten de site bezoeken, je kan de user experience van je klanten zien (Satisfied, Tolerating of Frustrated), Je kan zien waarvandaan je bezocht wordt en vanuit welk OS. Ook kan je met Root cause analysis inzien hoe een probleem is ontstaan.

- Digital experience management
  Met deze functie krijg je een inzicht in de gebruikerservaring met jouw webpagina. Je kan hier zien hoe de pagina voor de gebruiker laad. Ook biedt deze functie inzicht in je app als je deze hebt. Je kan dan zien waarmee een gebruiker je app bezoekt en als ze tegen problemen aanlopen hoe dit komt.

- Applicatoin monitoring and performance lifecycle management
  Dynatrace biedt ondersteuning voor applicatie monitoring. Ze ondersteunen de talen Java, .NET, Node.js en PHP. Deze functie laat je zien hoe lang een aanroep van je methode duurt voor je gebruiker. Je kan ook inzien waar in je code bottlenecks voor je CPU of netwerk oplevert.

  Ook databases die aan je oplossingen vast zitten worden automatisch gedetecteerd door Dynatrace. Voor elk statement dat er naar je database gaat kan je zien hoe lang dit heeft geduurd. Ook kan je inzicht verkrijgen in wat er met je database gebeurt. Zo kan je zien hoeveel statements er per seconde worden uitgevoerd, het CPU-gebruik en de actieve sessies. Dynatrace integreert met databases zoals: SQL Server, MySQL, PostgreSQL, OracleDB, MongoDB, Memcached en Redis.

  Ook kan je met deze functie Load tests uitvoeren op je applicatie.

- Cloud, container and infrastructure monitoring
  Dit is de monitoringsfunctie die Dynatrace aanbiedt. Je kan hiermee de infrastructuur van je servers monitoren (CPU, memory, etc...), van je netwerk connecties, van je virtuele monitors, van je microservices en van je Cloud. Ook kan je je logs gemakkelijk analyseren.

Kosten
De kosten van Dynatrace zijn *"Pay-as-you-go"*. Dit houdt in dat je als je gebruik maakt van Dynatrace alleen betaald voor de diensten die je afneemt.

Per host betaal je $0.30 per uur. Wanneer je een host 24/7 in de maand laat draaien dan kost je dit $216 per maand.
Je betaalt per 100 sessies $0.30.
Je betaalt per 50 web check runs $0.30.

(Dynatrace, sd)

3.5 VictorOps

VictorOps is te vergelijken met OpsGenie, het is een alerting tool. Beide hebben functionaliteiten die integreren met monitoringtools. De functies van VictorOps zijn echter niet zo gemakkelijk te vinden op de site als bij OpsGenie. In VictorOps staan een deel van de functionaliteiten uitgelegd onder demo en de rest van de informatie is in de “Knowledge base” te vinden wat onder resources verstopt staat. OpsGenie bied deze informatie direct op de site zelf aan onder features.

Hieronder staan de functionaliteiten die VictorOps aanbiedt:

- Incident timeline
  Dit is een feature die ervoor zorgt dat het incident en de daarbij genomen stappen in een duidelijke tijdlijn worden getoond. Dit kan handig zijn om te zien welke stappen teamleden al genomen hebben om het probleem op te lossen.

- On-call management
  Laat je op een simpele manier on call diensten waarnemen voor je collega’s die niet kunnen op dat moment. Stelt ook automatisch de andere teamleden op de hoogte.

- Post-mortem reporting
  Met deze functie kan je het probleem van start tot oplossing documenteren. Alle acties uit de tijdlijn worden geïmporteerd. Je kan acties die niet van belang zijn weglaten en commentaar geven op belangrijke acties. Vervolgens kan je dit rapport opslaan of afdrukken.

- Transmogrifier alert annotations and transformations
  Dit zorgt ervoor dat je automatisch annotaties kan toevoegen wanneer een server een alarm afgeeft. VictorOps geeft hier onder andere aan *“You already know the first steps you will take when you wake up in the middle of the night. So why not automate them?”* In de demo blijkt dat er naar een standaardprotocol gelinkt wordt en dit niet automatisch wordt uitgevoerd. Je kan hiermee ook bepaalde alarms naar info’s veranderen waardoor ze je niet wakker pagen. Ook kan je specifieke incidenten naar een specifiek team sturen. Verder kan je gebruik maken van “Hubot” een robot die je bijvoorbeeld een command kan geven om een server te herstarten.

- Control call
  Een optie die ervoor zorgt dat je snel een call aan kan maken wanneer er een groot probleem ontstaat. Aan deze call kan je alle benodigde teamleden toevoegen die dan een page ontvangen.

- Scheduling
  Er is een optie om een on call schedule aan te maken in de tool.

- Alerting
  VictorOps kan zijn gebruikers alerten via email, sms, push bericht en door te bellen. Ook kan elke gebruiker de manier waarop zij gealert worden zelf instellen.

Kosten
Net zoals OpsGenie werkt VictorOps ook met een betaal voor elke gebruiker systeem. Je hebt de keuze tussen twee pakketten. Het Standard en het Enterprise pakket.

Het Standard pakket kost $29,- per maand als je per jaar betaald of $34,- per maand wanneer je per maand betaald. In het Standard pakket zitten de volgende features: Unlimited Alerts, Advanced scheduling, Incident timeline, Native collaboration en Slack/HipChat integration, 500 api calls, Post-mortem reporting en support via email and phone.

Het Enterprise pakket kost $49,- per maand als je per jaar betaald of $59,- per maand als je per maand betaald. In het Enterprise pakket zitten de volgende features: alles wat in het Standard pakket zit, The transmogrifier, Enterprise Integrations, SSO, Built in voice conferencing, Unlimited API calls, Advanced reporting, Premium success management.

(VictorOps, sd)

3.6 BigPanda

BigPanda is te vergelijken met OpsGenie. Net zoals OpsGenie en VictorOps heeft BigPanda de mogelijkheid om te integreren met monitoringtools. Er zit echter geen alerting functie in die direct een probleem vermeld.

Hieronder staan de functionaliteiten die BigPanda aanbiedt:

- Automated Alert Correlation
  BigPanda trekt zelf automatisch verbanden tussen de meldingen die binnenkomen en bundelt deze samen in één incident. Waardoor er een duidelijker overzicht is wat waar precies fout gaat.

- Smart Ticketing
  In BigPanda zit een functionaliteit waarmee je automatisch tickets kan aanmaken in bijvoorbeeld JIRA. Deze tickets bevatten dan informatie over een geheel incident en niet per alert wat bij dat incident hoort.

- Centralized Visibility
  Deze functie laat je je monitoring tools integreren met BigPanda. Hierdoor ontstaat er één plek waar alle informatie van je servers terecht komt.

- Custom Views
  Deze functie zorgt ervoor dat je groepen kan aanmaken. Als bij TamTam bijvoorbeeld de groep Oxxio aangemaakt wordt dan kan in deze groep alle informatie en alerts van Oxxio gemanaged worden.

- Sharing function
  Je kan incidents sharen via SMS of email. Deze worden op een standaard manier gegenereerd en verzonden. Je hebt daar dus zelf geen invloed op.

Kosten
BigPanda vermeld helaas niet duidelijk op de site hoeveel geld zij vragen voor hun pakketten. Wel is duidelijk dat het enige verschil in pakket (Standard, Enterprise en Volume) het aantal monitored nodes is. De Standard versie ondersteund 2000 nodes, de Enterprise versie 10.000 en de Volume versie 10.000+. Verder krijg je dus alle hierboven genoemde functionaliteiten bij elk pakket.

(BigPanda, sd)

3.7 Pagerduty

Pagerduty is een alerting tool net zoals VictorOps en OpsGenie. Ook Pagerduty integreert met monitoring tools zoals Nagios.

Hieronder staan de functionaliteiten die Pagerduty aanbiedt:

- Event Grouping & Enrichment
  Met deze functie kan je alerts onder groeperen in één incident zodat je niet te veel alerts achter elkaar krijgt. Aan deze incidenten kan je inzichten toevoegen en instructies over hoe er gehandeld moet worden. Ook kan je meldingen bij groepen indelen en eventueel meldingen onderdrukken.

- Reliable and Rich Alerting
  Deze functie houd in dat er alerts kunnen binnen komen via SMS, pushberichten, email of via een telefoontje. Je kan de alerts aan de hand van urgentie laten afgaan. Je kan je eigen persoonlijke voorkeuren instellen voor de manier waarop je gealert wilt worden.

- Easy Scheduling
  Deze functie laat je een On-Call rooster maken. Je hebt hierbij de opties wekelijks, weekdagen, weekends en follow-the-sun. Je kan dit zelf ook nog extra bewerken als je niet van de standaards gebruik wil maken. Wanneer een On-Call dienst begint krijgt die persoon een notificatie. Verder kan je een rooster override doen wanneer een teamlid op vakantie gaat of ziek is. Ook kan je het rooster toevoegen aan je Google Calendar, Outlook en iCal.

- Mobile Incident Management
  Pagerduty biedt ook een app aan zodat je overal op de hoogte kan worden gehouden.

- Real-Time Collaboration
  Deze functie zorgt ervoor dat je stakeholders aan een probleem toevoegen. Ook biedt deze functie een optie om tools zoals via tools zoals HipChat, Atlassian en Slack alerts te sturen. Verder kan je alerts automatisch in je ticketsysteem opnemen.

- System & User Reporting
  Met deze functie kan je een duidelijk overzicht krijgen van hoeveel alerts er afgehandeld worden en hoe je teams presteren. Aan de hand hiervan kan je eventueel reageren door je team te vergroten als er teveel alerts binnen komen. Daarnaast kan je ook inzicht krijgen in welke applicatie veel incidenten vertoond.

- Always on Reliability
  Pagerduty garandeert dat ze nooit down gaan. Ze zijn hiervoor heel sterk verzekerd. Ook draait hun oplossing over meerdere datacenters en hosts zodat ze altijd available blijven.

- Enterprise-Grade Security & Controls
  Pagerduty geeft aan dat alle data geëncrypteerd wordt. Daarnaast hebben ze eigen firewalls en voldoen de datacenters die ze gebruiken aan de ISO 27001 en FISMA norm. Ook heb je als admin role control over je users. Hierdoor kan je zelf bepalen wat een user wel en niet mag.

- Service Grouping
  Laat je zien hoe de services er binnen de IT-infrastructuur uitzien. Het gaat hier dan bijvoorbeeld om tools zoals Pingdom.

- Platform Extensibility
  Pagerduty heeft op dit moment het grootste aantal integraties (150+) ten opzichte van hun concurrentie.

- Live Call Routing
  Zorgt ervoor dat een incident gerapporteerd door een gebruiker gelijk doorverwezen wordt naar de juiste persoon. Reageert deze persoon niet dan kan de gebruiker een boodschap achterlaten die aan het issue wordt gekoppeld.

Kosten
Pagerduty biedt twee pakketten aan die interessant zijn voor TamTam, het Basic en het Standard pakket.

Het Basic pakket kost $29,- per maand als je per jaar betaald of $34 dollar per maand wanneer je per maand betaald. In dit pakket zitten de volgende functionaliteiten: 25 globale smsjes of telefoontjes, Reliable notifications, Automatic Escalation, Support Knowledge Base, Alert Triage, Response Notes, On-call Schedules, Incident Urgencies, Unlimited API Requests, Email support en 1 Year Data Retention.

Het Standard pakket kost $49,- per maand als je per jaar betaald of $59,- per maand wanneer je per jaar betaald. In dit pakket zitten de volgende functionaliteiten: Alles wat in de Basic versie zit, Unlimited global sms and phone notifications, Enterprise integrations, Workflow extensions, Advanced Analytics, Single Sign-On, Team Organization, Custom incident Urgencies, Response Mobilizer, Response Bridge, Phone + Email support, Unlimited Data Retention, Service Agreements, Suppresion, Custom Permissions, Incident Subscription, Stakeholder User Licenses.

(Pagerduty, sd)

3.8 Xmatters

Xmatters biedt verschillende features aan op het gebied van monitoring. De oplossing waar dit hoofdstuk over gaat heet IT Management. Dit is een alerting tool vergelijkbaar met OpsGenie, VictorOps en Pagerduty.

Hieronder staan de functionaliteiten die Xmatters aanbiedt:

- Share Context Through Integrations
  Met deze functie kan je monitoring tools laten integreren met Xmatters. Ook tools die de optie hebben om een mail te versturen kan je integreren met Xmatters. Verder is er ook de mogelijkheid om zelf een integratie te bouwen.

- On-Call Scheduling
  Met deze functie kan je een on-call rooster maken. Ook kan je gemakkelijk tijdelijke vervanging regelen in het gevan van ziekte of vakantie.

- Dynamic Groups
  Deze functie laat je groepen aanmaken. Zo kan je groepen aanmaken van bijvoorbeeld een specifiek team.

- Customized Views
  Je kan je eigen view maken zodat je direct een overzicht krijgt van wat voor jou van belang is. Zo zie je gelijk wanneer je inlogt op Xmatters hoe jouw applicaties ervoor staan.

- Collaborate Across Teams
  Je kan simpel een team aanmaken wanneer er een alert afgaat. Dit zorgt ervoor dat de samenwerking als er een probleem ontstaat soepel verloopt. Je kan dit eventueel ook via de een geïntegreerde chat tool doen.

- Subscribe to Key Updates
  Deze functie is handig als je op de hoogte wil zijn van problemen waar je zelf niet direct bij betrokken ben. Je hoeft dan niet je collega's lastig te vallen die het probleem proberen op te lossen.

- Manage Communications Anywhere
  Xmatters heeft een app beschikbaar zodat je overal alerts kan ontvangen. Je kan vanuit de app ook je rooster zien, contact opnemen met teams of eigen groepen aanmaken en daar contact mee opnemen.

Kosten
Xmatters biedt drie verschillende pakketten aan. Starter, Base en Advanced. De pakketten hebben de volgende specificaties en prijzen:

Starter kost $16 per gebruiker per maand. In starter heb je toegang tot: ongelimiteerde integraties, Communicatie via SMS, spraak, mobiel, push, pager en email, 10 rollen, 6 integratie events per minuut, handmatige alerting, conference call alerting (GEEN CALLS), support tijdens kantooruren, uptime van 99.5%, ongelimiteerde push/email berichten en 25 SMS en spraakberichten per maand per gebruiker.

Base kost $39 per gebruiker per maand. In base heb je toegang tot: alle functies uit Starter, ongelimiteerd aantal rollen, 15 integratie events per minuut, een non-production environment, 24/7 support, uptime van 99.9%, meerdere user types, Multi linguale berichten en 50 SMS en spraakberichten per maand per gebruiker.

Advanced kost $59 per gebruiker per maand. In advanced heb je toegang tot: alle functies uit Base, 30 integratie events per minuut, Communicatie center, Messaging design tools, Client succes manager, uptime van 99.95% en 100 SMS en spraakberichten per maand per gebruiker.

(Xmatters, sd)

3.9 Panopta

Panopta is een monitoring en alerting tool. Het mooie van deze tool is dus dat het zowel functies in monitoring als in alerting heeft.

Hieronder staan de functionaliteiten die Panopta aanbiedt:

- Monitoring
  Met deze functie kan je de monitoring van websites, servers, netwerken en applicaties beheren. De websitefunctie laat je naast checken of de site up of down is ook checks uitvoeren om te kijken of gebruikers ook echt kunnen navigeren over je site.

  De server functie biedt je een overzicht van de prestaties van je server en of deze ergens moeite mee heeft.

  Met de networking functie kan je inzicht krijgen hoe de latency van over de wereld eruitziet.

  De applicatie functie geeft je inzicht in je applicaties zodat je problemen binnen je applicatie kan constateren en oplossen. Talen die ondersteund worden zijn onder andere .NET, Java en PHP.

  Ook is er een optie om een OnSight agent te deployen. Deze agent staat op eigen locatie waardoor de veiligheid van je bedrijf niet in gevaar komt. Het laat je gemakkelijk symptomen van een incident aanvullen met metingen van applicaties.

- Alerting
  Deze functie wordt gebruikt om mensen op de hoogte te stellen van problemen. Je kan hiermee tijdlijnen creëren met escalatie punten. Deze alert timelines kan je binden aan bepaalde groepen, specifieke incidenten en *"any other dimension you can think of"*.

  Je kan de alertfunctie laten integreren met tools zoals HipChat en OpsGenie

  Je kan een on-call schedule maken zodat de juiste persoon gecontacteerd wordt. Wanneer iemand ziek of op vakantie is kan je gemakkelijk een andere persoon als invaller aanwijzen.

  Ook Panopta biedt een app aan zodat je waar je ook bent op de hoogte kan blijven van de laatste alerts.

