Gm-measurements

Een afstudeerverslag door Tim van Hal omtrent de measurements microservice van Gerimedica.

Versienummer	1.0
Versietype	Final
Versiedatum	14-06-2022
Auteur	Tim van Hal
Studentnummer	s1115001
Afstudeerperiode	07-02-2022 t/m 08-07-2022
Afstudeerbedrijf	Gerimedica
Opleiding	Informatica
Specialisatie	Software Engineering
Onderwijsinstelling	Hogeschool Leiden

Versiebeheer

Versie	Versietype	Versiedatum	Auteur	Gewijzigd
0.1	Concept	30-03-2022	Tim van Hal	Opzetten document.
0.2	Concept	30-03-2022	Tim van Hal	Toevoegen voorwoord, bronnen, acceptatiecriteria en hoofdstukken 1 tot en met 5 en 9.
0.3	Concept	06-04-2022	Tim van Hal	Toevoegingen aan hoofdstukken 1, 8, 11, 12, de inhoudsopgave, de samenvatting en de begrippenlijst. Aanpassen verwoording na feedback van adviesformulier afstudeerplan.
0.4	Concept	13-04-2022	Tim van Hal	Toevoegen van literatuuronderzoek en software architectuur. Aanpassen reflectie.
0.5	Concept	20-04-2022	Tim van Hal	Verder refinen van competenties in verslag voor intervisie.
0.6	Concept	20-04-2022	Michiel Bruins	Eerste review voor doelen & resultaten
0.7	Concept	21-04-2022	Tim van Hal	Feedback verwerken van bedrijfsbegeleider.
0.8	Concept	21-04-2022	Tim van Hal	Laatste bewerkingen voor intervisie.
0.9	Concept	28-05-2022	Tim van Hal	Feedback verwerken van intervisie.
0.10	Concept	31-05-2022	Tim van Hal	Afschrijven reflectie en verwerken van testrapport. Puntjes op de i zetten.
0.11	Concept	13-06-2022	Tim van Hal	Herzien afstudeerverslag na feedback(formulier) afstudeerdocent.
0.12	Pre-final	14-06-2022	Tim van Hal	Laatste paar feedbackpunten verwerken van het feedbackformulier en afronden.
1.0	Final	14-06-2022	Tim van Hal	Final check & tweaks.

Voorwoord

Voor u ligt het afstudeerverslag van Tim van Hal in samenwerking met afstudeerorganisatie Gerimedica en opleidingsinstituut Hogeschool Leiden. Dit verslag en overige beroepsproducten die aan bod komen in het document zelf zijn gemaakt gedurende de afstudeerperiode tussen 7 februari 2022 en 8 juli 2022.

De nadruk van de scriptie wordt gelegd op het aantonen van de A- en B-competenties zoals beschreven door Hogeschool Leiden. Dit wordt bewezen aan de hand van de zogeheten beroepsproducten die zijn uitgewerkt voor het afstudeerbedrijf.

Mijn dank gaat uit naar bedrijfsbegeleider Michiel Bruins, Team B, alle andere betrokkenen bij Gerimedica en Hogeschool Leiden, waaronder afstudeerdocent Ron Arts.

Tim van Hal
Amsterdam 14 juni ‘22

Samenvatting

Gerimedica is een bedrijf wat software biedt aan specialisten ter ondersteuning van de chronische zorg, waaronder de ouderenzorg. Het bedrijf zich sinds kort ook op de opbreking van diens monoliet applicatie. Dit is mogelijk aan de hand van een microservice architectuur, waar momenteel al hard aan gesleuteld wordt. De afstudeeropdracht betreft het bouwen van een dergelijke microservice die uiteindelijk opgenomen wordt in de opkomende microservice architectuur binnen de afstudeerorganisatie. Deze microservice wordt uiteindelijk ook opgenomen in productie.

Het afstudeerverslag gaat in op het afstudeerbedrijf en de afstudeeropdracht. Ook worden de resultaten en opbrengsten van dien uitgebreid besproken. Er wordt ook gereflecteerd op het gemaakte werk en de uitgevoerde handelingen. Bovendien wordt de waardering gegeven over het afstudeertraject en -organisatie.

Het afstudeertraject heeft een doorlooptijd van twintig weken. Er is gewerkt aan de A-competenties en de volgende drie B-competenties: *software analyseren, software ontwerpen* en *software realiseren.* Deze competenties zijn aangetoond door middel van de gerealiseerde beroepsproducten.

Inhoudsopgave

1 - Inleiding

Gedurende het document kan de lezer kennis opdoen over het afstudeerbedrijf en de uitgevoerde werkzaamheden en beroepsproducten waaraan gewerkt is gedurende twintig weken tijd.

Allereerst wordt de achtergrond geschetst en wordt de afstudeerorganisatie beschreven. Hier wordt onder andere meer context gegeven over het bedrijf en worden belangrijke belanghebbenden opgesomd.

Na het beschrijven van de organisatie is het de beurt aan de opdracht. Het projectdoel wordt in hoofdstuk 4 aangekaart. Vervolgens wordt de kans die zich voordeed beschreven in hoofdstuk 5. Vervolgens wordt de concrete opdracht uitgebreid besproken in hoofdstuk 6.

De uitgangspunten, verantwoording en eigen verwachtingen volgen na het beschrijven van de opdracht. Bovendien is ook de aanpak van het project genoteerd in het hoofdstuk wat hierna volgt. Na deze twee hoofdstukken volgen de risico's die zich voordoen bij het project.

De resultaten van de afstudeerperiode worden beschreven in hoofdstuk 10. Hier worden alle belangrijke beroepsproducten opgesomd en globaal besproken. Vervolgens worden de uitwerkingen en oplossingen van dien uitgebreid besproken in de opbrengsten (hoofdstuk 11).

De reflectie volgt in hoofdstuk 12. Hier wordt teruggekeken naar de afstudeerperiode. Tevens wordt er gerefereerd naar de eigen verwachtingen en de A- en B-competenties die aangetoont dienen te worden.

Hoofdstuk 13 staat in het kader van het nawoord. Hier wordt onder andere besproken hoe de beroepsproducten zijn overgedragen en of deze in goede orde ontvangen zijn.

Tenslotte wordt er afgesloten met een begrippenlijst en een literatuurlijst. Hier kan men vitale begrippen en bronnen terugvinden die gebruikt zijn tijdens het opbouwen van het document.

2 - Achtergrond

Gerimedica is een organisatie die software realiseert ter ondersteuning van specialisten in de chronische zorg, waaronder de ouderenzorg. Momenteel biedt Gerimedica onder andere ondersteuning door middel van haar EPD (Elektronisch Patiënten Dossier) genaamd Ysis.

Gerimedica richt zich momenteel ook op echte interoperabiliteit in de zorg. Interoperabiliteit binnen de zorg gaat in op de digitale overdracht van medische gegevens. Daarentegen zijn hier nog geen goede standaard oplossingen voor. Gerimedica wilt zich maar al te graag inzetten om te zorgen dat dit in de toekomst wel zo is.

Binnen Gerimedica vindt er momenteel ook een ‘shift’ (ofwel verschuiving) plaats. Momenteel wordt er gewerkt op de monoliet applicatie binnen Gerimedica op te breken en een microservice architectuur op te stellen. Hierdoor is verschillende functionaliteit geclusterd, maar werkt het nog steeds samen als één geheel.

(Bruins, 2021)

3 - Beschrijving Afstudeerorganisatie

Hoofdstuk 3 gaat in op de beschrijving van de afstudeerorganisatie. Gedurende het tijdspad van 7 februari 2022 tot en met 8 juli 2022 wordt er gewerkt aan afstudeeropdracht.

Gerimedica een zelfstandig bedrijf, ooit gestart als een initiatief vanuit het UMC Amsterdam, samen met een 3-tal zorginstellingen. Gerimedica is uitgegroeid tot een eigen, gezonde organisatie. Het kantoor van het afstudeerbedrijf ligt aan Rijnlandlaan 3 te Amsterdam.

3.1 - Contactgegevens

De bedrijfsbegeleider, toegewezen vanuit Gerimedica, is Michiel Bruins. Hieronder zijn diens contactgegevens opgesomd:

- Naam: Michiel Bruins
- Email: m.bruins@gerimedica.nl
- Tel: +31 6 55584191

Vanuit het opleidingsinstituut is er ook een afstudeerdocent aangewezen: Ron Arts. De contactgegevens van dien zijn hieronder weergeven:

- Naam: Ron Arts
- Email: arts.r@hsleiden.nl

Het afstudeertraject wordt belopen door Tim van Hal. De contactgegevens van dien zijn hieronder op een rijtje gezet:

- Naam: Tim van Hal
- Email: s1115001@student.hsleiden.nl
- Tel: +31 6 41695479

Tim neemt tijdens de afstudeerperiode de rol van developer op zich. De developer doorloopt het werkproces deels in teamverband, maar werkt nog steeds zelfstandig aan een microservice.

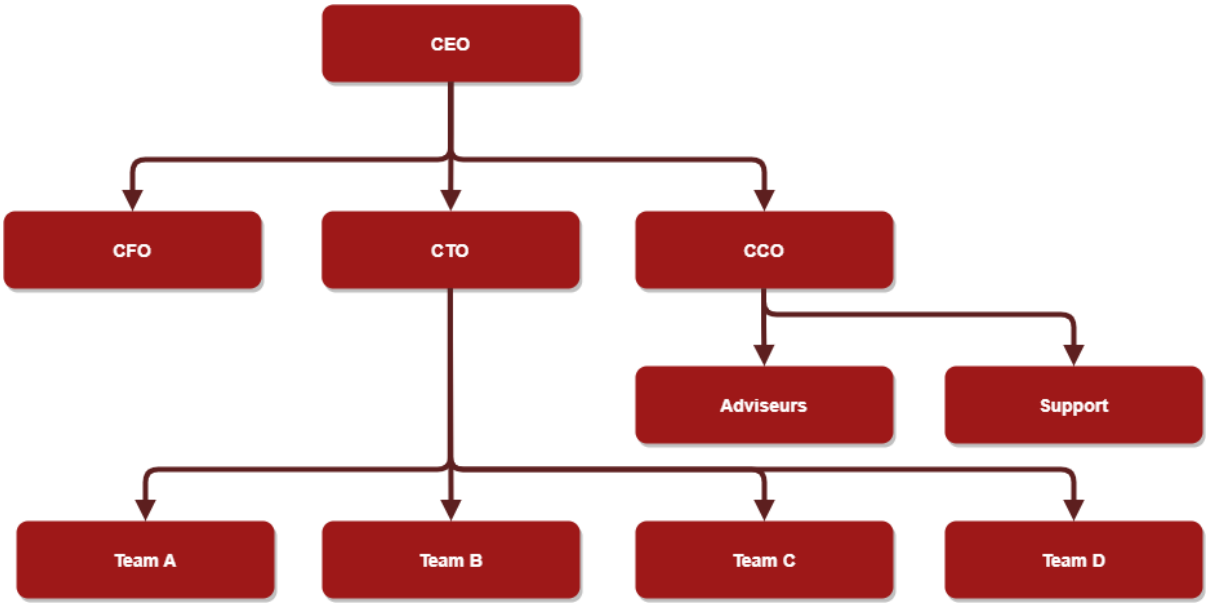
3.2 - Interne ondersteuning

Michiel Bruins staat altijd open om vragen te beantwoorden of om te hulp te schieten, wanneer wenselijk. Dit is ten alle tijden mogelijk in persoon of online met behulp van Slack. Bovendien kan Slack ook gebruikt worden om te overleggen met andere medewerkers.

Elke woensdag vindt er een wekelijkse meeting plaats om de progressie in kaart te brengen en om alle uitgevoerde en nog uit te voeren werkzaamheden te bespreken. Dit vindt plaats om 11:00 ‘s ochtends, tenzij anders aangegeven. Ook worden er agendapunten opgezet ter bespreking.

3.3 - Organisatiestructuur

De organisatiestructuur van Gerimedica kan weergeven met behulp van een zogeheten organogram. Een organogram is een diagram wat de opbouw van een bedrijf in kaart brengt ten behoeve van een beter overzicht.



Figuur 3.3a - Organogram van de organisatiestructuur van Gerimedica.

De CEO Thomas Ferguson staat bovenaan in het bedrijf. Hieronder vallen verschillende, andere managementrollen, waaronder de CFO Jeffrey Smits, de CTO Salvatore La Fiura en de CCO Henk de Jong. Onder de CCO vallen de adviseurs en de support staff, terwijl de CTO alle development teams onder de vleugel heeft. De CFO gaat over de financiële zaken binnen het bedrijf.

Gerimedica kan verder omschreven worden aan de hand van Mintzberg. Mintzberg definieert organisatiestructuren aan de hand van een set van vijf figuren. Elk figuur beschrijft een ander type organisatie. Hieronder is het figuur weergeven, passen bij Gerimedica.

Figuur 3.3b - De organisatiestructuur van Gerimedica volgens de ideologie van Mintzberg.

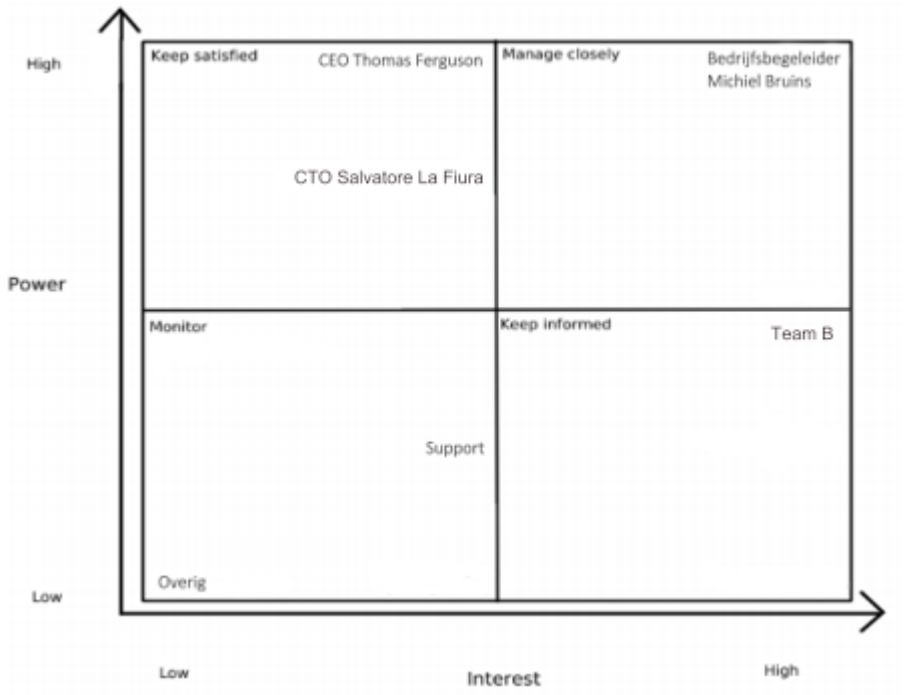
Volgens de ideologie van Mintzberg is Gerimedica een professionele organisatie. De werkvloer bestaat uit hoogopgeleide professionals. Bovendien zijn de uitgevoerde processen specialistenwerk.

Gerimedica heeft een kleine machtsafstand. Men kan makkelijk met iedereen een praatje maken op de werkvloer: zelfs met de CEO. De technostructuur (linker bolletje) bij Gerimedica is beperkt, maar adviseurs en support (rechter bolletje) zijn er voldoende.

Nota Bene: Het organogram gaat in op het meest vitale onderdeel van de afstudeerorganisatie. Andere afdelingen die niet relevant zijn voor de afstudeeropdracht zijn achterwege gelaten.

3.4 - Stakeholders

Hoofdstuk 3.4 beschrijft de verschillende stakeholders binnen het bedrijf. Het betreft alle betrokkenen bij het ontwikkelen van de gm-measurement microservice. Hieronder kan men een blik werpen op de bijhorende stakeholdermatrix.



Figuur 3.4a - De stakeholdermatrix met betrekking tot het afstudeertraject.

Hierboven zijn verschillende medewerkers binnen Gerimedica die betrokken zijn bij de afstudeeropdracht. De CEO en CTO zijn de managementrollen die betrokken zijn. Beide partijen hebben veel macht, maar de interesse ligt lager. De bedrijfsbegeleider heeft veel macht en tevens veel interesse. Verder zijn de developers van Team B de developers waarmee er gewerkt wordt gedurende het tijdsinterval. De support is soms ook van belang. Bijvoorbeeld bij ondersteuning van de VPN-verbinding.

3.5 - Bedrijfscultuur

Een goede bedrijfscultuur is een belangrijk onderdeel voor een gezonde, goed functionerende organisatie. Ter beschrijving van de bedrijfscultuur binnen Gerimedica, wordt de ideologie van Hofstede gebruikt. Hofstede doet dit aan de hand van diens culturele typologie.

Gerimedica is een bedrijf met een relatief kleine **machtsafstand**. Dit bleek ook al uit de organisatiestructuur van Mintzberg. Iedereen werkt in hetzelfde kantoor, op dezelfde plek en een praatje maken met de CEO is mogelijk.

