Portfolio Semester 6

Naam: Rienk Engbrenghof

Studentnummer: 429059

Klas: S6-RB03T

# Introductie

Mijn naam is Rienk Engbrenghof, ik ben 22 jaar en woon in Weert. Momenteel bestaat mijn ervaring met software development uit het doorlopen van de verschillende software semesters. En het volgen van de pre-master Information Management. Hier werd ook met Python gewerkt en dit doe ik ook in mijn vrije tijd. Mijn kennis over Enterprise software is momenteel minimaal.

Mijn doel is voor dit semester om meer te leren over Enterprise software en om mijn vaardigheden qua programmeren nog verder te ontwikkelen. Ook wil ik mij nog verder professioneel ontwikkelen op verschillende gebieden zoals bijvoorbeeld: communicatie en presenteren. Op deze opleiding heb ik ontdekt dat programmeren een van mijn grote interesses is en dat ik dit ook graag doe.

# Versiebeheer

|  |  |
| --- | --- |
| 25/2/2022 | Opzet gemaakt en introductie toegevoegd. |
| 16/3/2022 | Extra toevoegingen omtrent Sprint 1 toegevoegd. |
| 9/4/2022 | Feedback en Sprint 2 toegevoegd. |
| 1/6/2022 | Sprint 3 toegevoegd. |
| 15/6/2022 | Sprint 4 toegevoegd. |
| 17/6/2022 | Sprint 5 en extra informatie aan alle leerdoelen toegevoegd. |

# Link naar repository:

https://github.com/Pretter-S6

# Inhoudsopgave

[1. Leeruitkomsten 5](#_Toc106557678)

[1.1 Developing Enterprise Software as a Team Effort 5](#_Toc106557679)

[1.2 Context Based Research 7](#_Toc106557680)

[1.3 Preparation for Life-Long Learning 9](#_Toc106557681)

[1.4 Scalable Architectures 11](#_Toc106557682)

[1.4.1 Practical 11](#_Toc106557683)

[1.4.2 Technical 13](#_Toc106557684)

[1.5 Development and Operations (DevOps) 15](#_Toc106557685)

[1.5.1 Practical 15](#_Toc106557686)

[1.5.2 Technical 16](#_Toc106557687)

[1.6 Cloud Services 19](#_Toc106557688)

[1.6.1 Practical 19](#_Toc106557689)

[1.6.2 Technical 20](#_Toc106557690)

[1.7 Security by Design 23](#_Toc106557691)

[1.7.1 Practical 23](#_Toc106557692)

[1.7.2 Technical 24](#_Toc106557693)

[1.8 Distributed Data 26](#_Toc106557694)

[1.8.1 Practical 26](#_Toc106557695)

[1.8.2 Technical 27](#_Toc106557696)

[2. Conclusie 30](#_Toc106557697)

# Leeruitkomsten

## **1.1** [Developing Enterprise Software as a Team Effort](https://fhict.instructure.com/courses/12088/outcomes/16833)

**Omschrijving**

Zorg ervoor dat je laat zien hoe jij (en jullie als groep) beter zijn geworden in samenwerking gedurende het semester. Waaruit blijkt dat? Is de velocity toegenomen? Zijn jullie beter geworden in het naleven van het Scrum proces? Zijn jullie elkaar beter gaan begrijpen? Is de onderlinge communicatie verbeterd? Is het contact met de PO/stakeholders verbeterd? Ga een peer review met de groep uitvoeren. Welke resultaten en actiepunten zijn daar uitgekomen?

**Ontwikkelschaal**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Beschrijving** | **Type** | **Niveau** |
| 1.1 | Sprint 1 | Individueel  Groepsproject | Beginning |
| 1.2 | Sprint 2 | Individueel  Groepsproject | Beginning |
| 1.3 | Sprint 3 | Individueel  Groepsproject | Proficient |
| 1.4 | Sprint 4 | Individueel  Groepsproject | Proficient |
| 1.5 | Sprint 5 | Individueel  Groepsproject | Proficient |

**Onderbouwing**

1.1.1: Sprint 1

Voor de keuze van platform was bij proftaak al gekozen. Hierbij wordt gebruik gemaakt van Vue frontend en .NET core backend (opgedeeld in microservices). Dit is goed geschikt als enterprise software development platform. Dit proftaak project is al overdraagbaar en wij hebben het dan ook overgenomen van een andere groep. Binnen de proftaak werken wij ook al agile. Om snel tot nieuwe kennis te komen heb ik veel gesproken met Sjors. Zodat ik het project sneller begreep.

Voor mijn eigen project heb ik dan ook gekozen voor dezelfde architectuur. Ik doe hier alles hetzelfde behalve het frontend platform. Hierbij maak ik gebruik van een React frontend (wat ook geschikt is voor enterprise software development).

Mijn eerste indruk van de groep is dat vrijwel iedereen duidelijk kan communiceren en dat de onderlinge communicatie en dat het overdragen van informatie onderling geen probleem moet worden.

1.1.2: Sprint 2

Tijdens deze sprint zijn wij begonnen met het beter naleven van het scrumproces. Hiervoor stond de basis eerst nog niet goed en dat hebben wij nu wel weten te bereiken in Github. Ook de onderlinge communicatie is verbeterd: bijvoorbeeld wanneer iemand niet uit zijn taak komt, dan zeggen we dit en rouleren we taken om een verfrissende blik te richten op de taak. Voor het contact met de stakeholder hebben wij wekelijkse meetings opgezet. Om de stakeholders zo betrokken mogelijk te houden.

1.1.3: Sprint 3

Ook deze sprint hebben wij voortgang gemaakt op het gebied van als een team scrummen. We hebben fibonacci nummers toegepast in ons proces. Zodat we beter kunnen inschatten hoe veel moeite een taak kost. Dit omdat er 2 grote taken waren die veel tijd in beslag namen.

1.1.4: Sprint 4

Tijdens deze sprint hebben wij gereflecteerd, zowel individueel als gezamenlijk. Omdat de communicatie met de stakeholder stroef verliep. Hierdoor hebben wij tijdens de oplevering de knelpunten benoemd, en gezegd wat er volgens ons beter kon afgelopen 2 sprints. Jacques (onze stakeholder stemde hiermee in).

1.1.5: Sprint 5

Door de duur van deze sprint hebben wij niet echt meer progressie kunnen boeken. We zijn vooral als een team bezig geweest met het nadenken over de overdraagbaarheid.

**Reflectie op Voortgang**

1.2.1: Sprint 1

Ik denk dat ik door mijn proactieve houding snel de concepten heb kunnen begrijpen en daardoor nu al weet wat er van mij verwacht wordt. Ik ben over het algemeen tevreden met mijn progressie omtrent enterprise software en mijn positie binnen het proftaak team.

1.2.2: Sprint 2

Mijn voortgang binnen deze sprint was vrij langzaam. Dit omdat ik een taak (Fontys login) heb die vrij ingewikkeld is en ik mij er dan ook toe aan moest zetten om hulp te vragen. Dit had ik eerder moeten doen, zodat een nieuwe blik mij kon helpen.

1.2.3: Sprint 3

Ik denk dat ik tijdens deze sprint vooruit ben gegaan als in de zin van, ik heb beter gecommuniceerd over mijn taken en wat ik aan het doen was. Ook heb ik tijdens deze sprint mijn teamgenoten beter kunnen ondersteunen bij vragen. Hierdoor is de groepsdynamiek ook beter dan eerst.

1.2.4: Sprint 4

**We hadden als groep eerder moeten inzien dat het contact met de stakeholder erg minimaal was. Daarom zijn we ook gaan samenzitten. Door de matige communicatie leek het misschien alsof er minder is gedaan, terwijl dit niet het geval is. Het is achteraf goed dat we samen gereflecteerd hebben, al had dit beter eerder kunnen plaatsvinden. Zodat we meer tijd hadden om te laten zien dat we ook daadwerkelijk goed kunnen communiceren met stakeholders.**

1.2.5: Sprint 5

**Weinig op deze sprint aan te merken, door de duur is er weinig om op te reflecteren.**

## **1.2 Context Based Research**

**Omschrijving**

Belangrijk is dat je het DOT-framework aanhaalt, en minimaal laat zien welke strategieën en methoden je hebt toegepast. Je mag case studies aanhalen, maar lever dan ook bewijs aan dat je op de resultaten van de case studies positieve feedback hebt ontvangen. Verder kan je ook nog schrijven over de onderzoeksopdracht (waar je ook feedback op hebt gehad). Probeer verder eens te reflecteren op het proces van onderzoek: wat vind je er van? wat heeft het je gebracht? zou je dit in toekomstige projecten ook zo doen, of zou je het dan anders aanpakken? Zorg ervoor dat je zorgvuldig omgaat met bronvermeldingen. Doe geen uitspraken zodat dat je daarbij laat zien dat het ondersteund wordt door bronnen van experts.