- Reporting
  Deze functie wordt gebruikt om de statistieken te weergeven. Je kan met deze functie zelf dashboards instellen waar alle informatie in een oogopslag op zichtbaar is.

  Er is ook een optie om de status van een bepaalde pagina publiek te maken met een status page builder. Iedereen met een URL kan dan de status van de server zien.

Verder kan je rapporten automatisch genereren. Je kan ook rapporten met een interval automatisch laten mailen. Ook als er een incident plaatsvind dan kan je hier meteen een gedetailleerd rapport voor opvragen.

Kosten
Panopta biedt vier standaardpakketten aan. Standard, Performance, Corporate en Enterprise.

Het Standard pakket is niet interessant voor TamTam. De andere drie komen wel in aanmerking

Het Performance pakket kost $99,- per maand of $999,- per jaar. In dit pakket zitten de volgende functionaliteiten: 50 Network checks, 25 Server agents, 1 jaar data retentie, 10 status pagina's, 2 OnSight Agents, SNMP Polling en On-Call schedules.

Het Corporate pakket kost $249,- per maand of $2499,- per jaar. In dit pakket zitten de volgende functionaliteiten: 150 Network checks, 75 Server agents, 2 jaar data retentie, ongelimiteerde status pagina's, 4 OnSight Agents, SNMP Polling, On-Call schedules, Customized monitoring en consultatie sessies.

Het Enterprise pakket kost $599,- per maand of $5999,- per jaar. In dit pakket zitten de volgende functionaliteiten: 450 Network checks, 250 Server agents, 4 jaar data retentie, ongelimiteerde status pagina's, 12 OnSight Agents, SNMP Polling en On-Call schedules Customized monitoring, consultatie sessies en VIP Support.

(Panopta, sd)

4. Monitoring bij grote bedrijven

In dit hoofdstuk wordt gekeken naar de aanpak van andere bedrijven op het gebied van monitoring. Er zal vooral worden gekeken naar welke tools deze bedrijven gebruiken en of deze overeenkomen met de aanwezige en nieuwe tools van TamTam. Het doel hiervan is om een beeld te krijgen in de algemene handelswijze van bedrijven met meer monitoringservaring dan TamTam. Als bijvoorbeeld wordt geconstateerd dat geen enkel groot bedrijf op de wijze die TamTam nu hanteert handeld kan TamTam zich afvragen of zij wel op de goede weg is.

In dit document zal een matrix opgesteld worden van de tooling die TamTam nu gebruikt en misschien gaat gebruiken. Ook worden tools die grote bedrijven wel gebruiken maar TamTam niet in dit document verder uitgewerkt. Verder zal er aan de hand van deze matrix een conclusie worden getrokken. Als een tool die TamTam als optie heeft niet terugkomt bij de grote bedrijven zal deze uit de matrix worden weggelaten. Dit zodat de matrix overzichtelijk blijft.

4.1 Matrix bekende tools

In dit hoofdstuk is de matrix te vinden. Op de horizontale as staan de bedrijven. Op de verticale as staan de tools die bij TamTam in aanmerking komen.

Tooling TamTam	Atlassian	Spotify	Dropbox	Stack overflow	Vine
Pingdom	X	X	X		X
Pagerduty	X			X	X
Nagios			X		X
Cloudwatch	X				
New Relic	X	X			

Achteraf gezien blijkt deze tabel een stuk kleiner is dan verwacht. Van de tooling die aanwezig en geselecteerd zijn bij TamTam zien we alleen New Relic, Pingdom, Cloudwatch, Pagerduty en Nagios terug. New Relic, Cloudwatch, Nagios en Pingdom zijn al in gebruik bij TamTam en Pagerduty is een optie op het gebied van alerting.

De vormgeving van de monitoring bij TamTam komt overeen met de grotere bedrijven. Zowel Atlassian als Vine maken gebruik van een monitoringtool die ze laten integreren met een alerting tool (Pingdom met Pagerduty). TamTam doet dit in de huidige situatie ook (Pingdom en UptimeRobot met OpsGenie). Verder zien we ook nog terugkomen dat Nagios gebruikt wordt. Ook deze tool wordt al door TamTam gebruikt.

Aan de hand hiervan kan de conclusie worden getrokken dat TamTam goed op weg is. De algemene inrichting is goed, maar er moet een duidelijke standaard in het bedrijf gehanteerd gaan worden.

(Stackshare, sd)

4.2 Matrix onbekende tools

Een paar andere tools kwamen ook vaker naar voren. In de matrix hieronder is te zien welke tooling veel wordt gebruikt bij grotere bedrijven. Deze tools zijn nog *niet* door TamTam bekeken voor de nieuwe monitoringsoplossing.

Tools	Atlassian	Spotify	Dropbox	Stack overflow	Vine
Datadog	X	X			X
Sentry			X		X
Airbrake				X	
Bosun				X	

Het valt op dat de tool Datadog door drie grote bedrijven in gebruik is. Ook is te zien dat tools als Sentry door twee bedrijven wordt gebruikt. Verder is te zien dat Stack overflow Airbrake en Bosun gebruikt. Airbrake is een grote tool die ook nog door andere bedrijven wordt gebruikt. Bosun is echter een tool die naast Stack overflow maar door één ander bedrijf wordt gebruikt. Naar deze tool zal dus niet gekeken worden in de rest van dit document. De andere drie tools zullen verder uitgewerkt worden. Voor deze tools wordt er gekeken naar de functies die de tool aanbiedt, de prijs van de tool en of de tool voor TamTam interessant is.

(Stackshare, sd)

4.3 Datadog

Datadog biedt een combinatie van monitoren op infrastructuur en functioneel niveau aan. Ze integreren met Cloud hosts zoals Azure en Amazon. Zelf geven zij aan dat ze volledig inzicht bieden in moderne applicaties. Je moet datadog wel direct op een server installeren. Datadog wordt gebruikt door Atlassian, Spotify en Vine.

Hieronder staan alle functionaliteiten die Datadog aanbiedt:

- **Full observability for modern applications**
  Deze functie zorgt ervoor dat je een inzicht krijgt in alles waar je applicatie mee te maken krijgt. Je kan de prestaties van je applicaties monitoren, inzicht krijgen in de latency van je applicatie en inzicht verkrijgen in de cloud/on-prem/hybride applicaties op één plek. Verder heb je de optie om grafieken in te zien van je applicaties. Hiermee kan je bovengenoemde latency inkijken maar ook het aantal hits en errors per seconde.

- **Build real-time interactive dashboards**
  Met deze functie kan je zelf een dashboard maken met alle informatie erop die je wil zien. Er zijn extreem veel mogelijkheden alles wat datadog kan monitoren kan je op het dashboard plaatsen. Dit kan erg handig zijn als er gebruikers zijn die alleen geïnteresseerd zijn in bijvoorbeeld de latency.

- **Share what you saw, write what you did.**
  Met deze functie kunnen gebruikers als zij bij een grafiek een piek zien, bij deze grafiekpiek een notitie maken. Je kan dan teamleden het omrande stukje in de grafiek tonen en hier een opmerking bij plaatsen.

- **Get alerted on critical issues**
  Deze functie laat je alerts instellen voor de volgende gebeurtenissen:

  - Applicatie en systeem metingen
  - Up/down status checks
  - Network checks
  - Metingen van processen
  - Discrete events
  - Custom metingen via StatsD (tool van datadog)

  Deze alerting kan je zelf instellen. Je kan de prioriteit van het alert instelllen en na hoeveel minuten je op de hoogte gesteld wil worden. Ook kan je deze alerting laten integreren met tools zoals Pagerduty, OpsGenie en VictorOps.

- **API**
  Verder biedt Datadog een API aan waarmee je zelf uitbreidingen op datadog kan toevoegen.

Kosten
Voor TamTam zijn er twee pakketten interessant wanneer er wordt besloten om met Datadog te gaan werken.

Het eerste pakket heet Pro en kost per host $18 per maand of per host $180 per jaar. In dit pakket zitten de volgende functionaliteiten. Ongelimiteerde event en standaard metingen, Zelf te customizen dashboards, toegang tot integraties, hostmaps, Enterprise-niveau security, Ongelimiteerd aantal gebruikers, 15 maanden data retentie, ongelimiteerde alerts, 10 container monitoring per host, 100 custom metingen per host, Single sign on, piekdetectie en support op gebied van het forum, training videos, chat en email support.

Het tweede pakket heet Enterprise en kost per host $27 per maand of per host $276 per jaar. In dit pakket zitten de functionaliteiten uit de pro editie en daarnaast nog de volgende. 20 container monitoring per host, 200 custom metingen per host, Applicatie prestatie monitoring, afwijking detectie, samenvattingsrapportages en extra ondersteuning met een Dedicated succes manager en Onboarding support.

4.3.1 Interessant voor TamTam?

De tool biedt veel eisen aan die zijn gesteld aan de nieuwe monitoringsituatie. Er wordt echter niet voldaan aan een eis die wel als extra belangrijk werd aangegeven, namelijk de eis van simpelheid in gebruik. De tool is vrij groot en niet simpel om even snel op te zetten. Zelfs gebruikers met technische kennis zullen even moeten uitzoeken hoe alles werkt.

Verder zou een eventuele overstap naar deze tool zoveel tijd in beslag nemen dat overstappen niet interessant meer is. Het is dan nuttiger om die tijd te besteden aan het beter inrichten van huidige tooling.

Om deze reden concludeer ik dat de tool niet interessant is. Als de eis van gebruikersgemak minder belangrijk was geweest was deze tool geschikt geweest.

(Datadog, sd)

4.4 Sentry

Sentry is een simpele tool die kijkt naar fouten in de code. Sentry integreert onder andere met PHP en .NET. Sentry wordt gebruikt door Dropbox en Vine. Sentry is open source software en is te vinden op GitHub.

Hieronder staan alle functionaliteiten die Sentry aanbiedt:

- **Track releases**
  Sentry kan een verband trekken tussen een fouten in de code en een bepaalde release. Dit geeft gebruikers inzicht zodat zij precies kunnen zien wanneer een fout ontstaan is. Wanneer het probleem is opgelost kun je dubbele fouten onderdrukken en notificaties ontvangen wanneer er in de toekomst weer problemen mee komen.

- **Understand your environment**
  Elke fout die door Sentry wordt geconstateerd bevat duidelijke informatie over de omgeving (OS, prod server, release) zodat je weet onder welke omstandigheden deze fout zich voordeed.

- **Consider the users**
  In sentry kan er ook gefilterd worden op gebruikers. Met deze functie kan je precies inzien tegen wat voor fouten een specifieke gebruiker is aangelopen. Je kan ook zien hoe vaak deze fout is voorgekomen.

- **Add custom context**
  Deze functie laat je tags toevoegen en alerts ontvangen wanneer door de gebruiker gestelde eisen worden voldaan.

- **Integraties**
  Sentry integreert ook met tools zoals JIRA en OpsGenie.

Kosten
Op de site van Sentry worden drie pakketten aangeboden.

Het hobbyisten pakket is gratis maar biedt maar één gebruiker aan. Verder kan je maximaal 10.000 events per maand gebruiken.

Voor Proffesional betaal je $12,- per maand. Je krijgt hiervoor 50.000 events per maand, ongelimiteerd aantal gebruikers, kosten a.d.h.v. gebruik en Email support.

Ook is er een Enterprise pakket. Voor dit pakket moet je contact opnemen en worden aparte afspraken gemaakt.

4.4.1 Interessant voor TamTam

De tool werkt eigenlijk alleen op het gebied van code monitoring. Dit is enigszins interessant voor TamTam, maar in de huidige situatie wordt dit al gedaan door New Relic.

Als je Sentry vergelijkt met New Relic dan is Sentry een vrij “kale” tool. New Relic biedt veel meer inzichten in andere processen bijvoorbeeld wat er in je databases gebeurt. Om deze reden is Sentry naar mijn mening niet interessant voor TamTam. Verder is er ook al besloten dat dit onderdeel van monitoring buiten de scope valt voor de opdracht.

(Sentry, sd)

4.5 Airbrake

Airbrake is een tool die exceptions in je applicatie ontdekt. Vervolgens biedt Airbrake inzicht in de meldingen zodat gebruikers bugs snel kunnen oplossen. Airbrake ondersteund onder andere talen als .NET en PHP.

Hieronder staan alle functionaliteiten die Airbrake aanbiedt:

- **Intelligent error grouping**
  Met deze functie kan een gebruiker fouten op een gemakkelijke manier groeperen. De gebruiker kan de fouten sorteren zodat fouten die op elkaar lijken worden weergeven. Ook kunnen er trends over een specifieke foutmelding of voor een geheel project worden weergegeven. Verder kan een gebruiker ook nog custom groepen aanmaken zodat een groep specifieke fouten te zien krijgt.

- **Error details & aggregated error data**
  Met deze functie kan je snel het juiste bestand, methode en regel vinden waar het probleem zich voordoet. Je kan de gebruikers die bij het probleem betrokken zijn identificeren. Ook kan je inzien in welke browsers het probleem zich voordoet. Vervolgens kan een gebruiker zich ook nog extra verdiepen door middel van backtracing en extra losse informatie.

- **Deploy tracking, search and filter**
  Deze functie geeft inzicht in onder andere de impact van een deploy. Je kan hiervan zien of het aantal fouten gestegen of gedaald is. Met de search en filter functie kan een gebruiker vervolgens specifieke fouten vinden waar hij/zij in is geïnteresseerd.

- **Airbrake app**
  Airbrake kan je ook op iOS en Android installeren. Op deze manier kan je ook als je niet achter je pc zit op de hoogte gesteld worden van nieuwe fouten.

- **Integraties**
  Sentry integreert ook met tools zoals JIRA en OpsGenie.

Kosten
Airbrake biedt eigenlijk maar één pakket aan wat interessant is voor TamTam. De andere pakketten ondersteunen niet genoeg applicaties.

Het Growth pakket kost $249 per maand. In dit pakket zitten de volgende functies. Tot 200 foutmeldingen per minuut, ongelimiteerd aantal applicaties, 25 teamleden, 180 dagen data retentie en priority support.

4.5.1 Interessant voor TamTam?

Airbrake heeft veel overeenkomsten met Sentry. Eigenlijk doen beide tools precies hetzelfde. Om deze reden heb ik dan ook dezelfde mening als voor Sentry. De tool is niet interessant genoeg voor TamTam als je deze vergelijkt met New Relic. Verder is er ook al besloten dat dit onderdeel van monitoring buiten de scope valt voor de opdracht.

(Airbrake, sd)

5. Toolselectie nieuwe situatie

In dit hoofdstuk wordt een beslissing genomen over welke tools er in de nieuwe monitoringsituatie door TamTam gebruikt zullen gaan worden. Deze selectie zal gemaakt worden uit de al aanwezige tools bij TamTam en de tools die in de inventarisatie naar voren zijn gekomen. Deze tools zullen allemaal uitgezet worden tegenover een lijst met eisen. Deze lijst wordt opgesteld met behulp van het requirementsdocument.

De volgende tools komen in aanmerking op gebied van monitoring:

- Nagios Core
- Pingdom
- UptimeRobot
- Uptrends
- Statuscake
- Dynatrace
- BigPanda
- Xmatters
- Panopta
- Cloudwatch
- Azure portal

De volgende tools komen in aanmerking op gebied van alerting:

- OpsGenie
- VictorOps
- Pagerduty
- Panopta
- Xmatters

5.1 Functionaliteiten

Deze eisen zijn aan de hand van de requirements opgesteld. Dikgedrukte eisen zijn eisen die extra belangrijk waren in de ogen van de stakeholders.

Op gebied van monitoring:

- Systeem moet overzichten aanleveren met informatie zoals up-time en reactietijd.
- Tool moet op infrastructuur kunnen monitoren.
- Tool moet op functioneel niveau kunnen monitoren.
- Er moet een dashboard zijn waar je van alle sites ziet of deze up of down zijn.
- Tool moet uptime checks kunnen toevoegen.
- Er moet getest kunnen worden of bepaalde functionaliteiten werken.
- Tool moet meldingen kunnen geven met SMS
- Tool moet meldingen kunnen geven met email
- Systeem moet schaalbaar zijn.
- Je moet een server kunnen muten als deze voor onderhoud down gaat.
- Er moet een uptime report gegenereerd kunnen worden.
- **Het systeem moet simpel in omgang zijn.**

Het is niet de bedoeling van TamTam om veel tijd aan de monitoring te gaan besteden. Daarom is de eis dat het systeem simpel in omgang moet zijn als extra belangrijk gemarkeerd.

