Professionele Bachelor Toegepaste Informatica

Ell@ 2.0, de nieuwe eerstelijnsassistente voor verzorgenden

Aïmane Najja

Promotoren:

Bert Brouns
Kris Hermans

Cegeka - ONS
Hogeschool PXL

Bachelorpaper Academiejaar 2018-2019

Professionele Bachelor Toegepaste Informatica

Ell@ 2.0, de nieuwe eerstelijnsassistente voor verzorgenden

Aïmane Najja

Promotoren:

Bert Brouns
Kris Hermans

Cegeka - ONS
Hogeschool PXL Hasselt

Bachelorpaper Academiejaar 2018-2019

Dankwoord

Alle lof voor iedereen die mij doorheen mijn jaren aan de Hogeschool PXL gesteund, geholpen en gemotiveerd heeft.

Te beginnen bij mijn promotors. Zij stonden in voor het in goede banen leiden van mijn stage. Ik was oprecht blij toen ik voor aanvang van de stage te weten kwam dat meneer Hermans mij zou begeleiden bij mijn stage. Meneer Hermans stelt mij gerust maar weet mij ook van constructieve feedback te voorzien en dat stel ik zeer op prijs.
Hetzelfde geldt voor meneer Brouns. Hij geeft met zijn gedrevenheid, zijn passie en zijn expertise het perfecte voorbeeld waar wij als stagiairs naar moeten streven en laat daarnaast ook regelmatig weten tevreden te zijn over de voortgang en vertrouwen te hebben in de capaciteiten van ons als stagiairs.

Daarnaast wil ik ook mijn collega's bij ONS bedanken. Vanaf de eerste dag gaven ze me het gevoel dat ik een volwaardige collega was en dat mijn werk alsook mijn aanwezigheid gewaardeerd werd. Ik ben Cegeka ook zeer dankbaar dat ze de opdracht ter beschikking hebben gesteld. De applicatie waaraan ik heb mogen werken, zal worden gebruikt door een grote groep mensen en het streelt een programmeur als zijn werk veel wordt gebruikt. Ik mag mezelf gelukkig prijzen dat ik daarvoor de gelegenheid en de eer heb gekregen van Cegeka.

De medestudenten die in hetzelfde bootje zijn gestapt aan het begin van de opleiding en die tijdens het avontuur mijn vrienden zijn geworden, verdienen het ook om bedankt te worden. Zij zorgden voor een goede balans tussen plezier, goede resultaten en blijvende motivatie waarmee we streefden naar een gezamenlijk doel, namelijk het behalen van het diploma Professionele Bachelor Toegepaste Informatica.

De lectoren die mij de afgelopen jaren onderwezen hebben, zou ik graag één voor één willen bedanken omdat ze ook allen zelf het goede voorbeeld hebben gegeven. They walk the talk! Op die manier hebben ze er, althans in mijn ogen, voor gezorgd dat wij als studenten door hun sympathie gedreven bleven.

Tenslotte, maar eigenlijk in de eerste plaats, wil ik mijn moeder bedanken. Ik zou haar te kort doen als ik in een paar zinnen kon beschrijven waarom ik haar dankbaar ben.

Abstract

De eerstelijnsassistente, Ell@, die in deze stageopdracht ontwikkeld werd voor Landelijke Thuiszorg, is een progressieve webapplicatie. Dit betekent dat de verzorgenden Ell@ als een mobiele applicatie op hun smartphone kunnen downloaden en daar gemakkelijk ook offline gebruik van kunnen maken.

Ell@ werd in het leven geroepen omdat diezelfde verzorgenden en hun collega's van de administratie hun handen vol hadden en veel tijd verloren in herhalend papierwerk. Er moest een oplossing gevonden worden om de herhaling te automatiseren en eventueel nog meer.

Ell@ kan nu automatisch, real-time, supersnel, altijd en overal de planning tonen van de verzorgende. De verzorgenden kunnen, indien nodig, een afspraak aanpassen en bevestigen. Te bedenken dat men dit voorheen mondeling, schriftelijk, telefonisch, per mail of per sms deed, is Ell@ achteraf een zeer degelijke verlichting.

De applicatie werd gerealiseerd met behulp van het Angular-framework waarbij gebruikgemaakt werd van het NgRx-library om de staat van de applicatie bij te houden en te beheren in de frontend. De backend is volledig aan C# toevertrouwt om onder andere de databases aan te spreken en hieruit de juiste data te verkrijgen, te verwerken en vervolgens te versturen naar de frontend.

Naast het ontwikkelen van de applicatie, is er tijdens deze stage ook onderzoek gedaan naar het Flutter-framework. Dit framework zorgt ervoor dat ook een aantal native features zoals het scannen van de vingerafdruk en het ontvangen van notificaties aangeroepen kunnen worden zonder daarbij de voordelen van een PWA kwijt te spelen.

Inhoudsopgave

Lijst van gebruikte figuren

Lijst van gebruikte tabellen

Lijst van gebruikte afkortingen

API	Application Programming Interface
ASP	Active Server Pages
C4	Context, Container, Component en Code
eID	Electronic Identification Device
Ell@	Eerstelijnsassistente
FCM	Firebase Cloud Messaging
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
IT	Information Technology
NgRx	Angular Redux
OOP	Objectgeoriënteerd Programmeren
PoC	Proof of Concept
PWA	Progressive Web App
SSO	Single Sign-On
VM	Virtual Machine
VS	Visual Studio

Inleiding

Er is voor deze stage een duidelijk doel voor ogen: het ontwikkelen van een mobiele applicatie voor de verzorgenden in de thuiszorg ter vervanging van de mobiele apparaatjes die voorheen het papierwerk al moesten vervangen maar daarin naar efficiëntie toe tekortschoten.

Het maken van de applicatie is in opdracht van Cegeka dat hiervoor samenwerkt met Ons. De applicatie, genaamd Ell@, komt toe aan Landelijke Thuiszorg dat, naast nog vele andere zorgbedrijven, deel uitmaakt van Ons.

Mede door het feit dat Cegeka in nauw contact staat met de Hogeschool PXL, heeft Cegeka voor vele PXL-IT-studenten een hoog aanzien gecreëerd en wordt ze beschouwd als dé hoofdspeler in de IT-sector. Met dat in het achterhoofd en deze interessante, uitdagende opdracht die Cegeka ter beschikking stelde, is er over deze stage niet lang getwijfeld.

Met Ell@ kunnen de verzorgenden op gemakkelijke en geheel intuïtieve wijze hun planning raadplegen en na iedere afspraak de bijhorende details wijzigen en bevestigen. Ell@ zou de verzorgenden ook moeten toelaten om afwezigheden te wettigen maar dat is voor deze stage buiten de scope gerekend. De stage beperkt zich en focust zich op het uitbrengen van het prototype van het planningaspect.

De technologieën waarmee men Ell@ onder handen neemt, zijn vooraf vastgelegd door het stagebedrijf. In de backend steunt men op C# (ASP.NET Web API 2) om de applicatie van de juiste data te voorzien en in de frontend berust men op Angular 7 dat m.b.v. NgRx de staat van de applicatie onderhoudt.

Interessant om te onderzoeken voor Cegeka en Ons is een alternatieve frontendtechnologie die bepaalde functionaliteiten kan aanroepen, waar Angular beperkt in is, zoals de vingerafdruk scannen en notificaties ontvangen bijvoorbeeld. In deze paper is Flutter dat alternatief en belooft het naar eigen zeggen volledig te voldoen aan de verwachtingen.

I. Stageverslag

1 Bedrijfsvoorstelling

1.1 Situering stagebedrijf

Naast kantoren in Hasselt, Leuven, Antwerpen en Gent in België heeft Cegeka zich op het moment ook in verscheidene andere landen gevestigd. Duitsland heeft met een aantal van zes kantoren het hoogste aantal Cegeka-vestigingen. En zo mogen Nederland, Tsjechië, Slowakije, Oostenrijk, Italië en Roemenië ook gerust spreken over een IT-invloed van Cegeka. [1]

Cegeka ziet dat veel organisaties in de bedrijfswereld een groeiende nood aan IT-partners hebben die hen helpen de efficiëntste stappen te zetten zonder zich te laten beïnvloeden door de digitalisering en haar hypes. Er mag gezegd worden dat Cegeka degelijk zo'n IT-partner is. Met bijvoorbeeld Transavia in Nederland en Lufthansa in Duitsland, die in 2017 de samenwerking met Cegeka verlengd hebben, heeft Cegeka van aanzienlijke bedrijven het vertrouwen in haar IT-expertise gewonnen. [2]

Figuur 1: Cegeka in Europa

1.2 Situering stageafdeling

Cegeka biedt haar partners IT-expertise aan in de vorm van Softwareontwikkeling, Support & Helpdesk, Infrastructuur en Business Consultancy. [3] Deze stage vond plaats in de afdeling Softwareontwikkeling en heeft de stagiair meer bepaald aan de slag laten gaan en verder opgeleid als software developer. Met oog op mogelijke indienstname kent Cegeka aan elke stagiair een collega toe die hen begeleidt en op de voet volgt om de codekwaliteit te bewaken en het werktempo hoog te houden.