Binnen Gerimedica is ook sprake van **collectivisme** (tegenover individualisme). Dit houdt in dat men voornamelijk in teamverband werkt en dat men elkaar bovendien helpt. Er wordt samen gezocht en gewerkt aan oplossingen om mensen te ondersteunen in de zorgsector.

Feminiteit staat tegenover masculiniteit en is leidend binnen Gerimedica. Men helpt elkaar en er hangt een serieuze, maar tevens gezellige sfeer. Zo wordt er bijvoorbeeld pingpong gespeeld buiten werktijden. Men probeert binnen het bedrijf ook niet koste wat het kost beter te zijn dan een ander.

Gerimedica is vrij strikt wat betreft **onzekerheidsvermijding**. Datalekken mogen namelijk niet plaatsvinden binnen een bedrijf wat werkt met gegevens van vele Nederlands burgers. De opgestelde wetten en afspraken moeten daarom koste wat het kost gewaarborgd worden. Eventuele datalekken schaadt het imago van het bedrijf en kan leiden tot juridische problemen.

Tenslotte maakt Hofstede onderscheidt tussen korte termijn en lange termijn. Binnen Gerimedica wordt er gewerkt op **lange termijn**. De roots van het bedrijf liggen in het westen, wat betekent dat het kapitalistisch is. Ondanks dat de westerse politiek vaak gericht is op korte termijn, geldt dit niet per definitie voor Gerimedica: het bedrijf streeft onder andere naar het verbeteren van medische zorg, medisch onderzoek en de maatschappij in het algemeen. Dit is een visie op lange termijn.

4 - Projectdoel

Het project wordt uitgevoerd tijdens de afstudeerperiode. Deze is gestart op 7 februari 2022 en loopt ten einde op 8 juli 2022. Het betreft een measurements microservice die gemaakt wordt voor afstudeerorganisatie Gerimedica. Deze ‘gm-measurements’ microservice moet het mogelijk maken om meetgegevens, waaronder bijvoorbeeld glucose, vochtinname en defecatie te verwerken en beschikbaar te stellen aan haar gebruikers.

Voordat de realisatie begint wordt er onderzoek uitgevoerd en wordt de software architectuur ontworpen. Beide onderwerpen komen terug in een onderzoeksrapport en in een software architectuur document. De realisatie verloopt aan de hand van stories. Deze stories worden uitgewerkt in meerdere sprints van drie weken tijd.

Richting het einde van het afstudeertraject zijn er verschillende beroepsproducten gerealiseerd, waaronder dus de gm-measurements microservice. Deze zal uiteindelijk, via de gm-gateway node van Gerimedica, onderdeel uitmaken van de nieuwe microservice architectuur.

5 - Kans

Momenteel is Gerimedica een omschakeling aan het maken naar een microservice architectuur om de monoliet applicatie op te breken. Nieuwe functionaliteit die toegevoegd moet worden, wordt ook opgedeelt in microservices. Dit houdt daarentegen in dat het bedrijf een lang development proces in gaat, aangezien er dus meerdere microservices gebouwd moeten worden. Een langer development proces resulteert op zijn beurt weer in het langer moeten wachten op bepaalde functionaliteit binnen de software. Voor de developers is dit naar, omdat zij meer werk moeten verrichten en langer moeten werken met de legacy code. Voor de product owner is dit bijvoorbeeld ook niet fijn, aangezien de taken omtrent de microservice langer in de backlog blijven staan, aangezien het team meer belast raakt tijdens de sprints.

Er doet zich een kans voor bij Gerimedica: de realisatie van één van de microservices kan dienen als afstudeeropdracht: de gm-measurements microservice. Dit werkt verlichtend voor de development department, aangezien één van de microservices nu door iemand anders wordt overgenomen. Deze microservice is dan ook sneller beschikbaar.

Er wordt gewerkt aan de gm-measurements microservice. Verder wordt er een onderzoek uitgevoerd, een architectuur opgesteld en een teststrategie gevolgd. Bovendien worden de relevante college documenten netjes bijgehouden.

De epic en onderliggende stories zijn intern besproken en zijn beschikbaar gesteld in Jira. Hier wordt gebruik van gemaakt om de realisatie tot een goed einde te laten komen en professionele beroepsproducten neer te zetten richting de afsluiting van het afstudeertraject.

Tenslotte sluit het werk tevens aan op de specialisatie vanuit Hogeschool Leiden: Software Engineering. Werkzaamheden worden verricht onder de titel Developer Intern en de uitgevoerde werkzaamheden raken alle vlakken in de competentie rubrics van Software Engineering: er wordt namelijk geanalyseerd, ontworpen en gerealiseerd.

6 - Concrete opdracht

Dit hoofdstuk bevat informatie over de concrete opdracht die uitgevoerd wordt gedurende het afstudeertraject bij Gerimedica. Allereerst wordt de opdracht bondig besproken. Vervolgens worden de verschillende beroepsproducten aangekaart, wat ondersteund wordt door middel van een Product Breakdown Structure (PBS). Na de beroepsproducten worden de werkzaamheden opgesomd. Deze worden gelinkt aan de individuele beroepsproducten. Tenslotte sluit het hoofdstuk af met een opsomming van alle requirements die voorafgaand opgesteld zijn.

6.1 - De opdracht

Gerimedica streeft momenteel naar het opbreken van diens monoliet applicatie: het Elektronisch Patiënten Dossier (EPD) genaamd Ysis. Vervolgens wordt er een microservice architectuur opgebouwd. De opdracht betreft het bouwen van een dergelijke microservice voor Gerimedica. De microservice gaat specifiek in op meetgegevens van patiënten. Denk hierbij bijvoorbeeld aan bloedsuikerspiegel, lichaamstemperatuur, vochtinname, et cetera. Uiteindelijk wordt de microservice, in combinatie met alle beroepsproducten, opgeleverd (digitaal en verbaal) aan de afstudeerorganisatie.

6.2 - Beroepsproducten

Het belangrijkste beroepsproduct wat opgebouwd is gedurende de twintig weken bij Gerimedica, is de microservice. De overige beroepsproducten zijn opgebouwd ter ondersteuning van de microservice, dan wel in opdracht van het opleidingsinstituut. Hieronder zijn alle beroepsproducten op logische wijze opgesomd.

- Een microservice voor in productie.
  - Opstellen architectuur
  - Authentication
  - Business Logic
  - Ysis koppeling
- Een acceptatieplan
- Een (kwalitatief) literatuur onderzoeksrapport
  - *Hoe realiseert men een functionele, geteste interoperability microservice binnen een bestaande microservice architectuur met behulp van frameworks, libraries en design patterns gedurende de projectdoorloop?*
    - *Wat is een microservice en hoe helpt het Spring Boot framework hierbij tijdens de doorlooptijd van het project?*
    - *Wat is de HAPI library en hoe is deze inzetbaar in combinatie met het Spring Boot framework gedurende de realisatie?*
    - *Wat is de REST-assured library en hoe draagt het bij aan de teststrategie binnen de context van het project?*
    - *Welke design patterns zijn inzetbaar binnen de context van het project en hoe verbetert dit de code kwaliteit?*
- Een (kwantitatief) empirisch onderzoeksrapport

- *Wat is de snelste wijze van migratie zonder downtime tijdens migratie van een Axon server naar een non-Axon server?*
  - *Wat zijn de bottlenecks van een Axon server en op welke wijze is het mogelijk om downtime te vermijden tijdens het migreren van een Axon server naar een non-Axon server?*
  - *Wat is de invloed van de hoeveelheid events op de migratiesnelheid tijdens het migreren van een Axon server naar een non-Axon server?*

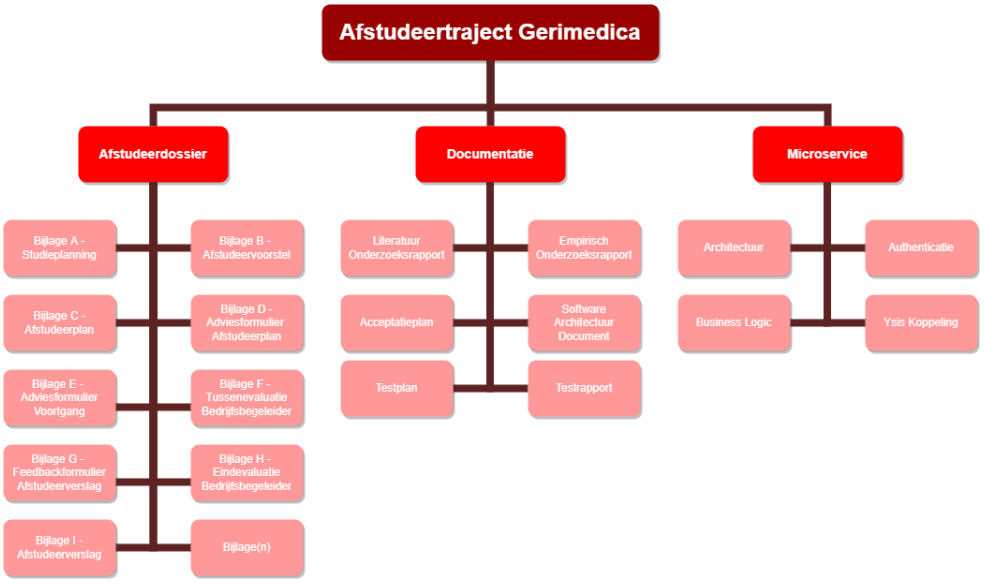
➢ Een software architectuur document
➢ Een uitgewerkte teststrategie (inclusief testscenario's)
- Tesplan
- Testrapport

➢ Een afstudeerdossier voor het opleidingsinstituut
- Bijlage A - Studieplanning
- Bijlage B - Afstudeervoorstel
- Bijlage C - Afstudeerplan
- Bijlage D - Adviesformulier Afstudeerplan
- Bijlage E - Adviesformulier Voortgang
- Bijlage F - Tussenevaluatie Bedrijfsbegeleider
- Bijlage G - Feedbackformulier Afstudeerverslag
- Bijlage H - Eindevaluatie Bedrijfsbegeleider
- Bijlage I - Afstudeerverslag

Nota Bene: Tussentijdse progressie is besproken gedurende het einde van sprints. Dit is opgenomen in het werkzaamheden hoofdstuk.

6.2.1 - Product Breakdown Structure

De verschillende beroepsproducten die zojuist zijn opgesomd zijn te weergeven in een Product Breakdown Structure (PBS). Dit biedt een duidelijk beeld van alle beroepsproducten die opgeleverd zijn.



Figuur 6.2.1a - De beroepsproducten gevisualiseerd door middel van een PBS.

De PBS is opgedeeld in drie verschillende hoofdonderdelen: afstudeerdossier, documentatie en microservice. Het afstudeerdossier bevat alle bijlagen die verstrekt worden door het opleidingsinstituut in combinatie met de overige beroepsproducten. De documentatie bestaat uit alle overige documenten, zoals onderzoek en software architectuur. Tenslotte is de microservice los opgenomen met een opsomming van de belangrijkste functionaliteit.

6.3 - Werkzaamheden

De werkzaamheden zijn uitgevoerd over een looptijd van twintig weken. De werkzaamheden staan direct in verbinding met de eerder genoemde beroepsproducten in dit hoofdstuk. Hieronder staan alle werkzaamheden nog een keer op een rijtje: deze zijn gelinkt aan de verschillende beroepsproducten.

Het belangrijkste onderdeel van de afstudeerperiode is de microservice die gebouwd wordt. Dit gebeurt tijdens een realisatiefase van twaalf weken. Deze twaalf weken worden doorlopen in vier sprints. Iedere sprint heeft een doorlooptijd van drie weken.

Gedurende de afstudeerperiode speelt onderzoek ook een vitale rol. Onderzoek heeft in eerste instantie plaatsgevonden in week twee en drie. Daarnaast heeft er, na behoefte aan een empirisch gericht onderzoek, een tweede onderzoeksperiode plaatsgevonden gedurende week vier en vijf. Beide trajecten hebben geresulteerd in een onderzoeksrapport: een literatuuronderzoek en een empirisch onderzoek.

Na uitvoering van het eerste onderzoek, is de architectuur van de applicatie opgesteld. Dit staat gedocumenteerd in het Software Architectuur Document (SAD). Gedurende de creatie van het SAD is er nauw contact gehouden met de betrokkenen om te garanderen dat de kwaliteiten van de architectuur van de microservice acceptabel is. Hiervoor is onder andere het acceptatieplan gebruikt.

Vlak voor de realisatie is het testplan gemaakt. In het testplan staat vooraf het testproces gedocumenteerd. Het testplan is hiermee de basis voor het testen wat gedurende en na de ontwikkelfase heeft plaatsgevonden. De resultaten van het testen zijn op logische wijze opgenomen in het testrapport, wat hierop voortborduurt.

Richting het einde van de afstudeerperiode worden alle beroepsproducten gebundeld in een afstudeerdossier. Hier staan verder ook de documenten in die door de opleiding zijn verstrekt: denk hierbij aan feedback- en evaluatieformulieren. Deze documenten zijn gedurende het gehele afstudeertraject opgebouwd.

De overdracht betreft een microservice en het afstudeerdossier voor archivering. De overdracht vindt zowel fysiek als digitaal plaats. Toelichting wordt verstrekt, indien wenselijk.

6.4 - Requirements

In dit hoofdstuk worden de requirements nog een keer op een rijtje gezet. De requirements hebben een prioritering ontvangen aan de hand van de MoSCoW-methode. De MoSCoW-methode beschrijft prioritering aan de hand van vier categorieën.

- **Must have**: wat de applicatie moet bevatten.
- **Should have**: wat de applicatie zou moeten bevatten.
- **Could have**: wat de applicatie kan bevatten.
- **Won't have**: wat de applicatie niet moet bevatten.

Hieronder staan de requirements weergeven in een overzichtelijke tabel.

Requirement	**Prioritering**
Controles	
Als een gebruiker wil ik controles kunnen opvragen, zodat ik alle metingen kan inzien.	Could have
Als een gebruiker wil ik controles kunnen bewerken, zodat ik specifieke metingen kan aanpassen.	Could have
Lengte en gewicht	
Als een gebruiker wil ik lengte en gewicht kunnen opvragen, zodat ik de metingen en de huidige situatie kan inzien.	Could have
Als een gebruiker wil ik lengte en gewicht metingen kunnen bewerken, zodat ik momentopnamen kan registreren.	Could have
Glucose	
Als een gebruiker wil ik glucose kunnen opvragen, zodat ik de glucose levels van een patiënt kan inzien.	Could have
Als een gebruiker wil ik glucose kunnen bewerken, zodat ik specifieke metingen kan aanpassen.	Could have
Vochtlijst	
Als een gebruiker wil ik de vochtlijst kunnen opvragen, zodat ik de vochtbalans kan inzien.	Could have
Als een gebruiker wil ik de vochtlijst kunnen bewerken, zodat ik vochtinname en uitscheiding kan registreren.	Could have
Defecatie	
Als een gebruiker wil ik defecatie kunnen opvragen, zodat ik alle metingen kan inzien.	Must have
Als een gebruiker wil ik defecatie kunnen bewerken, zodat ik alle uitscheiding kan registreren.	Must have

Inloggen	
Als een gebruiker wil ik inloggen, zodat ik toegang heb tot het systeem.	Must have
Intern	
Als een developer wil ik de logging tool kunnen gebruiken, zodat ik alle uitgevoerde handelingen kan inzien.	Must have
Als een developer wil ik de CI gespecificeerd hebben, zodat ik de pipeline zowel automatisch als handmatig kan gebruiken.	Must have
Als een developer wil ik Prometheus kunnen gebruiken, zodat ik kan monitoren met dashboards.	Must have

Figuur 6.4a - Requirements met prioritering, weergeven in tabelvorm.

De belangrijkste onderdelen van de microservice zijn must have. Sommige type metingen zijn momenteel lager geprioriteerd, wegens het feit dat er nog overlegd wordt of deze metingen in de toekomst opgesplitst worden of niet of wegens het feit dat ze vrij complex zijn.
Inloggen is cruciaal voor de applicatie. Zonder authenticatie komt de gebruiker niet voorbij de gateway, wat betekent dat er ook geen gebruik van de microservices kan worden gemaakt.
Alles onder intern is een eis vanuit de developers en testers en zijn op alle omgevingen (testomgevingen en productieomgevingen) van toepassing.

6.5 - Acceptatiecriteria

In dit hoofdstuk worden de acceptatiecriteria aan de lezer geïntroduceerd. Het betreft de functionele en technische criteria zoals beschreven in het acceptatieplan.

