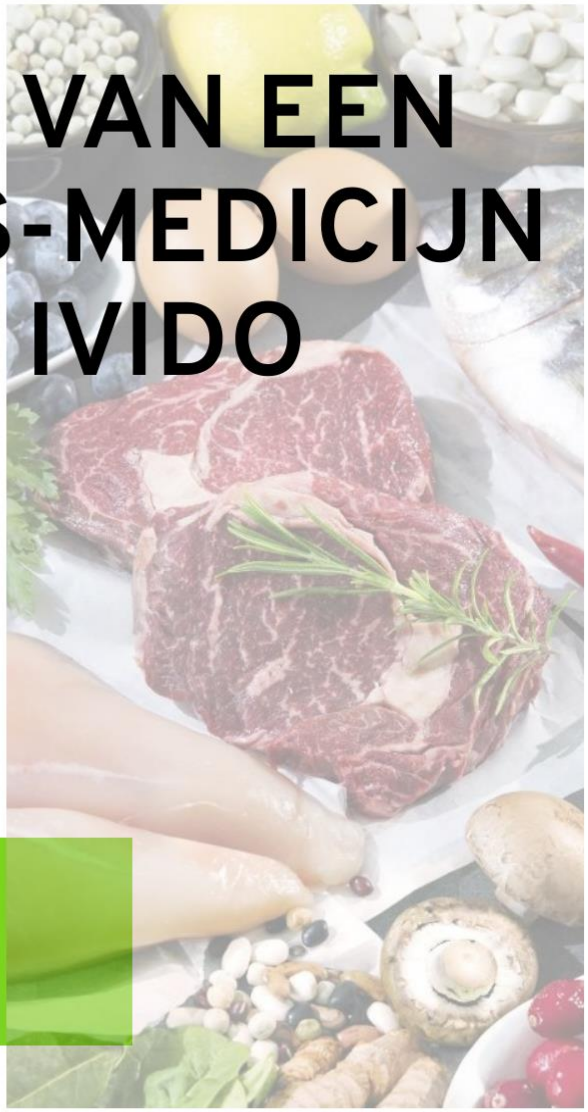
AFSTUDEERVERSLAG HBO-ICT:SE

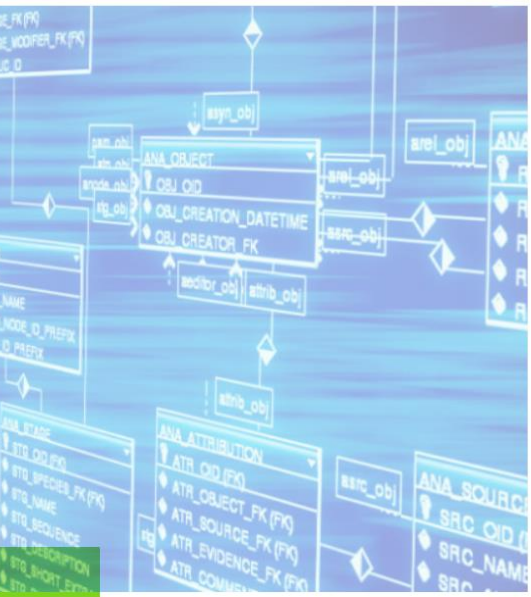
ONTWIKKELEN VAN EEN LEEFSTIJL-ALS-MEDICIJN MODULE VOOR IVIDO

VOEDING OP MAAT



Naam: Michiel Maas
Studentnummer: 16136640

Datum: 08-01-2021



DE HAAGSE HOGESCHOOL

IVIDO

Voorwoord

Dit verslag beschrijft het proces en de keuzes gemaakt bij het ontwerpen en ontwikkelen van een module om voedselconsumptie van een gebruiker bij te houden in de Persoonlijke Gezondheidsomgeving van IVIDO. Dit project vond plaats als afstudeer project van de opleiding HBO-ICT: Software Engineering aan de Haagse Hogeschool.

Bij het maken van de feature en het verslag heb ik veel help gekregen van verschillende mensen. Deze wil ik graag bedanken voor hun medewerking, feedback en tijd.

Als eerst wil ik IVIDO bedanken voor het opzetten van de opdracht en het aanbieden van hun resources om de opdracht uit te voeren. In het bijzonder wil ik graag Tristan Garssen bedanken, voor zijn rol als opdrachtgever. Zijn focus op resultaat en hulp bij verbinden van de stakeholders, hebben veel geholpen om de opdracht tot een goed einde te brengen. Ook wil ik graag Gerda In ’t Veld bedanken voor haar rol als begeleidend examinator. Haar constructieve feedback heeft mij veel geholpen bij het maken van de verslaglegging van het afstudeer traject.

Verder wil ik graag Mariska Maas bedanken voor het proeflezen van dit verslag en hoe zij mij altijd helpt het beste uit mijzelf te krijgen. Ook wil ik Stephen Fry bedanken voor zijn werk aan de show QI waarmee hij mij door de laatste paar weken van het afstuderen heeft geholpen.

Michiel Maas
Delft, januari 2021

Referaat
Een feature in een Persoonlijke Gezondheidsomgeving die patiënten, en hun zorg- professionals, inzicht geeft in hun dieet.

Kernwoorden:

Afstuderen	HHS	MERN	noSQL	Database Migratie
Requirements	Scrum	Agile	SE	Architecture Design
Microservice				

Inhoudsopgave

Bijlage

1 Inleiding

Als afsluiting van de opleiding HBO-ICT heb afgelopen maanden heb ik bij IVIDO mijn afstudeeropdracht vervuld. IVIDO is een startup die een Persoonlijke Gezondheidsomgeving ontwikkelt die patiënten en hun zorgverleners beter laten samenwerken. Zij hebben mij gevraagd om een Leefstijl-als-Medicijn module te ontwerpen en te ontwikkelen. Met deze module willen ze patiënten inzicht geven in hun leefstijl en helpen deze te verbeteren.

In dit verslag beschrijf ik hoe ik mijn beroepstaken heb vervuld, en de afstudeeropdracht heb uitgevoerd. Dit verslag dient gebruikt te worden door de examinatoren en gecommitteerde om te bepalen of ik het afstudeerproces goed heb uitgevoerd, en ik de titel van Bachelor of Science waardig ben.

Eerst heb ik de requirements opgesteld door onder andere interviews af te nemen met stakeholders. Met deze requirements heb ik een architectuur ontwerp opgesteld van hoe de module eruit zou moeten zien. Toen bleek dat ik een database migratie nodig was heb ik een nieuwe database ontworpen en de data daarin over gezet. Door een aantal onverwachte tegenvallers bleef kon de module niet als verwacht opgeleverd worden. Met de tijd die over was ik het MVP ontwikkeld die een deel van de requirements vervuld, maar nog niet volledig geïntegreerd is in het programma van IVIDO.

Aan het eind van het verslag is een woordenlijst toegevoegd die veelgebruikte termen in het verslag zal toelichten. Ook zijn er meerdere bijlagen die dieper ingaan op verschillende onderdelen de afstudeeropdracht.

2 Bedrijf & Probleemdomein

IVIDO (Eye-vi-do) probeert het gat tussen patiënt en zorgverlener te verkleinen. Dit gebeurt aan de hand van een webapp die zowel op desktop als op mobiel te gebruiken is. Deze webapp is een zogenaamde 'Persoonlijke Gezondheidsomgeving' (hierna afgekort tot PGO). Patiënten kunnen via een PGO direct in contact komen met zorgprofessionals. De PGO verzamelt, onder andere, informatie over de medische geschiedenis van de patiënt, gebruikte medicijnen, medische metingen en medische vragenlijsten. Op deze manier probeert IVIDO betere en meer persoonlijke zorg te bieden.
De PGO van IVIDO wordt in samenwerking met HINQ[1], de digitale zorgnetwerk coöperatie, ontwikkeld. HINQ maakt het mogelijk voor "zorginstellingen/zorgprofessionals om schotloos (sic) samen te werken in zorgnetwerken", en helpt met de certificaten die IVIDO nodig heeft om de medische gegevens te verwerken. IVIDO zelf is een jong bedrijf (gestart in 2015), heeft haar kantoor in de buurt van Station Laan van Noord Oost Indië en heeft nu ongeveer 20 werknemers. De bedrijfscultuur wordt omschreven als "Innovatief, jong, flexibel en ambitieus".
Voor iedere gebruiker die is aangesloten op het systeem krijgt IVIDO een vergoeding van het Ministerie van Volksgezondheid. Overige inkomsten komen van het ontwikkelen van modules en content voor zorgleveranciers en het voorzien van licenties aan zorgprofessionals.

Tabel 1 beschrijft de medewerkers binnen IVIDO die betrokken zijn geweest bij de afstudeeropdracht.

Tabel 1: Betrokken personen bij afstudeeropdracht

Persoon	**Rol**
Tristan Garssen	Opdrachtgever / COO
Jari Maijenburg	Bedrijfsmentor / Lead Developer
Edwin de Wit	Product Owner

[1] https://hinq.nl/

3 Opdrachtdefinitie

3.1 Aanleiding

“In de afgelopen 8 jaar is het aantal mensen met één of meer chronische ziekten met 17% gestegen. In 2011 hadden in totaal 5,3 miljoen Nederlanders een chronische ziekte. Vooral onder ouderen komen chronische ziekten vaak voor. De stijging van de laatste jaren is echter zichtbaar binnen alle leeftijdsgroepen.” Zo stelt het RIVM.[2]
Zorgprofessionals moedigen burgers aan om een betere levensstijl aan te nemen om deze trend tegen te gaan. Denk dan aan gezond eten, niet roken, genoeg slapen, sporten en verminderen van stress. Zo kreeg IVIDO ook de vraag of zij, via de PGO, patiënten konden aanmoedigen om hun levensstijl te verbeteren en daarmee een ‘levensstijl-als-medicijn’ toe te dienen.

3.2 Probleemstelling

Op dit moment heeft IVIDO nog geen module die dit verwezenlijkt. Er is eerder een poging gedaan om een module voor de PGO te ontwikkelen die dat dit probleem aanpakt, maar op dat moment was het systeem nog niet genoeg ontwikkeld om het te ondersteunen. Ondertussen is de PGO veel verder ontwikkeld en is er nu functionaliteit die het mogelijk maakt deze module goed te ondersteunen. Maar omdat IVIDO snel aan het groeien is, heeft IVIDO geen tijd om de dit te ontwikkelen. Daarom zijn ze opzoek gegaan naar een afstudeerder die het kon oppakken.

3.3 Doelstelling

IVIDO beschikt over een extra module in de bestaande PGO die het aannemen en onderhouden van een gezonde levensstijl makkelijker maakt.
De module ondersteunt het monitoren van voeding. De patiënt kan invullen wat hij of zij eet op welk moment, en toont een overzicht van de voedingswaarden, mineralen en vitamine. De zorgprofessional kan dit inzicht gebruiken om een advies te geven aan de patiënt over hoe hij of zij hun leefstijl kan verbeteren. De module kan uitgebreid worden met andere vormen van monitoring en andere overzichten.
De module ondersteund als eerst het monitoring van voeding omdat dit de grootste oorzaak is van chronische aandoeningen in Nederland. Door deze eerst aan te pakken is de module het meest effectief. Ook is het monitoring van voeding complex. Door een abstracte basis te leggen die het complexe voeding monitoring ondersteund, wordt het later makkelijker om andere, meer eenvoudige, activiteiten te monitoren zoals roken, sporten of slapen.
Aan het eind van het afstuderen is de module geïmplementeerd in de PGO. Het bied functionaliteiten waarmee voeding kan worden gemonitord. Tijdens het project worden er verschillende versies uitgebracht om feedback van de gebruikers te verzamelen.

Uiteindelijk liepen de plannen heel anders. De module is niet geïmplementeerd in de PGO, tijdens het proces heb ik de module niet kunnen testen door gebruiken en toont het alleen een overzicht van de gegeten producten.

[2] https://www.rivm.nl/nieuws/aantal-chronisch-zieken-neemt-toe

3.4 Eindresultaat

Tijdens het afstuderen worden er een aantal documenten opgesteld en producten ontwikkeld. Deze tussenproducten lichten de gemaakte keuzes toe, en dienen als bewijs van de beheersing van beroepstaken. Deze tussenproducten zijn als volgt:

- Requirements Rapport
- Architecture Design
- Test Plan
- Code Project
- *Migratie Verslag*[3]

Requirements Rapport
Dit rapport beschrijft het proces van het opstellen van de requirements. Het gaat in op de gebruikte methodes, stakeholder management en prioritering van de requirements. Het document is geschreven voor de betreffende stakeholders.
Het rapport vervult de Beroepstaak A3 (Vergaren en analyseren van requirements), en is te vinden in bijlage A.

Architecture Design
Het Architecture Design document beschrijft het proces van het ontwerpen van de software architectuur van de module. Het document beschrijft welke methodes zijn gebruikt om het systeem te ontwerpen, hoe mogelijke implementaties aan elkaar zijn afgewogen en hoe de uiteindelijke module ontworpen zou moeten worden.
Het document vervult de Beroepstaak C6 (Ontwerpen software), en is te vinden in bijlage B.

Test Plan
De tools en methode van de test suite voor de module zijn beschreven in het Test Plan. Het ligt toe waarom de tools zijn gekozen, hoe ze zijn toegepast en met welke overwegingen. Ook heeft het een test rapport die de code coverage van het project aan toont. Dit is te vinden in bijlage C.

Code Project
Het laatste geplande beroepsproduct is de Code van het Project. Omdat ik een geheimhoudingsverklaring heb getekend voordat ik bij IVIDO ben begonnen, kan ik de code zelf niet opleveren. Om nog steeds een bewijs aan te leveren dat de code van kwaliteit is, zal er een verklaring worden aangeleverd. Deze verklaring zal komen vanuit een medewerker van IVIDO die kwaliteit van de code bevestig en aangeeft of de module naar wens is. Een overzicht van de module is te vinden in het Module Overzicht in bijlage D.

Migratie Verslag
Tijdens het traject bleek dat een Database Migratie nodig was. Deze migratie koste meer tijd van verwacht. Om deze tijdsinvestering te verantwoorden is er een Migratie Verslag opgesteld. Dit verslag beschrijft de stappen die zijn genomen in het ontwerpen van de nieuwe database, opschonen van de bestaande database en het migreren naar de nieuwe database. Het verslag is te vinden in bijlage E.

[3] Het Migratie Verslag was niet deel van de originele planning

4 Aanpak

Het afstudeer traject bestaat uit 17 werkweken. In deze tijd moet de afstudeeropdracht worden vervult, de beroepstaken op niveau worden uitgevoerd en het afstudeerverslag worden geschreven.

Het plan is om de afstudeeropdracht met een Agile werkmethode uit te voeren. Met deze methode kan het ontwikkelproces gemakkelijk van richting veranderen naar vraag van de stakeholders. Er is voor Agile gekozen omdat het zeer goed inspeelt op de vraag van de stakeholders. Persoonlijk vind ik de methode ook fijn om mee te werken omdat ik merk dat plannen vaak veranderen.

Voordat ik begin met het ontwikkelen van het product, ben ik van plan eerst het requirements te eliciteren en de software architectuur te ontwerpen. De requirements worden gedocumenteerd in een Requirements Rapport. Aan de hand van de requirements wordt een Architecture Design gemaakt. Dit design wordt getest met een Proof of Concept.

4.1 Globale Planning

De eerste week van het afstuderen is uitgetrokken om het Plan van Aanpak op te stellen. Dit plan is te vinden als bijlage F. Door gesprekken te voeren met de belangrijkste stakeholders wil ik de scope van de opdracht te bepalen. Met deze informatie heb ik de eerste globale planning gemaakt. Hierin zijn de oorspronkelijke gekozen beroepstaken opgenomen. De tijdsduur is geschat op basis van ervaring met het maken van soortgelijke documenten en in overleg met de opdrachtgever. Zie Tabel 2 en Tabel 3 voor een overzicht van de planning.

Tabel 2: Fases in Afstudeertraject

Fase	Resultaat	Duur
Aanpak en Onderzoek	Plan van Aanpak	1 Week
Requirements	Requirements Rapport	2 Weken
Architecture	Architecture Design en Proof of Concept	2 Weken
Ontwikkelen & Testen	Module en Test Plan	10 Weken
Oplevering	Afstudeerverslag	2 Weken

Tabel 3: Eerste Globale Planning

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

Voor opstellen van het Requirements Rapport zijn twee weken uitgetrokken. Dit is op basis van de het aangeleverd onboarding document, en de kennis die is verkregen bij het opstellen van het afstudeerplan. Ik verwacht dat een goed idee had van wat de opdracht zal moeten worden, en dat ik met enkele gesprekken en interviews de definitieve requirements goed op papier kan zetten.

Voor de Architectuur beroepstaak zijn twee weken uitgetrokken. De eerste week zal worden gebruikt voor het bepalen van de patterns en de tools, en het uitschrijven in het Architecture Design Document. De andere week zal gebruikt worden om een Proof of Concept te maken. Dit Proof of Concept zal testen of de tools goed samen zouden werken, en een basis zijn om te laten verifiëren door belangrijke stakeholders.

Veruit de grootste tijd voor het project is ingepland voor het ontwikkelen en opstellen en uitvoeren van het testplan. Dit zal pas plaats vinden nadat de twee vorige taken zijn afgerond. Op deze manier kan ik gelijk beginnen met de goede tools en duidelijke requirements. Er is zo veel tijd uitgetrokken omdat dit het meeste werk lijkt te zijn. Op de achtergrond verwachtte ik ook nog de requirements wat bij te schaven en de het architectuur design up te daten wanner ik nieuwe ontdekkingen maak.

In deze periode van tien weken zal ik in sprints van twee weken de module ontwikkelen. Ik zal regelmatig afspraken met stakeholders om feedback te krijgen op de voortgang en richting van het project. Tijdens het ontwikkelen ben ik van plan de testen te schrijven zodat die up-to-date blijven met de code.

De laatste twee weken van het afstuderen zijn behouden als buffer. Deze periode kan gebruikt worden om mogelijke vertraging op te vangen.

Risico's
De kans is aanwezig dat tijdens het project die prioriteiten veranderen en dat de planning aangepast moet worden. Om het proces hiervoor te beschermen zijn die twee bufferweken aan het eind van het traject ingepland. De Agile werkmethode helpt ook bij het bijsturen van het project. De regelmatige gesprekken met stakeholders helpen het project op de goede koers te houden.

Tijdens het project is de planning vaak veranderd. Het opstellen van de requirements duurde langer dan verwacht en ik heb uiteindelijk een database migratie moeten uitvoeren die niet gepland was. De Agile methode heeft me goed geholpen om tijdig te kunnen bijsturen.

4.2 Eliciteren van Requirements

Bij het project zijn vele verschillende stakeholders betrokken. Dit zijn medewerkers en partners van IVIDO. Deze stakeholders hebben verschillende expertises, en verschillende verwachtingen van de module. Om dit goed aan te pakken is het volgende plan opgesteld.

Met alle stakeholders wordt een initieel interview afgenomen. Hierin wordt bepaald hoe zijn of haar expertise gebruikt kan worden bij het ontwikkelen van de module en wat hij of zij graag willen zien in de module.

Met deze initiële informatie wordt een functionele schets gemaakt van de module. Door verduidelijkende interviews wordt er geïtereerd op deze functionele schets, totdat alle aspecten goed in beeld zijn. Met dit ontwerp wordt vervolgens eerste lijst van requirements opgesteld. In samenwerking met de stakeholders worden de requirements aangescherpt en de prioritering bepaald. Dit gebeurt totdat alle stakeholders tevreden zijn.

In de interviews worden de stakeholders veel vragen gesteld. Door vragen te herhalen met een andere formulering wil ik mogelijke uitzonderingen op regels achterhalen. Ook zal gevraagd worden om diagrammen te schetsen die beschrijven hoe functionaliteiten elkaar volgen. Verder zal er ook gebruik gemaakt worden van casussen, marktonderzoek en zal de documentatie van een eerder geprobeerde implementatie bestudeerd worden.

Risico's
Fouten bij het opstellen van de requirements kunnen leiden tot het maken van een verkeerd product. Daarom is het belangrijk om dit zorgvuldig te doen. Door de requirements samen met de correcte stakeholders op te stellen, wordt hun wens het best vertegenwoordigd. Maar er moet ook opgelet worden dat het niet te veel tijd gaat kosten. Om de Scope van het project te bewaren is het belangrijk dat ik snel de MVP van het project te bepalen, en 'Nee' kan zeggen tegen wensen die buiten de Scope vallen.

4.3 Architecture Design

Nadat de requirements zijn vast gesteld kan er begonnen worden aan het Architecture Design. Hier wordt bepaald welke tools en implementaties het beste passen bij de module. Deze keuzes worden gemaakt met de Attribute Driven Design (ADD) methode. Deze methode maakt een iteratief ontwerp en gebruikt de Niet Functionele Requirements op de beslissingen te maken. Door de vele stakeholders verwacht ik veel Niet Functionele requirements, en de iteratieve methode zal helpen omdat schaal van het project groot is.

Het architectuur design zal worden afgestemd met de Lead Developer. Zijn kennis van de code base en de tools die IVIDO gebruikt is belangrijk om mee te nemen in de overwegingen. Ik heeft hij het beste zicht op hoe de module aangesloten zou kunnen binnen het cluster van IVIDO.

Alle keuzes worden beschreven in een Architecture Design document. Dit dient als een verklaring van de keuzes. Met de gemaakte keuzes ben ik van plan een Proof of Concept te maken. Hier mee wil ik testen of het ontwerp klopt. Daarnaast wil ik het ook gebruiken om verifiëren of mijn idee van de module klopt, door het voor te leggen aan stakeholders.

Risico's
Een fout in het architectuurontwerp kan veel tijd kosten. Er moet een betere oplossing komen voor het probleem, en de fout moet hersteld worden. Door zorgvuldig onderzoek te doen naar de opties, wil ik de beste keuzes te maken. Het Proof of Concept dient ook als een 'smoke test' om foute keuzes vroeg te herkennen.
Het kan zijn dat er tijdens het ontwerpen blijkt dat de Scope drastisch aangepast moet worden. Wanneer dit gebeurd zal ik het roer om te gooien. Het doel van de opdracht is om een werkend product op te leveren aan IVIDO. Als deze aanpassingen een beter product oplevert, dan moet het geïmplementeerd worden.

4.4 Ontwikkeling van Module

Ik zal met Scrum op een Agile manier de module ontwikkelen. Door agile te werken kan ik de vraag van de stakeholders goed vervullen. Het is tevens ook de methode die IVIDO gebruikt. Door aan te sluiten bij hun sprint team, komt er een ritme in het project.
In het sprint team zal ik mee doen aan de daily standups, sprint meetings en sprint review. Omdat mijn opdracht los staat zal ik meer verantwoordelijk zijn om mijn zelf te reviewen en de volgende sprint te plannen. Dit zal gebeuren in Jira, de issuetracker die IVIDO gebruikt. Er wordt een doel gekozen voor de volgende sprint en de tasks met de hoogste prioriteit die aansluiten op dat doel worden opgenomen.
Door regelmatig af te spreken met stakeholders kan er feedback gegeven worden op het product. Door een korte feedback loop op te zetten kan ik het snelste leren van mijn fouten en de stakeholders het beste tegemoet komen.