Figuur 2: Cegeka-aanbod IT-expertise

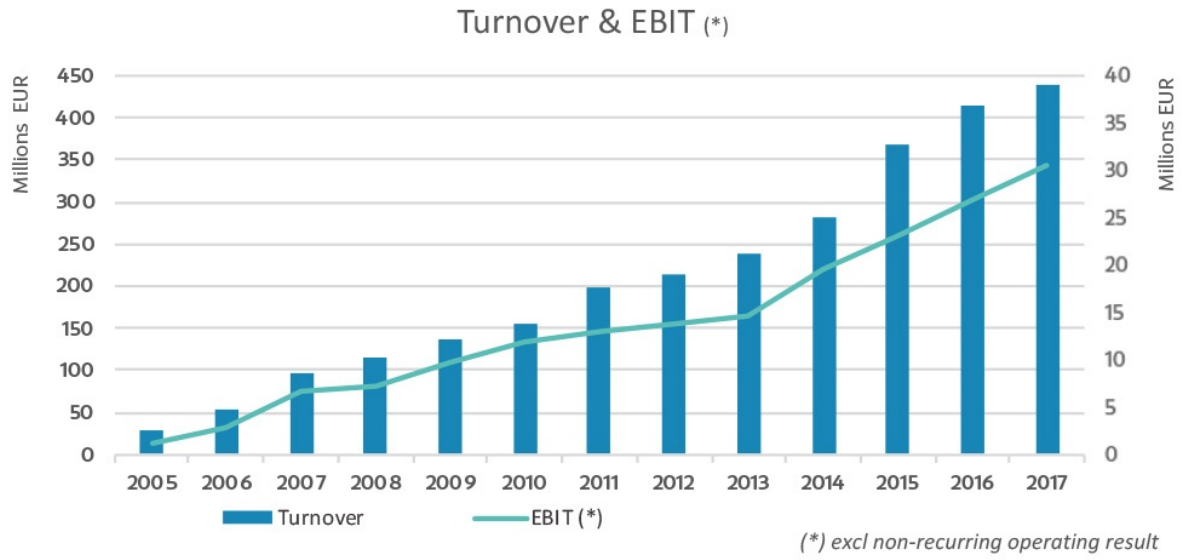
1.3 Unique sellingpoints

Naar eigen zeggen is Cegeka zich als geen ander bewust dat de IT-wereld vooral draait om cijfers maar streeft ze er ook naar om op een menselijke manier een tastbaar verschil te maken bij het opleveren van resultaten. Cegeka hecht even veel waarde aan soft skills, bijvoorbeeld het oor hebben naar de zakelijke pijnpunten van klanten, als aan kennis en expertise. Als werkgever verwacht Cegeka dat haar werknemers naast het beschikken over een aantal eigenschappen zoals ondernemerschap en respect, vooral uit hun comfortzone durven treden als het gaat om het oplossen van problemen van klanten.

Verder hanteert Cegeka in het ontwikkelen van applicaties de Agile methodologie. Projecten worden bij aanvang met een uitgebreide analyse aangepakt waarbij er met een kritische blik naar de vereisten wordt gekeken om verspilling te vermijden. Er wordt gewerkt in sprints van telkens twee weken en om het ontwikkelen te versnellen worden de build-, test- en deploy-processen doorgaans geautomatiseerd via Azure DevOps.

1.4 Cijfers

In 2017 behaalde Cegeka een omzet van 440 miljoen euro wat een groei van 6,3% betekende. Cegeka kan rekenen op het vertrouwen van meer dan 2500 tevreden klanten en daar zijn samen meer dan 4200 werknemers verantwoordelijk voor. [4]



Figuur 3: Omzet Cegeka per jaar

2 Voorstelling stageopdracht

2.1 Probleemstelling

Het Ell@-traject, dat enkele jaren geleden van start is gegaan, had al in zijn eerste iteratie het doel voor ogen om het toenmalige proces te optimaliseren. De administratie gebeurde nog volledig op papier en dit kan naarmate een bedrijf groeit, ook een groei aan administratie betekenen met daaraan de nodige kosten. Ook werkte het thuiszorgpersoneel tot voor kort uitsluitend op basis van mondelinge of handgeschreven communicatie. Dit betekende dus dat het eerstelijnspersoneel hun eigen planning neer moesten schrijven op het F53-formulier (om geleverde prestaties in de gezinszorg te melden).

De thuiszorgsector staat in direct contact met kansengroepen die dreigen buiten de digitale samenleving te vallen (ouderen, kansarmen, laaggeschoolden, anderstaligen, ...) en heeft op die manier een krachtig middel in handen om deze kritieke doelgroep toegang te laten vinden tot de digitale wereld. Hiermee slaat Ell@ een interessante, extra vlieg mee in één klap.

De eerste versie van Ell@, oftewel Ell@ 1.0, bestond uit een mobiel elektronisch toestel waarop onder andere de planning van de verzorgende geraadpleegd kan worden en de patiënt de geleverde prestaties elektronisch kan bevestigen. [5] De toestellen lijken jammer genoeg sinds een Windows 10-update vast te lopen en dit is enkel te verhelpen door nieuwe toestellen aan te schaffen. Maar deze moeten in veel te grote aantallen besteld worden, veel meer dan het aantal zorgpersoneelsleden.

Figuur 4: Ell@ 1.0

Aangezien de wetgeving voorheen besliste dat enkel het gebruik van de eID-kaart in combinatie met de pincode rechtsgeldig was, zag Landelijke Thuiszorg zich genoodzaakt te blijven werken met de F53-formulieren. Te weinig cliënten onthouden hun pincode om van het formulier te kunnen afstappen en volledig te kunnen overschakelen naar het gebruik van de Ell@-toestellen. [6]

Echter een verandering in de wetgeving heeft ervoor gezorgd dat een handtekening van de klant niet meer verplicht is op voorwaarde dat op de factuur elke geleverde prestatie apart vermeld wordt met minimaal de volgende gegevens: de datum, het begin- en einduur of duur, het type zorg (gezinszorg, poetshulp, karweihulp of professionele oppashulp) en de gebruikersbijdrage. In dat geval is een factuur die betaald werd door de gebruiker, het bewijs dat hij akkoord gegaan is met elk van de prestaties die vermeld worden op die factuur.

2.2 Doelstellingen

Die verandering in de wetgeving betekende het startsignaal van Ell@ 2.0. Nu er geen handtekening bij iedere prestatie en dus ook geen toestel voor Ell@ om de identiteitskaarten te lezen meer nodig is, kan men afstappen van de F53-formulieren en de Ell@-toestellen en is er bijgevolg beslist om van Ell@ een mobiele webapplicatie te maken.

Het doel is om de gebruikerservaring van Ell@ te bevorderen zodat er geen opleidingen of uitgebreide handleidingen meer voorzien moeten worden om de werking van Ell@ uit te leggen. Ook hoeft het managementpersoneel niet meer keer op keer mondeling, schriftelijk, telefonisch, per mail

of per sms de planningen door te communiceren na iedere aanpassing want Ell@ 2.0 werkt, in tegenstelling tot Ell@ 1.0, real-time.

Geen formulieren betekent voor het administratief personeel ook minder papierwerk waardoor men tijd wint om meer van andere taken gedaan te krijgen. Dit laatste is vooral voor de werkgever interessant en daarom is Ell@ 2.0 juist zo gegeerd.

2.3 Technologieën

2.3.1 Visual Studio 2017

Visual Studio is een integrated development environment (IDE) ontwikkeld door Microsoft. Visual Studio ondersteunt naast C# nog tientallen andere programmeertalen zoals C, C++, VB.NET, F# en TypeScript. Dit zijn voorbeelden van de ingebouwde programmeertalen. Apart kan er ook ondersteuning voor andere talen zoals Python, Ruby en Node.js geïnstalleerd worden. Daarnaast biedt Visual Studio ook ondersteuning voor *markup languages* als XML en HTML en is CSS daarbij ook gewoon mogelijk. Java wordt enkel via diensten van derden ondersteund in Visual Studio.

Figuur 5: Logo Visual Studio

Er zijn verschillende edities van Visual Studio beschikbaar: de Community, de Professional, de Enterprise, de Test Professional en de Express-editie. De Community-editie is de meest gebruikte editie door studenten en freelancers en deze is gratis, maar ook de meest eenvoudige. De Professional-editie bevat extra functionaliteiten en is de goedkoopste commerciële editie van Visual Studio. De Enterprise-editie is naast de functionaliteiten van de Professional nog aangevuld met een set tools voor softwareontwikkeling, databaseontwikkeling, samenwerking, statistiek, architectuur, testen en rapportering. De overige edities worden minder gebruikt of bestaan zelfs niet meer. [7]

De eerste keer dat Visual Studio een geheel product leverde, dat vele afzonderlijke programmeertools bundelde, dateert van 1997. Ondertussen zit men aan Visual Studio 2019 dat men op 2 april 2019 heeft uitgebracht. [8]

Voor deze stageopdracht is er gebruikgemaakt van Visual Studio 2017 om vanuit de backend de applicatie te voorzien van de nodige gegevens in de frontend. Dat is een vrij voor de hand liggende keuze die gemaakt is, omdat zowel C# als Visual Studio producten zijn van Microsoft en om die reden is het C#-programmeren in Visual Studio ook geoptimaliseerd. Voorzieningen zoals IntelliSense zijn tegenwoordig zó ver gevorderd dat programmeurs tegenwoordig maar een aanzet hoeven te doen en IntelliSense in de meeste gevallen de juiste suggesties weergeeft voor uit te voeren taken of aan te vullen code.

Daarbovenop is de Ell@-applicatie slechts een onderdeel van een reeds lopend project binnen Ons. Daarom dat voor de frontend-API er vele stukken code konden worden hergebruikt die elders door andere Ons-teams al ontwikkeld werden. Voor code die door meerdere teams hergebruikt wordt, maakt men binnen Ons een NuGet-package zodat deze maar hoeft toegevoegd te worden aan het Visual Studio-project en meteen gebruikt kan worden zonder de code handmatig te moeten kopiëren. Ook hier bestaat naast een console ook een userinterface (UI) voor binnen Visual Studio. In feite heeft Ons zelfs een eigen *package source* waar programmeurs van Ons gemakkelijk de NuGet-packages in kunnen vinden.

