Professionele Bachelor Toegepaste Informatica

Ell@ v2

De nieuwe EersteLijnsAssistente

voor de Landelijke Thuiszorg

Roy Plu

Promotoren:

Dhr. Bert Brouns — Ons/Cegeka
Dhr. Kris Hermans — Hogeschool PXL

Bachelorpaper Academiejaar 2018-2019

Professionele Bachelor Toegepaste Informatica

Ell@ v2

De nieuwe EersteLijnsAssistente

voor de Landelijke Thuiszorg

Roy Plu

Promotoren:

Dhr. Bert Brouns — Ons/Cegeka
Dhr. Kris Hermans — Hogeschool PXL

Bachelorpaper Academiejaar 2018-2019

Dankwoord

In de eerste plaats wil ik Cegeka bedanken voor de kans om hun team te mogen vervoegen en deze leerrijke stage bij hen aan te mogen vatten. Ik heb de voorbije maanden enorm veel kunnen leren over de processen binnen het bedrijf en zal deze ervaringen ook zeker meedragen in mijn toekomstige IT-carrière.

In het bijzonder zou ik ook graag mijn bedrijfspromotor Bert Brouns bedanken voor de snelle begeleiding en feedback wanneer die nodig was. Verder zou deze stageopdracht niet tot stand zijn gekomen zonder de fantastische hulp en begeleiding van Alexander Martens, die het project altijd heeft opgevolgd en in goede banen leidde voor Aïmane en mij.

Hiernaast wil ik ook nog mijn hogeschoolpromotor Kris Hermans bedanken, die meteen klaarstond wanneer er een vraag was en zich duidelijk inzet om zijn studenten een vlotte stage te bezorgen.

Naast de mensen die mij op de voet hebben gevolgd tijdens het project zou ik nog graag alle collega's bij Ons bedanken, dankzij hun professionele begeleiding heb ik het project tot een geslaagd einde kunnen brengen. In het bijzonder Thomas Wittemans, die zich tijdens de eerste 2 maanden van de stageperiode ook had toegelegd om ons te begeleiden wanneer nodig.

Ook wil ik mijn ouders bedanken voor de kans die ze me hebben gegeven om deze studierichting te volgen en me hierbij te steunen.

En als laatste zou ik graag nog mijn medestudent Aïmane Najja bedanken voor de vlotte samenwerking die we hebben gehad tijdens dit project!

Ik kon me geen betere stage wensen!

Roy Plu

Abstract

Veel administratie gebeurt binnen Ons Landelijke Thuiszorg nog op papier, waardoor er nog vaak fouten gemaakt worden en dit veel tijd kost om door al het papierwerk te gaan. Om dit probleem op te lossen heeft Ons, in samenwerking met Cegeka, een vernieuwde versie van het Ell@-project opgestart om de verouderde hardware die binnen de thuiszorg gebruikt wordt te moderniseren. De reden hiervoor is dat ze niet meer afhankelijk willen zijn van een apart toestel dat een elektronische identiteitskaart van een cliënt nodig heeft om te kunnen functioneren. Verzorgenden zullen hun planning kunnen bekijken met een eigen smartphone.

De opdracht bestaat eerst en vooral uit het ontwikkelen van een Progressive Web App in Angular 7, een frontend framework dat gebruikmaakt van HTML, CSS & TypeScript, die verzorgenden toelaat om hun planning met een lijst van afspraken te bekijken. Deze planning bevat alle informatie die de verzorgende nodig zou moeten hebben over een afspraak en de bijbehorende cliënt. Ook kan de verzorgende een afspraak bevestigen wanneer een bezoek plaatsgevonden heeft. Hiernaast is er ook een frontend- en module-API ontwikkeld in ASP.NET Web API 2 die de bestaande systemen binnen Ons integreert en de nodige data van de planning bevat.

Tijdens de stage is er ook onderzoek gedaan naar een aantal andere mogelijkheden om dit project te realiseren. Als proof of concept is er een React Native en Redux *native Cross-Platform* app gebouwd die dezelfde functionaliteiten bevat als de Angular 7 app die gebruikmaakt van NgRx voor *state management*.

Aan de hand van deze twee applicaties is het mogelijk een vergelijking te maken van specifieke criteria die belangrijk zijn binnen Ons en Cegeka om een volwaardige applicatie te bouwen die aan hun eisen voldoen. De focus ligt op *state management* en *native authentication*, en of deze mogelijk zijn met de gekozen technologie. Hiernaast zijn er nog een aantal zelfgekozen criteria opgesteld namelijk *performance*, *data usage*, *distribution* en *notifications* die belangrijk zijn om een goede gebruikerservaring te realiseren met de applicatie.

In samenwerking met Aïmane Najja, die zijn onderzoek doet rond Flutter, kan Ons met deze informatie bepalen of de uitkomst van dit onderzoek in de toekomst een haalbaar alternatief zou zijn om mogelijk verder uit te werken.

Inhoudsopgave

Lijst van gebruikte figuren

Lijst van gebruikte tabellen

Lijst van gebruikte afkortingen

Afkorting	Betekenis	Uitleg
CI/CD	*Continuous Integration & Continuous Deployment*	Automatisatie van het bouwen, testen en implementeren van software
DoD	*Definition of done*	bepaalde acceptatiecriteria voor elke user story.
fAPI	*Frontend-API*	De API die rechtstreeks in verbinding staat met de applicatie.
FPS	*Frames per second*	De mate van beeldverversing.
IDE	*Integrated development environment*	computersoftware die een softwareontwikkelaar ondersteunt bij de ontwikkeling van computersoftware.
LT	*Landelijke Thuiszorg*	Overkoepelend netwerk van gezinszorg, kraamzorg, poetshulp, dagopvang, etc...
mAPI	*Module-API*	de API die in verbinding staat met de databases en gegevens doorsluist naar een fAPI.
NgRx	*Angular Reactive Extensions*	Reactive Extensions voor het webframework Angular
NPM	*Node Package Manager*	Package manager voor JavaScript frameworks
PoC	*Proof of Concept*	een methode om te demonstreren of bijvoorbeeld een idee, technologie of functionaliteit haalbaar is.
POJO	*Plain Old JavaScript Object*	Simpelste object mogelijk in JavaScript, een set van *key-value pairs*
PWA	*Progressive Web App*	websites gebouwd met webtechnologie die zich gedragen als een mobiele app.
SSO	*Single Sign-On*	Eenmalig inloggen voor toegang tot meerdere applicaties binnen het netwerk.
UI	*User Interface*	De omgeving waarmee een gebruiker kan interageren.

Inleiding

Ell@ (Eerstelijns assistente) is een innovatietraject dat enkele jaren geleden gestart is bij Ons in samenwerking met het vroegere VASCO DataSecurity (nu OneSpan). In de Thuiszorg gebeurt nog veel administratie op papier, Ell@ poogt hier verandering in te brengen.

In de simpelste vorm laat het Ell@ proces toe dat de verzorgende zijn/haar planning kan bekijken en, met toestemming van de cliënt/patiënt, kan aangeven wanneer een bezoek plaatsgevonden heeft.

Het project werd opgestart om de communicatie tussen een verzorgende & cliënt/patiënt te verbeteren en om de verzorgende de controle te geven over zijn/haar eigen planning. Ook zal de applicatie de mogelijkheid geven om de gegevens van een cliënt/patiënt gemakkelijk te weergeven met bijkomende informatie. In de applicatie moet de verzorgende ook voornamelijk zijn/haar geleverde prestaties kunnen bevestigen.

Dit eindwerk is opgedeeld in twee onderdelen: de stageopdracht en de onderzoeksopdracht. De stageopdracht omvat het hierboven beschreven project waar gedurende twaalf stageweken aan gewerkt is. De onderzoeksopdracht daarentegen is zelf opgesteld en zal als extra aanvulling dienen bij de stageopdracht met de bedoeling de kennis de verbreden over een volkomen nieuw onderwerp. Dit onderwerp dient ook relevant te zijn binnen de implementatie van de stageopdracht en/of voor het bedrijf.

De onderzoeksvraag die is opgesteld gaat een vergelijking maken tussen een Angular + NgRx Progressive Web App (PWA) & een React Native + Redux "*Native Cross-Platform*" applicatie. Met deze vergelijking wordt er bewezen of React Native een mogelijk alternatief zou zijn voor Ons om te gebruiken voor de bepaalde toepassing van de stageopdracht. Dit onderzoek zal verder worden toegelicht in het tweede onderdeel van dit eindwerk samen met een resultaat en uiteindelijk een conclusie.

I. Stageverslag

1 Bedrijfsvoorstelling

1.1 Globale situering

Het stagebedrijf Cegeka is een toonaangevend ICT-bedrijf dat kwalitatieve oplossingen op maat biedt aan zijn klanten. Het van oorsprong Belgische bedrijf is opgericht in 1988 te Hasselt door André Knaepen. Hun activiteiten bestaan vooral uit de uitwerking van complexe bedrijfstoepassingen. Ze streven altijd naar een oplossing op maat om de doelstellingen van de klant te bereiken. [1]

Figuur 1 - Cegeka gebouw Hasselt

Klanten kiezen voor Cegeka omdat ze al jaren bekend staan als een betrouwbare partner in de doorgaans 'harde' IT-wereld. Ze streven naar innovatie en focussen op de business en de resultaten. Cegeka helpt bedrijven bij het schrijven van een ijzersterk digitaal verhaal door een uitgebreid IT-aanbod, een strategische denkwijze en een praktische aanpak aan te bieden.

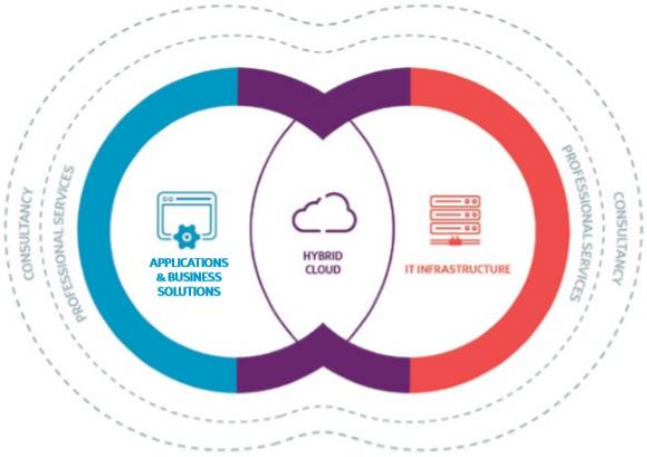
De grote vraag naar ICT-diensten over België zorgde voor een aantal regionale uitbreidingen, zo heeft Cegeka vestigingen in zowel Hasselt, Leuven, Antwerpen & Gent. Ook heeft Cegeka vestigingen verspreid over heel Europa. Ze hebben kantoren in België, Nederland, Oostenrijk, Duitsland, Frankrijk, Luxemburg, Italië, Roemenië, Slowalijke en de Tsjechische Republiek. Zo kunnen ze elke klant in elk Europees land de best mogelijke services bieden. [2]

Figuur 2 - Cegeka in Europa

1.2 Interne situering

Figuur 3 – Gebouw Ons Wijgmaal

De stageopdracht zelf verliep binnen de afdeling informatica van Ons in Wijgmaal. Ons is een netwerk van organisaties en verenigingen die zich vooral focussen op dienstverlening binnen verschillende sectoren. Ze zijn actief in sectoren zoals het verenigingsleven, thuiszorg, kinderopvang, vorming en opleiding, mantelzorg, groene zorg en diensten betaald met dienstencheques. Ons biedt werk aan meer dan 9000 werknemers. Het project dat tijdens de stage gerealiseerd wordt gaat voornamelijk gebruikt worden door de verzorgenden van de Landelijke Thuiszorg binnen Ons. [3]



Figuur 4 - Organigram Cegeka

De diensten van Cegeka zelf zijn onderverdeeld in vier hoofdafdelingen, namelijk: Applications, Business Solutions, Infrastructure en Professional Services. De afdeling informatica van Ons valt onder Applications.

- Applications: Ontwikkeling en implementatie van maatwerkapplicaties en oplossingen ontworpen voor de publieke, sociale en private sector.
- Business Solutions: Levert voornamelijk oplossingen op basis van Microsoft Dynamics 365, een innovatief platform dat traditionele ERP-, CRM- en BI-oplossingen combineert.
- Infrastructure: Consultancy op vlak van infrastructuurproblemen, outsourcing van werkplekken en de verkoop, installatie en configuratie van hardware oplossingen en projecten.
- Professional Services: Verantwoordelijk voor het aanbieden van IT consultancy services bij externe klantprojecten. [4]

1.3 Unique Selling Points

De informaticawereld draait meestal om koude, harde feiten en cijfers. Deze zijn uiteraard cruciaal. Maar Cegeka wil een verschil maken door zijn menselijke aanpak, door empathie met de zakelijke pijnpunten van zijn klanten, en door de vastberadenheid om een merkbaar verschil te maken.

Bij Cegeka zijn 'soft' skills zoals empathie, luistervaardigheid en flexibiliteit even belangrijk als technische kennis en expertise. De opbouw van een team dat een brede kennis heeft over alle aspecten van informatica is minstens even belangrijk als de opleidingen van experts in elk deelgebied.

Van Cegeka werknemers wordt verwacht dat ze zich scharen achter een aantal waarden en principes, zoals *respect*, *ownership*, leergierigheid, wendbaarheid en ondernemerschap. En van iedereen wordt verwacht dat hij of zij uit de comfortzone durft te stappen om de problemen van de klant op te lossen. [4]

1.4 Specifieke bedrijfsgegevens

2017 (meest recente gegevens) was een jaar van voortdurende expansie, innovatie en verbetering voor Cegeka. Dit komt tot uiting in dejaarcijfers: 440 miljoen euro aan omzet, een stijging van 6% ten opzichte van 2016. Het 14e jaar op rij dat Cegeka een stijgende groei realiseerden.

Ook behaalde Cegeka een operationele winst van EUR 30,5 miljoen, of een groei van 21% (bijna 7% omzet). Dit geeft een EBITDA-marge van 9,9% ten opzichte van 9,4% in 2016. De autonome groei is 5,8% (of EUR 24 miljoen) en is daarom redelijk stabiel in vergelijking met vorig jaar. [4]

Figuur 5 - Sales en EBIT Cegeka

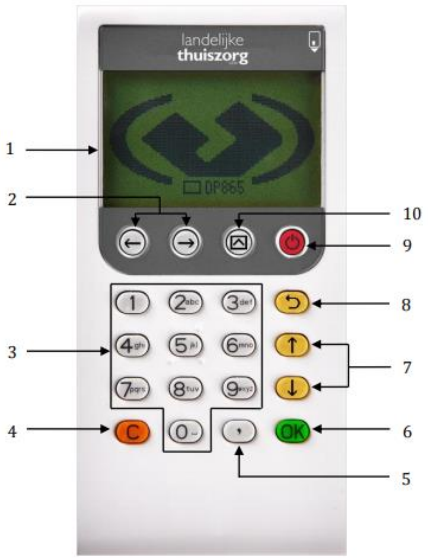
2 Voorstelling stageopdracht

Veel administratie gebeurt binnen Ons Landelijke Thuiszorg nog op papier, waardoor nog vaak fouten gemaakt worden en dit veel tijd kost om door al het papierwerk te gaan. Om dit probleem op te lossen heeft Ons, in samenwerking met Cegeka, een vernieuwde versie van het Ell@-project opgestart om de verouderde hardware die binnen de Thuiszorg gebruikt wordt te moderniseren en om te zetten naar een mobiele applicatie, omdat ze niet meer afhankelijk willen zijn van een apart toestel dat een elektronische identiteitskaart van een cliënt nodig heeft om te kunnen functioneren. Een verzorgende zal zijn/haar planning kunnen bekijken met een eigen smartphone.