**Ontwikkelschaal**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Beschrijving** | **Type** | **Niveau** |
| 1.1 | Sprint 1 | Individueel  Groepsproject | Beginning |
| 1.2 | Sprint 2 | Individueel  Groepsproject | Beginning |
| 1.3 | Sprint 3 | Individueel  Groepsproject | Proficient |
| 1.4 | Sprint 4 | Individueel  Groepsproject | Proficient |
| 1.5 | Sprint 5 | Individueel  Groepsproject | Proficient |

**Onderbouwing**

1.1.1: Sprint 1

Voor mijn individuele project heb ik voor het emerging trends research verschillende vragen opgesteld met bijbehorende onderzoeksmethoden. Dit onderzoek is gestructureerd op basis van het dot framework. Een van de deelvragen overweegt ook of er misschien andere (betere) opties zijn dan wat ik nu implementeer voor mijn individuele project.

Voor het groepsproject heb ik kritisch gekeken naar de verschillende onderzoeksvragen die al gemaakt waren. Hierbij heb ik verschillende aanvullende/vervangende onderzoeksvragen opgesteld die het project verrijken. Hierbij vullen we de bestaande onderzoeksvragen aan met methoden, maar stellen we ook nieuwe vragen en methoden op. Dit om tot de leerdoelen te komen.

1.1.2: Sprint 2

Binnen sprint 2 heb ik de feedback op mijn onderzoeksvragen verwerkt. Mede doordat het onderzoek niet zo veel tijd mag innemen, heb ik de onderzoeksvragen moet inperken. Ook ben ik voor de proftaak bezig geweest met het aanpassen van de onderzoeksvragen. Deze feedback werd gegeven door Tom Langhorst.

1.1.3: Sprint 3

Binnen sprint 3 heb ik mijn onderzoek succesvol afgerond. Hierin heb ik alle onderzoeksvragen beantwoord door middel van de DOT methoden. Ook ben ik bezig geweest met onderzoek naar Messaging in de proftaak.

1.1.4: Sprint 4

Aangezien mijn onderzoek al af was, heb ik hier niet verder naar gekeken. Wel heb ik voor de leerdoelen nog verder onderzoek gedaan. Bijvoorbeeld een available product analysis voor de verschillende cloudservices.

1.1.5: Sprint 5

Tijdens de afrondende fase ben ik niet echt bezig geweest met verder context research, het leerdoel was ook al voldoende. Dus dit had geen prioriteit.

**Reflectie op Voortgang**

1.2.1: Sprint 1

Omdat een deel van het onderzoek van de proftaak al is gedaan, is het van belang om kritisch naar dit onderzoek te kijken. Mijn producten voor zowel individueel als voor groepsproject zijn nog niet gevalideerd. Maar het zal niet lang duren voordat dit wel het geval is. We hebben al verschillende nieuwe vragen opgezet en uitgewerkt voor proftaak. Hierbij maken we ook gebruik van het DOT framework.

1.2.2: Sprint 2

Vooral op mijn eigen onderzoeksvragen kan ik vooruitgang boeken, hierbij moesten de vragen specifieker voor de opdracht, zoals bijvoorbeeld het noemen van namen en talen. Ook moesten de vragen duidelijker, zoals bijvoorbeeld benoemen wat benodigd is binnen de vraag.

1.2.3: Sprint 3

Mijn voortgang qua research based onderzoeken was goed, na het verwerken van de feedback op het onderzoek heb ik het onderzoek naar een beter niveau gebracht. Ik denk dat mijn niveau van research based onderzoeken al vrij goed was. En dat dit ook terug te zien was in mijn documentatie.

1.2.4: Sprint 4

Tijdens deze sprint niet heel veel voortgang geboekt, dus wel nog aan de slag geweest met het research based onderzoeken. Maar geen nieuwe ervaringen of kennis opgedaan.

1.2.5: Sprint 5

Tijdens deze sprint niet bezig geweest met context based research, voornamelijk door de fase waarin deze sprint zich bevindt.

## **1.3 Preparation for Life-Long Learning**

**Omschrijving**

Voor deze leeruitkomst is het belangrijk dat je een blik vooruit werpt. Hoe kijk jij naar je verdere carrière? Heb je al een beeld bij het afstuderen of eventueel een type bedrijf waar je later zou willen werken? Met welke onderwerpen zou je je hier mee bezig willen houden? Als je naar jezelf kijkt, ben je hier dan al goed op voorbereid of zijn er nog punten waar je je in zou kunnen ontwikkelen? Verder is het voor deze leeruitkomst belangrijk dat je je verdiept in de Sprint 5 onderwerpen. Hier kies je er eentje van, waar je individueel in sprint 5 mee aan de slag gaat.

**Ontwikkelschaal**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Beschrijving** | **Type** | **Niveau** |
| 1.1 | Sprint 1 | Individueel  Groepsproject | Orienting |
| 1.2 | Sprint 2 | Individueel  Groepsproject | Beginning |
| 1.3 | Sprint 3 | Individueel  Groepsproject | Proficient |
| 1.4 | Sprint 4 | Individueel  Groepsproject | Proficient |
| 1.5 | Sprint 5 | Individueel  Groepsproject | Proficient |

**Onderbouwing**

1.1.1: Sprint 1

Zelf ben ik bezig met het zoeken van een aansluitende stage. Deze moet het liefst aansluiten bij wat ik leuk vind (programmeren, backend). Ik steek hier samen met Bas veel tijd in omdat wij het liefst in het buitenland stage willen lopen. Ikzelf denk dat het buitenland ook beter aansluit bij mijn ambities die ik heb. Ik heb dan ook de minor pre-master Information Management gedaan en zie mijzelf niet mijn hele leven programmeren. Het liefst wil ik zo snel mogelijk doorgroeien en ik denk hiervoor een groter bedrijf nodig te hebben.

Binnen het groepsproject ben ik vooral bezig in de professionele omgeving. Deze sluit aan bij de werkomgeving waar wij later in gaan werken, zoals algemeen bekend in de ICT-wereld.

1.1.2: Sprint 2

Ook deze sprint ben ik volop bezig geweest met het zoeken van een afstudeerstage, het liefst in het buitenland. Ik heb contact met ABB (in Nederland) en HARMAN (in Duitsland). Ik heb een duidelijk beeld dat ik bij een groter bedrijf wil werken, het liefst in het buitenland. Mijn beeld over wat ik na mijn stage wil is al duidelijk, namelijk beginnen als programmeur en daarna doorgroeien binnen een bedrijf.

1.1.3: Sprint 3

Ook tijdens deze sprint ben ik druk bezig geweest met het zoeken van een bedrijf in het buitenland. Ondertussen heb ik 2 gesprekken gehad met een bedrijf in München en wacht ik af of ze een contract aanbieden. Ook ben ik voor de zekerheid een back-up stage aan het zoeken in Nederland.

1.1.4: Sprint 4

Tijdens deze sprint is het gelukt om een stage te regelen bij CGI in Duitsland, samen met Bas zal ik na de zomervakantie die kant opgaan. Ook had ik als backup-stage een stage geregeld bij ProDrive in Eindhoven. Idealiter wil ik na mijn stage daar aan de slag.

1.1.5: Sprint 5

Tijdens deze sprint vooral bezig geweest met het zoeken van een huis voor de stage in Duitsland.

**Reflectie op Voortgang**

1.2.1: Sprint 1

Ik denk dat ik al redelijk weet hoe ik mezelf later zie en hoe ik mijzelf het beste kan ontwikkelen. Echter is het wel van belang om hierin stappen te blijven maken. Hiervoor zou ik nog meer tijd in het zoeken van een passende stage kunnen steken. Maar ook een nog professionelere houding zou mij kunnen helpen.

1.2.2: Sprint 2

Ik denk dat ik goede progressie heb geboekt op dit gebied deze sprint. Ik heb uitgevonden dat mensen benaderen via LinkedIn beter werkt dan het invullen van career forums. Ik ben nog altijd even gedreven om een gepaste stage te vinden. En qua motivatie om dit semester en de stage af te ronden zit het ook goed.

1.2.3: Sprint 3

Ik denk dat ik tijdens deze sprint een matige vooruitgang heb gehad. Dit vooral omdat ik druk ben geweest met andere taken. Waardoor het zoeken van een geschikte stage aan de zijkant is komen te liggen. Voor de volgende sprint moet ik tijd zoeken om een geschikte stage te vinden, aangezien dit vrij belangrijk is voor mijn toekomst.

1.2.4: Sprint 4

Een goede voortgang, aangezien ons (Bas Cieraad en mijn) harde werken wordt beloond met een stage in het buitenland die we graag wilden. Ik denk dat mijn professionele houding op het eind van dit semester zeker goed zit. En dat ik met een goede instelling naar Duitsland ga.

1.2.5: Sprint 5

Mijn professionele houding en drive naar een goede toekomst blijft. Het is nog niet gelukt om een huis te vinden. Maar dit is een kwestie van tijd.