Risico's
Het kan goed zijn dat de Scope te groot is voor de beschikbare tijd. Dit zal er voor zorgen dat niet aan alle eisen van de stakeholders kan worden voldaan. Wanneer dit opkomt is het belangrijk om de focussen op het Minimum Viable Product (MVP). Door de tasks goed te prioriteren wil het snelst werken naar een goed eind product.

4.5 Testing

Tijdens het ontwikkelen van de module zullen ook testen worden geschreven. Het schrijven van de testen loopt synchroon met het ontwikkelen. Het testen heeft drie doelen:

- Controleren of aanpassingen geen negatief effect hebben op de andere code
- De testen tonen de afwezigheid van bugs.
- De testen tonen aan dat requirements zijn behaald

Om een brede dekking te krijgen ben ik van plan de volgende soorten testen te gebruiken:

- **Unit Test**: Testen componenten onafhankelijk van elkaar testen.
- **Integration Test**: Testen de samenwerking tussen alle composten testen.
- **Functional Tests**: Testen de front-end door een gebruiker simuleren
- **Performance Test:** Testen naar het gebruik van de resources van de module.
- **User Test:** Stappen plan dat beschrijft welke resultaat wordt verwacht wanneer een gebruiker een specifieke (volgorde van) handeling(en) uitvoert.

Ik wil graag 80% line-coverage hebben aan het eind van het project. Dit percentage is goed te behalen en dekt een groot stuk van de codebase.
De testmethode en de resultaten zullen worden beschrijven in het Test Plan.

Risico's
Als de planning uitloopt moeten er prioriteiten worden gesteld. Testen is erg belangrijk, maar kan alleen gedaan worden als de code al is geschreven. Hierom is de prioriteit dan bij het ontwikkelen van de module. Onder te testen is er ook onderscheid te maken. De unit en integration tests zijn primair en kunnen samen veel gedaan krijgen.
De performance, functional en user tests zie ik als secondair. Deze tests gaan over het gebruik van de module, en niet de functionaliteiten zelf. Daarom kunnen zij, als er geen tijd meer over is, worden weggelaten.

5 Requirements

Het opstellen van de requirements was een grote onderneming. Er waren veel verschillende stakeholders en veel verschillende visies over het product. In dit hoofdstuk zal ik beschrijven hoe ik het heb aangepakt en hoe ik de problemen heb opgelost. Een meer gedetailleerde beschrijving van de van dit process is te vinden in bijlage A.

5.1 Stakeholders

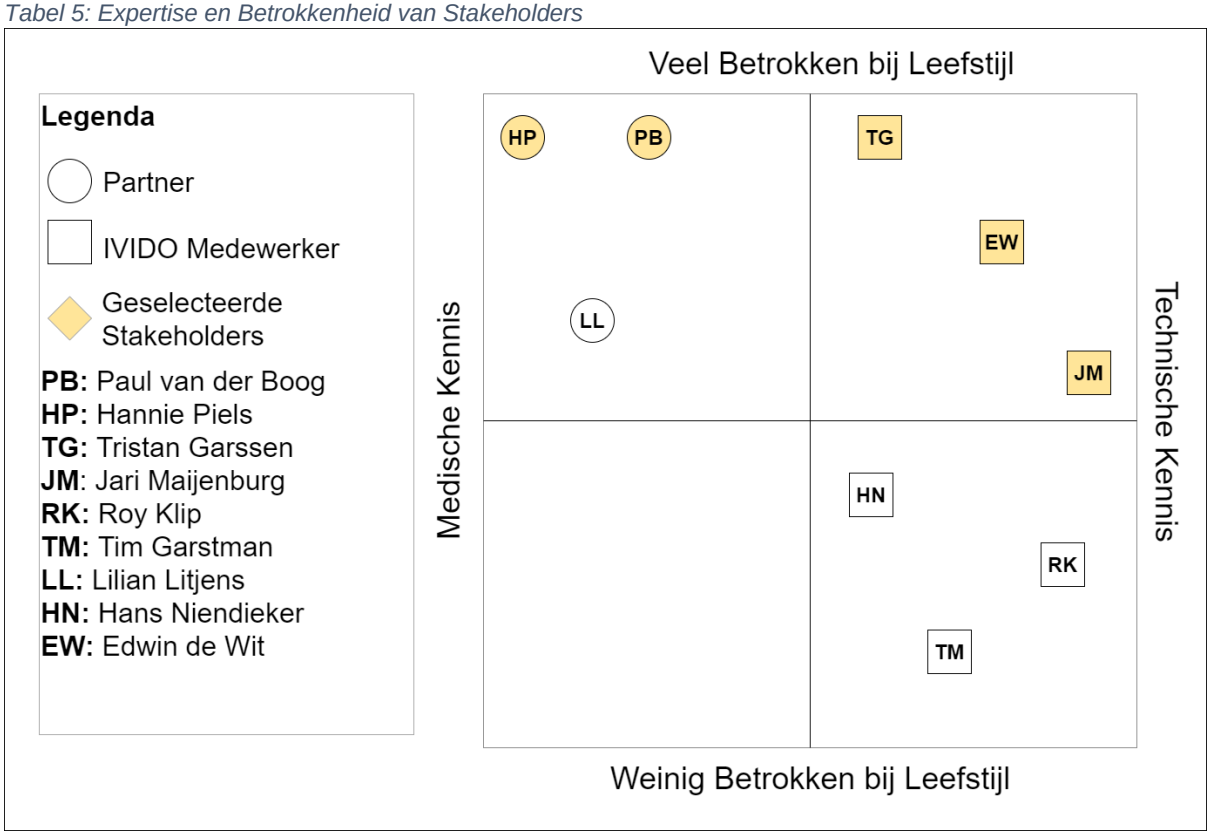
De opdrachtgever heeft een korte lijst met stakeholders geven. Uit gesprekken met deze stakeholders heb ik nieuwe stakeholders kunnen achterhalen. Toen geen nieuwe stakeholders meer kon vinden heb ik ze allemaal in een lijst gezet. Hierbij heb ik ook beschreven welke expertises zij hebben en welke groep die zij vertegenwoordigen. Zo wist ik met wie ik moest praten voor elk onderwerp. Deze lijst is te vinden in Tabel 4.

Tabel 4: : Expertise en Betrokkenheid van Stakeholders

Stakeholders			
Naam	**Functie**	**Vertegenwoordigd**	**Reden voor requirements**
Paul van der Boog	Arts bij LUMC	Arts / Zorgprofessionals	Kan medische en functionele vraag van zorgprofessionals omschrijven.
Hannie Piels	Lifestyle Coach	Lifestyle Coach	Heeft kennis over leefstijl coaching en heeft de voeding applicatie Bonstat beheerd.
Tristan Garssen	COO IVIDO	Project Manager / Opdrachtgever	Heeft visie over het project en richting die de PGO op moet
Jari Maijenburg	CTO IVIDO	Technische Infrastructuren, Back-end Design en bedrijfsmentor	Veel kennis over het technische ontwerp van de PGO en niet Functionele Technische requirements.
Roy Klip	Lead Front-end Developer IVIDO	Front-end Design van de PGO	Veel kennis over de front-end van IVIDO
Tim Garstman	UX Designer IVIDO	UX Design	Weet hoe een applicatie overzichtelijk en gebruiksvriendelijk gemaakt kan worden.
Lilian Litjens	CFO IVIDO, BLOOM Consult	Voeding-Technologie consult	Veel kennis over voeding en hoe de voeding applicatie gebruikt moet worden.
Hans Niendieker	CEO IVIDO, Bestuur HINQ	Zorginstelling HINQ	Veel kennis over innoveren in de zorg en de visie van IVIDO en hoe de module erin past.
Edwin de Wit	Product Owner IVIDO	De features in de PGO als geheel	Heeft een visie over de samenhang van de producten binnen de PGO

In de lijst staan negen stakeholders. Zes hiervan zijn medewerkwerkers van IVIDO, drie hiervan zijn externe partners. Deze externe partners helpen IVIDO met de Medische aspecten van de PGO. Met al deze stakeholders zijn interviews afgenomen over hun visie op het project. Na een aantal interviews viel het op dat er veel verschillende verwachtingen waren van de module. De wensen lagen zo ver uit elkaar dat het resulterende product niet in de gegeven tijd ontwikkeld zou kunnen worden. Om meer richting te geven aan het product, en om meer tijd verlies te voorkomen, is samen met de opdrachtgever besloten om het aantal stakeholders te verminderen.
Om te helpen bij deze beslissingen is er een overzicht gemaakt van de betrokkenheid bij het project en de expertises van de stakeholders. Zie Tabel 5 voor dit overzicht.

Tabel 5: Expertise en Betrokkenheid van Stakeholders



Uit eindelijk zijn er vijf stakeholders gekozen om mee door te gaan. De selectie van stakeholders waren het meest betrokken bij Leefstijl en spande het volledige spectrum van expertises. De stakeholders zijn gekozen om de volgende redenen.

- Paul van der Boog: Door zijn vele medische kennis en visie op het product, is hij een belangrijke stakeholder in het proces.
- Hannie Piels: Als leefstijlcoach is zal zij ook een gebruiker zijn van de module. Hierdoor hier haar mening erg waardevol.
- Tristan Garssen: Als opdrachtgever is hij verantwoordelijk voor het product, en is een goede schakel binnen IVIDO.
- Edwin de Wit: Als Product Owner kan hij veel zeggen over de richting van de PGO
- Jari Maijenburg: Als Lead Developer kan hij veel zeggen over de technische implementatie van het project.

Nu de er minder stakeholders betrokken worden bij het project, kan er een beter focus worden gelegd op het hoe het eind product eruit moet zien. De andere stakeholders zijn nog steeds belangrijk voor het onderwerp Leefstijl, maar worden pas meer betrokken bij het ontwikkelen van de module bij een volgende versie.

5.2 Marktonderzoek

In een van de eerste gesprekken met Paul van der Boog en Hannie Piels is besloten dat de module een soort ‘Voedingsmonitoring Applicatie’ zou worden. In de App Store zijn er veel van dit soort applicaties te vinden. Om mij in te leven als gebruiker heb ik een drietal van deze applicaties gedownload en voor een week gebruikt. Op deze manier hoopte ik uit te vinden wat een gebruiker belangrijk vindt in zo’n applicatie. Ik zal kort toelichten wat ik heb gevonden, een volledige analyse is te vinden als bijlage van het Requirements Rapport (A). Ik heb de drie grootste applicaties op dit gebied gedownload en gebruikt.

- **MyFitnessPal:** Met 40 miljoen gebruikers de grootste applicatie
- **Lose It!**: Focust zich sterk op de interface en het gebruikers gemak
- **FatSecret:** Focust zich op het gemotiveerd houden van hun gebruikers

Overeenkomsten
Wat mij opviel is dat de applicaties in features veel overeenkomen, maar in aanpak veel verschillen. De volgende features zijn in alle drie de apps te vinden:

- Een inleiding in het gebruik van de applicaties
- Een ‘gepersonaliseerd’ voedingsplan
- Een uitgebreide dataset aan voeding.
- Herinneringen om je voeding in te vullen. Mailtjes én/of Push Notificaties
- De mogelijkheid om barcodes te scannen met de camera en zo het product te vinden.

Verschillen in aanpak
Ieder app had een andere aanpak. MyFitnessPal kan gelinkt worden aan andere apps die de data met elkaar delen. Zelf vond ik dat “Lose It!” het makkelijkste te gebruiken was, en FatSecret had een sociaal aspect. Gebruikers kunnen hun overwinningen en foto’s van hun gezonde eten delen op het platform. Zo kunnen alle andere gebruikers het zien, en elkaar aanmoedigen. Dit inzicht was interessant omdat het aantoonde dat het veranderen van het dieet op verschillende manieren aangepakt kan worden.

Geleerde Informatie
Uit het mijn persoonlijke onderzoek heb ik de volgende lessen geleerd. Deze wil ik meenemen in het ontwerpen van de module.

- Een intuïtief menu is maakt het gebruik erg fijn.
- Een uitgebreide database is belangrijk om voeding goed in te voeren.
- Een introductie helpt gebruiker veel in het begrijpen van de app.

5.3 Interviews

De meeste requirements zijn geëliciteerd door interviews af te nemen. De interviews zijn genotuleerd met toestemmen van de stakeholders. Door hun omvang zijn ze niet opgenomen als bijlagen. In plaats daarvan zal ik een aantal belangrijke moment toelichten. Deze gesprekken zijn in chronologische volgorde gevoerd.

Initiële Visie
In het eerste gesprek met Paul van der Boog en Hannie Piels er gekeken naar de Scope. Het originele plan was om een overkoepelende module te maken die verschillende applicaties voor leefstijlverbetering zou koppelen. Samen besloten we dat dat dit realistisch was om te maken in het timeframe. Met het oog op kwaliteit over kwantiteit is er gekozen om te focussen op een leefstijldoel: Voeding. Dit was om de volgende redenen:

- Het is de grootste oorzaak van chronische aandoeningen op later leeftijd
- Het is relatief makkelijk aan te pakken, door voeding bij te houden
- Paul en Hannie hebben veel medische kennis over dit onderwerp.

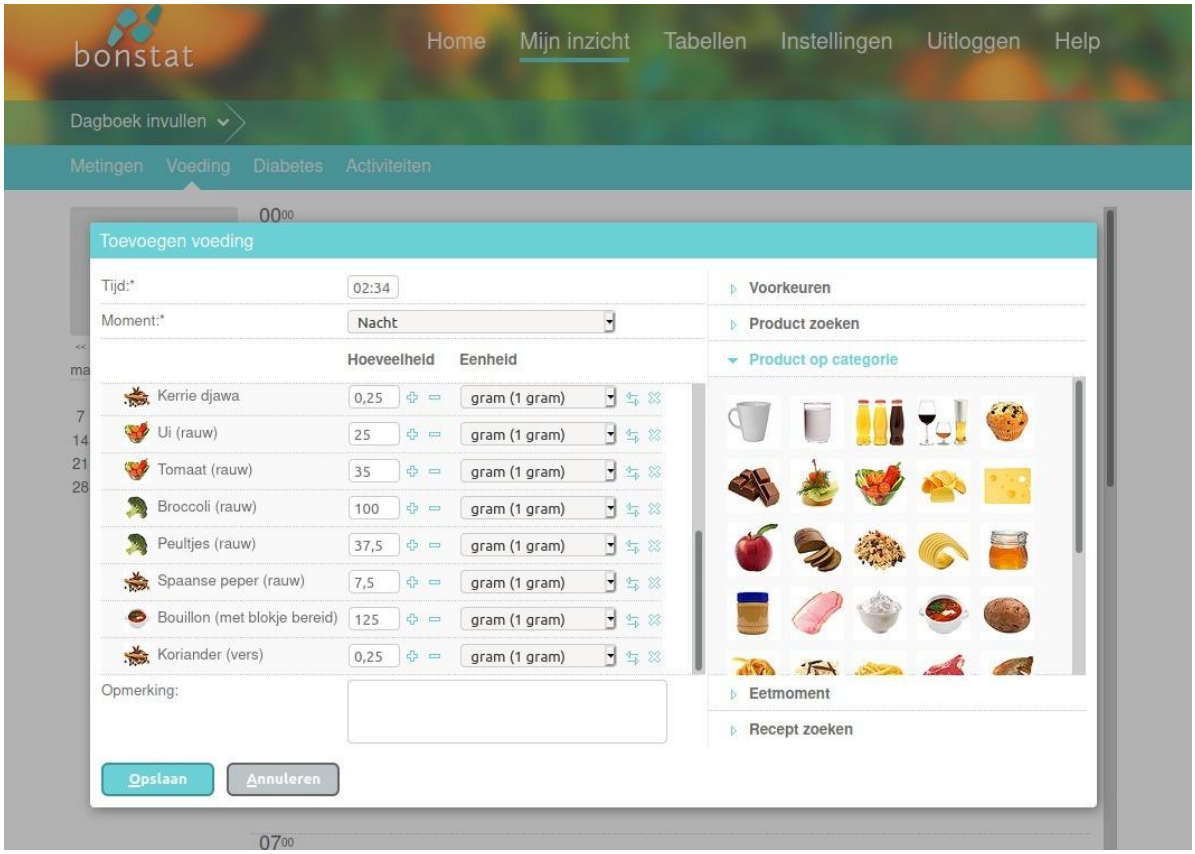
Deze beslissing heeft de doelstelling zoals die is beschreven in 3.3 bepaald.

De applicatie zelf zou makkelijk moeten zijn in gebruik. Het bijhouden van voeding is voor de meeste mensen een klus. Door dit makkelijker te maken hoopten we dat het meer gebruikt zal worden. Ook moest de applicatie niet te streng zijn. Om structurele verbetering te krijgen in iemand voedingspatroon, moet zijn of haar leefstijl ook veranderen. Dit is beter te bereiken in samenwerking met de arts en patiënt, op een constructieve manier.

Bonstat Applicatie

In 2005 zijn Paul en Hannie samen een het bedrijf Bonstat opgezet. Zij wilden patiënten met nierproblemen beter te ondersteunen. Via een website konden de patiënten hun voedingen bijhouden. Zorgprofessionals konden deze informatie inzien, en dit gebruiken bij hun behandeling.

Deze website[4] is ontwikkeld door het software bureau ASolutions. Zie Figuur 1 voor een screenshot van de site. Paul en Hannie zijn tevreden over de functionaliteiten van de website, maar vinden dat hij onderhand is verouderd. Daarom willen Paul en Hannie de applicatie updaten. Hiervoor hebben zijn IVIDO benaderd. Zij zijn een partnership aangegaan om de applicatie te implementeren in de PGO.



Figuur 1: Screenshot van Bonstat Applicatie

Het idee was om de functionaliteiten van de applicatie over te zetten in de PGO. De Bonstat applicatie voldoet namelijk al aan functionele wensen van Paul en Hannie.
Ik heb de code van de Bonstat applicatie gekregen om te kijken of ik deze kon gebruiken. Het idee kwam om op de back-end van applicatie te hosten op de server van IVIDO zodat alle functionaliteiten hergebruikt konden worden. Dit zou veel tijd kunnen schelen.

[4] https://www.bonstat.nl/

Afstappen van Bonstat
Het analyseren van de code van Bonstat was erg moeilijk. Dit was omdat er geen comments te vinden in het project. In correspondentie kreeg ik te horen dat dit niet behoorde tot de ontwikkel filosofie van ASolutions. “We schrijven de code zo duidelijk, dat comments niet nodig zijn”. Verder waren er nog meer complicaties:

- De applicatie draaide Java, dit wordt haast niet gebruikt binnen IVIDO
- De applicatie werd te groot geacht voor de microservice infrastructuur van IVIDO.
- De applicatie maakte gebruik van SOAP, IVIDO maakt gebruik van REST.
- Veel van de dependencies die de applicatie gebruikte werden niet meer ondersteund. Dit is een security problemen.

Dit alles maakte dit plan een stuk minder aantrekkelijk. In overleg met de Paul en Hannie, is er besloten om van het plan af te zien. De back-end van Bonstat zou niet meer worden gebruikt bij het ontwikkelen van de module.

De database daarentegen zou wel nog gebruikt kunnen worden. In de database waren belangrijke connecties gelegd tussen producten, productgroepen en eenheden. Deze data maakt het erg makkelijk voor gebruikers om hun genuttigde producten te vinden.

Visie van Management
Twee weken in het proces is gesproken met Hans Niendieker. Als CEO van IVIDO heeft hij het heel druk, en was het moeilijk om een afspraak met hem te maken. In het gesprek vertelde ik wat de planning was, en dat de focus lag op Voeding. Hij was verbaasd over deze richting omdat het niet overeenkwam met zijn eigen visie voor de PGO. Hij zou graag zien dat het meer overkoepelend was. Op deze manier zou het meer Business Value toevoegen van het PGO product. Hij vertelde over tools binnen die IVIDO die gekoppeld konden worden de ‘Personal Battery Tool’ die helpt bij het meten van stress en de ‘iCOPE’ module die patiënten helpt bij het aannemen van een nieuwe leefstijl.
Dit dreigde de opdracht een hele andere kant op te duwen. Ook was ik niet bekend met de tools waar hij het over had. Zijn mening als CEO van IVIDO is erg belangrijk, dat maakte het moeilijk om ‘Nee’ te zeggen tegen deze koersverandering. Uiteindelijk is er compromis gesloten. De focus op Voeding zou worden doorgezet, maar de ‘iCOPE’ module zou worden opgenomen in de Scope. Deze module zou de verandering in leefstijl ondersteunden die een nieuw dieet met zich meebrengt.

Patient Journey en Casus
Toen de richting voor de module duidelijk was, is er met Paul en Hannie gekeken naar de Patient Journey. Dit is de ‘reis’ die een patiënt aflegt bij het gebruiken van de PGO. Samen hebben we gekeken naar elke ‘stap’ die een patiënt moet zetten, en daarbij de passende functionaliteiten beschreven.
Ik heb extra mijn best gedaan om mogelijke uitzonderingen op het verhaal te vinden door het proces van verschillende kanten aan te belichten. Ook heb ik expres ‘domme’ vragen gesteld zodat stappen nog een keer uitgelegd konden worden. Als er dan verschillen waren in de uitleg kon ik hier naar doorvragen.
Ik heb veel geleerd bij het opstellen van de ‘Patient Journey’. Hannie merkte op dat ik nog wel wat fundamentele aspecten miste van de behandeling, en stelde voor om een casus te schrijven. Deze casus zou beschrijven hoe een behandeling er normaal uitziet.

5.4 Casus

De stakeholder Hannie Piels heeft een opleiding gedaan om Leefstijl Coach. Met deze praktische kennis kon ze uitleggen wat de tekortkomingen zijn van de huidige begeleiding en hoe de module deze kan verbeteren. Ze heeft een Casus geschreven over een patiënt die leefstijlondersteuning krijgt. Deze casus is te vinden als bijlage van het Requirements Rapport, bijlage A.

Het persona in de casus heet Jaap. Hij is 52 jaar, heeft longproblemen en fors overgewicht. Op verzoek van zijn vrouw gaat hij naar de leefstijl coach, met wie hij samen zijn dieet vastlegt. De casus laat goed zijn dat de patiënt niet altijd al zijn voeding (correct) invult. Hij vergeet wel eens wat of hoeveel hij eet. Dit komt omdat best veel tijd kost, en je niet gewent bent om zo over je voeding na te denken.