2.3.2 ASP.NET WebAPI2

ASP.NET Web API is een framework dat HTTP-services bouwt die toegankelijk zijn voor verschillende applicaties op verschillende platformen zoals web- en desktopapplicaties en mobiele applicaties. In tegenstelling tot ASP.NET MVC verstuurt Web API data door in plaats van een webpagina. Ze maken wel op dezelfde wijze gebruik van controllers en routing.

Figuur 6: Logo ASP.NET Web API

Aangezien de frontend in Angular ontwikkeld wordt, volstaat het voor de Ell@-applicatie om enkel de API te schrijven in .NET en dan is er, gezien de .NET-ervaring in huis, gekozen deze API te ontwikkelen in ASP.NET Web API 2. [9]

2.3.3 Visual Studio Code

Visual Studio Code is een gratis en open-source broncode-editor ontwikkeld door Microsoft voor Windows, Linux en macOS. Dat is ook meteen het grote voordeel van VS Code: haar gebruik ongeacht welke besturingssysteem en ongeacht welke programmeertaal. Om die reden is VS Code ook zo populair tegenwoordig bij programmeurs. Volgens een peiling op StackOverflow is momenteel VS Code veruit de meest gebruikte ontwikkelomgeving. [10]

Figuur 7: Logo Visual Studio Code

Figuur 8: Populairste IDE's 2019

Zonder VS Code moesten programmeurs bijna voor elke programmeertaal een dure IDE aanschaffen, zoals bijvoorbeeld NetBeans, Eclipse of Android Studio. Deze zijn meestal groot en daarom vaak langzamer wegens het geheugengebruik. VS Code is ook nog eens makkelijker te installeren. In feite, er vindt zelfs geen installatie plaats. VS Code kan na het downloaden meteen gebruikt worden vanuit de downloadmap, in tegenstelling tot IDE's waarbij door een aantal stappen gelopen moet worden vooraleer ze gebruikt kunnen worden. VS Code is zó intelligent dat bestanden maar een extensie nodig hebben om automatisch van de juiste code-aanvullingsfunctie voorzien te worden.

VS Code is, slechts een aantal jaren na haar release, de meest gebruikte ontwikkelomgeving. Op 29 april 2015 werd VS Code aangekondigd door Microsoft en een jaar later op 14 april 2016 werd ze uitgebracht. [11] [12] In datzelfde jaar stond Visual Studio Code 13de gerangschikt in de statistieken van StackOverflow en nu dus eerste met bijna 20% voorsprong op haar concurrenten.

Alhoewel het bij Ons geen verplichting was om VS Code te gebruiken, werd het wel grotendeels door alle programmeurs om eerdergenoemde redenen gebruikt. De reden dat zowel Visual Studio als VS Code gebruikt worden, is dat beide interessante functies aanbieden. Visual Studio bevat veel UI's. Het koppelen van een project aan een proces is bijvoorbeeld een pluspunt van Visual Studio. Andere voordelen die Visual Studio biedt is de Team Explorer en de goede C#-ondersteuning. VS Code heeft een veel snellere en volledigere zoekfunctie, biedt makkelijk ondersteuning voor veel meer talen en is zoals gezegd veel lichter en sneller te gebruiken.

2.3.4 Angular

Figuur 9: Logo Angular

Angular is een frontendframework dat in het leven is geroepen omdat moderne websites programmeren een te complexe opgave is. Angular (en daarnaast ook andere frontendframeworks zoals React en VueJS) handelt allerlei taken af die nodig zijn om webapps in verschillende browsers te presenteren.

Angular is ontwikkeld in 2010 door Google en de eerste versie hiervan heette toen AngularJS. Vanaf 2016 werd AngularJS hernoemd naar Angular omdat Google het herschreven heeft en het niet meer als intern projectje wou blijven beschouwen. [13]

Angular is een *component based* framework. Dat wil zeggen dat elke webapplicatie die met Angular gebouwd is, een samenstelling is van opzichzelfstaande componenten. Dit geeft, naast het feit dat Angular ook een aantal *Design Patterns* met zich meebrengt, ook het programmeren meer structuur en op deze manier is het ook makkelijker om componenten te hergebruiken, wat dan ook het DRY-principe in de hand werkt. Deze componenten zijn gefocust op twee dingen, namelijk een template en haar data. De template is grotendeels HTML en CSS aangevuld met onder andere Angulars databinding die ervoor zorgt dat de verwerkte data gevuld wordt in de template. [14]

Zo biedt Angular nog tal van andere voordelen waardoor het de voorkeur geniet bij de programmeurs binnen Ons.

2.3.5 NgRx

Figuur 10: Logo NgRx

NgRx is een framework dat gemaakt is om de *state* in Angular-applicaties te managen. Een *state* is noodzakelijk in grote applicaties waarbij er veel dataverkeer is zodat onveranderde data niet iedere keer over HTTP moet worden opgehaald maar uit de *store* kan worden gehaald. De *store* bevat data die, bij het opstarten van de applicatie, opgehaald wordt over HTTP. Zolang er geen veranderingen gebeuren aan de data die de *store* in huis heeft, kan op elke pagina de data supersnel weergegeven worden. Dit is naar gebruiksvriendelijkheid toe zeker een aanwinst.

Het grootste probleem dat NgRx oplost, is het continu in sync houden van de data in de applicatie met de data uit de database. Dit zou Angular ook alleen kunnen afhandelen, maar dan zit men soms te veel tussen componenten onderling te communiceren en krijgt men een wirwar van *databinding*. Met NgRx is het mogelijk dat twee componenten, die niet direct gerelateerd zijn met elkaar en tóch dezelfde data moeten presenteren, alsnog onmiddellijk allebei gesynchroniseerd zijn met de data uit de databases. En het is exact om die reden dat er door Ons gekozen is om daar nuttig gebruik van te maken. NgRx is open source. Iedereen kan dus een bijdrage leveren aan dit framework en binnen Ons doet men dit ook. [15] [16]

2.3.6 Azure DevOps

Figuur 11: Logo Azure DevOps

Azure zegt op haar site het volgende over DevOps: “In DevOps worden de mensen, processen en technologieën samengebracht om klanten continu waarde te leveren. DevOps, een samenstelling van dev (development; ontwikkeling) en ops (operations; activiteiten), is een softwareontwikkelingspraktijk waarbij de ontwikkeling en IT-activiteiten worden gekoppeld.”. [17]

Processen die anders veel tijd in beslag nemen omdat afzonderlijke teams daarvoor verantwoordelijk waren, worden nu automatisch en continu afgehandeld door Azure DevOps. Hiermee wordt de oplevering van de software versneld en de kwaliteit van de software verbeterd omdat ze continu getest en gemonitord wordt waardoor de klanttevredenheid ook zal verhogen. Niet alleen de klanten, maar ook de programmeurs zijn tevreden als ze meer gedaan kunnen krijgen op kortere tijd en hun codekwaliteit daar niet onder lijdt. Reden te meer voor Ons om aan de slag te gaan met Azure DevOps.

2.3.7 Cypress

Figuur 12: Logo Cypress

Cypress is een framework om testen te schrijven voor JavaScript en dus ook voor Angular. Cypress kan gebruikt worden om zowel *end-to-end-,* als *unit-* en integratietesten te schrijven. Voor de Ell@-applicatie is Cypress enkel gebruikt om end-to-endtesten te schrijven. Bijvoorbeeld is er getest of er genavigeerd kan worden naar bepaalde componenten, of de data in die componenten correct is, of de data ook aangepast kan worden en of de aanpassingen worden doorgevoerd naar andere componenten.

Waar Cypress vooral in uitblinkt, is de mogelijkheid om te tijdreizen in het logvenster van de browser. Met de cursor hoveren over een tussenstap van een test geeft duidelijk weer wat er in die tussenstap gedaan wordt in de browser of er wordt duidelijk getoond wat er precies is foutgelopen. Ook zullen, elke keer dat er aanpassingen worden opgeslagen in de code van de tests, de testen opnieuw worden herladen en uitgevoerd. Verder is het ook niet meer nodig om *waits* of *sleeps* in te stellen om ervoor te zorgen dat er niet meteen naar de volgende test gesprongen wordt. Cypress wacht automatisch op antwoorden van beweringen vooraleer ze verdergaat. [18]

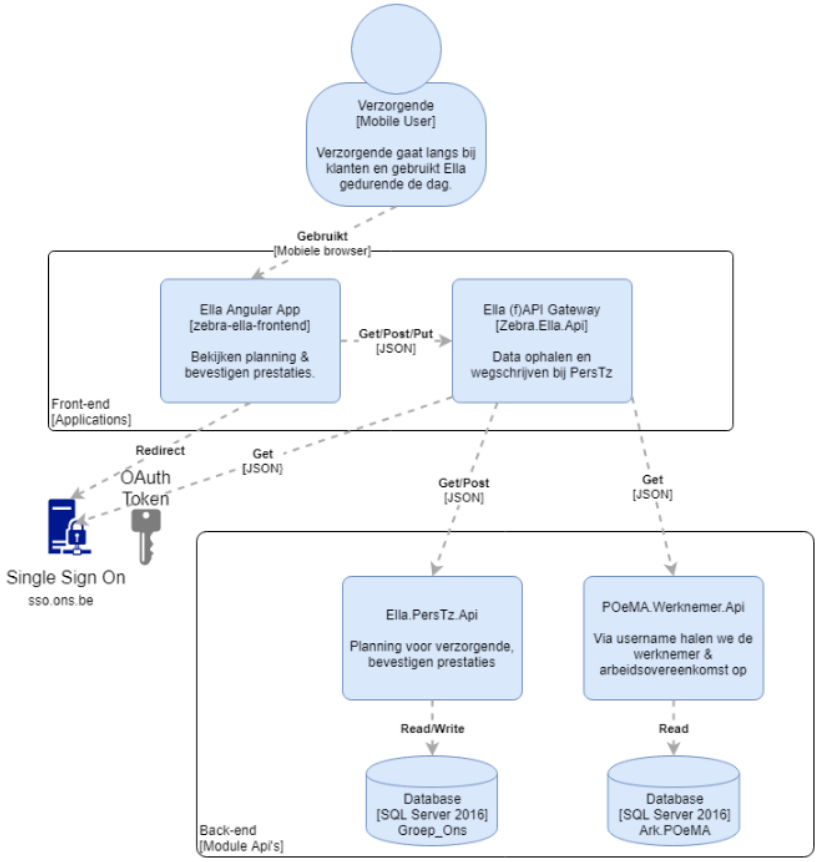
Tenslotte is vooral het gemak waarmee Cypress geïnstalleerd wordt en gebruikt kan worden dat er voor Ons voor gezorgd heeft dat het de gebruikte testframework is binnen haar ontwikkelteams.