Op gebied van alerting:

- Tool moet contact kunnen opnemen met een specifieke persoon.
- Tool moet een mail sturen zodra een site down is.
- Tool moet met het opnemen van contact een duidelijke boodschap meesturen zodat duidelijk is wat er gedaan is en welke foutmelding zich voordoet.
- Tool moet rekening houden met de prioriteit.
- Een issue moet kunnen escaleren.
- DevOps, Projectteams en OS moeten tegelijk op de hoogte worden gesteld van een probleem.
- Je moet een server kunnen muten als deze voor onderhoud down gaat.
- Tool moet naast mail op andere manieren contact kunnen leggen.
- Je moet groepen kunnen managen in de tool.

Voorkeuren (could have’s) op het gebied van alerting:

- Tool moet contact blijven zoeken totdat je op het probleem reageert.
- Een on call functie.

Deze eisen worden allemaal in een tabel tegenover de tools gezet. Vervolgens wordt met een X aangegeven of de tool deze functionaliteit aanbiedt.

5.2 Monitoring tool

Omdat deze selectie vrij groot was werd de tabel erg onoverzichtelijk. Daarom is besloten om deze tabel in excel uit te werken. Zie het bestand *"Monitoring tools tegenover eisen"*. De tabel is ook hieronder in te zien.

Eisen	Nagios	Pingdom	UptimeRobot	Uptrend	Statuscake	Dynatrace	BigPanc	Panopt	Azure	AWS
Systeem moet overzichten aanleveren met informatie zoals uptime en reactietijd	X	X	X	X	X	X		X	X	X
Tool moet op infrastructuur niveau kunnen monitoren	X					X		X	X	X
Tool moet op functioneel niveau kunnen monitoren	X	X	X	X	X	X		X	X	X
Er moet een dashboard zijn waar van alle sites zichtbaar is of deze up of down zijn.	X	X	X	X	X			X		
Tool moet uptime checks kunnen toevoegen	X	X	X	X	X	X		X	X	X
Systeem moet schaalbaar zijn	X	X	X	X	X	X	X	X		
Je moet een server kunnen muten als deze voor onderhoud down gaat	X	X	X		X	X		X		
Er moet een uptime report gegenereerd kunnen worden	X	X		X	X	X	X	X		
Het systeem moet simpel in omgang zijn		X		X	X				X	X
Tool moet meldingen kunnen geven met email	X	X	X	X	X	X	X	X	X	X
Tool moet meldingen kunnen geven met sms		X	X	X	X		X	X		
Er moet getest kunnen worden of bepaalde functionaliteiten werken	x	X		X	X	X		X	X	X
	x met custom plugins									

Uit deze tabel blijkt dat de tools over het algemeen aan de meeste eisen voldoen. Er komen echter wel enkele tools naar voren die net een stapje voor hebben op andere, maar er zijn ook tools die bepaalde functionaliteiten missen en om die reden afvallen.

5.2.1 Geschikte tools

De volgende tools hebben de eerste selectiefase doorstaan:

- Nagios Core
- Pingdom
- Statuscake
- Panopta
- AWS
- Azure

Nagios Core
Nagios voldoet samen met twee andere tools (waarvan er een is afgevallen) aan de eis monitoren van infrastructuur. De tool mist het punt van meldingen geven met sms, maar doet dit wel via email. Nagios zou dus wel kunnen integreren met een tool zoals OpsGenie die vervolgens wel alerts van Nagios als sms-berichten kan versturen.

Ook mist Nagios de gebruikersvriendelijkheid in de vorm van simpel in omgang zijn. Dit is een nadeel wat ook opgevangen kan worden door een andere tool, er ontstaat dan echter wel een situatie waar meerdere tools in gebruik zijn en een eis was dat dit zou afnemen.

Verder biedt Nagios via custom plugins de optie om te testen of bepaalde functionaliteiten van een website werken. Deze eis valt buiten de scope, maar het is wel interessant om deze optie beschikbaar te hebben om eventueel later te implementeren.

Nagios is al in gebruik bij TamTam waardoor er in de orginisatie al kennis aanwezig is over deze tool. Daarnaast is Nagios Core open source software en dus volledig gratis in gebruik. De andere tools bieden soms wel een gratis versie aan, maar dan missen er heel veel van de nu aangevinkte functionaliteiten. De tool moet om deze reden echter wel zelf onderhouden worden.

Panopta

Panopta is een erg uitgebreide tool. Net zoals Nagios voldoet ook Panopta aan de eis infrastructuur monitoren. Panopta biedt naast de monitoring functie ook nog een alerting functie aan. De tool bied dus eigenlijk beide functies aan waarnaar wordt gezocht.

Het enige nadeel is dat de tool doordat deze zo uitgebreid is niet simpel in gebruik is. Ook is er nog geen ervaring aanwezig met deze tool en het overzetten van checks en het opzetten van de monitoring zal daardoor meer tijd kosten. In het algemeen zal er in Panopta meer tijd gaan zitten dan in een tool zoals Pingdom.

Ook is Panopta een vrij dure tool. Een nuttige versie (150 network checks en 75 server agents) kost al $2499,- per jaar. Je hebt dan wel een tool die eigenlijk alle gezochte functionaliteiten heeft.

Pingdom

Pingdom voldoet aan alle eisen op één na, het monitoren van infrastructuur. Het is echter wel essentieel voor TamTam dat dit ook mogelijk is. Wanneer Pingdom dus als tool wordt geselecteerd zal de monitoring van de infrastructuur op een andere manier afgevangen moeten worden.

Verder voldoet Pingdom aan alle gestelde eisen. Daarnaast is ook Pingdom nu al in gebruik binnen TamTam wat ook nog een extra voordeel is. Pingdom kost TamTam nu €443.40 per jaar.

Statuscake

Statuscake heeft dezelfde eigenschappen als Pingdom. Ook statuscake kan niet de infrastructuur monitoren en dit zal dus op een andere manier afgevangen moeten worden.

Ook Statuscake voldoet verder aan de gestelde eisen. Het verschil tussen Pingdom en Statuscake zit in de prijs. Statuscake kost €290.28 per jaar en is dus in totaal €153.12 goedkoper dan Pingdom.

AWS

AWS kan zowel de uptime als de infrastructuur monitoren. Het monitoren van de infrastructuur is een service die onderdeels is van Cloudwatch. De standaard versie van Cloudwatch biedt deze functionaliteiten al aan. Je hoeft dan verder ook geen extra software op je server te installeren om deze te gaan monitoren.

Verder kun je AWS ook de up-time van je website laten testen. Dit wordt net zoals in de andere geselecteerde tools gedaan door een HTTP request te sturen en te kijken of de webpagina met een goede code reageert.

Azure

Voor Azure geldt eigenlijk hetzelfde als voor AWS. Ook met deze tool kan je de infrastructuur en uptime monitoren. Net zoals bij AWS zijn deze functies aanwezig in de standaard versie van Azure's bijgeleverde monitoring waardoor er naast de kosten van het afnemen van de servers, geen extra kosten worden gemaakt. Ook op Azure hoef je geen extra software te installeren voor het monitoren van de infrastructuur.

5.2.2 Conclusie & opties

Op het gebied van monitoring zijn er dus enkele oplossingen die TamTam kan gaan gebruiken. Ik stel voor om de keuze te maken uit één van deze drie voorstellen.

Optie 1
Een combinatie van Statuscake en Nagios. Dit is de eerste optie omdat de functionaliteiten van Pingdom en Statuscake hetzelfde zijn, maar Statuscake is goedkoper. Deze combinatie biedt een goedkope maar wel een functionele monitoringsomgeving aan.

In de huidige situatie wordt Nagios nog gemonitord door Sentia. Sentia brengt hier uiteraard ook kosten voor in rekening. De bedoeling is dat Nagios gemonitord kan gaan worden door TamTam zelf waardoor het echt gratis wordt.

Wanneer voor deze oplossing wordt gekozen zal TamTam jaarlijks aan kosten €1016,28 kwijt zijn.

Optie 2
Een combinatie van Pingdom en Nagios. Voordeel hiervan is dat deze tools nu al beide in gebruik zijn. Het nadeel hiervan is dat Pingdom duurder is in gebruik.

In de huidige situatie wordt Nagios nog gemonitord door Sentia. Sentia brengt hier uiteraard ook kosten voor in rekening. De bedoeling is dat Nagios gemonitord kan gaan worden door TamTam zelf waardoor het echt gratis wordt.

Wanneer voor deze oplossing wordt gekozen zal TamTam jaarlijks aan kosten €443.40 kwijt zijn. Dit is echter wel met de versie van Pingdom die nu wordt gebruikt en vrij basic is. Er is namelijk ook belang naar uitgebreidere testfuncties en hier zijn er nu maar drie van inbegrepen in het huidige pakket. Waneer je bijvoorbeeld 25 van deze checks wil kunnen uitvoeren kost dit al $2388,- per jaar.

Waarschijnlijk heeft TamTam echter nog meer van deze checks nodig. Hier wordt een speciaal enterprise plan van $10.956,- voor aangeboden. In dit pakket kan je 100 advanced checks laten uitvoeren. In dit opzicht is Pingdom dus een stuk duurder dan Statuscake. Daarom is Statuscake ook optie een geworden.

Optie 3
AWS en Azure native monitoring. Deze optie kost geen extra geld omdat de benodigde functies al meegeleverd worden in de standaard monitoring die AWS en Azure aanbieden. Er worden dan alleen nog maar kosten gemaakt aan de afgenomen hosting bij AWS en Azure. Deze kosten moeten echter toch wel gemaakt worden omdat er anders geen omgeving is om de site in te hosten.

Optie 4

Panopta is een tool die van alle markten thuis is. Het voordeel van deze tool is dat er maar één tool in gebruik is voor de gehele monitoring en alerting optie. Wordt er voor deze oplossing gekozen dan zal TamTam jaarlijks aan kosten $2499,- kwijt zijn. Er moet dat wel rekening worden gehouden met het feit dat er geen extra kosten bijkomen voor een alerting tool.

5.3 Alerting tool

Ook de vergelijkingstabel van de alerting tools is uitgewerkt met behulp van excel. Zie het bestand *"Monitoring tools tegenover eisen"*. De tabel is ook hieronder in te zien.

Eisen	OpsGenie	VictorOps	Pagerduty	Panopta	Xmatters
Tool moet contact kunnen opnemen met specifieke personen	X	X	X	X	X
Tool moet een mail sturen zodra een site down gaat	X	X	X	X	X
Tool moet met het opnemen van contact een duidelijk boodschap meesturen zodat de gebruiker op de hoogte is wat er is gedaan en welke foutmelding zich voortdoet	X	X	X	X	
Tool moet rekening houden met de prioriteit	X	X	X	X	
Een issue moet kunnen escaleren	X	X	X	X	
DevOps, Projectteams en OS moeten tegelijk op de hoogte worden gesteld van een probleem	X	X	X	X	X
Je moet een server kunnen muten als deze voor onderhoud down gaat	X	x		X	
Tool moet naast mail ook op andere manieren contact kunnen leggen	X	X	X	X	X
Je moet groepen kunnen managen in de tool	X	X	x	X	X
Tool moet contact blijven zoeken totdat de gebruiker op het probleem reageert	X	X		X	
Tool moet een on call functie hebben	X	X	X	X	X
		Kleine x zit alleen in enterprise versie	Kleine x zit alleen in Standard versie		

Uit deze tabel is te zien dat alle tools op Pagerduty en Xmatters na in aanmerking komen om in gebruik te worden genomen.

5.3.1 Geschikte tools

De volgende tools hebben de eerste selectiefase doorstaan:

- OpsGenie
- VictorOps
- Panopta

OpsGenie
OpsGenie is de alerting tool die nu in gebruik is binnen TamTam. Deze tool is ook al opgezet zodat deze gebruikt kan worden binnen TamTam. Ook is er voor deze tool een integratie geschreven die werkt met de TamTam support tool. Op dit moment zijn er zes gebruikers aanwezig die allemaal $91 per maand kosten. Wordt er voor OpsGenie gekozen dat zal TamTam jaarlijks aan kosten $1092,- kwijt zijn.

VictorOps
VictorOps is een tool die eigenlijk identiek is aan OpsGenie. Het enige probleem met deze tool is dat slechts één eis alleen in de duurdere versie van de nieuwe tool zit namelijk het muten van een server. De versie met het muten van een server kost $588,- per jaar per gebruiker. De versie zonder kost $348,- per jaar per gebruiker. Met zes gebruikers wat nu het geval is zou VictorOps $3528,- per jaar of $2088,- per jaar kosten.

Panopta
Deze tool is al eerder behandeld in het kopje monitoring. Zoals te zien is voldoet Panopta ook op het gebied van monitoring aan vrijwel alle eisen. Wordt er voor deze oplossing gekozen dan zal TamTam jaarlijks aan kosten $2499,- kwijt zijn. Er moet dat wel rekening worden gehouden met het feit dat er geen extra kosten bijkomen voor een monitoring tool.

5.3.2 Conclusie & opties

De geselecteerde tools voldoen allemaal aan de gestelde eisen alleen in sommige gevallen voldoen de tools alleen aan alle eisen met hun duurste versie. Ik stel daarom voor om de keuze te maken uit één van deze twee voorstellen.

Optie 1
Blijf gebruik maken van OpsGenie. Deze tool is in vergelijking met de andere tooling vrij goedkoop en is al geïntegreerd in het bedrijf. Ook is er een speciale integratie geschreven specifiek voor de TamTam support desk. Wanneer er voor OpsGenie wordt gekozen en er blijven zes gebruikers in gebruik dan betekend dit dat er bij de monitoringsvoorstellen voor optie 1 of 2 is gekozen. Als er voor voorstel 1 (Statuscake) is gekozen dan komt het totaalbedrag per jaar op €2108.28. Is er gekozen voor voorstel 2 (Pingdom) dan komt het totaal bedrag per jaar op €1471.70 uit.

Er moet hier wel rekening worden gehouden met de pakketsoort die bij de berekening gebruikt wordt. Van Statuscake wordt het duurste pakket behandeld met ongelimiteerde checks. Van Pingdom wordt het huidige pakket gebruikt waar niet 50 checks in zitten en 3 uitgebreidere checks.

Mochten er nieuwe gebruikers toegevoegd gaan worden aan OpsGenie dan zal de prijs ook nog oplopen, omdat per nieuwe gebruiker maandelijks $15,- betaald moet worden.

Optie 2
Stap over naar Panopta. Nogmaals, Panopta biedt beide functionaliteiten aan. De overstap naar Panopta zal echter veel tijd kosten en het is niet de goedkoopste oplossing die er is. Wanneer er voor Panopta wordt gekozen zullen de jaarlijkse kosten €2353,22 (omgerekend uit dollars met koers €1,- is $1,06) bedragen.

6. Advies

Persoonlijk raad ik aan om voor de Alerting en Monitoring tool gebruik te maken van beide opties 1. Dit houdt in dat TamTam de Tools Nagios, Statuscake en OpsGenie gaat gebruiken. De tools UptimeRobot en Pingdom kunnen dan komen te vervallen.

Statuscake biedt dezelfde functionaliteiten aan als Pingdom, maar doet dit voor een betere prijs in een vriendelijke interface. Nagios is wordt nu al afgenomen door TamTam bij Sentia. Zij hosten deze monitoringtool in hun cloud en monitoren de servers (die TamTam bij hen afneemt) op niveau van infrastructuur voor TamTam.

OpsGenie is ook al in gebruik en hier is zelfs al een custom integratie voor aanwezig met de support tool van TamTam. Het abbonement wat nu wordt afgenomen bij OpsGenie is ook goedkoper dan de alternatieven. Wel zullen er meer gebruikers moeten worden toegevoegd en één abbonement hoger afgenomen moeten worden. Dit heeft als reden dat er anders een limiet zit op het aantal telefoontjes dat de tool kan maken. OpsGenie blijft dan echter wel evenduur en goedkoper dan de verschillende concurenten.

De totaalkosten hiervan komen uit op €1318.58 per jaar. Ook moet er dan alleen een overstap worden gemaakt van Pingdom en UptimeRobot naar Statuscake. Nagios en OpsGenie kunnen gewoon in gebruik blijven.

Er moet wel rekening gehouden worden dat als er meer gebruikers aan OpsGenie toegevoegd gaan worden de prijs van deze tool op zal gaan lopen. Dit zal €15,- per extra gebruiker gaan bedragen. Bij de andere twee tools zullen er geen extra kosten in rekening komen, tenzij er bij Statuscake extra SMS credits worden aangeschaft. Dit is echter niet nodig aangezien OpsGenie ook kan SMSen.