De casus brengt twee punten aan het licht. Ten eerste laat het zien hoe ingewikkeld het bijhouden van de voeding kan zijn. Het is belangrijk dat de module is gemakkelijk maakt, zodat het goed gebruikt wordt. Daarnaast gaf het ook dat het ‘Dietary History’ niet heel accuraat hoeft te zijn. De leefstijlcoach hoeft alleen een globaal inzicht te krijgen in het dieet van de patiënt. De adviezen die een coach geeft zijn vaak algemene dingen zoals “Drink minder koffie” of “Eet niet zo veel snoep tussendoor”. Hiervoor hoeft de coach niet zo veel te weten over het dieet.

Ik heb geen nieuwe requirements kunnen eliciteren uit casus, maar het heeft me wel geholpen met het prioriteren van al bestaande requirements. Zo werd het duidelijk dat het invullen van de voeding erg belangrijk was, en dat de overzichten makkelijker konden.

5.5 Productbeschrijving

Met de Patient Journey, de casus en alle eerder verzamelde informatie is er een productbeschrijving gemaakt. Hierin wordt de kern van de module beschreven.

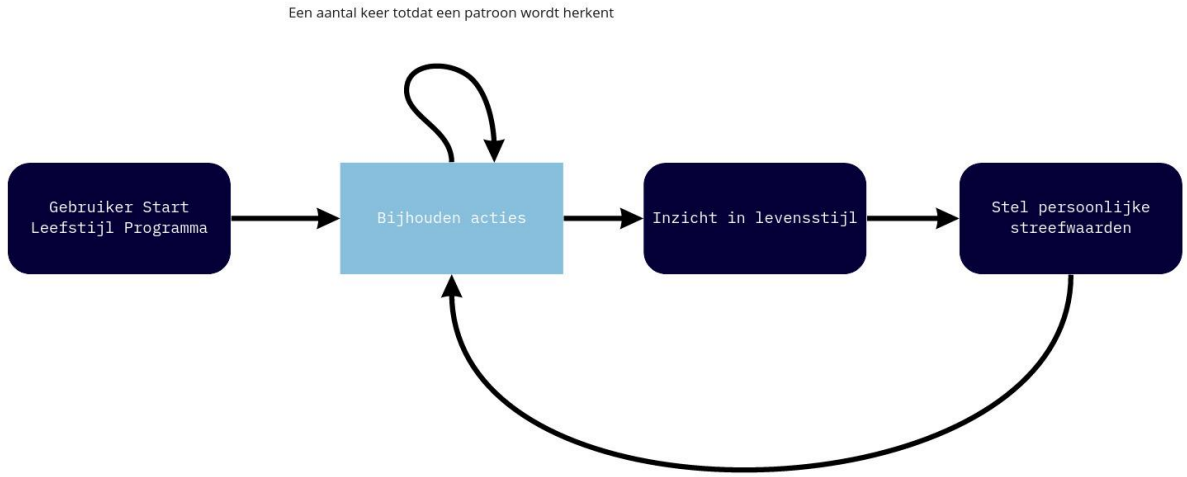
Beschrijving
De leefstijlmodule is er voor patiënten om hun acties te monitoren en zo een inzicht te krijgen in hun leefstijl. Met dit inzicht kan er gekeken worden wat er gedaan kan worden om de leefstijl van de patiënten te verbeteren.

Proces

1. De gebruiker start het leefstijl programma. Er zijn verschillenden doelen waar de gebruiker op kan focussen en data input op kan geven.
2. De gebruiker monitort relevante acties die bij dit doel horen. Zo kan een gebruiker bijhouden hoe vaak hij of zij sport, of wat hij of zij allemaal eet. Dit moet een aantal keer bijgehouden worden zodat er patronen in de leefstijl kunnen worden gevonden.
3. Aan de hand van deze invoer krijgt de gebruiker een inzicht in zijn of haar levensstijl. Met dit inzicht kunnen verstandige keuzes worden gemaakt over welke aanpassingen de gebruiker moet nemen om zijn of haar leefstijl te verbeteren.
4. Aan de hand van deze aanpassingen kunnen er persoonlijke streefwaarden worden opgesteld. Dit kan dan iets zoals ‘Minimaal 5000 stappen per dag’ of ‘Probeer 8 uur per dag te slapen’.

5. De gebruiker gaat daarna weer door met het bijhouden van zijn of haar acties. Met de kennis dat het inzicht heeft gebracht is het de bedoeling dat minder slechte keuzes in leefstijl worden gemaakt.
6. Na een bepaalde tijd kan opnieuw worden gekeken hoe de leefstijl van de gebruiker invloed heeft op zijn of haar gezondheid, en kunnen de streefwaarden worden aangepast.

Zie Figuur 2 voor een diagram het proces.



Figuur 2: Bedoeld gebruik van de module

Product Visie

Het is de bedoeling dat het product een ondersteuning biedt aan het huidige proces van leefstijl coaching, het moet niet het zorgproces overnemen. Daarom is het ook niet de bedoeling dat de module het zelf met een advies komt, maar eerder dat er aan de hand van het inzicht in de leefstijl zelf actie ondernomen wordt.
Omdat het deze ondersteunende rol inneemt is het ook niet de bedoeling dat het erg streng is voor de gebruiker. Een gebruiker hoeft niet iedere dag alles bij te houden. Zolang er genoeg data is om een patroon te herkennen kan het product gebruikt worden. Zo zijn streefwaarden ook eerder richtlijnen in plaats van maxima en minima.
Vanuit het zorgperspectief zien de artsen graag dat het product in samenwerking wordt gebruikt. Door de zorgprofessional en de patiënt dezelfde inzichten en functionaliteiten te geven voelt de behandeling meer als een gezamenlijke proces in plaats van dat de patiënt verplichtingen wordt opgelegd door de zorgprofessional. Dit zorgt voor meer verantwoordelijkheid voor de patiënt en verhoogt de kans dat de veranderingen een leefstijl langer aanhouden.

Plaats in PGO

Deze leefstijl module is kan erg waardevol zijn voor de PGO van IVIDO. Het is bedoeling dat gebruikers van de module uiteindelijk een gezondere leefstijl aannemen. Een gezonde leefstijl verlaagt de kans op aandoeningen zoals diabetes type-2 of kanker. Dit heeft als gevolg er mensen langer en gezonder leven, dat er minder druk komt op de zorg en daarmee ook geld bespaard kan worden door zorgverzekeringen. Zorgverzekeringen zijn daarom bereid om te investeren applicaties met dit soort functionaliteiten.

5.6 Vastleggen van Requirements

Ik merkte dat in de laatste paar gesprekken de focus steeds meer op de implementatie en het ontwerp lag, en niet meer zo zeer op de functionaliteiten. Hierdoor heb ik besloten om de requirements vast te leggen, en over te gaan naar de volgende stap.

Dit was ook nodig omdat de planning te bewaken. Door de process af te ronden en over te gaan naar het Architectuur Design kon de Scope niet verder uitlopen. Een probleem was wel dat sommige punten nog niet helemaal duidelijk waren. Zo was de precieze implementatie van de 'iCOPE' module nog onduidelijk, en miste er nog details over het precieze invoer proces. De rest van de lijst met requirements was al vaak nagekeken, en goed gekeurd. De uiteindelijke lijst met requirements is te vinden in Tabel 6.

De prioritering van de requirements is volgens de MoSCoW methode gedaan. In overleg met de stakeholders en de opdrachtgever is prioriteit bepaalde. De weging was als volgt:

- **Constraint (X)**: Een verplichting aan het product, buiten het ontwerp om. Geen enkele ontwerp keuze kan invloed hebben op.
- **Must (M):** Een van de belangrijkste dingen aan het product. Zonder deze eis kan de MVP niet opgeleverd worden.
- **Should (S):** Een Should weegt het zelfde als een Must, maar kan bereikt worden via een omweg of een 'Quick Fix'. In het uiteindelijke ontwerp zou dit wel goed geïmplementeerd moeten worden.
- **Could (C):** Een mogelijke feature van het systeem, maar hoeft niet deel te zijn van deze versie van het product.
- **Won't (W):** Zal geen deel uitmogelijk van deze versie van het product. Dit kan ook gebruikt worden om te beschrijven wat er expliciet niet in het product komt te zitten.

Er is ook onderscheid gemaakt tussen Functionele en Niet Functionele Requirements. Een functionele requirement beschrijft iets dat het product moet kunnen doen, een Niet Functionele Requirement beschrijft een eigenschap van het product. Deze eigenschap is daarna ook nog expliciet gedefinieerd.

Er zijn wegingen gemaakt aan prioriteit die de stakeholder geeft aan een requirement De prioriteit van de stakeholder is bepaald door de stakeholder die de requirement heeft aangedragen. De Stakeholder Prioriteit verschilt van de Requirement Prioriteit. De stakeholder Prioriteit gaat alleen uit van de verwachting van de stakeholder, terwijl de Requirement Prioriteit rekening houd met de implementatie en architectuur. De waarden zijn op de volgende manier gedefinieerd:

- **High:** De stakeholder wil dit het snelst ontwikkelt zien worden
- **Medium:** De stakeholder wil dat dit pas ontwikkelt wordt wanneer de basis van het product staat.
- **Low**: De stakeholder zou het fijn vinden om dit te zien in de applicatie, maar mag in de laatste fase ontwikkeld worden.

Tabel 6: Requirements van Leefstijl Module

ID	Beschrijving	P	SG	RT	RE	SP	AP	DB	SB	V
M-1	De module moet voor 8 januari ontwikkeld zijn	X	M	NF	Deadline	H	H	PP	AO	Deadline behaald
O-1	De module moet geïntegreerd zijn in de PGO van IVIDO	X	O	NF	Compatibility	H	H	RI-1	TG	Bereikbaar vanuit PGO
O-2	De front-end moet ontwikkeld worden in React	X	O	NF	Compatibility	H	H	RI-5	RK	Geen andere code
O-3	De module moet gehost worden in het cluster van IVIDO	X	O	NF	Deployment	H	H	RI-5	JM	Zelfde cluster
M-2	De module mag niet meer dan €50,- per maand aan aanvullende kosten brengen	X	M	NF	Cost	M	H	RI-11	TG	Kosten Overzicht
O-4	De module moet zijn resources evenredig schalen met gebruik	M	O	NF	Scalability	H	H	RI-5	JM	Test
M-3	De module ondersteund uitbreiding van meerdere vormen van leefstijl verbetering	M	M	NF	Extensibility	H	H	RI-1	TG	Test
Z-1	De behandelingdeelnemer kan de ingevoerde voeding inzien	M	Z	F	-	H	M	RI-1	PB	Test
P-1	De behandelingdeelnemer kan voeding invoeren	M	P	F	-	H	L	RI-1	PB	Test
O-5	De module mag maximaal 100MB aan RAM opnemen	M	O	NF	Resource Management	M	H	RI-6	JM	Health Monitoring
O-6	De module moet binnen 5 seconden opstarten	M	O	NF	Resource Management	M	H	RI-6	JM	Test
O-7	De module moet voldoen aan alle punten van de OWASP Top 10	M	O	NF	Security	M	H	PP	AO	Maatregelingen getroffen
O-8	De module moet een test code coverage hebben van minstens 80%	M	O	NF	Testability	M	M	PP	AO	Static Code Analysis
P-2	De schermen moeten responsive zijn	S	P	NF	Compatibility	H	M	RI-1	HP	Test
O-9	De module moet minder dan 10 uur aan onderhoud per maand kosten	S	O	NF	Maintainability	H	H	RI-5	JM	Beoordeling door Lead Developer
M-4	De module moet het iCOPE-proces ondersteunen	S	M	F	-	M	L	RI-8	HN	Test en Verificatie door Stakeholder
Z-2	De patiënt moet gekoppeld kunnen worden aan persoonlijke referentiewaarden	S	Z	F	-	M	L	RI-2	PB	Test
P-3	De patiënt kan voorkeursinstelling opslaan	C	P	F	-	M	L	RI-2	HP	Test
Z-3	De behandelingdeelnemer moet de voedingswaarden kunnen vergelijken met persoonlijke referentiewaarden	C	Z	F	-	M	L	RI-2	PB	Test
P-4	De behandelingdeelnemer moet een introductie krijgen over het gebruik van de module	C	P	F	-	L	L	BA	G	Test
P-5	De patiënt kan aan de hand van een barcode voedingsmiddelen vinden in de module	W	P	F	-	M	L	BA	G	Test

Toelichting Tabel 6

- ID: Referentie naam van requirement
- Beschrijving: Beschrijving van de requirement
- P: Prioriteit van de Requirement volgens de MoSCoW methode
    - X: Constraint, eis buiten het process waar niet om heen kan
    - M: Must, moet deel zijn van het ontwerp
    - S: Should, moet deel zijn van het ontwerp, maar kan ook behaald worden met een omweg.
    - C: Could, kan deel zijn van het ontwerp wanneer als er tijd voor is
    - W: Won't, is niet deel van het huidige ontwerp
- SG: Stakeholder Groep, groep van stakeholder waarvoor deze requirement het meest relevant is:
    - M: Management, process beheer en management functies binnen IVIDO
    - O: Ontwikkelaar: programmeurs binnen IVIDO
    - P: Patiënt, gebruiker van de module die zijn leefstijl wil verbeteren
    - Z: Zorgprofessional, gebruiker van de module met veel medische kennis die een patiënt begeleid.
- RT: Requirement Type
    - F: Functioneel, beschrijft een functie die de module moet hebben
    - NF: Niet Functioneel, beschrijft een eigenschap die de module moet hebben
- RE: Requirement Eigenschap, beschrijft de eigenschap die een Niet Functionele Requirement probeert te beschrijven
- SP: Stakeholder Prioriteit, beschrijft de prioriteit die de relevante stakeholder Groep stelt aan de requirement. Dit wordt gebruikt om architectuur keuzes te maken.
    - H: High, hoogste prioriteit
    - M: Medium, gemiddelde prioriteit
    - L: Low, Lage prioriteit
- AP: Architecture Prioriteit, beschrijft de invloed dat de requirement heeft om de architectuur van de module. Dit wordt gebruikt om architectuur keuzes te maken.
    - H: High, hoogste prioriteit
    - M: Medium, gemiddelde prioriteit
    - L: Low, Lage prioriteit
- DB. Document Bron, beschrijft het document waarin requirement staat beschrijven
    - RI-X: Requirement Interview
    - PP: Project Plan
    - BA: Bestaande Apps
- SB: Stakeholder Bron, Persoon of Organisatie die de Requirement heeft aangeleverd
    - AO: Afstudeeropdracht
    - TG: Tristan Garssen
    - JM: Jari Maijenburg
    - RK: Roy Klip
    - PB: Paul van der Boog
    - HN: Hans Niendieker
    - G: Gebruikers Onderzoek
    - HP: Hannie Piels
- Verificatie: Manier waarop de requirement kan worden geverifieerd.

Ook is er een weging gemaakt van de impact op de architectuur. Dit is gedaan in samenwerking met de Lead Developer. De schalen zijn op de volgende manier gedefinieerd:

- **High:** De implementatie heeft invloed op bijna alle andere keuzes
- **Medium:** De implementatie heeft mogelijk invloed op andere implementaties
- **Low**: De implementatie heeft vrijwel geen invloed op andere requirements

Als laatste is er ook een verificatie methode besloten. Door dit vast te leggen was er een duidelijke manier te bepalen of er aan de requirement werd voldaan. Voor de meeste Technische Requirements kunnen testen worden geschreven, voor sommige niet functionele testen moet de evaluatie door gemaakt worden door de stakeholder die het requirements heeft aangedragen. Hier bij kan onduidelijkheid bij ontstaan, daarom is het belangrijk om vast te leggen wat de verwachtingen zijn. Voor traceability is naast de verantwoordelijke stakeholder, ook beschreven in welke bron het is hoor het eerst is opgekomen.

Tijdens de planning heb ik besloten dat ik ADD wilde gebruiken (zie punt 4.3). Hierom heb ik de Niet Functionele Eigenschap, Stakeholder Prioriteit en Architectuur Impact meegenomen in de beschrijvingen van de requirements. Deze zijn nodig bij de ADD Methode.

Veranderingen in de requirements zijn bijgehouden in een versie document. Deze is te vinden als bijlagen van het Requirements Rapport, zie bijlagen A.

5.7 Invloed op planning

Het opstellen van de requirements langer duurde dan verwacht. Hierdoor moet ik de planning aanpassen. Ik verwachtte nog steeds even veel tijd kwijt te zijn met het ontwerpen van de architectuur, en wil graag de laatste week weken aan buffer behouden voor mogelijke andere vertragingen. Dit betekend dat ik tijd van de ontwikkelfase opneem. Dit is mogelijk om dat er meer tijd is uitgetrokken voor deze beroepstaak dan elke andere. Zie Tabel 7 voor de aangepaste tijdbesteding en Tabel 8 voor de nieuwe tijdlijn.

Tabel 7: Fases in Afstudeertraject

Fase	Resultaat	Duur
Aanpak en Onderzoek	Plan van Aanpak	1 Week
Requirements	Requirements Rapport	4 Weken (2)
Architecture	Architecture Design en Proof of Concept	2 Weken
Ontwikkelen & Testen	MVP van Opdracht en Testplan	8 Weken (10)
Oplevering	Afstudeerverslag	2 Weken
Originele Planning	Twee weken uitloop van requirements	-

Tabel 8: Aangepast planning na uitloop requirements

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

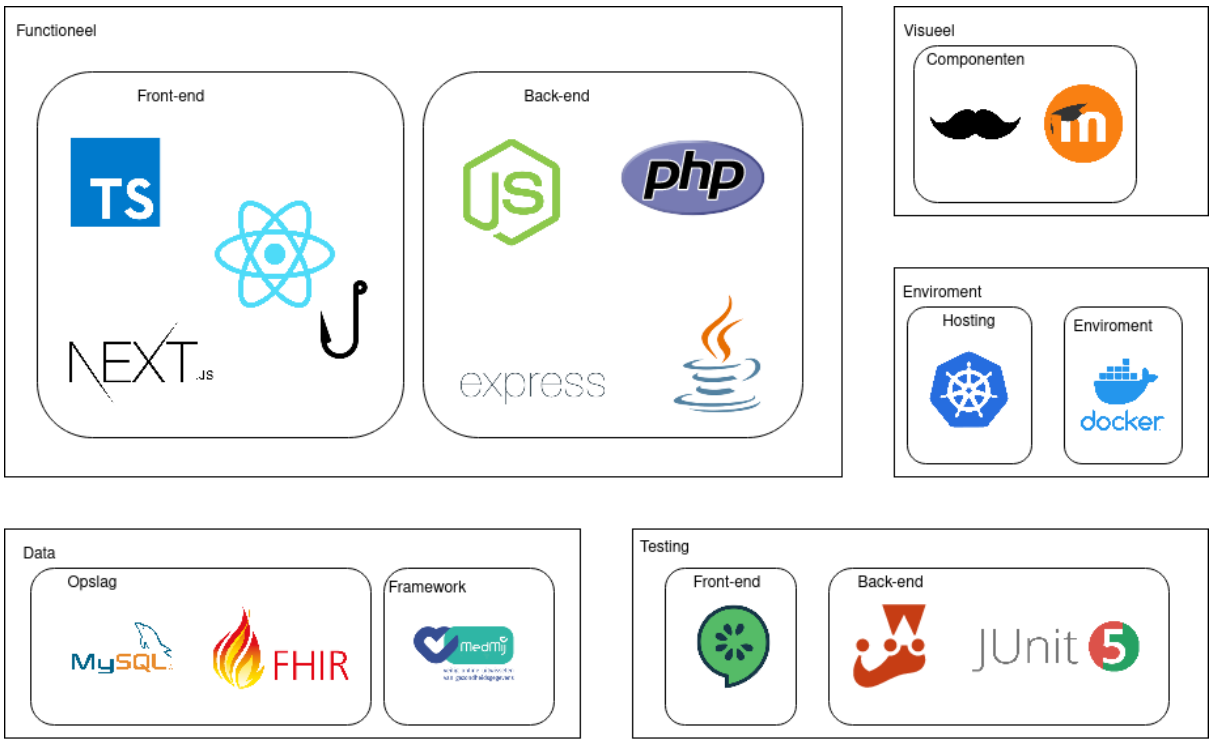
Het uiteindelijke ontwerp heeft maar een klein aantal van de requirements kunnen vervullen. De prioritering heeft veel geholpen bij het bepalen van de belangrijkste features.

6 Architecture Design

De volgende stap in het proces was het maken van het Architecture Design. Met de opgestelde requirements konden de beste tools en Implementaties voor de opdracht worden bepaald.
Als eerst is gekeken welke tools IVIDO gebruikt. Voor dit project is het verstandig om tools te kiezen die IVIDO al gebruikt, omdat de opdracht overgeleverd moet worden aan het eind. Als de tools al bekend zijn bij de developers, kunnen zij de module makkelijker ondersteunden.
Daarna heb ik het ontwerp proces uitgevoerd. Dit proces is gegaan volgends de ADD methode. Deze methode maakt een iteratief ontwerp en gebruikt de eigenschappen van Niet Functionele Requirements om de keuzes te maken. Omdat er zo veel Niet Functionele Requirements waren, is dit een effectieve methode.
Een gedetailleerde beschrijving van het ADD proces is te vinden het Architecture Design Document in bijlage B.

6.1 Huidige Tech Stack van IVIDO

De eerste versie van de PGO is ontwikkeld in PHP met het Moodle Framework. Toentertijd hebben zijn er daarvoor gekozen door de flexibiliteit. Maar merken ze dat ze de houdbaarheid van het Moodle Framework bereikt en zijn ze aan het veranderen.
De nieuwe architectuur heeft een Microservice Infrastructuur. Dit is makkelijker te schalen en uit te breiden. Het omschrijven naar het ontwerp is een grote klus, en omdat ze veel proberen te innoveren gaat dit gestaagd. Daarom gebruiken ze nu meerdere technologieën door elkaar. Zie Figuur 3 voor een overzicht van alle tools die IVIDO gebruikt.



Figuur 3: Huidige Techstack IVIDO

De PGO is aangesloten op het MedMij Framework. Dit Framework maakt het mogelijk om Medische Data uit te wisselen tussen aangesloten partijen. Dit is erg belangrijk voor IVIDO omdat de uitwisselingen van medische data de basis is van hun business model. Deze medische data is gevormd in FHIR-Resources, de Industry Standard voor medische data opslag.

De module die ik ga ontwerpen moet geïntegreerd worden in de PGO. Omdat de PGO wordt herschreven naar de microservice architectuur moet de module ontworpen worden als een microservice. Er werd mij verzocht zo veel mogelijk tools te kiezen IVIDO al gebruikt. Dit zou de overdracht en integratie veel gemakkelijker maken. Voor oplossingen waar zij nog geen tools voor hadden gekozen kreeg ik de vrijheid om zelf een keuze te maken zodat ik het voor hun kon uittesten. De volgende dingen werden mij aangeraden door de Technical Lead:

- Gebruik een programmeertaal die bekend is bij IVIDO. Met Node.js als voorkeur
- Ontwerp de module als een Microservice. Dit past goed in hun architectuur.
- Medische gegevens worden opgeslagen in FHIR Structuren (JSON). Hierom wordt een noSQL database aangeraden.