2.4 Uitwerking stageopdracht

2.4.1 Architectuur: Overzicht bouwstenen voor Ell@

De Ell@-applicatie neemt de verzorgenden als startpunt. Zij gebruiken Ell@ op dagbasis voor het raadplegen van hun planning en om de afspraken te bevestigen. Zij voeren hun acties uit op de frontendapplicatie en deze communiceert met haar frontend-API om eerst data te ontvangen (bij het raadplegen van de planning) en daarna wijzigingen door te voeren aan de data (bij het bevestigen van een afspraak) en deze terug te sturen naar de databases. In feite passeert de data eerst langs de verantwoordelijke module-API's en die voeren de wijzigingen door aan de databases.

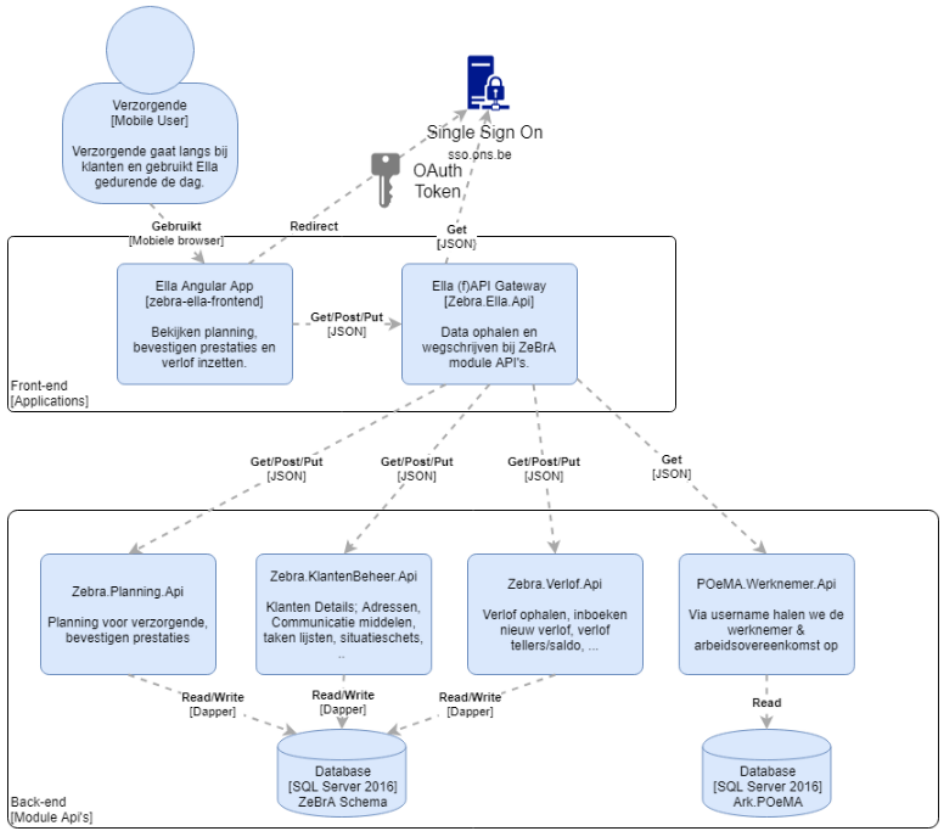
Het C4-model is bedoeld om visueel te kunnen communiceren en daar lijken veel softwareontwikkelaars het moeilijk mee te hebben. Daarom dat het C4-model, in tegenstelling tot allerlei diagrammen, verlangt naar abstractie en duidelijkheid op verschillende niveaus. C4 staat voor vier niveaus: Context (niveau 1), Container (niveau 2), Component (niveau 3) en Code (niveau 4). Onderstaande diagrammen zijn containerdiagrammen. Een UML-diagram is een goed voorbeeld van vier niveaus diep in het C4-model. [19] Onderstaande figuren zijn containerdiagrammen waarmee de Ons-teams communiceren en om de werking van de applicatie te illustreren.



Figuur 13: C4-containerdiagram Ell@ tijdens de stage

Dit diagram toont wat er gebeurt bij gebruik van de applicatie. De frontend is de app zoals de gebruiker ze te zien krijgt. De frontend heeft een API om met de backend te communiceren want er moet data over en weer kunnen gaan. De fAPI roept enerzijds de PersTz-API aan voor het tonen en bevestigen van afspraken en anderzijds de werknemer-API voor het filteren van de afspraken op basis van de ingelogde gebruiker. Op deze manier zijn de handelingen van de gebruiker verbonden met de databanken.

Dit diagram vult het vorige diagram aan met twee backend-API's: de klantenbeheer-API, die meer details over de klant ter beschikking stelt, en de verlof-API, die instaat voor het aangeven en raadplegen van verlof.



Figuur 14: C4-containerdiagram Ell@ gewenst na de stage

2.4.2 Stageverloop

1. Agile Scrum

Bij aanvang van de stage is er een planning gemaakt om min of meer een idee te krijgen van het te verzetten werk en wat wanneer gedaan zou worden. Naarmate de stage vorderde is deze planning nog een aantal keren veranderd. Dit valt te verklaren omdat Agile van een planning een heel flexibel begrip maakt. Zo staat in de aanvankelijke planning nog dat er tijdens de stage ook gewerkt zal worden aan de afwezigheden voor de verzorgenden. Echter is tijdens het paasverlof na stageweek 6 intern door Ons beslist om de scope aan te passen en enkel te focussen op de afsprakenplanning van Ell@ op voorwaarde dat deze dan ook echt getest kan worden door de verzorgenden en daarmee een werkend product opgeleverd wordt.

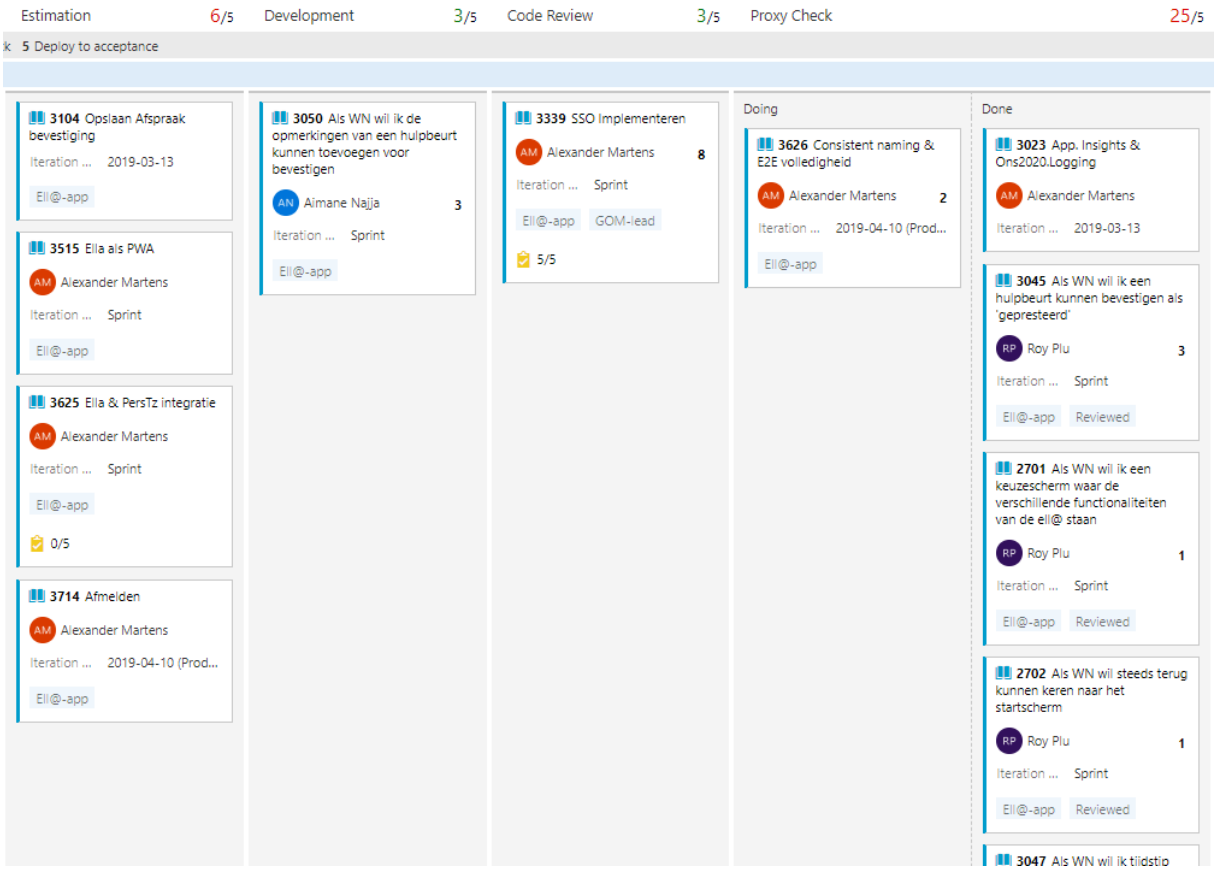
Week 1 & 2	RAADPLEGEN PLANNING (OP WEEKBASIS)
Week 3 & 4	RAADPLEGEN PLANNING (OP WEEKBASIS) RAADPLEGEN DETAILS KLANT
Week 5 & 6	BEVESTIGEN (OF WIJZIGEN) PRESTATIE - VAN / TOT UUR - KLANT ? - VERVOER VAN/ NAAR CLIENT + AANTAL KM ? - KOSTEN INGEVEN - OPMERKING
Week 7 & 8	RAADPLEGEN PRESTATIES (OP WEEKBASIS)
Week 9 & 10	INDIENEN AFWEZIGHEID (VERLOF, ZIEKTE, ...)
Week 11 & 12	RAADPLEGEN VERLOFSALDO