OpsGenie

Nagios®

7. Verwijzingen

Airbrake. (sd). *Airbrake*. Opgehaald van Airbrake: https://airbrake.io/

Amazon. (sd). *AWS Amazon*. Opgehaald van Amazon CloudWatch: https://aws.amazon.com/cloudwatch/?sc_channel=PS&sc_campaign=acquisition_NL&sc_publisher=google&sc_medium=cloudwatch_b&sc_content=cloudwatch_p&sc_detail=cloudwatch&sc_category=cloudwatch&sc_segment=161201487542&sc_matchtype=p&sc_country=NL&s_kwcid=AL!4422

BigPanda. (sd). *Solutions*. Opgehaald van BigPanda: https://bigpanda.io/solutions/

Datadog. (sd). *Product*. Opgehaald van Datadog: https://www.datadoghq.com/product/

Dynatrace. (sd). *Capabilities*. Opgehaald van Dynatrace: https://www.dynatrace.com/capabilities/

Microsoft. (sd). *How to: Monitor Apps in Azure App Service*. Opgehaald van Microsoft: https://docs.microsoft.com/nl-nl/azure/app-service-web/web-sites-monitor?toc=%2fazure%2fapp-service%2ftoc.json

Nagios. (sd). *Nagios*. Opgehaald van Nagios Core: https://www.nagios.com/products/nagios-core/#features

New Relic. (sd). *Features*. Opgehaald van New Relic: https://newrelic.com/application-monitoring/features

OpsGenie. (sd). *Features*. Opgehaald van OpsGenie: https://www.opsgenie.com/features

Pagerduty. (sd). *Features*. Opgehaald van Pagerduty: https://www.pagerduty.com/features/

Panopta. (sd). *Features*. Opgehaald van Panopta: https://www.panopta.com/features/

Pingdom. (sd). *Why Pingdom?* Opgehaald van Pingdom: https://www.pingdom.com/company/why-pingdom

Sentry. (sd). *Features*. Opgehaald van Sentry: https://sentry.io/features/context/

Stackshare. (sd). *Stackshare*. Opgehaald van Stackshare: https://stackshare.io/stacks#!

Statuscake. (sd). *Alternative to Pingdom*. Opgehaald van Statuscake: https://www.statuscake.com/alternative-to-pingdom/

UptimeRobot. (sd). *UptimeRobot*. Opgehaald van UptimeRobot: https://uptimerobot.com/

Uptrends. (sd). *Uptrends*. Opgehaald van Uptrends: https://www.uptrends.nl/

VictorOps. (sd). *VictorOps*. Opgehaald van VictorOps: https://victorops.com/alerts-collaboration-tool-demo/

Xmatters. (sd). *It-management*. Opgehaald van Xmatters: https://www.xmatters.com/products/it-management/

F Statuscake proof of concept

Statuscake proof of concept.

Monitoring

1 juni 2017

Contents.

1. Introduction 173

2. Proof of concept 174

2.1 Creating an uptime-test 175

2.2 Integrating Statuscake with OpsGenie 179

2.3 Creating a test to indicate the full loading time 181

2.4 Create a SSL check 184

3. Conclusion 185

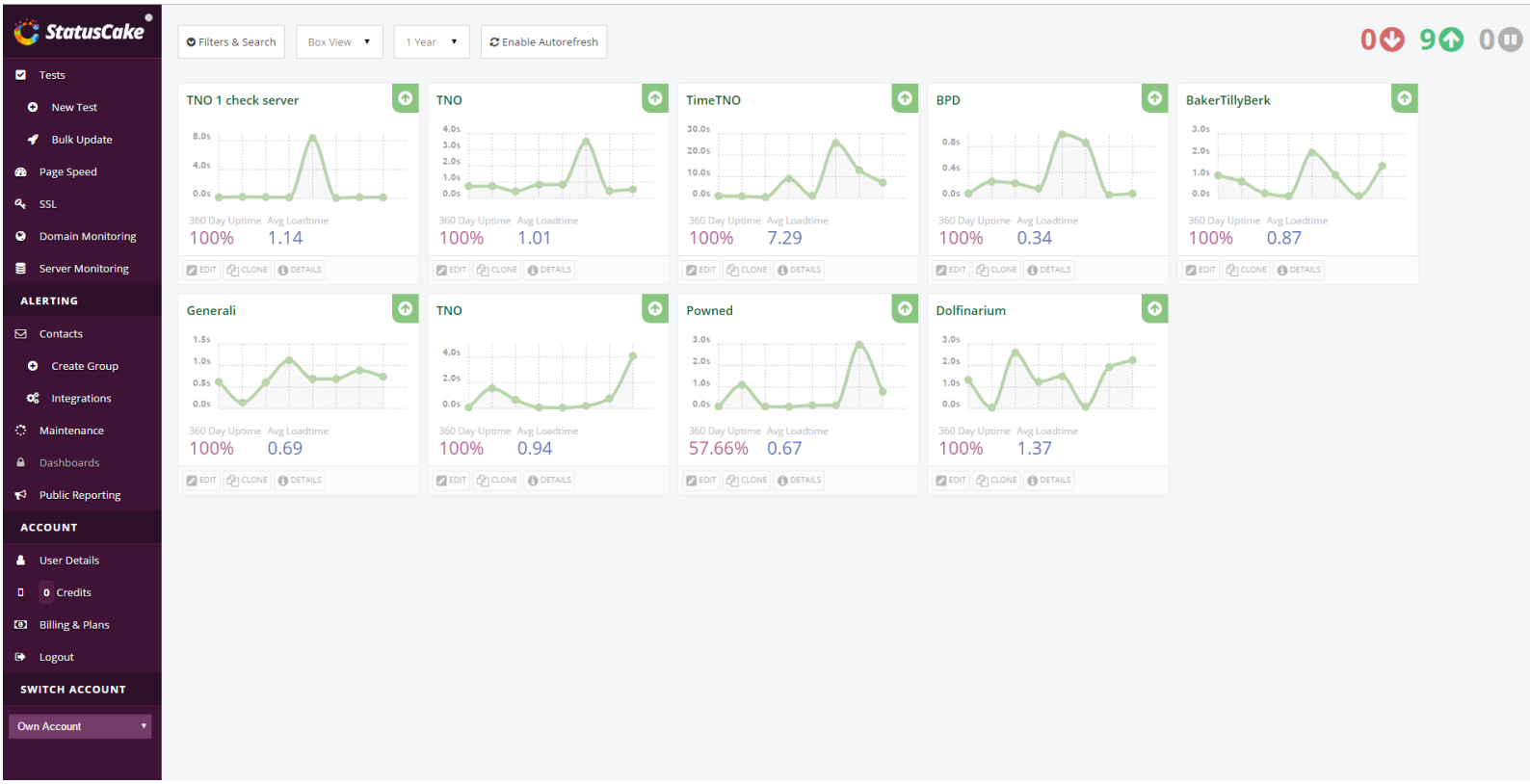
1. Introduction

In this document is information recorded about the Statuscake proof of concept. The target of this document is to explore how Statuscake works and if it meets TamTams requirements. In Statuscake the following actions will be attempted:

- Creating an uptime-test
- Integrating statuscake with OpsGenie
- Creating a test which indicates how long it takes the full site to load
- Creating a test when the SSL of a website expires

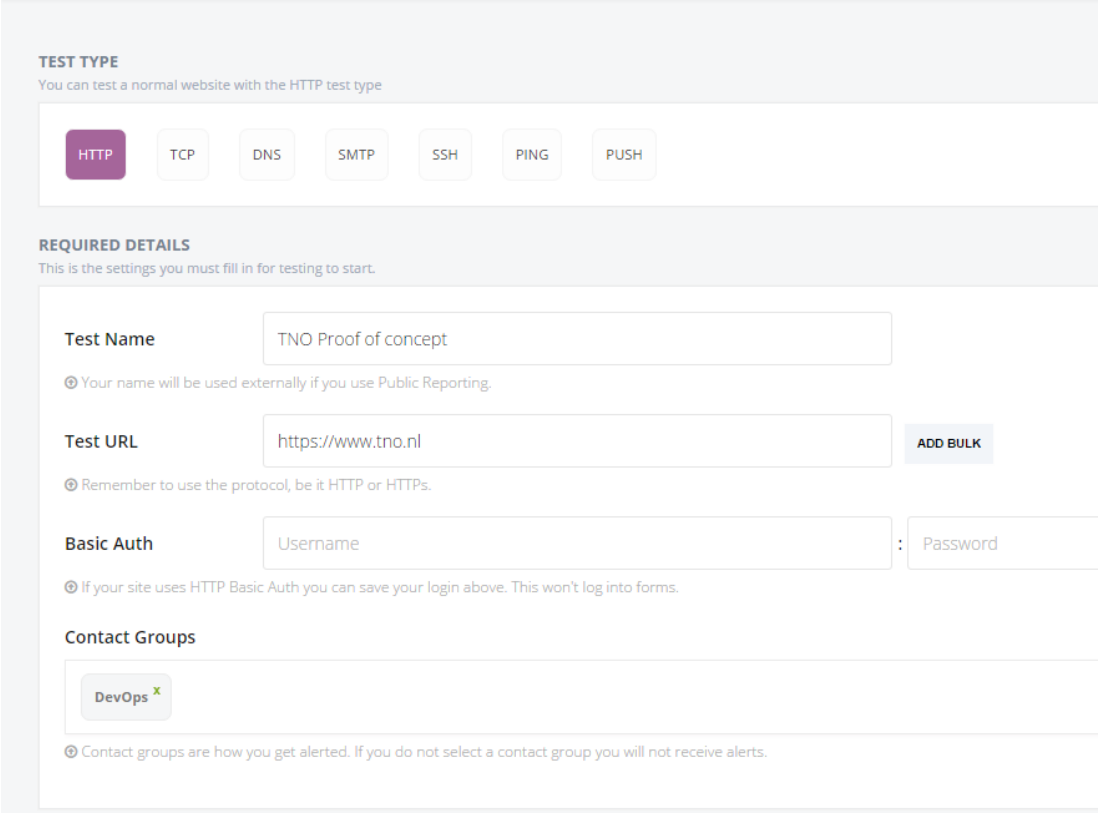
2. Proof of concept

The first thing you see when you log in on Statuscake is an overview of all your tests. From this screen you can navigate to the different test options, alerting options and the account info.



2.1 Creating an uptime-test

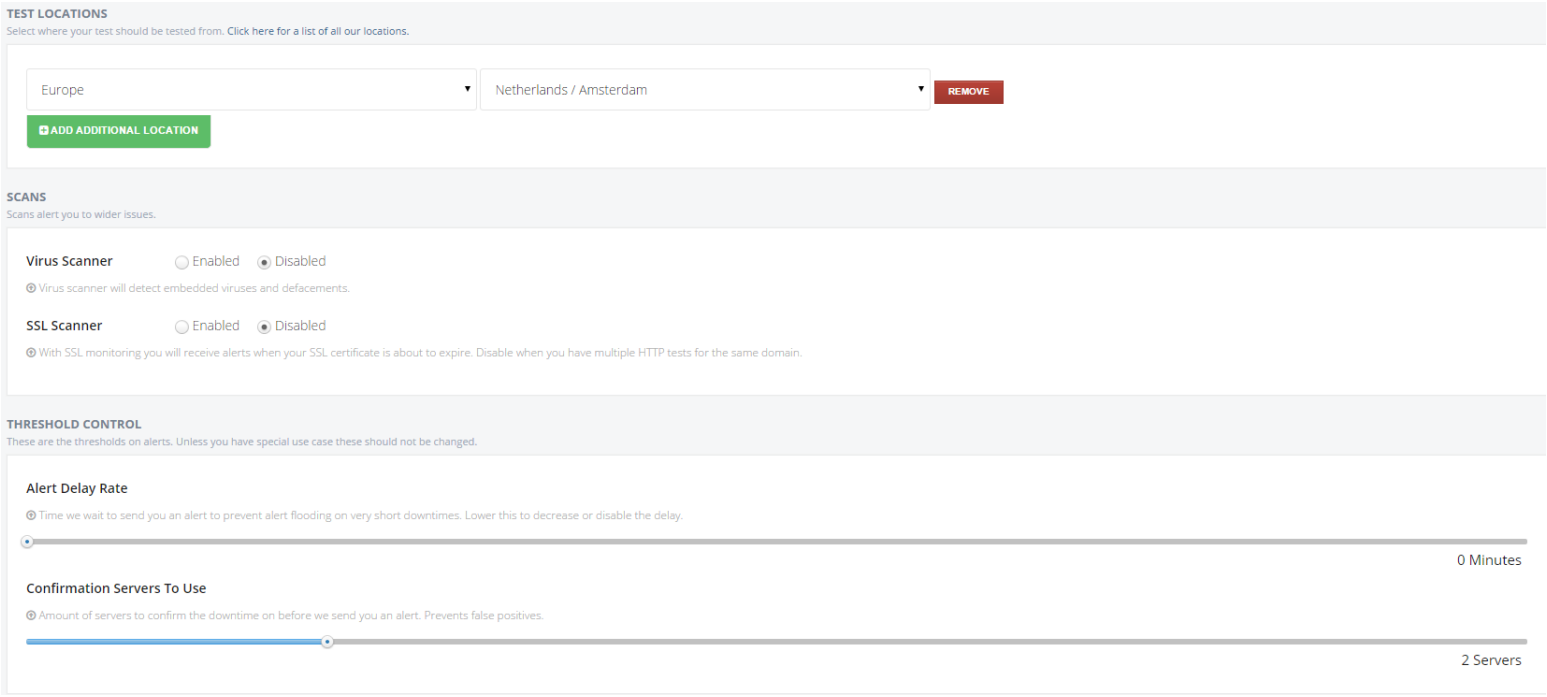
To create an uptime-test navigate to “New Test” in the sidebar. The first steps to create an uptime-test are selecting the test type (HTTP by default) and giving your test a name. Then you have to set the URL. The other two options are optional. You can let the tool login on your site if it uses HTTP basic auth. You can also indicate which team needs to be contacted if there is something wrong with the check.



In this proof of concept the TNO site was used. We used the full URL including the HTTPS. We didn’t fill in any login information since this isn’t needed for the site. The group to contact is set to DevOps. You can also select multiple groups to contact when the test fails.

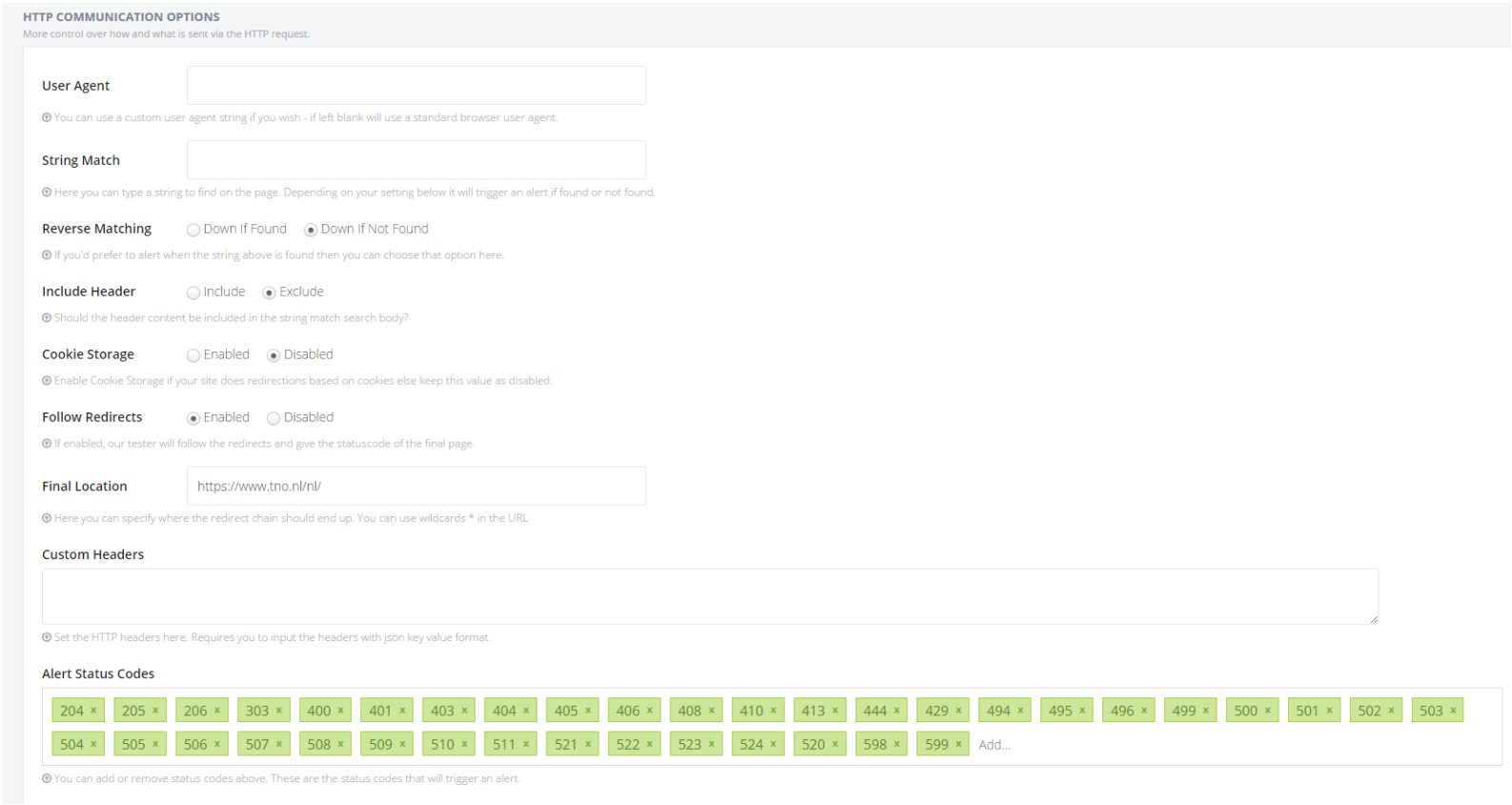
The next step is selecting the location to test from. You can choose no locations and the location will be random. You can also specify one or multiple locations from all over the world. After selecting a location you can indicate if you want Statuscake to run any scans. They offer a virus and SSL scanner. The virus scanner scans for viruses and the SSL scanner goes off if the SSL certificate is about to expire.