## **1.4 Scalable Architectures**

### 1.4.1 Practical

**Omschrijving**

Belangrijk hier is dat je de non-functional requirements goed uitwerkt. Hiermee laat je zien dat je weet wat er nodig is om een applicatie in een Enterprise context te draaien. Vervolgens laat je zien hoe je in het ontwerp en bij de implementatie hier rekening mee hebt gehouden. Wat heb je gedaan om te voldoen aan de non-functional requirements? Na oplevering dien je het resultaat van de doelstellingen ook nog te demonstreren en hier positieve feedback op te ontvangen, zodat je dit mee kan nemen in je PDR. Wat Enterprise Development betreft, laat je zien dat je goed gebruik maakt van enterprise specifieke technieken die je taal of framework aanbiedt.

**Ontwikkelschaal**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Beschrijving** | **Type** | **Niveau** |
| 1.1 | Sprint 1 | Individueel  Groepsproject | Beginning |
| 1.2 | Sprint 2 | Individueel  Groepsproject | Beginning |
| 1.3 | Sprint 3 | Individueel  Groepsproject | Proficient |
| 1.4 | Sprint 4 | Individueel  Groepsproject | Proficient |
| 1.5 | Sprint 5 | Individueel  Groepsproject | Advanced |

**Onderbouwing**

1.1.1: Sprint 1

Voor het groepsproject is al onderzoek gedaan naar de Enterprise architectuur. De vorige groep heeft dit al geïmplementeerd. Het groepsproject voldoet dus al ruim voldoende aan scalable architectuur. Hierbij is gebruik gemaakt van microservices die ook async kunnen communiceren. De verschillende platforms kunnen allemaal veel data verwerken.

Het is handig dat dit voor het groepsproject al is gedaan, hierdoor kon ik ook sneller aan de slag voor individueel. Hier gebruik ik namelijk dezelfde architectuur die voldoet aan de verschillende eisen. Deze non-functionele requirements heb ik dan ook opgesteld, vergeleken en zullen deze week worden beoordeeld.

1.1.2: Sprint 2

Binnen sprint 2 ben ik verder gegaan met het juist implementeren van de architectuur. Hierbij heb ik de gateway aangepast naar een ocelot gateway. Dit zorgt ervoor dat de applicatie ook makkelijker schaalbaar is. Verder heb ik in de backend nu ook gebruik gemaakt van het entity framework. Dit maakt een hoop processen gemakkelijker en is ook betrouwbaarder.

1.1.3: Sprint 3

Tijdens deze sprint heb ik de volledige applicatie werkend gekregen in kubernetes, kubernetes helpt ook met de schaalbaarheid en biedt een dashboard waar je alles kan overzien.

1.1.4: Sprint 4

Voor deze sprint heb ik aangetoond dat Kubernetes ook daadwerkelijk schaalbaar is. Mijn gehele applicatie en de structuur daarvan zijn schaalbaar opgebouwd. En om dit te testen heb ik K6 toegevoegd.

1.1.5: Sprint 5

Vooral bezig geweest met de afronding tijdens deze sprint. Onderzocht of de applicatie blijft runnen tijdens de loadtesten. Voor de rest door de duur van deze sprint niks gedaan.

**Reflectie op Voortgang**

1.2.1: Sprint 1

Mijn voortgang binnen scalable architectures was vrij prima omdat er dus al veel voorgedaan stond in het groepsproject. Omdat ze binnen het groepsproject al rekening hebben gehouden met de verschillende kwaliteitsattributen, kon ik deze ook vrij gemakkelijk overnemen in mijn individuele architectuur. Sjors heeft mij dan ook veel geholpen bij het beter begrijpen van de bestaande non-functionele requirements.

1.2.2: Sprint 2

Door mijn voorsprong vanuit de proftaak, kon ik deze sprint al beginnen met het finetunen van de architectuur. Hierdoor denk ik dat ik nu al mijn NFR’s kan aantonen. Het is nu van belang om volgens de richtlijnen van de programma’s te programmeren, zodat de applicatie ook daadwerkelijk schaalbaar blijft.

1.2.3: Sprint 3

Ik denk dat ik goede voortgang heb gemaakt, omdat mijn hele applicatie nu runt in Kubernetes. Dit betekent dat al mijn keuzes rondom schaalbaarheid zijn meegenomen en alles nu werkt.

1.2.4: Sprint 4

Het meeste voor dit leerdoel stond al, de gehele omgeving is zo schaalbaar mogelijk opgezet. Nu was het alleen nog de taak om te laten zien dat dit ook zo functioneerde. Ik denk dat ik met K6 hier een mooi overzicht van heb kunnen maken.

1.2.5: Sprint 5

Ik denk dat het niveau van dit leerdoel al een tijd op proficient lag. Daarom denk ik met K6 een advanced niveau te hebben behaald. Hierbij denk ik zo veel mogelijk attributen toegevoegd te hebben om een schaalbare applicatie te hebben.

### 1.4.2 Technical

Als developer, wil ik een schaalbare architectuur, zodat ik wanneer het nodig is kan upscalen, downscalen of kan switchen van instantie. Er zijn veel verschillende schaalbare architecturen. Zoals bijvoorbeeld: de N-Tier architectuur, waarin de applicatie is opgedeeld in 3 verschillende lagen: logic tier, presentation tier en de data tier. Een andere architectuur is de web-queue-worker architectuur. Hierin is de gebruiker het web en zit er een queue tussen de worker en het web, in de queue worden berichten opgeslagen zodat ze snel kunnen worden gebruikt. Daarnaast heb je ook nog het event-driven architectuur. Hierin heb je verschillende publishers, die een evenement aan de consumer doorgeven. Dit doen ze via een event manager (bijvoorbeeld een Kafka cluster). Tot slot heb je een microservice architectuur, waar binnen dit project voor is gekozen. Binnen het SAD staan de verschillende diagrammen die toelichten hoe deze architectuur in elkaar zit.

**Analyse en design**

* Binnen proftaak onderzoek gedaan naar de verschillende architectuur patterns die er zijn.
* Binnen het SAD zijn verschillende keuzes gemaakt die de schaalbaarheid van de applicatie en de architectuur waarborgen.

**Implementatie**

* Zoals in het SAD en de implementatie (Github) te zien, heb ik de microservice architectuur juist toegepast. Hierbij wordt schaalbaarheid gewaarborgd.
* Zoals in de Github repo van de API-Gateway te zien, heb ik Ocelot toegepast. Dit zorgt voor een (makkelijk) schaalbare API-Gateway.
* Zowel in Ocelot als in Kubernetes zit loadbalancing, hierbij heb ik op Kubernetes voor elke instantie 2 replicas aangemaakt. Hierdoor kan Kubernetes switchen wanneer dit nodig is. Kubernetes kent ook een dashboard waar je overzichtelijk kan zien wat de instanties doen.

**Scenario’s**

|  |  |
| --- | --- |
| **Scenario** | **Pretter** |
| Het verkeer naar de UserService wordt twee keer zo groot. | Kubernetes zal de load van de service balancen en wanneer nodig verkeer over een andere replica laten gaan. |
| De eerste replica van de TweetService is niet meer beschikbaar. | Kubernetes zal de load verschuiven naar de andere beschikbare replica. |

#### Toelichting

**Messaging**

Messaging kan erg handig zijn bij schaalbare architecturen. Wanneer er bijvoorbeeld een grote hoeveelheid aan verkeer langs een service gaat. Dan kan het van belang zijn dat je berichten worden opgeslagen. Dit kan zijn wanneer een service niet meer beschikbaar is. Of als er bijvoorbeeld moet worden geswitcht naar een andere instantie van de service.

RabbitMQ is een open source message broker die berichten kan managen. Het kan messages naar verschillende services tegelijk sturen, messages asynchroon versturen en eventueel ook tijdelijk opslaan.

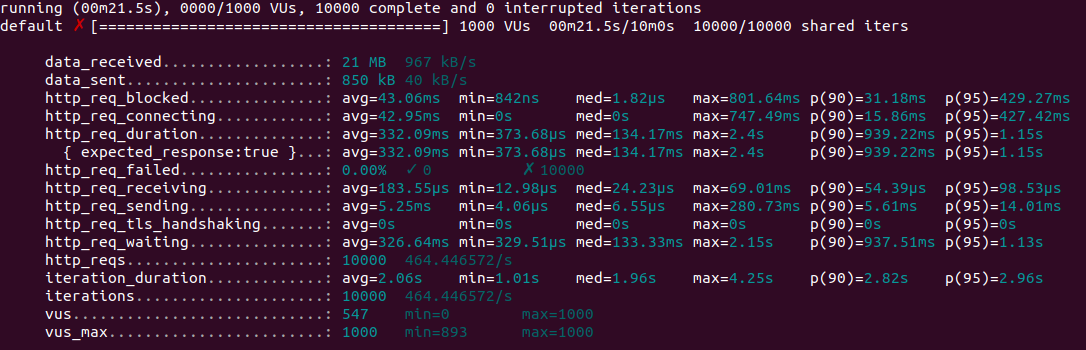
**Keuzes**

Er is gekozen voor een microservice architectuur. Waar alle services, de API-Gateway en de frontend apart worden geployed als containers. Deze containers worden vervolgens in Kubernetes gebruikt. Er bestaan van elke container meerdere instanties, zodat er geschakeld kan worden. Kubernetes regelt ook de load balancing hier. En dit is zichtbaar op het dashboard.