6.2 Aanpak ADD Methode

De ADD methode werkt met iteraties in het ontwerp. Aan het begin van iedere iteratie wordt er een niveau gekozen waarvan het ontwerp wordt geanalyseerd. Voor elk component op dat niveau wordt de best passende keuze gemaakt. Dit gebeurd door eerst te bepalen aan welke eigenschappen het component moet voldoen. Daarna worden er een aantal implementaties bedacht voor dat component. Voor elke gevonden implementatie wordt gekeken hoe goed deze aansluit op de eigenschappen. Op basis van onderzoek en eigen interpretatie wordt een cijfer bepaald. Deze cijfers worden als volgt gewogen:

1. Sluit zeer slecht aan bij de driver
2. Heeft maar enkele aansluiting met de beschreven driver
3. Heeft redelijke aansluiting op driver maar is misschien niet optimaal
4. Heeft goede aansluiting op driver maar heeft wat extra werk nodig.
5. Sluit uitstekend aan op driver

Wanneer er geen keuzes meer gemaakt kunnen worden, wordt de iteratie afgerond. De volgende iteratie behandelt een nieuw onderwerp op de zelfde schaal dat nog niet is behandeld, of een ander onderwerp op een kleinere schaal.

De eerste implementatie zal zich focussen op de applicatie in het geheel. Hier zullen de programmeertalen en database worden besloten. De tweede iteratie zal meer ingaan het Globale Structuur. Volgende iteraties zullen gaten in het ontwerp opvullen.

6.3 Eerst Iteratie: Applicatie in het geheel

In de eerste iteratie is er alleen gekeken naar de tools en programmeertalen die gebruikt konden worden door de module. Er was al besloten dat de front-end geschreven moest worden in React. Dit was nodig om de om de front-end te integreren in de PGO. Dus bleven alleen de keuzes voor de back-end programmeertaal en de database implementatie over.

Back-end programmeertaal
De programmeertaal van de back-end is vrijwel de belangrijkste keuze voor dit product. Dit bepaald welke libraries gebruikt kunnen worden en heeft invloed de manier van ontwikkelen en de integratie in het systeem. Om deze reden moest er goed worden nagedacht over de keuze. De tools moet goed kunnen schalen, te onderhouden zijn en aansluiten bij de PGO. De volgende drie programmeertalen zijn overwogen:

Java
Ik heb zelf veel ervaring met programmeren in Java. Het is een robuuste taal, wordt op veel plekken ondersteund en heeft een breed aanbod aan libraries. Het is wat 'log' voor het de microservice architectuur die IVIDO in gedachten heeft.

C#
Ik heb ook veel ervaring met C#. Het .NET Framework maakt het sterk in Web Development. Het werkt wat sneller dan Java, maar wordt verder helemaal niet gebruikt binnen IVIDO.

Node.js
Node.js kreeg ik op aanraden mee van de Lead Developer en wordt veel gebruikt binnen IVIDO. Ik heb er weinig ervaring mee, maar omdat het werd aangeraden, neem ik het mee. Het is snel en licht, wat het een aantrekkelijke microservice maakt.

Een analyse van de invloed van de implementaties op de drivers is te vinden in Tabel 9.
Tabel 9: Analyse van back-end taal keuze

ID	Eigenschap	Invloed	Prioriteit	Java	C#	Node.js
M-1	Deadline	Hoog	Constraint	5	4	3
O-1	Compatibility	Hoog	Constraint	4	2	5
O-4	Scalability	Hoog	Must	3	4	5
M-3	Extensibility	Hoog	Must	3	1	5
O-5	Resource Manag.	Hoog	Must	2	4	5
O-6	Resource Manag.	Hoog	Must	1	2	5
O-9	Maintainability	Hoog	Should	3	1	5

Uit de analyse komt Node.js erg goed uit de test. Het scoort laag op de deadline driver, omdat ik de taal eerst nog moet leren voordat ik het goed kan gebruiken. Maar omdat het zeer goed aansluit me de rest van de code in IVIDO is het erg handig om te kiezen. Ook leek het me een mooie uitdaging. Node.js is een taal ik al langer wilde leren om het op veel plekken gebruikt wordt.

Database Implementatie
De module moet voedingsdata in een database. De PGO zelf heeft ook een database, maar kan niet zo maar gebruikt worden. Alle data in deze database moet een FHIR resource voorstellen. Dit is een van de eisen die MedMij stelt aan hun certificatie. Dit maakt het een onhandige keuze. Er moet dus een passende oplossing worden gezocht.

Eigen Database
De module zou een eigen database kunnen hebben voor alle data. Dit maakt het ontwerpen eenvoudig. Maar omdat sommige data uit de PGO database ook opgeslagen zal moeten worden in de eigen database, is er kans op data incongruentie.

Mix van Eigen Database en FHIR
Een eigen database voor voedingsdata, en de PGO database voor medische data, is ook een mogelijkheid. Het combineren van de bronnen kan wel verwarrend worden. Hoe wordt bepaald welke data in welke database wordt opgeslagen.

Alleen FHIR Database
Geen eigen database scheelt hosting en onderhoud. Dit zorgt er wel voor dat de PGO database gevuld wordt met niet medische data. Dit is in strijd met ontwerp van het MedMij Framework. Ook moet de data geformatteerd worden naar FHIR voor dat het opgeslagen wordt. Omdat er geen FHIR resources zijn voor voeding wordt dit erg omslachtig.

Een analyse van de invloed van de implementaties op de drivers is te vinden in Tabel 10.
Tabel 10: Besluit Gebruik van Database

ID	Eigenschap	Invloed	Prioriteit	Eigen	Mix	FHIR
M-1	Deadline	Hoog	Constraint	4	3	1
O-1	Compatibility	Hoog	Constraint	2	4	4
M-3	Extensibility	Hoog	Must	3	4	2
O-7	Security	Hoog	Must	1	4	5
O-9	Maintainability	Hoog	Should	1	4	2

Uit de resultaten blijkt dat de Mix het beste aansluit op de drivers. Hoewel het wat verwarrend kan zijn, werkt deze oplossing naar de beste krachten van beide databases.

Database Structuur
Met de keuze van dat er een nieuwe database ontworpen zou worden, moest er ook besloten welke structuur de database moet hebben. Er zijn twee opties overwogen. SQL, en noSQL. Hoewel de keuze weinig invloed zal hebben op de functionaliteiten, heeft het wel impact op de Scalability en Extensibility van de applicatie.
Een analyse van de invloed van de implementaties op de drivers is te vinden Tabel 11.
Tabel 11: Besluit Database Structuur

ID	Eigenschap	Invloed	Prioriteit	SQL	noSQL
O-4	Scalability	High	Must	2	5
M-3	Extensibility	High	Must	2	4

Kubernetes schaalt resources verticaal. Omdat noSQL dit ondersteund, en MySQL niet, is dat de beste keuze.

Er kunnen geen keuzes meer te maken op dit niveau. Alle volgende keuzes gaan over de implementatie van de code. De iteratie wordt afgesloten, en de volgende word gestart.

6.4 Tweede Iteratie: Globale Structuur

Deze iteratie ging over de Globale Structuur van de module. Er wordt gekeken naar welk ontwerp het beste past bij de module en hoe de verschillende componenten met elkaar zouden moeten communiceren.
Requirement O-3 stelt dat de module gehost moet worden in het cluster van IVIDO. De algemene consensus is dat microservices hier het beste voor zijn. Ze zijn makkelijk te schalen (O-4), licht (O-5), starten snel (O-6) en goed te onderhouden zijn(O-9).
Ik heb gekeken naar generieke architectuur patronen om inspiratie op te doen voor mogelijke ontwerpen. De ontwerpen zijn dus gebaseerd op traditionele architectuur patronen, maar terug geschaald naar een microservice ontwerp.

Monolith
De monolith is een enkel project, waarbinnen alles van de module wordt geregeld. Wanneer er een nieuw leefstijl doel wordt toegevoegd, wordt dit in het zelfde project geschreven.

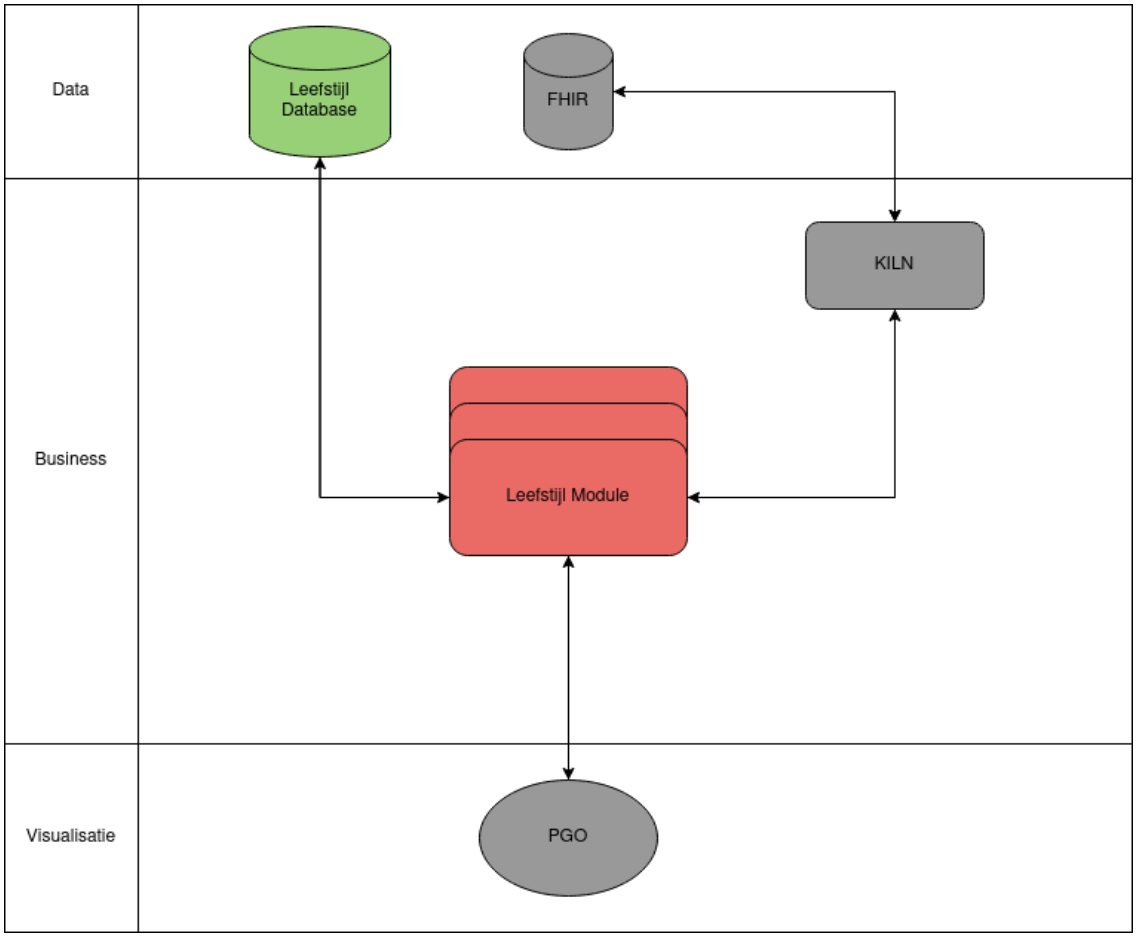
Nanoservice Manager
Het Nanoservice Manager ontwerp maakt gebruik van meerdere ‘Nano-service’ die aangestuurd worden voor een centrale Nanoservice Manager. Elk leefstijl doel zal een eigen ‘Nano-service’ hebben.

Meerdere Microservices
Voor dit ontwerp zou elk leefstijldoel een eigen microservice hebben. Maar in tegenstelling tot de Nanoservice Manager, worden deze services niet aangestuurd op een centraal punt.

Zie Tabel 12 voor een overzicht van hoe de ontwerpen wegen aan hun drivers.
Tabel 12: Besluit Globale Architectuur

ID	Eigenschap	Invloed	Prioriteit	Monolith	Manager	Micro
M-3	Extensibility	High	Constraint	3	4	5
O-3	Deployment	High	Constraint	5	3	2
O-4	Scalability	High	Must	2	4	5
O-5	Resource Manag.	Medium	Must	4	3	2
O-7	Security	Medium	Must	5	4	2
O-8	Testability	Medium	Must	5	3	3
O-9	Maintainability	High	Should	5	3	2

Monolith het makkelijkst te onderhouden, deployen, beveiligen en testen. Het lijkt allemaal terug te komen naar het KISS (Keep It Simpel Stupid) principe. De andere twee ontwerpen waren erg over-engineerd. De module zelf is niet zo ingewikkeld, dus hoeft ook niet heel ingewikkelde infrastructuur te hebben. Zie Figuur 4 voor schets van het Monolith Ontwerp.



Figuur 4: Architectuurschets Monolith

6.5 Derde Iteratie: Implementaties

In de derde iteratie bleken de meeste punten al belicht. Er is gekeken naar wat meer specifieke implementatie keuzes. De meeste van deze keuzes hadden weinig invloed op de architectuur. Onder deze keuzes vielen de volgende punten:

- Design Pattern voor het invoeren en opslaan van data
- Design Pattern voor het analyseren van de data
- Verantwoordlijkheden van front-end en back-end projecten.

In deze iteratie is er ook gekeken naar de databron van de voedingsdata. Er zijn verschillende bronnen die gebruikt kunnen op de voedingswaarden van de producten te bepalen. Uit eerdere gesprekken met Paul van der Boog en Hannie Piels is opgekomen dat de database van de Bonstat Applicatie te gebruiken was. Om de keuze beter te onderbouwen zijn er meerdere oplossingen geanalyseerd.

De data die gebruikt moest worden had een aantal eisen. Zo moest de bron van de data betrouwbaar zijn en bekend bij zorgprofessionals, het gaat immers om een medische behandeling. Verder is het ook belangrijk dat de gegevens verder uitgebreid kunnen worden om meer voedingsopties toe te voegen, en moeten de data gekoppeld zijn aan elkaar om gebruikers te helpen bij het invullen. Dit gaf de volgende opties:

NEVO Tabel
De Data uit de NEVO Tabel, wordt ieder jaar door het Rijksinstituut voor Volksgezondheid en Milieu de Nederlandse Voedingstoffenbestand gepubliceerd. Dit maakt het erg betrouwbaar en vertrouwd. Maar omdat de producten alleen sorteert zijn op hun productgroep is maar op weinig manieren te gebruiken.

Spoonacular
Spoonacular is een API voor voedingswaarden. Tegen betaling kan een grote selectie aan producten en voedingsdata worden opgehaald. Maar omdat gebruikers zelf data kunnen toevoegen is het niet zo betrouwbaar.

Bonstat Database
De data in de Bonstat Database heeft meer producten van de NEVO Tabel en is goed gekoppeld. Het gebruikt voedingswaarden uit de NEVO Tabel, wat het zeer betrouwbaar maakt. Tabel 13 is een overzicht van hoe de opties afwegen tegen de drivers.
Tabel 13: Beslissing Databron

ID	Eigenschap	Invloed	Prioriteit	NEVO	Bonstat	Spoon
A	Betrouwbaarheid	Low	Must	5	4	3
B	Uitbreidbaarheid	Medium	Could	2	3	1
C	Koppeling	High	Should	1	5	3
D	Bekend	Low	Should	5	4	2

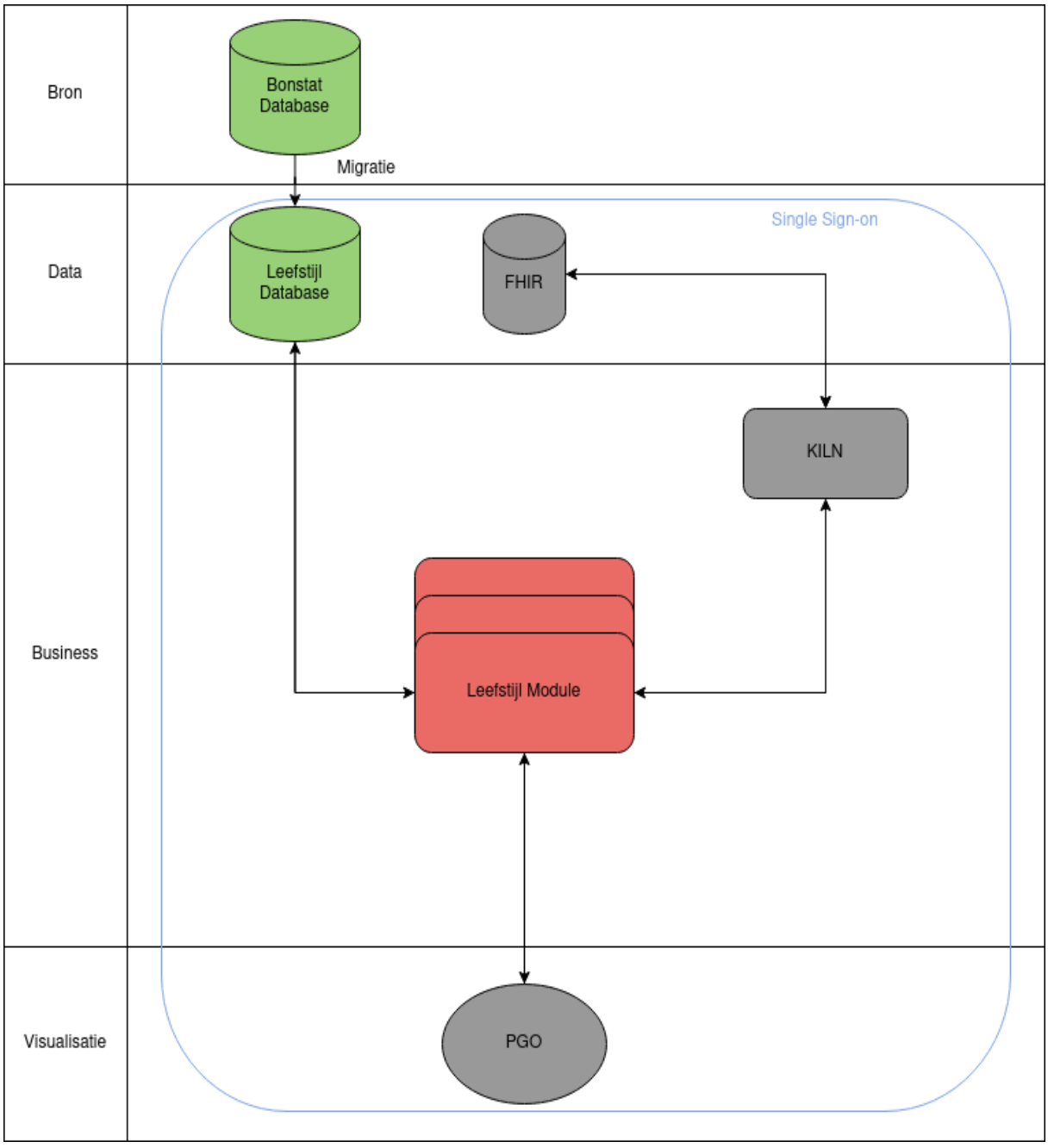
Omdat de Bonstat Database de data uit de NEVO Tabel gebruikt, en daarbij ook alle koppelingen heeft, is het de beste kandidaat. Wel betekent dit dat er een dus een database Migratie moet plaats vinden. Dit was niet deel van de originele planning. Maar de data uit de Bonstat Database is nodig om het beste product op te leveren. Ook zou de migratie helpen met de Maintainability van de module, omdat de data er al in zit. Dit maakt de tijdsinvestering waard.

6.6 Uiteindelijk Ontwerp

Alle tools en Implementaties zijn besloten, en alle Niet Functionele Requirements zijn voorzien. Hierbij wordt het ADD afgesloten Tabel 14 geeft een overzicht van alle gemaakte keuzes binnen het ontwerp process. Figuur 5 toont het uiteindelijke architectuur ontwerp.
Tabel 14: Toelichting gemaakte implementatie keuzes

1: Basis		
Aspect	**Keuze**	**Reden**
Backend Taal	Node.js	Licht, snel en schaalt goed
Database Implementatie	Mix van eigen database en FHIR	Voordelen van eigen vrijheid van inrichten en toegang tot medische gegevens
Database Structuur	noSQL	Is makkelijk aan te passen en schaalt goed.

2: Structuur		
Structuur	Monolith	Goed te onderhouden en te integreren in de PGO
Integratie	Single Sign-on	Uiteindelijke implementatie
3: Implementatie		
Invoer	Builder Pattern	Makkelijk te implementeren en uit te breiden
Overzicht	Translator	Makkelijk uit te onderhouden
Databron	Bonstat Data	Heeft alle gewenste functionaliteiten
Verantwoordelijkheden	Lazy Front-end	Makkelijk te testen en te onderhouden.



Figuur 5: Uiteindelijk Architectuur Ontwerp

6.7 Proof of Concept

Met het uiteindelijke Architectuur ontwerp is een Proof of Concept op gesteld. Door een simpele versie van het eindproduct te ontwikkelen wilde ik de volgende doelen bereiken:

Testen of de tools goed samenwerken
Veel van de gemaakte keuzes zijn gemaakt op theoretische argumenten. Om te verifiëren of deze kloppen worden ze praktijk getest.

Leren van tools
Ik had weinig ervaring met React, en haast geen met Node.js. Door het Proof of Concept te maken kan ik de talen leren, en heb ik een goed basis voordat ik begin aan de module.

Feedback krijgen van stakeholders
Door een simpele versie te maken van de module kan ik controleren of ik requirements van de stakeholders goed heb begrepen.

Het Proof of Concept was alleen bedoeld als een schets. Zodra ik de doelen heb bereikt begin ik met het ontwikkelen van het eindproduct in een nieuw project. Dit geeft mij de ruimte om fouten te maken en verschillende dingen uit te proberen.

Ontwikkelen van Proof of Concept
Het Proof of Concept bestond uit twee projecten. Een front-end project geschreven in React, en een back-end project geschreven in Node.js. Deze twee projecten communiceerden met een REST-API en was aangesloten op een MongoDB database. Er was een invul pagina om voeding in te vullen, en een overzicht pagina waar het dieet kon worden ingezien.

De tools bleken erg goed met elkaar samen te kunnen werken. De routing van Next.js helpt veel bij het ontwikkelen een React App. De functionaliteiten van Mongoose helpen bij de koppeling tussen Node.js en een MongoDB. De dynamische front-end van React sluit goed aan op de het asynchrone Node.js. Dit alles gaf mij veel vertrouwen in de gekozen tools.

Feedback op Proof of Concept
Toen het Proof of Concept af was heb ik een afspraak gemaakt met stakeholder Hannie Piels voor Feedback. Zij had een duidelijke visie van hoe de module moest werken. Door het Proof of Concept aan haar te laten zien, kon ik de beste feedback krijgen op het ontwerp.