After that you can indicate the threshold. You can choose how long Statuscake waits until they send out an alert and how many confirmation servers they use.



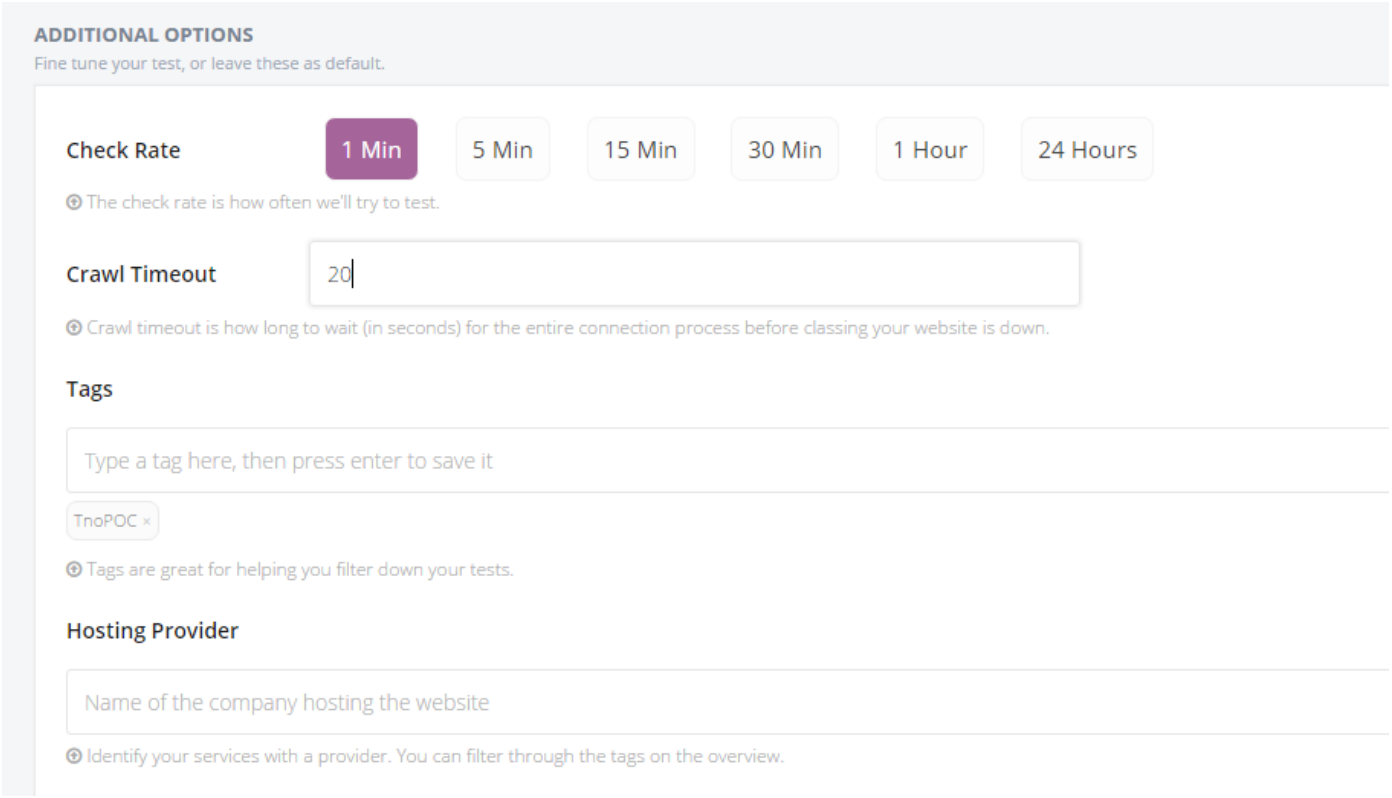
For this test the location that was selected was the Netherlands. We chose to not use any of the scan options. The chosen threshold was 0 minutes when the test fails and 2 other servers which also check if the site is realy down.

After that you have quite a big range of options available. You can indicate a user agent if you don't want to use a standard browser. You can check if the a certain string is available on the site. You can set redirection information. After that you can also choose to set custom headers and choose on which status codes the test should alert.



In this situation we chose to not use a specific user agent or string match. We did however enable the follow redirects option since https://www.tno.nl redirects to https://www.tno.nl/nl/. This is also indicated at the final location spot. There were no custom headers used and the alert status codes are the default options delivered by Statuscake.

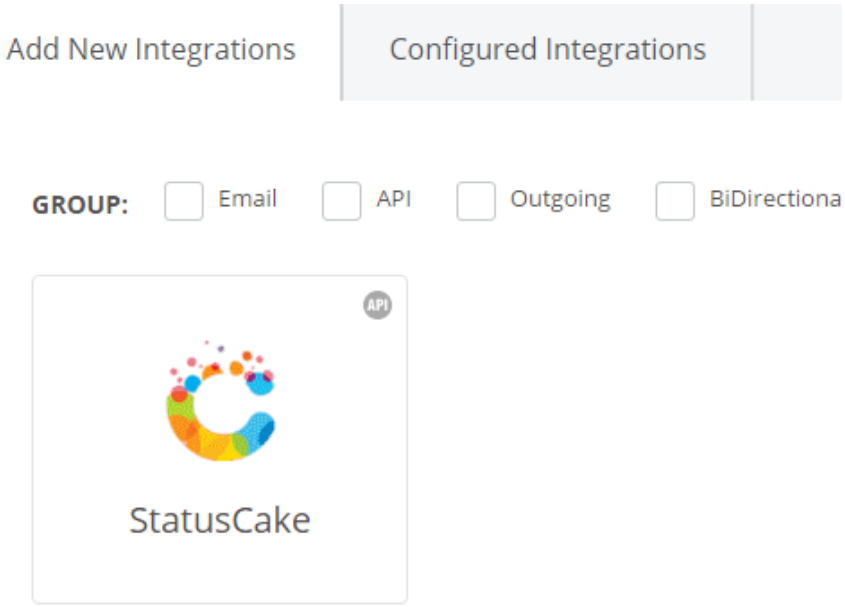
Last but not least you can choose the rate at which statuscake will check your page, how long statuscake should wait before classifying your site as down, tags, and you can indicate the hosting provider your using.



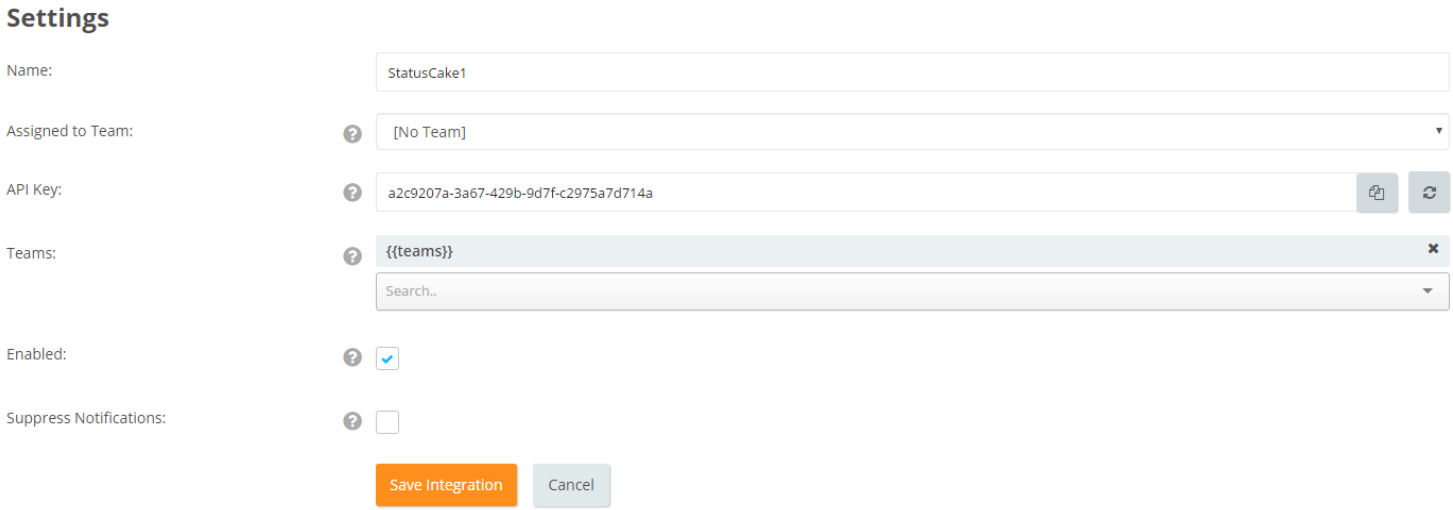
Here we chose to check TNO each minute and we classified it as down if it didn't react in 20 seconds. We added the tag TnoPOC so we know this is our POC test. We didn't specify a host in this case but it is recommended to specify if your using AWS or Azure to host your site.

2.2 Integrating Statuscake with OpsGenie

Integrating Statuscake with OpsGenie is quite simple. The first step logging into OpsGenie and navigating to the intigrations page in the sidebar. There you select add new integration and search for Statuscake.

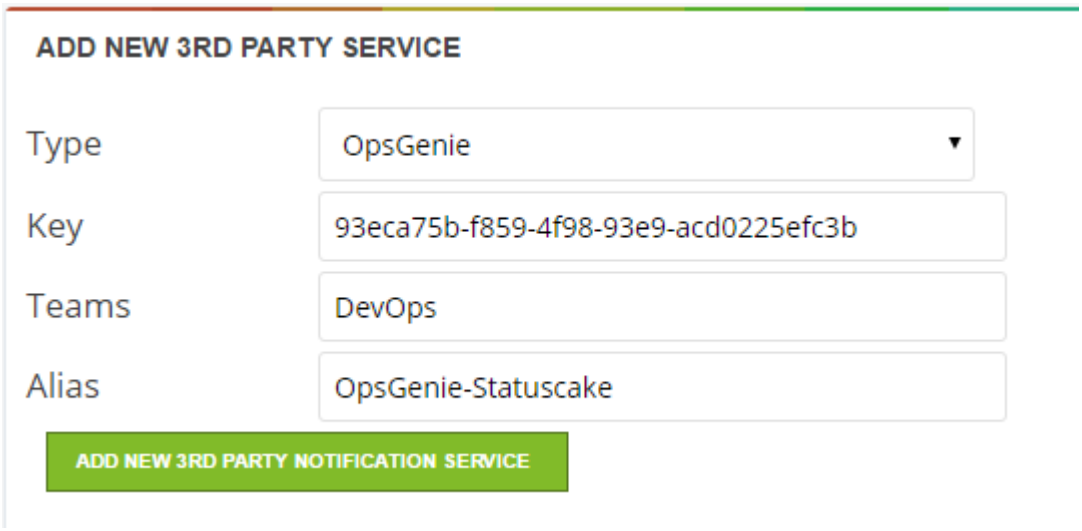


After selecting Statuscake you can give the integration a name and assign teams to it. You can also choose to give just one team access to the integration. The API key is later used in Statuscake. You can choose if you want to enable notification and if so if you want to supress them.

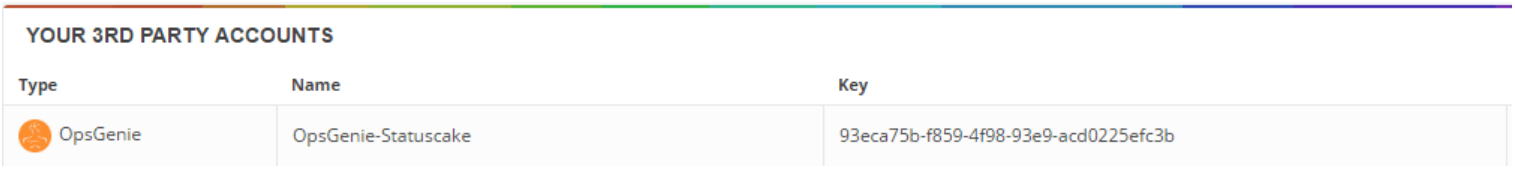


Once your done choosing your settings you click save and you can navigate to Statuscake.

In Statuscake navigate to the integrations page in the sidebar. Once you arrive here you can choose to add a new service. Select OpsGenie and use the key you got from the integration made in OpsGenie. You can then specify the teams you want to alert and give the integration a name like OpsGenie-Statuscake.

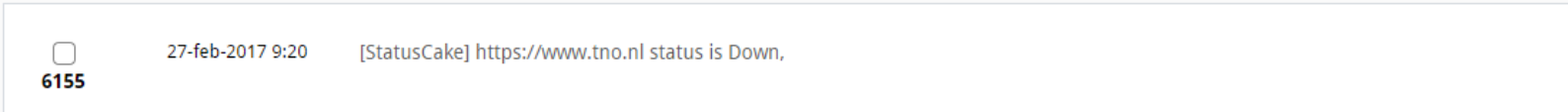


We indicated that the DevOps team needs to be alerted and that the name of the integration should be "OpsGenie-Statuscake". After you choose to add you will notice that there is a 3rd party account added. It should look like this.



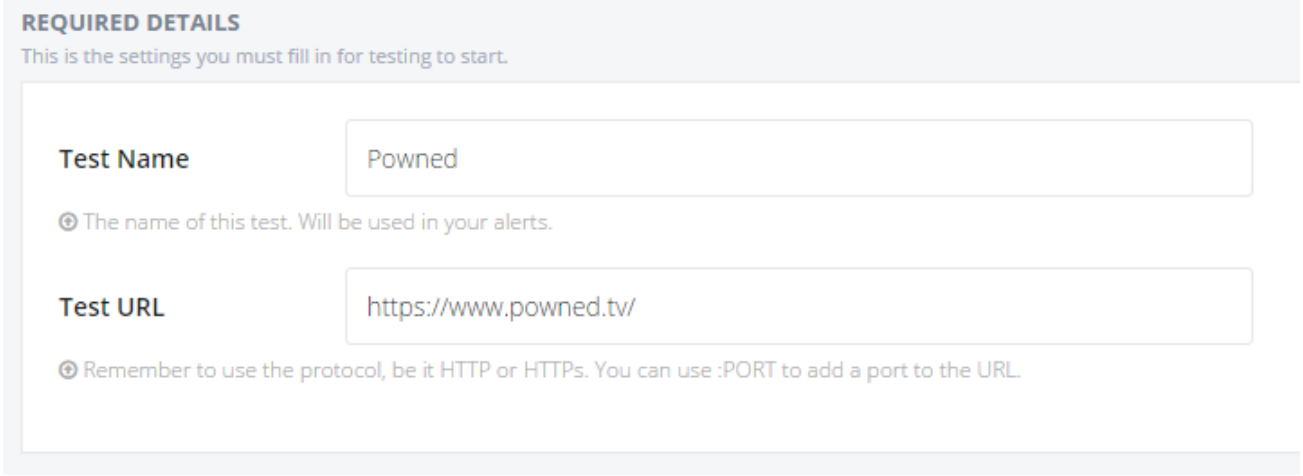
And that's all there needs to be done. You can send a test alert to check if the integration works and of course you can also edit or delete the integration.

After you've made the integration alerts should arrive in OpsGenie and look like this.