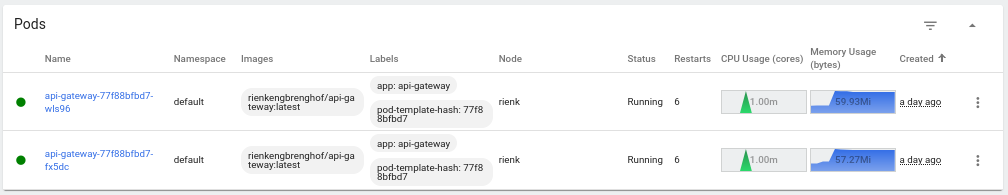
**K6**

K6 is een load testing programma dat load op het cluster kan uitoefenen. Om te zien hoe het programma reageert.

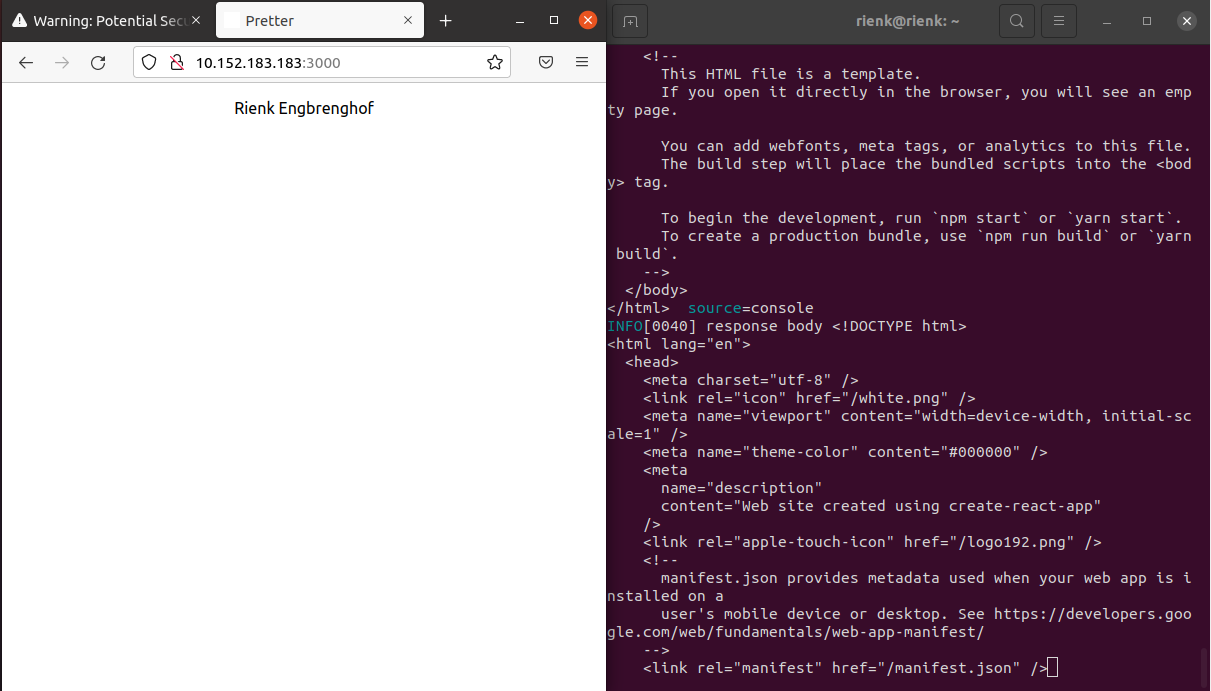
Onderstaande foto laat zien dat K6 10000 requests naar de frontend doet (een van de testen) en dat hij dit binnen 21.5 seconden kan doen.



Vervolgens is in Kubernetes kan je zien dat het verkeer over beide instanties een piek ervaart. Dus dat de load evenredig wordt verdeeld over de verschillende instanties van de API-Gateway.



**Ook tijdens de loadtest lukt het om in de applicatie in te loggen.**

****

## **1.5 Development and Operations (DevOps)**

### 1.5.1 Practical

**Omschrijving**

Let op dat Dev(Sec)Ops niet alleen gaat over het opzetten van een CI/CD pipeline. Denk terug aan de case study: toen hebben we het er over gehad dat ook op andere manieren (zelfs los van het inzetten van technologie) ervoor kan zorgen dat een project soepeler verloopt. Denk aan het SDLC proces, verwachtingsmanagement, betere afspraken (bv. via SLA of een Change Management procedure) en communicatie, etc.

**Ontwikkelschaal**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Beschrijving** | **Type** | **Niveau** |
| 1.1 | Sprint 1 | Individueel  Groepsproject | Beginning |
| 1.2 | Sprint 2 | Individueel  Groepsproject | Beginning |
| 1.3 | Sprint 3 | Individueel  Groepsproject | Proficient |
| 1.4 | Sprint 4 | Individueel  Groepsproject | Proficient |
| 1.5 | Sprint 5 | Individueel  Groepsproject | Proficient |

**Onderbouwing**

1.1.1: Sprint 1

Ook op gebied van DevOps heeft het groepsproject zo goed als alles al. Het groepsproject heeft automatisch testen, code coverage en deployment naar docker. Door middel van het scrumboard is het makkelijk om prioriteiten omtrent requirements aan te passen en/of hier nieuwe requirements neer te zetten. Het project is ook opgedeeld in delen die allen CI/CD bevatten.

Voor mijn individuele project ben ik ook al bezig geweest met het opzetten van een DevOps omgeving. Hierin heb ik dezelfde opdeling van project als in het groepsproject en heb ik CI/CD opgezet op mijn repositories.

1.1.2: Sprint 2

Binnen de proftaak hebben we betere afspraken gemaakt over het indelen van uren. We hebben er voor gekozen om geen gebruik van punten te maken, maar van uren. Zodat we makkelijk kunnen terugrekenen hoeveel uur iedereen bezig is geweest. Binnen mijn individueel heb ik geen verwachtingsmanagement. Ik heb wel een lijst met eisen, maar deze hebben geen prioriteit gebonden gekregen.

1.1.3: Sprint 3

Mijn DevOps omgeving bevat nu genoeg om een volledige applicatie te hosten. Hierbij overweegt het ook de security aspecten. Hierbij wordt het systeem automatisch gepusht naar Kubernetes. Door het gebruik van Git is het vrij makkelijk om te veranderen van prioriteit.

1.1.4: Sprint 4

Voor mijn pipeline heb ik gekeken naar het implementeren van OWASP ZAP, echter bleek dat hiervoor de hele omgeving moest runnen in GitHub. Dit is echter niet hoe mijn omgeving in elkaar zit. Daarom heb ik OWASP ZAP opgezet in mijn VM omgeving. Hier werkt dit nu ook en is het mogelijk om alle OWASP onveiligheden te zien op het dashboard.

Ook heb ik met K6 dus een nieuw aspect van testen toegevoegd aan de applicatie.

1.1.5: Sprint 5

Mijn niveau binnen deze sprint lag al goed, mijn gehele omgeving werkte. Voor het project heb ik nu loadtesten, code quality check en security testen. Voor een verdere implementatie zou ik de unit-testen en integratie testen uitwerken.

**Reflectie op Voortgang**

1.2.1: Sprint 1

Mijn voortgang voor dit onderwerp was ook goed, dit heeft wederom te maken met dat ik het groepsproject als voorbeeld kon nemen. Binnen het groepsproject zijn er duidelijke afspraken/goede communicatie vooralsnog.

1.2.2: Sprint 2

Binnen de proftaak hebben we goede voortgang gemaakt over het indelen van de verwachtingen en afspraken. Ik zou in mijn individueel ook nog meer prioriteit en verwachtingen aan taken kunnen toevoegen. Deze zou ik ook kunnen bespreken met de docenten. Aangezien er gezegd is dat de focus op de proftaak ligt (technisch), is het belangrijk om te evalueren waar ik voor individueel de focus moet leggen.

1.2.3: Sprint 3

Mijn voortgang op gebied van DevOps is prima, de omgeving was vorige sprint al op redelijk niveau, maar tijdens deze sprint is ook echt gebleken dat mijn implementatie werkt en effectief is.

1.2.4: Sprint 4

Goede voortgang op dit gebied gemaakt, ik denk dat het mooier was geweest als OWASP ZAP in mijn pipeline was komen te zitten. Maar daarvoor had ik de pipeline compleet anders moeten inrichten. Ook het load testen is een nieuw aspect dat ik heb geleerd en begrijp.

1.2.5: Sprint 5

Tijdens deze sprint door de korte duur weinig kunnen doen. Heb de testen iets verder uitgewerkt. Ik ben trots op de gehele applicatie en de bijbehorende DevSecOps.

### 1.5.2 Technical

Als developer, wil ik een goede DevOps omgeving, zodat ik het agile en automatiseringsproces kan ondersteunen. En om een zo kortmogelijke release tijd en hoge software kwaliteit te garanderen.

**Analyse en design**

* Het proces is ingericht volgens OTAP (ontwikkeling, test, acceptatie en automatisering). Hierbij wordt eerst de code ontwikkeld, vervolgens geautomatiseerd getest en in geval van acceptatie gedeployed.