De opdracht bestond eerst en vooral uit een Progressive Web App in Angular 7 te ontwikkelen, Angular 7 is een frontend framework dat gebruikmaakt van HTML, CSS en TypeScript. De app laat een verzorgende toe om zijn/haar planning te bekijken met een lijst van afspraken. Deze planning bevat alle informatie die de verzorgende moet hebben over een afspraak en de bijbehorende cliënt. Ook kan de verzorger een afspraak bevestigen wanneer een bezoek plaatsgevonden heeft. Hiernaast is er ook een frontend-API en een module-API ontwikkeld in ASP.NET Web API 2 die de bestaande systemen binnen Ons integreert en de nodige data van de planning en de afspraken bevat.

2.1 Probleemstelling

Omdat meer dan 50% van de verzorgenden binnen Ons het oude Ell@-toestel bijna dagelijks gebruiken om hun planning te raadplegen en prestaties digitaal door te sturen vormt de verouderde hardware dus een groot probleem voor Ons. Op het toestel kunnen verzorgenden hun planning raadplegen en hun prestaties bekijken, vervolgens moeten cliënten de geleverde prestaties ondertekenen met hun E-ID en pin. Hierna moeten de verzorgenden surfen naar een website en het toestel verbinden met hun laptop om hun geleverde prestaties door te geven.

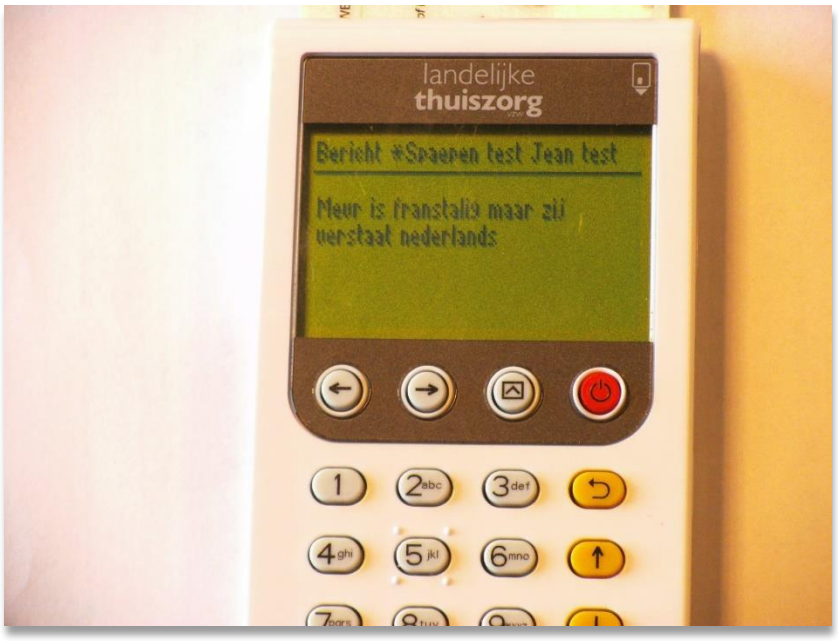


Figuur 6 – Oud toestel Ell@ v1

Dit brengt veel stappen met zich mee, waardoor op veel problemen gestoot kunnen worden. Eerst en vooral moeten de verzorgenden een opleiding volgen om het toestel correct te kunnen gebruiken. Ook is er geen ondersteuning voor meer recente technologieën, zo is het nodig om de prestaties in te leveren via Mozilla Firefox gebruikmakend van een externe plugin. Naast de correcte browser moet men ook rekening houden met de versie van het besturingssysteem op de laptop, aangezien het toestel niet correct werkt op Windows 10 en Apple producten niet ondersteund worden. Bijgevolg zullen nieuwe toestellen die wel werken met de nieuwste updates aangekocht moeten worden, wat heel duur zou zijn aangezien Ons een minimum aantal moet aankopen.

Naast de technische zaken die tijdens het proces van prestaties inleveren kunnen voorkomen, zijn er nog mogelijke problemen die het toestel met zich meebrengt. Ten eerste moeten cliënten de pincode van hun e-ID onthouden, mochten ze deze vergeten moet de klant naar de gemeente om de pincode opnieuw op te vragen. Wat niet altijd evident is wanneer een verzorgende met oudere cliënten moet werken. Ook het scherm en de toetsen van het toestel zijn moeilijk leesbaar en te gebruiken door oudere mensen.

Al deze variabelen hebben bijgevolg een invloed op het proces van een prestatie te bevestigen en kan ertoe leiden dat er ergens een fout gemaakt kan worden, zowel door de verzorgende als door de klant.



Figuur 7 – Ell@ v1 scherm met opmerking

Cegeka wil deze specifieke hardware dus niet langer gebruiken en hier liever een nieuwe versie rond ontwikkelen die bestaat uit een site of Progressive Web App.

Aangezien het project al in een bepaalde vorm bestaat, bevindt het zich in een interessante fase, met deze informatie kan er gekeken worden naar wat er zoal in het project moet toegevoegd worden en ook op welke vlakken het verbeterd kan worden. Er zijn ook al een aantal zaken aanwezig voor het huidige project waar verder op gebouwd zal worden.

2.2 Doelstellingen

De doelstelling van het project is om de administratie, die op dit moment nog met Ell@ v1 gebeurt binnen Ons Thuiszorg, te vergemakkelijken en ook kijken hoe dit mogelijk anders aangepakt kan worden door onderzoek te doen naar andere alternatieve technologieën. Aan de hand van deze onderzochte technologieën kan Ons bepalen of ze een mogelijk alternatief kunnen vormen voor de huidige Angular applicatie (Ell@ v2) die gemaakt is tijdens de stageperiode.

Ons wil eerst en vooral de oude manier van administratie vervangen door een applicatie te ontwikkelen die overal te gebruiken is (voornamelijk op de gsm van de verzorgenden). De opdracht bestaat uit een planning- en afwezighedenscherm helemaal uit te werken en te integreren in een Angular Progressive Web App. Hiernaast worden er ook bijkomende componenten, zoals een *Single Sign On* (SSO) login, geïntegreerd in de app.

3 Uitwerking stageopdracht

3.1 Omgeving / Gebruikte technologieën

3.1.1 Angular 7

Angular is het webapplicatie framework dat gebruikt wordt om de frontend te ontwikkelen. Het is een op TypeScript gebaseerd framework dat gebruikmaakt van *dependency injection*, *object binding, declarative templates*, *end to end tooling* en geïntegreerde *best practices* om op een gestructureerde manier te kunnen werken. [5] Angular stelt ontwikkelaars in staat om applicaties te bouwen voor internet, mobiel of de desktop. Het framework wordt onderhouden door het Angular Team van Google. [6]

Angular zorgt voor de *interface* waar de eindgebruiker uiteindelijk mee gaat interageren en zo gebruik kan maken van alle voorziene functies van de applicatie.

3.1.2 NgRx state management

NgRx is een framework, gebaseerd op Redux van React, om *reactive* applicaties te bouwen in Angular. Ngrx voorziet *state management*, isolatie van *side effects, entity collection management, router bindings*, *developer tools* en *code generation* om de ervaring van ontwikkelaars te verbeteren bij de ontwikkeling van verschillende soorten applicaties. [7]

NgRx zorgt ervoor dat alle opgehaalde gegevens bijgehouden worden in één gecentraliseerd en onveranderlijk object (*Single State Tree*).

3.1.3 TypeScript, HTML & CSS

Kennis van TypeScript is nodig om Angular te gebruiken. Dit is een sterk getypeerde taal en ook een superset van JavaScript. Verder gebruikt Angular ook HyperText Markup Language (HTML), een taal om webpagina's te ontwikkelen en Cascading Style Sheet (CSS), style sheets om de lay-out van de webpagina's te formatteren.

3.1.4 ASP.NET Web API 2

ASP.NET Web API 2 is een framework dat een ontwikkelaar toelaat om Web API's te bouwen. Het bestaat onder andere uit HTTP-based services bovenop het .NET-framework dat gebruik maakt van dezelfde conventies en modellen, net zoals ASP.NET MVC. Deze services kunnen gebruikt worden door een groot aantal *clients*, browsers en *mobile devices*. [8]

3.1.5 Cypress

Cypress is een *all-in-one testing framework* en *assertion library* specifiek bedoeld om end-to-end testen te schrijven. Cypress helpt bij het vergemakkelijken en automatiseren van testen, inclusief *mocking* en *stubbing*. Cypress bevat een *test runner* om snel een overzicht te krijgen van alle gemaakte testen en om deze in real time uit te voeren. [9]

3.1.6 Visual Studio Code

Visual Studio Code (ook wel *VS code* genoemd) is een "*lightweight* maar krachtige broncode-editor" ontwikkeld door Microsoft. Het bevindt zich halverwege tussen een teksteditor en een Integrated Development environment (IDE) met ingebouwde ondersteuning voor JavaScript, TypeScript, NodeJS, etc... [10]

Visual Studio code is vooral gebruikt om de frontend in te ontwikkelen. Ook de end-to-end testen van Cypress werden hiermee geschreven.

3.1.7 Visual Studio

Microsoft Visual Studio is een IDE van Microsoft. Het biedt een complete set ontwikkelingstools om computerprogramma's in diverse programmeertalen voor met name Windows-omgevingen te ontwikkelen. Visual Studio bevat ook een broncode-editor die zowel IntelliSense (het code-aanvulling-component) als refactoren ondersteunt. [11]

Visual Studio werd gebruikt voor de backendontwikkeling en het inchecken van *changes* op de *project repository.*

3.1.8 Azure DevOps

Azure DevOps is een service van Microsoft die *development teams* een aantal tools aanbiedt om de ontwikkeling te vergemakkelijken. Azure DevOps omvat een *pipeline workflow*, waar men gebruik maakt van Continuous Integration/Continuous deployment (CI/CD) om continue *builds* te maken van de laatste changes. Ook bevat Azure DevOps een Kanbanbordsysteem waar de gemaakte planning in opgevolgd kan worden. Het Kanbanbord geeft een overzicht van alle geplande *user stories* en welke ontwikkelaar een specifieke *user story* heeft opgenomen.

Hiernaast is er ook de mogelijkheid om privé Git-repo's aan te maken voor projecten waar het *development team* gemakkelijk aan de nodige projecten kan. Wanneer een ontwikkelaar een aanpassing maakt kan deze ingecheckt worden en hebben alle andere ontwikkelaars meteen toegang tot deze aanpassing. [12]

3.1.9 Slack

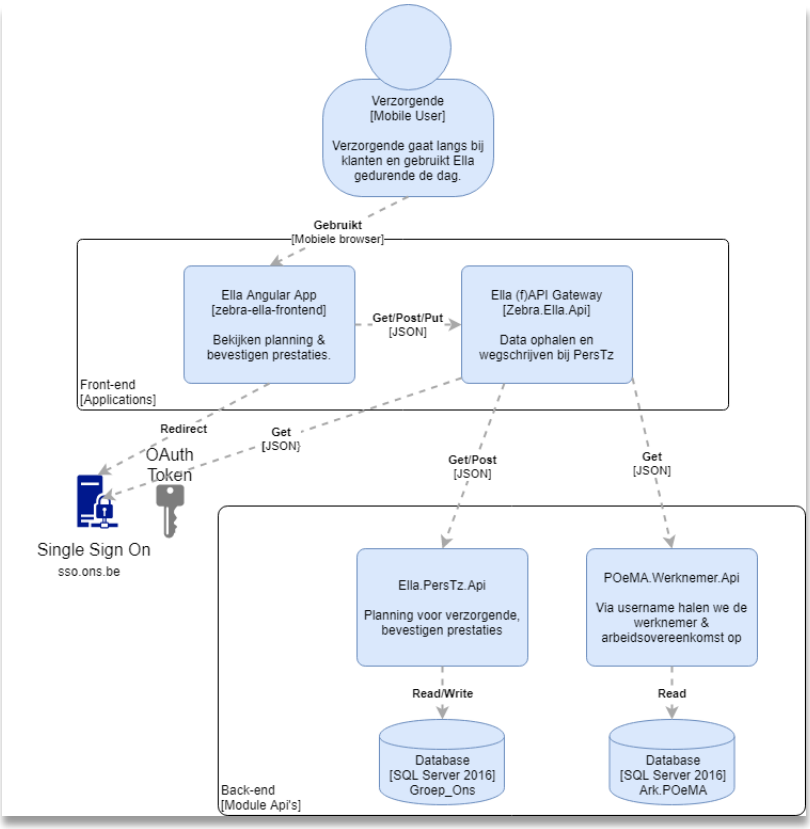
Slack is een zakelijk communicatieplatform dat alle gesprekken binnen een bedrijf samenbrengt op één plek. Met Slack kan men probleemloos groeps- en privégesprekken voeren, bestanden uitwisselen en taken verdelen. Het platform werkt op een groot aantal (mobiele) apparaten en zorgt ervoor dat je altijd en overal productief kan zijn. [13]

3.2 Planning en implementatie stageopdracht

De stageopdracht heeft twaalf weken geduurd. Tijdens deze periode is er een tijdsplanning bijgehouden van wat er op welk moment precies gebeurd is. Er was geen traditionele planning op voorhand aangemaakt aangezien de teams bij Ons op een Agile manier te werk gaan, waardoor de vereisten van het project continu konden veranderen. Wel werden er naderhand *user stories* aangemaakt van alle nodige functies binnen die applicatie die vervolgens ingeschat werden.

3.3 High level overview

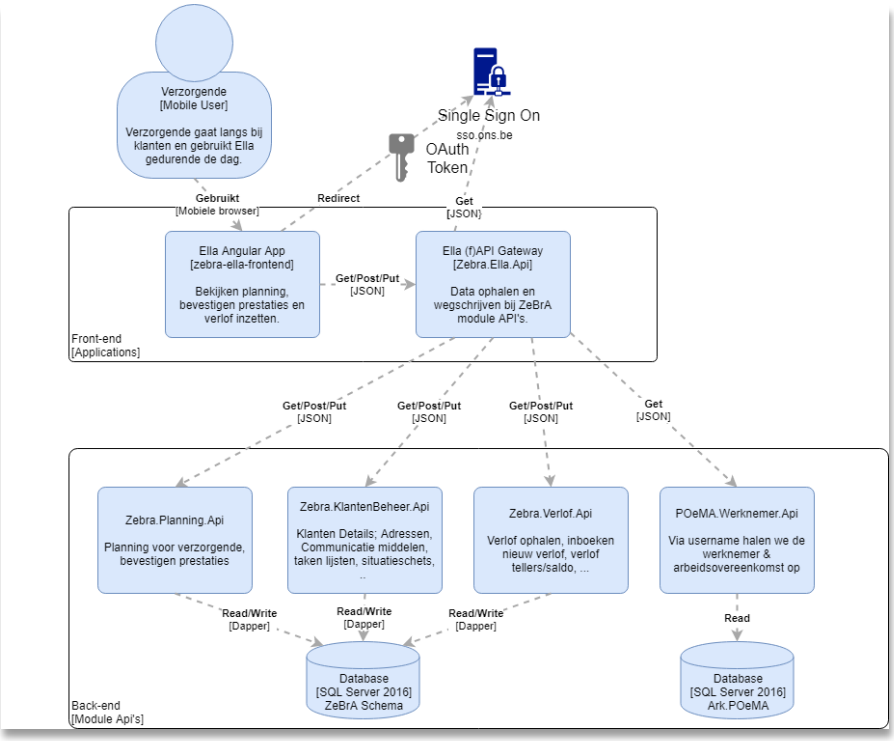
Voor de start van de stageweken waren er al een aantal voorbereidingen gemaakt om het project zo vlot mogelijk te laten verlopen. Zo is er een C4-diagram opgesteld om een overzicht te krijgen van hoe de applicatie nu precies in elkaar moet zitten, compleet met welke module-API's (mAPI) en databases geïntegreerd moeten zijn. Dit geeft al een goed idee over hoe alle systemen met elkaar verbonden moeten zijn.