Tabel 1: Aanvankelijke planning

Letterlijk betekent Agile lenig of behendig en dit beschrijft dan ook de scrumwerking in de projecten van Ons. Kennis kan niet gepland worden maar wel ontstaan uit ervaringen en men maakt daarom pas beslissingen over wat gedaan moet worden op basis van wat er bekend is. Daarom worden de projecten van Ons gesplitst in sprints. Voor Ell@ is beslist om te werken in sprints van twee weken. Hierdoor is het makkelijker een bepaald doel voor ogen te leggen en daar dan in team naar toe te streven.

2. Sprintplanning

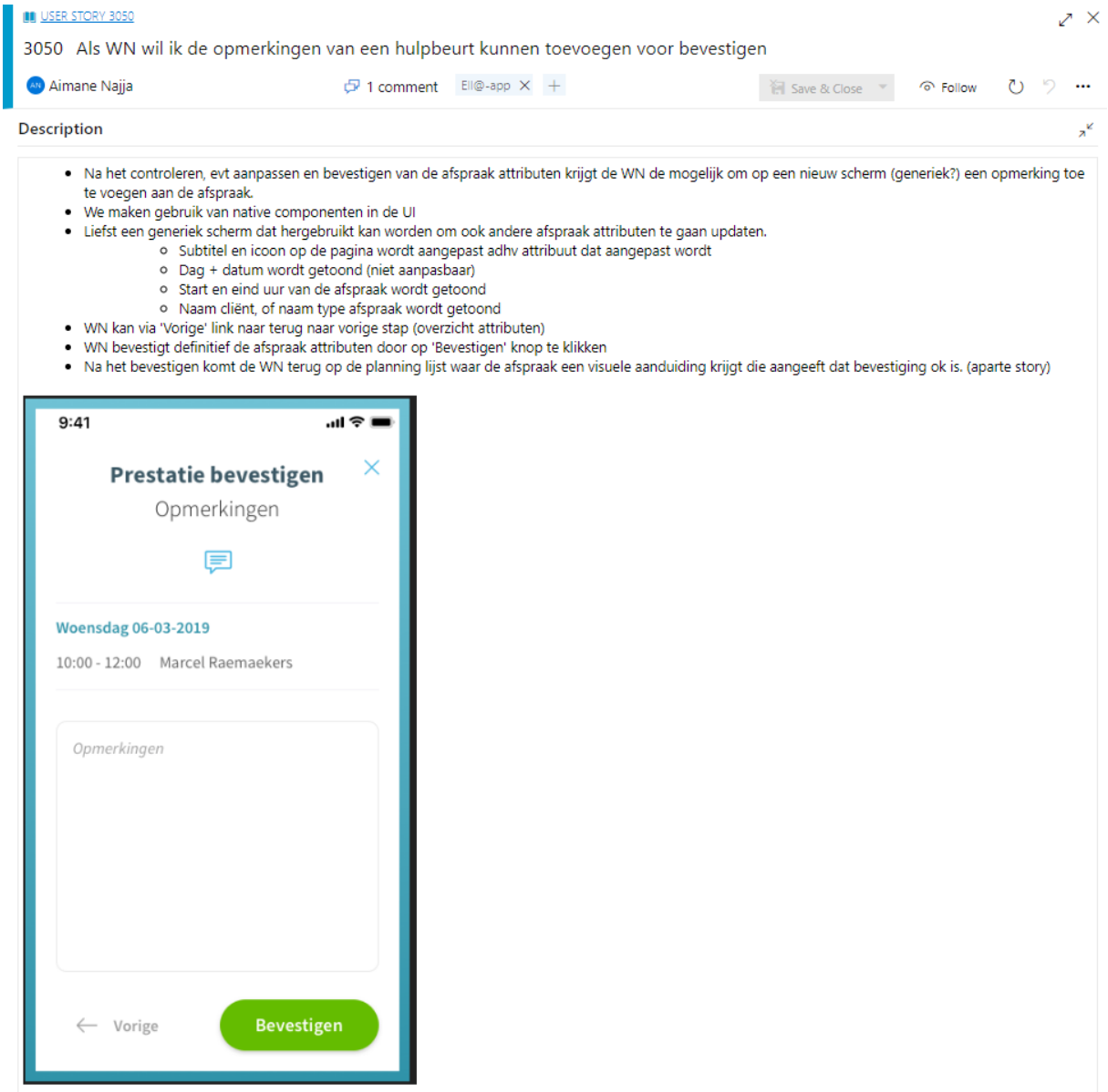
Vooraleer er *user stories* vanuit de backlog opgenomen kunnen worden in de sprintplanning, moeten deze in team ingeschat worden. Er worden aan iedere *user story* door middel van technisch overleg *story points* toegekend. Dit zijn getallen uit de rij van Fibonacci die moeten bepalen hoeveel tijd en werk een *user story* gaat kosten. Hoe hoger het getal, hoe meer werk uiteraard. Als de *user stories* elk een *story point* toegekend hebben gekregen, kan ieder teamlid een *user story* op zich nemen en daaraan werken.



Figuur 15: Kanban-bord Ell@

3. User Story

Elke *user story* is geschreven volgens de template en heeft daarom een titel waarin in één zin duidelijk moet zijn wat er met die *user story* bereikt moet worden. In de beschrijving geeft Ons eventueel verdere uitleg van vereisten om een *user story* af te ronden. Ook worden er voor iedere *user story* van de frontend screenshots toegevoegd die genomen zijn van de Ell@-Sketch. Sketch is een onlineservice waar de prototype van Ell@ op geplaatst is en bekeken en doorlopen kunnen worden.



Figuur 16: User Story Ell@

4. Code Review

Nadat er voor een *user story* testen geschreven zijn, mag ze ingecheckt worden en in het Kanban-bord naar Code Review gesleept worden waarna de stagebegeleiders de aanpassingen nakijken en er feedback over geven. Zij slepen de *user story* terug naar Development en als de opmerkingen verwerkt zijn mag de *user story* naar Proxy Check gesleept worden.

Figuur 17: Voorbeeld Code Review

Proxy Check wordt gedaan door meneer Pieters, de functioneel analist van Ons, die het Kanban-bord ook opvult met de *user stories*. Hiervoor is door de stagebegeleiders een testomgeving opgezet waarnaar gesurft kan worden op het Ons-netwerk zonder daarvoor in de lokale omgeving zaken te laten draaien. De Proxy geeft op iedere *user story* een ja of een neen afhankelijk van hoe tevreden hij is over het resultaat van de ontwikkeling met respectievelijk Done of Development als volgende fase van de *user story*.

5. Definition of Done

- 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 18: Definition of Done

Uiteindelijk als een *user story* alle fases doorlopen heeft en voldoet aan de voorwaarden om als afgewerkt beschouwd te kunnen worden, mag de *user story* verschoven worden naar Done en kan er aan een volgende *user story* gewerkt worden. Voor Ell@ betekende de *Definition of Done* vooral dat er code zowel in de frontend als in de backend is ontwikkeld, dat er testen geschreven zijn en deze slagen, dat er een *Code Review* gedaan en verwerkt is en tot slot de proxy de *user story* aanvaard heeft.

6. Daily Scrum

Doorheen het ontwikkelproces binnen Ons wordt dagelijks een scrummeeting gehouden om de teamleden een idee te geven waaraan gewerkt wordt door eenieder. Belangrijk is ook dat de teamleden weten wat er door eenieder gedaan gaat worden vandaag en of daar hulp bij gevraagd is. Zo'n scrummeeting duurt gewoonlijk niet langer dan een kwartier en wordt staand gehouden vandaar dat het ook wel eens een stand-up genoemd wordt.

2.4.3 Resultaat

1. Single Sign On

Ons heeft een SSO-software ontwikkeld dat voor al haar applicaties gebruikt wordt om de gebruikers te authenticeren. De verzorgenden hoeven op deze manier maar eenmaal in te loggen en daarna gebeurt binnen bepaalde tijd op datzelfde toestel de login automatisch.

Figuur 19: Single Sign On

Figuur 20: Landingspagina

2. Landingspagina

Over de landingspagina valt weinig te vertellen. De verzorgenden zouden hier oorspronkelijk de keuze moeten hebben tussen het bekijken van de planning en het bezoeken van het afwezighedenscherm. Vanwege de aanpassing van de scope, is voorlopig enkel de keuze van de planning overgebleven.