Bron	Naam	Architecturele relevantie	Geadresseerd in:
Acceptatieplan	Beheerbaarheid	De beheerbaarheid gaat in op onderhoudbaarheid van het product en logging/monitoring.	2.1 - Beheerbaarheid
Acceptatieplan	Beveiliging	De beveiliging gaat in op de modificatie en toegankelijkheid van data.	2.2 - Beveiliging
Acceptatieplan	Standaards	Standaards gaat in op afspraken en standaardisatie rondom de realisatie van de microservice.	2.3 - Standaards
Acceptatieplan	Functioneel	Functionele eisen staan los van de	2.4 - Functioneel

		niet-functionele/technische eisen en zijn de use cases die aan bod komen bij de realisatie van de microservice.	

Figuur 6.4b - De classificering van de acceptatiecriteria volgens het acceptatieplan.

Bovenstaande tabel weergeeft de classificering van de acceptatiecriteria die opgesteld staan in het acceptatieplan. Het betreft een viertal groepen: *beheerbaarheid, beveiliging, standaards* en *functioneel*. Hieronder vallen verschillende acceptatiecriteria die zijn opgesteld met behulp van stakeholders.

De niet-functionele/technische eisen vallen onder beheerbaarheid, beveiliging en standaards. De functionele eisen vallen onder functioneel. Alle acceptatiecriteria met betrekking tot het project zijn in detail terug te vinden in het acceptatieplan *'s1115001_Acceptatieplan_1.0.pdf'*.

6.6 - Projectplanning

De projectplanning gaat in op de werkzaamheden die uitgevoerd worden over twintig weken tijd. Allereerst worden aannames opgesomd die van toepassing zijn op de projectplanning. Vervolgens wordt alles gevisualiseerd aan de hand van een Gantt-Chart. Een Gantt-Chart biedt een duidelijk beeld van alle werkzaamheden en hoe deze in elkaar overgaan.

6.6.1 - Aannames

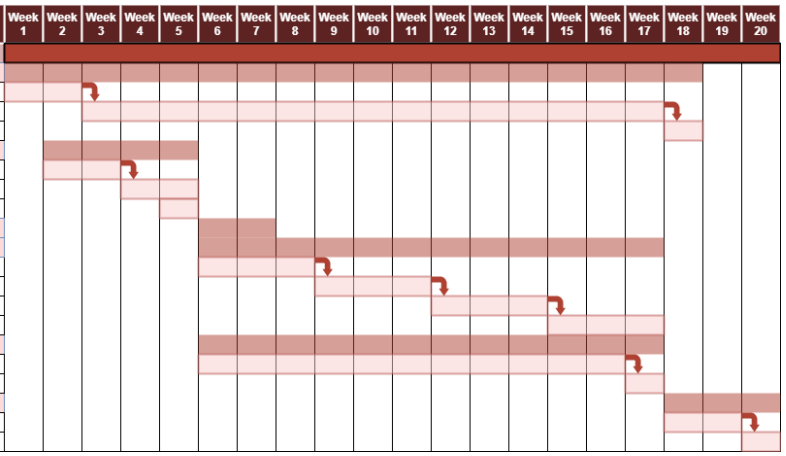
De aannames zijn belangrijke opmerkingen met betrekking tot de projectplanning die handig zijn om in het achterhoofd te houden. De Gantt-Chart die volgt vloeit voort uit deze aannames.

- De realisatiefase is onderverdeeld in sprints van drie weken, aangezien er gewerkt wordt als onderdeel van een scrum team. Dit is daarentegen enkel ten behoeve van het volgen van het bedrijfsproces van de afstudeerorganisatie: de microservice wordt alsnog zelfstandig gerealiseerd.
- De laatste drie weken zijn ingezet voor uitloop, aangezien de afstudeerzitting ergens in deze periode zal plaatsvinden. De realisatie- en testfase zijn dan, onder normale omstandigheden, afgerond.
- Feedback en advies formulieren worden aangeleverd vanuit de afstudeerorganisatie en/of de afstudeerdocent: deze worden dus niet opgenomen in de projectplanning.

6.6.2 - Gantt-Chart

Dit hoofdstuk biedt een visualisatie van de projectplanning in de vorm van een Gantt-Chart. De Gantt-Chart heeft daarentegen enkele (luttele) aanpassingen gekregen sinds de start van het afstudeertraject. De planning is hieronder weergeven.

#	Taaknaam	Duratie
1	Afstudeertraject	20 weken
2	Afstudeerdossier	18 weken
3	Afstudeerplan	2 weken
4	Afstudeerverslag	15 weken
5	Bundeling	1 week
6	Opstellen architectuur	4 weken
7	Literatuuronderzoek	2 weken
8	Software Architectuur document	2 weken
9	Testplan	1 week
10	Empirisch onderzoek	2 weken
11	Realiseren microservice	12 weken
12	Sprint 1	3 weken
13	Sprint 2	3 weken
14	Sprint 3	3 weken
15	Sprint 4	3 weken
16	Testen	12 weken
17	Functioneel	11 weken
18	Testrapport	1 week
19	Oplevering	3 weken
20	Uitloop	2 weken
21	Overdracht	1 week



Figuur 6.5.2a - De projectplanning van de afstudeerperiode, weergeven in een Gantt-Chart.

Het afstudeerdossier wordt opgebouwd gedurende achttien weken tijd. De laatste week dient daarentegen alleen voor het bundelen van alle documenten.

Week twee tot en met week vijf worden gebruikt voor de algemene opstelling van de architectuur. Dit resulteert onder andere in een literatuuronderzoek, een Software Architectuur Document (SAD) en een teststrategie (testplan en -rapport).

De twaalf weken die volgen na het opstellen van de architectuur zijn gebruikt voor de realisatie en het testproces. Dit resulteert in de microservice en een testrapport met de resultaten van het testen.

De laatste drie weken zijn, zoals besproken in de aannames, gereserveerd voor de oplevering van de documenten en de afstudeerzitting. Onder normale omstandigheden zijn alle overige documenten dan klaar.

De aanpassing die gemaakt is aan de planning is de toevoeging van een tweede onderzoek. Het eerste onderzoek is een kwalitatief literatuuronderzoek, maar gedurende de tweede onderzoeksperiode is de focus gelegd op een kwantitatief, empirisch onderzoek. Het onderwerp van het tweede onderzoek valt qua realisatie buiten de scope van de opdracht. Om deze reden loopt het tweede onderzoek voor korte duur parallel aan de realisatie: het heeft zakelijk nut voor het bedrijf, maar niet specifiek voor de afstudeeropdracht. Het tweede onderzoek heeft inhoudelijk dus geen invloed op de software architectuur of de realisatie.

Nota Bene: Feedback en adviesformulieren verstrekt door Hogeschool Leiden of het afstudeerbedrijf zijn niet opgenomen in de projectplanning. Dit staat ook in de aannames beschreven.

7 - Uitgangspunten, verantwoording en verwachtingen

Dit hoofdstuk bespreekt eerst de uitgangspunten van de afstudeeropdracht. De gebruikte aanpak gedurende de afstudeerperiode wordt besproken in het hoofdstuk wat volgt. De eigen verwachtingen sluiten vervolgens het hoofdstuk af met de eigen verwachtingen. Hier wordt uiteindelijk ook op gereflecteerd.

7.1 - Uitgangspunten

De uitgangspunten met betrekking tot de opdracht dienen om grenzen te stellen aan alle werkzaamheden en producten die uitgevoerd en gerealiseerd moeten worden. De opdracht zelf is het bouwen van een ‘measurements’ microservice voor het afstudeerbedrijf, maar wat valt hier nou precies allemaal onder? Hieronder is een lijst opgesteld met alle uitgangspunten die van toepassing zijn op het afstudeertraject.

- De opdracht betreft een ‘measurements’ microservice realiseren ter bevordering van de microservice architectuur.
- De microservice bevat enkel metingen/measurements: deze staan opgenomen in Ysis. Verdere modellen worden niet opgenomen in de microservice.
- De microservice wordt opgebouwd aan de hand van een acceptatieplan.
- De opdracht bevat het ontwerpen en bouwen van de microservice onder architectuur.
- Voorafgaand aan de realisatie wordt onderzoek uitgevoerd.
- Er wordt een teststrategie opgesteld ter bevordering van het testproces.
- Extra werkzaamheden, zoals migratie, vallen buiten de scope van de opdracht.
- Het onderhouden/beheren van de microservice is niet onderdeel van de opdracht.

7.2 - Verantwoording van de gekozen aanpak

Dit hoofdstuk kaart de verwachtingen van de gekozen aanpak aan. Het heeft betrekking op de projectplanning zoals deze beschreven staat in het hoofdstuk ‘Concrete Opdracht’.

Allereerst wordt er gewerkt aan het opbouwen van het afstudeerplan. Dit is relatief snel gedaan en heeft weinig tijd nodig, maar gaat wel over in het afstudeerverslag. Dit is allemaal onderdeel van het afstudeerdossier, wat gedurende het gehele traject wordt opgebouwd.

Vanaf de tweede week wordt de focus ook gelegd op het opstellen van de architectuur. Dit wordt gedaan door eerst onderzoek uit te voeren, vervolgens de software architectuur te ontwerpen en te documenteren en tenslotte door een teststrategie op te stellen. Gedurende deze periode heeft er daarentegen twee keer onderzoek plaatsgevonden. Normaliter was dit slechts één keer. Dit heeft te maken met het feit dat er naast een kwalitatief onderzoek, ook behoefte was aan een kwantitatief onderzoek. Het tweede onderzoek is daarentegen parallel gemaakt met het Software Architectuur Document (SAD) en het testplan. Dit zit hem in het feit dat het tweede onderzoek wel zakelijk nut heeft voor het bedrijf, maar buiten de scope van de opdracht valt.

De realisatie vindt plaats gedurende een doorlooptijd van twaalf weken. Er wordt ontwikkeld door middel van Scrum. Dit betekent dat de twaalf weken opgedeeld worden in vier sprints van drie weken. Op deze manier kan er zelfstandig gewerkt worden aan de opdracht, maar kan er alsnog voldaan worden aan het bedrijfsproces van de afstudeerorganisatie.

De laatste drie weken staan in het kader van de oplevering: alles wordt gebundeld en overgedragen. De achterliggende gedachte is dat de afstudeerzitting ergens in deze periode zal plaatsvinden. Hier is rekening mee gehouden tijdens het opbouwen van de planning, wat betekent dat alle beroepsproducten afgerond zijn met ingang van de laatste drie weken.

7.3 - Eigen verwachtingen

Dit hoofdstuk bespreekt de eigen verwachtingen omtrent het doorlopen van het afstudeertraject. Er wordt onder andere gesproken over het doorlopen van het proces.

Ondanks dat er veel geleerd is gedurende de carrière bij het opleidingsinstituut, zijn er leerdoelen opgesteld waar graag nog verder aan ontwikkeld wordt: planning en communicatie.

Er wordt verwacht dat de projectplanning daarom goed gevolgd wordt. Mochten er enige aanpassingen volgen aan de projectplanning, dan wordt dit gedocumenteerd en wordt dit nageleefd.

Binnen het kader communicatie wordt er verwacht dat er voldoende contact gehouden wordt met de overige medewerkers van de afstudeerorganisatie (bijvoorbeeld door online of fysieke meetings).

Verder wordt er verwacht dat er gemotiveerd en bovendien professioneel gewerkt wordt gedurende het tijdsinterval. Dit zal blijken uit de evaluatieformulieren die aangereikt worden vanuit de afstudeerorganisatie.

Tenslotte wordt er verwacht dat de gerealiseerde microservice acceptabel is om in productie te kunnen. Ook wordt verwacht dat alle overige beroepsproducten succesvol worden overgedragen en bijdragen aan het bewijzen van de opgezette competenties om de opleiding af te ronden.

Nota Bene: de eigen verwachtingen komen opnieuw aan bod bij de reflectie. In de reflectie wordt er onder andere teruggekeken op de eigen verwachtingen.

8 - Aanpak

De aanpak gaat in op de verschillende methoden en technieken die worden ingezet gedurende de twintig weken bij de afstudeerorganisatie om tot de gewenste resultaten te komen, zoals besproken met de stakeholder(s). Allereerst worden de verschillende methoden aangekaart, waarna de tools besproken worden.

8.1 - Methoden

In dit hoofdstuk worden de methoden bekend gemaakt. Er wordt gesproken over Scrum, het MVC pattern en verschillende design patterns die horen bij de aanpak van de afstudeeropdracht.

8.1.1 - Scrum

Scrum is een aanpak behorend tot de bundeling van methoden genaamd Agile Methodologies. Scrum wordt ingezet tijdens de realisatiefase: de realisatie vindt plaats door de opdeling in sprints. Aan het einde van de sprints worden de producten geleverd en wordt er gereflecteerd op het werk.

8.1.2 - Code Reviews

Gedurende de realisatie wordt er gebruik gemaakt van code reviews met behulp van collega’s om de werking van de code te garanderen. Dit gebeurt voordat de code gedeployed wordt op verscheidene omgevingen, waaronder ook de productieomgeving. Code reviews helpen bovendien bij het waarborgen van de code kwaliteit.

8.1.3 - MVC pattern

Het MVC pattern is een architectural pattern die een applicatie opdeelt in verschillende lagen. MVC maakt onderscheidt tussen de models, de view en de controllers. Het MVC pattern legt de basis voor de architectuur en wordt daarom gebruikt voor de realisatie van de microservice.

8.1.4 - Design patterns

Design patterns zijn blauwdrukken die oplossingen bieden voor complexe, vaak voorkomende code problematiek. Voorafgaand heeft er een literatuuronderzoek plaatsgevonden naar mogelijk bruikbare design patterns. Dit wordt meegenomen naar het realisatieproces.

8.2 - Tools

Dit hoofdstuk bespreekt de belangrijkste tools die aan bod komen bij de realisatie van de microservice. Bovendien betreft het de tools die binnen de scope van de afstudeeropdracht vallen.

8.2.1 - Ysis-Integration-Boot

Ysis-Integration-Boot is een clustering van belangrijke frameworks en libraries wat gebruikt wordt binnen Gerimedica voor de realisatie van diens microservices en andere applicaties. Ysis-Integration-Boot is eigenhandig opgesteld door het bedrijf voor toevoeging van nieuwe services en is nodig om te kunnen integreren in de bestaande omgeving.
Ysis-Integration-Boot bundelt onder andere Spring Boot, REST-assured, et cetera. Deze hoeven vervolgens niet handmatig toegevoegd te worden.

8.2.2 - Pipeline

Een geautomatiseerde pipeline van GitLab in een Docker container wordt gebruikt om code automatisch te testen en te deployen. Het deployen gebeurt naar een testomgeving en productieomgeving. De pipeline zal ook gebruikt worden tijdens de realisatie van de microservice ter bevordering van de productiecode.

8.2.3 - Prometheus

Prometheus is een relatief nieuwe tool binnen Gerimedica wat ingezet wordt om informatie te verkrijgen over gegevensgebruik in een dashboard. Prometheus wordt toegevoegd aan de microservice voor monitoring en alerting. Het wordt vaak gebruikt in combinatie met Grayfana voor dashboards.

8.2.4 - Graylog

Graylog is een logging tool, wat binnen Gerimedica gebruikt wordt om alle handelingen binnen het systeem te loggen. Deze logs worden opgeslagen in een database, wat ten alle tijden beschikbaar is voor de medewerkers.

8.2.5 - Flyway

Flyway is een handige tool voor database migratie. Flyway draagt bij aan het uitvoeren van SQL-scripts binnen een applicatie door direct de scripts tegen de database te runnen. Als de scripts aangepast worden of als er nieuwe scripts toegevoegd worden aan het project, dan worden deze automatisch uitgevoerd door Flyway. Op deze manier is de database altijd goed gestructureerd.

9 - Risico’s

Dit hoofdstuk beschrijft de verschillende risico’s die van toepassing zijn op de uitvoering van het project. De risico’s vloeien voort uit de projectplanning en de projectaanpak. Opgestelde risico’s gaan hand in hand met een set maatregelen die als reactie dienen, mocht er iets mis gaan. Hieronder staan eerst een set aan risico’s opgesteld.

- ➢ Het niet of slecht functioneren van de VPN bemoeilijkt het werken op afstand. Dit heeft een remmende werking.
- ➢ De focus kan in een later stadium op een andere microservice gelegd worden. In dit geval moet de software architectuur ook opnieuw aangepast te worden.
- ➢ Het wegvallen van de testomgeving en/of pipeline betekent dat de code minder goed tot niet functioneel getest kan worden. Dit is niet wenselijk voor productiecode.

9.1 - Maatregelen

Maatregelen zijn opgesteld om schade omtrent voorgedane risico’s te dekken of te verminderen. Hieronder zijn verschillende risico’s weergeven.

- ➢ Support staff is aanwezig om te hulp te schieten indien er iets niet aan de haak blijkt te zijn omtrent de VPN-verbinding.
- ➢ Gerimedica heeft eerder een test microservice gemaakt genaamd gm-documents. Deze microservice kan ter referentie gebruikt worden voor de ontwerpen, wat de ontwerptijd aanzienlijk verminderd.
- ➢ Gerimedica beschikt over meerdere testomgevingen.