Figuur 8 – C4 Diagram Ell@ - AS IS [14]

De bovenstaande C4 diagram [14] geeft de precieze flow tussen alle systemen weer van de Ell@ applicatie, zo begint de diagram bij de verzorgende, dit is de eindgebruiker die gebruik gaat maken van de applicatie op een mobile device wanneer ze langs gaan bij klanten. De interface die de eindgebruikers te zien gaan krijgen is de Angular PWA, deze bevat hun planning en alle nodige functionaliteiten om hun afspraken te bevestigen. Om deze applicatie te beveiligen maakt Ons gebruik van hun Single Sign-On systeem. Elke verzorgende zal zijn eigen gebruikersnaam en wachtwoord krijgen om in te kunnen loggen op de applicatie. Eenmaal ingelogd krijgen de

verzorgenden toegang tot de functionaliteiten van de applicatie. Wanneer de applicatie gegevens ophaalt of doorstuurt staat deze in verbinding met de Ella frontend-API (fAPI). Deze dient als een gateway tussen de applicatie en de mAPI's, die in verbinding staan met de interne databases. Dit is nodig aangezien een rechtstreekse connectie tussen de applicatie en de mAPI's niet aangeraden is. De fAPI zorgt voor een aantal mappings en conversies om de legacy data te kunnen vertalen naar de nieuwe applicatie.



Figuur 9 - C4 Diagram Ell@ - TO BE [14]

Figuur 9 - C4 Diagram Ell@ - TO BE is een recent toegevoegde C4 diagram die toont hoe de connecties tussen de systemen er in de toekomst uit gaan moeten zien. Dit is niet meer inbegrepen in de huidige stageopdracht en valt dus out of scope.

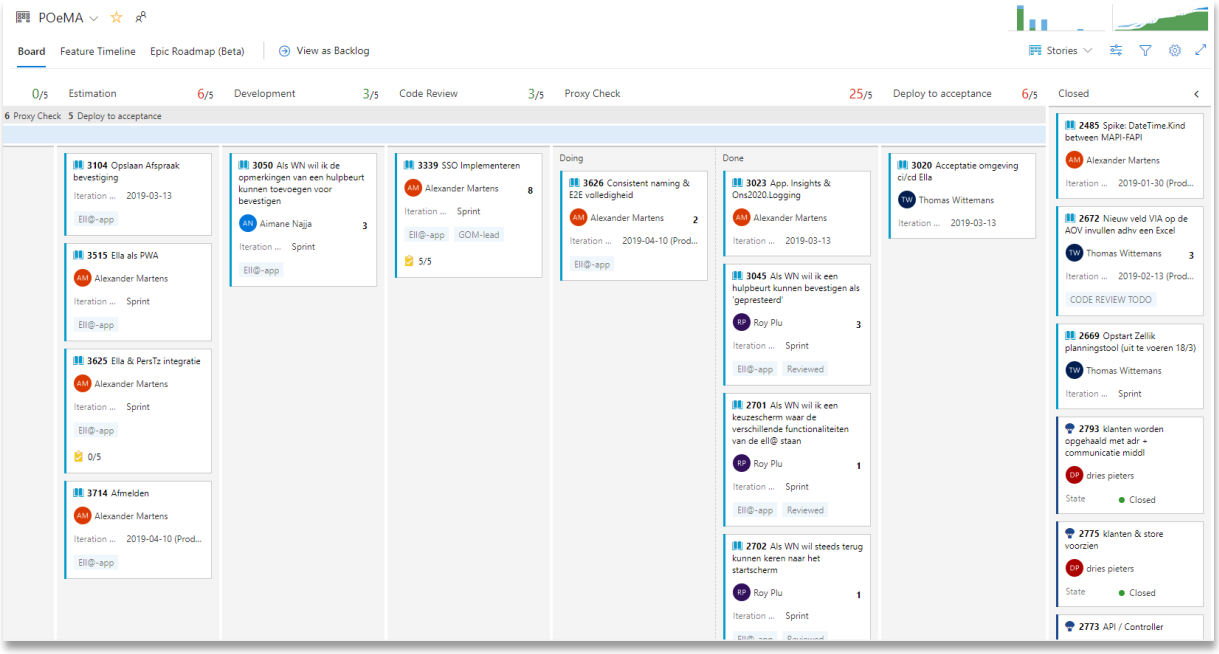
- Code voor BackEnd en FrontEnd is ontwikkeld
- Unit Testen / Integratie Testen zijn geschreven
- Automatische End To End testen zijn geschreven
- Loadtest(en) zijn geschreven / geconfigureerd
- Technische Documentatie is beschikbaar
- CI / CD is geconfigureerd
- Data Conversie / Migratie is gebeurd
- Code Review is gebeurd

Figuur 10 - Definition of Done (DoD)

Tijdens de ontwikkeling van een story moet er ook gekeken worden naar een aantal parameters waar elke ontwikkelaar bij Ons zich aan moet houden. Deze parameters zorgen ervoor dat eenmaal een story afgewerkt is, het zeker is dat deze in orde is. Eenmaal deze punten allemaal afgecheckt zijn mag de story pas op "done" gezet worden. Vandaar de naam "Definition of Done" (DoD). Het opvolgen van de DoD zorgt ervoor dat de kwaliteit van de code hoog blijft en dat de code ook meteen *production ready* is.

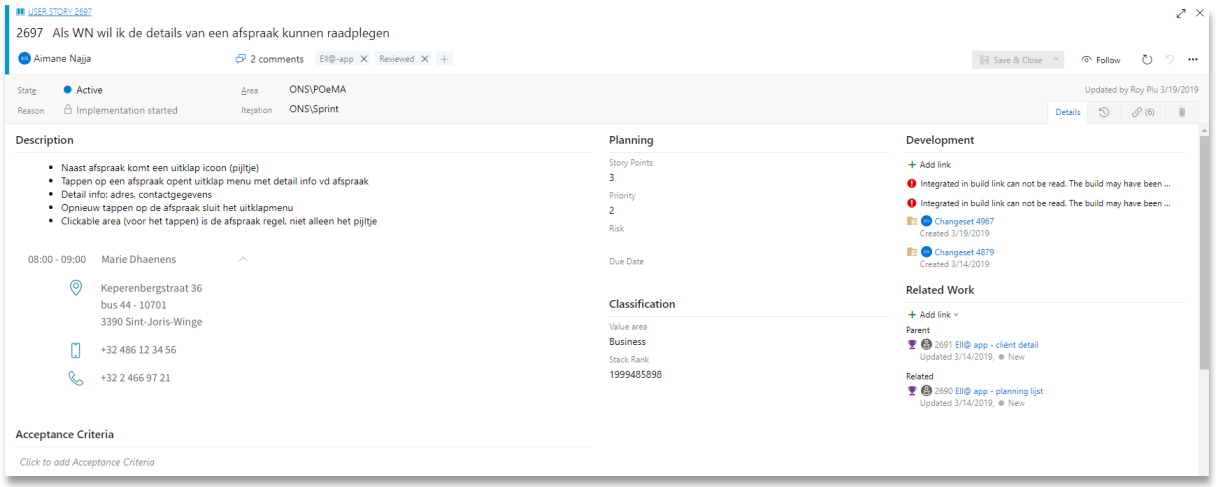
3.4 User Stories

Een *user story* is een korte beschrijving van wat een gebruiker (*user*) wil of moet kunnen in een applicatie. Een *user story* bestaat uit enkele zinnen gewone spreektaal in functie van de eindgebruiker. *User stories* worden gebruikt binnen Agile *software development* als een manier om de *requirements* te beschrijven. Deze Agile werkwijze hanteert men ook bij Ons. Alle *user stories* worden weergeven op een Kanbanbord binnen de Azure DevOps van Ons, waar de *stories* worden opgesplitst op vlak van de fase waarin een specifieke story zit.



Figuur 11 – Azure DevOps Kanban

Elk van deze *user stories* is volledig uitgeschreven inclusief alle *requirements*. *Requirements* zijn de functionaliteiten die zeker geïmplementeerd moeten zijn in een specifieke *user story*. Ook is er een deel van de Sketch prototypes [15] te zien wanneer de *developer* ook UI moet ontwikkelen voor die specifieke story.



Figuur 12 – User Story Ell@

3.5 Daily scrum / standup

Een dagelijkse *standup* wordt gehouden om de stand van zaken te bespreken. Tijdens deze standup bespreekt het team wat er de voorbije dag(en) gedaan is en wie wat precies heeft gedaan. Ook wordt er besproken wat er nog moet gebeuren aan de applicatie voor die dag. De proxy van het team wordt hier normaal gezien ook altijd in mee betrokken. Wanneer iemand niet fysiek aanwezig kan zijn voor de scrum meeting wordt er gebeld met het hele team via Skype for Business.

3.6 Estimation moments

De inschatting van de *story points* voor de *user stories* wordt gedaan tijdens een *estimation moment*. Dit moment wordt gebruikt om in team te beslissen over hoeveel tijd er ongeveer besteed moet worden aan een *user story*. Hierin geeft elk teamlid zijn eigen input over de inschatting en zo worden de *story points* van een *user story* bepaald.

3.7 Pair programming

Pair programming is het programmeren van broncode in duo's. Deze Agile werkwijze gaat ervan uit dat twee programmeurs samen achter één computer zitten te werken aan hetzelfde stuk code. Er is sprake van een duidelijke rolverdeling. [16] De "*Driver*" is de persoon die op dat moment de code aan het schrijven is en de "*Navigator*" wijst op fouten en denk na over de volgende stappen die genomen moeten worden. Ook is er de afspraak gemaakt om elke 15 minuten te wisselen van rol met medestudent Aïmane Najja. Deze werkwijze werd over de hele stageperiode gehanteerd.

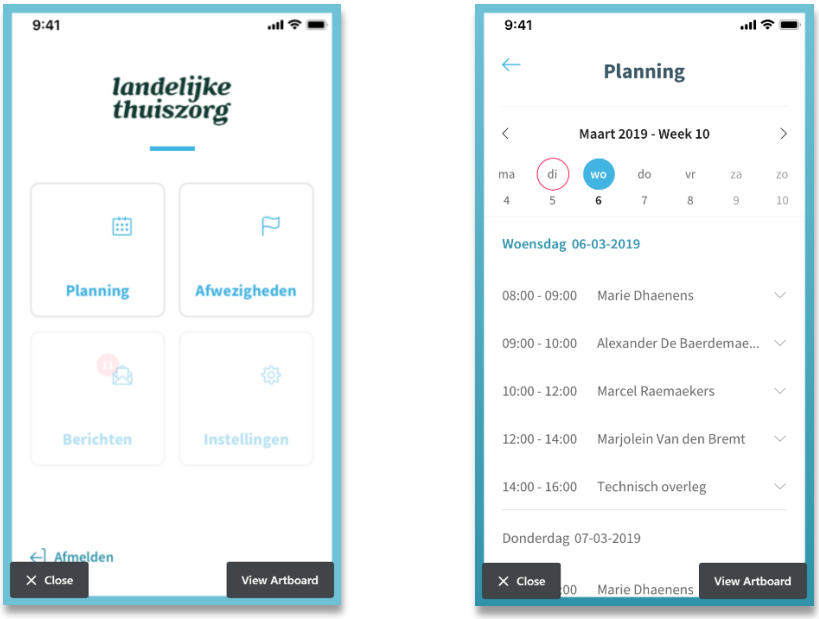
3.8 Code reviews

Een *code review* is een activiteit die binnen Agile uitgevoerd wordt om de kwaliteit van de software te waarborgen, waarbij één of meerdere mensen een programma voornamelijk controleren door delen van de geschreven broncode van een bepaalde *user story* te bekijken en zo nodig te verbeteren, en dit doen ze na de implementatie van die *user story*. Ten minste één van de mensen mag niet de auteur van de code zijn. [17]

Tijdens de stageperiode is er elke keer dat er een *user story* af was een code review gedaan door een collega binnen Ons, dit gebeurde meestal door de begeleider.

3.9 UI Mockups/Prototypes

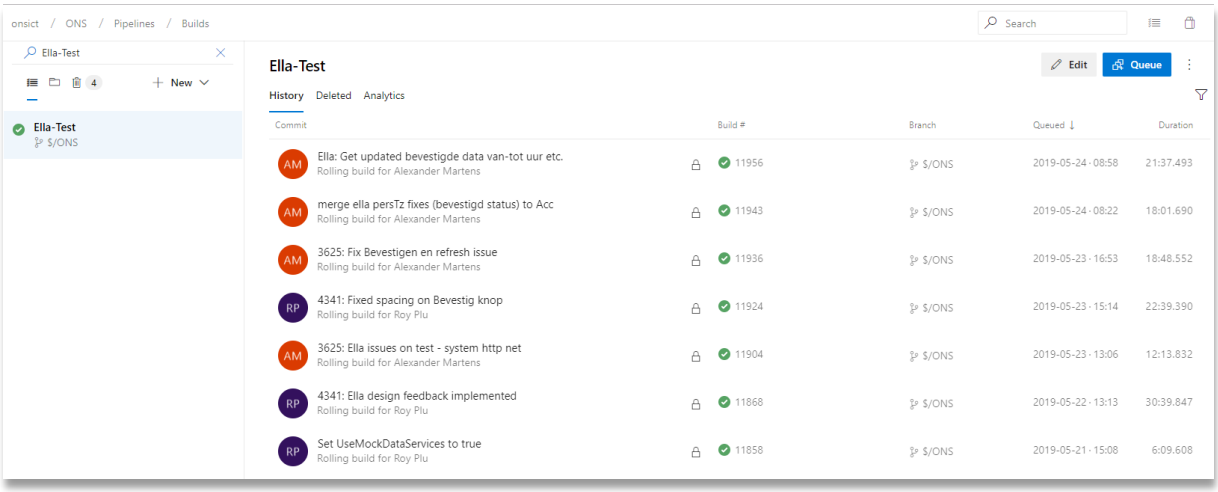
Voor de start van de stage zijn er al *User Interface* (UI) mockups gemaakt door de collega's binnen Ons. Deze mockups zijn gebruikt als basis om de UI van de Ell@ applicatie te ontwikkelen. De mockups zelf zijn ontwikkeld in Sketch. [15] Aan de hand van de mockups kan door een prototype van de applicatie genavigeerd worden waarbij alle gevraagde functionaliteiten volledig uitgewerkt en gevisualiseerd zijn. De bedoeling is om de stijl en de layout van elke mockup over te nemen en één op één na te bouwen in de Ell@ applicatie.



Figuur 13 – UI Mockups Ell@

3.10 **Project Flow**

Wanneer een *user story* afgewerkt is en ingediend kan worden gaat dit door de CI/CD *pipeline* van Azure DevOps. Deze CI/CD *pipeline* automatiseert het *delivery & deployment* proces van alle software en alle andere lopende projecten van Ons zijn hier ook op te vinden. Bij elke *check-in* van *changes* op het project worden er automatische testen *gerunned* en automatische *builds* gemaakt, die daarna terechtkomen op een productieomgeving van Ons.



Figuur 14 – Azure DevOps Build pipeline

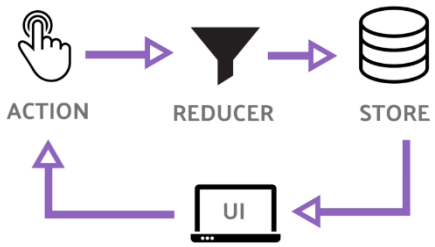
3.11 **Applicatie**

Figuur 15- Ell@ v2 logo

3.11.1 NgRx

De Ell@ applicatie, ontwikkelt in Angular 7, maakt gebruik van NgRx. NgRx (*Angular Reactive Extensions*) is een *Node Package Manager* (NPM) *package* dat kan geïnstalleerd worden in een Angular app en bevat een gecontroleerde state container die helpt om performante frontend applicaties te schrijven. Hierdoor blijft alle data van de applicatie centraal en overzichtelijk.

De implementatie van NgRx verloopt aan de hand van het Redux *pattern*, dat uit vier concepten bestaat, namelijk: State, Store, Actions en Reducers.



Figuur 16 - Redux Pattern

De state kan omschreven worden als een *plain old JavaScript object* (POJO), deze POJO is *immutable* en kan niet zomaar aangepast worden. De store is de container die de state bevat.