3. Planning

Het volgende scherm biedt de verzorgenden meteen duidelijke en relevante informatie. Bovenaan het scherm zien ze de huidige maand, het huidige jaar en de huidige week. De mogelijkheid is er om naar de volgende week te navigeren of een maandje terug in de tijd te kijken. Meer dan de helft van het scherm bestaat uit afspraken en men kan in de kalender bovenaan klikken op een dag in de week waarna het scherm onder automatisch zal scrollen naar aangeklikte dag. Swipen naar onder is ook gewoon een optie als men dat sneller vindt.

Figuur 21: Planning

De planningcomponent is zoals men in Angular noemt een *smart component*. Dit is een component dat een aantal *dumb components* overkoepelt die niet weten waar de data vandaan komt zolang ze het maar toebedeeld krijgen. Het zijn de *smart components* die werken met de data en het verdelen aan hun *dumb components*. In dit geval bevat de planningcomponent de logica om de juiste weken te tonen in de kalendercomponent. De kalendercomponent weet niets van die logica af maar moet het gewoon aannemen en tonen.

4. Kalender

De kalendercomponent is de eerste *dumb component* waar tijdens de stage aan gewerkt is. De vraag hier is dat steeds de dagen van de huidige week getoond worden en dat de dag van vandaag omcirkeld en het dagnummer van vandaag vetgedrukt wordt.

Figuur 22: Kalender

5. Huidige weekdagen

"Huidige weekdagen" is de component dat verantwoordelijk is voor het tonen van de weekdagen en hun bijhorende datums. Deze is ook verantwoordelijk voor het kleuren van de huidige dag en het scrollen naar de huidige dag.

Figuur 23: Huidige weekdagen

6. Afspraken

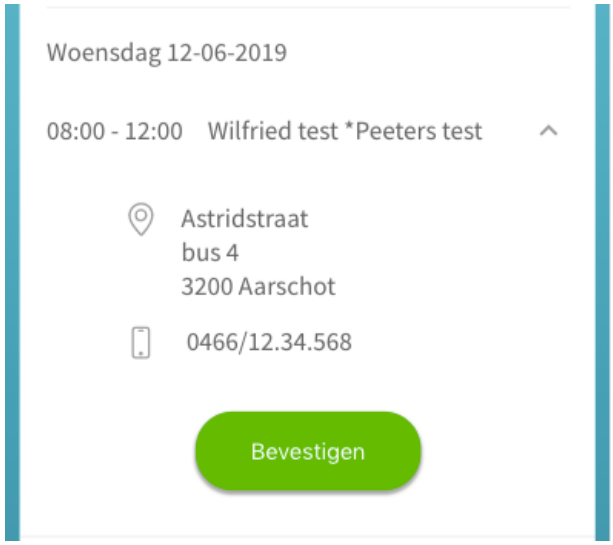
Vervolgens is er de afsprakencomponent dat een *dumb component* is van de "huidige weekdagen"-component. De afsprakencomponent hoort niet alle afspraken van de week maar enkel die van vandaag te krijgen. De afsprakencomponent doet deze filter niet zelf maar krijgt de afspraken gefilterd aan.

De afspraken hebben een begin- en een einduur die getoond moeten worden en een omschrijving oftewel een klantnaam die overigens ook getoond moeten worden. Indien er meerdere afspraken op één dag gepland zijn, wordt de afsprakencomponent herhaald.

Figuur 24: Afspraken

7. Afsprakendetails

De afsprakendetailscomponent valt onder de afsprakencomponent en geeft de nodige details weer van een afspraak zoals het adres en de telefoonnummers. Indien er aan een afspraak ook een klant gebonden is, kunnen er meer details bekeken worden door op de link ernaar te klikken. In dat geval opent de klantendetailscomponent in een full screen pop-upvenster. Hetzelfde geldt voor als men klikt op de "Bevestigen"-knop.



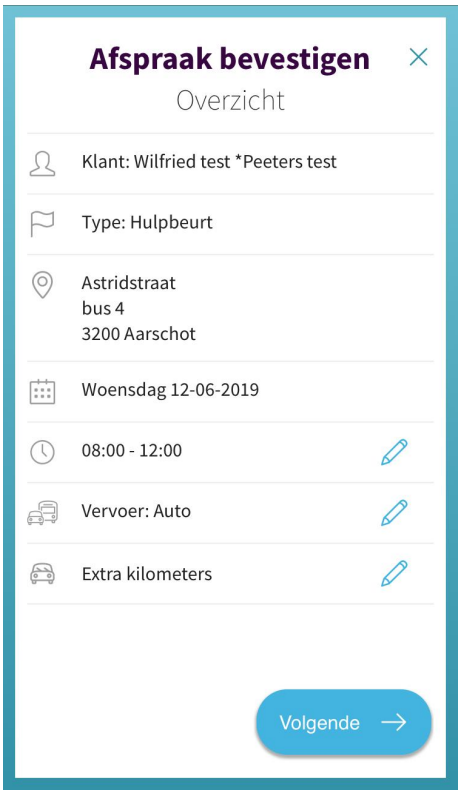
Figuur 25: Afsprakendetails

8. Afspraak bevestigen

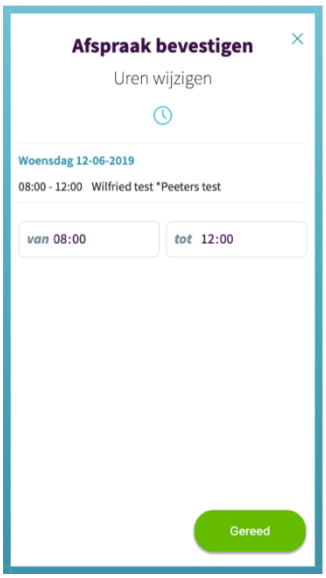
Tot slot is er nog een full screen venster voor het bevestigen van een afspraak. Deze component laat de verzorgende toe om een afspraak te bevestigen en indien nodig achteraf aan te passen.

De verzorgende kan bijvoorbeeld minder of meer dan de voorziene tijd op een afspraak zijn geweest. Er kan op weg naar de afspraak een omleiding zijn geweest waar dan de extra kilometers van genoteerd kunnen worden. Ook het type vervoer kan gewijzigd worden want de kilometervergoedingen verschillen natuurlijk per vervoerstype.

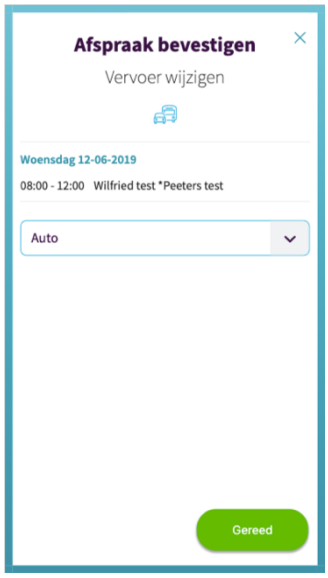
Technisch gezien blijft men in hetzelfde venster ook al wordt één van de drie aanpassingen gedaan of op “Volgende” geklikt. Angular toont met behulp van logica de juiste weergave op het venster. Zie hieronder de vier vensters die geopend zullen worden en waarop de eigenlijke wijzigingen doorgevoerd kunnen worden.



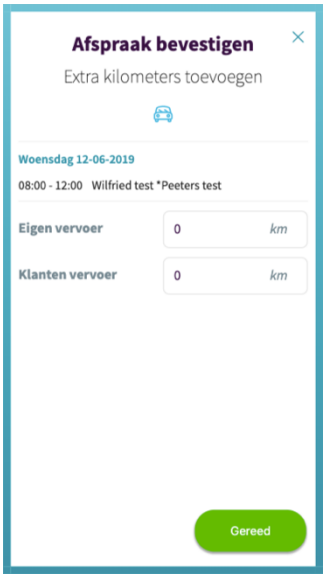
Figuur 26: Afspraak bevestigen



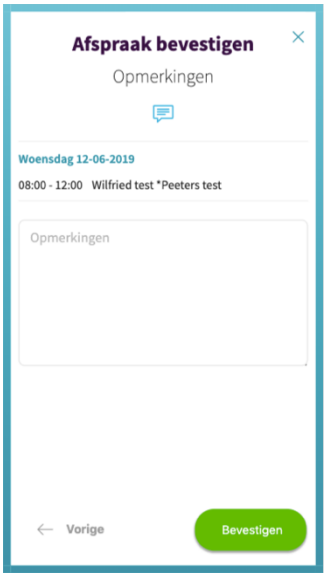
Figuur 27: Uren wijzigen



Figuur 28: Vervoer wijzigen



Figuur 29: Extra kilometers toevoegen



Figuur 30: Opmerkingen + Bevestigen

2.5 Conclusie

Bij aanvang had ik naar mijn mening voldoende kennis van Angular om aan de stage te beginnen. Het NgRx-library had ik al eens van gehoord omdat teamgenoten uit een vorig project het al eens gebruikten voor hun taken en nu was het dus aan mij om er mee te werken. Het Redux-patroon is technisch de grote aanwinst voor mijn IT-kennis en ik zie mezelf er niet meer zonder programmeren, althans niet voor complexe projecten.

Ik ben dankbaar dat Ons de Agile Scrum-methode respecteert omdat ik zo doorheen mijn stage elke dag een goed beeld had over mijn taken. Ik wist wat ik moest doen en er was nooit sprake van chaos of stress. Problemen worden meteen aangepakt wanneer ze zich voordoen en dat gaf me nog meer een gecontroleerd gevoel.