10 - Resultaat

Dit hoofdstuk bespreekt globaal de verschillende resultaten die voortvloeien uit het doorlopen van het afstudeertraject. Hier worden bundig de beroepsproducten besproken. De complete uitwerking en toegepaste oplossingen komen aan bod in de opbrengsten (hoofdstuk 11).

10.1 - Afstudeerdossier

Het afstudeerdossier is een collectie van de documenten verstrekt vanuit het opleidingsinstituut en de overige beroepsproducten voor Gerimedica. Het afstudeerdossier dient uiteindelijk overgedragen te worden aan beide betrokken partijen voor archivering. Een visualisatie van het afstudeerdossier is te vinden in het hoofdstuk genaamd ‘Concrete Opdracht’.

10.2 - Acceptatieplan

Het opgestelde acceptatieplan bevat alle functionele en non-functionele eisen waaraan minimaal voldaan moet worden om de microservice te accepteren. Het acceptatieplan is opgesteld in overleg met de stakeholder(s). De architecturele eisen van de software architectuur zijn opgebouwd aan de hand van de acceptatiecriteria en er is verwezen naar dit plan.

10.3 - Onderzoek

Gedurende het afstudeertraject is er tweemaal onderzoek uitgevoerd. Het eerste onderzoek betreft een kwalitatief literatuuronderzoek en ging voornamelijk in op de gebruikte methoden en technieken. Het tweede onderzoek is een kwantitatief empirisch onderzoek wat de focus heeft gelegd op mogelijkheden rondom migratie binnen het Axon framework. Onderstaand zijn beide onderzoekscomponenten beschreven.Hal

10.3.1 - Kwalitatief onderzoek

Het kwalitatieve onderzoek heeft zich gericht op het verzamelen van secundaire data over de gebruikte methoden en technieken die gebruikt werden gedurende de ontwikkelfase. De hoofdvraag behorend tot dit onderzoek luidt als volgt:

Hoe realiseert men een functionele, geteste interoperability microservice binnen een bestaande microservice architectuur met behulp van frameworks, libraries en design patterns gedurende de projectdoorloop?

De bovenstaande hoofdvraag is beantwoord aan de hand van een set van vier vooraf gedefinieerde deelvragen. De vier deelvragen passend bij de hoofdvraag zijn hieronder opgesomd.

1. *Wat is een microservice en hoe helpt het Spring Boot framework hierbij tijdens de doorlooptijd van het project?*
2. *Wat is de HAPI library en hoe is deze inzetbaar in combinatie met het Spring Boot framework gedurende de realisatie?*
3. *Wat is de REST-assured library en hoe draagt het bij aan de teststrategie binnen de context van het project?*
4. *Welke design patterns zijn inzetbaar binnen de context van het project en hoe verbetert dit de code kwaliteit?*

Bovenstaande deelvragen hebben allemaal betrekking op de hoofdvraag. Het onderzoek heeft bijgedragen aan het opdoen van kennis over onderwerpen waar voorheen weinig tot niet eerder mee gewerkt is. Het betreft onder andere gebruikte methoden en technieken voor de realisatie van de microservice.

10.3.1.1 - Dataverzameling

De dataverzameling van het literatuuronderzoek heeft als volgt plaatsgevonden:

- ➢ Er wordt gebruik gemaakt van zowel **primaire** als **secundaire dataverzameling**. Dit betekent dat de informatie zelf opgedaan is of uit eerdere artikelen vergaard is. Dit is mede wegens bestaande documentatie en standaardisatie.
- ➢ Er wordt daarnaast gebruik gemaakt van **kwalitatieve data-analyse**. Dit betekent dat data voornamelijk verzamelt en verwerkt wordt in woorden en niet getallen en diagrammen. Het is data die niet zomaar berekend kan worden, maar juist verwerkt moet worden.
- ➢ Verder wordt er gebruik gemaakt van **descriptieve dataverzameling**. Hieronder verstaat men het verzamelen van data waar de nadruk ligt op het voeren van meer beschrijvend onderzoek en niet experimenteel onderzoek.

10.3.1.2 - Onderzoeksmethoden

Het onderzoek maakt gebruikt van verschillende onderzoeksmethoden, passend bij het type onderzoek. Hieronder staan de verschillende onderzoeksmethoden opgesomd, geclusterd per deelvraag.

- ➢ **Eerste deelvraag:** Hier wordt gebruik gemaakt van descriptief onderzoek, deskresearch en kwalitatief onderzoek.
- ➢ **Tweede deelvraag:** Hier wordt naast descriptief onderzoek, deskresearch en kwalitatief onderzoek ook gebruik gemaakt van toegepast onderzoek om de inzetbaarheid tussen Spring Boot en HAPI te polsen.
- ➢ **Derde deelvraag:** Hier wordt ook gebruik gemaakt van descriptief onderzoek, deskresearch, kwalitatief onderzoek en toegepast onderzoek met betrekking tot de REST-assured library en de syntax van dien.
- ➢ **Vierde deelvraag:** Hier wordt voornamelijk descriptief onderzoek, deskresearch en kwalitatief onderzoek gebruikt om kennis over verschillende design patterns te vergaren.

10.3.2 - Kwantitatief onderzoek

Het kwantitatieve onderzoek gaat in op migratiesnelheid en downtime met betrekking tot het Axon framework. Het onderwerp in kwestie viel buiten de scope van de opdracht, maar het heeft wel een bijdrage geleverd aan de kennis van de afstudeerorganisatie. De hoofdvraag van dit onderzoek die vooraf opgesteld is luidt:

Wat is de snelste wijze van migratie zonder downtime tijdens migratie van een Axon server naar een non-Axon server?

De hoofdvraag is ondersteunt door middel van een set van twee deelvragen. De deelvragen zijn hieronder op een rijtje gezet.

1. *Wat zijn de bottlenecks van een Axon server en op welke wijze is het mogelijk om downtime te vermijden tijdens het migreren van een Axon server naar een non-Axon server?*
2. *Wat is de invloed van de database grootte op de migratiesnelheid tijdens het migreren van een Axon server naar een non-Axon server?*

Bovenstaande deelvragen hebben een gepast, logisch antwoord geboden op de eerder genoemde hoofdvraag. Ondanks dat het onderwerp buiten de scope van de opdracht viel, heeft het afstudeerbedrijf er op lange termijn wel wat aan. Daarnaast is er nieuwe kennis verworven over het Axon framework: een nieuwe techniek waar nog niet eerder mee gewerkt is.

10.3.2.1 - Dataverzameling

De dataverzameling van het empirisch onderzoek heeft als volgt plaatsgevonden:

- Er wordt gebruik gemaakt van **primaire dataverzameling**. De informatie is opgedaan uit zelf onderzoeken en niet uit eerdere onderzoeken en andere wetenschappelijke bronnen.
- Er wordt daarnaast gebruik gemaakt van **kwantitatieve data-analyse**. Dit houdt in dat er veel gebruik gemaakt wordt van getallen en cijfers. Bovendien kunnen deze waarden ook weergeven worden in diagrammen. Dit vormt de basis van de bijpassende conclusies.

10.3.2.2 - Onderzoeksmethoden

Het onderzoek maakt gebruikt van verschillende onderzoeksmethoden, passend bij het type onderzoek. Hieronder staan de verschillende onderzoeksmethoden opgesomd, geclusterd per deelvraag.

- **Eerste deelvraag:** Hier wordt gebruik gemaakt van toegepast onderzoek en vergelijkend onderzoek om de mogelijkheid tot database migratie zonder downtime te polsen en de bottlenecks van de Axon server te identificeren.
- **Tweede deelvraag:** Hier wordt gebruik gemaakt van experimenteel onderzoek en vergelijkend onderzoek om de migratiesnelheid met verschillende hoeveelheden events aan de tand te voelen en gemiddelden te vergelijken.

10.4 - Software Architectuur Document

Kort na het uitvoeren van het kwalitatieve onderzoek, is het Software Architectuur Document (SAD) opgesteld. De inhoudelijke hoofdstukken opgenomen in het SAD zijn *Architecturele Eisen, Logical View, Implementation View* en *Deployment View.* Deze hoofdstukken bieden de lezer stapsgewijs kennis over de opgestelde architectuur, beginnend bij de eisen en eindigend bij de externe communicatie.

Nota Bene: Meer inhoudelijke informatie over de software architectuur is te vinden in het SAD.

10.5 - Teststrategie

Het testplan en het testrapport zijn belangrijke onderdelen van de teststrategie. Gedurende het testplan is voor de ontwikkelfase opgesteld. Het beschrijft alle verschillende soorten wijzen van testen die opgenomen zijn in het testproces, wat doorlopen is sinds de start van de realisatie. Dit heeft uiteindelijk geresulteerd in een testrapport. Het testrapport bevat de resultaten van het testen. Daarnaast bevat het verschillende testscenario's van de happy flow van alle functionaliteit met diens potentiële extensies.

Nota Bene: Voor een uitgebreide beschrijving en visualisatie van de verschillende soorten testen, kan men terecht bij het testplan 's1115001_Testplan_2.0.pdf'.

10.6 - Microservice

Het belangrijkste eindproduct van de afstudeerperiode is de microservice die gebouwd is ter bevordering van de microservice architectuur die de afstudeerorganisatie wilt bereiken, na opbreking van diens monoliet applicatie.

Voor de realisatie van de microservice is er ook een kleine proof of concept gebouwd voor het polsen van de mogelijkheid tot gebruik van generics in JpaRepositories. De proof of concept heeft bijgedragen om te kijken of de application flow meer generiek kon worden.

Daarnaast is er ook een paper prototype gemaakt. Het paper prototype ging in op het visitor pattern en de toepasbaarheid hiervan binnen de microservice. Het betreft een design pattern die gebruikt wordt voor het mappen van verschillende metingen.

De microservice zelf is opgebouwd gedurende twaalf weken tijd. Deze twaalf weken zijn opgedeeld in vier sprints van drie weken. Voor meer inhoudelijke informatie wordt men doorverwezen naar het opbrengsten hoofdstuk.

11 - Opbrengsten

Hoofdstuk 11 gaat in op de opbrengsten. Het betreft de uitwerkingen en oplossingen van de beroepsproducten die globaal besproken zijn in hoofdstuk 10.

11.1 - Onderzoek

Dit hoofdstuk staat in het kader van het uitgevoerde onderzoek. Bij beide onderzoeken wordt eerst de hoofdvraag opnieuw gepresenteerd. Vervolgens worden de verschillende deelconclusies aangekaart. Uiteindelijk wordt er afgesloten met de eindconclusie en/of aanbevelingen.

Nota Bene: Meer informatie omtrent het onderzoek en de uitwerking van dien, kan men vinden in de onderzoeksrapporten.

11.1.1 - Verantwoording

Gedurende de afstudeerperiode zijn er twee onderzoeken uitgewerkt. De keuze tot het uitwerken van twee onderzoeken volgde bij nader overleg tijdens het ontmoetingsgesprek met de afstudeerdocent en bedrijfsbegeleider. Op basis hiervan is er besloten om een nieuw empirisch onderzoek op te stellen met kwantitatieve data-analyse. Verder wijkt het eerste onderzoek af zoals deze beschreven staat in het voorstel: na de verstrekte feedback van de afstudeer pitch is het onderzoek herzien en aangescherpt.

Voor beide onderzoeken geldt dat de deelvragen SMART zijn opgesteld en dat ze bovendien een passend antwoord geven op de SMART hoofdvraag.

Beide onderzoeken hebben gebruik gemaakt van erkende onderzoeksmethoden om data te vergaren: denk bijvoorbeeld aan deskresearch of toegepast onderzoek. Een totaallijst is terug te vinden in de desbetreffende onderzoeken en hoofdstuk 10.

De gebruikte bronnen hebben ten alle tijden betrekking op het onderzoek en er is verwezen in APA-stijl. Beide onderzoeken verstrekken bovendien een literatuurlijst aan haar lezers waar deze bronnen zijn opgesomd.

Zoals eerder benoemd, wordt er zowel gebruik gemaakt van kwalitatieve als kwantitatieve data-analyse. Kwalitatief onderzoek is terug te vinden in het literatuuronderzoek, terwijl kwantitatief onderzoek is terug te vinden in het empirisch onderzoek.

Alle deelconclusies die getrokken zijn, verhouding zich goed tot de desbetreffende deelvragen en geven een logisch antwoord op diens hoofdvraag. Er is teruggekeken naar het onderzoek om te zorgen dat er niet spontaan extra, irrelevante informatie beschikbaar gesteld wordt.

11.1.2 - Literatuuronderzoek

Voorafgaand is er met stakeholders besproken welke methoden en technieken onderdeel gaan uitmaken van de microservice. Het literatuuronderzoek heeft de focus gelegd op deze libraries en frameworks en de inzetbaarheid van dien gedurende het ontwikkeltraject. Ook is er onderzoek gedaan naar design patterns. Het onderzoekscomponent met de onderzoeksvragen valt terug te lezen in hoofdstuk 10.3.1.

11.1.2.1 - Conclusie eerste deelvraag

Er is geconcludeerd dat een microservice een belangrijk component is wat deel uitmaakt van een veel grotere microservice architectuur. Er is onderlinge communicatie en er is sprake van *flexibiliteit, schaalbaarheid, continuïteit* en *herbruikbaarheid.* Bovendien zijn ze ook robuust. Realisatie binnen Java wordt makkelijk gemaakt door middel van Spring: het biedt vele features aan haar gebruikers en is bovendien veilig en snel.
(Amazon Web Services, 2022) (Altvater, 2019) (SentinelOne, 2018) (Spring, 2022) (Spring, 2022)

11.1.2.2 - Conclusie tweede deelvraag

Er is geconcludeerd dat HAPI een compleet uitgewerkte implementatie biedt van de internationale HL7-FHIR standaard voor de overdracht van medische data. Het gebruik van de open-source library is mogelijk in combinatie met Spring Boot door middel van DTO's. Mappers en adapters kunnen gebruikt worden voor omschakeling tussen models en DTO's. Hiermee wordt onnodige problematiek vermeden.
(Davis, 2018) (HAPI FHIR, 2022) (HAPI FHIR, 2022) (Waterman, 2020) (Baeldung, 2021)

11.1.2.3 - Conclusie derde deelvraag

Er is geconcludeerd dat de in 2010 opgerichte REST-assured library de testability bevordert van REST services. Het is bruikbaar met Maven en Gradle en dus ook Spring Boot. De syntax van REST-assured werkt door middel van zogeheten method chaining: krachtige functies worden als een enkele statement geschreven:
'given().when().get().then().assertThat();'. Dit maakt het overzichtelijk en simpel te leren.
(Hamilton, 2021) (Bealdung, 2020) (Hamcrest, 2019) (Hamilton, 2021) (Bealdung, 2020) (Haleby, 2022)

11.1.2.4 - Conclusie vierde deelvraag

Er is geconcludeerd dat vaak terugkerende, complexe problemen oplosbaar zijn door design patterns die fungeren als blueprints en op deze manier standaardoplossingen bieden binnen de wereld van software engineering. Ze gaan onder andere ook code smells tegen, wat betekent dat dit leidt tot minder toekomstige problematiek, zoals extensies of onderhoud. Binnen het huidige project zijn onder andere *singletons, adapters, commands, builders* en *composites* te gebruiken.
(Refactoring Guru, 2022) (Refactoring Guru, 2022) (Refactoring Guru, 2022) (Refactoring Guru, 2022) (Refactoring Guru, 2022) (Refactoring Guru, 2022) (Refactoring Guru, 2022)

11.1.2.5 - Eindconclusie

Concluderend, een microservice is een vitaal onderdeel van een microservice architectuur en is te realiseren binnen Java door middel van het Spring Boot framework. Spring stelt veel features beschikbaar voor haar gebruikers en is ook te gebruiken in combinatie met HAPI: een open-source Java library met een complete implementatie van de HL7-FHIR standaard voor gegevensoverdracht in de zorg. Dit kan door middel van DTO's, adapters en mappers. REST-assured is een java library die het mogelijk maakt om krachtige tests te schrijven voor REST services in slechts een enkele statement door middel van method chaining. Het is overzichtelijk en heeft een kleine learning curve. Design patterns zijn blueprints voor vaak terugkerende problematiek binnen de IT en verminderen code smells in de productiecode. Dit maakt bijvoorbeeld toekomstige extensie en onderhoud makkelijker. Binnen de context van het huidige project is het onder andere mogelijk om gebruik te maken van *singletons, adapters, commands, builders* en *composites*.

11.1.3 - Empirisch onderzoek

Het empirisch onderzoek is in een later stadium uitgevoerd gedurende de afstudeerperiode. Het is naast de realisatie uitgevoerd: het onderwerp heeft geen betrekking op de opdracht, aangezien het buiten de scope valt. Er was dus geen noodzaak om het onderzoek vooraf aan de opdracht uit te voeren. De onderzoeksvragen zijn terug te vinden in het onderzoekscomponent, wat onderdeel uitmaakt van hoofdstuk 10.3.2.