Actions drukken unieke gebeurtenissen uit die zich in de applicatie voordoen. Van gebruikersinteractie met de pagina, externe interactie via netwerkverzoeken en directe interactie met apparaat-API's, deze en meer gebeurtenissen worden beschreven met actions. De reducer vertaalt deze actions naar wijzigingen in de state. [18]

De effects die gebruikt worden zijn ook een onderdeel van NgRx, namelijk de @ngrx/effects library. Effects kunnen beschouwd worden als achtergrondprocessen binnen een applicatie. Effects laten toe om alle asynchrone werkzaamheden of bijwerkingen te centraliseren. Ze zorgen ervoor dat wanneer elke asynchrone activiteit voltooid is, er een synchrone actie verzonden wordt. De effects luisteren naar actions in de applicatie, en worden gebruikt voor het aanspreken van een service voor API-calls.

3.11.2 Smart/Dumb components

De applicatie is op een gestructuurde manier opgebouwd gebruikmakend van *smart* en *dumb components. Smart components* (of *container components*) houden de state bij en kunnen deze manipuleren aan de hand van selectors. Op geen enkel moment komen de *components* rechtstreeks in aanraking met de reducers.

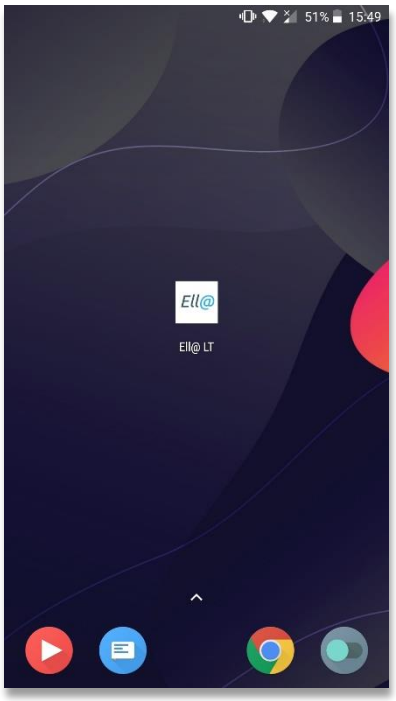
Dumb components dienen om de gegevens via *input-bindings weer te geven* en de gebruikersinterface te tonen van de applicatie en zijn onafhankelijk van business logica.

Meerdere *dumb components* kunnen onder één *smart component* zitten. Wanneer er een action gebeurd wordt dit doorgegeven aan de *parent smart component.*

3.11.3 PWA

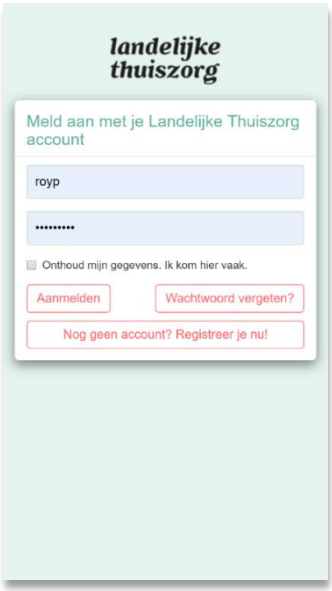
Kort gezegd is een PWA een website gebouwd met webtechnologie die zich gedraagt als een app. Het hele scherm van een mobiel apparaat wordt benut, net als bij een gewone app. Ook heeft men de mogelijkheid om de app offline te gebruiken bij een PWA door middel van service workers die caching voorzien. [19] Aangezien we een login voorzien (Zie 3.11.4) moet de gebruiker verbonden zijn met het internet om de applicatie te gebruiken, waardoor deze functionaliteit wegvalt.

Om de Ell@ applicatie te installeren moet de gebruiker eerst via de browser navigeren naar de webpagina van Ell@. Eenmaal op de webpagina toont deze een *prompt* aan de gebruiker met de vraag of de applicatie op het hoofdscherm geïnstalleerd mag worden (De *prompt* werkt voorlopig alleen nog maar bij Android, deze functionaliteit is nog niet geïmplementeerd bij iOS devices, waarbij de gebruiker de app nog handmatig moet toevoegen aan het hoofdscherm).



Figuur 17 - Ell@ PWA op hoofdscherm

3.11.4 Login (SSO)



Figuur 18 - Login (SSO) Ell@

Het eerste scherm dat de gebruiker ziet wanneer de gebruiker de applicatie opent is de Single Sign-on login van Ons. Hiermee wordt de authenticatie van de verzorgende gedaan.

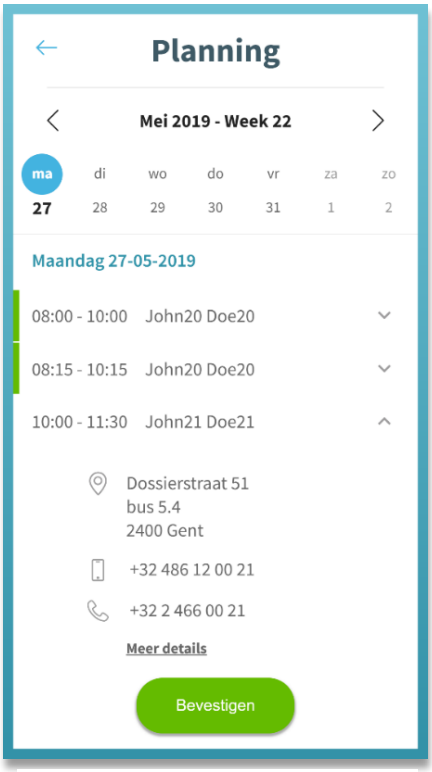
Elke verzorgende heeft zijn eigen gebruikersnaam en wachtwoord en kan hiermee inloggen op alle systemen en applicaties van Ons/Landelijke Thuiszorg (LT). De verzorgende moet zo maar één set gegevens onthouden, waardoor er minder wachtwoordopvragingen nodig zijn. Dit vergemakkelijkt het proces van inloggen.

3.11.5 Planning

Eenmaal ingelogd komt de gebruiker terecht op het hoofdscherm van de Ell@ applicatie. Hier wordt de gebruiker verwelkomt met zijn naam en heeft men de mogelijkheid om terug af te melden.

Op dit scherm zal de verzorgende uiteindelijk naar meerdere modules kunnen navigeren in de applicatie. Zo zal er een afwezighedenmodule, een berichtenmodule en een instellingenmodule toegevoegd worden in een latere fase van het project. Aangezien deze functionaliteiten *out of scope* vielen voor de huidige stageperiode, is er alleen gefocust op de planningmodule.

Figuur 19 - Hoofdscherm Ell@



Figuur 20 - Planning Ell@

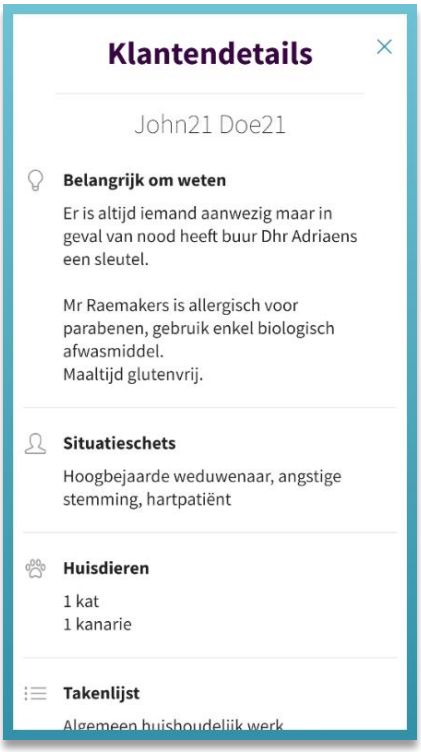
Wanneer de gebruiker verder gaat naar de planningmodule krijgt deze een overzicht te zien van alle afspraken van die week die toebehoren aan de verzorgende. Hier heeft de verzorgende de mogelijkheid om te navigeren door de voorbije weken. Er kan tot één maand terug genavigeerd worden naar afspraken, en één week in de toekomst. Deze functionaliteit wordt mogelijk gemaakt door het gebruik van Moment.js, een *open source* datum library. De mogelijkheid om automatisch te scrollen naar elke dag van de week door er op te klikken is ook geïmplementeerd.

Bij elke afspraak wordt het uur wanneer de verzorgende aanwezig moet zijn getoond. Daarnaast wordt er weergegeven wat voor soort afspraak het is en wanneer er een klant aan verbonden is zal de naam getoond worden. Een afspraak kan ook openklappen door er op te klikken, door dit te doen krijgt de verzorgende de nodige extra gegevens te zien zoals het telefoonnummer, GSM-nummer en adres (kan verschillen per soort afspraak). Ook krijgt de verzorgende de optie om nog meer details over de klant te bekijken en de optie om de afspraak zelf te bevestigen.

3.11.6 Klantendetails

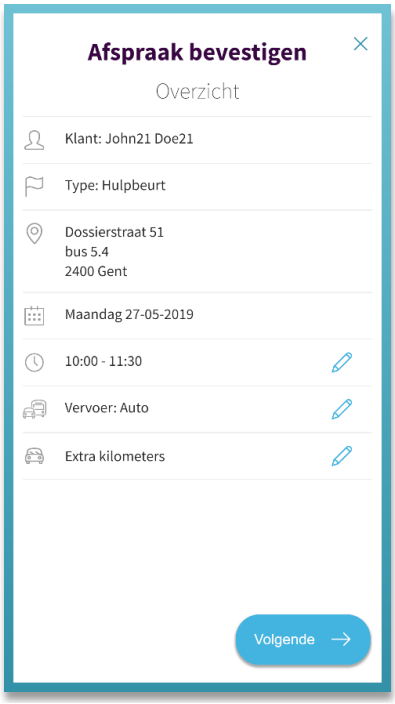
Wanneer een verzorgende klikt op “Meer details” onder een afspraak komt het klantendetailsscherm tevoorschijn. Hierin wordt de extra informatie over de klant uitgelicht en opgedeeld in een aantal categorieën.

In de takenlijst kan de verzorgende bijvoorbeeld zien wat de veel voorkomende taken gaan zijn die de verzorgende zal moeten uitvoeren tijdens deze afspraak. Ook bijkomende details zoals welke huisdieren een klant heeft of een “Belangrijk om weten” worden weergeven. Deze informatie kan mogelijk belangrijk zijn voor de verzorgende om goed voorbereid en ingelicht te zijn over de situatie en de klant zelf.



Figuur 21 - Klantendetails Ell@

3.11.7 Bevestigen van een afspraak



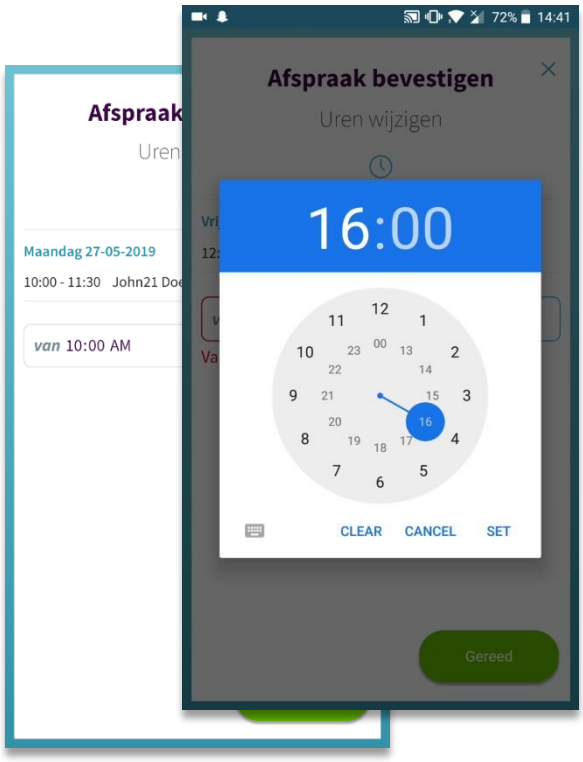
Figuur 22 - Bevestig afspraak scherm Ell@

Wanneer een verzorgende klikt op de groene “Bevestigen” knop onder een afspraak opent er een fullscreen dialog waar de verzorgende een overzicht krijgt over de informatie van de afspraak zelf. Op dit scherm kan de verzorgende ook een aantal details wijzigen moest dit nodig zijn.

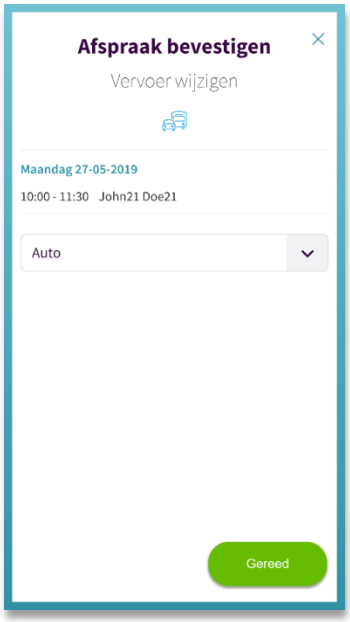
De details die een verzorgende kan wijzigen zijn de tijd van de afspraak, het vervoer dat gebruikt wordt om zich naar de afspraak te verplaatsen en de eventuele extra kilometers die toegevoegd moeten worden. Het wijzigen van elk van deze details is opgedeeld in aparte schermen waar de verzorgende toegang tot heeft.

Het eerste veld dat de verzorgende kan wijzigen is de tijd van de afspraak. Aan de hand van *native timepickers* (zien er verschillend uit op iOS en Android) kan een verzorgende het begin en einduur van een afspraak aanpassen.

Deze aanpassingen worden gecontroleerd door middel van validators die verifiëren of het beginuur voor het einduur valt.



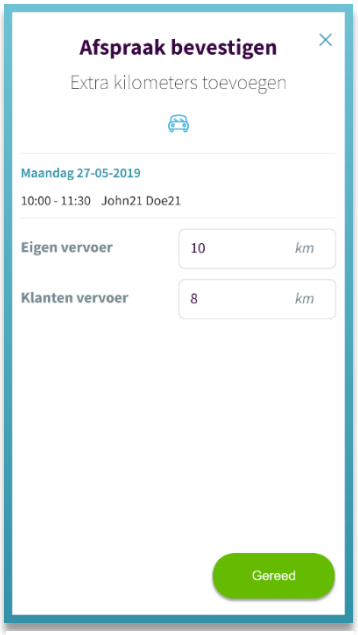
Figuur 23 - Uren wijzigen Ell@



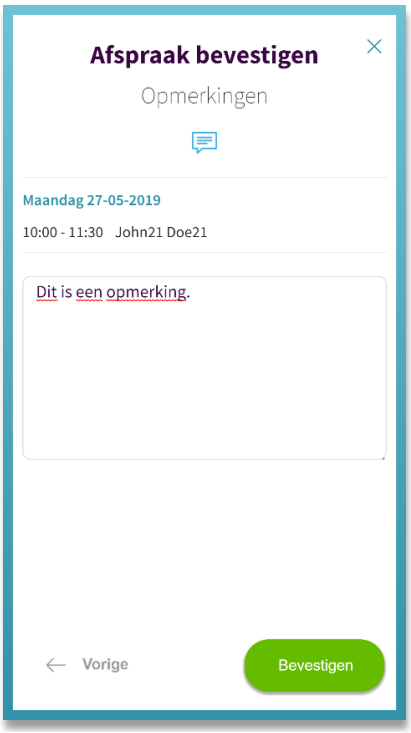
Figuur 24 - Vervoer wijzigen Ell@

Het type vervoer is het tweede veld dat een verzorgende kan wijzigen bij een afspraak. Elk soort vervoer heeft een andere vergoeding bij de Landelijke Thuiszorg, daarmee is het noodzakelijk dit correct aan te duiden zodat de afspraak juist kan worden doorgegeven. Een verzorgende heeft de optie tussen de auto, het openbaar vervoer, de fiets, te voet gaan of “andere” aan te duiden.

Het laatste veld dat bewerkbaar is bij het bevestigen van een afspraak is de extra kilometer. Hier zijn *inputs* voorzien voor zowel “Eigen vervoer” als “Klanten vervoer”. “Eigen vervoer” moet worden ingevuld wanneer de verzorgende bijvoorbeeld moest omrijden door een wegblokkade om tot bij de klant te geraken, terwijl “Klanten vervoer” ingevuld moet worden wanneer er verplaatsingen gebeuren tijdens de afspraak zelf, dit kan bijvoorbeeld voorkomen wanneer de verzorgende boodschappen gaat doen voor de klant.