Bij de demonstratie van de Proof of Concept reageerde Hannie verbaasd. Volgens haar had ik het ontwerp verkeerd begrepen. Zij vertelden dat de overzichten moesten bestaan uit de producten en niet uit de voedingswaarden. Ook was klopte de manier van voeding selecteren niet met de manier die zij had voorgesteld. Deze informatie was verhelderend, want ik kon haar beschrijving nu relateren aan het ontwerp van het Proof of Concept.
In dit gesprek vertelde zo ook dat bij de vorige poging van het implementeren van de module functionele ontwerpen waren gemaakt. Het bleek dat ik deze documenten niet heb gekregen, toen ik vroeg om de documentatie van de vorige poging. Het was tegelijkertijd erg frustreren en opluchtend om dit te leren. Aan de ene kant was het frustrerend omdat de ontwerpen mij hadden kunnen helpen bij het opstellen van de requirements. Aan de andere kant was het een opluchting omdat het ontwerp nu veel duidelijker werd.

Met deze ontwerpen heb ik drie nieuwe requirements kunnen opstellen. Zie Tabel 15

Tabel 15: Nieuwe Requirements aan de hand van ontwerpen

ID	Beschrijving	P	SG	RT	RE	SP	AP	DB	SB	V
P-3	De module moet producten aanraden gebaseerd op eerder ingevulde producten	C	P	F	-	H	H	RI-14	HP	Test
P-4	De behandelingdeelnemer moet producten met verschillende niveaus van granulariteit kunnen toevoegen aan het dieet	C	P	F	-	H	M	RI-14	HP	Test
Z-4	De behandelingdeelnemer moet het overzicht sorteren op producten, productgroepen en voedingswaarden.	C	Z	F	-	M	L	RI-14	HP	Test

Hoewel reactie van Hannie op het Proof of Concept was redelijk kritisch was, zag ik deze stap zelf als een groot succes. Alle drie de doelen van deze stap zijn gehaald. De tools bleken erg goed samen te werken, ik heb veel geleerd over de nieuwe programmeertalen en heb de eindelijk de goede ontwerpen van de module gekregen.

6.8 Impact op Planning

Het ontwerpen van de architectuur en het maken van het Proof of Concept verliepen zonder al te veel problemen. Het is afgemaakt binnen de geplande twee weken.
Tijdens het ontwerp heb ik wel besloten dat er een Database Migratie uitgevoerd moet worden. Dit was niet deel van de originele planning, dus moet er geschoven worden. Ik verwachtte dat de migratie twee weken zou duren, gebaseerd op ervaring. Een van deze weken komt dan uit de buffer. Zie Tabel 16 en Tabel 17 voor de aangepaste planning.

Tabel 16 Fase van afstuderen inclusief Database Migratie

Fase	Resultaat	Duur
Aanpak en Onderzoek	Plan van Aanpak	1 Week
Requirements	Requirements Rapport	4 Weken (2)
Architecture	Architecture Design en Proof of Concept	2 Weken
Database Migratie	Migratie Plan	2 Weken (0)
Ontwikkelen & Testen	MVP van Opdracht en Testplan	7 Weken (10)
Oplevering	Afstudeerverslag	1 Week (2)
Originele Planning	-	-

Tabel 17 Aangepaste planning inclusief Database Migratie

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

De migratie bleek meer tijd te kosten van verwacht. De database was erg verwarrend ontworpen waardoor het moeilijk was om de correcte data te kunnen selecteren. Omdat ik hier te weinig tijd heb had ingepland had ik veel minder tijd over voor het ontwikkelen van de module.

7 Database Migration

De data uit de Bonstat Database heeft erg veel waarden voor de module. Handmatig zijn er productgroepen opgesteld, productgroepen gesorteerd op moment van de dag en hebben producten duidelijke eenheden. De stakeholders zien graag deze functionaliteiten terug in de module. Veel van deze data is met de hand ingevoerd, en is niet makkelijk te verkrijgen op een andere manier. Omdat ik in 6.5 heb besloten om een noSQL database te gebruiken, en de Bonstat Database een MySQL database is, moet er een migratie uitgevoerd worden.

Het doel van de migratie is om de meest waardevolle data uit de database te selecteren, en deze over te zetten naar de database van de module. Deze data kan ik dan gebruiken om het invullen van voeding veel makkelijker te maken voor gebruikers. De database heeft ook data over features die niet deel zijn van de MVP, maar later wel toegevoegd kunnen worden aan de module. Als ik deze mee neem in de migratie, hoeft dit later niet meer te gebeuren.

Een uitgebreid verslag van de migratie is te vinden bijlagen E.

7.1 Bestuderen Bonstat Database

Ik heb de data gekregen van Hannie Piels. Zij heeft een dump gemaakt van alle data uit de MySQL database, op privacy gevoelige persoonsdata na. Deze data heb ik ingeladen in een het database management tool DataGrip, zodat ik het kon bestuderen.

De eerste indruk was dat de database zeer groot was. In totaal waren 174 tables en bijna een miljoen (983957) rows in de database. Een klein deel hiervan, 22 tables, was relevant voor de migratie. Deze tabels ging over de productgroepen, eenheden en voedingswaarden en kon ik gebruiken in de module.

Daarna viel het me op dat de kwaliteit van de database erg slecht was. Hierdoor duurde het lang voordat ik begreep hoe de database werkte was het erg moeilijk de goede data uit de database te halen. Ik zal een aantal fouten toelichten:

Heel veel dubbele waarden
In de product tabel stonden voedingsmiddelen die een gebruiker kon kiezen om in te vullen. Maar 48% van alle rows waren unieke producten. Vaak was de dubbele entry ‘expired’, wat onlogisch voor dit soort entiteiten, Deze dubbele data zorgt er voor dat queries lager duren en dat er een slechter overzicht is van de data in de tabel.

Verwarrende Data Opslag
De nutrition table was erg moeilijk te lezen. Het verwijst naar zich zelf, en maakt zo een hiërarchie en werden specifieke voedingswaarden onder generieke gekoppeld. Maar deze groepering leek niet altijd te kloppen omdat de totalen anders niet klopten.

Slechte normalisering
Veel tables zijn slecht genormaliseerd. De product table heeft de velden ‘name’ en ‘newName’. Het verschil tussen de twee was niet duidelijk. Het is beter maar een van de twee te gebruiken. Dit maakt de tabel meer overzichtelijk.

Logging
Bijna alle tables hebben ingebouwde logging. Velden zoals CreatedBy and ModifiedData beschrijven wat er met de data is gebeurd. Maar door dit in de tabel zelf te plaatsen wordt de tabel minder overzichtelijk. Ook maakt dit dat joins van deze tabels meer geheugen kosten. Het is slimmer om al deze logging in een aparte table te houden.

Dit alles maakte dat het moeilijk om de correcte data eruit te krijgen. De nutrition tabel in bijzonder was zo slecht ingericht dat ik de data ervan niet vertrouwde. Nu was het zo dat een deel van de voedingsdata van Bonstat afkomstig is uit de NEVO Tabel. Daarom heb besloten om voedingswaarden waarden uit de NEVO Tabel te gebruiken in plaats van die uit de Bonstat Database. De volgende data heb ik wel gemigreerd:

- Producten die ook te vinden waren in de NEVO Tabel
- Eenheden van de geselecteerde producten
- Productgroepen van de geselecteerde producten

Omdat het bestuderen van de data zo lang duurde heb ik besloten om alleen de data die nodig was voor de MVP te migreren. De Bonstat Database heeft ook data voor mogelijke toekomstige features, zoals recepten, maar deze zijn is dus niet gemigreerd.

7.2 Nieuwe Database

De module heeft een eigen database nodig om de gebruikersgegevens en de voedingsdata op te slaan. Tijdens de Architecture Design (6.3) heb ik besloten om een noSQL database te gebruiken.

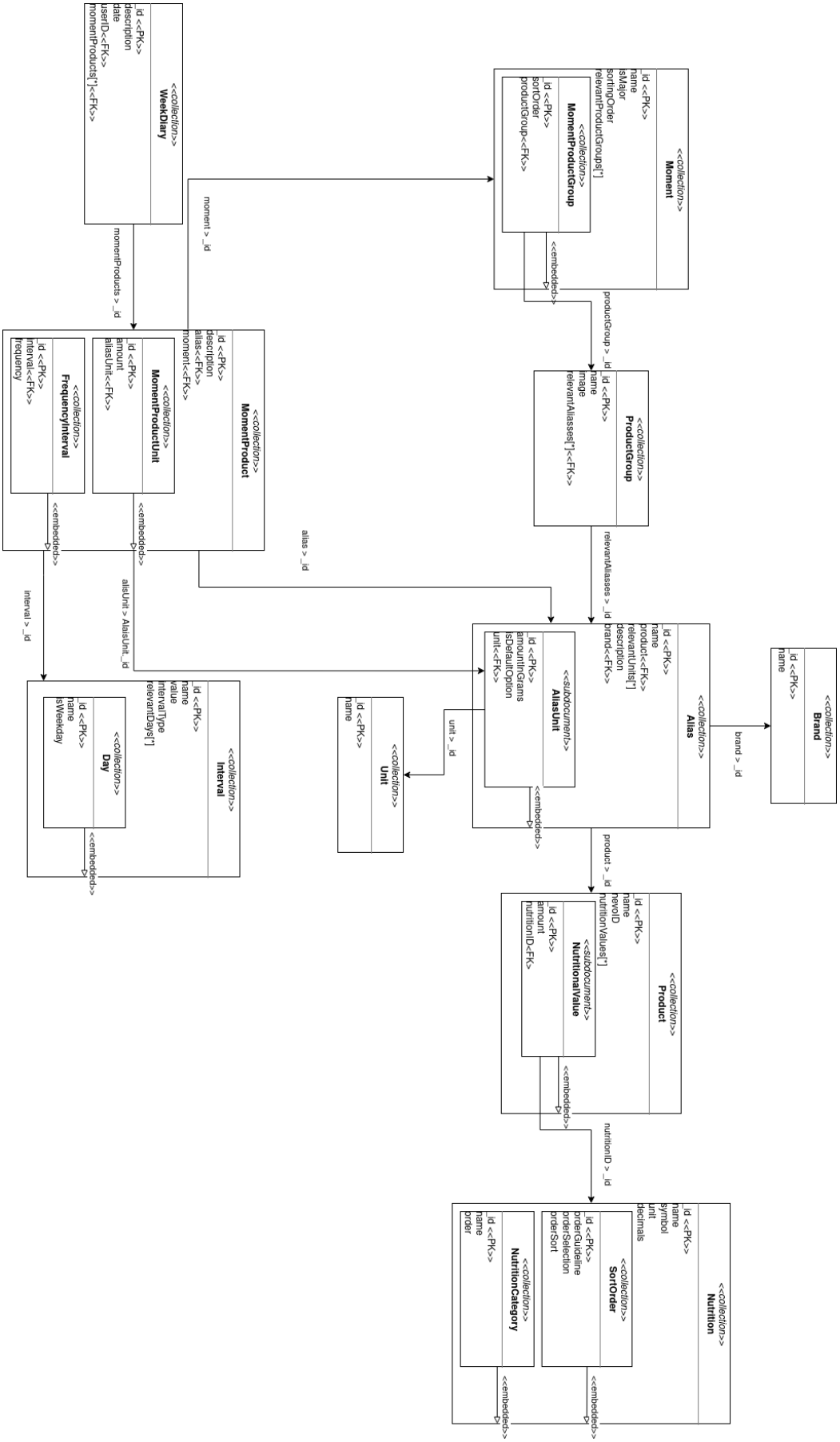
Voor het ontwerp van de nieuwe database heb ik een aantal richtlijnen opgesteld. Omdat noSQL anders werkt dan SQL kan ik niet zomaar het oude ontwerp overnemen. Door het opstellen van de richtlijnen zorgde ik er voor dat het ontwerp zo consistent mogelijk bleef.

Herstel Fouten in Origineel Ontwerp
Door de fouten in het originele ontwerp te herstellen in het nieuwe ontwerp maak ik de module meer robuust. Zo moet er een nieuwe ontwerp komen voor de opslag van voedingswaarden en moet de normalisering verbeterd worden.

Neem Functionaliteit mee in Ontwerp
Door bij het ontwerpen van de database structuur ook te denken aan de functionaliteiten kan ik de database verstandig ontwerpen. Door collections die veel data met elkaar moeten delen goed te koppelen, kan ik het query’en op de database versimpelen.

Gebruik subdocumenten voor One-To-One en One-To-Many verbindingen
noSQL maakt het mogelijk om subdocumenten en arrays op te stellen als data-type voor velden. Dit kan ik gebruiken voor de Foreign Keys en Associative Classes.

Breid de Database uit met Functionaliteiten van de Module
Naast de voedingsdata, moet er ook andere data worden opgeslagen in de database. Zo moeten er ook ontwerpen komen voor de diëten en de hoeveelheden van de gegeten voedingswaarden. Het nieuwe ontwerp is te zien in Figuur 6.



Figuur 6: Database Ontwerp van de Module

De database structuur van de module heeft veel voordelen op die van de Bonstat Database.

Aliassen
Producten worden los opgeslagen van hun Alias. Ik merkte dat producten in de Bonstat Database vaak een andere naam hadden, maar verwezen naar het zelfde voedingswaarden in de NEVO Tabel. Om dit te ondersteunen worden deze nu los opgeslagen. Zo er makkelijk een nieuwe Alias worden toevoegt en hoeven de voedingswaarden maar op een plek worden aangepast.

Opslag van Nutrition
Ik heb het opslaan van voedingswaarden aangepast in het nieuwe ontwerp. Producten hebben in feite een Many-to-Many relatie met de Nutrition Collection. Als Associative Class is er een NutritionalValue Subdocument toegevoegd met een Amount field. Deze geeft aan hoeveel van de voedingswaarden erin zit.

Ontwerp naar Functionaliteiten
Volgens het ontwerp zoekt een gebruiker van Momenten, naar Productgroepen, naar Producten naar Eenheden. In het ontwerp zijn al deze stappen zeer eenvoudig te maken met de resultaten van de vorige query.

7.3 Database Migratie

Met ontwerp van de nieuwe database kon ik de slag met het opstellen van het migratie plan. De migratie had een aantal stappen waar ik extra rekening mee moest houden bij het opstellen van het plan.

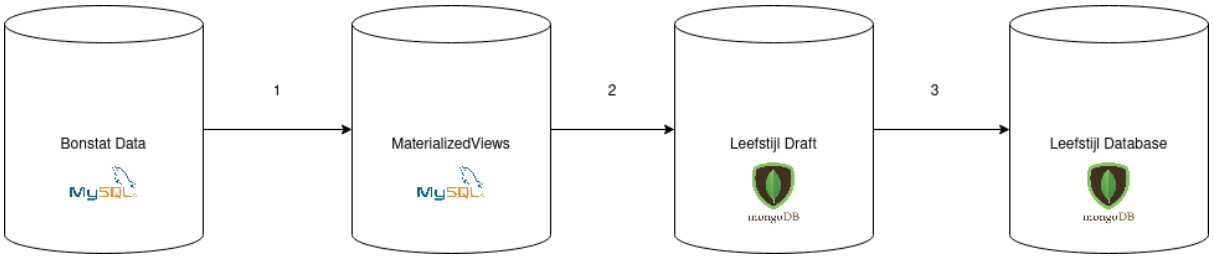
Data Cleaning
De data in de Bonstat Database moet eerst goed gefilterd worden voordat ik het kan gebruiken. Ik kon alleen producten gebruiken die ik ook in de NEVO Tabel kon vinden.

SQL naar noSQL
De results van de queries uit de SQL database moeten eerst omgezet worden naar de nieuwe de nieuwe noSQL vorm voordat ze opgeslagen kunnen worden.

Nieuwe Primary Keys
Ik vind het netter om met ObjectIds te werken al Primary Keys, dan de integers van de Bonstat Database. Dit maakt de migratie wat ingewikkelder omdat de Foreign Keys van alle entiteiten ook aangepast moeten worden.

NEVO Tabel
De NEVO Tabel moet ook ingeladen worden en aan de correcte producten gelinkt worden. Om de migratie te ondersteunen heb ik gebruik gemaakt van een Node.js applicatie. Deze applicatie voert de queries, maakt de translaties naar de nieuwe modellen slaat de gegevens op in de nieuwe database. Omdat ik ook Node.js gebruik voor de back-end van de module weet ik dit makkelijk op te zetten, en kan ik de zelfde tools gebruiken als daar.

Aan de hand met deze aandachtspunten heb ik het Migratie Plan op gesteld. Zie Figuur 7.



Figuur 7: Diagram van het Migratie Plan

Het migratie plan bestaat uit vier stappen en vier database. De aanpak is wat omslachtig, maar ik hier gekozen omdat de migratie wat ingewikkelder was dan normaal. Door het process in verschillende stappen op te delen wilde de migratie makkelijker is uit te voeren, en te herstellen wanneer er een fout wordt gemaakt. Het werkt als volgt:

Stap 1: Data Cleaning
De eerste stap is het filteren van de data in de Bonstat Database. Hier worden de relevante data uit de Bonstat Database geselecteerd en mogelijke foute gegevens gecorrigeerd.

Stap 2: Materialized View Database
Subselecties van relevante data uit de Bonstat Database worden opgeslagen in ‘tussen’ database. Deze ‘Materialized View’ Database helpt bij het maken van queries.

Stap 3: Leefstijl Draft Database
De relevante data uit de Bonstat Database wordt met hulp van de Materialized View Database opgehaald en opgeslagen in een Draft Database. De data uit de NEVO Tabel wordt opgehaald en aangesloten op de relevante gegevens. Door de data eerst op te slaan in deze Draft database kan ik nog wat aanpassingen maken voordat ik het finaliseer.

Stap 4: Definitieve Versie
Ondersteunende velden uit de Draft Database worden verwijderd en wordt opgeslagen in de definitieve database. Als laatste worden de niet voeding gerelateerde data ingeladen. Dit zijn bijvoorbeeld de verschillende frequenties voor voeding en de dagen van de week.

Het uitvoeren van het Migratie ging goed. Het plan om de migratie in stappen uit te voeren was een slimme keus. Dit gaf me de ruimte om de grote taak in stappen op te delen. Ik heb ik een aantal keer een stap terug moeten doen toen ik er achter kwam dat ik een fout had gemaakt. Dit was makkelijk te doen omdat de stappen in de migratie klein waren en ik alle queries had opgeslagen.
Ik ben blij dat ik dit deze methode heb gekozen voor de migratie want het werkt erg goed en maakte het uitvoeren erg makkelijk. Het resultaat was goed, en heb geen aanpassingen hoeven te maken aan het originele ontwerp.

7.4 Migratie Verslag

Het uitvoeren van de migratie duurde veel langer dan verwacht. Maar ik vond het wel heel erg belangrijk dat de migratie gebeurde. De verbinden tussen de producten voegen heel veel waarde toe aan het product. Om de tijdsinvestering die het heeft gekost te verantwoorden is er een Migratie Verslag opgesteld.
In het verslag ga ik dieper in op de fouten van de Bonstat Database en licht ik mijn keuzes en de stappen in het Migratie uitgebreid toe. Het verslag is te vinden als bijlagen E.

7.5 Invloed op Planning

Zelfs met de extra geplande tijd liep de database migratie uit op de planning. Ik heb de planning weer moeten aanpassen. Het zou krap worden omdat er nog een groot deel van de code geschreven moest worden, en de integratie in de PGO nog uitgevoerd moest worden. Beide zijn grote stappen die veel tijd kosten. Door te focussen op een simpel ontwerp, en een minder uitgebreid Test Plan, hoopte ik de opdracht af te krijgen voor de deadline. Tabel 18 geeft een overzicht van de afgelopen en resterende fases van het afstuderen en Tabel 19 toont de aangepaste planning.

Tabel 18 Fases van afstuderen na uitvoeren van migratie

Fase	Resultaat	Duur
Aanpak en Onderzoek	Plan van Aanpak	1 Week
Requirements	Requirements Rapport	4 Weken (2)
Architecture	Architecture Design en Proof of Concept	2 Weken
Database Migratie	Migratie Plan	4 Weken (0)
Ontwikkelen en Testen	MVP van Opdracht	5 Weken (10)
Oplevering	Afstudeerverslag	1 Week (2)
Originele Planning	Twee weken uitloop van requirements	-

Tabel 19 Aangepaste planning uitvoeren van migratie

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

De ontwikkeling zelf ging me goed af. De ervaring van het programmeren hielp om al snel wat in elkaar te kunnen zetten. Tijdens het ontwikkelen bleek dat de integratie niet uitgevoerd kon worden. De gaf wat meer ruimte in de tijd, waardoor ik aandacht heb kunnen geven aan het schrijven van de testen.
Het werk bij het prioriteren van de requirements, maken van duidelijk Architecture Design en de bedachte database ontwerp zorgde ervoor dat ik een duidelijk plan had, en dat ik alleen nog maar de puzzelstukjes op zijn plaats hoefde te leggen.

8 Ontwikkeling

Na al het voorwerk kon ik eindelijk beginnen aan mijn favoriete onderdeel. Al het werk tot nu toe was voorbereiding voor het programmeren. Door al deze voorbereiding wist ik precies, wat ik moest doen en hoe ik het ging uitvoeren. Ik had drie doelen opgesteld:

Implementeren van overzicht en invullen
De twee belangrijkste features van de module zijn het invullen en het overzicht van de diëten. Nu ik de correcte ontwerpen, goede data heb moet dit goed lukken.

Integreren in de PGO
De front-end van de module moet gebouwd worden in het PGO Project. Dit moet samenwerken met mijn eigen back-end en gehost worden in het cluster van IVIDO. Zo kan het samenwerken met de rest van de services van IVIDO, zoals de hun database.

Testen schrijven
Met het schrijven van code, komt ook het schrijven van testen. Ik had al een test suite opgesteld, en enkele testen geschreven met de tools. Maar deze zijn al wat verouderd omdat de data al is veranderd. De tools kan ik meenemen, maar de testen zelf niet meer.

Doordat er niet zo veel meer tijd over was, moest ik goed gefocust blijven op de MVP. Volgens de planning had ik nog een week over van de huidige sprint en twee hele sprints daarna. Deze zou ik als volgt besteden:

- In de resterende week zou ik de back-end updaten met de nieuwe data
- De volgende sprint zou ik focussen op de features en de integratie
- De laatste sprint kon ik gebruiken voor testen schrijven en refactoring.