Deze stage voelde voor mij net als het volgen van rijlessen omdat ik vond dat ik goed genoeg kon rijden tot de instructeur mij wees op details waarvan ik eerder niet eens doorhad dat het fouten waren. Op dezelfde manier hebben mijn collega's me tijdens de stage regelmatig gecorrigeerd op een altijd respectvolle manier en nog belangrijker, ze toonden me ook elke keer duidelijk het verschil tussen mijn foutieve aanpak en waar ik naartoe moet streven. Dat is het type feedback waar men het meeste van leert en het siert Ons dat ze haar werknemers stimuleert zich zo goed mogelijk te ontwikkelen.

Achteraf kan ik zeggen dat het leerproces het belangrijkste was. Er is inzicht verkregen in een project van grotere schaal en er is samengewerkt tussen verschillende teams waarbij er kennis werd gedeeld om zo efficiënt mogelijk te werk te gaan. Waar het op de schoolbanken vaak eenrichtingsverkeer was van docent tot student, was er op het stagebedrijf een heel verkeer en netwerk van kennisdeling. Voor iedere technologie wist ik op het einde wie daarvoor het juiste aanspreekpunt was om met mijn werk te kunnen verdergaan als ik tegen een probleem aanliep en ik hoop dat er binnenkort ook beroep gedaan kan worden op mij door mijn collega's voor zaken waarvan zij denken dat ik ze goed beheers.

Deze stage bestond dus niet simpelweg uit het maken van een applicatie maar grotendeels ook uit het integreren van jezelf binnen een professioneel team en dus een teamspeler worden en daar de taken en verantwoordelijkheden van op je schouders nemen. Ik ben tevreden over mijn prestaties en het product dat opgeleverd is. Mijn stagebedrijf kan de applicatie gebruiken en zo zal mijn aandeel ook na de stage blijven leven.

II. Onderzoekstopic

1 Probleemstelling

Momenteel wordt de applicatie ontwikkeld voor de verzorgenden binnen Landelijke Thuiszorg. Meer bepaald wordt het een progressieve webapplicatie (PWA). Een PWA is een nieuwe manier van applicaties ontwikkelen dat veelbelovende mogelijkheden met zich meebrengt. Tot nu toe zijn het vooral Android-toestellen die een PWA het meest ondersteunen. De functionaliteiten op iOS-systemen daarentegen zijn op het moment helaas minder toegankelijk.

De meest opvallende feature van een PWA is de mogelijkheid om ze offline te kunnen gebruiken. De PWA maakt hiervoor gebruik van de *service worker*. Deze gaat de inhoud van de applicatie vanuit de cache downloaden op voorwaarde dat de browser een offline cache heeft. [20]

Ondanks vele andere interessante mogelijkheden van de PWA bieden nog niet alle browsers hier vandaag een even goede ondersteuning voor. Zo ondersteunt bijvoorbeeld Mozilla Firefox minder functionaliteiten dan Google Chrome. Om deze reden leek het voor Ons interessant om onderzoek te doen naar een alternatieve technologie waarbij de beperkingen verholpen worden zonder daarmee de essentiële voordelen van een PWA te verliezen.

De technologie die in dit onderzoek wordt behandeld, is Flutter [21]. Flutter is een nieuwe technologie en een nieuwe manier van mobiele applicaties maken. Het is ook vrij nieuw, want Google kondigde het in mei 2017 aan en bracht in december 2018 de eerste stabiele versie van Flutter uit. Omdat Flutter nog relatief nieuw is maar toch behoorlijk veel belooft te kunnen, is er veel interesse opgewekt waarmee aan dit onderzoek gestart kon worden. Het stagebedrijf had nood aan bepaalde functionaliteiten en dit onderzoek moet uitwijzen of die functionaliteiten door Flutter verkregen kunnen worden.

De vraag is wat Flutter precies te bieden heeft. Met andere woorden: "***Wat zijn de mogelijkheden van Flutter en hoe verschillend is de ontwikkeling van Flutter-applicaties ten opzichte van de huidige softwareontwikkeling?***". De twee relevantste nadelen van een PWA, zijnde notificaties ontvangen en vingerafdrukken scannen, vormen het doel van dit onderzoek.

2 Onderzoeksmethode

De Flutter-site zelf biedt een link naar een talrijke verzameling van voorbeeldprojecten en handleidingen voor veelgebruikte concepten. [22] [23] Eigenlijk is de Flutter-site de ideale plek om de studie over Flutter te beginnen.

Vervolgens is het verstandig om het internet te doorzoeken naar eventuele cursussen, liefst zo betrouwbaar mogelijk, die toegewijd zijn aan Flutter. Ook zijn er artikels te vinden van voorgangers die hun ervaringen en bevindingen delen met betrekking tot Flutter en waarschijnlijk zelfs de vergelijking maken met Angular. Artikels die nuttig zijn voor dit onderzoek, worden hier dan ook kritisch in verwerkt met telkens een verwijzing naar de bron.

Na de studie wordt, met de kennis die verworven is uit de studie, een deel van de Ell@-applicatie nagebouwd en wordt deze aangevuld met de mogelijkheden die Flutter biedt als oplossing voor de PWA-beperkingen. Afspraken moeten opgehaald worden en deze moeten bevestigd kunnen worden met de vingerafdruk. Bij wijziging van de data, ontvangt de gebruiker een notificatie. Eventuele nadelen van Flutter zullen achteraf ook besproken worden in dit onderzoek.

3 Literatuurstudie

De literatuurstudie zorgt in dit onderzoek voor meer uitleg over de technieken om notificaties te ontvangen bij het wijzigen van afspraken door middel van authenticatie. De bronnen worden niet zozeer vergeleken met elkaar, maar worden wel geïmplementeerd in de *proof of concept.* De PoC is dus een combinatie van technieken die uit onderstaande bronnen gehaald zijn.

3.1 Authenticatie

3.1.1 Flutter Packages

Flutter biedt een grote verzameling voorgedefinieerde *packages* waarmee sneller applicaties gebouwd kunnen worden zonder alles zelf te moeten ontwikkelen. [24] Deze *packages* kunnen door het Flutter-team zelf ontwikkeld zijn en hebben dan doorgaans een score dicht bij 100. Ook andere ontwikkelaars kunnen *packages* maken en deze worden dan geanalyseerd op basis van drie criteria: populariteit, correctheid en onderhoud. [25] Hoe meer een *package* gebruikt wordt, hoe minder foutmeldingen een *package* oploopt en hoe vaker een *package* wordt geüpdatet, des te hoger de score van de *package* zal liggen. [26] Om de *packages* te kunnen gebruiken in de applicatie, moeten ze gedefinieerd worden in pubspec.yaml zoals onder de *Installing*-tab staat uitgelegd voor iedere *package.*

3.1.2 local_auth

De local_auth *package* zorgt ervoor dat een gebruiker zich kan authenticeren door zijn vingerafdruk, zijn ogen of zijn gezicht te scannen. Als deze overeenkomt met de op het toestel bewaarde vinger-, gezichts- of oogscans, dan geeft de *package* een positief resultaat en anders negatief.

De local_auth *package* bestaat uit drie klassen waarvan LocalAuthentication de belangrijkste is. De andere twee zijn voor het tonen van de juiste boodschappen op Android of iOS. De klasse LocalAuthentication bevat drie methodes: authenticateWithBiometrics, canCheckBiometrics en getAvailableBiometrics. De getAvailableBiometrics-methode gaat controleren of het toestel de vingerafdrukken, het gezicht of de ogen kan lezen en geeft een verzameling van de ondersteunde biometrieken terug. De canCheckBiometrics-methode gaat kijken of die verzameling al dan niet leeg is en de authenticateWithBiometrics-methode doet het eigenlijke werk. [27]

```
_localAuthentication.authenticateWithBiometrics(
      localizedReason: "Please authenticate to complete your transaction.",
      useErrorDialogs: true,
      stickyAuth: true,
);
```

Een voorbeeldproject van Pawan Kumar, Google Developer Expert voor onder andere Flutter en Dart, toont het gemak waarmee authenticatie kan worden gerealiseerd. [28] In zijn voorbeeld gebruikt roept hij de drie methodes van de LocalAuthentication-klasse aan bij het klikken op een RaisedButton. Het resultaat van elk van de methodes geeft hij weer in drie verschillende labels. In zijn bijhorende YouTube-video toont Kumar ook enkele specifieke instellingen om de local_auth *package* te laten werken op zowel het iOS- als het Android-platform. [29] Voor zowel iOS als Android moet toelating gegeven worden en daarvoor moet voor beide een regel toegevoegd worden aan de platformgebonden code.

3.2 Firebase

Firebase is een platform dat backendservices aanbiedt voor ontwikkelaars van web en mobiele applicaties. Het doel van Firebase is om ontwikkelaars te ontlasten van het maken en het onderhouden van vaak ingewikkelde services.