Figuur 25 - Extra kilometers toevoegen Ell@



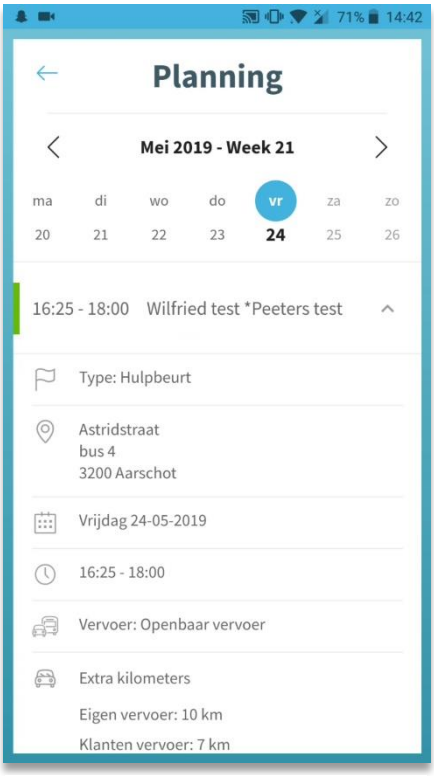
Figuur 26 - Opmerking toevoegen Ell@

Wanneer een verzorgende verder navigeert door op “Volgende” te klikken komt deze terecht op het laatste scherm voordat de afspraak uiteindelijk bevestigd wordt. Op dit scherm is een opmerkingenveld voorzien waar de verzorgende mogelijke extra opmerkingen kan achterlaten die niet van toepassing zijn bij de vorige velden. Deze opmerkingen worden mee opgeslagen met de afspraak.

Eenmaal alle stappen afgerond zijn en de verzorgende op “Bevestigen” klikt worden alle tijdelijke aanpassingen aan de afspraak permanent gemaakt, sluit de dialog en komt het planningscherm weer tevoorschijn.

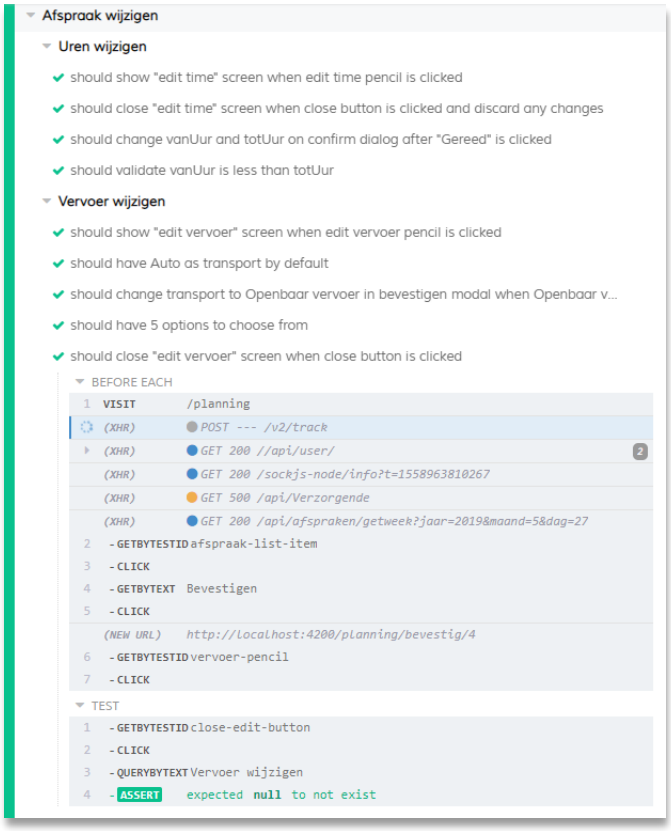
Na het bevestigen van een afspraak komt de verzorgende terug terecht op het scherm van de planning. De afspraak heeft nu een status “Bevestigd”, en dit is aangeduid door middel van de groene statusbalk naast de afspraak.

Wanneer de verzorgende de bevestigde afspraak terug openklapt komen nu alle details tevoorschijn inclusief de gewijzigde details.



Figuur 27 - Bevestigde afspraak Ell@

3.12 End-to-end testing



Figuur 28 - End-to-end testen Ell@

Voor elke actie of functionaliteit die uitgevoerd kan worden in de applicatie zijn er end-to-end testen geschreven met het Cypress framework.

Elke test gaat in *real-time* over de applicatie en doorloopt een reeks vooropgelegde stappen. Deze stappen kunnen gaan van het klikken op een bepaald element tot het checken of een specifiek DOM element een bepaalde *value* of tekst heeft.

De mogelijkheid om dieper in te gaan op een bepaalde stap en verder te debuggen is aanwezig in het framework.

Wanneer alle stappen van een test succesvol eindigen is de test voltooid en springt de Cypress *test runner* naar de volgende test.

3.13 fAPI / mAPI / Database

Om de afspraken van een verzorgende op te halen is er een frontend-API voorzien, deze frontend-API, gemaakt in ASP.NET Web API 2, voorziet de verwerking van alle gegevens om deze daarna te kunnen *serven* aan de PWA. De fAPI bevat ook gemockte afspraakgegevens waar het grootste deel van de stageperiode mee is gewerkt.

Op het einde van de stageperiode is de *switch* gemaakt naar het project te verbinden met de legacy database van Ons, Dit is een SQL Server database die ook gebruikt wordt voor Ell@ v1. Om de verbinding tussen de fAPI en deze database te leggen is er gebruik gemaakt van een module-API, die queries uitvoert op de database en deze doorgeeft aan de fAPI.

4 Besluit

4.1 Persoonlijke reflectie project

Bij het terugkijken op het project kan ik zeggen dat we een kwaliteitsvolle applicatie hebben kunnen afleveren die de functionaliteiten bevat die er moesten inzitten. De feedback die Aïmane en ik hebben gekregen doorheen de stageperiode was doorgaans heel positief. Op het einde van de stageperiode hebben we nog een korte presentatie gegeven over Ell@ aan een aantal collega's binnen Ons en iedereen was volgens mij wel onder de indruk.

De reden waarom we zo'n kwaliteitsvolle applicatie hebben kunnen afleveren is mede door de Agile werkwijze die we gehanteerd hebben doorheen het project. De vele momenten waarop we in groep samen hebben gezeten om te discussiëren over de applicatie en wat goed/niet goed is waren ten voordele van de stabiliteit en gebruiksvriendelijkheid van de app.

Ook de opvolging van de DoD heeft hier zeker bij geholpen, waarbij we ons strak hebben gehouden aan alle punten nodig om kwaliteitsvolle code af te leveren. Eerst komt het ontwikkelen van de code, zowel frontend als backend. Dit werd op de voet gevolgd door de begeleiders en wanneer er vragen waren konden Aïmane en ik deze meteen stellen. Eenmaal de code geschreven was werden er end-to-end testen geschreven, die eerst wel wat moeite vergden, aangezien dit met een voor ons onbekend framework moest gedaan worden, maar die ook zeker hun nut hebben gehad om efficiënt te kunnen zien of de opgeleverde code geen *bug*s of *issues* bevatte. Eenmaal dit afgerond was en de code ingechecked en gebuild is in de CI/CD *pipeline* kregen we ook meteen feedback op onze code aan de hand van code reviews van de begeleiders, wat voor mij persoonlijk een grote meerwaarde is, zo kon ik meteen zien of mijn code *production ready* was of niet en snel de nodige aanpassingen maken.

4.2 Persoonlijke reflectie eigen functioneren

Tijdens de eerste paar weken had ik toch wat tijd nodig om er in te komen en mezelf helemaal te geven, zo maakten we meteen kennis met een aantal voor mij onbekende technologieën die ook geïmplementeerd moesten worden in de applicatie. Uiteraard was er ruimte om vragen te stellen en heb ik dit ook gedaan wanneer nodig, waardoor de volgende weken veel vlotter liepen op vlak van het implementeren van NgRx en backend gerelateerde zaken, waar ik van mezelf wist dat deze initieel wat zwakker waren. Ook mijn sterktes heb ik kunnen tonen, zo is mijn benadering en inzien op vlak van UI/UX zeker van pas gekomen, ook mijn kennis op vlak van het Angular framework hebben me zeker geholpen bij het ontwikkelen van de frontend en hier een duidelijke, logische en gestructureerde applicatie van te maken.

Over het algemeen is mijn inzicht op vlak van de business en de processen binnen een bedrijf ook enorm gestegen, de Agile werkwijze (die we ook al toegepast hebben binnen de opleiding) heeft me beter doen inzien hoe het er nu precies aan toe gaat in een echt bedrijf en wat er precies van een ontwikkelaar verwacht wordt.

Tijdens het ontwikkelen van de applicatie zijn we hier en daar ook wel eens op een probleem gestoten, waardoor we soms langer moesten werken aan een *user story* dan eerst gedacht. Dit was eerst ook een zorg voor mij, want Aïmane en ik waren niet zeker of we wel op het juiste tempo aan het werken waren, maar de collega's van Ons verzekerden ons dat we op een goed tempo zaten (voor stagiaires) en dat wat er gepland was zeker af ging geraken, wat ons ook gelukt is.

4.3 Terugkoppeling opleiding

Tijdens mijn opleiding aan de Hogeschool PXL ben ik voor de eerste keer in aanraking gekomen met het Angular framework, dit heeft mijn ogen geopend naar hoe het ontwikkelen van moderne webapplicaties er aan toe gaat, ook al was het vak Web Expert hier maar een inleiding voor om de basiskennis mee te geven. Aangezien Angular een heel bekend en veelgebruikt framework is in het huidige IT-landschap en in professionele werkomgevingen heb ik hier veel uit kunnen leren en kon ik deze kennis ook meenemen naar dit stageproject. Ook heb ik op mezelf wat geëxperimenteerd en kleine projecten opgestart rond Angular na het volgen van dit vak om mijn kennis te verbreden.

Ook de meer algemene programmeervakken die ik doorheen de jaren heb gevolgd hebben hun nut gehad, het object geörienteerd denken en logisch redeneren is allemaal van pas gekomen bij elke nieuwe uitdaging en elke nieuwe technologie die ik voorgeschoteld heb gekregen. Vooral op vlak van de .NET vakken, waar de frontend-API en module-API in gebouwd zijn. Tijdens deze lessen heb ik de basiskennis opgedaan van hoe de verschillende datalayers van een applicatie er precies uit moeten zien, hoe men moet verbinden met een database en hoe deze data doorgestuurd kan worden.

Hiernaast heb ik ook veel kennis opgedaan bij het vak Software Engineering, dat mij introduceerde tot de vele design patterns en het SOLID principe, wat belangrijke concepten zijn om performante, goed gestructuurde en leesbare code te kunnen schrijven.

II. Onderzoekstopic

1 Inleiding onderzoeksopdracht

1.1 Probleemstelling / Vraagstelling

De probleemstelling die gevormd is voor een mogelijk onderzoekstopic is of de huidige implementatie van een Angular PWA de beste oplossing zou zijn voor dit specifiek project. Ell@ zal voornamelijk op mobiele apparaten gebruikt worden door verzorgenden van de Landelijke Thuiszorg. Om een mobiele applicatie te ontwikkelen zijn er meerdere opties elk met hun eigen voordelen en nadelen. In de volgende hoofdstukken van het onderzoek zal dit verder beschreven en toegelicht worden.

De vraag die hierbij gesteld kan worden is of er een alternatief voor de Angular PWA zou kunnen zijn en of dit alternatief ook een betere en/of gemakkelijkere oplossing zou zijn, zowel voor het team van ontwikkelaars als voor de eindgebruikers van de Ell@-applicatie.

Eenmaal een haalbaar onderzoeksonderwerp dat nauw aansluit bij het huidige project is bedacht, is er vanuit Ons ook gevraagd om naar een aantal criteria zeker te kijken. Deze criteria omvatten een aantal functies die de applicatie zeker moet bezitten om dit als een alternatief te kunnen zien ten opzichte van de Angular PWA.

Vanuit deze stellingen is er een algemene onderzoeksvraag opgesteld om een duidelijke vergelijking uit te kunnen voeren tussen de twee oplossingen, enerzijds de Angular PWA, anderzijds de nieuwgekozen technologie voor deze onderzoeksopdracht, die ook alle gevraagde criteria bevat om een functioneel alternatief te kunnen vormen om mogelijk te gebruiken binnen Ons in de toekomst.

1.1.1 Omschrijving van de onderzoeksvraag

Tijdens de stage is er onderzoek gedaan naar een aantal andere mogelijkheden om dit project te realiseren. Hierbij is er besloten om een React Native + Redux "*Native Cross-Platform*" applicatie te bouwen als *Proof of Concept* (PoC) die dezelfde functionaliteiten bevat als de Angular applicatie die gebruikmaakt van NgRx voor *state management*.

Aan de hand van deze twee applicaties is er een vergelijking gemaakt op basis van specifieke criteria die belangrijk zijn binnen Ons en Cegeka om een volwaardige applicatie te bouwen die aan hun eisen voldoet. Vanuit Ons is gevraagd om vooral te kijken naar *state management en native authentication*, en of deze mogelijk zijn met de gekozen technologie. Hiernaast zijn er zelf nog een aantal criteria opgesteld waar in deze onderzoeksopdracht ook naar gekeken wordt namelijk: *performance, data usage, distribution en notifications.* Deze criteria zijn zeer belangrijk om een goede gebruikerservaring te hebben met de applicatie.

In samenwerking met Aïmane Najja, die zijn onderzoek doet rond het mobiele framework Flutter, kan Ons met deze informatie bepalen of het onderzoek in de toekomst een haalbaar alternatief zou zijn om mogelijk verder uit te werken.

1.2 Methode van onderzoek

1.2.1 Beschrijving aanpak

Ten eerste is er onderzoek gedaan naar een mogelijk alternatief voor het gebruikte framework bij Ell@ (namelijk Angular met NgRx *state management*) dat voldoet aan alle eisen van het stagebedrijf. De applicatie moet eerst en vooral zeker op mobile werken, waardoor er al meteen gefocust kan worden op een aantal bepaalde technologieën specifiek bedoeld voor mobile.

Zo zijn er drie grote overkoepelende concepten die elk hun eigen manier hebben om een applicatie te tonen aan de gebruiker. Deze drie zijn: "*Native apps, Hybrid apps & Web apps*". Aangezien de applicatie binnen Ons al een *web app* is, is dit idee snel weggevallen, aangezien dit geen gewenste of heel nuttige resultaten zal opleveren wanneer men een vergelijkbare technologie gaat vergelijken met de huidige Angular implementatie.

Native apps daarentegen zijn compleet verschillend, een *native app* wordt specifiek ontwikkeld voor een platform (Android, iOS, etc…) in een eigen codeertaal. Deze kan gebruik maken van alle functionaliteiten van het toestel. Ook is er geen internetverbinding nodig om deze te kunnen openen en gebruiken.

Als laatste zijn er nog de *hybrid apps* die, zoals een *web app*, gebruik maken van een browser om de applicatie te tonen aan de eindgebruiker. Het verschil t.o.v. *web apps* is dat deze de applicatie in een *webview* 'omhullen' om deze zo *native* mogelijk te laten lijken, Dat wil zeggen: Het oogt als een normale applicatie op je *homescreen* en kan ook gedownload worden via de *app store* van bijvoorbeeld Apple of Google, maar het blijft nog altijd een *web app* met een aantal beperkingen, een *native app* heeft meer features en meer toegankelijkheden.

Al deze technologieën hebben hun voor- en nadelen, *native* is zo goed als altijd de beste oplossing, maar kost meer tijd, aangezien er dan voor meerdere besturingssystemen een aparte applicatie gemaakt moet worden.

Aangezien men wil dat de PoC op een zo groot mogelijk bereik van apparaten kan werken, is er gekozen voor een technologie die *native* applicaties kan ontwikkelen, maar ook *cross-platform*, waardoor er geen twee compleet verschillende *codebases* moeten zijn. Deze technologie genaamd "React Native" maakt gebruik van bestaande webtechnieken, zoals Javascript en een vorm van HTML & CSS om de applicatie te ontwikkelen, om deze daarna om te zetten naar *native* code die verschilt per besturingssysteem.