**Implementatie**

* De DevOps omgeving is opgezet in GitHub. Hier wordt de code en de taken opgeslagen. Ook de CI/CD pipeline bevindt zich in GitHub. De pipeline test eerst de code, vervolgens controleer SonarQube de kwaliteit en eventuele bugs. En hierna wordt de code eventueel gepusht naar Docker.
* Alle delen van de applicatie zijn individueel te deployen. Ze worden ook allemaal individueel als Docker images gepusht. In de virtuele omgeving van Kubernetes worden deze images automatisch gepulled.

**Scenario’s**

|  |  |
| --- | --- |
| **Scenario** | **Pretter** |
| Er is een visuele aanpassing gedaan op de frontend. | Wanneer de push wordt ontvangen. Zullen allereerst de test worden gerund, vervolgens zal sonarcloud de code scannen en tot slot zal bij goedkeuring er een docker image gebouwd worden. Wanneer dit gebeurd dan pullt Kubernetes deze en heeft de deployment de nieuwste versie. |
| Een test van de frontend push faalt. | De gefaalde test zal ervoor zorgen dat de pipeline faalt, hierdoor zal er eerst moeten worden voldaan aan deze test. |

#### Toelichting

**Management types**

Change management, change management gaat over de aanpak wanneer er een transitie of transformatie plaatsvind qua een organisatie haar doelen, processen of technologieën. Het doel van change management is om veranderingen effectief en gecontroleerd te laten verlopen. Ook helpt het personeel bij het accepteren van veranderingen. Agile support deze change omgeving. Omdat changes kunnen worden gescheduled als stories of features.

Risk management, risk management gaat over het identificeren, analyseren en controleren van risico’s voor een organisatie of systeem. Deze risico’s kunnen van alles zijn, bijvoorbeeld financiële risico’s, technologische risico’s, strategische risico’s of bijvoorbeeld ongelukken of rampen. Hierbij is het van belang om constant te monitoren, zodat je snel kan ingrijpen wanneer er een risico plaatsvind.

Security management, dit is een proces die moet zorgen dat er meer grip wordt verkregen op veiligheidsrisico’s. Het is een systematische kijk op de beveiliging van applicaties/organisaties. Hierin zit risk management ook verwerkt. Er wordt een risico analyse uitgevoerd, vervolgens vindt risk management plaats. Daarna worden er sancties en policies opgesteld en vervolgens worden het systeem geaudit.

Release management, release management is het managen en het toewerken naar een release. Het begint met plannen, bouwen, reviewen, testen. Vervolgens wordt het gedeployed, wanneer alles goedgekeurd is. Deze management vorm omvat alle stappen van development tot deployment.

Verwachtingsmanagement, hierbij doe je een poging om de kloof te dichten tussen wat een ander verwacht en wat er geleverd kan worden. Dus je stelt eisen aan de verwachtingen en koppelt dit terug aan belanghebbenden. Hiermee kan je belanghebbenden meer tevreden houden.

**Testen**

Er zijn verschillende soorten testen die de software quality kunnen waarborgen: Unit tests, Integration tests, UI tests, End-to-end tests en Acceptance tests zijn hier voorbeelden van. Binnen een applicatie is het van belang om zo veel mogelijk diversiteit qua testen toe te voegen. Door beknopte tijd binnen het project ben ik tot dusver alleen tot unittesten gekomen. Het is echter van belang om verder te kijken.

Integration tests gaan 1 stap verder dan Unit testen, binnen mijn applicatie zouden deze mogelijk zijn met bijvoorbeeld Enzyme. Enzyme rendert components in memory en hierdoor is het mogelijk om meer te testen dan alleen 1 klasse.

UI testen kan binnen react met react zelf, het biedt de mogelijkheid om de UI te testen. Voornamelijk of dat de UI gereageerd zoals beloofd.

End to end testen kan binnen een applicatie zoals de mijne met bijvoorbeeld playwright. Hierbij wordt verwacht dat de hele applicatie testbaar is. Hiervoor moet dus de hele applicatie gebouwd worden. Vervolgens kan er worden gekeken of de applicatie handelt zoals hij moet handelen.

Tot slot kan de acceptance test worden uitgevoerd, hierin controleer je of het systeem alles doet waar hij voor ontworpen is. Dit kan ook worden gezien als het aftekenen van de use-cases en test-cases.

## **1.6 Cloud Services**

### 1.6.1 Practical

**Omschrijving**

Ook hier geldt dat de leeruitkomst niet alleen over deployment gaat. Het uitrollen van een applicatie naar een cloudplatform (bv. via Kubernetes/Rancher) is mooi, maar kijk ook eens naar andere managed cloud services. Wat biedt een cloud provider allemaal aan? Wat zou je voor je project kunnen gebruiken en in hoeverre zou het een bijdrage kunnen leveren aan het makkelijker behalen van je non-functional requirements doelstellingen? Bedenk dat met het slim integreren van bestaande cloud services je bepaalde zaken waarschijnlijk sneller voor elkaar krijgt dan wanneer je dit zelf zou programmeren.

**Ontwikkelschaal**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Beschrijving** | **Type** | **Niveau** |
| 1.1 | Sprint 1 | Individueel  Groepsproject | Beginning |
| 1.2 | Sprint 2 | Individueel  Groepsproject | Beginning |
| 1.3 | Sprint 3 | Individueel  Groepsproject | Proficient |
| 1.4 | Sprint 4 | Individueel  Groepsproject | Proficient |
| 1.5 | Sprint 5 | Individueel  Groepsproject | Proficient |

**Onderbouwing**

1.1.1: Sprint 1

Het groepsproject maakt al gebruik van verschillende cloud services en bevat ook een onderzoek hiernaar. Het deployt bijvoorbeeld een docker container en runt het code coverage rapport via sonarcloud. Door een gesprek met Sjors begrijp ik nu de keuzes van het vorige groepje omtrent het gebruiken van cloud services op sommige plekken binnen het project. Wanneer de microservices apart gehost worden, kan je er ook voor kiezen om voor sommige meer cloud resources in te schakelen dan voor andere.

Voor mijn individuele project heb ik ook sonarcloud toegepast en wil ik dus ook gaan werken met containers.

1.1.2: Sprint 2

Tijdens sprint 2 heb ik eigenlijk alleen gekeken naar kubernetes en het geprobeerd toe te passen. Hierbij ben ik niet zo zeer opgeschoten. Maar dit maakt ook niet uit want het had tijdens deze sprint geen prioriteit.

1.1.3: Sprint 3

Alles draait op het cloud platform kubernetes. Hierin is overzichtelijk te zien wat de resources zijn en wat er gebruikt wordt. Want ik heb kubernetes opgezet met een mooi dashboard. Alles runt lokaal maar het haalt de docker images online op.

1.1.4: Sprint 4

Ook voor cloudservices was ik al een heel end, daarom heb ik tijdens deze sprint vooral gekeken naar monitoring en messaging in mijn SAD document. En een vergelijking gemaakt op basis van technieken en prijzen.

1.1.5: Sprint 5

Door de korte duur van deze sprint weinig kunnen doen voor dit leerdoel. Echter heb ik al veel werk verricht voor dit leerdoel. Ook heb ik nog omschreven wat mijn keuzes zouden zijn voor dit leerdoel wanneer ik het verder zou implementeren.

**Reflectie op Voortgang**

1.2.1: Sprint 1

Mijn voortgang op het gebied van cloud services is prima. In vorige semesters hebben wij ook gewerkt met verscheidende cloud services. Wederom heeft het voorbeeld van het groepsproject mij geholpen met het toepassen van cloud services.

1.2.2: Sprint 2

Ik heb ervaren dat het nog geen probleem is om de applicatie niet te deployen. Hierbij hoor ik om mij heen dat het andere wel al gelukt is, en deze personen kunnen mij wellicht hiermee dan ook verder helpen. Dit zou een hoop tijd schelen.

1.2.3: Sprint 3

Tijdens deze sprint heb ik hele goede voortgang geboekt. Het heeft me lang gekost om Kubernetes volledig uit te zoeken, maar nu werkt alles zoals ik het bedoeld heb. Hier ben ik zeer tevreden over.

1.2.4: Sprint 4

Tijdens deze sprint heeft dit leerdoel niet de meeste aandacht gekregen, vooral omdat ik al wist dat ik geen service zou kunnen gaan hosten in de cloud. Wel heb ik volgens het research based een onderzoekje gedaan naar de verschillende mogelijkheden binnen cloudservices.

1.2.5: Sprint 5

Over het algemeen een goede progressie geboekt voor dit leerdoel. Het is duidelijk wat de voor en nadelen zijn van het hosten van een service. En voor mij zijn de mogelijkheden op dit gebied ook een stuk helderder.

### 1.6.2 Technical

Als developer, wil ik services in de cloud hosten, zodat ik de voordelen van cloudservices (schaalbaarheid, eenvoudigheid, betrouwbaarheid en veiligheid) kan benutten.