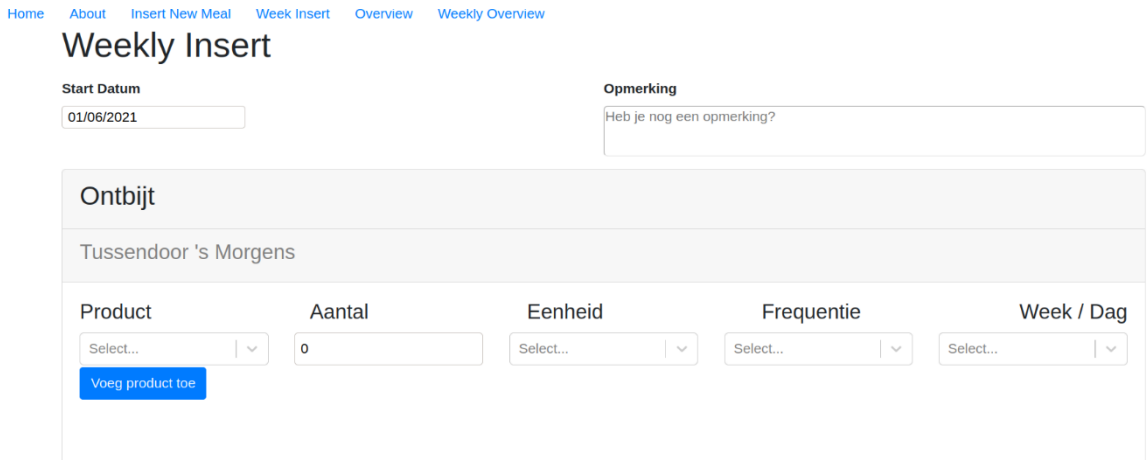
Een overzicht van de front-end en de functionaliteiten is te vinden in het Module Overzicht. Deze is te vinden in bijlage E.

8.1 Eerste Iteratie: Integreren met nieuwe database

De eerste stap die ik heb ondernomen is het koppelen van de nieuwe database aan het project. De datamodellen waren anders dan dat ik had voorspeld. Hierdoor moest ik dingen aanpassen zoals de controllers en de database querying. Ik kreeg dit eerder af dan ik had gepland omdat ik onderhand al veel ervaring had opgebouwd met de tools die ik gebruikte.

Ik kreeg de vraag van Paul en Hannie of zij een demo kon krijgen om mijn voortgang te zien. Sinds de laatste demonstratie aan Paul en Hannie heb ik vooral gewerkt aan de database migratie. Dit betekend dat ik niet heel veel nieuwe te laten zien had. Ik zag dit wel als een moment om mijn begrip van de nieuwe ontwerpen te testen. Ik heb de front-end van de Proof of Concept overgenomen en wat kleine aanpassingen gemaakt volgens de ontwerpen.

Dit was nog niet de front-end van de PGO, want dit zou te veel werk zijn. Ik wilde eerst de back-end in orde krijgen voordat ik met de integratie zou beginnen. Ook was ik bekend met de oudere front-end, waardoor de aanpassingen makkelijker gemaakt waren. Zie Figuur 8 voor de aangepaste versie.



Figuur 8: Eerste versie van het invul scherm

Paul en Hannie waren niet onder in de indruk tijdens de demonstratie. Zij zagen weinig voortang in het project. De ontwerpen waren dichter bij wat zij wilde hebben, maar er waren nog geen nieuwe features bijgebouwd. Ik vertelde dat er veel voortgang is geweest in het project, maar dan vooral in aan achter de schermen. Hun antwoord hierop was “Ja, maar dat kunnen wij niet zien”. Voor hun was er te weinig gebeurd om constructieve feedback te geven op de aanpassingen. Achteraf gezien had ik de demo beter kunnen voorbereiden om uit te leggen wat ik aan de back-end had gedaan. Ik heb wel veel feedback kunnen krijgen op mijn aanpassingen in de front-end. Ik leek de ontwerpen eindelijk goed te begrijpen en kreeg ook vragen over wat punten die nog onduidelijk waren.

Aan het eind van de demo hebben we samen besloten pas rond het einde van de het traject weer samen te komen. Dit gaf mij meer dan genoeg tijd om aan de module te werken, tot een staat waar ik tevreden over was. Omdat ik de ontwerpen goed leek te begrijpen kon dit. Dan in de laatste feedback ronde konden ze kleine punten van feedback geven die ik voor de deadline nog kon afronden. Ik had ze verteld dat ik alleen een MVP zou opleveren met een invul pagina en een overzicht pagina. Omdat ik al een paar keer dingen had beloofd die ik niet kon waarmaken wilde ik nu duidelijk maken wat ze konden verachten zodat ik een eindproduct kon opleveren waar zijn tevreden mee waren.

Dit vond ik erg fijn want dit gaf mij de ruimte om lekker door te werken. Ik merkte dat ik erg afgeleid werd van dit soort demo’s en dat mijn focus telkens moest draaien. Nu had een paar weken de tijd om lekker door te werken en echt voortang te maken.

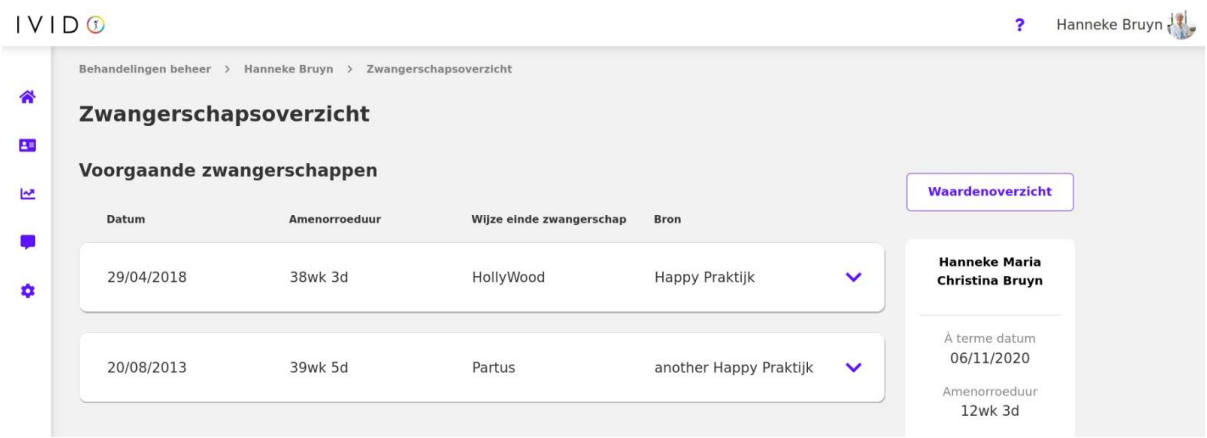
Aan het eind van de week liep de eerste halve sprint af. Ik had mijn geplande taken goed kunnen afronden, en was tevreden over mijn ontwerp. Ik heb ik kunnen delen met mijn collega’s die ook onder de indruk waren. Ik vertelde dat ik in de volgende sprint mij wilde focussen op de integratie en de features, en heb toen een afspraak kunnen maken met front-end Lead binnen IVIDO over de integratie. Ik heb mijn volgende sprint gepland volgens het Jira board. Dit koste veel werk want ik moest zelf overal kaartjes voor aanmaken en prioriteren, maar het gaf me wel een goed overzicht.

8.2 Tweede Iteratie: Integreren in de PGO

Deze sprint begin ik met een gesprek met de front-end Lead Developer binnen IVIDO. Ik legde uit wat ik wilde doen. De front-end moest overgezet worden naar het PGO project, en ik wilde de back-end hosten in het cluster. Ik kreeg van hem wat tips en pointers, en hij vertelde mij waar mijn code zou komen te staan in het project. Hij was nog niet zo zeker over hoe de integratie moest gaan, dat zou hij nog navragen. In de tussentijd kon ik aan de slag.

De code kwam in het nieuwe front-end project van de PGO. Dit was geschreven in React, dit zou de oude PHP front-end vervangen. Er was nog maar één andere pagina te vinden met wat knoppen, maar er was al wel een duidelijke structuur aangelegd. In het project was tooling opgezet zodat het ontwikkelen ging volgens de standaarden van IVIDO. Daarnaast was er ook een IVIDO-Component-Library. Een library met blokken waarmee ik de front-end kon bouwen. Deze makkelijk te gebruiken, en zorgde dat alles er mooi en het zelfde uitzag. Deze library was nog niet compleet, zo meeste er bijvoorbeeld een number-input-field. Toen ik dat aangaf werd deze snel aangemaakt en kon ik heb de volgende dag al gebruiken.

Ik heb de andere pagina binnen het project gebruikt als voorbeeld voor mijn pagina. Deze andere pagina maakte gebruik van een lokale statische data bestand voor het ophalen van gegevens. Dit leek me wel makkelijk, want dan hoefde ik nog niet na te denken over de verbinding met de back-end. Omdat de database modellen al bekend waren heb ik mijn statische data gemodelleerd naar de modellen, om de uiteindelijke transitie naar de back-end connectie makkelijk te laten verlopen. Figuur 9 is de pagina in kwestie.



Figuur 9: Pagina in de PGO die is gebruikt voor inspiratie.

Het werk aan ontwerp schoot erg op, en al snel had ik een ontwerp waar ik blij mee was. Ik heb contact opgenomen met de Lead Front-end Developer om te praten over de integratie van de back-end.

In dit gesprek vertelde hij mij dat de integratie met de back-end van het nieuwe project pas eind januari plaats zou kunnen vinden. Om er voor te zorgen dat alle elementen goed en veilig met elkaar samenwerken waren ze bezig met het maken van een authentication service. Deze service moet alle calls controleren op veiligheid en maakt het mogelijk om de data uit de database overal te gebruiken. Deze service zou pas eind januari klaar zijn, dat is na de deadline van het inleveren van het afstudeerverslag.

Dit was weer opnieuw een grote tegenvaller. Zonder deze service kon ik niet mijn module integreren in hun back-end. Dit betekend dat ik ook geen werkende versie kon deployen die mensen konden testen. Omdat veel van het ontwikkelen zelfstandig heb gedaan, heb ik gemist dat het ontwikkelen van de service liep uit. Snel na het horen van dit nieuws heb ik contact opgenomen met de opdrachtgever, om te kijken hoe ik het afstuderen goed zou kunnen afronden.

Ik stelde voor om 2 versies te maken van de uiteindelijke module. De eerste versie zou gebruik maken van een dummy data bestand, net zoals de pagina die al in het project stond. Deze versie kon dan gedeployd worden en kon door gebruikers getest worden. Er kon dan niets worden opgeslagen, en zou dienen als een Proof of Concept. Een tweede versie zou gebruik maken van een lokale back-end en de nieuwe database met de gemigreerde data. Deze tweede versie zou dienen als MVP. Het toont aan dat de verbindingen tussen de back-end, front-end en database correct samenwerken en heeft de twee belangrijkste features in het opslaan en inzien van de voeding. Het zou identiek zijn aan het oorspronkelijk verwachte MVP, behalve dat het lokaal wordt gehost in plaats van op het cluster.

De oplossing om twee versies op te leveren had twee grote voordelen. Als eerste zouden de functionaliteiten van de module nog steeds ontwikkeld worden. De dummy versie kan gebruikt worden bij demonstraties, en wanneer de authentication service af zou zijn, zou het snel geïntegreerd kunnen worden in de PGO. Het tweede voordeel is dat dit veel tijd scheelt. Het integreren is redelijk ingewikkeld, zeker als je er onbekend mee bent. Door de ingewikkelde stappen van het hosten van de back-end en database over te slaan, blijft er meer tijd over voor het project zelf. Het was eerst de bedoeling dat ik dit zelf zou doen, maar omdat het niet meer door ging kon een kwalitatief beter product opleveren.

De opdrachtgever was overtuigd van het plan en ging akkoord.

8.3 Derde Iteratie: Lokale Versie

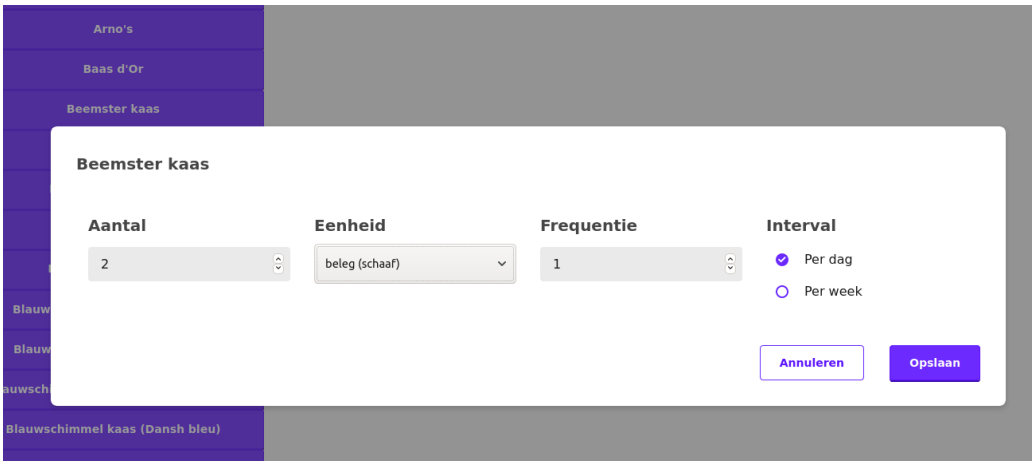
Omdat er een nieuwe plan was, moest ik ook een nieuwe planning maken. Het schrappen van de integratie leverde mij veel tijd op. Ik deze tijd wil ik graag gebruiken om de kwaliteit van de module te verbeteren en uitgebreide testen te schrijven voor de code. Beter code zorgt ervoor dat de module beter te onderhouden is. Door testen te schrijven kunnen bugs worden gevonden en opgelost, en kan de kwaliteit van de code worden aangetoond.

Omdat mijn opdracht erg los stond van waar de andere developers mee bezig waren, merkte ik dat andere developers mij niet zo goed kon helpen. Ik kreeg weinig input bij de daily standups, en kon zelf weinig toevoegen bij de sprint reviews. Om tijd die ik nog over had zo efficiënt mogelijk te gebruiken, heb ik besloten om uit het sprint team te stappen. Omdat ik niet meer in het sprint team zat, hoefde ik ook niet meer gebruik te maken van Jira. Jira mij veel overzicht gaf, vond ik het wel veel tijd kosten om het in mijn eentje bij te houden. Het Trello board, dat ik eerder heb gebruikt in het project, vond ik een veel efficiëntere manier van werken.

Al deze dingen gaven mij de ruimte om het project te ontwikkelen zoals ik zelf het fijnst vond. Ik heb toen gekozen om mini ‘sprints’ te maken van vier dagen. Om de laatste periode zo efficiënt mogelijk te maken, vond ik het handiger om de
sprints veel korter te maken. Een periode van vier dagen gaf mij genoeg tijd om een aspect goed op te pakken en uit te voeren, en daarna te kijken wat er nog moest gebeurden. Zo zat ik maar vier dagen ‘vast’ voordat ik van koers kon veranderen.

De focus van het ontwikkelen lag op de twee belangrijkste features. Het invullen van een dieet, en het inzien van het overzicht. Deze twee features wilde ik zo eenvoudig mogelijk maken zodat ik de kwaliteit hoog kon houden.

Front-end
De front-end was al in een gevorderd stadium, zie Figuur 10. Deze heb ik in de vorige iteratie kunnen maken met statisch data, Ik hoefde deze alleen nog maar te linken aan mijn eigen back-end. Ik had al alle data van het in de database staan, dus de ontwerpen waren makkelijk aan te passen.

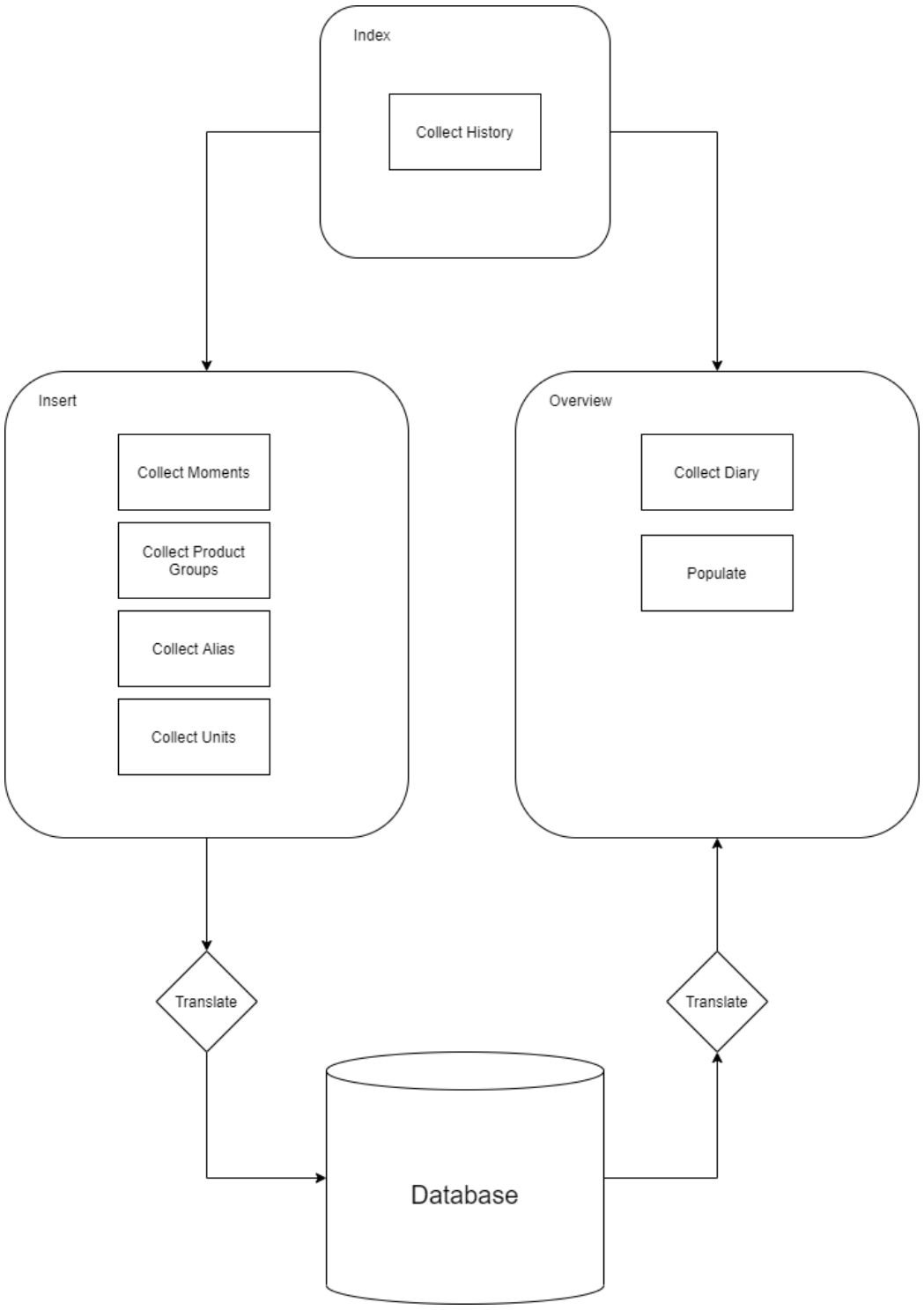


Figuur 10: Uiteindelijk Invulscherm

Voor de derde keer moest ik het invullen van het dieet programmeren, maar deze keer had ik het duidelijkste ontwerp. Ik had het ontwerp hoe het eruit moest doen, de vorm waarin de data werd opgeslagen en de ervaring van de vorige twee keren. Dit zorgde er voor dat ik een slimme manier had gevonden om het dieet in te vullen. Het invullen, verzamelen en het insturen van de voedingsdata werkte erg goed. Ik hoefde de state maar op een plek op te slaan, en door gebruik te maken van een translator, kon de front-end eenvoudig blijven.

Het overzicht van de voeding kreeg wat extra updates om beter te passen bij de ontwerpen. In het ontwerp waren er drie verschillende overzichten. Een van producten, een van productgroepen en een van voedingswaarden. Ik had nog geen logica gemaakt voor het gebruiken van de voedingswaarden, en was dat ook net van plan om ontwerp simpel te houden. In plaats daarvan heb ik de het overzicht van de producten en de productgroepen samengevoegd.

Het ontwerp gaf ook aan dat er index pagina waarin de gebruiker een overzicht kreeg van alle ingevulde diëten. Dit was makkelijk genoeg de te maken. Figuur 1Figuur 11 geeft een overzicht van het functionele ontwerp van de module.

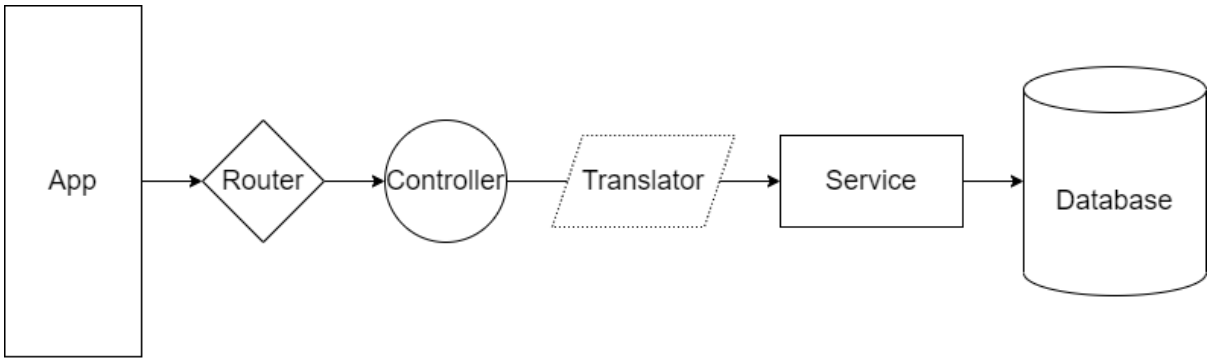


Figuur 11: Functioneel Ontwerp van de module

Back-end

Het ontwikkelen van de back-end verliep ook voorspoedig. De functionaliteiten gingen vooral over het ophalen, opslaan en verwerken van data. Veel van de acties leken op elkaar, dit leidde mij tot het maken van een vast patroon. Door een patroon te bedenken, en die toe te passen voor elke flow binnen de back-end kon de kwalitatief goede code schrijven. Daarbij werd het ook makkelijker om te testen, omdat dit patroon dan door gezet kon worden.

Ik heb gekozen om alle stappen te encapsuleren. Dit was het meest overzichtelijk en het makkelijkst te repliceren. Voor elke Model uit de database dat gebruikt wordt binnen de module is er een flow aangemaakt zoals in Figuur 12.



Figuur 12: Structuur van back-end

De app start de module op. De relevante router vangt de requests op, en stuurt deze de gekoppelde controller. De controller handelt de request af, en maakt een verzoek naar zijn service om de acties uit te voeren op de database. In sommige gevallen, wordt de request eerst door een translator geheeld. Deze translator format de request zodat het beter af te handelen is in de front-end. Dit houd de front-end simpel en overzichtelijk. Dit sluit aan bij de keuze van de 'Lazy Front-end' die gemaakt is in het Architecture Design.
Door alle componenten op deze manier te encapsuleren hield ik mij aan het single-responsibility principle. Deze schrijft dat een class maar een reden zou hebben om aangepast te worden. Deze duidelijke splitsing van verantwoordelijkheden maakte het makkelijk om te ontwikkelen en te onderhouden.