1.2.2 Onderzoeksvragen

Belangrijke criteria om rekening mee te houden:

***Native authentication**:* Kan het gebruikte framework gebruik maken van de vingerafdrukscanner of gezichtscanner van een *mobile device* als authenticatie?

State management: Heeft het gebruikte framework een manier om één *source of truth* te hebben voor alle data die is ingeladen?

Performance: Hoe presteert het gebruikte framework op *mobile devices*?

Data usage: Verbruikt de applicatie veel data wanneer men gegevens ophaalt?

Distribution: Is het gemakkelijk om de applicatie te verspreiden naar meer gebruikers?

Notifications: Kan de applicatie gebruikers verwittigen wanneer deze snel informatie nodig hebben?

Deze onderzoeksvragen zijn het belangrijkst om te beantwoorden aan de hand van het gevoerde onderzoek. Om deze te beantwoorden is er een *Proof of Concept* die deze criteria heeft geïmplementeerd. Hierna zijn er metingen en vaststellingen gedaan met de PoC om tot een besluit te komen.

1.2.3 Opgezette experimenten

Aan de hand van een PoC in React Native + Redux gaan er experimenten uitgevoerd kunnen worden op vlak van de specifieke criteria die bepaald zijn. Deze experimenten zullen resultaten opleveren die gebruikt kunnen worden om een doorgevoerde vergelijking en analyse te maken tussen de gemaakte applicatie binnen het stagebedrijf en de PoC.

1.3 Literatuurstudie

1.3.1 Introductie

Met deze literatuurstudie wordt aangetoond wat de best mogelijke optie is om een React Native applicatie mee te ontwikkelen en wat de verschillen en overeenkomsten tussen de mogelijke ontwikkelmanieren zijn. Er zijn meerdere mogelijkheden om een React Native applicatie te ontwikkelen, namelijk: Vanilla React Native, React Native met Expo en als laatste React Native met ExpoKit.

Elk van deze opties heeft zijn voor- en nadelen en worden elk apart verder toegelicht om uiteindelijk een vergelijking op te stellen en de gekozen optie voor te stellen.

1.3.2 Vanilla React Native

Er zijn twee hoofdmethoden om een React Native app te initialiseren en te ontwikkelen: Expo en de React Native CLI, met nog een aparte methode genaamd ExpoKit die ook besproken wordt in deze vergelijkende studie.

De initialisatie van een app met de React Native CLI (oftewel Vanilla React Native), via het *react-native init* commando, biedt flexibiliteit ten koste van het ontwikkelgemak. Een React Native-app gemaakt met *react-native init* is een pure React Native-app, omdat geen van de native code voor de

ontwikkelaar verborgen is. [20] Zo heeft de ontwikkelaar nog steeds de mogelijkheid om native modules te schrijven in Objective-C, Swift, Java en Kotlin. [21]

Als een ontwikkelaar van plan is om *third party packages* te gebruiken is een project ontwikkelen met React Native ook de beste oplossing. Als vuistregel geldt dat als de documentatie voor een *third party package* aangeeft dat het *react-native link* commando uitgevoerd moet worden als onderdeel van het installatieproces, dit pakket alleen kan worden gebruikt met een pure React Native-app. [20]

Bij het kiezen van deze ontwikkelmethode heeft een *developer* ook altijd de keuze over welke React Native versie hij exact wil gebruiken. Naast alle voordelen zijn er ook een aantal nadelen die bij het gebruik van een pure React Native ontwikkelmethode komen, zo heeft een *developer* niet de mogelijkheid om te ontwikkelen voor iOS zonder een Mac, wat een belangrijk struikelblok kan vormen voor veel developers wanneer ze een applicatie willen bouwen voor Apple toestellen. [21]

1.3.3 React Native met Expo

De tweede methode om een React Native project op te zetten is om gebruik te maken van Expo, Expo is een toolchain opgebouwd rond React Native om snel een app op te starten. Het biedt een reeks hulpmiddelen aan die de ontwikkeling en het testen van een React Native-app vereenvoudigen en de ontwikkelaar voorziet van componenten, interface-elementen en services die meestal alleen beschikbaar zijn *in third party native* React Native componenten. [21]

Expo bestaat uit 5 met elkaar verbonden tools:

- Expo Dev tools: Dit is een browsergebaseerde grafische gebruikersinterface voor het snel opzetten van *test devices* aan de hand van een QR-code en emulators voor Expo-ontwikkeling, het streamen van apparaat- en bundlerlogboeken en het implementeren van updates.

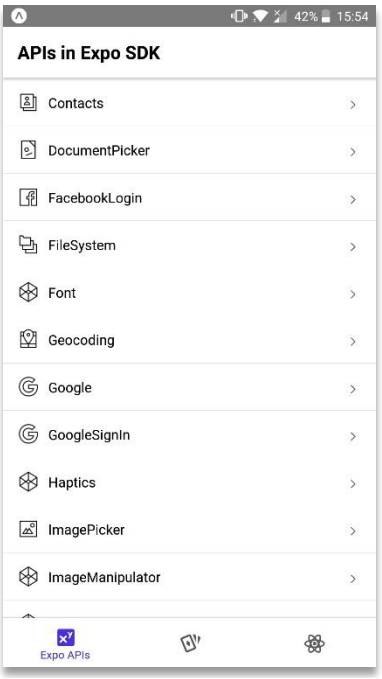
- Expo CLI: De command line interface van Expo, wordt gebruikt om terminal commando's mee uit te voeren zoals *expo init* om een project te initialiseren, *expo start* om een project te starten met de Expo Dev tools en *expo build:android* of *expo build:ios* om builds te maken van het project voor een specifiek besturingssysteem. [22]

Figuur 29 - QR-code Expo

- Expo Client: Een app voor Android en iOS.
  Met deze app kan een Expo React Native project uitgevoerd worden in de Expo Client op het apparaat zonder dat het project gebuild of geïnstalleerd moet worden. Hierdoor kunnen ontwikkelaars hot reloaden op een echt apparaat, of ontwikkelcode delen met iemand anders zonder de installatie te hoeven doen.

- Expo Snack: gehost op https://snack.expo.io, met deze tool kan er gewerkt worden aan een React Native-app in de browser, met een live preview van de code.

- Expo SDK: Dit is de SDK met een grote verzameling aan JavaScript-API's die native functionaliteit bieden die niet te vinden is in het standaard Vanilla React Native-pakket, waaronder het werken met de accelerometer van het apparaat, camera's, meldingen, geolocatie en vele andere. Deze SDK wordt mee geïnstalleerd bij elk nieuw project gemaakt met Expo. [20]



Figuur 30 - Voorbeeld APIs in Expo SDK

Een aantal voordelen van Expo zijn de snelle en eenvoudige projectinstallatie in vergelijking met Vanilla React Native. De Expo CLI en de Expo Dev tools zijn handige tools die de mogelijkheid geven om een app te runnen op een eigen *device* via een QR-code. Ook is er de optie om via de servers van Expo de applicatie te builden voor beide platformen, zo is er geen nood aan een Mac om iOS applicaties te ontwikkelen. Ook *Over the Air updates* zijn mogelijk met Expo, dit wil zeggen dat de app geupdated kan worden zonder opnieuw te *deployen* naar een app store.

Het grootste nadeel van Expo is dat de ontwikkelaar vastzit aan de componenten die in de Expo SDK zitten, zo is er geen mogelijkheid om *third party packages* te installeren met een standaard Expo project. Het schrijven van *native modules* in Swift of Java is ook geen optie bij Expo. Wanneer de ontwikkelaar toch nood heeft aan deze features bestaat er nog een derde oplossing, namelijk ExpoKit. [21]

1.3.4 React Native met ExpoKit

Als laatste is er ExpoKit, ExpoKit is een soort hybride tussen Expo-apps en Vanilla React Native. In wezen is het nog steeds een React Native-project, maar er zijn een paar dingen die verschillen over het *buildsysteem*, de ervaring van de ontwikkelaar en de beschikbare functies. ExpoKit laat de ontwikkelaar toe om een deel functionaliteiten van Expo nog te gebruiken met een Native iOS/Android project. Dit wordt ook "*Ejecting*" genoemd, aangezien men loskoppeld van een standaard Expo project naar een project dat wel *native code* en *third party packages* ondersteund [23]

Wanneer is het nodig om een standaard Expo-app los te koppelen naar ExpoKit?

- De Expo SDK voorziet de nodige *native features* niet voor een specifieke *use case*.
- Een aantal *native features* worden nog niet ondersteund door het Expo platform, zoals bijvoorbeeld Bluetooth of WebRTC. [24]

Het grote voordeel van het loskoppelen van een Expo project is dat een ontwikkelaar nog steeds toegang heeft tot de vele API's in de Expo SDK en de geïntegreerde Expo componenten, terwijl de ontwikkelaar de niet inbegrepen native functies kan overbruggen oftewel "*bridgen*" met native Android/iOS code of *third party packages*. Een nadeel hiervan is dat het project de mogelijkheid kwijtraakt om aan *hot reloading* te doen via de Expo Dev tools. Na het loskoppelen is het gebruik van Android Studio of Xcode vereist. [24]

1.3.5 Persoonlijke reflectie literatuurstudie

Elk van de onderzochte ontwikkelmethoden heeft zijn voordelen en nadelen, Zo is er de optie om de standaard methode met Vanilla React Native te behouden wanneer het nodig is om *native code* en *third party* package support te hebben, Expo wanneer de use case heel algemeen is en geen specifieke extra implementaties nodig heeft, ten voordele van het gemak van de ontwikkelaar door middel van hot reloading, automatische builds en de ondersteuning van zowel Android als iOS, en als laatste ExpoKit, dat de vele functionaliteiten van de Expo SDK integreert, ten koste van de vooropgenoemde *features.*

Uiteindelijk is mijn keuze op Expo gevallen, eerst en vooral omdat het gemak van hot reloading niet te onderschatten is. Ook de ondersteuning van de vele native functionaliteiten zoals de camera, de *fingerprintscanner* en de notificaties die ik heb ingebouwd in mijn PoC maken gebruik van de Expo SDK. De reden waarom ik niet voor ExpoKit heb gekozen, ook al is dit een hybride van zowel Vanilla als Expo, is omdat de use case van de PoC zich veel beter leent aan standaard Expo. Ik zou veel belangrijke functionaliteiten verliezen door het loskoppelen en ik was niet van plan om native iOS/Android code te schrijven bovenop React Native.

2 Uitwerking onderzoeksopdracht

2.1 Omgeving/Gebruikte technologieën

2.1.1 React Native

React Native is een *open source framework* dat, net als React, ook gemaakt is door Facebook. Het framework maakt gebruik van React en stelt ons in staat om native applicaties te schrijven. Het is niets meer dan een implementatie van React voor iOS en Android.

2.1.2 Expo

Expo is een tool gebouwd rond React Native die een developer de mogelijkheid geeft om snel een app op te starten. Expo voorziet manieren om applicatieontwikkeling te versimpelen en geeft de gebruiker een hele suite aan componenten en services die meestal alleen beschikbaar zijn via *third-party native* React Native modules. Deze worden allemaal voorzien in de zogenaamde Expo SDK. Meer over Expo is te lezen in de literatuurstudie.

2.2 Proof of Concept

2.2.1 Bevindingen voor ontwikkeling PoC

Om een Proof of Concept rond de onderzoeksopdracht uit te bouwen is er eerst research gedaan rond de gelijkenissen en verschillen tussen de huidige Angular applicatie voor Ons en de geplande React Native applicatie. De onderstaande tabel geeft een algemeen overzicht over de vele functies die beide

frameworks bezitten. Deze zijn met elkaar vergeleken om al op een aantal bevindingen uit te komen voordat een PoC is opgesteld. Deze bevindingen zijn nog niet concreet en worden verder toegelicht in de uitwerking van de PoC en de conclusie aan de hand van gemeten data.

Criteria	Angular PWA	React Native
Soort app	Progressive Web App	Mobile App
Store management	Store management met NgRx	Store management met Redux
Implementatie om view te weergeven	WebView	Gecompileerde Native code
Notificaties	Push notificaties alleen mogelijk op android	Push notificaties mogelijk
Performantie	Mogelijk lagere *performance*	Mogelijk hogere *performance*
Dataverbruik	Dataverbruik zal mogelijk hoger liggen	Dataverbruik zal mogelijk lager liggen
Distributie	Distributie alleen mogelijk voor Android play store	Distributie naar beide stores mogelijk
Authenticatie	Native authentication mogelijk, met restricties	Native authentication mogelijk
Camera	Hangt af van *device* tot *device* (Android, iOS)	Toegang tot camera mogelijk

Tabel 1 – Vergelijking bevindingen Angular vs. React Native

2.2.2 Proof of Concept

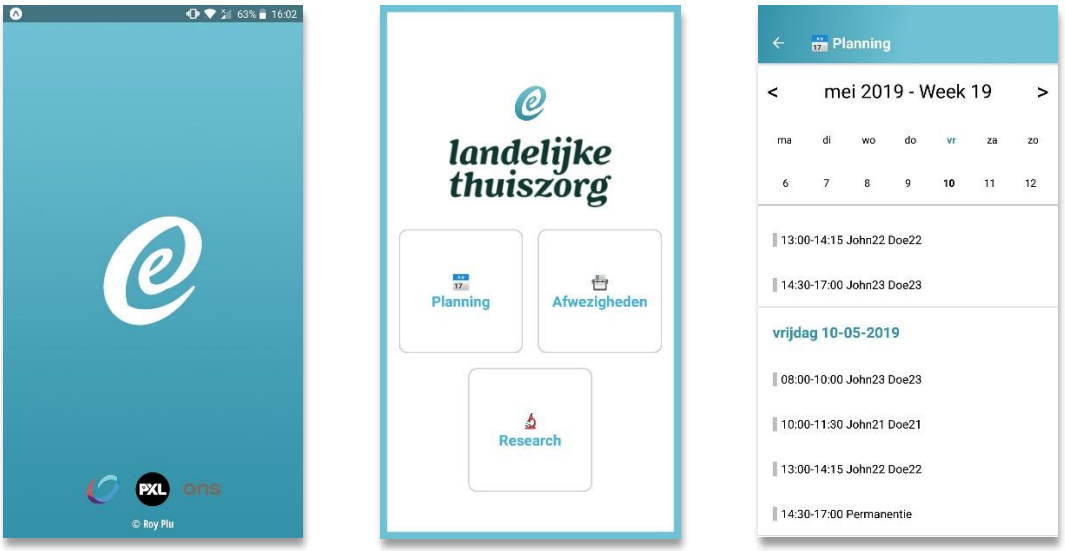
Figuur 31 - Ell@ PoC logo

Het doel van deze PoC is om aan te tonen dat de gemaakte applicatie als mogelijk alternatief kan dienen voor de Angular PWA. Ook gaat er gekeken worden naar een aantal belangrijke functionaliteiten om onderzoek naar te doen die vanuit Ons gevraagd zijn en of deze implementeerbaar zijn.

2.1 Uitwerking en resultaten

Gemeenschappelijke functionaliteiten

Planning



Figuur 32 – Hoofdschermen & planning PoC

De eerste doelstelling die bereikt moet worden om React Native als een bruikbaar alternatief te zien, is de functionaliteit na te bootsen van de Angular PWA. Dit is de grootste prioriteit en moet een idee geven van wat de mogelijkheden zijn van React Native.