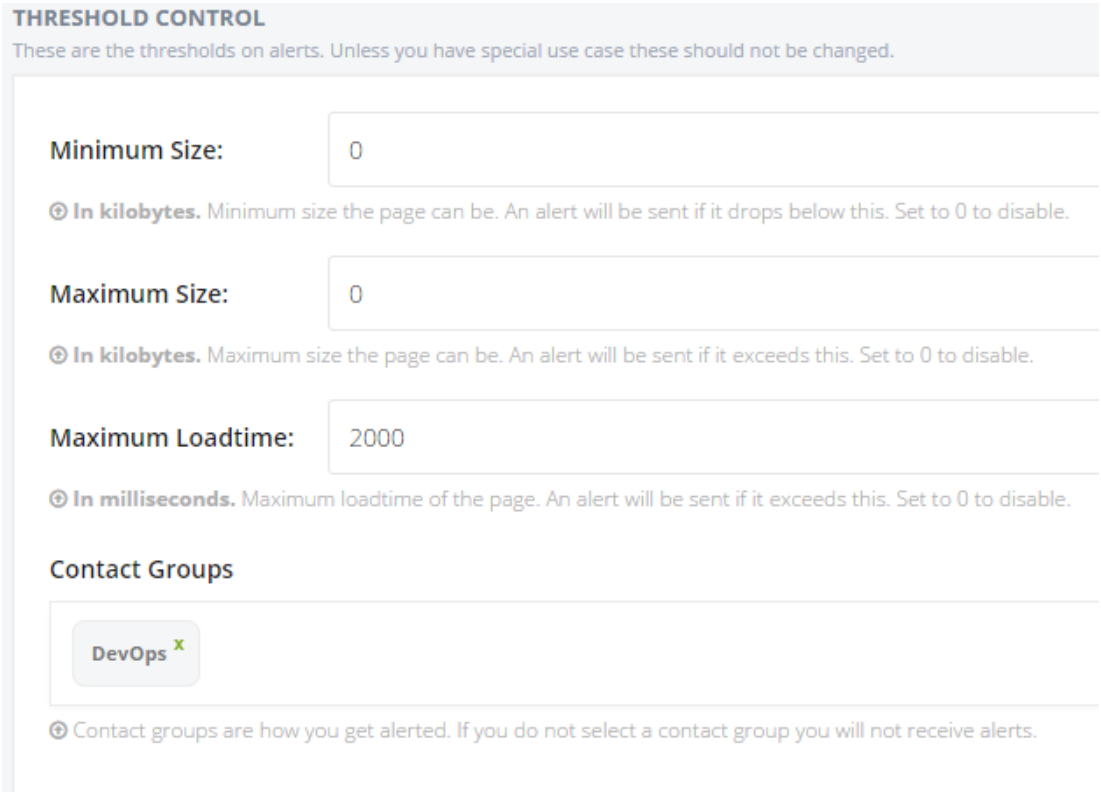


2.3 Creating a test to indicate the full loading time

This test is used to check how long it takes for the full page with all functions to load. This is to get insight in how long real users have to wait before being able to use your site. To create a test navigate to the page speed test. There you can choose to add a new test and view all the page speed test you already have. First you need to give your test a name and indicate which URL the test should check.

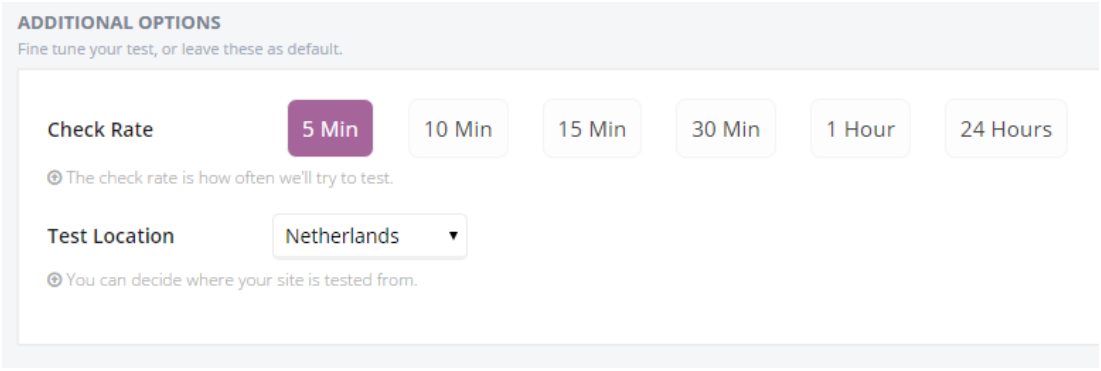


Then you can choose the minimum and the maximum size the site can be. You can also specify in milliseconds what the maximum loadtime of the site can be. You can choose to ignore al these alerting options by leaving these thresholds on 0. You can also select the group which will get alerted when something goes wrong in the test.



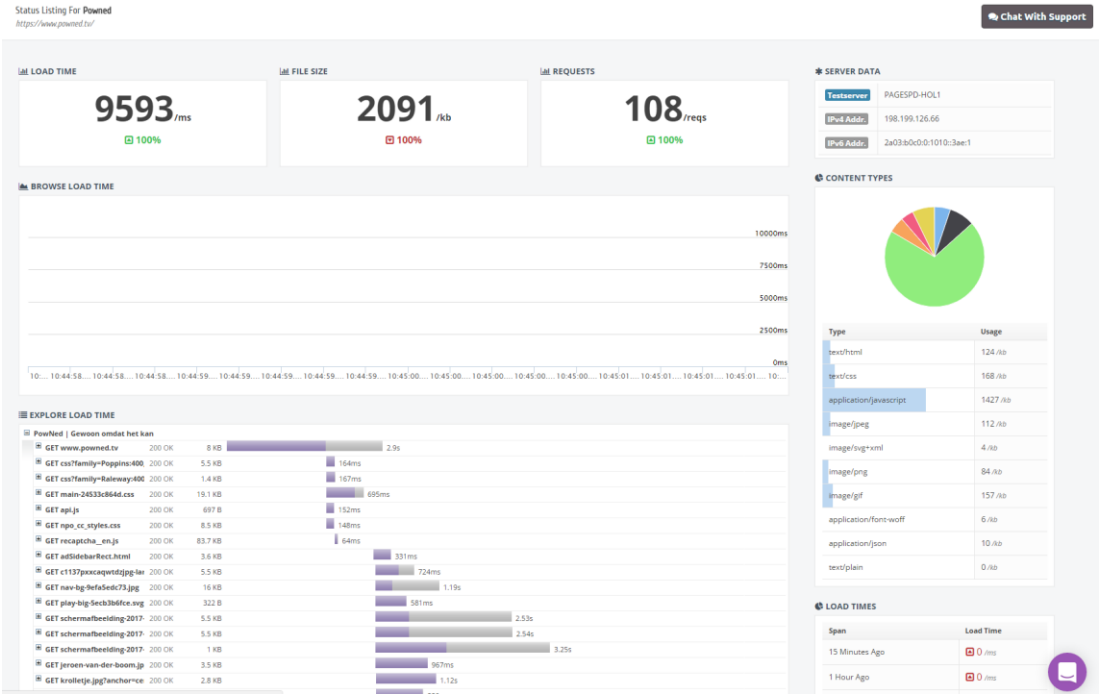
In our case we chose to ignore the min and max size of the page, but we did choose a threshold for the maximum loadtime of 2 seconds (2000 milliseconds). The group which will be contacted in case the page doesn't load fast enough is the DevOps team. You can add multiple groups here if you want.

Last you have to indicate the check rate and the location to test from.



We chose to check every 5 minutes and check from the Netherlands since this site is aimed at dutch visitors.

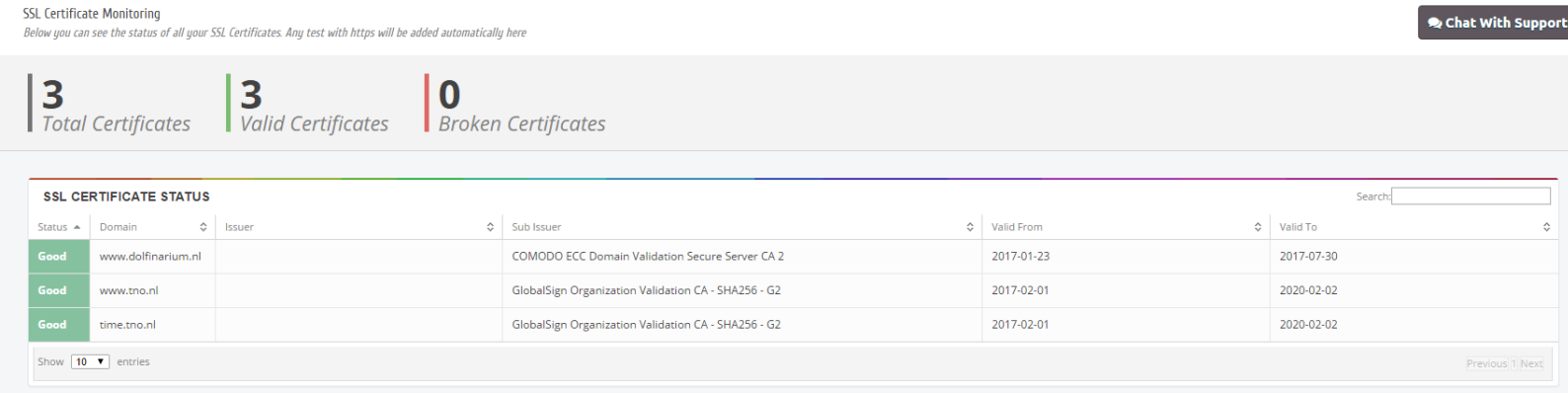
After you've saved the settings the test will be created. You can look into the process of loading the page when you select the page speed test from the overview. You will get a page that looks like this.



If you see this page you've succeeded in making a page speed test.

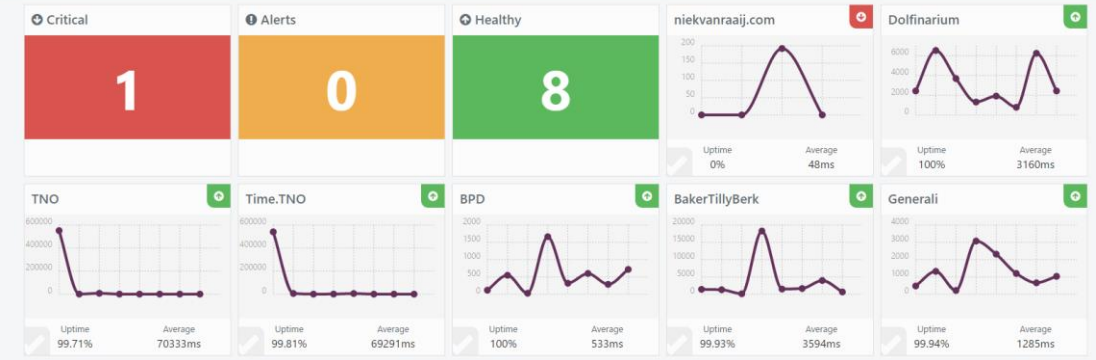
2.4 Create a SSL check

Creating a SSL check is extremely easy. If you add an uptime-test and you monitor an URL with HTTPS will be automatically added to the SSL monitoring page. You can navigate to this overview by clicking on SSL in the sidebar. Then you will see an overview of all the current pages which are being monitored.



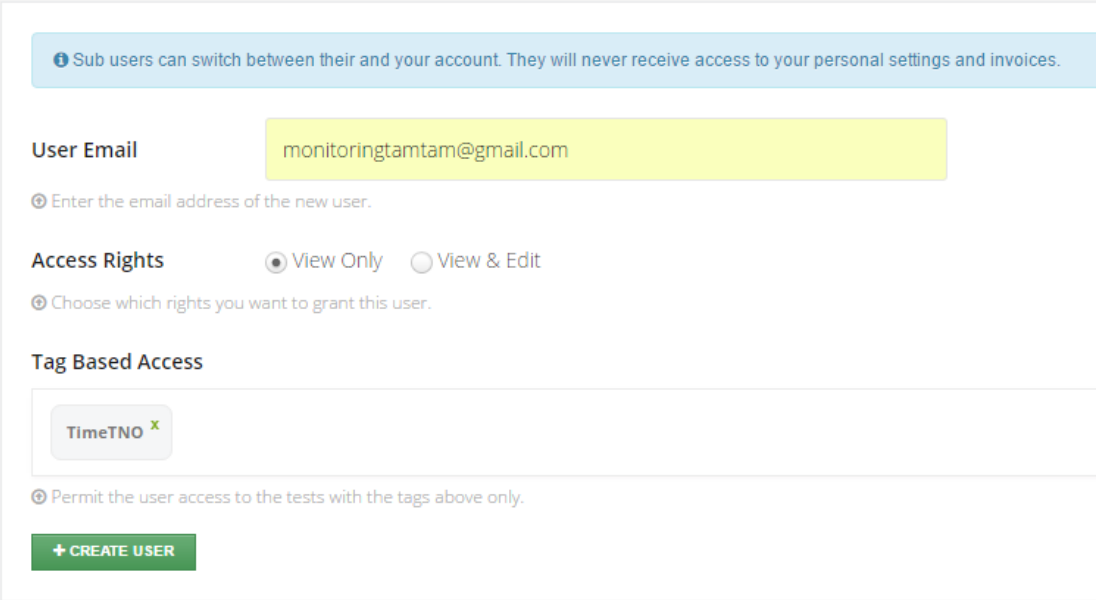
2.5 Extra features in business edition

Dashboard

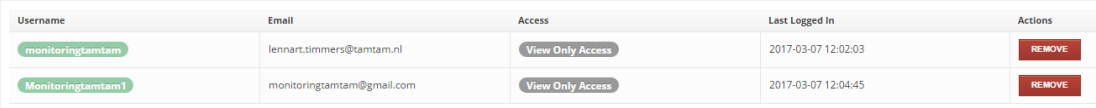


With this function you get a clear overview of all your current tests, if they are failing and how they quickly the response is send. This feature is ideal if you want to know the status of all your available tests. Failing tests also end up at the top of the board.

Adding subusers

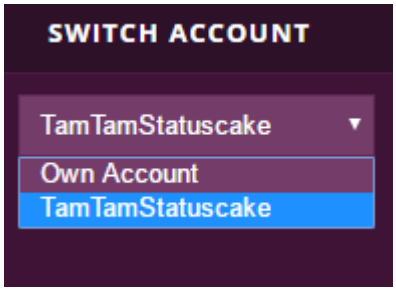


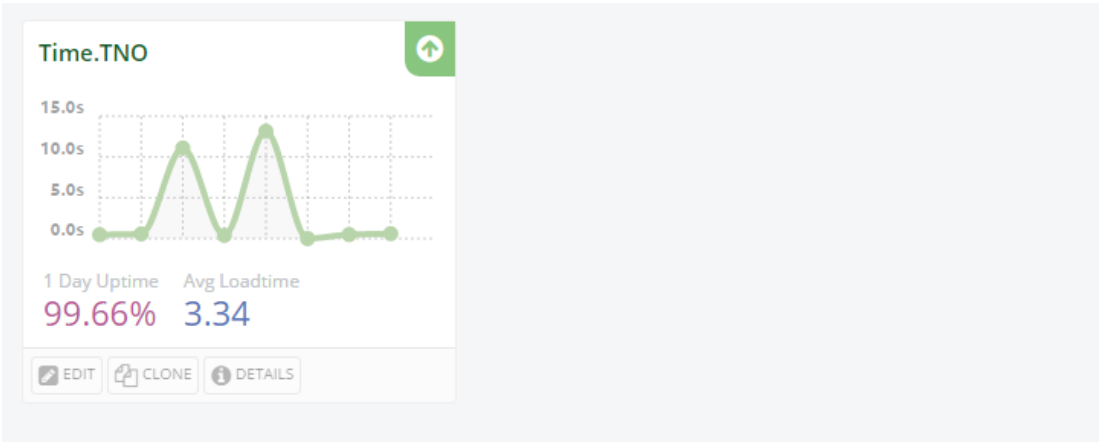
You can add subusers who can only see and optionally also edit certain tests. You can do this by assigning tags to a new user and choosing which rights they will receive.



You can also see when the user last logged in, their access level and you can remove them from statuscake.