Resultaat
Door hard te focussen was de MVP eerder klaar dan verwacht. Met de duidelijke plan kon ik veel voortgang maken. De front-end werkte goed samen met de back-end, en omdat alle database al volledig ontworpen was vielen alle stukjes gemakkelijk op hun plek. Wat ook hielp is dat het ontwerp redelijk eenvoudig was. Door alleen de broodnodige features te ontwikkelen bleef de module overzichtelijk.

8.4 Vierde Iteratie: Refactoring

In de laatste paar dagen van het afstuderen had ik nog tijd over. De MVP was af, de Testen waren allemaal geschreven, en het verslag was af. Met de tijd die ik overhad heb ik nog gewerkt aan het eindproduct. Er waren nog wat kleine dingen die ik kon verbeteren.

Zo heb ik de front-end kunnen optimaliseren door overbodige rerenders weg te halen, de React Componenten meer generiek te maken en kleine laadschermpjes toe te voegen op plekken waar gegevens opgehaald moesten worden uit de database.
Ook heb ik de statische versie kunnen deployen naar het cluster zodat het getest kon worden en getoond kon worden bij demonstraties van de PGO.

Door de drukke agenda's van alle stakeholders was er geen tijd meer gevonden voor een laatste feedback ronde van Paul van der Boog en Hannie Piels voordat het verslag ingeleverd moest worden. Er wordt er wel gepland, maar deze vindt plaats na de deadline.

9 Testing

Ik had veel plannen voor het testen. Ik wilde user tests schrijven die uitgevoerd konden worden gebruikers, functional tests die front-end van de module door liepen, integration tests die het systeem in het geheel testen en veel unit tests die enkele elementen zouden controleren. Maar omdat de planning was veranderd, heb ik dit anders moeten aanpakken. Een volledige beschrijving is te vinden in het test plan, deze is te vinden als bijlage C.

9.1 Doel

Ik had drie doelen voor het schrijven van de testen.

Aantonen van kwaliteit van de Code
Door testen te schrijven kan je verifiëren of code werkt zoals je verwacht.

Regressie Testen
Om te controleren dat nieuwe features, oudere stukken code niet stuk maken zijn er regressie testen. Omdat het plan is om de module later uit te breiden, zijn deze erg handig.

Verifiëren van Requirements
Voor functionele requirements zullen testen worden geschreven die aantonen dat de functionaliteit werkt.

9.2 Aanpak

Ik alleen testen geschreven voor de back-end. Het front-end project was nog vol in ontwikkeling en werd vaak aangepast. Als ik hier testen voor zou schrijven, zouden deze vaak aangepast moeten worden. Dit was erg onhandig. De back-end daarin tegen was meer geïsoleerd en zeer stabiel. Dit maakt het veel beter om te testen.

Voor het schrijven van de testen heb ik gebruik gemaakt van Jest. Jest is het meest gebruikte testing framework voor javascript. Het heeft een uitgebreide API om mee te testen en wordt voor veel verschillende tools ondersteund. Het mist alleen de functionaliteiten om API calls te maken. Om deze te testen heb ik gebruik gemaakt van Super-test. Deze library was erg licht, en had veel functionaliteiten om requests te testen.

Ik heb gebruik gemaakt van 2 soorten testen.

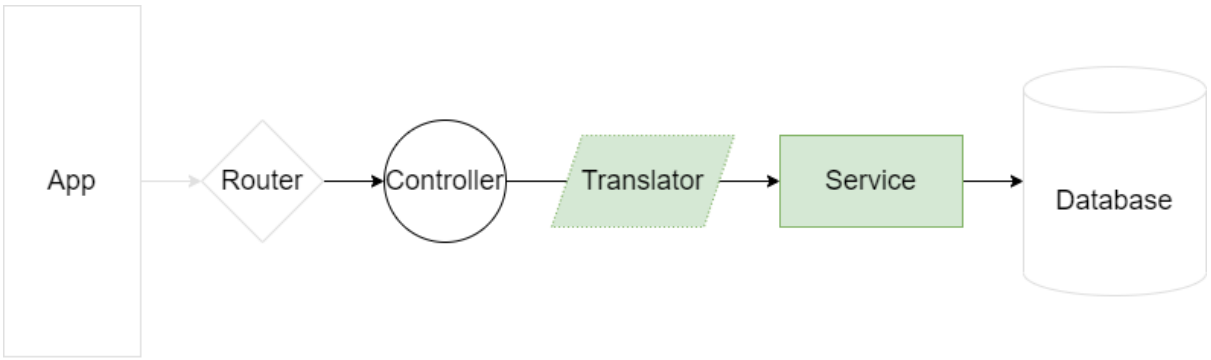
Unit Tests
Deze testen controleren de functionaliteiten van een enkele component onafhankelijk van andere testen. Deze testen moeten klein zijn, en snel kunnen runnen. Om de componenten onafhankelijk van andere te kunnen testen, heb ik gebruik gemaakt van Mocking. Deze testen dienen vooral voor regressie testen.

Integration Tests
Deze testen controleren de functionaliteiten van de back-end in het geheel. Er wordt service gestart, en er worden handelingen uit gevoerd die een gebruiker nabootsen. Het doel is om te kijken of alle componenten in de back-end goed met elkaar samenwerken. In deze testen wordt er ook gebruik gemaakt van een Database. Om corruptie van de live-database te vermeiden is er gebruik gemaakt van een in-memory database. De integration testen worden gebruikt om functionele requirements te verifiëren.

Voor het test suite heb ik twee richtlijnen opgesteld. Als eerste wilde ik 80% line coverage hebben. Dit percentage vond ik haalbaar, en groot genoeg om een relevant gedeelte van de codebase te dekken. Daarnaast wilde ik ook een verdeling van 80% unit tests tegenover 20% integration tests. Unit tests runnen veel sneller en focussen beter enkele functionaliteiten. Integration tests zijn erg waardevol omdat ze veel testen, zijn redelijk sloom. Door deze verdeling op te stellen zorg ik er voor dat de test suite niet te langzaam wordt.

9.3 Unit Tests

De structuur, zoals beschreven in Figuur 12, maakte het schrijven van de Unit testen erg makkelijk. (Bijna) Elk component is maar afhankelijk van een andere. Om een component los te testen van het andere element, heb ik gebruik gemaakt van mocking. In Figuur 13, zie je een voorbeeld van hoe een unit test van een controller is geschreven. Door de translator en de service te mocken, konden de functionaliteiten van de controller worden geïsoleerd en getest.



Figuur 13: Testen van Controller met hulp van Mocking

9.4 Integration Tests

Voor elke geïmplementeerde functionele requirement is een integration test geschreven. Zo kon ik testen of alle functionaliteiten in de module werkte zoals verwacht. Deze gingen over het invoeren van voeding, en het inzien van het voeding overzicht. Andere functionele requirements hadden verbinding nodig met de Core API, of waren niet te implementeren omdat de data hiervoor ontbrak.

De integration tests maakten gebruik van een in-memory database. Het gebruik van de live database is gevaarlijk, omdat dit voor data corruptie kan zorgen, en losse database voor testing is onhandig en moet onderhouden worden. De Data is gevuld met een eigen gemaakte dataset. Deze data was realistisch zodat het lezen van testen makkelijk is.

9.5 Code Coverage

De resultaten van de code coverage waren beter dan verwacht. De tools werkte goed samen, en door het encapsuleren waren de componenten makkelijk te testen.

In totaal zijn er 18 test files geschreven met daarin 98 tetst. Twee van de test files zijn integration tests, en de zestien andere zijn unit testen. Dit blijft goed binnen mijn 80/20 verdeling. Van alle tests is 71.1% afkomstig van de unit tests. Dit is iets onder mijn doel, maar dit kan ook betekenden dat de integration test nu juist beter is getest.

Op twee uitzonderingen is het project voor 100% getest. Bij het verbinden met de met de Routers is er een try-catch block. Als er tijdens het een error opkomt, wordt er een error message geprint, zie Figuur 14. Omdat het geen functionaliteiten heeft, heb ik het niet getest. Dit zelfde geld voor het verbinden met de database, en de middleware.

```
41      addRoutes() {
42  2x    console.log("Setting up Routes")
43  2x    try {
44  2x      setUpRoutes(this.express)
45  2x      console.log("\t Successfully set routes")
46      } catch (err) {
47        console.error(" \t Problem with setting up routes")
48        console.error("\t" + err)
49      }
50    }
```

Figuur 14: Niet getest catch block in code

De andere uitzondering is bij het verbinden met de database. Als de code in een test omgeving runt, wordt er verbonden met een in-memory database, en niet met de echte. Dit is om mogelijke data corruptie te voorkomen. Om deze reden wordt niet test wat er gebeurd als er met de live database gebeurd. Die Figuur 15 voor de code.

```
17  2x  const connect = async () => {
18  2x    E if (TEST) {
19  2x      await InMemoryDatabase.connect()
20      } else {
21        await mongoose.connect(DATABASE_FULL_PATH, DATABASE_OPTIONS)
22      }
23  };
```

Figuur 15: Niet geteste code bij het verbinden met de database

Wanneer deze twee files worden uitgesloten van de test coverage is er een 100% test coverage van het hele project. Zie Tabel 20 voor een overzicht.

Tabel 20: Test Coverage van project met uitzonderingen

Aspect	Percentage	Verdeling
Statements	100%	397/397
Branches	100%	48/48
Functions	100%	104/104
Lines	100%	390/390

10 Oplevering

Tijdens het project waren er veel uitdagingen. Deze heb goed kunnen aanpakken en uiteindelijk het afstudeertraject tot een goed einde kunnen brengen. Ik heb een MVP van de module opgeleverd en heb alle geplande beroepstaken, en meer, kunnen vervullen.

10.1 Tussenproducten

Requirements Rapport
Een rapport van 23 pagina's over de requirements van de module en hoe ze tot stand zijn gekomen. De requirements zijn beschreven en op verschillende aspecten geprioriteerd. Elke requirement is verder toegelicht voor relevante stakeholder groepen. Ook uitzonderingen beschrijven op de requirements. Het Rapport is te vinden als bijlage A.

Architecture Design
Dit verslag van 39 pagina's beschrijft de hoe de architectuur en de tools voor de module zijn gekozen. Via de ADD methode zijn alle elementen in de module afwogen aan hun relevante eigenschappen, en zijn daarmee de meest gepaste oplossingen gekozen. Er zijn verschillende schetsen gemaakt van mogelijke architectuur ontwerpen, de beste van deze is gekozen. Dit document is te vinden als bijlage B.

Test Plan
Voor de back-end is er een uitgebreide test suite gemaakt. Vrijwel 100% van alle statements, branches, functions en lines zijn getest. Er zijn twee uitgebreide integration testen geschreven die functionele requirements verifiëren. Een volledig verslag van de test aanpak is te vinden in het 17 pagina lange Test Plan. Zie bijlage C voor het Test Plan.

Proof of Concept
Aan de hand het Architecture Design is een Proof of Concept gemaakt. Deze heeft de gekozen tools getest op hun samenwerking. Hieruit bleek dat de tools erg samenwerkten en verifieerde dat de keuzes goed waren. De feedback van de stakeholders op het Proof of Concept was kritisch, maar heeft er voor gezorgd dat ontwerp veel duidelijker werd.

Code Project
De MVP van het ontwerp is ontwikkeld met de belangrijkste functionaliteiten. Er zijn twee versies opgeleverd. Een versie met Statische Data die gedeployd is en getest kan worden op een test omgeving. En een lokale versie die alle back-end en database functionaliteiten ondersteund. De front-end is gebouwd in het PGO project, maakt gebruikt van de IVIDO componenten. Zie bijlage D voor een overzicht van de module.

Migratie Verlag
Dit verslag beschrijft de stappen die zijn genomen bij het uitvoeren van de database migratie. Dit was complex omdat oude database verwarrend was, en de migratie van MySQL naar noSQL ging. Er is een nieuwe database ontwerp gemaakt die goed gebruik maakt van de noSQL functionaliteiten, en er is eigen tooling geschreven om de migratie te ondersteunen. Dit process is beschreven in het 20 pagina lange Migratie Verslag. Het verslag is te vinden als bijlage E.

10.2 Vervulling Requirements

Onder de omstandigheden heb ik zo veel van de requirements als kon waargemaakt. De twee belangrijkste functionaliteiten zijn geïmplementeerd, de front-end is geschreven in het PGO project en er is een MVP opgeleverd voor de deadline.
Voor alle requirements die ik niet heb kunnen waarmaken heb ik een duidelijke reden geven waarom dat niet is gebeurd. De grootste reden hiervan is dat er tijdens het opstellen van de requirements van uit was gegaan dat de integratie in de PGO mogelijk was.

Constraints	
✓	**(M-1) - De module moet voor 8 januari ontwikkeld zijn**
Voor de deadline waren alle tussenproduct af. Er was zelf nog tijd voor optimalisatie.	
✗	**(O-1) - De module moet geïntegreerd zijn in de PGO van IVIDO**
Omdat de authentication service nog niet af was, kon de integratie niet plaats vinden.	
✓	**(O-2) - De front-end moet ontwikkeld worden in React**
De front-end is ontwikkeld in React, is geschreven in het PGO Project.	
✗	**(O-3) - De module moet gehost worden in het cluster van IVIDO**
De functionele versie is alleen lokaal te hosten. Een statische versie staat wel op het cluster.	
✓	**(M-2) - De module mag niet meer dan €50,- per maand aan aanvullende kosten brengen**
Er is geen gebruikt gemaakt van externe libraries die geld kostten.	

Musts	
✗	**(O-4) - De module moet zijn resources evenredig schalen met gebruik**
Dit is alleen te testen met de back-end op het cluster. De gekozen tools zouden moeten schalen.	
✗	**(M-3) - De module ondersteund uitbreiding van meerdere vormen van leefstijl verbetering**
Er zijn geen testen geschreven voor een eigenschap zoals dit. Maar het architectuur ontwerp is zo gemaakt om het te ondersteunen.	
✓	**(Z-1) - De behandelingdeelnemer kan de ingevoerde voeding inzien**
Deze requirement is aangetoond met een integration test	
✓	**(P-1) - De behandelingdeelnemer kan voeding invoeren**
Deze requirement is aangetoond met een integration test	
✓	**(O-5) - De module mag maximaal 100MB aan RAM opnemen**
Op de lokale omgeving gebruikt de module ongeveer 75MB. Dit kan anders zijn op het cluster.	
✓	**(O-6) - De module moet binnen 5 seconden opstarten**
Op de lokale omgeving start de module binnen 2.5 second op. Dit kan anders zijn op het cluster.	
✗	**(O-7) - De module moet voldoen aan alle punten van de OWASP Top 10**
Veel van de maatregelingen waren niet te treffen, omdat de integratie niet volledig was uit te voeren.	
✓	**(O-8) - De module moet een test code coverage hebben van minstens 80%**
Het percentage ligt zelfs tegen de 100% aan.	

Should	
✗	**(O-9) - De module moet minder dan 10 uur aan onderhoud per maand kosten**
Er zijn geen ontwerpen gemaakt voor een mobile versie. Dit had een lagere prioriteit.	
✗	**(P-2) - De schermen moeten responsive zijn**
Er zijn geen ontwerpen gemaakt voor een mobile versie. Dit had een lagere prioriteit.	
✗	**(M-4) - De module moet het iCOPE-proces ondersteunen**
Deze feature heeft alleen waarde als de module werkt. Dit gaf het een lagere prioriteit.	
✗	**(Z-2) - De patiënt moet gekoppeld kunnen worden aan persoonlijke referentiewaarden**
De persoonlijke referentiewaarden zijn alleen te bereiken door services in het IVIDO cluster.	

Could	
✗	**(P-3) - De module moet producten aanraden gebaseerd op eerder ingevulde producten**
De data die de producten aan elkaar linkt was niet beschikbaar.	
✗	**(P-4) - De behandelingdeelnemer moet producten met verschillende niveaus van granulariteit kunnen toevoegen aan het dieet**
De data die de producten in granulariteit sorteert was niet beschikbaar.	
✗	**(P-5) - De patiënt kan voorkeursinstelling opslaan**
Gebruiker data is afkomstig uit de Core API. Deze was niet bereikbaar tijdens development.	
✗	**(Z-2) - De patiënt moet gekoppeld kunnen worden aan persoonlijke referentiewaarden**
De persoonlijke referentiewaarden zijn alleen te bereiken door services in het IVIDO cluster.	
✗	**(Z-3) - De behandelingdeelnemer moet de voedingswaarden kunnen vergelijken met persoonlijke referentiewaarden**
Omdat de requirement Z-2 niet is vervuld, kon deze ook niet vervuld worden.	
✗	**(Z-4) - De behandelingdeelnemer moet het overzicht sorteren op producten, productgroepen en voedingswaarden.**
De functionaliteit van voedingswaarden is niet geïmplementeerd. Hierdoor konden deze overzichten niet gemaakte worden.	
✗	**(P-6) - De behandelingdeelnemer moet een introductie krijgen over het gebruik van de module**
Omdat IVIDO nog wilde nadenken over de implementatie van producttours, is dit niet gedaan.	

Won't	
✗	**(P-7) - De patiënt kan aan de hand van een barcode voedingsmiddelen vinden in de module**
Deze requirement was een Won't en is daarom niet geïmplementeerd.	
✗	**(Z-5) - De module beredeneerd een advies voor behandeling**
Deze requirement was een Won't en is daarom niet geïmplementeerd.	

11 Afwijkingen van afstudeerplan

Er zijn redelijk wat verschillen tussen het afstudeerplan en de uiteindelijke uitwerking. Hier zijn verschillende oorzaken voor. Deze zijn zowel menselijk, organisatorisch als omgeving.

11.1 Meer focus op voeding

In het afstudeerplan werd beschreven hoe de focus op ‘Levensstijl als medicijn’ gelegd. In het plan zou de module focussen op verschillende elementen en het thema als een groot geheel aanpakken. Maar al snel bleek dat het niet reëel was dit alles te doen in de beperkte tijd. Er is gekozen om de focus op voeding te leggen. Voeding is de grootste oorzaak is van chronische aandoeningen, makkelijk aan te pakken is en er veel deskundige stakeholders waren op het gebied.

11.2 Oplevering & Testen

Het doel was om een uitgebreide module te maken die tijdens het traject al gedeployd werd en getest kon worden door gebruikers. Dit is uiteindelijk niet gelukt. Aan het einde van het traject is een versie met Statische Data gedeployd als Proof of Concept. Ik heb geen feedback kunnen krijgen van mensen die de module hebben kunnen gebruiken, en hierdoor heb ik ook geen user tets kunnen opstellen.

11.3 Tijd Besteding

In het afstudeerplan wordt een tijdsbesteding opgesteld. In de uitvoering is er heel veel uitgeweken van die tijdsplanning. Zo had ik verwacht dat ik zo’n tien weken kon besteden aan het ontwikkelen, maar in de realiteit werd dit maar de helft. Het opstellen van de requirements duurde twee keer zolang als gepland, en ik heb twee weken moeten besteden aan een ongeplande database migratie. Tabel 21 laat zijn hoe de tijd tijdens het afstuderen is verdeeld en Tabel 22 toont het verschillen tussen de originele planning en de uitvoering.

Tabel 21: Uiteindelijke Tijdsverdeling

Fase	Resultaat	Duur
Aanpak en Onderzoek	Plan van Aanpak	1 Week
Requirements	Requirements Rapport	4 Weken (2)
Architecture	Architecture Design en Proof of Concept	2 Weken
Database Migratie	Migratie Plan	4 Weken (0)
Ontwikkelen en Testen	MVP van Opdracht	5 Weken (10)
Oplevering	Afstudeerverslag	1 Week (2)
Originele Planning	-	-

Tabel 22: Overzicht Uiteindelijke Tijdsbesteding

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

12 Evaluatie

12.1 Beroepstaken

A1: Analyseren van probleemdomein & opstellen probleemstelling
Ik heb tevreden over hoe ik deze beroepstaak heb vervult. Ik heb me verdiept in achterliggende literatuur en heb veel gesprekken gehouden met stakeholders. Hierdoor kreeg ik een goed idee van wat het probleem precies was, en hoe ik het kon oplossen. Omdat ik met veel mensen moest samenwerken heb ik veel kanten van IVIDO leren kennen. Met deze kennis heb ik uitgebreide bedrijfsanalyse kunnen schrijven.

A3: Eliciteren en analyseren van requirements
Het analyseren van de requirements was ingewikkeld. Door het grote aantal stakeholders en verschillende visies was het moeilijk om een duidelijk doel op te stellen. Door de groep kleiner te maken heb ik het de focus van het project veel beter kunnen definiëren en zo ook beter requirements kunnen opstellen. Dit was een goede keus en heeft mij veel geholpen.
Ik heb verschillende tools en methodes gebruikt tijdens het eliciteren. Zo heb ik interviews afgenomen en manieren van ondervragen gebruikt, heb ik een Patient Journey opgesteld, een casus geanalyseerd en een marktonderzoek gedaan. Ik ben blij dat ik deze verschillende methodes heb gebruikt, omdat iedere methode het eindproduct van een andere kant belichtte.
Ik baal wel dat het zo lang duurde voordat ik het volledige ontwerp vast had staan. Soms vond ik het erg moeilijk om te begrijpen wat een stakeholder nu precies bedoelde. Maar toen ik de oude functionele ontwerpen kregen werd alles snel duidelijk. Ik had gevraagd voor alle oude documentatie, maar deze documenten had ik niet gekregen.

C6 - Ontwerpen Software
Ik uitgebreid nagedacht over de software architectuur van de module. Via de ADD methode heb ik alle belangrijke componenten kunnen wegen aan hun Architectural Drivers en zo de beste implementaties kunnen vinden.
Een aantal implementaties stonden al vast. Zo moest ik React gebruiken, en zou de module in de vorm van een Microservice zijn. Maar voor de rest had ik vrije keus. Zo heb ik verschillende architecturen, programmeertalen en database structuren aan elkaar gewogen.
Ik heb de harde keuze gemaakt om een Database Migratie uit te voeren. Dit was niet gepland, maar plek nodig te zijn om het beste eindproduct op te leveren.
Deze keuze is een bewijs van mijn controle over het project, en de visie op de toekomst.

C7 - Ontwerpen Database
Deze beroepstaak was niet opgenomen in het afstudeerplan, maar heb ik toch uitgevoerd tijdens de Database Migratie. Deze migratie was complex en koste veel tijd, daarom wil ik er nog wat aandacht aan besteden.
Ik heb een nieuwe database moeten ontwerpen voor de module. Deze database was geïnspireerd door een MySQL database waar veel fouten in stonden. In het nieuwe ontwerp heb ik de fouten in de oude database hersteld en verbeteringen toegepast zodat het beter aansloot op de module. Verder moest het oude ontwerp vertaald worden van SQL naar noSQL. Bij deze vertaling heb ik goed gebruik gemaakt van de voordelen die noSQL bied boven SQL.