**Analyse en design**

* In het SAD zijn verschillende keuzes gemaakt over waarom sommige onderdelen wel of niet in de cloud gerund worden. Zo kon sonarcloud bijvoorbeeld ook lokaal gerund worden.

**Implementatie**

* Voor het controleren van de kwaliteit van de code wordt gebruik gemaakt van de sonarcloud service. Hierbij ontvangt de pipeline een report en indien er iets niet correct is wordt de pipeline gecanceld.
* Voor development wordt gebruik gemaakt van een online database. Alle images worden gedeployed naar Kubernetes.
* Er wordt ook gebruik gemaakt van Docker hub, hierbij worden mijn Docker images naar de cloudservice van Docker gepusht. Vervolgens zijn ze online beschikbaar voor Kubernetes.

**Scenario’s**

|  |  |
| --- | --- |
| **Scenario** | **Pretter** |
| Sonarqube heeft problemen door een windows update. | Mijn applicatie heeft hier geen last van omdat het gebruik maakt van de cloud service. |
| Er is erg veel dataverkeer. | Door het gebruik van een online database is er voldoende dataverkeer mogelijk. |
| Het login endpoint wordt erg veel gebruikt. | De applicatie laat het verkeer over de verschillende instanties gaan, en schaalt het CPU gebruik op. |

#### Toelichting

**Voordelen cloud services**

Het naar de cloud brengen van je services brengt voordelen met zich mee, voorbeelden hiervan zijn: het vermindert je eigen IT kosten, omdat je het niet meer zelf hoeft te hosten. Schaalbaarheid, omdat het gehost wordt bij een externe, moeten zij up of downscalen wanneer het nodig is. Het overstappen op cloud services dwingt ook om schaalbaar op te bouwen.

Het is een veilige omgeving waarin alles wordt opgeslagen. Hierdoor hoef je zelf niet meer aan de slag met het opzetten van een veilige omgeving. En komt je service minder vaak in gevaar.

Automatische updates, een cloud provider zal automatisch updates verschaffen, dit helpt ook met de veiligheid van jouw applicatie.

**Nadelen cloud services**

Cloud services brengen ook een aantal nadelen met zich mee, zoals dat je bijvoorbeeld afhankelijk wordt van de connectie naar de cloud services toe. Ook wordt je in het algemeen afhankelijk van de cloud provider. Zo kan er bijvoorbeeld daar een risico plaatsvinden waardoor je service niet meer beschikbaar is.

Een laatste nadeel zou kunnen zijn dat je kiest voor de verkeerde cloud provider. Waardoor je vast kan komen te zitten aan een langdradig contract. Of waardoor je service een langere periode niet meer beschikbaar is.

**Mogelijkheden**

Er zijn verschillende mogelijkheden om mijn services naar de cloud te deployen. Onder dit kopje zullen de verschillende cloudservices worden besproken.

**AWS**

|  |  |
| --- | --- |
| Is integreerbaar met mijn applicatie (ASP.NET, REACT en MSSQL). | JA |
| Staat bekend als veilig. | JA |
| Goede schaalbaarheid. | JA |
| Goede keuze in geval van 100.000 gebruikers. | NEE |
| Prijs in geval van 100.000 gebruikers. | +-10k |

Beschikbare monitoring tool voor AWS

AWS heeft een ingebouwde monitoring tool die CloudWatch heet. Het programma haalt metrics op om op basis daarvan statistiek te genereren. Hierdoor kan je verschillende patronen herkennen. Het is ook mogelijk om alarmen te zetten die services stoppen of starten.

**Interserver**

|  |  |
| --- | --- |
| Is integreerbaar met mijn applicatie (ASP.NET, REACT en MSSQL). | JA |
| Staat bekend als veilig. | JA |
| Goede schaalbaarheid. | JA |
| Goede keuze in geval van 100.000 gebruikers. | NEE |
| Prijs in geval van 100.000 gebruikers. | +-3k |

Beschikbare monitoring tool voor Interserver

Binnen Interserver zijn verschillende mogelijkheden. Een van deze mogelijkheden is Nagios. Dit is een extern programma dat de cloud kan monitoren in een dashboard. Nagios is ook mogelijk om op te zetten binnen mijn productie omgeving.

**Microsoft Azure**

|  |  |
| --- | --- |
| Is integreerbaar met mijn applicatie (ASP.NET, REACT en MSSQL). | JA |
| Staat bekend als veilig. | JA |
| Goede schaalbaarheid. | JA |
| Goede keuze in geval van 100.000 gebruikers. | JA |
| Prijs in geval van 100.000 gebruikers. | +-6k |

Beschikbare monitoring tool voor Azure

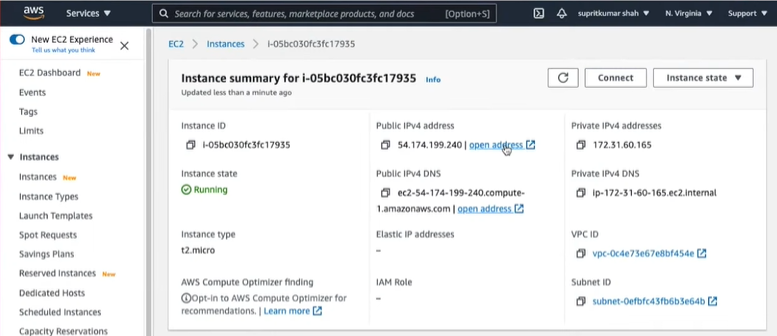
De monitoring tool binnen Azure heet Azure Monitor. Van de bovengenoemde monitoring tools is dit de meest uitgebreide tool. Je kan van alle services de resources zien, hoe het verkeer verloopt en zien waar het verkeer vandaan komt. Deze tool zorgt ervoor dat er een geheel nieuwe wereld opengaat, je krijgt toegang tot nieuwe functionaliteiten als: statistische inzichten, visualisaties in diagrammen, analyseren van logs, reageren op alarmen en het integreren van andere applicaties.

**Messaging**

Binnen al deze applicaties is het mogelijk om messaging via RabbitMQ op te zetten, deze service zou dan ook moeten worden gehost. Er zijn geen extra kosten voor het gebruik van RabbitMQ in een omgeving met 100.000 gebruikers, zolang je het dan op je eigen services host.

**Verdere implementatie**

Om cloudservices verder aan te tonen zou ik 1 service daadwerkelijk in de cloud kunnen hosten.



Hierbij zou ik mijn docker container van bijvoorbeeld de userservice kunnen deployen naar AWS. Hierbij is het mogelijk om de API te exposen en dus de UserService in de cloud aan te spreken. De rest van het cluster zou dan gebruik maken van deze cloudservice.

## **1.7 Security by Design**

### 1.7.1 Practical

**Omschrijving**

Het onderwerp Security by Design is veel breder dan alleen authenticatie/autoristatie. Laat zien hoe je in de ontwerpfase al rekening met security hebt gehouden, door bijvoorbeeld het opstellen van misuse cases of een risico analyse. Laat zien wat je met deze informatie hebt gedaan in de opzet van je architectuur. Als je de OWASP Top 10 aanhaalt, laat dan goed zien welke maatregelen je daarvoor hebt ingetekend in je ontwerp en welke richtlijnen je voor de implementatie van het project hebt opgesteld. Ook hier geldt dat je op een manier zal moeten aantonen dat je uiteindelijk aan de OWASP TOP 10 voldoet. Wellicht is hier ook tooling voor die je in je CI/CD pipeline kan hangen om dit aan te tonen.

**Ontwikkelschaal**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Beschrijving** | **Type** | **Niveau** |
| 1.1 | Sprint 1 | Individueel  Groepsproject | Orienting |
| 1.2 | Sprint 2 | Individueel  Groepsproject | Beginning |
| 1.3 | Sprint 3 | Individueel  Groepsproject | Proficient |
| 1.4 | Sprint 4 | Individueel  Groepsproject | Proficient |
| 1.5 | Sprint 5 | Individueel  Groepsproject | Proficient |

**Onderbouwing**

1.1.1: Sprint 1

Op het gebied van security by design ben ik nog oriënterend bezig. Ik ben mij nog aan het verdiepen in de verschillende toepassingen die de security verhogen. Echter is bij het groepsproject dit wel al toegepast en zijn er verschillende keuzes gemaakt die de security verhogen. In mijn individuele project zal ik een hoofdstuk toevoegen waar ik dan alle punten van de OWASP Top 10 zal toevoegen.

Omdat ik het groepsproject gebruik als voorbeeld, maak ik deze keuzes ook. Hierbij moet je denken aan keuzes voor veilige frameworks, protocollen en opslag.

1.1.2: Sprint 2

Naar aanleiding van de meeting over security, ben ik gaan overwegen of mijn applicatie de juiste opzet heeft. Hierbij heb ik in het individuele stuk een risico analyse binnen mijn testfase gemaakt. Zodat de risico’s van eventuele kwaaltjes overzien kunnen worden.

1.1.3: Sprint 3

Tijdens deze sprint heb ik veel security design keuzes overwogen. Ik heb een volledig document opgesteld waarin alle OWASP top 10 punten staan, en wat ik per punt heb gedaan aan security.