11.1.3.1 - Conclusie eerste deelvraag

Er is geconcludeerd dat Axoniq een demo applicatie verstrekt om commands en events te handelen met een bijgeleverde Axon server. Hier zitten daarentegen wel wat limitaties aan: de Axon server heeft een *soft limit* van 10.000 commands per issue. Meerdere issues maakt de event store bovendien trager. Ook kan een te grote query leiden tot een *resource exhaustion*. Door de grote hoeveelheid op te vragen events te splitsen over meerdere queries wordt de Axon server minder belast en kan *resource exhaustion* vermeden worden. De data kan vervolgens gebundeld worden en in een keer naar de database gemigreerd worden. (AxonIQ, 2022) (Fowler, 2005)

11.1.3.2 - Conclusie tweede deelvraag

Er is geconcludeerd dat het mogelijk is om de totale tijd van database migratie onder invloed van de hoeveelheid events te meten in milliseconden door gebruik te maken van de *Spring AOP Expression Language*. De migratiesnelheid nam toe naarmate de hoeveelheid events om te migreren toenam, maar de gemiddelde migratiesnelheid per event is juist afgenomen. Grotere hoeveelheden events in een keer migreren duur gemiddeld dus minder lang per event. (Baeldung, 2022) (Baeldung, 2022)

11.1.3.4 - Eindconclusie

Concluderend, door de aanwezige hoeveelheid events in zo groot mogelijke heaps naar de database in kwestie te migreren, duurt de migratie over het algemeen minder lang. Dit komt door de afname van de gemiddelde migratietijd per event, naarmate de hoeveelheid events stijgt. Wel moet men rekening houden met de bottlenecks die zich mogelijk kunnen voordoen bij de Axon server, zoals de *soft limit* en *resource exhaustion*.

11.2 - Microservice

Dit hoofdstuk beschrijft de microservice en vitale onderdelen van dien. Er wordt onder andere gesproken over het bouwen onder architectuur van de microservice. Daarnaast wordt belangrijke functionaliteit van de microservice onder de loep gehouden.

11.2.1 - Software Architectuur

De microservice is gerealiseerd onder architectuur. Deze architectuur is opgesteld in een Software Architectuur Document (SAD). Hoofdstuk 10.4 heeft de verschillende onderwerpen van het SAD geïntroduceerd.

Nota Bene: verdiepende informatie over de software architectuur kan men vinden in het Software Architectuur Document.

11.2.1.1 - Implementation View

In dit hoofdstuk wordt een beknopte samenvatting gegeven over de implementation view. De implementation view biedt een technische invulling van de logical view. (zie software architectuur document). Dit is mogelijk aan de hand van een package diagram, een applicatiemodel en toelichting van belangrijke componenten en frameworks.

De basepackage is *nl.gerimedica.measurements*. Hieronder vallen de volgende folders: *clients, config, models, dtos, repositories, exceptions, utils* en *resources.db.migration*. Een package diagram is aanwezig in het architectuur document.
DTO's en models zijn gesplitst om consistentie te behouden met een eerder opgebouwde microservice.
De lagenstructuur is opgebouwd aan de hand van een applicatiemodel. Het applicatiemodel beschrijft een generieke application flow voor verschillende typen measurements. Dit gaat gebruikt worden voor alle metingen binnen gm-measurements. Ook het applicatiemodel maakt onderdeel uit van het architectuur document.
De belangrijkste componenten en frameworks binnen het afstudeerproject zijn Ysis-Integration-Boot (een bedrijfseigen bundeling van belangrijke componenten), Spring Boot, Swagger en Flyway. Deze onderdelen zijn gebruikt tijdens de realisatie van de microservice.

11.2.1.2 - Deployment View

De gm-measurements service wordt altijd aangesproken via de gm-gateway: dit is een node waar alle verbindingen via verlopen. Naast gm-gateway is er ook CAS. De CAS-server is een authenticatie server voor het authenticeren van alle Gerimedica werkproducten.
Verder zijn er ook interne werkproducten van Gerimedica zelf aanwezig: gm-code-lists, Ysis Inzicht voor statistieken, Ysis (het EPD) en het ECD.
Tenslotte zijn er nog externe werkproducten van partners. Hier haalt Gerimedica onder andere haar externe metingen vandaan. Dergelijke werkproducten zijn Ons (Nedap), MijnCaress (Pinkroccade), SDB ECD (SDB-groep), Lable Care (Lable) en PUUR (Ecare).
Een deployment diagram met visualisatie is beschikbaar gesteld in het software architectuur document.

11.2.1.3 - Ontwerpverantwoording

De microservice is gebouwd onder architectuur zoals deze beschreven staat in het Software Architectuur Document *'s1115001_Software_Architectuur_Document_3.0.pdf'*. Sinds versie 3.0 zijn er meerdere wijzigingen doorgevoerd in het ontwerp. Deze staan hieronder bondig besproken en in de ontwerp alternatieven.

Alle onderwerpen in het document, waaronder bovengenoemde semantics, applicatiemodel en externe verbindingen zijn opgesteld aan de hand van standaarden binnen het afstudeerbedrijf. Dit is onder andere de wens van de stakeholders om onderlinge consistentie te realiseren tussen de nieuwe en toekomstige microservices.

Er is op zoek gegaan naar feedback bij leden van het developer team. De feedback van deze stakeholders is vervolgens verwerkt in het ontwerp: dit blijkt uit de versiebeheer van het document. Zo is de application flow van de microservice verandert van een non-generieke opzet naar een generieke opzet: deze keuze is besproken met een stakeholder. Daarnaast is het eerder toegepaste state pattern vervangen door een visitor pattern wegens concurrency (zie hoofdstuk 11.2.2.2).

Het proof of concept en het dynamische prototype worden besproken in hoofdstuk 11.2.2. Het proof of concept gaat over de toepassing van generics in een JpaRepository. Het paper prototype test de werking van het visitor pattern.

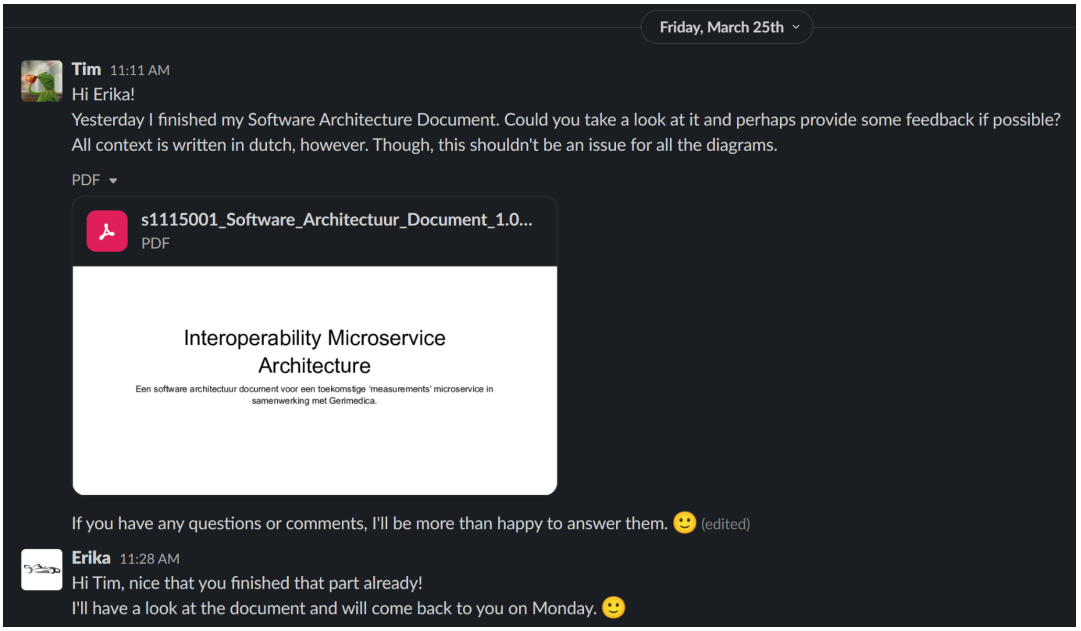
Nota Bene: Wijzigingen aan het Software Architectuur Document zijn uitgebreid opgesomd in de versiebeheer van dien.

11.2.1.4 - Ontwerphouding

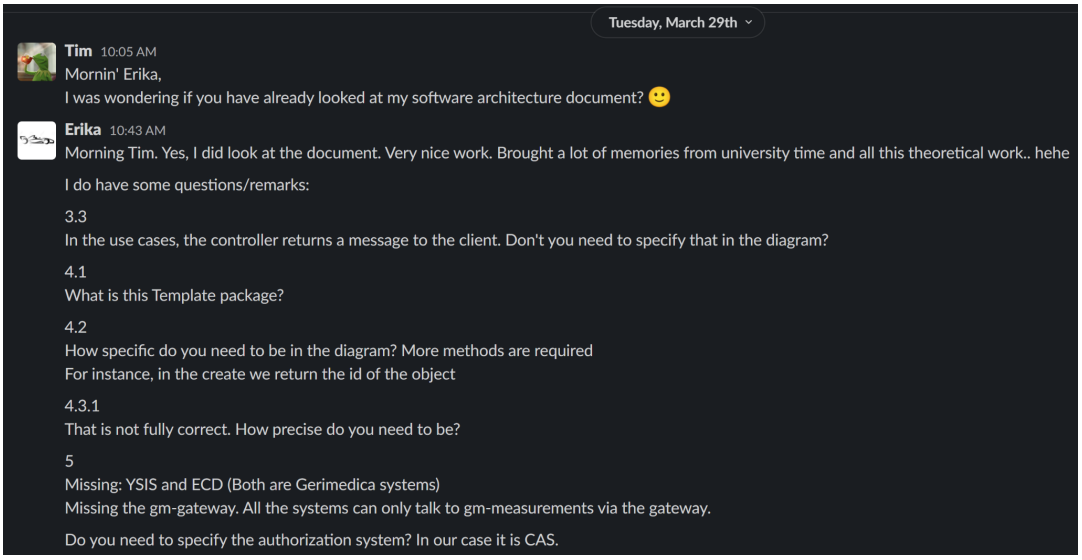
Gedurende het opzetten van het ontwerp is er, zoals genoemd in hoofdstuk 11.2.1.3, rekening gehouden met verschillende stakeholders: het ontwerp is ontstaan uit verschillende iteraties die stuk voor stuk aangepast zijn aan de hand van verstrekte feedback.

Hieronder zijn drie afbeeldingen weergeven omtrent gegeven feedback. De eerste afbeelding laat zien hoe er zelfstandig gevraagd is om feedback. De afbeelding die volgt is de ontvangen feedback. De derde afbeelding toont de versiebeheer van het software architectuur document na het verwerken van de ontvangen feedback.

De vierde en laatste afbeelding toont het initialiseren van een discussie omtrent de validation van ontvangen data in de applicatie en de wijze van implementatie van dien. Het overleg met de stakeholder heeft geleid tot een wijziging in de validatie.



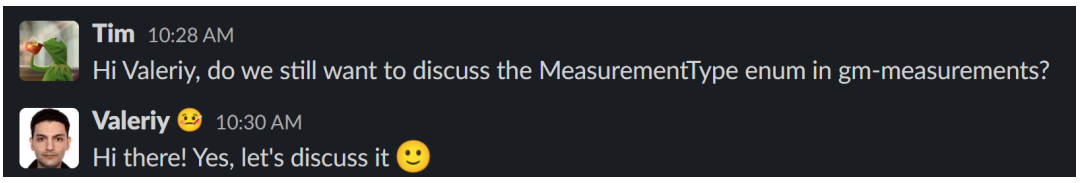
Figuur 11.2.1.4a - Het vragen van feedback aan een stakeholder op 25 maart 2022.



Figuur 11.2.1.4b - De ontvangen feedback omtrent het ontwerp op 29 maart 2022.

1.1	Pre-final	29-03-2022	Tim van Hal	Verwerken feedback vanuit bedrijf.

Figuur 11.2.1.4c - De verwerkte feedback aantoonbaar in de versiebeheer van het ontwerp document.



Figuur 11.2.1.4d - Initiatie van een discussie over implementatie alternatieven.

11.2.1.5 - Ontwerp Alternatieven

Voor het ontwerp van de microservice zijn er meerdere alternatieven afgewogen. Dergelijke alternatieven met betrekking tot het ontwerp zijn hieronder weergeven.

De eerste alternatieven gaan in op de application flow: generic of non-generic. Een optie is om voor elk type meting een aparte service en persistence te maken. De optie die hier tegenover staat is een generieke opzet met een enkele service en persistence waar alle type metingen gebruik van kunnen maken.
Het eerste ontwerp had voor elk type meting aparte controllers, services en persistences. Dit is, zoals eerder benoemd in de ontwerpverantwoording, in een later stadium veranderd naar de generiek opzet na overleg met de stakeholders.

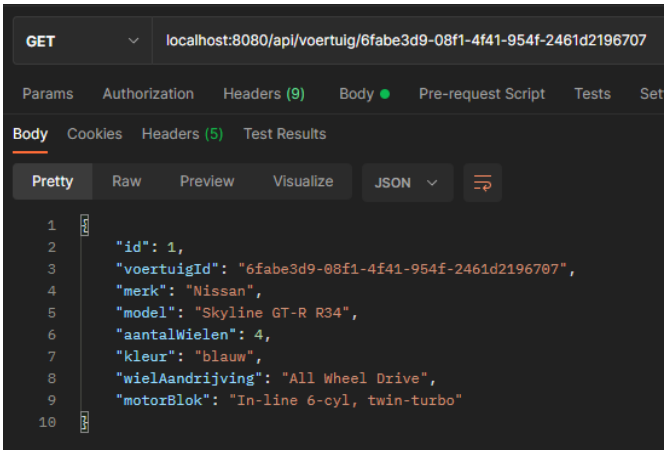
Verdere alternatieven beschrijven het gebruik van verschillende design patterns binnen de microservice. De eerste mogelijkheid om verschillend gedrag per type meting te realiseren was het gebruik van een state pattern. De tweede mogelijkheid was de toepassing van een visitor pattern.
Uiteindelijk is het toegepaste state pattern vervangen door het visitor pattern wegens ‘concurrency’ binnen de microservice: het state pattern verandert namelijk de interne staat van de applicatie, wat het visitor pattern niet doet.

11.2.2 - Prototypes

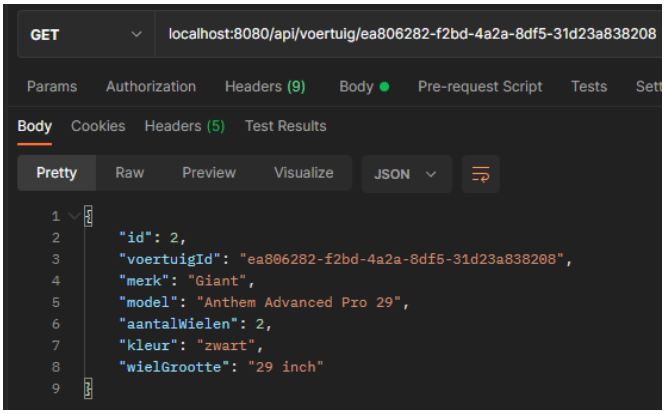
Dit hoofdstuk gaat in op de prototypes die gemaakt zijn voor bepaalde onderdelen van de microservice. Allereerst wordt er een proof of concept aangekaart met betrekking tot een generieke opzet van de persistence. Vervolgens wordt er gesproken over een paper prototype met betrekking tot het visitor pattern wat geïmplementeerd is.

11.2.2.1 - Generics Proof of Concept

De proof of concept die gemaakt is ging in op het gebruik van generics binnen de JpaRepository. Een eerder ontwerp bevatte aparte klassen en endpoints gemaakt per measurement, maar tijdens de realisatie van de eerste story is er besloten om een meer generieke application flow te realiseren. Om te testen of dit mogelijk was met een JpaRepository, is er een proof of concept opgesteld met verschillende voertuigen als onderwerp. Hieronder zijn twee verschillende weergaven met betrekking tot de proof of concept. Het eerste figuur toont het resultaat van het opvragen van een auto, terwijl het tweede figuur met dezelfde endpoint een fiets opvraagt.



Figuur 1.2.2.1a - Het opvragen van een auto binnen de proof of concept.



Figuur 1.2.2.1b - Het opvragen van een fiets binnen de proof of concept.

Een GET-request komt binnen bij de controller waar een id wordt gebruikt om een voertuig op te vragen. De controller stuurt dit vervolgens door naar de service die op zijn beurt verbinding maakt met de repository. De methode in de repository heeft een returnwaarde genaamd '*<T extends Voertuig> Voertuig*' (zie appendix). Dit houdt in dat deze methode

zowel een fiets als een auto kan terugsturen, zolang deze maar een voertuig is. Het resultaat van het sturen van de GET-request is in het eerste geval een auto en in het tweede geval een fiets.
Aangezien er ondervonden is dat het mogelijk is om generics in te zetten bij de repository, betekent dit dat dit ook mogelijk is voor de microservice. De inzetbaarheid van dien is terug te vinden in het ontwerp.