Het feit dat ik dit punt extra aandacht geeft, betekend dat ik er erg trots op ben. Voor iets dat niet gepland was en met haast ontwikkelt moest worden, vind ik dat een mooi resultaat heb aangeleverd.

D14 - Realiseren Software

Het originele plan voor de module was dat het verschillende leefstijldoelen zou koppelen en de gebruiker volledig inzicht zou krijgen in hun leefstijl. Het eindproduct is heel anders geworden. Het focust zich alleen op voeding, heeft alleen de belangrijkste functionaliteiten en is niet gehost op het cluster van IVIDO.
Ondanks al de vele obstakels heb ik een goed werkende MVP kunnen opleveren. De front-end is geschreven in het PGO project en gebruikt de elementen van de IVIDO Component Library. Het inzien van de voeding is volgens het ontwerp van de stakeholders en het invullen van de voeding maakt gebruik van een uitgebreide data set. Op de valreep zijn nog een aantal optimalisaties uitgevoerd in de front-end waardoor het sneller werkt en meer gebruiksvriendelijk is.
Het is erg jammer dat de module niet gehost is in het cluster, maar had ik geen invloed op. Ik gekozen voor de oplossing om een versie met Statische Data te deployen als Proof of Concept en een volledig functionele versie te maken die op een lokaal werkt.
Ik ben tevreden met product dat ik heb opgeleverd. Ik geloof dat de code die ik heb geschreven kwalitatief goed is en soepel werkt. Door encapsulatie van de componenten is het ook nog erg goed te beheren. Ik had graag meer features willen implementeren, maar verschillende redenen waar ik geen invloed op had, heb ik dit niet kunnen doen.

D15 - Testen

Het testen van de code ging vele male beter dan verwacht. Ik dacht ergens tijdens het traject dat ik helemaal geen tijd meer over zou hebben voor het schrijven van testen. Op het einde bleek er toch nog tijd over te zijn, en heb ik een uitgebreide test suite kunnen opstellen.
In totaal heb ik 18 test files schreven met daarin 98 tests. De test coverage, op 2 kleine uitzonderingen na, is 100%.
Door de encapsulatie van de componenten waren de componenten erg goed te testen. De afhankelijke componenten konden eenvoudig gemockt worden, zodat het de functionaliteiten geïsoleerd getest konden worden. De integration testen maken gebruik van een in-memory database die de echte database simuleert.
Ik vind het jammer dat ik geen front-end testen of user testen heb kunnen schrijven. Maar in plaats daarvan heb ik mij gefocust op het testen van de back-end. Het hoge testing percentage is een teken dat het test suite goed is opgesteld.

Gc - Kritisch, onderzoekend en methodisch werken

Ik heb heel veel verschillende methodes gebruikt tijdens het afstudeer traject. Voor het opstellen van de requirements heb ik interviews afgenomen, een casus opgesteld en een Marktonderzoek gedaan. Voor het opstellen van de Architecture Design heb ik ADD gebruikt en een Proof of Concept gemaakt. Voor het testen heb ik gebruik gemaakt van unit testen met mocking en integration testen met een in-memory database. Deze methodes waren goed uit te voeren en hebben ook goede resultaten geboekt.
Voor het ontwikkelen heb ik verschillende methodes gebruikt. Ik begon met een Trello board en ben daarna met het Scrumteam van IVDIO gaan samenwerken toen ik begon met ontwikkelen. Maar toen ik merkte dat ik veel efficiënter werkte op mij zelf, ben ik in de laatste

fase weer terug gegaan naar het Trello board. Deze laatste keuze heeft zijn vruchten afgeworpen omdat dit mij veel productiever maakte.
Een aantal keer in het project heb ik drastische keuzes moeten maken. Ik moest het aantal stakeholders terugbrengen om meer controle te krijgen op het project. Later heb ik besloten om een ongeplande database migratie uit te voeren om het beste eindproduct op te leveren. Wanneer bleek dat ik de module niet kon integreren heb ik een concreet alternatief bedacht voor hoe ik het eindproduct kon opleveren. Al deze zijn gemaakt omdat ik niet vond dat deze nodig waren om een goed eindproduct op te leveren. Van al deze keuzes heb ik mijn opdrachtgever moeten overtuigen van dat het waard was om het te doen.
Mijn kritische houding tijdens het traject heeft er voor gezorgd dat ik het project tot een goed einde heb kunnen brengen.

Gf - Leren Leren
Tijdens het afstuderen heb ik heel veel dingen geleerd. Ik heb geleerd om te programmeren in React en Node.js, hoe ik mocking kan gebruiken bij het schrijven van testen en hoe het is om te werken in een startup. Maar het meeste heb ik geleerd over het managen van een project. Ik heb een aantal harde keuzes moeten maken om er voor te zorgen dat ik project tot een goed einde kon brengen. Door een actieve houding aan te nemen en goede argumenten te geven kon ik de opdrachtgever overtuigen dat mijn ideeën goed waren.
Al deze dingen heb ik geleerd door lef te hebben, en het gewoon te doen. Het traject stond onder enige tijdsdruk, waardoor er weinig keuze had om te twijfelen. Ik heb geleerd dat wanneer ik met zelfvertrouwen keuzes maak, ik niet alleen het beste resultaat boek, maar dat ik ook de mensen om mij heen mee kan krijgen.

12.2 Tussenproducten

Requirements Rapport
Ik tevreden over de inhoud van het Requirements Rapport. Alle requirements zijn uitgebreid toegelicht, op meerdere punten geprioriteerd en gerelateerd aan de relevante stakeholder groepen. Ik kreeg positieve feedback over de inhoud van de stakeholders die het hebben gelezen.
Maar ik twijfel nog over de manier van opstellen. Het was nodig om het opstellen om er voor te zorgen dat het project door kon, maar niet alles dat er in kwam stond vast. Daarom is het verslag later verouderd, en verloor het waarde als naslag werk. Bij nader inzien had ik het Rapport liever wat meer dynamisch gehouden en het later pas in definitieve vorm opgeleverd.

Architecture Design
Ik ben erg tevreden over mijn Architecture Design. De keuzes zijn weloverwogen en goed gedocumenteerd. Ik heb duidelijke diagrammen gemaakt die de ontwerpen toelichten en de onderwerpen volgde elkaar goed op. Er is geen enkele keuze in het Architecture Design waar ik later spijt van heb gehad. Zelfs niet de migratie van de database.
Wat ik jammer vind is de derde iteratie niet dieper ingaat op de structuur van de module, maar meer bijzaken behandelen. Maar deze iteratie was alsnog belangrijk omdat het dingen heeft besloten zoals de databron, de ‘Lazy Front-end’ en de Translator.

Proof of Concept
Het Proof of Concept heeft zijn drie taken erg goed gedaan. Het was duidelijk de tools erg goed met elkaar samenwerkten door de snelheid waarmee ik kon ontwikkelen. Ik heb veel geleerd over de nieuwe programmeertalen en het Proof of Concept gaf mij de ruimte om fouten te maken. Als laatste wilde ik feedback krijgen over het ontwerp van de stakeholder. De demonstratie heeft er tot geleid dat ik de functionele ontwerpen heb kunnen krijgen, en daarmee het ontwerp eindelijk goed heb kunnen begrijpen.
Ik ben erg tevreden over wat het Proof of Concept mij heeft opgeleverd.

Migratie Verslag
Ik heb het Migratie Verslag vooral geschreven om de tijd die het heeft gekost te verantwoorden. Ik wilde duidelijk maken hoe belangrijk de migratie was, en waarom de migratie zo lang duurde. Ik vind dat ik dit goed heb kunnen doen in het verslag.
Ik heb fouten in de oude database kunnen aantonen, en beschrijven waarom het nieuwe database ontwerp beter was. Ook heb ik kunnen beschrijven waarom de migratie zo ingewikkeld was en welke stappen ik heb genomen om het uit te voeren.
Hoewel het op sommige plekken misschien iets te uitgebreid, vind ik dat het verslag een zeer sterk argument geeft voor het migreren van de database.

Code Project
De uiteindelijke module heeft veel minder functionaliteiten dan origineel gepland. Ik heb maar twee functionele requirements kunnen implementeren en de module is niet gehost in het cluster van IVIDO. Ik vind dit erg jammer want ik had graag een uitgebreid eindproduct willen opleveren. Dan had ik meer kunnen programmeren, meer kunnen leren over de tools die ik gebruikte en de stakeholders erg blij kunnen maken.
Maar onder alle omstandigheden ben ik erg trots op wat ik heb opgeleverd. Door twee verschillende versies op te leveren kom ik het dichts bij het resultaat wat ik wilde. De versie met statische data is gedeployd op het cluster van IVIDO en kan gebruikt worden als Proof of Concept bij demonstraties van de PGO. De MVP heeft de twee belangrijkste functionaliteiten en koppelt de front-end, back-end en database.
De kwaliteit van de code vind ik erg goed. Door de encapsulatie is de module makkelijk te onderhouden en goed te testen. De front-end is geschreven in het PGO project, en sluit goed aan op de andere content van IVIDO.
Alle voorbereiding die ik heb gedaan voordat ik begon met ontwikkelen heeft mij veel geholpen. Ik had de juiste prioriteiten, de correcte tools, een duidelijke architectuur een goed ontworpen database. Door mij te focussen op het MVP heb ik in de korte tijd die nog beschikbaar was een stabiel eindproduct kunnen opleveren.

Test Plan
Het testen ging veel beter dan verwacht. De test-coverage ligt veel hoger dan het doel dat ik had gesteld. Ook heb ik goed gebruik kunnen maken van mocking bij de unit tests en van een in-memory database bij integration tests.
Ik heb mijn stappen goed beschrijven in het test plan. De bedachte methode werkt goed en heb ik duidelijk kunnen uitleggen met diagrammen.
Ik had graag nog tests geschreven voor de front-end. Dit leek mij een leuke uitdaging en een goede kans meer te leren. Maar hier heb ik van afgezien omdat het front-end project te instabiel was.

12.3 Aanpak tijdens afstudeerperiode

Er waren veel uitdagingen tijdens het project. Deze uitdagingen heb ik op verschillende manieren opgelost. In het begin duurde het even voordat ik mijn draai had gevonden, maar hoe verder het traject vorderde, hoe meer controller ik kreeg over de opdracht.

Planning
De originele planning is veel afgeweken van de uiteindelijke uitvoering. Bij het plannen had ik maar twee weken gereserveerd voor uitloop, de buffer.
Bij nader inzien was mijn planning naïef. Ik dacht dat ik het maken van de requirements en de architectuur soepel zou gaan, en dat ik veel tijd hebben voor ontwikkelen en testen. Als ik de tijd beter had verdeeld, had ik niet alleen minder uitloop gehad, maar had ik ook beter beloftes kunnen maken over de module omdat ik minder tijd had staan voor ontwikkelen.

Werk Methode
Ik heb methodes gebruikt om gestructureerd te werken. In de eerste paar weken heb ik gebruik gemaakt van een Trello Board. Dit werkte wel, maar ik mist wat structuur. Later, ongeveer toen ik begon met het maken van het Proof of Concept, ben ik aangesloten bij het Scrum team. Dit gaf het traject meer structuur, wat mij veel heeft geholpen om de opdracht meer te gronden. Maar in de laatste fase van het traject ben ik weer uit het team gestapt om mij te focussen op het eind product.

Dit gaf mij volledige controle over mijn manier van werken, en zo kon ik het indelen hoe ik het zelf het fijnst vond. De ‘sprints’ van vier dagen hielpen mij om mijn tijd zo efficiënt mogelijk te verdelen, en het gebruik van Trello in plaats van Jira scheelde mij veel tijd. Van alle manieren van werken, vond ik deze het fijnst. Maar ik begrijp ook dat deze manier werken niet altijd mogelijk en/of verstandig in een bedrijf. Daar is samenwerken essentieel om goed te kunnen verrichten en is het verplicht om deel te zijn van het Scrum team. De enige rede dat ik deze keuze heb kunnen maken was omdat mijn opdracht los stond van waar de rest van het team me bezig was. Als dit niet zo was, had ik dit nooit gedaan.

Eigen Initiatief nemen
Tijdens het afstuderen heb ik gemerkt dat wanneer ik de teugels in eigen hand nam, ik het meest gedaan kreeg.

- Toen ik besloot dat het eliciteren van het requirements te lang duurde, heb ik zelf de knoop doorgehakt. Hierdoor kon ik de stap afronden en door met het process.
- Later, toen ik merkte dat de Bonstat Data nodig was om het correcte eindproduct op te leveren, heb een database migratie uitgevoerd. Hierdoor heb ik het beste product kunnen maken, en kan IVIDO er verder op itereren.
- Wanneer bleek dat de module niet geïntegreerd kon worden in de PGO, heb ik een goed alternatief bedacht. Dit zorgde ervoor dat ik de opdracht goed kon afronden.
- Om mijn tijd zo efficiënt mogelijk te gebruiken, ben ik uit het Scrum team gestapt. Dit heeft er voor gezorgd dat ik de opdracht goed heb kunnen afronden.

Elk van deze keuzes heb ik op eigen initiatief genomen en hebben allemaal een zeer positief effect gehad op het eindproduct. Telkens wanneer ik het heft in eigen handen nam, heb ik heb ik project een stuk de goede kant op kunnen duwen. Toen ik dit merkte ben ik dit steeds vaker gaan doen, en ik ben erg blij met het resultaat dat het mij heeft opgeleverd.

12.4 Leerpunten

Eigen Initiatief nemen
Elke grote beslissing in het project heb gemaakt op mijn eigen initiatief. Het verkleinen betrokken stakeholders, het uitvoeren van de Database Migratie en de twee verschillende versies van het eindproduct. Elk van deze keuzes er een grote invloed gehad op het eindproduct en gezorgd dat ik het de afstudeeropdracht tot een goed einde heb kunnen brengen.
Ik heb dit al een aantal keer beschreven, maar ik vind dat dit echt het belangrijkste is dat ik heb geleerd tijdens het afstuderen. Deze vaardigheid vind ik erg waardevol en kan ik voor de rest van mijn carrière gebruiken.

Stakeholder Management
Zodra ik de lijst met stakeholders zag had er al een belletje moeten rinkelen. Negen is veel te veel voor een opdracht van deze schaal. Als ik al eerder met de opdrachtgever had samengezeten en de dit getal had terug gebracht had dit met veel tijd gescheeld. Toen de groep met stakeholders veel kleiner was, merkte ik dat het verhaal veel duidelijker werd.
In volgende projecten ga ik mijn best doen om in, de eerste fases van het project, het aantal stakeholders te beperken tot de belangrijkste.

Duidelijker Communiceren
Tijdens het afstuderen zijn een aantal dingen fout gegaan door slechte communicatie. Dit is erg jammer omdat makkelijk aan te verbeteren is, en beter is voor alle betrokkenen.
Als ik door had gevraagd na de oude documentatie, had ik misschien de functionele ontwerpen eerder kunnen krijgen. Ik kreeg te horen dat ik alles had, maar ik had blijkbaar mist ik dus nog wat dingen.
Als ik eerder met de Technical Lead had gepraat over de integratie, had ik geweten dat de module niet gehost had kunnen worden voor het eind van het afstuderen. Om eerlijk te zijn was ik hier al van uit gegaan, omdat ik het van zelfsprekend vond dat dit kon.
Door de Corona maatregelingen heb ik veel thuis gezeten, waardoor het communiceren moeilijker werd. Maar dit moet ik niet gebruiken als excuus voor de miscommunicatie, dit had ik juist moeten zien als een reden om beter mijn best te doen met communiceren.
In volgende projecten moet ik beter mijn best doen om duidelijk te communiceren met de mensen met wie ik samenwerken. Liever dat ik het een keer te veel vraag, dan dat ik het helemaal niet doe.

Meer respect hebben voor Scope
In het begin van het project heb ik hoog van de toren geblazen dat ik het ontwikkelen van de module zeker wel aankon. Door mijn hoogmoed heb ik beloftes gemaakt die ik later niet kon waarmaken. Ik had geen rekening gehouden met tegenslagen, zeker niet de genen die ik heb moeten doormaken. Dit zorgde voor een moeilijke relatie tussen mij en de stakeholders, omdat ik iedere keer met mijn staart tussen mijn benen moest bekennen dat de Scope kleiner werd.
In de toekomst ben ik van plan om minder te beloven, en meer waar te maken. Door deze aanpak te nemen kan ik mijn stakeholders tevreden houden en mijzelf een hoop stress besparen.

12.5 Advies aan IVIDO

Nu het afstudeertraject is afgelopen moet ik de module opleveren aan IVIDO. IVIDO heeft aangegeven dat zij graag verder willen met het ontwikkelen van de module, en deze willen uitbreiden. Als de ontwikkelaar van de module ben ik in de beste positie om advies te geven aan IVIDO hoe zij dit het beste kunnen doen.

Potentie
De module heeft heel veel potentie. Uit het marktonderzoek bleek al dat er veel vraag is naar een app die je inzicht geeft in je de voeding. De meerwaarde die de PGO van IVIDO kan toevoegen aan zo'n app is zeer groot. Een zorgprofessional kan veel meer met de voedingsdata, dan een leek die in zijn eentje zo'n op gebruikt. Uit de casus bleek ook dat de huidige van manier van werken vaak te kort schiet. Door het makkelijker te maken om in te vullen, en het gelijk te digitaliseren, kan de kwaliteit van de zorg veel beter worden.
Later kan de module ook uitgebreid kunnen worden met meer leefstijldoelen zoals stoppen met roken, meer bewegen en beter stress beheren. De apps die nu op de markt staan bieden deze vaak maar een van deze leefstijldoelen aan. Door deze dingen te bundelen hoeft de gebruiker maar één app te gebruiken.
Een leefstijl module zoals IVIDO die voor ogen heeft is heel erg waardevol, en ik ben van mening dat de dit project zeker moeten doorzetten.

Kwaliteit van Module
Wat nu is opgeleverd kan alleen dienen als een Proof of Concept. Het kan de stakeholders overtuigen van de potentie en daardoor investering binnen krijgen. Maar voordat het in de gebruikt kan worden bij behandelingen, moet er nog veel gebeuren.

Er kan veel verbeterd worden aan het invullen van het dieet. De ontwerpen van Hannie Piels zijn goed, maar kosten veel werk om te implementeren. Verder heeft de zorgprofessional de voedingswaarden van het dieet en de referentiewaarden van de patiënt nodig om een overwogen advies te geven aan de patiënt.

Om dit goed te doen moeten er veel resources worden geïnvesteerd. Ik stel voor om er meerdere developers en een architect die er voor zorgt dat de module op de correcte plek geïntegreerd kan worden in de PGO. Een UX'er die zorgt voor de gebruikerservaring is erg belangrijk en iemand die de medische wensen vanuit de zorg kan vertalen naar features in het project is ook niet overbodig.

Prioritering
IVIDO is druk bezig met het aanbrengen van structuur in de PGO. Zo is ze bezig om de oude PGO wordt stukje bij beetje overgezet naar de nieuwe microservice architectuur. Om er voor te zorgen dat front-end consistent is wordt de IVIDO-Component-Library ontwikkelt, en ze is drukbezig om het cluster zo efficiënt mogelijk in te richten. Daarnaast worden er nog een aantal kleinere projecten met verschillende stakeholders ontwikkelt.

Al deze dingen zou ik een hogere prioriteit geven, dan de leefstijl module. Wanneer deze dingen zijn afgerond wordt de module ook makkelijker te ontwikkelen. Probeer tot dan met het Proof of Concept een investering te krijgen in de module, en begin met ontwikkelen wanneer er genoeg resources beschikbaar zijn.

13 Literatuur

1. Ministerie van Volksgezondheid, Welzijn en Sport. (2018, November). *Nationaal Preventieakkoord.* https://www.rijksoverheid.nl/onderwerpen/gezondheid-en-preventie/documenten/convenanten/2018/11/23/nationaal-preventieakkoord
2. Alexander, I. & Stevens, R. (2002). *Writing better requirements*. London Boston: Addison-Wesley.
3. Casciaro, M. (2016). *Node.js design patterns* Packt Publishing.
4. Black, R. (2014). *Advanced software testing*. V Santa Barbara, CA: Rock Nook.
5. Shin, K., Hwang, C., & Jung, H. (2017). NoSQL database design using UML conceptual data model based on Peter Chen's framework. *International Journal of Applied Engineering Research*, *12*(5), 632-636.

14 Woordenlijst

- PGO: Persoonlijke gezondheidsomgeving. Een online platform waar medische gegevens worden verzameld. Dit is het product van IVIDO
- Gebruiker: Een gebruiker van de PGO van IVDO.
- Leefstijl-als-medicijn: Het begrip dat een mensen gezonder en langer kan leven door een beter leefstijl aan te nemen.
- Leefstijl doelen: Een doel waar een mens zich op kan focussen om zijn leefstijl te verbeteren. Arts en Leefstijl classificeert de 'De zes pijlers als:
    - Gezond eten
    - Regelmatig sporten
    - Niet Roken
    - Minstens acht uur slaap per dag
    - Gematigd alcohol gebruik
    - Voorkom stress[5]
- Module: De module is de Leefstijl-als-medicijn module. Een gebruiker kan deze gebruiken om hun leefstijl doelen te bereiken.
- ‘Voeding op Maat’ Programma of Programma: Een onderdeel van de module die gebruikt wordt door gebruikers om hun om voeding bij te houden.
- Patiënt: Gebruiker van de module met een chronische, of aankomst chronische aandoeningen, die uit opdracht van de zorgprofessional de module gebruikt
- Zorgprofessional: Iemand met veel medische kennis die werkt bij een zorginstelling en patiënten begeleid bij het gebruik van de module. Bijvoorbeeld een Diëtist, Therapeut of huisarts.
- Behandelingdeelnemer: Dit beschrijft een patiënt of een zorgprofessional. Vaak hebben deze hetzelfde doel.
- Ontwikkelaar: Dit is een ontwikkelaar of systeembeheerder die aan de module, of de deployment daarvan, werkt.
- Voedingsmiddel: Een voedingsproduct dat een behandelingdeelnemer wil invullen in het programma.
- Gerecht: Meerder voedingsmiddelen samen vormen in gerecht. Deze kan een behandelingdeelnemer invullen om gemakkelijk meerdere voedingsmiddelen te registreren
- Voedingswaarde: Een bouwsteen waaruit een voedingsmiddel uit bestaat. Bijvoorbeeld eiwit, koolhydraat, vitamine B12, Kalium, etc.
- Productgroep: Een classificatie van een groep voedingsmiddelen. Bijvoorbeeld Brood, Fruit, Gebak en Koek, etc.
- Cluster: Een netwerk van mini ‘servers’ die met elkaar samenwerken. Hier wordt vaak het Kubernetes Cluster van IVIDO bedoeld waarop de PGO, en alle ondersteunde microservices worden gehost.

[5] https://www.artsenleefstijl.nl/over-ons/wat-is-leefstijl/