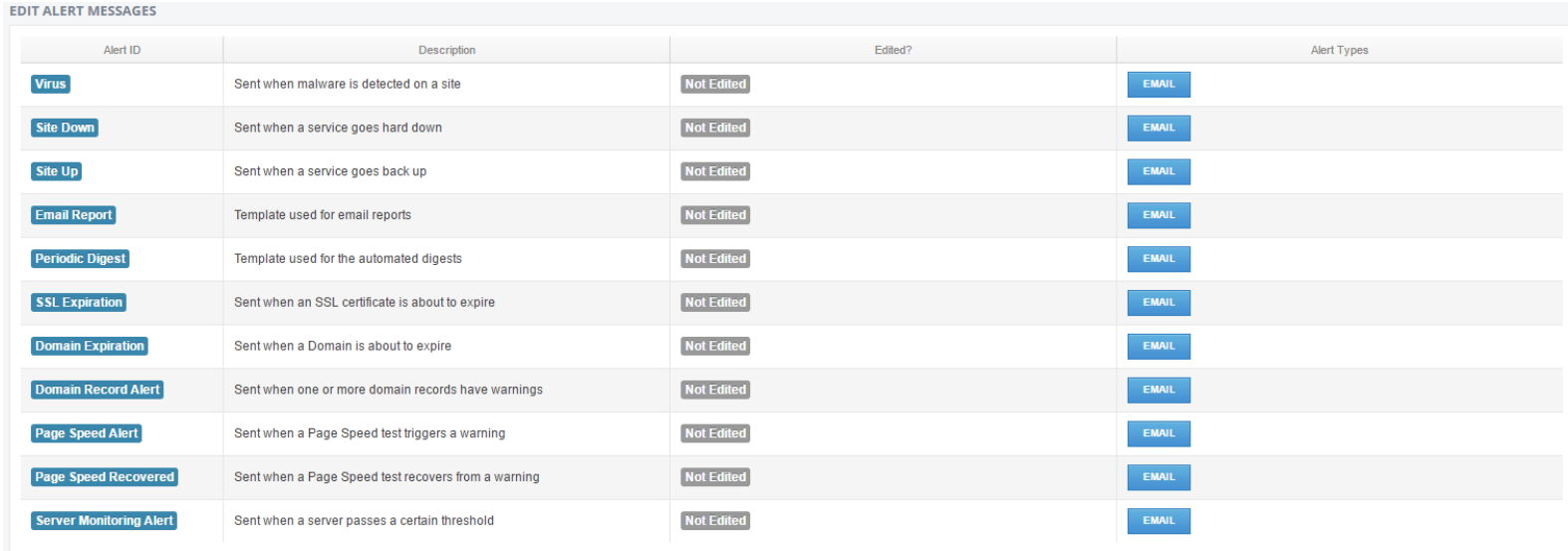
When you log in as a sub user you have to switch to the TamTam account overview. You can do this by in the bottom left corner at the switch account option.





As visible in this picture the user can only access the Time.TNO test. He can see everything related to this test but nothing else.

Email templates



You can also edit the default email templates to contain information or certain senteces you specify.

These options are currently available for custom email templating. You can add these tags in the message and it wil show the value which belongs to the failing test.

For Alerts:

||TITLE|| - Test Name

||SITE|| - Website URL

||TYPE|| – Test Type (HTTP/PING/DNS etc)

||QUOTE|| - TestID and alert number e.g (12345 – 1)

||REASON|| - Downtime Cause

||TIME|| - Total Downtime for test

||HTTPCODE|| – Status code given for downtime

||TESTID|| – The ID of the test that's reporting

||CHECKRATE|| – The check rate of the test reporting

||HOST|| – Hosting Provider

||CONFIRMEDTOTAL|| – Number of confirmations for downtime

||MESSAGE|| – Free text field, content set per test

||TAGS|| – Include test's tags

3. Conclusion

The proof of concept is succefull. We were able to fullfill all the set targets. Statuscake is an interesting tool since it meets all the demands which were set and is simple to use. The price of the tool is also a lot better than the pricing of the currently used tool Pingdom. Pingdom offers very limited tests in their packages while Statuscake offers unlimited test in their tool.

I would recommend using Statuscake as a replacement for Pingdom and UptimeRobot.

TAM / TAM
part of DEPT

G Architectuur document

Architectuur Monitoring.

1 juni 2017

Inhoudsopgave.

1. Inleiding 191

2. Architectuur applicatie 192

3. Architectuur nieuwe monitoring 193

3.1 Uptime 193

3.2 Infrastructuur 194

3.3 Mobile app monitoring 196

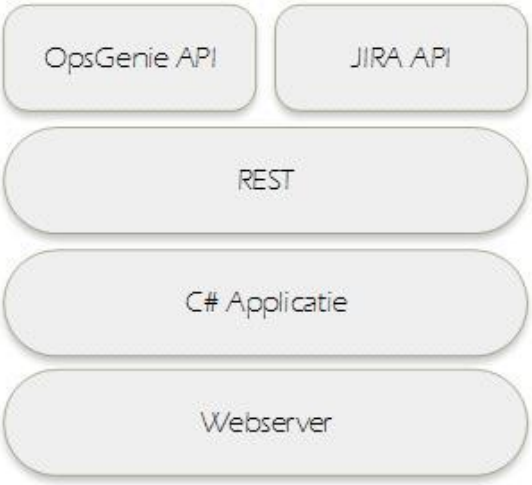
3.4 Alerting 197

1. Inleiding

In dit document wordt de architectuur van de nieuwe monitoringsituatie in kaart gebracht. Aan de hand van dit document zal vervolgens de nieuwe situatie opgebouwd gaan worden. In dit document zal besproken worden hoe de architectuur van de applicatie en de architectuur van de nieuwe monitoringsituatie eruit zal gaan zien.

2. Architectuur applicatie

De architectuur zal er als volgt uitgaan zien:



De modules doen het volgende:

- Webserver
  Omdat de applicatie om de zoveel tijd zal controleren of er open alerts zijn moet de applicatie ergens gehost worden. Dit zal op een webserver gaan gebeuren. Hierdoor kan de applicatie elk uur de code uitvoeren.

- C# applicatie
  In dit gedeelte zal de logica staan. Deze logica zorgt ervoor dat wanneer er een Alert meer dan 1 uur bestaat in OpsGenie, er automatisch een ticket wordt aangemaakt in JIRA.

- REST
  Dit is het protocol wat wordt gebruikt door de API's van OpsGenie en van JIRA.

- OpsGenie API
  Deze API zal elk uur aangesproken worden door de C# applicatie. Met deze API worden de bovenstaande 10 open alerts opgehaald.

- JIRA API
  Deze API wordt gebruikt om een nieuw JIRA ticket aan te maken, wanneer er in de C# applicatie is geconcludeerd dat een alert langer dan een uur openstaat.

3. Architectuur nieuwe monitoring

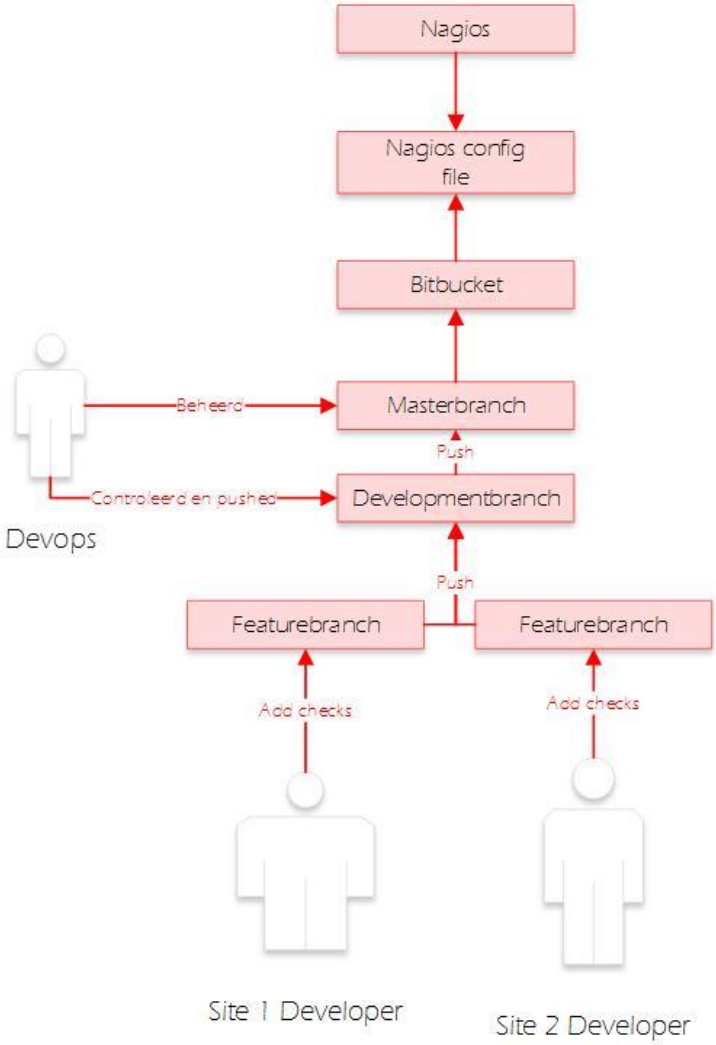
De nieuwe situatie van de monitoring zal veranderen ten opzichte van hoe dit in de huidige situatie wordt gedaan. In dit hoofdstuk is in te zien hoe in de nieuwe situatie een probleem wordt geconstateerd en er hiervoor een alert wordt uitgestuurd. In totaal zijn er vier verschillende tools die in de nieuwe situatie in gebruik genomen gaan worden.

De problemen zullen gedetecteerd gaan worden met drie verschillende tools. Er zal gemonitord gaan worden op niveau van infrastructuur en uptime.

3.1 Uptime

Voor het monitoren van uptime zal Nagios ingezet gaan worden. Nagios wordt op dit moment al gebruikt door TamTam voor de monitoring van een aantal sites door middel van HTTP checks. Verder wordt Nagios ook ingezet door de hosting partner van TamTam Sentia. De servers die TamTam bij Sentia afneemt worden door hen op gebied van infrastuctuur gemonitord met Nagios.

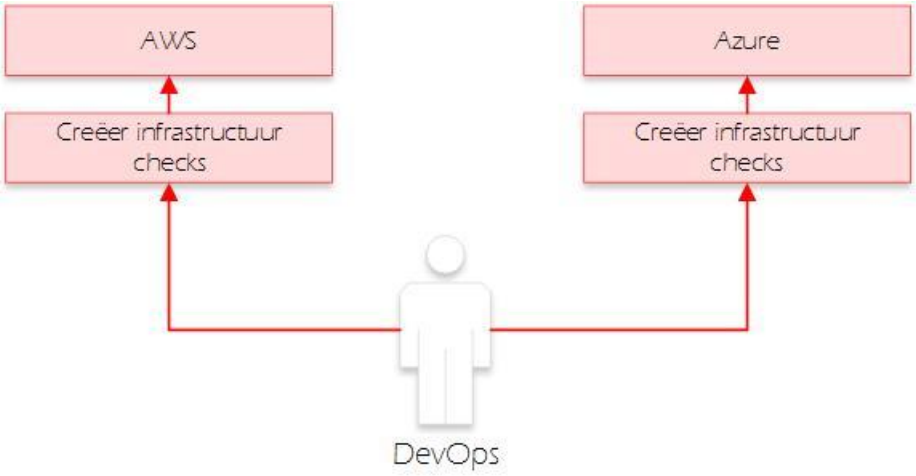
In de nieuwe situatie zal TamTam al haar HTTP checks in Nagios onderbrengen. Dit kan gedaan worden door een check in de config file van Nagios toe te voegen. Omdat er geen fouten in de config file mogen staan mogen developers deze niet direct updaten. Dit zal er als volgt uit gaan zien:



Zoals te zien zal er gebruik gemaakt gaan worden van BitBucket. Op deze manier kunnen alle developers de config file uitchecken en bewerken. Dit zullen zij op een eigen branch doen en dat vervolgens pushen naar de developmentbranch. Deze branch zal door DevOps gecontroleerd worden op fouten. Zitten deze er niet in dan zal de Development branch met de Master gemerged worden en dan zijn de checks toegevoegd in Nagios.

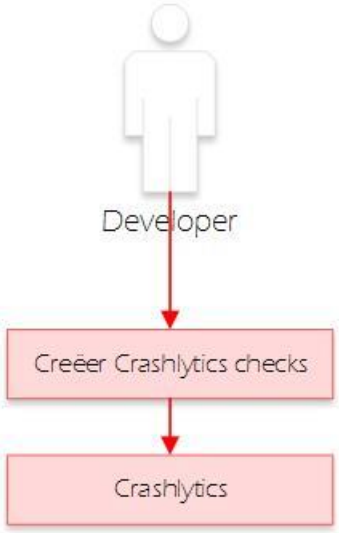
3.2 Infrastructuur

In onderstaand figuur is te zien dat de nieuwe infrastructuur monitoring door DevOps ingericht zal worden. Dit zal op de platformen van AWS en Azure met hun native tooling gebeuren. Aan de hand van de keuzes op gebied van de hosting zal er door DevOps besloten worden wat er gemonitord moet worden. Er is één uitzondering die altijd gemonitord zal worden en dat is de billing. Wanneer dit boven een bepaald plafond stijgt moet er een alert af gaan zodat er niet onverwacht veel geld verdwijnt.


AWS
Azure
Creëer infrastructuur checks
Creëer infrastructuur checks
DevOps

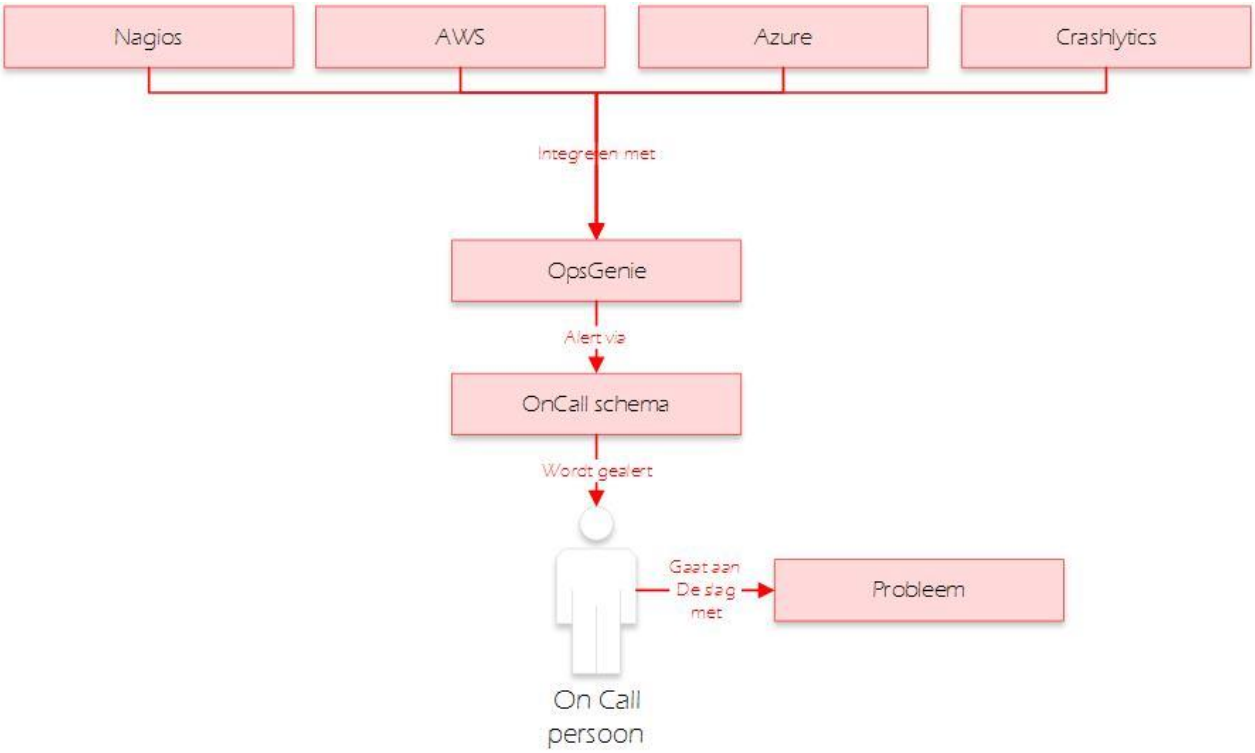
3.3 Mobile app monitoring

Tijdens de advies presentatie kwam nog naar voren dat Arthur voor de mobiele apps ook monitoring heeft ingericht. Dit wordt nu gedaan door middel van de tool Crashlytics. Er worden verder geen andere tools gebruikt voor mobiele apps. Om deze reden is er dan ook gekozen om met Crashlytics te blijven werken. De checks zullen toegevoegd gaan worden door de developers aangezien zij weten welke functionaliteiten essentieël zijn binnen de applicatie. Dit zal er als volgt uit gaan zien:



3.4 Alerting

Voor de alerting zal gebruik gemaakt gaan worden van OpsGenie. Deze tool integreert met Nagios, Azure en AWS. Wanneer een van deze tools dus een alert creëerd dan wordt deze ook in OpsGenie aangemaakt. De nieuwe situatie zal er als volgt uit gaan zien:



Er ontstaat een alert in OpsGenie door een integratie. Vervolgens kijkt OpsGenie welke persoon er on-call is in het schema. Vervolgens gaat OpsGenie deze persoon contacteren op de standaardwijze of als een gebruiker een persoonlijke wijze heeft ingesteld wordt op die manier contact gezocht. Vervolgens pakt deze persoon het probleem op of delegeert dit naar de juiste verantwoordelijke.