Nota Bene: code snippets van de proof of concept maken onderdeel uit van Appendix A.

11.2.2.2 - Paper Prototype

Naast de realisatie van een proof of concept, is er ook een dynamisch prototype gemaakt. Het paper prototype ging specifiek in op het visitor pattern: een design pattern wat verschillende typen objecten bezoekt en vervolgens aan elk type object verschillend gedrag toekent.

De reden dat er gekozen is om hierover een paper prototype te maken is, omdat er besproken is met de stakeholders dat er vervanging moest komen voor het toegepaste state pattern. Het state pattern was aan vervanging toe wegens het begrip concurrency: als request B op hetzelfde moment binnenkomt als request A en request B verandert de interne staat van de microservice, dan geeft request A mogelijk een ongewenst antwoord terug. Een oplossing voor dit probleem is de introductie van het visitor pattern, waar het paper prototype voor gemaakt is. Hieronder is een afbeelding weergeven van het prototype.

Nota Bene: verdere afbeeldingen met betrekking tot de verschillende uitkomsten van het paper prototype zijn terug te vinden in de appendix.

Figuur 11.2.2.2a - Paper prototype van het visitor pattern.

Bovenstaande prototype begint met een dier die binnenkomt. Vervolgens gaat de visitor op op bezoek bij het desbetreffende dier. Vervolgens wordt er een keuze gemaakt welke dier gemaakt wordt op basis van de binnengekomen data. In het geval van het prototype wordt er eerst een aap gemaakt, vervolgens een beer en ten slotte een cheeta. Dit paper prototype heeft bijgedragen om een beter beeld te scheppen over de werking en implementatie van het visitor pattern. Dit is later toegepast in de microservice: dit is terug te vinden in het ontwerp.

11.2.3 - Integratie

Zoals eerder benoemd, wordt binnen Gerimedica momenteel gewerkt aan opbreking van de monoliet applicatie en het opzetten van een microservice architectuur. Integratie binnen het bestaande systeem is een onderwerp dat meermaals is teruggekomen gedurende de afstudeeropdracht.
Allereerst is er rekening gehouden met consistentie: de basis van de microservice verhoudt zich hetzelfde als een eerder gebouwde microservice en toekomstige microservices om consistentie onderling te waarborgen. Dit is van belang voor verschillende stakeholders binnen het bedrijf voor onderhoud en eventuele, toekomstige uitbreiding.
Voor de realisatie van de microservice is er ook gebruik gemaakt van een bundeling van libraries en frameworks genaamd ysis-integration-boot (zie hoofdstuk 8.2.1). Deze bundeling is nodig om de applicatie binnen de bestaande omgeving aan de praat te krijgen.
Externe communicatie met de gm-measurements microservice is beschreven in de deployment view van het Software Architectuur Document (zie hoofdstuk 11.2.1.2). Hieronder valt onder andere een authenticatie server die nodig is om toegang te krijgen tot de nieuwe microservice en een andere microservice voor het opvragen van een lijst standaardwaarden die gebruikt wordt binnen de nieuwe microservice.

11.2.3.1 - Integratie Alternatieven

De integratie heeft ook verschillende alternatieven die gebruikt konden worden. Deze staan hieronder opgesomd.
Zoals hierboven genoemd, is er gebruik gemaakt van ysis-integration-boot om er voor te zorgen dat alle nodige libraries en frameworks beschikbaar waren binnen de microservice. Daarentegen was het ook mogelijk om alle benodigde libraries handmatig als dependencies toe te voegen. Ondanks dat dit ongebruikte libraries tegen gaat, is het alsnog omslachtig en maakt het de pom.xml bovendien onleesbaar. Vandaar dat er gekozen is om gebruik te maken van ysis-integration-boot.
Daarnaast wordt er ook gebruik gemaakt van een andere microservice om bepaalde standaardwaarden omtrent medische gegevens in lijstvorm op te vragen (zie gm-code-list in deployment view). Deze lijsten met waardes worden vervolgens gecached. Deze lijsten hadden daarentegen ook hardcoded toegevoegd kunnen worden, maar dan weet men niet of de waarden daadwerkelijk kloppen: ze kunnen bijvoorbeeld outdated zijn.

11.2.4 - Migratie

Gedurende de afstudeerperiode komt migratie in meerdere vormen en maten terug: het komt terug binnen de realisatie van de microservice en voor een onderzoek. Beide onderwerpen worden hieronder besproken.
Voor het opslaan van de vele hoeveelheid data die er aan te pas komt bij de metingen, moet er een database beschikbaar zijn. Na overleg met de stakeholders is er besloten om een postgres database op te stellen voor deze datasets.
De structuur van de database is uiteraard opgebouwd aan de hand van meerdere scripts (.sql-bestanden). Deze zijn aanwezig als resource in het project. De reden hiertoe zit hem in een tool die gebruikt wordt voor de microservice: flyway. Flyway maakt het mogelijk om database scripts uit te voeren zodra deze voor het eerst aanwezig zijn in het project, of zodra ze in een later stadium aangepast worden. Op deze manier is het ten alle tijden

makkelijk om de database in te richten, ook tijdens migratie of als de environment een reset krijgt..
Daarnaast is er onderzoek gedaan naar de migratiemogelijkheden rondom het Axon framework (zie hoofdstuk 11.1.3). Het oude systeem werkt namelijk met Axon Events, maar Gerimedica wilt deze events graag opslaan in een non-Axon server (namelijk postgres). Er is daarom onderzoek uitgevoerd richting de migratie van een Axon server naar een postgreSQL database.

Nota Bene: De daadwerkelijke migratie viel buiten de scope van de opdracht wegens de enorme omvang van dien.

11.2.4.1 - Migratie alternatieven

Ook voor migratie zijn er verschillende alternatieve afgewogen. De desbetreffende alternatieven zijn hieronder weergeven.
Zoals eerder besproken, wordt er gebruik gemaakt van flyway voor het automatisch uitvoeren van de sql-scripts om de database op juiste wijze in te richten. Een alternatief hiervoor was het handmatig uitvoeren van de SQL-statements in een query. Dit wordt daarentegen vervelend, omdat dit meermaals moet gebeuren: er zijn meerdere environments en de database met testdata wordt zo nu en dan gewiped. Het gebruik van flyway is daarom ideaal.

11.2.5 - Kwaliteitskenmerken

Gedurende de afstudeeropdracht en het opstellen van de microservice, is er rekening gehouden met verschillende kwaliteitskenmerken. Bovendien zijn ze ook aan de orde gekomen bij gesprekken met stakeholders. De kwaliteitskenmerken die terug te vinden zijn in de gerealiseerde microservice zijn hieronder opgesomd.

11.2.5.1 - Functional Suitability

Functional Suitability (ofwel functionele geschiktheid) gaat in op het feit of de applicatie wel voldoet aan alle behoeften tot functionaliteit van diens gebruikers. De microservice vervangt een bestaand systeem, wat betekent dat alle bestaande functionaliteit aanwezig moet zijn in de nieuwe microservice (*functional completeness*). Deze acties moeten daarnaast ook het gewenste resultaat hebben (*functional correctness*). Verder moet alle functionaliteit wel praktisch nut hebben (*functional appropriateness*). De functional completeness en correctness zijn besproken met de stakeholders en is mede gerealiseerd door het testen van de service. De complete lijst aan functionaliteit is terug te vinden in het acceptatieplan. Functional appropriateness is gerealiseerd door een kritische blik te werpen op het praktisch nut van alle gewenste functionaliteit vooraf. Dit is binnen Team B besproken.

11.2.5.2 - Compatibility

Compatibility (ofwel uitwisselbaarheid) beschrijft de uitwisselbaarheid van informatie met andere werkproducten en het uitvoeren van diens functionaliteit in de bestaande omgeving. Er is gelet op het feit of de microservice goed kan functioneren binnen een gedeelde omgeving (*Co-existence*). Verder is er ook rekening gehouden met het uitwisselen van gegevens (*interoperability*). Interoperability is een vaak terugkerende term binnen Gerimedica. Co-existence is mogelijk door middel van Kubernetes. Hier draaien meerdere

instanties van services over meerdere omgevingen. Dergelijke omgevingen bestaan onder andere wegens het ontwikkelen/testen van de software of bijvoorbeeld het in productie gaan van de service. Verschillende omgevingen hebben verschillende *pods* binnen Kubernetes. Dit is per omgeving aangegeven met behulp van de 'replicaCount' variabele.
Interoperability handelt de microservice ook af. Zo komt het in aanraking met meetgegevens van externe cliënten, maar het verstrekt bijvoorbeeld ook de meetgegevens aan een intern werkproduct wegens monitoring redenen. Een complete lijst is terug te vinden in het hoofdstuk over de deployment view van de software architectuur.

11.2.5.3 - Security

Security (ofwel beveiligbaarheid) gaat in op de mate van databescherming tegen onbevoegden en het verlenen van toegang voor geautoriseerden.
De service houdt daarom rekening met de toegankelijkheid voor geautoriseerden (*Confidentiality*) en het aanpassen van data mits de gebruiker bevoegd is (*Integrity*). Dit kan enkel als de gebruiker is ingelogd en bovendien bevoegd is om een specifieke handeling uit te voeren. Dit is mogelijk door middel van de authenticatie server. Hiermee kan een gebruiker bewijzen dat hij daadwerkelijk is wie hij voordoet (*Authenticity*).
Verder wordt het bewijzen dat acties plaats hebben gevonden (*Non-repudiation*) en de traceerbaarheid van acties naar een entiteit (*Accountability*) mogelijk door het loggen van alle handelingen die uitgevoerd worden op de service. Deze logs worden opgeslagen en zijn ten alle tijden beschikbaar.

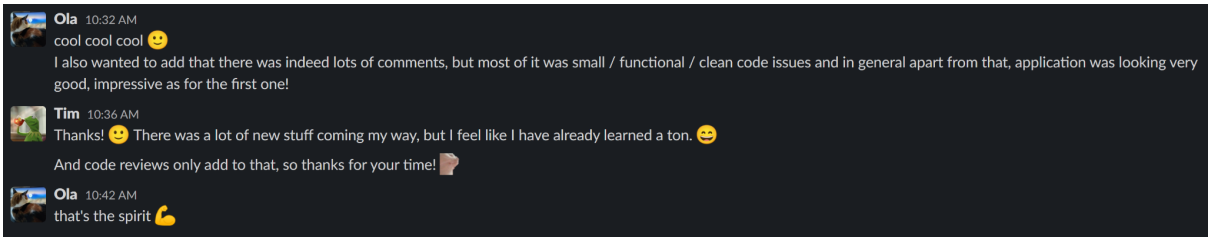
11.3 - Testrapport

Tijdens de afstudeerperiode is er een teststrategie gevolgd (zie hoofdstuk 10). Deze teststrategie is vooraf gedefinieerd in het testplan. Vervolgens heeft het testen tijdens de ontwikkelfase geresulteerd in het testrapport. Het testrapport legt de focus op de resultaten die voortvloeien uit het uitvoeren van de teststrategie, zoals deze beschreven staat in het testplan.
Functional testing heeft plaatsgevonden met behulp van Postman. Om alle verschillende endpoints te testen zijn de tests opgeslagen. Deze tests zijn vervolgens gebundeld in een collectie binnen Postman, waar collega's gebruik van hebben gemaakt.
Unit testing en *integration testing* hebben plaatsgevonden met behulp van JUnit 5 en REST-assured. De unit tests zijn onder andere gerealiseerd met Mockito om bepaalde onderdelen te mocken. Vervolgens heeft er een vergelijking plaatsgevonden met een assert.
De *automatische teststraat* heeft, naast het automatisch bouwen en deployen, ook gediend om de code te testen. Het heeft in een geïsoleerde omgeving de unit tests en integration tests gedraaid. De statistieken van alle pipelines zijn ook terug te vinden in de bijlagen. Desondanks het percentage gefaalde pipelines, hebben veel van deze pipelines bijgedragen aan het repareren van dien.
Voor alle functionaliteit zijn uiteindelijk testscenario's gemaakt. Dergelijke testscenario's bevatten de happy flow en ook, mits van toepassing, de verschillende extensies.
Alle testresultaten zijn terug te vinden in Appendix C.

Nota Bene: verdiepende informatie/context over onderdelen van de testresultaten zijn terug te vinden in het testrapport, wat onderdeel uitmaakt van het afstudeerdossier.

11.4 - Leren leren

Tijdens het afstudeertraject is de focus ook gelegd op het ‘leren leren’. Dit is gebeurd aan op drie fronten: zelfbeeld, zelfredzaamheid en intrinsieke motivatie.
Om het zelfbeeld te versterken zijn er actief maatregelen getroffen. Zo hebben er code reviews plaatsgevonden om code kwaliteit en skills nog verder te verbeteren. Er is bovendien zelf benadrukt dat deze code reviews hieraan bijdragen. Het bewijs van dien is hieronder weergeven in figuur 11.4a.

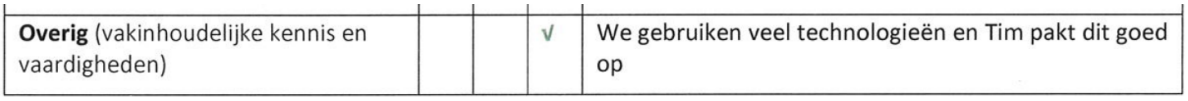


Figuur 11.4a - Actief verbeteren van het zelfbeeld en het erkennen van dien.

Naast het zelfbeeld, is zelfredzaamheid een belangrijke factor. Bedrijfsbegeleider Michiel Bruins heeft in de evaluatie een goed gegeven voor zelfstandigheid. Dit staat hieronder weergeven.

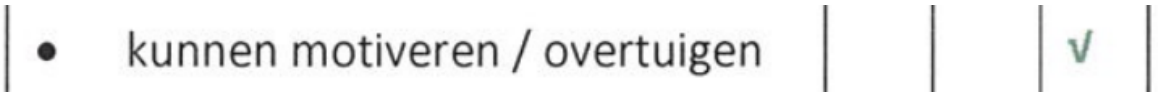
Figuur 11.4b - Goede zelfstandigheid aldus de evaluatie.

Het eigenhandig verzamelen van hulpbronnen en het effectief inzetten van dien blijkt onder andere ook uit het snel eigen maken van de grote technology stack van Gerimedica. De evaluatie van de bedrijfsbegeleider benadrukt dit:



Figuur 11.4c - Het oppakken van de technology stack aldus de evaluatie.

Tenslotte komt intrinsieke motivatie terug in de vorm van motivatie en een leergierige houding. De mogelijkheid tot motiveren en overtuigen is goed aldus de bedrijfsbegeleider (zie figuur 11.4d). Daarnaast is de leergierige houding ook terug te vinden in figuur 11.4a.

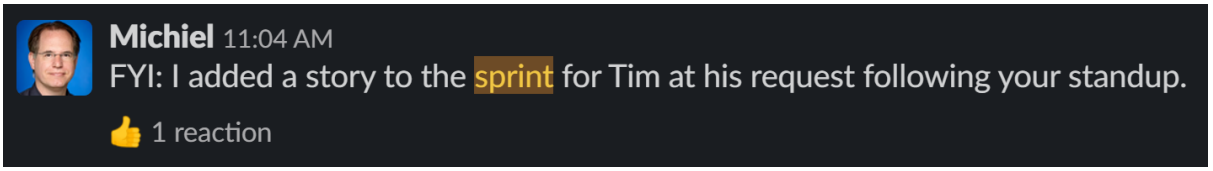


Figuur 11.4d - Goed kunnen motiveren / overtuigen aldus de evaluatie.

Nota Bene: Beide evaluaties van de bedrijfsbegeleider zijn in zijn geheel terug te vinden in het afstudeerdossier.

11.5 - Professioneel werken

Het afstudeertraject is uitgevoerd door middel van professioneel werken. Het eerste waaruit dit blijkt is de probleem analyse: hoofdstuk 5 beschrijft het probleem achter het probleem vanuit diverse views en gebruikt dit voor de oplossing. Zo is er gekeken vanuit de bril van de developers en van de product owner.
Het planmatig werken komt terug in de vorm van de methoden van aanpak en specifieke onderdelen. Zo is er voor het gehele project en alle beroepsproducten, waaronder bijvoorbeeld onderzoek en ontwerpen, de watervalmethode gebruikt. Voor een specifiek onderdeel (de ontwikkelfase) is de ontwikkelmethode Scrum toegepast in sprints van drie weken. Dit is overzichtelijk weergeven in de planning. Verder kaart figuur 11.5a de toevoeging van een story aan een sprint toe.



Figuur 11.5a - De toevoeging van een story aan het sprint board.