1.1.4: Sprint 4

Tijdens deze sprint heb ik dus OWASP ZAP toegepast en laten zien in mijn applicatie. Ook heb ik Oauth in mijn applicatie ingebouwd, hiervoor moest ik de frontend volledig op HTTPS laten runnen, maar ook de API-Gateway. Hier ben ik nog mee bezig, vandaar dat Oauth nog niet in mijn opgeleverde applicatie zit. In development werkt het wel, in productie nog niet.

1.1.5: Sprint 5

Voor dit leerdoel heb ik in de voorafgaande sprints vrij veel gedaan. Door de duur van deze sprint heb ik alleen de testen iets verder uitgewerkt.

**Reflectie op Voortgang**

1.2.1: Sprint 1

Mijn voortgang op dit gebied is nog niet optimaal. Ik dien mij meer te verdiepen in hoe ik veiligheid kan garanderen door middel van de juiste designkeuzes. Wanneer ik mij hierin meer heb verdiept dan kan ik dit ook toepassen in mijn eigen project en eventueel het groepsproject controleren op onveiligheden.

1.2.2: Sprint 2

Op gebied van security kan ik nog meer grond winnen, vooral door me nog meer te verdiepen in hoe de architectuur en techniek (React and C#) het beste geïmplementeerd worden.

1.2.3: Sprint 3

Ik denk dat ik deze sprint goede progressie heb gemaakt op gebied van security, ik heb veel punten overwogen. En weet waar eventuele risico’s in mijn applicatie zouden kunnen liggen. Hierdoor is het voor mij ook duidelijker dat het belangrijk is om security vanaf het begin te overwegen.

1.2.4: Sprint 4

Goede voortgang voor dit leerdoel, heb naar veel invalshoeken gekeken en naar mogelijkheden om de security te verhogen. Ook heb ik veel nagedacht over implementaties waardoor de security verhoogd kan worden. Het is duidelijk voor mij dat dit vanaf het begin van de applicatie benodigd is.

1.2.5: Sprint 5

Voor dit leerdoel heb ik in het algemeen een erg goede voortgang denk ik. Tijdens deze sprint heb ik de puntjes op de i gezet.

### 1.7.2 Technical

Als developer, wil ik security implementeren vanaf het design stadium. Zodat ik eventuele beveiligingsrisico’s kan voorkomen en niet op het einde nog alles voor security hoef te doen.

**Analyse en design**

* Gebruikte frameworks geanalyseerd en hiervan de best practices gebruikt.
* OWASP implementeren.
* Testplan opgezet in documentatie en automatisch testen toegevoegd in de pipeline. Ook test sonarcloud op eventuele veiligheidsgebreken.

**Implementatie**

* In het juist opzetten van de architectuur heb ik voor veiligheidsredenen en best practices ervoor gekozen om de API-Gateway als enige punt van aanspraak te houden. Hierdoor is de attack surface een stuk kleiner.
* In de applicatie is authenticatie toegevoegd. Hierdoor kunnen onbevoegden niet alles doen.
* Binnen de applicatie worden errors juist afgevangen. Hierbij wordt er gezorgd dat er veilig gefaald kan worden.
* Sonarcloud toont beveiligingsrisico’s.
* Wachtwoorden zijn encrypted.

**Scenario’s**

|  |  |
| --- | --- |
| **Scenario** | **Pretter** |
| Er wordt van buitenaf een endpoint van de UserService aangesproken. | Endpoints van de services zijn alleen beschikbaar en bereikbaar voor de API-Gateway. |
| Een methode krijgt een error omdat er geen verbinding met de database is. | Errors worden afgevangen, de applicatie zal de data niet kunnen ophalen maar zal niet crashen. |

#### Toelichting

**GDPR**

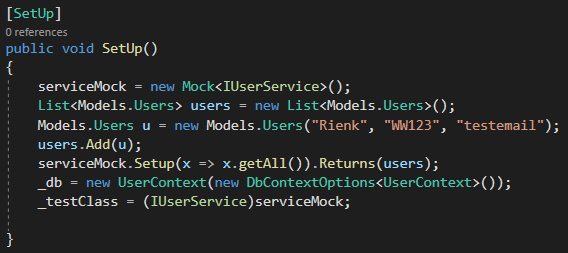
Hoe lang een bedrijf persoonsgegevens mag opslaan op basis van de AVG, hangt af van waarvoor de gegevens gebruikt worden. Wel is het zo dat wanneer de data niet meer gebruikt wordt, het na een bepaalde tijd verwijderd dient te worden.

Bij een datalek waar het om persoonsgegevens gaat, is het belangrijk dat risk management in acht wordt gehouden. Een eventuele lek zou kunnen zorgen voor enorme boetes voor een bedrijf. Daarom is het van belang dat de data veilig wordt opgeslagen, idealiter worden er zo min mogelijk persoonsgegevens opgeslagen.

Binnen mijn applicatie zou ik dit kunnen verbeteren door OAuth2 of OpenID toe te voegen aan de applicatie. Dit zorgt ervoor dat gebruikers inloggen op een SSO van een andere instantie (bijvoorbeeld Google of Fontys). Hierdoor hoef ik zelf niet de persoonsgegevens op te slaan in mijn database. Dit ontneemt een hoop risico’s.

**Verdere implementatie**

Voor verdere implementatie zou ik mijn unittesten en integratietesten verder uitwerken. Dit zou ik doen met mockdata gemaakt door Moq.



Hierbij wordt de nep data eenmalig opgezet in de SetUp. Bovenstaande afbeelding laat mijn setup zien. Vervolgens maken alle testklasses gebruik van deze setup.

## **1.8 Distributed Data**

### 1.8.1 Practical

**Omschrijving**

Deze leeruitkomst kan je vanuit verschillende invalshoeken benaderen. Als je voor een microservice architectuur hebt gekozen, dan ligt het voor de hand om iedere service zijn eigen database te geven. Hiermee wordt de data al gedistribueerd binnen je omgeving. Je kan ook kijken naar andere type datastores (er is meer in het leven dan alleen relationele databases). Wat zijn de data requirements die gelden voor jouw project en welke data stores kunnen helpen bij het behalen van deze requirements? Hoe schaalbaar is je ontwerp eigenlijk? Wat nu als het datavolume snel toeneemt, zit je dan snel aan de limieten van je database, of is deze native schaalbaar? Kijk hierbij ook eens naar wat de cloud providers hier voor aanbieden.

**Ontwikkelschaal**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Beschrijving** | **Type** | **Niveau** |
| 1.1 | Sprint 1 | Individueel  Groepsproject | Orienting |
| 1.2 | Sprint 2 | Individueel  Groepsproject | Beginning |
| 1.3 | Sprint 3 | Individueel  Groepsproject | Beginning |
| 1.4 | Sprint 4 | Individueel  Groepsproject | Proficient |
| 1.5 | Sprint 5 | Individueel  Groepsproject | Proficient |

**Onderbouwing**

1.1.1: Sprint 1

Bij het groepsproject is al gebruik gemaakt van een geschikte architectuur waar data juist wordt gedistribueerd. Ik moet mijzelf echter nog meer verdiepen in de vereisten die komen kijken bij het juist distribueren.

In het groepsproject wordt de data niet centraal opgeslagen. Ik zou dit wederom als voorbeeld kunnen gebruiken maar ik denk dat ik mijn data wil gaan opdelen in verschillende databases die online worden gehost.

1.1.2: Sprint 2

Het opzetten van een gedistribueerde data omgeving is gelukt. Hierbij zou eventueel nog gekeken kunnen worden naar het dubbel opzetten van de data stores. Dit kwam ook naar voren bij het ethiek spel dat wij hebben gespeeld tijdens deze sprint. Hierbij kwamen ook vragen naar voren van hoe schaalbaar de datastores zijn. En aangezien deze in mijn omgeving online gehost worden, door een provider, zit dit hier wel goed.

1.1.3: Sprint 3

Tijdens deze sprint heb ik ervoor gezorgd dat er een data omgeving is voor zowel development als voor productie. Dit is nu ook gelukt en in Kubernetes wordt gebruik gemaakt van een andere datastore. Ook deze datastore heeft meerdere databases, zodat de data juist wordt opgedeeld.

1.1.4: Sprint 4

Voor dit onderwerp heb ik nog gekeken naar de verschillende manieren waarop data gescyned kan worden. Dit heb ik ook verder uitgewerkt in het SAD. De data is verdeeld over meerdere databases, alleen nog niet met datasyncen.

1.1.5: Sprint 5

Door de korte sprint weinig progressie. Omschreven hoe ik het zou doen als ik het verder zou implementeren. Gekeken naar de verschillende oplossingen die er zijn. Hierbij rekening gehouden met datasync.

**Reflectie op Voortgang**

1.2.1: Sprint 1

Mijn voortgang op dit gebied is wat ik er van had verwacht, zoals gezegd moet ik me er meer in verdiepen maar het heeft momenteel geen urgentie. Ik moet feedback vragen of het juist is als ik het opdeel in meerdere databases, dan kan ik dit toepassen en de juiste progressie boeken.