TAM / TAM
part of DEPT

H Testdocument

Testrapport Monitoring.

1 juni 2017

Inhoudsopgave.

1. Inleiding 200

2. Automatisch tickets aanmaken in JIRA 201

2.1 Bepalen equivalentieklassen 201

2.2 Logische testgevallen 202

2.3 Fysieke testgevallen 203

2.4 Unit Test 204

3. OpsGenie integraties testen 205

1. Inleiding

Dit document is opgesteld om de nieuwe monitoringsituatie te testen. In dit document worden de volgende onderdelen getest.

- Automatisch tickets aanmaken in JIRA
- De integraties met OpsGenie

Voor het automatisch aanmaken van tickets in JIRA zal er een logisch en fysiek testgeval opgesteld worden. Ook zal er met behulp van unit testen gecontroleerd worden of deze testgevallen kloppen.

Voor de integraties met OpsGenie zal ik een vorm van exploratory testing toepassen. Per integratie zal ik kijken of de nieuwe integraties alerts aanmaken. Gebeud dit niet dan zal ik het probleem oplossen en beschrijven hoe ik dat heb gedaan. Ik verwacht echter dat dit niet nodig zal zijn, omdat deze integraties ook al getest zijn door OpsGenie zelf.

Als er tijdens de testen problemen worden geconstateerd zullen deze opgenomen worden in dit rapport.

2. Automatisch tickets aanmaken in JIRA

In de nieuwe situatie moest er automatisch een ticket aangemaakt worden in JIRA wanneer een OpsGenie alert langer dan een uur open had gestaan. Om er zeker van te zijn dat deze geschreven functie precies doet wat deze moet doen is deze goed getest in een lokale omgeving van JIRA. In de JIRA van TamTam mag deze functie namelijk niet irrelevante tickets aan maken.

2.1 Bepalen equivalentieklassen

Attribuut	**Beperking**
Summary	Max 255 karakters
Description	Geen beperking
IssueType	Moet een bestaand issuetype in JIRA zijn
Projectkey	Moet een bestaande key in JIRA zijn

Aangezien Description geen beperkingen heeft wordt deze genegeerd in de testcases.

Hieronder worden andere beperkingen aangegeven die niet met de syntax te maken hebben.

- Er wordt een speciale account gebruikt voor het toevoegen van deze tickets. Deze account mag alleen op het project DESK nieuwe tickets toevoegen. Wanneer deze op een ander bestaand project een ticket toevoegd mag dit niet slagen.

Attribuut	**Geldige klassen**	**Ongeldige klassen**
Summary	>=1 Tekstlengte <=255	1< Tekstlengte >255
IssueType	Aanwezige standard issuetypes	Niet bestaande types en sub-task issuetypes
Projectkey	DESK	testboard waar gebruiker geen rechten heeft en ongeldige waardes

2.2 Logische testgevallen

Hieronder zijn de opgestelde logische testgevallen te vinden. De waardes zijn gebaseerd op de equivalentieklassen.

Testnummer	**Summary**	**IssueType**	**Projeckey**	**Verwacht resultaat**
1	>=1 Tekstlengte <=255	Aanwezige standard issuetype	DESK	Ticket aangemaakt in Desk met als type Task, summary “test” en description “test”
2	1< Tekstlengte	Aanwezige standard issuetype	DESK	Ticket is niet aangemaakt. summary is te kort.
3	Tekstlengte >255	Aanwezige standard issuetype	DESK	Ticket is niet aangemaakt. summary is te lang.
4	>=1 Tekstlengte <=255	Niet bestaande type	DESK	Ticket is niet aangemaakt omdat het issue niet bestaat.
5	>=1 Tekstlengte <=255	sub-task issuetype	DESK	Ticket is niet aangemaakt omdat een subissue een parent moet hebben
6	>=1 Tekstlengte <=255	Aanwezige standard issuetype	testboard waar gebruiker geen rechten heeft	Ticket is niet aangemaakt omdat gebruiker geen toegang tot TES heeft
7	>=1 Tekstlengte <=255	Aanwezige standard issuetype	ongeldige waarde	Ticket is niet aangemaakt omdat een ongeldige projectkey niet geaccepteerd wordt.

2.3 Fysieke testgevallen

Hieronder zijn de opgestelde fysieke testgevallen te vinden. De tabel is gebaseerd op de logische testgevallen.

Testnummer	**Summary**	**IssueType**	**Projeckey**	**Verwacht resultaat**
1	Test	Task	DESK	Ticket aangemaakt in Desk met als type Task, summary “test”
2	(lege string)	Task	DESK	Ticket is niet aangemaakt. summary is te kort.
3	Tekst langer dan 255 tekens*	Task	DESK	Ticket is niet aangemaakt. summary is te lang.
4	Test	NepIssueType	DESK	Ticket is niet aangemaakt omdat het issue niet bestaat.
5	Test	Sub-task	DESK	Ticket is niet aangemaakt omdat een subissue een parent moet hebben
6	Test	Task	TES	Ticket is niet aangemaakt omdat gebruiker geen toegang tot TES heeft
7	Test	Task	UNTA	Ticket is niet aangemaakt omdat projectkey niet geaccepteerd wordt.

*Tekst =
asdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdf
sdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfs
dfsdfsdffsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdf

In total 258 tekens

2.4 Unit Testen

Om bovenstaande testen uit te voeren zijn er unit testen geschreven. De waardes die hierin zijn gebruikt zijn afkomstig uit de logische testgevallen. Voor elk logisch testgeval is een test geschreven. Deze zien er als volgt uit.

```
AlertHandler handler = new AlertHandler();
[TestMethod]
| 0 references | 0 changes | 0 authors, 0 changes
public void CreateTicket()
{
    JiraTicket ticket = new JiraTicket("Test","", "DESK", "Task");
    Assert.IsTrue(handler.CreateJiraIssue(ticket) == true);
}
[TestMethod]
| 0 references | 0 changes | 0 authors, 0 changes
public void CreateTicketEmptySummaryString()
{
    JiraTicket ticket = new JiraTicket("", "", "DESK", "Task");
    Assert.IsTrue(handler.CreateJiraIssue(ticket) == false);
}
[TestMethod]
| 0 references | 0 changes | 0 authors, 0 changes
public void CreateTicketSummaryStringToLong()
{
    JiraTicket ticket = new JiraTicket("asdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsdfsd
    Assert.IsTrue(handler.CreateJiraIssue(ticket) == false);
}
[TestMethod]
| 0 references | 0 changes | 0 authors, 0 changes
public void CreateTicketIssueTypeFake()
{
    JiraTicket ticket = new JiraTicket("test ", "", "DESK", "NepIssueType");
    Assert.IsTrue(handler.CreateJiraIssue(ticket) == false);
}
[TestMethod]
| 0 references | 0 changes | 0 authors, 0 changes
public void CreateTicketIssueSubTask()
{
    JiraTicket ticket = new JiraTicket("test ", "", "DESK", "Sub-task");
    Assert.IsTrue(handler.CreateJiraIssue(ticket) == false);
}
[TestMethod]
| 0 references | 0 changes | 0 authors, 0 changes
public void CreateTicketKeyNoAcces()
{
    JiraTicket ticket = new JiraTicket("test ", "", "TES", "Task");
    Assert.IsTrue(handler.CreateJiraIssue(ticket) == false);
}
[TestMethod]
| 0 references | 0 changes | 0 authors, 0 changes
public void CreateTicketKeyDoesntExist()
{
    JiraTicket ticket = new JiraTicket("test ", "", "UNTA", "Task");
    Assert.IsTrue(handler.CreateJiraIssue(ticket) == false);
}
```

Zoals te zien is er een test gefaald en is de rest van de testen geslaagd. De gefaalde test maakt een ticket aan in een project waar deze geen toegang tot heeft. Toen ik echter ging kijken op de account waarmee tickets aangemaakt worden zag ik dat deze wel toegang had tot het bord TES. Nadat ik de rechten bij deze account had ingetrokken slaagde deze test ook.

```
[TestMethod]
| 0 references | 0 changes | 0 authors, 0 changes
public void CreateTicketKeyNoAcces()
{
    JiraTicket ticket = new JiraTicket("test ", "", "TES", "Task");
    Assert.IsTrue(handler.CreateJiraIssue(ticket) == false);
}
```

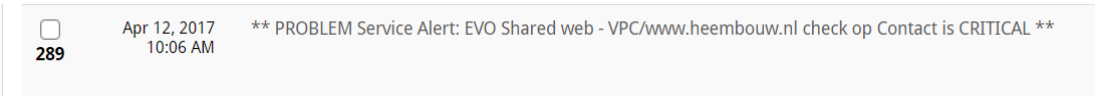
3. OpsGenie integraties testen

Om de nieuwe OpsGenie situatie werkend te krijgen heb ik de volgende integraties toegevoegd aan OpsGenie.

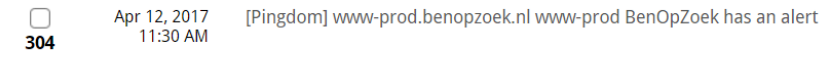
- Nagios
- Pingdom
- Azure
- AWS
- Crashlytics

Nadat ik deze integraties had toegevoegd heb ik gelijk een test alert vanuit elke situatie gestuurd. Met geen enkele integratie heb ik problemen ondervonden. Hieronder staan screenshots per Tool die aantonen dat de integratie werkt.

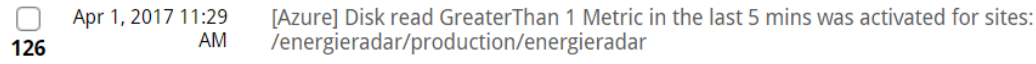
Nagios



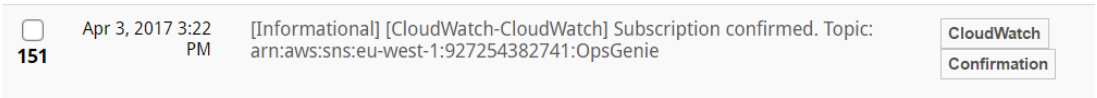
Pingdom



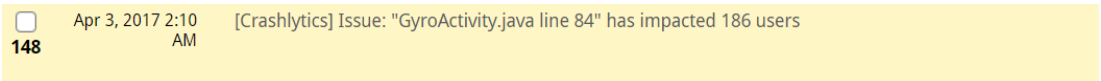
Azure



Cloudwatch



Crashlytics



I Requirements devops.tamtam.nl

Requirementsdocument.

Devops.TamTam.nl

1 juni 2017

Versiebeheer

Datum	Samenvatting
19-4-17	Initiële opzet document. Eerste geconstateerde requirements uit het gesprek met Niek
20-4-17	Verwerken van gesprek met Leon
24-4-17	Verwerken feedback op requirementsdocument en finaliseren.

Inhoudsopgave.

Versiebeheer **207**

1. Inleiding **209**

2. Scope en Stakeholders **210**

2.1 Scope 210

2.2 Stakeholders 210

3. Functionele requirements **212**

4. Niet-functionele requirements **214**

5. Business requirements **214**

1. Inleiding

Dit rapport bevat de wensen en eisen die door TamTam gesteld worden aan de website Devops.TamTam.nl. De requirements zijn gebaseerd op gesprekken met de stakeholders. In dit document zal eerst een scope opgesteld worden en de stakeholders zullen geïdentificeerd worden.

2. Scope en Stakeholders

2.1 Scope

Voor TamTam moet er een systeem gemaakt worden waarmee er automatisch nieuwe en out of service projecten in JIRA tickets aangemaakt kunnen worden. Dit moet in de vorm van een site zodat deze beschikbaar gesteld kan worden binnen het bedrijf. Er moeten pakketten kunnen aangemaakt worden, zodat project leads een keuze krijgen tussen de verschillende vormen van hosting wanneer zij het intake formulier voor een nieuw project invullen.

Ook moeten medewerkers van de servicedesk met behulp van een formulier out of serviceprojecten kunnen insturen. Deze informatie moet dan automatisch in de vorm van een aantal standaardtickets en de tickets die bij een pakket horen aangemaakt worden in het JIRA hosting bord.

Er is een kans dat er requirements naar voren komen die wel belangrijk zijn maar niet binnen de scope vallen. Deze requirements zullen wel opgeschreven worden, maar er zal wel bij vermeld worden dat deze buiten de scope vallen.

Alle requirements zullen geprioritiseerd worden met de methode MOSCOW (could have, should have, must have en won’t have but would like). Hiermee kan ik aangeven welke requirements een hogere prioriteit hebben en deze requirements als eerste toevoegen.

2.2 Stakeholders

De volgende stakeholders zijn betrokken bij deze oplossing

- **Medewerkers van TamTam**
  Het eindproduct van dit project zal door de medewerkers van TamTam gebruikt gaan worden om voor nieuwe projecten een hosting omgeving te laten inrichten en out of service projecten aan te maken.

- **Leon van Rees**
  Opdrachtgever van deze opdracht, Lead DevOps en Operational Services

- **Niek van Raaij**
  DevOps engineer

- **Medewerkers afdeling DevOps**
  Zij zullen de tickets die met deze tool aangemaakt worden gaan oppakken en verwerken. Ook zullen zij admin rechten op de site hebben waarmee ze standaard tickets en pakketten mee kunnen beheren.

3. Functionele requirements

Req	Requirement	Bron
1	Het systeem moet tickets kunnen aanmaken in het HOSTING bord van TamTam. **(must have)**	Gesprek Niek, Leon
2	Een admin moet out of service tickets kunnen aanmaken. **(must have)**	Gesprek Niek
3	Een admin moet out of service tickets kunnen updaten. **(should have)**	Gesprek Niek, Leon
4	Een admin moet out of service tickets kunnen verwijderen. **(must have)**	Gesprek Niek, Leon
5	Een admin moet out of service tickets kunnen inzien. **(must have)**	Gesprek Niek
6	Een admin moet pakketten kunnen aanmaken. **(must have)**	Gesprek Niek
7	Een admin moet pakketten kunnen bewerken. **(should have)**	Gesprek Niek
8	Een admin moet pakketten kunnen verwijderen. **(must have)**	Gesprek Niek
9	Een admin moet pakketten kunnen inzien. **(must have)**	Gesprek Niek
10	Een gebruiker moet een out of service project kunnen registreren.	Gesprek Niek, Leon
11	Een gebruiker moet een intake formulier voor nieuwe hosting kunnen invullen. **(must have)**	Gesprek Niek, Leon
12	Een ticket bestaat uit een samenvatting en een beschrijving. **(must have)**	Zelf vastgesteld
13	Een pakket heeft een naam, beschrijving en optioneel een template-url **(must have)**	Gesprek Niek
14	Een product heeft een naam, setup kosten en maandelijkse kosten. **(must have)**	Gesprek Niek, Leon
15	Een admin moet producten kunnen toevoegen aan pakketten. **(must have)**	Gesprek Niek, Leon
16	Een admin moet tickets kunnen toevoegen aan pakketten. **(must have)**	Gesprek Niek, Leon
17	Een pakket bestaat uit tickets en producten. **(must have)**	Gesprek Niek, Leon
18	Een admin moet tickets kunnen verwijderen. **(must have)**	Gesprek Niek
19	Een admin moet producten kunnen verwijderen. **(must have)**	Gesprek Niek
20	Een admin moet tickets kunnen bewerken. **(should have)**	Gesprek Niek
21	Een admin moet producten kunnen bewerken **(should have)**	Gesprek Niek
22	Een admin moet bestaande tickets kunnen inzien **(must have)**	Gesprek Niek
23	Een admin moet bestaande producten kunnen inzien **(must have)**	Gesprek Niek
24	Het systeem moet automatisch een ticket met subtickets aanmaken wanneer het out of service of het nieuwe hosting formulier is ingestuurd door een gebruiker. **(must have)**	Gesprek Leon, Niek
25	In het hoofdticket moet de naam van het project staan. **(must have)**	Gesprek Leon, Niek
26	In het hoofdticket moet het type ticket aangeduid worden namelijk, “out of service” of “nieuwe hosting”. **(must have)**	Gesprek Niek

27	Het subticket moet een taak hebben die uitgevoerd moet worden. **(must have)**	Gesprek Niek

4. Niet-functionele requirements

Req	Requirement	Bron
NF1	De formulieren moeten niet te ingewikkeld zijn om in te vullen.	Gesprek Niek

4.1 Business requirements

Req	Requirement	Bron
BR1	Intern moet devops.tamtam.nl ook in gebruik genomen worden door project leads en de servicedesk	Gesprek Niek