Bovenstaand beschreven planning gaat hand in hand met methodisch werken. Er zijn daarentegen aanpassingen geweest aan de planning: zo zijn er direct aanpassingen gemaakt na feedback van de pitch en heeft er een wijziging plaats gevonden door toevoeging van een tweede (kwantitatief) onderzoek. Het volgen van de planning was positief voor het methodisch werken, aangezien het een vaste workflow afdwong: onderzoek -> ontwerp -> realiseren & testen. Daarnaast zijn de deadlines ook nageleefd: alle formulieren zijn netjes opgevraagd en ingeleverd. Ook is er actief deelgenomen aan de intervisie. Dit blijkt uit de aanwezigheid van de desbetreffende documenten in het afstudeerdossier. Daarnaast was Michiel Bruins ook erg positief over stiptheid. Dit blijkt uit figuur 11.5b.

Stiptheid				
• afspraken nakomen • analytisch vermogen • inzet / werktempo • kritisch t.a.v. eigen functioneren			√ √ √ √	Tim is enthousiast en gedreven.

Figuur 11.5b - Stiptheid aldus de evaluatie van de bedrijfsbegeleider.

Volgens figuur 11.5c is er goed gecommuniceerd tijdens het afstudeertraject. Dit blijkt uit alle bulletpoints die opgenomen zijn binnen het kader communicatie van de evaluatie. Het snel kunnen inwerken werd ook als positief ervaren en staat weergeven in figuur 11.5d.

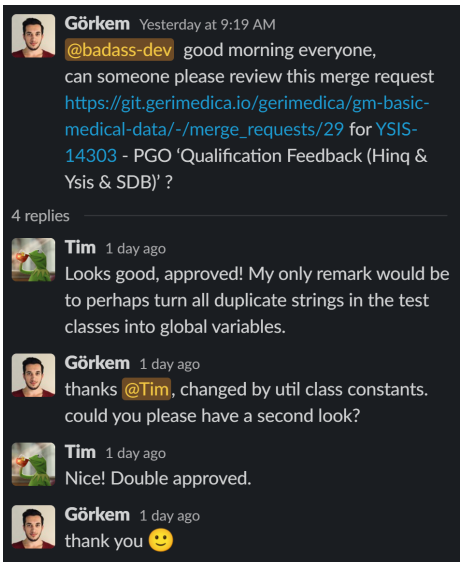
Communicatie			
• luisteren		√	
• gedachten formuleren		√	
• contact met medewerkers	√		Tim werkte in het begin nog erg solistisch, maar sinds hij inhoudelijk meedoet in de sprint (incl. stand-ups) is er veel meer interactie en samenwerking.
• contact met de leiding		√	
• samenwerken	√		
• omgaan met kritiek		√	
• schriftelijke vaardigheden		√	
• documenteren		√	

Figuur 11.5c - De communicatie aldus de evaluatie van de bedrijfsbegeleider.

• kan snel inwerken		√	Complexe codestack snel eigen gemaakt

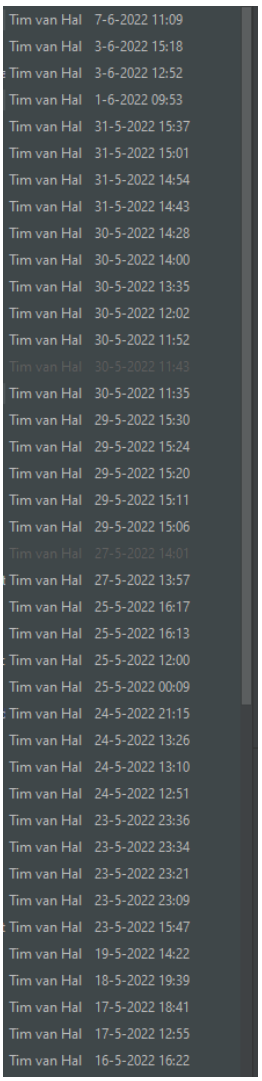
Figuur 11.5d - Het snel kunnen inwerken aldus de evaluatie van de bedrijfsbegeleider.

Er is pro-actief achter feedback aangegaan: dit blijkt wederom uit figuur 11.2.1.4a. Daarnaast is er ook feedback gegeven aan anderen in de vorm van code reviews. Een voorbeeld hiervan is weergeven in figuur 11.5e. Het aan afspraken houden blijkt wederom uit figuur 11.5b. Anderen aan afspraken houden blijkt juist uit figuur 11.5f. Het ontvangen vertrouwen om de applicatie zelfsturend uit te werken blijkt op zijn beurt uit figuur 11.5g.



Figuur 11.5e - Feedback leveren door middel van een code review.

Figuur 11.5f - Het wijzen op de wekelijkse meeting.

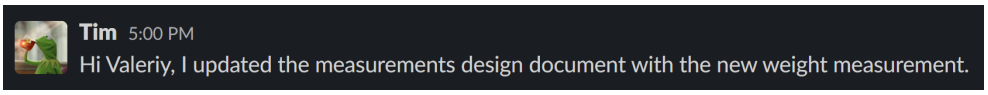


Figuur 11.5g - Code toevoegingen per auteur.

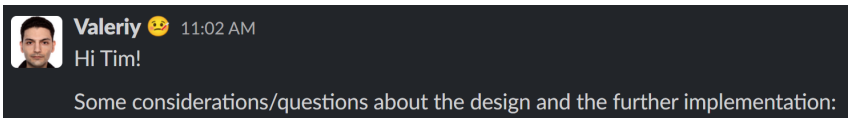
Tenslotte is het eigen handelen in samenhang met de omgeving geanalyseerd en verbeterd: met name op het gebied van communicatie. Om dit te bewijzen kan opnieuw figuur 11.5c gebruikt worden. Hier staat beschreven dat er eerst solistisch gewerkt is, maar dat de interactie en samenwerking in het team toenam na deelname aan de sprints.

11.6 - Innovatie

Innovatie is in verschillende vormen en maten aan de orde geweest gedurende het afstudeertraject. Er zijn zelf ideeën gemaakt voor measurements die gedocumenteerd zijn in een draft. Deze oplossingen zijn vervolgens gedeeld en besproken binnen de organisatie. Dit blijkt uit figuren 11.6a en 11.6b. Daarnaast is figuur 11.2.1.4d een voorbeeld van een dergelijke bespreking over voor- en nadelen met betrekking tot een enumeration.



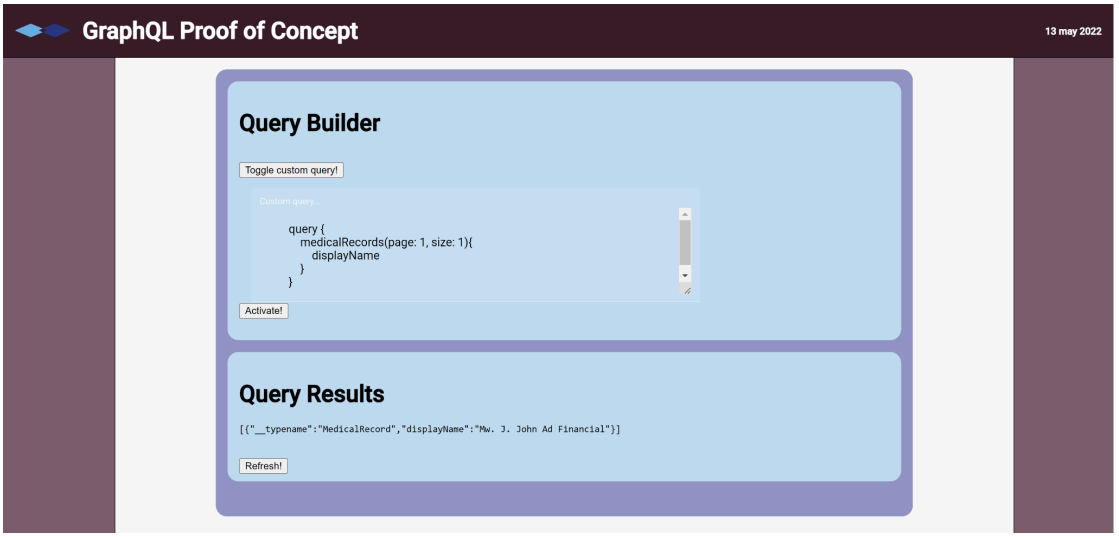
Figuur 11.6a - De toevoeging van een nieuw type meting aan de draft.



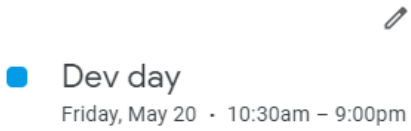
Figuur 11.6b - Een stakeholder met opmerkingen over de draft.

Integratie van perspectieven en het op de hoogte willen van actuele ontwikkelingen binnen het bedrijf komt meermaals terug. Bijvoorbeeld in de vorm van een hackathon en een developer day.
Gedurende het afstudeertraject heeft er met de teamleden van team B en team D een hackathon plaatsgevonden op 13 mei 2022. Dergelijke hackathons hebben eerder plaatsgevonden en dragen niet alleen bij aan teambuilding, maar ook aan het gebruiken van nieuwe technieken die het bedrijf nog niet inzet, maar wel wilt proberen/onderzoeken.
Er is gekozen voor GraphQL: GraphQL is een tool die ingezet wordt om enkel de benodigde velden uit een JSON-string op te vragen. Op deze manier hoeven er geen aparte DTO's geschreven te worden voor elke gewenste combinatie aan velden. Dit wordt namelijk al gauw omslachtig als het object in kwestie tientallen velden heeft. Door middel van GraphQL is het mogelijk om de niet gewenste velden achterwege te laten. Voor de hackathon is de front-end gemaakt (zie figuur 11.6c).
De hackathon resulteerde uiteindelijk dus in een applicatie die de tool GraphQL gedemonstreerd heeft. Daarnaast heeft er ook een korte presentatie plaatsgevonden waar de front- en back-end aangekaart is. Ten slotte heeft het team na de demo/presentatie besloten om deze techniek in een later stadium te onderzoeken om te kijken of het bruikbaar is voor de technology stack binnen Gerimedica.

Verder zijn actuele ontwikkelingen ook meegekregen gedurende meerdere presentaties tijdens de jaarlijkse developer day. Tijdens dev day zijn alle developers met elkaar naar kantoor gekomen om presentaties bij te wonen en technische inzichten te delen. Zo zijn bijvoorbeeld onderwerpen als Kubernetes en Cypress aan bod gekomen, maar ook is er een praktische code opdracht in groepsverband uitgevoerd.

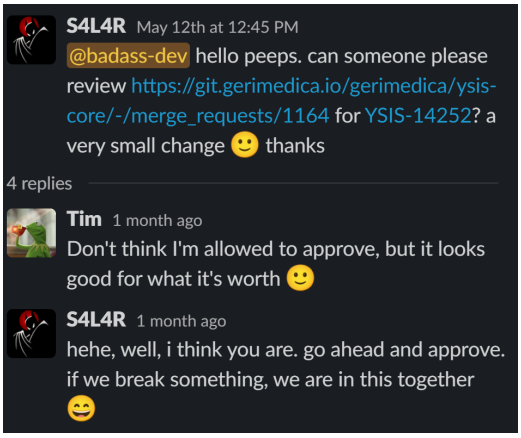


Figuur 11.6c - De front-end van het GraphQL hackathon project.

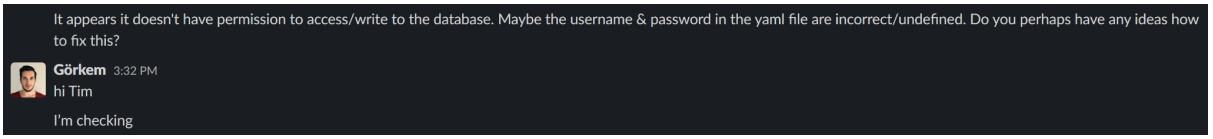


Figuur 11.6d - 20 mei 2022 was de tweede officiële dev day binnen Gerimedica.

Gedurende het afstudeertraject zijn er ook risico's genomen. Een voorbeeld van dien is het bekijken en approven van een merge request van een mede developer, zonder met de desbetreffende applicatie te hebben gewerkt. Dit is hieronder weergeven in figuur 11.6e. Verder zijn collega's ook gevraagd om met meerdere ideeën te komen, zoals figuur 11.6f laat blijken.



Figuur 11.6e - Het reviewen van andermans code binnen het bedrijf.



Figuur 11.6f - Het vragen naar mogelijke oplossingen aan een collega.

12 - Reflectie

In dit hoofdstuk wordt er gereflecteerd op het gehele afstudeertraject bij Gerimedica, waar er gedurende twintig weken tijd gewerkt is als developer. Er wordt een blik geworpen op verschillende onderdelen van de afstudeerperiode en vervolgens wordt er een mening geformuleerd.

12.1 - Aanpak, planning en uitkomsten

Persoonlijk ben ik van mening dat er op professionele wijze gewerkt is gedurende de afstudeerperiode. De aanpak van het project is op correcte wijze gevolgd: ik heb veel ervaring opgedaan omtrent de bedrijfsprocessen van Gerimedica, waar ik erg dankbaar voor ben.
De planning heeft richting het begin van het traject een kleine aanpassing gehad. Verder heb ik de planning, naar mijn idee, goed gevolgd. Er was wel sprake van enkele afwijkingen daarentegen: zo is er tweemaal onderzoek uitgevoerd na een verzoek tot empirisch onderzoek. Het tweede onderzoek en het testplan zijn enkele weken later uitgevoerd dan initieel gepland: deze documenten zijn, na overleg met stakeholder(s), op de back-burner gezet om eerst te focussen op de software architectuur en de fundering/skelet van de microservice. Uiteindelijk zijn alle beroepsproducten vooralsnog gerealiseerd.
Persoonlijk ben ik erg positief te spreken over de uitkomsten van het afstudeertraject. Dit geldt niet alleen voor de gemaakte beroepsproducten, maar ook voor het doorlopen van de bedrijfsprocessen.

12.2 - Persoonlijke leerdoelen

Gedurende mijn tijd bij Hogeschool Leiden heb ik leerpunten opgesteld om te verbeteren. Deze leerdoelen zijn plannen/organiseren en communiceren. Dit zijn voor mij persoonlijk twee leerdoelen waar ik in een eerder stadium in mijn leven vaker moeite mee heb gehad en dus heb ik er voor gekozen om hier meer de focus op te leggen. Ik kijk dan ook met trots terug op de uitwerking van deze leerdoelen tijdens mijn afstudeertraject.
Zoals eerder benoemd, ben ik van mening dat ik de planning goed heb gevolgd. Daarnaast zorgde ik er ook voor dat ik altijd op tijd was voor meetings en afspraken met collega’s. Ik heb bovendien ook zelf meetings georganiseerd. Ik heb dus weer meters gemaakt op het vlak van plannen en organiseren.
Verder heb ik ook gewerkt aan communicatie: ik had wekelijkse meetings met de bedrijfsbegeleider en dagelijks was er een stand-up met mede-developers. Bovendien waren er vaak ook nog andere meetings waar ik onderdeel van uit heb gemaakt en schroomde ik niet om de hulpvraag te stellen als ik er zelf even niet uit kwam. Al met al heb ik veel interactie gehad met mijn collega’s en is mijn communicatie in mijn ogen dus ook flink verbeterd. Dit blijkt tevens uit de evaluaties van de bedrijfsbegeleider.

12.3 - Zakelijke doelstellingen

De uitvoering van de A-competenties gedurende de afstudeeropdracht is naar mijn idee goed gelukt. Bij beide onderzoeken heb ik stilgestaan om zowel de hoofd- en deelvragen SMART op te stellen en om ervoor te zorgen dat de deelvragen een logisch antwoord geven op de hoofdvraag. Ik vind het top dat ik zoveel verschillende onderzoeksmethoden heb kunnen gebruiken bij beide onderzoeken en dat er zowel kwalitatief, als kwantitatief onderzoek heeft plaatsgevonden. Het trekken van de hoofd- en deelconclusies en het verwijzen in APA is naar mijn mening ook goed gelukt. Het enige verbeterpunt dat ik mezelf geef zit hem in het tweede onderzoek: na nader overleg is namelijk besloten het tweede onderzoek op de back-burner te zetten, waardoor het wat later afkwam. Persoonlijk vind ik dat het iets te lang op de back-burner heeft gestaan. De volgende keer zou ik het wat eerder af willen maken om het overzicht voor mijzelf te behouden.

Persoonlijk vind ik dat ik erg gemotiveerd te werk ben gegaan gedurende de afstudeerperiode: ik vond het werk altijd uitdagend, leerzaam en bovendien leuk om te doen. Ik heb van verschillende mensen gehoord dat ik mij erg goed ingezet heb (onder andere door de CTO, de bedrijfsbegeleider en leden van het team). Daarnaast vroeg ik ook om zoveel mogelijk fouten op te sporen tijdens de code reviews om de kwaliteit van mijn werk en tevens mijn zelfbeeld te versterken/verbeteren. Ten slotte ben ik ook geprezen door de bedrijfsbegeleider op het feit dat ik bijzonder weinig hulp nodig had van andere developers en dat ik de technology stack vrij snel eigen heb gemaakt. Bovendien was ik erg goed in kunnen motiveren en overtuigen. Dit was erg fijn om te horen.