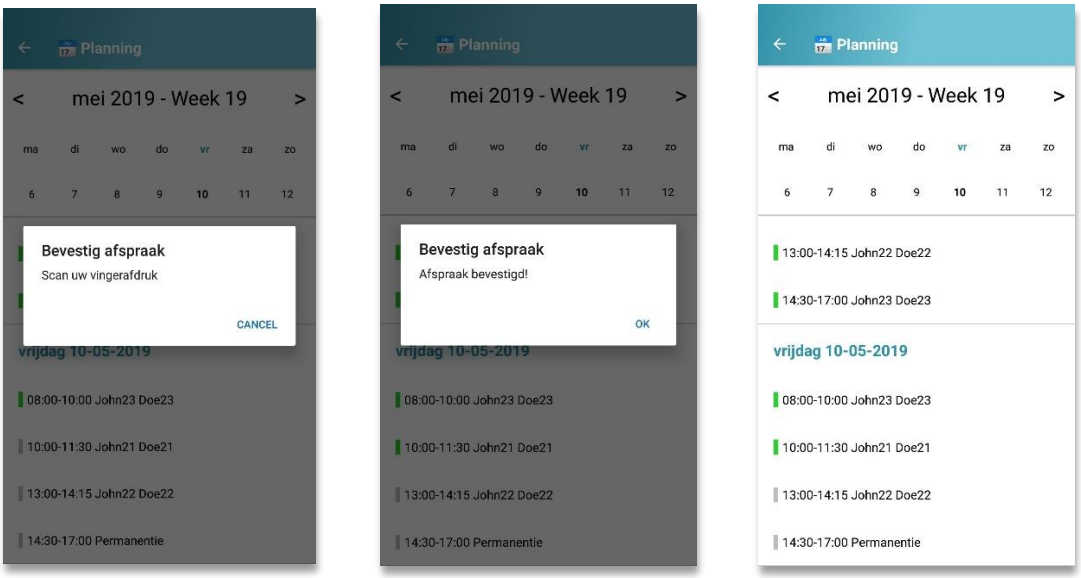
Figuur 32 – Hoofdschermen & planning PoC geeft een beeld van de schermen die gemaakt zijn voor de PoC met de focus op het aantonen van de herbruikbaarheid van de code die geschreven is voor de Angular PWA. Aangezien React Native een JavaScript framework is en ook gebruik maakt van een eigen versie van HTML en CSS, is het vrij simpel om de al geschreven code (in HTML/CSS/TypeScript) over te nemen en om te zetten naar React Native. Zo zijn alle basisfunctionaliteiten die in de planningmodule van de Angular PWA zitten ook geïmplementeerd in de PoC. Alle afspraken van de huidige week worden opgehaald en correct getoond, De huidige dag wordt getoond en het navigeren tussen de weken maakt net zoals bij de Angular PWA gebruik van Moment.js. Deze gegevens worden opgehaald aan de hand van een Node.js API server met een axios API-call.

Voor de PoC is er gebruik gemaakt van de *native* React-Navigation voor de routing van de applicatie en de *native* Flatlist componenten om de afspraken te tonen. Het is natuurlijk ook mogelijk om exact dezelfde layout als de Angular PWA te bouwen met de elementen aanwezig in React Native. De overeenkomsten in gebruikte technologieën maken van React Native heel duidelijk een waardige oplossing en de hoge herbruikbaarheid van bestaande code versterkt dit nog meer.

Native functionaliteiten

Planning

Fingerprint scanner



Figuur 33 – Bevestigen afspraak met vingerafdruk

Naast het aantonen dat React Native de functionaliteiten van de Angular PWA kan overnemen, zijn er nog een aantal native functionaliteiten die vanuit Ons gevraagd waren om te implementeren. De reden hiervoor is om aan te tonen dat React Native toegang heeft tot *device features* die een PWA niet ondersteund of waar er nog restricties op staan.

```
scanFingerprint = async item => {
  let result = await LocalAuthentication.authenticateAsync(
    Alert.alert("Bevestig afspraak", "Scan uw vingerafdruk", [
      {
        text: "Cancel",
        onPress: () => console.log("Cancel"),
        style: "cancel"
      }
    ])
  );
  console.log("Scan Result:", result);
  if (result.success == true) {
    Alert.alert("Bevestig afspraak", "Afspraak bevestigd!");
    approvedAfspraak = this.props.afspraken.filter(
      afspraak => afspraak == item
    );

    this.setState(
      {
        approvedAfspraken: this.state.approvedAfspraken.concat(item.nr)
      },
      () => {}
    );

    console.log(this.state.approvedAfspraken);
  }
  this.state.result = result;
};
```

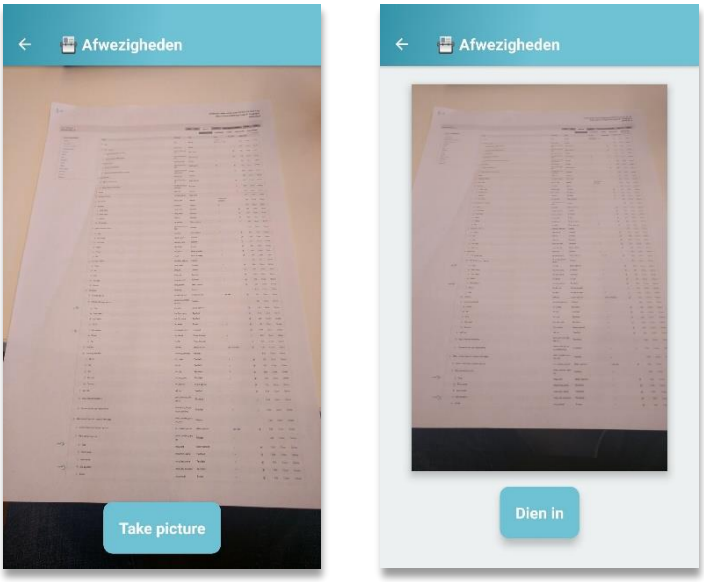
Figuur 34 - Code vingerafdrukscanner React Native

Één van deze features is de vingerafdrukscanner. Bij het klikken op een afspraak in de PoC moet de verzorgende deze bevestigen aan de hand van een vingerafdruk. Wanneer de vingerafdruk overeenkomt met de vingerafdruk opgeslagen op het apparaat wordt de afspraak bevestigd en de status op groen gezet. Dit zou ook de authenticatie kunnen zijn bij het inloggen en zo het hele proces van inloggen en authenticeren van een verzorgende kunnen vergemakkelijken.

Het grote voordeel van React Native ten opzichte van de Angular PWA is dat deze gebruik kan maken van de vingerafdrukscanner van een mobiel apparaat, wat tot op het moment van schrijven niet mogelijk is bij een PWA.

Afwezigheden

Camera



Figuur 35 – Voorbeeld indienen ziekenbriefje met camera

Een andere *feature* waar React Native toegang tot heeft is de camera van een mobiel apparaat. Deze is geïmplementeerd in een afwezighedenmodule (die *out of scope* viel voor het Ell@-project). In de afwezighedenmodule opent de camera op volledig scherm en is er de mogelijkheid om een foto te nemen van een afwezigheidsbriefje/ziekenbriefje en dit in te dienen.

```
<Camera
  style={{ flex: 1 }}
  type={this.state.type}
  ref={ref => {
    this.camera = ref;
  }}
>
  <View
    style={{
      flex: 1,
      backgroundColor: "transparent",
      flexDirection: "row"
    }}
  />
  <TouchableOpacity
    onPress={this.snapPhoto.bind(this)}
    style={styles.pictureButton}
  >
    <Text style={styles.ButtonText}>Take picture</Text>
  </TouchableOpacity>
</Camera>
```

Figuur 36 - Camera component React Native

Deze functionaliteit is geïmplementeerd om een voorbeeld te geven over hoe een verzorgende een afwezigheid kan aantonen en indienen vanuit de app.

De camera is een van de functionaliteiten die een PWA wel ondersteund, maar dit hangt ook weer af van welk besturingssysteem het mobiel apparaat heeft. Kort na de start van deze PoC heeft iOS een grote vooruitgang gemaakt op het ondersteunen van PWA's in hun ecosysteem, zo hebben Apple devices nu ook beperkte toegang tot de camera.

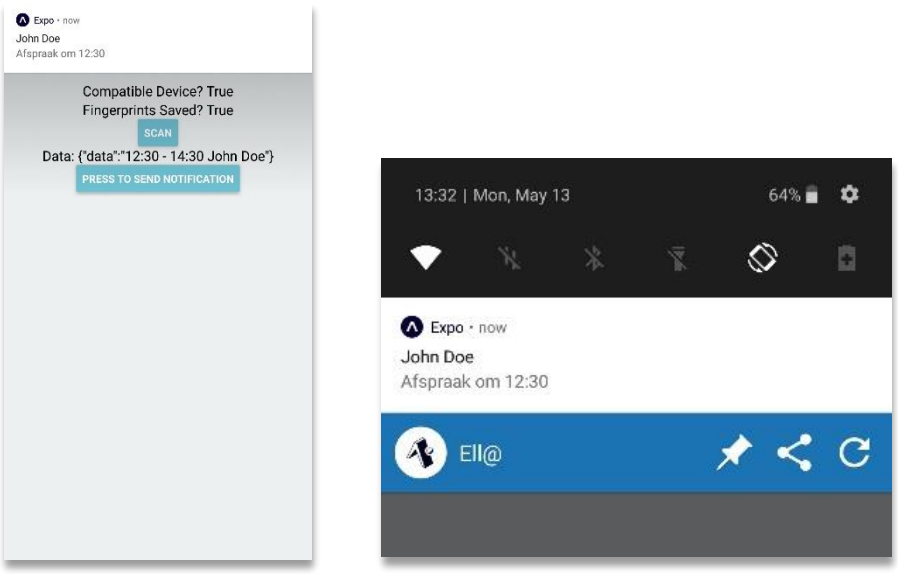
PWAs and mobile device features: what PWAs can and can't do

Feature	PWA on Android (Chrome browser)	PWA on iOS (Safari browser)
Camera and microphone		
Audio and video capture	✓	✓
Advanced camera controls	✓	✗
Recording media	✓	✗
Real-time communication	✓	✓

Figuur 37 - PWA camera features: Android vs iOS

Research

Notifications



Figuur 38 – Voorbeeld notificatie

De laatste native functionaliteit inbegrepen in de PoC is de mogelijkheid om notificaties te tonen. Zogenaamde *"push" notifications* kunnen heel nuttig zijn en belangrijke informatie snel tonen aan een verzorgende.

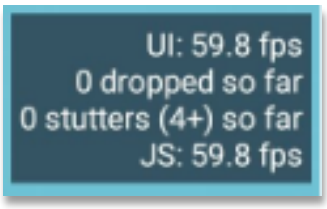
```
sendPushNotification = async () => {
  if (Platform.OS === "android") {
    Notifications.createChannelAndroidAsync("chat-messages", {
      name: "Afspraken",
      priority: "max",
      vibrate: [0, 250, 250, 250],
      sound: true
    });
  }

  const message = Notifications.presentLocalNotificationAsync({
    to: this.state.notificationToken,
    sound: "default",
    title: "John Doe",
    body: "Afspraak om 12:30",
    data: { data: "12:30 - 14:30 John Doe" },
    android: { channelId: "chat-messages" }
  });
};
```

Figuur 39 - Code notificaties React Native

Push notifications bij een PWA bouwen voort op de Web Push-standaard en ServiceWorkers. Indien geconfigureerd, wordt een ServiceWorker geactiveerd door een inkomende pushmelding. Het kan dan bijgewerkte gegevens ophalen en een melding voor de gebruiker weergeven. De meldingen worden weergegeven als normale systeemmeldingen op de smartphone. [25] Bij React Native zit deze feature ingebouwd in de Expo API. Een goed voorbeeld hiervan zou zijn dat de app een melding geeft wanneer een afspraak gaat plaatsvinden en met welke klant.

Ook werken de *push notifications* op zowel iOS als Android, wat weer een groot voordeel is voor React Native ten opzichte van de Angular PWA. Android is het enige besturingssysteem dat meldingen van een PWA kan ontvangen, want web notificaties worden nog niet ondersteund in de laatste versie van iOS. *Support* voor deze functionaliteit is wel gepland door Apple in de nabije toekomst.

Performance

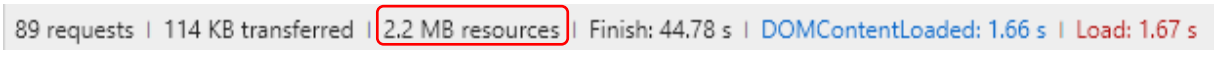


Figuur 40 – Framerate React Native

Figuur 41 – Framerate Angular PWA

Om de *performance* van zowel de PWA als de PoC te kunnen meten zijn er een aantal *tools* beschikbaar, deze *tools* laten toe om een meting te weergeven van de *framerate* oftewel FPS (Frames Per Second). De *tool* die gebruikt is bij React Native zit ingebouwd in Expo en bevat een *performance profiler,* deze toont een constante 60fps bij normaal gebruik van de applicatie, wat de aangeraden framerate is voor een *native app*. Ter vergelijking loopt de Angular PWA in de browser ook tegen ongeveer 60fps. Op vlak van *framerate* is er dus weinig verschil tussen beide oplossingen, maar een native app geeft nog altijd een vlottere ervaring ten opzichte van *web based* applicaties.

Data usage

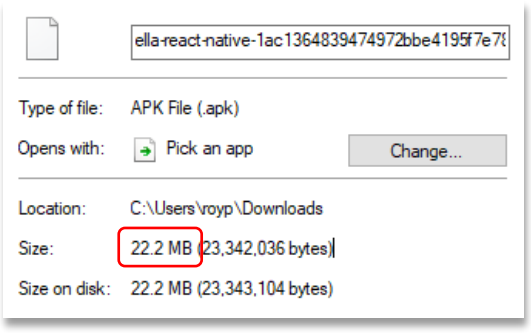


Figuur 42 – Dataverbruik Angular PWA (na force refresh)

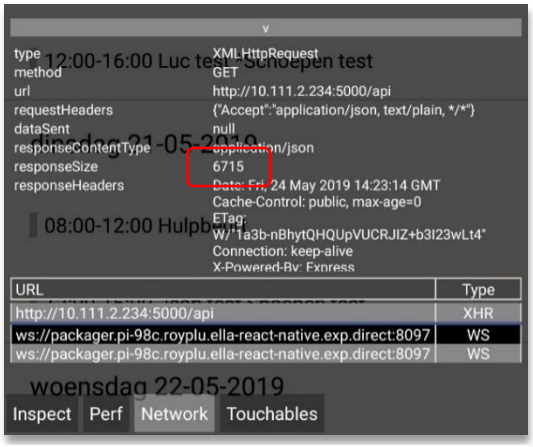
Figuur 43 – Dataverbruik ophalen afspraken (Angular PWA)

Ook op vlak van dataverbruik is er een vergelijking opgesteld. Deze metingen zijn gedaan om aan te tonen welke van beide applicaties meer geheugen verbruikt, dit kan bijvoorbeeld een grote impact hebben op de uiteindelijke kosten van een gsm-abonnement van een verzorgende. Aangezien een verzorgende op verplaatsing werkt, zal deze niet altijd verbinding hebben met Wi-Fi en mogelijk mobiele data moeten aanzetten. De al eerder aangehaalde tools van React Native en de PWA kunnen dit ook opmeten.

Aangezien een PWA *web based* is, komt er geen voorafgaande installatie van een app aan te pas, dit is ook een groot voordeel van een PWA. Aan de metingen is te zien dat er in totaal 2.2MB aan *resources* opgehaald wordt om de PWA te laten functioneren. Hiervan is maar 6.7KB van het ophalen van de afspraken van een week. Dit betekent dat de rest (2.193MB) bedoeld is voor de vele iconen, layout en styling op te halen voor de PWA, die deze (voor een bepaalde tijd) in de service worker cache opslaat.



Figuur 44 - Bestandsgrootte React Native PoC



Figuur 45 - Dataverbruik ophalen afspraken (React Native)

Dit is niet het geval bij React Native, aangezien er een build moet gemaakt worden van de applicatie en de .APK *file* (.IPA voor iOS) een bepaalde grootte inneemt. Bij het *builden* van de PoC is de uiteindelijke bestandsgrootte 22.2MB (Kan mogelijk nog verkleind worden). Bij het gebruik van de PoC haalt deze dezelfde gegevens op (een week van afspraken) als die van de PWA, deze komen overeen in grootte (responseSize = 6715 → 6.7KB).