Over dit onderwerp heb ik ook met Mark gesproken, voornamelijk over de dataopslag. Hierbij zei hij dat mijn keuzes goed zijn (per service een database). Dit ga ik dan ook toepassen.

1.2.2: Sprint 2

Op het gebied van gedistribueerde data is mijn voortgang goed. De toepassing in mijn individuele project klopt. En ik heb deze sprint naar de ethische vraagstukken gekeken. Vooral binnen proftaak maar deze kwesties zijn ook zeer toepasbaar op mijn individuele project.

1.2.3: Sprint 3

Ook op basis van dit punt is mijn progressie goed, de gekozen data structuur sluit goed aan bij microservices. En in de Kubernetes omgeving is het makkelijk om de data op te schalen wanneer er meer data verkeer zou plaatsvinden.

1.2.4: Sprint 4

Idealiter had ik op dit gebied nog datasyncen willen toepassen, zodat mijn productie omgeving nog realistischer was geweest. Echter heb ik op deze manier ook de data goed gedistribueerd.

1.2.5: Sprint 5

Over het algemeen goede progressie gemaakt voor dit leerdoel. Ook veel nieuwe inzichten opgedaan omtrent het syncen van data en het werken met meerdere plekken waar data is opgeslagen.

### 1.8.2 Technical

Als developer, wil ik mijn data distribueren, zodat ik de grote hoeveelheid aan data en de gevoeligheid ervan kan waarborgen. Ook de beschikbaarheid van de data is hierbij een belangrijke factor.

**Analyse en design**

* In het SAD is te zien dat er een opdeling van data plaatsvind. Hierbij is ook overwogen of er gebruik moest worden gemaakt van meerdere databases. Binnen de productie is gebruik gemaakt van meerdere replica’s van de database.
* Analyse over de data regulaties waar normaliter aan voldaan moet worden gedaan.

**Implementatie**

* In productie is de data opgedeeld in een User en Tweet database. Hierbij worden in Kubernetes meerdere instanties van de database aangemaakt. Dit ook voor beschikbaarheid en schaalbaarheid.
* De data is encrypted en volgens de geanalyseerde richtlijnen opgeslagen.

**Scenario’s**

|  |  |
| --- | --- |
| **Scenario** | **Pretter** |
| Er gaat een grote hoeveelheid aan data naar de UserDB op de productie. | Dit zou geen probleem moeten zijn, aangezien er meerdere replica’s van de database zijn. Kubernetes kan verkeer hiernaar omzetten. |
| Er vindt een datalek plaats in de UserDB. | Wanneer alles volgens richtlijnen is ingericht, zal er tijdens een lek ook volgens de richtlijnen gehandeld moeten worden. Maar risico´s zijn al kleiner omdat de richtlijnen in acht zijn genomen. |

#### Toelichting

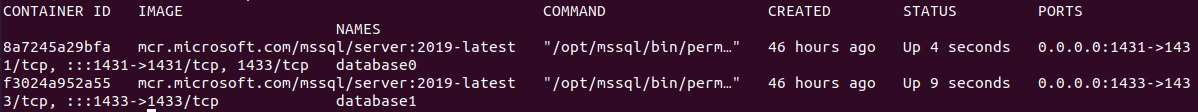
**Data syncing**

Met RabbitMQ is het mogelijk om data naar 2 databases tegelijk te sturen. Hierdoor zou je technisch gezien dezelfde data in beide databases krijgen. Vervolgens zou je deze data kunnen ophalen uit de database die het minst belast is.

Echter zijn er ook verschillende tools die ervoor zorgen dat de databases automatisch gesynchroniseerd worden. Zoals ApexSQL Data diff. Dit is een programma dat controleert op verschillen tussen databases. En ze vervolgens kan synchroniseren.

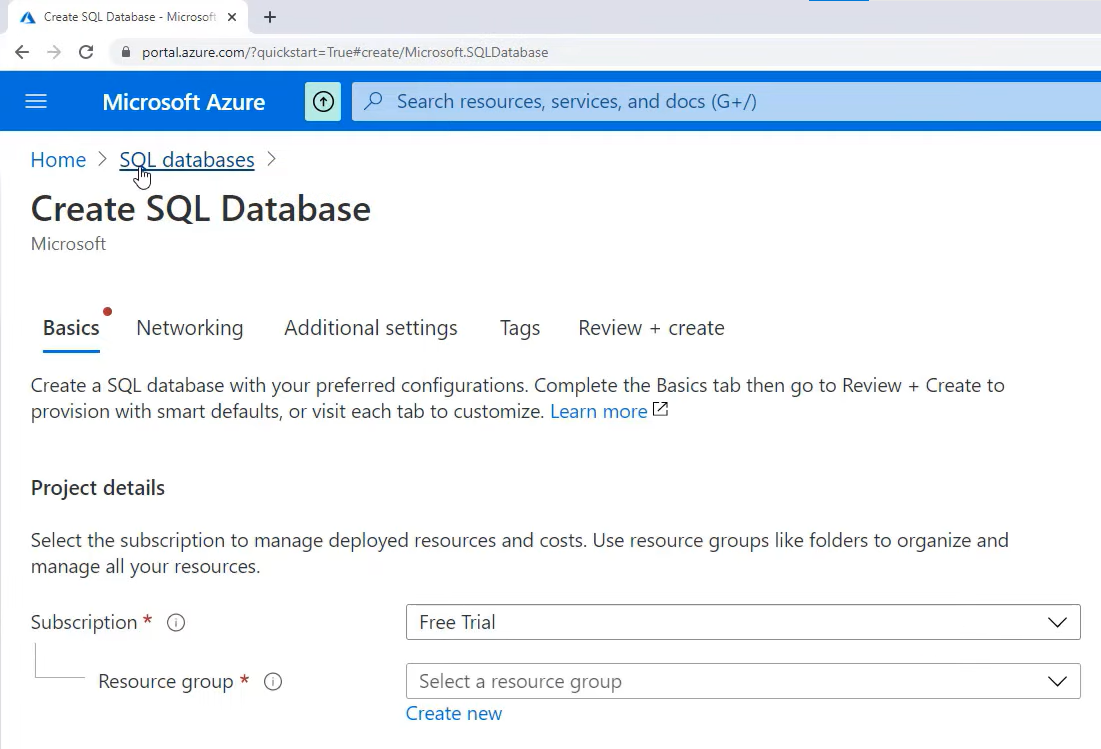
Op het internet staan ook verschillende oplossingen, die alleen SQL gebruiken. Hierbij wordt in de statement ook de andere database aangepast. Dit is echter een moeilijke implementatie aangezien wij gebruik maken van het entity framework.

Zoals onderstaande foto laat zien, is de data verdeeld over meerdere databases.

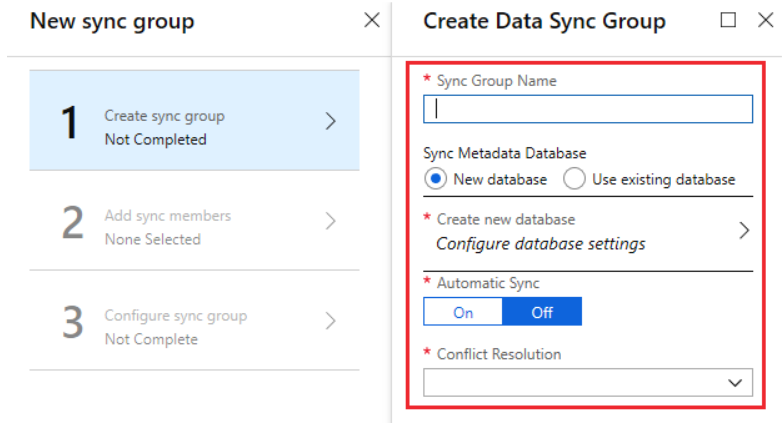


**Verdere implementatie**

Voor een verdere implementatie zou ik mijn MSSQL databases in Azure draaien.



Dit geeft ook een goede optie om de databases te syncen.



Hiermee zou ik dus verschillende databases kunnen maken die onderling syncen. Hierdoor heeft het systeem keuze uit verschillende databases en kan het de database gebruiken die het minste verkeer heeft. Dit kan worden gerealiseerd met messaging, mocht het te lang duren om een call naar de database te voltooien. Dan dient de applicatie te switchen naar een andere database.

# Conclusie

Tijdens dit semester heb ik een goede progressie gemaakt qua mijn leerdoelen. Ik denk dat ik mij op technisch en op professioneel vlak goed heb ontwikkeld tijdens dit semester.

Voor de laatste 5 leerdoelen had ik het voordeel dat ik de proftaak applicatie als voorbeeld kon nemen. Hierdoor had ik geen overzicht van de DevOps omgeving en was mijn applicatie structuur schaalbaar vanaf het begin.

Al met al ben ik zeer blij over mijn progressie binnen mijn semester. Dit semester was voor mij het eerste semester waar mijn applicatie zowel binnen development en productie draaide. En ik denk dat dit ook het dichts bij de realiteit komt. Hierdoor heb ik het gevoel dat dit een nuttig semester was voor mij.