Ook heb ik me bezig gehouden met professioneel werken. Het hoofdstuk kans beschrijft het probleem achter het probleem, wat tevens gebruikt is als input voor de oplossing. Bovendien is het beschreven vanuit diverse views: die van de developers en van de product owner. Daarnaast ben ik ook blij met mijn planmatig werken: het project is keurig netjes uitgevoerd met behulp van ondersteunende documentatie en met behulp van een realistische planning. Ik vond het fijn om te horen dat mijn bedrijfsbegeleider mijn planning goed vond: de ontwikkelfase was uitgespreid over een periode van twaalf weken en ingedeeld in sprints van drie weken. Bovendien is de ondersteunende documentatie op logische volgorde uitgevoerd, zoals het hoort binnen het werkproces: zo heeft het literatuuronderzoek geholpen bij het vergaren van kennis en heeft het software architectuur document geholpen bij het realiseren van de microservice. Verder ben ik erg positief over de communicatie gedurende het afstudeertraject: mijn bedrijfsbegeleider en ik hadden wekelijks een meeting omtrent progressie en hij vond het erg fijn dat ik altijd agendapunten meenam. Daarnaast merkte hij op dat ik altijd op schema liep. Dit was tof om te horen. Zoals eerder benoemd ben ik ook geprezen op mijn proactieve houding. Ik ben zelf op zoek gegaan naar feedback bij zowel mijn bedrijfsbegeleider als mijn mede-developers en ik vond het bijzonder tof dat ik ook anderen kon helpen en feedback kon geven. Ondanks dat ik in een team werkte, werkte ik geheel zelfstandig aan de microservice. Persoonlijk vond ik het fijn dat er zoveel vertrouwen was dat ik zelfsturend mijn opdracht mocht uitvoeren. Op het gebied van reflectie vind ik dat ik de samenhang met mijn omgeving geanalyseerd heb en dit vervolgens goed verbeterd heb. Dit blijkt tevens uit de evaluatie: de bedrijfsbegeleider vond dat ik eerst erg solistisch te werk ging, maar later steeds meer interactie had met het team. Dit deed mij goed om te horen.

In mijn optiek ben ik goed bezig geweest met innovatie. Ik heb zelf ideeën en oplossingen toegepast, zoals bijvoorbeeld het state pattern of de validatie. Daarnaast heeft er ook een hackathon plaatsgevonden, die ik als leuk en creatief heb ondervonden. Hier heb ik zelfstandig de front-end voor gerealiseerd en eigenhandig de GraphQL tool geïntegreerd. Hier ben ik erg trots op, aangezien ik nog nooit eerder met de tool gewerkt heb en het in korte tijd werkend heb kunnen krijgen. Bovendien gaat GraphQL nu verder onderzocht worden door het bedrijf na de demo. Daarnaast vind ik dat ik goed interesse heb getoont in de actuele ontwikkelingen binnen het bedrijf. Zo ben ik meerdere meetings bijgewoond die niet per direct relevant waren voor mijn opdracht, maar ik vond het gaaf om te zien waar de rest van mijn team aan werkte. Daarnaast vind ik dat ik gecalculeerd risico's genomen heb. Zo heb ik bijvoorbeeld ook de code gereviewed van mijn teamleden. Dit was niet direct onderdeel van mijn opdracht, maar dit vond ik wel leerzaam en leuk om te doen. Mocht er iets misgaan in de pipeline of in de environment, dan zou dit gezamenlijk opgelost worden. Ik vond het bovendien ook erg fijn dat niet alleen mijn teamleden mij af en toe hielpen en met nieuwe ideeën kwamen, maar dat ik hen ook heb kunnen helpen als ze vastliepen.

12.4 - Risico's

Eerder is er gesproken over risico's die de afstudeeropdracht met zich meebrachten (zie hoofdstuk 9). Over het algemeen zijn deze weinig tot niet teruggekomen tijdens de afstudeerperiode.
Wel is de VPN aan het begin van het traject kort weggevallen door het verlopen van de geldigheid van het VPN-certificaat. Ik vind het goed van mezelf dat ik als reactie hierop met de support gesproken om dit probleem op te lossen. Een enkele verbeterpunt dat ik aan mezelf geef is dat ik de volgende keer wellicht wat sneller aan de bel moet trekken: ik heb niet direct contact opgenomen met de support, aangezien ik op dat moment nog niet afhankelijk was van de VPN-verbinding. Dit kan ik in het vervolg beter wel doen, mocht ik spontaan toch wel de VPN nodig hebben.

12.5 - Opgedane ervaring(en) in de beroepsorganisatie

Het afstudeertraject bij Gerimedica beschouw ik als zeer prettig en bovendien educatief. Ik ben met veel bestaande kennis begonnen aan mijn afstudeeropdracht, maar ik heb daarnaast ook een aantal nieuwe dingen geleerd. Ik heb meer ervaring opgedaan met Spring Boot, ik heb mijn kennis omtrent Gitlab-CI en Docker opnieuw aangescherpt sinds de afronding van het hoofdfase project. Daarnaast heb ik ditmaal ook meer meegekregen van de bedrijfsprocessen binnen Gerimedica, waar ik veel van geleerd heb.
Daarnaast heb ik ook kennis opgedaan van nieuwe technologieën, zoals REST-assured en Flyway. Deze onderwerpen waren voorheen onbekend bij mij. Ik vind het erg top dat ik, naast het kunnen toepassen van veel bestaande kennis, alsnog veel nieuwe dingen geleerd heb.

12.6 - Uitgevoerde beroepstaken

Gedurende de afstudeerperiode is de focus gelegd op het bewijzen van drie verschillende B-competenties: *software analyseren, software ontwerpen* en *software realiseren*. Ik ben zeer tevreden met het resultaat van dien: ik heb het gevoel dat mijn documentatie nog een stuk professioneler is geworden en dat ik overzichtelijker te werk ben gegaan. Wel vind ik dat ik iets te veel focus heb gelegd op documentatie. In de toekomst wil ik dit ook iets beter afstemmen met stakeholders, zodat ik niet eindig met bijvoorbeeld twee onderzoeken. Ondanks dit verbeterpunt ben ik alsnog erg blij met het resultaat.

Het uitvoeren van software analyseren is naar mijn idee goed gelukt: er is onderzoek uitgevoerd voor meerdere aspecten (ontwikkelproces, frameworks en design patterns). Persoonlijk vond ik de onderzoeken leerzaam en betekenisvol. Ik vind dat het vaststellen van de methoden en technieken met de stakeholders ook goed ging. Idem dito voor de requirements met prioritering. De acceptatiecriteria zijn zowel functioneel als technisch en zijn op professionele wijze geclassificeerd in een acceptatieplan. Ik ben van mening dat ik dit professioneel aangepakt heb. Verder vind ik dat integratie met het huidige systeem prima gelukt is. Ik vind het jammer dat migratie wegviel wegens de omvang van dien, maar het onderzoek naar migratie was alsnog zeer leerzaam.

Het werken aan software ontwerpen ging vrijwel vlekkeloos: het ontwerp is onder andere opgebouwd aan de hand van meerdere meetings met stakeholders. Ook is er een aantal opmerkingen en feedback verstrekt vanuit het bedrijf over het ontwerp: dit blijkt onder andere uit de versiebeheer. Op basis hiervan heb ik aanpassingen gemaakt, dan wel toelichting gegeven. Persoonlijk vind ik dat ik dit goed aangepakt heb. Ook ben ik van mening dat het ontwerp goed vertaald is naar code. Ik ben tevreden met de opgestelde teststrategie, al was het testplan iets later gerealiseerd dan voorheen gepland. Verder ben ik blij met de proof of concept, aangezien het een simpel, maar goed overzicht gaf van hoe het ontwerp uitpakte.

Software realiseren is als laatste uitgevoerd en is naar mijn mening geslaagd: de realisatie is gebouwd zoals deze beschreven staat in de ontwerpen, maar er is hier en daar een afwijking. Dit heeft te maken met het feit dat er oplossingen aanwezig waren die moeilijk te ontwerpen zijn (denk bijvoorbeeld aan de toepassing van het visitor pattern). Deze zijn alsnog gedocumenteerd in het verslag. Er zijn minimaal drie ISO-25010 standaarden besproken in het hoofdstuk kwaliteitskenmerken: *functional suitability, compatibility* en *security*. De applicatie is opgebouwd in een lagenstructuur en volgens het software architectuur document. Daarnaast is er getest met zowel JUnit 5 (unit tests) als REST-assured (integration tests). Bovendien zijn deze tests naast lokaal, ook meermaals, uitgevoerd in de automatische teststraat. De pipelines hebben naast automatisch testen, ook de applicatie automatisch gebouwd en gedeployed naar meerdere omgevingen. Persoonlijk vond ik het werk echt uitdagend, aangezien de technology stack vrij groot was, maar al met al vond ik het ook erg leuk en educatief.

12.7 - Begeleiding door afstudeerorganisatie

De begeleiding door de afstudeerorganisatie was in mijn optiek zeer goed. Er is sprake van een goede bedrijfscultuur (zie hoofdstuk 3.5). Zowel de bedrijfsbegeleider als mede-developers helpen elkaar ook altijd en vonden het ook geen probleem om mij hulp te bieden, zowel op locatie als op Slack. Bovendien vond ik de dagelijkse contactmomenten ook erg tof. Al met al ben ik zeer positief over de begeleiding die geleverd is vanuit het afstudeerbedrijf.

12.8 - Begeleiding door opleidingsinstituut

De begeleiding vanuit de opleiding was in mijn ogen ook prima. De afstudeerdocent was over het algemeen wat terughoudender, wegens het aantonen van mijn zelfredzaamheid. Persoonlijk vind ik van mijzelf dat ik hier goed op gehandeld heb door bijvoorbeeld het sturen van reminder mailtjes.
Een verbeterpunt die ik mezelf wel wil meegeven is om de volgende keer te proberen vaker op andere wijzen contact op te nemen: zoals met Microsoft Teams of via de telefoon. Op deze manier vergroot ik de kans op een reactie.
Verder waren de handige tips, uitleg of feedback van Ron altijd duidelijk: ik wist direct wat er van me verwacht werd, aangezien de opmerkingen altijd rechtdoorzee waren. Dank hiervoor.

12.9 - Algemeen

Als puntje bij paaltje komt heb ik mijn afstudeertraject als zeer prettig en leerzaam ondervonden. Ik kijk met trots terug op de werkzaamheden die ik heb uitgevoerd en de producten die ik heb opgeleverd. Daarnaast ben ik ook blij met de opgedane ervaring binnen Gerimedica en alle nieuwe onderwerpen waar ik kennis over vergaard heb. Ook wil ik mijn waardering tonen voor de mogelijkheid tot deze mooie afstudeeropdracht. Mijn dank gaat uit naar alle betrokkenen.

13 - Nawoord

De opdrachtgever was erg content met de kwaliteit van de beroepsproducten. De microservice draait in meerdere omgevingen en de opgebouwde documenten zijn tevens overgedragen.

De microservice bevat onder andere een README waar beschreven staat wat geregeld moet zijn om de applicatie lokaal te draaien. De microservice zelf is daarnaast in meerdere omgevingen gedeployed door middel van een GitLab-pipeline. Hier wordt automatisch gebouwd en getest (zowel unit als integration tests).

De opdrachtgever is ook tevreden over het proces: dit blijkt uit beide evaluaties. Daarnaast waren de leden van het team ook erg positief te spreken over de inzet en commitment gedurende de afstudeerperiode.

Begrippenlijst

Ysis - Het elektronisch patiënten dossier van Gerimedica, wat gebruikt wordt door onder andere doctoren.

ECD - Een elektronisch cliënten dossier, wat voornamelijk ingezet wordt door verpleegsters.

REST-assured - Een test framework wat binnen Gerimedica gebruikt wordt voor integratie testen.

Flyway - Een tool die automatisch database scripts uitvoert.

Graylog - Een tool wat gebruikt wordt om alle handelingen in een systeem te loggen.

Prometheus - Een monitoring tool wat gebruikt wordt binnen Gerimedica in combinatie met Grafana.

Grafana - Een tool die overzichtelijke dashboards biedt aan haar gebruikers.

Axon - Een framework wat gebruik maakt van commands en events om handelingen uit te voeren.

HAPI - Een complete implementatie van de HL7-FHIR standaard in Java.

HL7-FHIR - Een internationale standaard voor gegevensoverdracht in de zorg.

Resource Exhaustion - Het overweldigen van een component door bijvoorbeeld heel veel data in één keer op te vragen.

Soft limit - Een limiet die aanwezig is op de huidige actie binnen de applicatie.

Literatuurlijst

Bruins, M. (2021, August 11). *Interoperabiliteit in de zorg, wat is dat?* Gerimedica. Retrieved March 30, 2022, from https://www.gerimedica.nl/interoperabiliteit-in-de-zorg-wat-is-dat/

Altvater, A. (2019, September 13). *What are Microservices? Code Examples, Best Practices, Tutorials and More*. Retrieved from Stackify: https://www.stackify.com/what-are-microservices/

Amazon Web Services. (2022). *Microservices*. Retrieved from AWS: https://www.ams.amazon.com/microservices/

Baeldung. (2021, December 29). *Entity To DTO Conversion for a Spring REST API*. Retrieved from Baeldung: https://www.baeldung.com/entity-to-and-from-dto-for-a-java-spring-application

Bealdung. (2020, May 27). *A Guide to REST-assured*. Retrieved from Baeldung: https://www.baeldung.com/rest-assured-tutorial

Davis, K. (2018, August 18). *What Is HAPI FHIR Server? How Do We Deploy It?* Retrieved from DZone: https://dzone.com/articles/what-is-hapi-fhir-server-and-how-to-deploy-it-1

Haleby, J. (2022, February 10). *GettingStarted*. Retrieved from GitHub: https://www.github.com/rest-assured/rest-assured/wiki/GettingStarted

Hamcrest. (2019). *Hamcrest Tutorial*. Retrieved from Hamcrest: http://hamcrest.org/JavaHamcrest/tutorial

Hamilton, T. (2021, December 25). *REST Assured Tutorial: How to test API with Example*. Retrieved from Guru99: https://www.guru99.com/rest-assured.html

HAPI FHIR. (2022, February 28). *HAPI FHIR*. Retrieved from HAPI FHIR: hapifhir.io

HAPI FHIR. (2022, February 28). *Introduction to HAPI FHIR*. Retrieved from HAPI FHIR: https://hapifhir.io/hapi-fhir/docs/getting_started/introduction.html

Refactoring Guru. (2022). *Adapter*. Retrieved from Refactoring Guru: https://www.refactoring.guru/design-patterns/adapter

Refactoring Guru. (2022). *Builder*. Retrieved from Refactoring Guru: https://www.refactoring.guru/design-patterns/builder

Refactoring Guru. (2022). *Code Smells*. Retrieved from Refactoring Guru: https://www.refactoring.guru/refactoring/smells

Refactoring Guru. (2022). *Command*. Retrieved from Refactoring Guru: https://www.refactoring.guru/design-patterns/command

Refactoring Guru. (2022). *Composite*. Retrieved from Refactoring Guru: https://www.refactoring.guru/design-patterns/composite

Refactoring Guru. (2022). *Design Patterns*. Retrieved from Refactoring Guru: https://www.refactoring.guru/design-patterns

Refactoring Guru. (2022). *Singleton*. Retrieved from Refactoring Guru: https://www.refactoring.guru/design-patterns/singleton

SentinelOne. (2018, September 25). *API vs. Microservices: A Microservice Is More Than Just an API*. Retrieved from SentinelOne: https://www.sentinelone.com/blog/api-vs-microservices-2/

Spring. (2022). *Spring Quickstart Guide*. Retrieved from Spring: https://spring.io/quickstart

Spring. (2022). *Why Spring?* Retrieved from Spring: https://www.spring.io/why-spring

Stichting Nuts. (2021). *Over de stichting.* Retrieved from Nuts: https://nuts.nl/stichting/

Waterman, S. (2020, January 03). *Rethinking the Java DTO*. Retrieved from Scott Logic: https://blog.scottlogic.com/2020/01/03/rethinking-the-java-dto.html

AxonIQ. (2022). About Axon Framework. Opgehaald van AxonIQ Dev Portal: https://developer.axoniq.io/axon-framework/overview

AxonIQ. (2022). AxonIQ Products. Opgehaald van AxonIQ: https://www.axoniq.io/axoniq-products

AxoniQ. (2022, February). Migration. Opgehaald van Axon Reference Guide: https://docs.axoniq.io/reference-guide/axon-server/migration

baeldung. (2022, March 11). Introduction to Pointcut. Opgehaald van Baeldung: https://www.baeldung.com/spring-aop-pointcut-tutorial

baeldung. (2022, April 12). Introduction to Spring AOP. Opgehaald van Baeldung: https://www.baeldung.com/spring-aop

Fowler, M. (2005, december 12). Event Sourcing. Opgehaald van martinFowler: https://martinfowler.com/eaaDev/EventSourcing.html