Met deze metingen kan er vanuit worden gegaan dat de PoC wel meer geheugen vergt om de applicatie eerst te downloaden en geïnstalleerd te hebben op het apparaat, maar hierna maar kleine *API-calls* moet doen om een week van afspraken op te halen, wat veel minder data vergt. Dit in tegenstelling tot de PWA, die de volledige applicatie geregeld terug gaat moeten ophalen (op iOS is er een tijdslimiet van 2 weken om de applicatie gecached te houden), ook staat er standaard een bestandsgroottelimiet van 52MB op de cache van een PWA. [26] Het is mogelijk dat na langer gebruik van beide applicaties de React Native PoC veel minder gaat gebruiken over tijd dan de PWA.

State management

```
export const GET_AFSPRAKEN = "ella-react-native/repos/LOAD";
export const GET_AFSPRAKEN_SUCCESS = "ella-react-native/repos/LOAD_SUCCESS";
export const GET_AFSPRAKEN_FAIL = "ella-react-native/repos/LOAD_FAIL";

export default function reducer(state = { afspraken: [] }, action) {
  switch (action.type) {
    case GET_AFSPRAKEN:
      return { ...state, loading: true };
    case GET_AFSPRAKEN_SUCCESS:
      return { ...state, loading: false, afspraken: action.payload.data };
    case GET_AFSPRAKEN_FAIL:
      return {
        ...state,
        loading: false,
        error: "Error while fetching afspraken"
      };
    default:
      return state;
  }
}

export function listAfspraken(date) {
  return {
    type: GET_AFSPRAKEN,
    payload: {
      request: {
        url: `/getweek?jaar=${date.year}&maand=${date.month}&dag=${date.day}`
      }
    }
  };
}
```

Figuur 47 - State management bij React Native

```
export function reducer(state = initialState, action: AfsprakenActions): AfsprakenState {
  switch (action.type) {
    case AfsprakenActionTypes.GET_AFSPRAKEN:
      return {
        ...state,
      };
    case AfsprakenActionTypes.GET_AFSPRAKEN_SUCCESS:
      return { ...state, afspraken: action.payload.afspraken };
    case AfsprakenActionTypes.BEVESTIG_AFSPRAAK_SUCCESS:
      return {
        ...state,
        afspraken: (Object.values(state.afspraken) as AfspraakViewModel[]).map(c => {
          if (action.payload.nr === c.nr) {
            return action.payload;
          }
          return c;
        }),
        openAfspraken: state.openAfspraken.filter(nr => nr !== action.payload.nr),
      };
    case AfsprakenActionTypes.OPEN_AFSPRAAK: {
      if (!state.openAfspraken.includes(action.payload)) {
        return { ...state, openAfspraken: [...state.openAfspraken, action.payload] };
      }
      return { ...state, openAfspraken: state.openAfspraken.filter(nr => nr !== action.payload) };
    }
    default:
      return state;
  }
}
```

Figuur 46 - State management bij Angular PWA

State management is ook aanwezig in de PoC, deze werkt nagenoeg hetzelfde als de PWA en maakt gebruik van dezelfde concepten als NgRx zoals de Actions en Reducers.

```
render() {
  return (
    <Provider store={store}>
      <View style={styles.container}>
        <StatusBar hidden />
        <AppContainer />
      </View>
    </Provider>
  );
}
```

Figuur 48 - Provider React Native

De *<provider>* tag maakt de store toegankelijk in elk genest component in de *<AppContainer />*. Hierdoor kan elk component aan de gegevens in de store na deze te verbinden met de state en de actions, gebruikmakend van *mapStateToProps* en *mapActionsToProps*.

Aangezien NgRx een veelgebruikte library is binnen Ons en ze dit in elke webapplicatie integreren is deze feature een *must-have* om React Native als een waardig alternatief te kunnen zien.

```
const mapStateToProps = state => {
  if (state.afspraken.data) {
    let storedAfspraken = state.afspraken.data.afspraken.map(afspraak => ({
      key: `${afspraak.nr}`,
      ...afspraak
    }));

    let storedKlanten = state.afspraken.data.klanten.map(klant => ({
      key: `${klant.nr}`,
      ...klant
    }));
```

Figuur 49 - mapStateToProps React Native

Distribution

De distributie van apps ligt op dit moment nog heel verschillend bij de 2 technologieën die vergeleken zijn. Native apps zoals de PoC kunnen gemakkelijk verspreid worden naar nieuwe gebruikers door middel van zowel de Google play store als de Apple app store. Dit is ook de meest voorkomende manier om een app te distribueren.

Over de laatste paar maanden is er een grote vooruitgang geweest op het distribueren van PWA's, zo is het recent mogelijk gemaakt om in de Google play store deze op dezelfde manier te verspreiden als een native app.

Figuur 50 - PWA vs. Apple

Vooral bij Apple was dit nog een probleem, voor een lange tijd was er helemaal geen ondersteuning voor PWA's, zowel op vlak van functionaliteit als distributie in de app store. Deze situatie is stilaan het verbeteren en de toekomst van PWA's op beide platformen ziet er goed uit.

2.2 Conclusie

Criteria	Angular PWA	React Native
Soort app	Progressive Web App	Mobile App
Store management	Store management met NgRx	Store management met Redux
Implementatie om view te weergeven	WebView	Gecompileerde Native code
Notificaties	Push notificaties alleen mogelijk op Android	Push notificaties mogelijk op beide platformen
Performantie	~60FPS	~60FPS
Dataverbruik	Initieel dataverbruik lager (2.2MB), uiteindelijk dataverbruik hoger over tijd	Initieel dataverbruik hoger (22MB), uiteindelijk dataverbruik lager over tijd
Distributie	Distributie naar beide stores mogelijk door recente veranderingen	Distributie naar beide stores mogelijk
Authenticatie	Geen vingerafdrukscanner functionaliteit	Vingerafdrukscanner functionaliteit aanwezig

Camera	Hangt af van *device* tot *device* (Android, iOS)	Toegang tot camera mogelijk

Tabel 2 - Vergelijking resultaten Angular vs. React Native

Was React Native de beste keuze?

Uit de gemeten gegevens en kijkend naar de opgestelde probleemstelling kan de conclusie worden gevormd dat React Native een fantastische keuze zou zijn om het stageproject in te ontwikkelen. De vele voordelen die React Native met zich meebrengt kunnen niet geëvenaard worden aan de huidige staat van PWA's, die nog een aantal belangrijke functionaliteiten mist op technisch vlak, zoals push notificaties, volledige camerafunctionaliteit en authenticatie aan de hand van native identificatietechnologieën, om *feature complete* te zijn.

De mogelijkheid om de React Native app exact hetzelfde te stylen als de Angular PWA is ook aanwezig, alle elementen om de layout van de UI prototypes in na te bouwen bestaan in React Native.

Ook de instap van Angular naar React Native verliep vlot, aangezien beide frameworks op meerdere vlakken met elkaar overeenkomen bij zowel *state management* als hoe de layout en styling wordt opgesteld. Ook de transitie van TypeScript naar standaard JavaScript verliep probleemloos en kan zeer interessant zijn voor een bedrijf als Ons, aangezien een groot deel van de code die geschreven is voor het stageproject herbruikbaar is in React Native. Dit kan voor Ons heel aantrekkelijk zijn om in de toekomst te gebruiken voor zowel hun bestaande projecten als hun nieuwe projecten.

Welk framework is het best voor de ontwikkeling van de gevraagde app?

Bij het vergelijken van de 2 frameworks is het duidelijk dat React Native het beste framework is als het doel is om een app te ontwikkelen die alle functionaliteiten van een smartphone kan benutten, de *native feel* kan behouden en altijd ondersteund zal worden door de 2 populairste *app stores*.

Ten voordele van PWA's kan gezegd worden dat er recent grote en belangrijke veranderingen zijn gemaakt (vooral aan Apple's kant) om deze te ondersteunen. Het gemak van in een web framework te werken en deze dan op elk mogelijk platform uit te kunnen brengen is een grote troef en zal de komende paar maanden en jaren nog versterkt worden door meer ondersteuning en toegevoegde functionaliteiten.

Op dit moment kan er nog steeds besloten worden dat PWA's nog in de kinderschoenen staan en de voordelen van een *native app* nog niet kan overtreffen.

Welke andere niet-onderzochte frameworks zouden ook het onderzoeken waard zijn?

Naast React Native en PWA technologieën zijn er nog een aantal andere technologieën die een vergelijkbaar resultaat kunnen opleveren. Aangezien beide frameworks "*Cross-platform*" werken kan de focus gelegd worden op de overblijvende mogelijkheden die ook op zowel Android als iOS werken. Zo is er Nativescript, een *open source framework* dat ontwikkelaars toelaat om Native iOS en Android applicaties te bouwen met Angular, Vue.js of standaard JavaScript. Dit framework is zeer vergelijkbaar met React Native, maar mist nog een grote *community* van *developers* om goede ondersteuning te hebben in het "*Cross-platform*" applicatieontwikkelingslandschap.

Ten tweede is er ook nog Flutter. Flutter is een nieuw en opkomend *open source framework*, dat gebruik maakt van Google's programmeertaal genaamd "Dart". Net zoals React Native kan het framework gebruikt worden voor de ontwikkeling van applicaties voor beide populaire mobiele besturingssystemen. Flutter is redelijk nieuw in het mobile development landschap, waardoor het nog een aantal beperkingen heeft op moment van schrijven. Niettemin krijgt Flutter meer en meer tractie bij ontwikkelaars en worden er geregeld nieuwe updates vrijgegeven. Een verdere beschrijving van Flutter en de uitwerking van een PoC is uitgeschreven in het eindwerk van medestudent Aïmane Najja, die zijn onderzoeksopdracht rond dit specifiek framework heeft gebaseerd.

2.3 Aanbevelingen

Een volgende logische onderzoeksvraag zou zijn welke van beide onderzochte frameworks (React Native en Flutter) de beste optie zou zijn moest Ons verder willen met een cross-platform native technologie. Een vergelijking tussen deze twee kan opgesteld worden aan de hand van de eerste vergelijkingen die gemaakt zijn ten opzichte van de PWA.

De uiteindelijke resultaten hiervan die zeer relevant zijn voor Ons kan men beperken tot hoeveel werk een applicatie, gemaakt in een van deze frameworks, zou kosten, hoe hoog de herbruikbaarheid van bestaande code ligt en welke native functionaliteiten het framework ondersteund.

2.4 Reflectie

Bij het bedenken van een onderzoeksopdracht ben ik eigenlijk meteen op het idee gekomen om een mobiel framework te vergelijken met het project dat tijdens de stageopdracht ontwikkeld is. Na wat onderzoek te doen en zelf vergelijkingen op te stellen ben ik gestrand op mijn keuze van React Native en heb ik hier zeker geen spijt van gehad. De flexibiliteit die het framework me gaf tijdens het ontwikkelen van de PoC heeft z'n vruchten afgeworpen en gaven me de mogelijkheid om alle nodige functionaliteiten gemakkelijk toe te voegen. Ook het gebruik van Expo maakte de ontwikkeling een stuk aangenamer door de vele API's en *hot reloading*.

Uiteindelijk kan ik zeggen dat wat ik wou bereiken met dit onderzoek en de vergelijking van de PoC met de Angular PWA geslaagd is, ik heb op elk van de criteria een antwoord kunnen vormen en dit verder kunnen toelichten om te bewijzen dat React Native een waardig alternatief kan zijn voor Ons om toekomstige projecten mee te ontwikkelen.

Conclusie

Na de ontwikkeling van zowel het Ell@-project als de onderzoeksopdracht kan er geconcludeerd worden dat beide succesvol zijn afgerond, alle gevraagde onderdelen die af moesten geraken binnen de 12 stageweken zijn aanwezig en zijn klaar voor productie.

Feedback van de verzorgenden van de Landelijke Thuiszorg hebben we jammer genoeg nog niet kunnen krijgen tijdens de stageperiode, de bedoeling is om kort na de stageperiode een aantal verzorgenden met de huidige versie van Ell@ al op het werkveld te zetten. Het uitbrengen van de volledige Ell@ app zelf staat gepland voor 2020, aangezien er nog meerdere modules in de toekomst toegevoegd gaan worden.

Ook de onderzoeksopdracht is succesvol tot stand gekomen met de gevraagde *features* vanuit Ons en zal een grote meerwaarde zijn voor het stagebedrijf op vlak van hun toekomstig gebruik van mobiele technologieën.

Bibliografie

[1] Cegeka, "Cegeka," [Online]. Available: https://www.cegeka.com/en/be/.

[2] Cegeka, "About us - Cegeka," [Online]. Available: https://www.cegeka.com/nl-nl/over-ons.

[3] ONS, "Visie van ONS," [Online]. Available: http://www.ons.be/Visie/tabid/563/language/nl-NL/Default.aspx.

[4] Cegeka, "Cegeka - Annual report 2017," [Online]. Available: https://annualreport.cegeka.com/.

[5] "Wikipedia - Angular," [Online]. Available: https://en.wikipedia.org/wiki/Angular_(web_framework).

[6] "Wikipedia - Angular (nl)," [Online]. Available: https://nl.wikipedia.org/wiki/Angular.

[7] "NgRx Docs," [Online]. Available: https://ngrx.io/docs.

[8] "Wikipedia - ASP.NET," [Online]. Available: https://en.wikipedia.org/wiki/ASP.NET.

[9] V. Gagliardi, "Valentino G blog - Cypress," [Online]. Available: https://www.valentinog.com/blog/cypress/.

[10] "Wikipedia - Visual Studio Code," [Online]. Available: https://en.wikipedia.org/wiki/Visual_Studio_Code.

[11] "Wikipedia - Microsoft Visual Studio," [Online]. Available: https://nl.wikipedia.org/wiki/Microsoft_Visual_Studio.

[12] Microsoft, "Azure Microsoft," Microsoft, [Online]. Available: https://azure.microsoft.com/nl-nl/services/devops/.

[13] R. A. Planet, "Android Planet - Slack," Android Planet, [Online]. Available: https://www.androidplanet.nl/spotlight/slack/.

[14] S. Brown, "C4 Model," [Online]. Available: https://c4model.com/.

[15] B. B.V., "Sketch," [Online]. Available: https://www.sketch.com/.

[16] "Wikipedia - Pair programming," [Online]. Available: https://nl.wikipedia.org/wiki/Pair_programming.

[17] "Wikipedia - Code Review," [Online]. Available: https://en.wikipedia.org/wiki/Code_review.

[18] NgRx Community, "NgRx guide," [Online]. Available: https://ngrx.io/guide/store.

[19] B. Wagener, "S&D Interactive Media," [Online]. Available: https://www.sdim.nl/blog/wat-zijn-progressive-web-apps/.

[20] D. Ward, "Gitconnected - Expo vs React Native CLI," 25 September 2018. [Online]. Available: https://levelup.gitconnected.com/expo-vs-react-native-cli-a-guide-to-bootstrapping-new-react-native-apps-6f0fcafee58f.

[21] Y. Kruhlyk, "Apiko - Expo vs Vanilla React Native," 4 December 2018. [Online]. Available: https://apiko.com/blog/expo-vs-vanilla-react-native/.

[22] V. Immonen, "Expo.io Blog - Expo CLI 2.0 release," 12 September 2018. [Online]. Available: https://blog.expo.io/expo-cli-2-0-released-a7a9c250e99c.

[23] "Stackoverflow - Difference between ExpoKit and React Native project," 16 Maart 2018. [Online]. Available: https://stackoverflow.com/questions/49328412/difference-between-expokit-and-react-native-project.

[24] W. Ancheta, "Envatotuts+ - Detaching Expo Apps to ExpoKit: Concepts," 14 Maart 2018. [Online]. Available: https://code.tutsplus.com/tutorials/detaching-expo-apps-to-expokit-concepts--cms-30661.

[25] Vaadin Ltd., "Vaadin - Push notifications," [Online]. Available: https://vaadin.com/pwa/learn/push-notifications.

[26] C. Love, "Love2Dev," 11 03 2019. [Online]. Available: https://love2dev.com/blog/what-is-the-service-worker-cache-storage-limit/.