3.2.1 Firebase Authentication

Firebase biedt voor het authenticeren van gebruikers een tiental loginproviders. Voor dit onderzoek wordt enkel gebruikgemaakt van de combinatie e-mail en wachtwoord. [30] In Flutter gebeurt de registratie van een gebruiker met de methode, createUserWithEmailAndPassword, die uiteraard een e-mailadres en een wachtwoord verwacht. Het is aangeraden om een bestaand e-mailadres te gebruiken omdat Firebase ook de mogelijkheid biedt het wachtwoord te resetten door een e-mail naar dat adres te versturen.

```
_auth.createUserWithEmailAndPassword(email: email, password: password);
```

Als de gebruiker gecreëerd is in de app, kan er ingelogd worden met het gekozen e-mailadres en wachtwoord. Flutter geeft deze waardes dan mee aan de methode, signInWithEmailAndPassword. Deze methode retourneert een FirebaseUser en als deze niet leeg is, kan er genavigeerd worden naar een volgende pagina. [31]

```
final newUser = _auth.signInWithEmailAndPassword(email: email, password: password);
if (newUser != null) Navigator.pushNamed(context, HomeScreen.id);
```

3.2.2 Cloud Firestore

Firebase biedt een NoSQL cloud database aan, genaamd Cloud Firestore, voor het opslaan van data en deze data in sync houden op de gekoppelde applicaties door middel van realtime listeners. [32] Deze database bouwt verder op de successen van Realtime Database. Deze voorganger van Firestore sloeg de data echter enkel op als JSON-objecten terwijl Firestore ondersteuning biedt voor meerdere datatypes. [33]

Voor het ophalen van gegevens uit Firestore kan er in Flutter beroep gedaan worden op de methode, collection('*de naam van de collectie*').getDocuments(). Aangezien deze methode over HTTP requests zal sturen, is de returnwaarde een Future. Een Future is de Flutter-variant van een Task in C# en geeft aan dat de methode asynchroon wordt uitgevoerd om de rest van de applicatie niet te laten wachten. Als de gegevens toch nodig zijn om andere regels code uit te voeren kan in Flutter hier op gewacht worden, net als in C#, door het *keyword* 'async' vóór de methode te plaatsen en de overkoepelende methode waarin de asynchrone methode aangeroepen wordt, ook te bestempelen als Future door het *keyword* 'await' ervoor te plaatsen.

```
async {
      final afspraken = await _firestore.collection('afspraken').getDocuments();
      …
```

Het wijzigen van een document in Firestore is even gemakkelijk. De methode, collection('*de naam van de collectie*').document('*het id van het document*').updateData(), verwacht een Map. Een Map is een verzameling van sleutels en hun waardes. In dit geval is de sleutel, die meegegeven moet worden aan de updatemethode, de naam van het veld en is de waarde vanzelfsprekend de nieuwe

waarde die in dat veld gestoken wenst te worden. Het is niet nodig om alle velden in de Map te steken als enkel één veld gewijzigd moet worden, maar volstaat het een Map te maken van één *key/value pair*.

```
_firestore.collection('afspraken')
          .document(documentID)
          .updateData({'bevestigd': true});
```

Om de gegevens realtime te tonen in Flutter moet er gebruik gemaakt worden van de klasse StreamBuilder. Deze klasse verwacht een stream van gegevens en zal elke verandering aan de meegegeven stream observeren. De methode, collection('de *naam van de collectie*').snapshots(), haalt de gegevens van de collectie op als een stream en deze kan dan ook gebruikt worden voor de StreamBuilder.

```
StreamBuilder<QuerySnapshot>(
       stream: _firestore.collection('afspraken').snapshots(),
       …
),
```

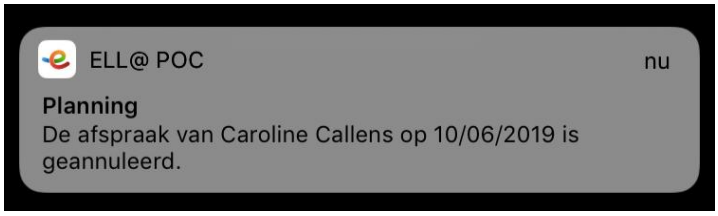
Telkens wanneer de stream verandert, wordt de buildermethode opnieuw uitgevoerd. Gewoonlijk retourneert de buildermethode een Widget, voornamelijk gebruikt als UI-elementen in Flutter, die steeds geüpdatet zal worden door de StreamBuilder.

```
StreamBuilder<QuerySnapshot>(
       …
       builder: (context, snapshot) {
                    …
                }
),
```

Op de plaats waar de Widget, die de buildermethode van de StreamBuilder retourneert, moet komen op het scherm, wordt de StreamBuilder gezet. Omdat de StreamBuilder ook een Widget is, kan dit zomaar. Nagenoeg alles in Flutter is een Widget. De StreamBuilder zal er op het scherm net zo uitzien als de Widget die het returnt, maar het enige verschil is dat de Widget dus realtime zal worden geüpdatet bij iedere wijziging aan de gespecificeerde stream van Firestore. [34]

3.2.3 Firebase Cloud Messaging

Een notificatie wordt door FCM verstuurd als een JavaScript-object met een *title* en een *body*. In het voorbeeld op figuur 31 is "Planning" de *title* en "De afspraak van Caroline Callens op 10/06/2019 is geannuleerd" is de *body* van de notificatie.



Figuur 31: Firebase Cloud Message op iOS

Voor het kunnen ontvangen van notificaties voor iOS-applicaties moet de ontwikkelaar van de app beschikken over een Apple developer account en daarop een sleutel aanmaken om verbinding te maken met de Apple Push Notifications-service. Daarvoor moet deze sleutel gedownload worden en op Firebase worden geüpload bij de betreffende app.

Voor het ontvangen van notificaties bestaat een Flutter *package*, genaamd firebase_messaging. Deze *package* is ontwikkeld door het Flutter-team en ze zeggen erbij dat de *package* nog in ontwikkeling is. Desondanks heeft de *package* een score van 100. In onderstaande figuur is te zien op welke vlakken er nog aan de *package* gewerkt moet worden.

	App in Foreground	App in Background	App Terminated
Notification on Android	`onMessage`	Notification is delivered to system tray. When the user clicks on it to open app `onResume` fires if `click_action: FLUTTER_NOTIFICATION_CLICK` is set (see below).	Notification is delivered to system tray. When the user clicks on it to open app `onLaunch` fires if `click_action: FLUTTER_NOTIFICATION_CLICK` is set (see below).
Notification on iOS	`onMessage`	Notification is delivered to system tray. When the user clicks on it to open app `onResume` fires.	Notification is delivered to system tray. When the user clicks on it to open app `onLaunch` fires.
Data Message on Android	`onMessage`	`onMessage` while app stays in the background.	*not supported by plugin, message is lost*
Data Message on iOS	`onMessage`	Message is stored by FCM and delivered to app via `onMessage` when the app is brought back to foreground.	Message is stored by FCM and delivered to app via `onMessage` when the app is brought back to foreground.

Figuur 32: Ondersteuning FCM

De FirebaseMessaging-klasse voorziet een enkele methodes: requestNotificationPermissions en subscribeToTopic. Er zijn nog meer methodes aanwezig in de klasse maar dit zijn de voornaamste. Voor iOS moet de eerste keer een melding getoond worden om toelating te vragen voor het sturen van notificaties. Dit wordt gedaan met de methode requestNotificationPermissions.

```
_firebaseMessaging.requestNotificationPermissions(
        const IosNotificationSettings(sound: true, badge: true, alert: true));
```

Aan het JavaScript-object dat door FCM gestuurd wordt, kan ook een topic (zie later) meegegeven worden. Om notificaties te ontvangen van een bepaalde topic moet hier op worden gesubscribed.

```
_firebaseMessaging.subscribeToTopic('afspraken');
```

In bovenstaand voorbeeld zullen de notificaties met het topic ‘afspraken’ ontvangen worden.

```
DATA='{"notification": {"body": "this is a body","title": "this is a
title"}, "priority": "high", "data": {"click_action":
"FLUTTER_NOTIFICATION_CLICK", "id": "1", "status": "done"}, "to": "<FCM
TOKEN>"}'
curl https://fcm.googleapis.com/fcm/send -H "Content-Type:application/json"
-X POST -d "$DATA" -H "Authorization: key=<FCM SERVER KEY>"
```

Figuur 33: Notificatie versturen via de terminal

Bovenstaande figuur toont de twee commando’s waarmee een notificatie verstuurd kan worden naar één specifieke gebruiker met behulp van een token. Deze verschilt een klein beetje van het type notificatie dat in dit onderzoek gebruikt wordt.

3.3 Cloud Functions voor Firebase

Om de grootte van de applicatie te verkleinen en de werking van de applicatie te versnellen is het niet altijd verstandig om alle logica in de applicatie zelf te steken. Ook naar veiligheid toe brengt dit de nodige risico's met zich mee. Doorgaans zou de ontwikkelaar hiervoor zelf een backendserver moeten opzetten waar de applicatie door middel van een API mee kan communiceren. Op deze manier wordt kan het zware werk verricht worden door de server en kan ook het gedrag van de app aangepast worden door de ontwikkelaar zonder dat de gebruikers daarvoor een nieuwe versie van de app moeten downloaden. [35]

Om consistent te blijven met de overige gebruikte Firebase-services, wordt in dit onderzoek gebruikgemaakt van Cloud Functions. Cloud Functions gaat als het ware de gebruikte services aan elkaar lijmen. Er moet namelijk een Cloud Message verzonden kunnen worden met het topic 'afspraken' wanneer er aan de collectie 'afspraken' in Cloud Firestore iets veranderd wordt zodat de gebruikers van deze verandering door de applicatie op de hoogte kunnen worden gebracht met een gedetailleerde notificatie.

3.3.1 Cloud Firestore Triggers

Een Cloud Function kan aangeroepen worden door een Cloud Firestore *trigger*. Cloud Firestore biedt vier soorten *triggers* aan: *onCreate, onUpdate, onDelete* en *onWrite*. De *onCreate trigger* wordt afgetrapt wanneer een document aan een collectie wordt toegevoegd. De *onUpdate trigger* vindt plaats wanneer aan een bestaand document een aanpassing gedaan wordt. De *onDelete trigger* wordt gevuurd wanneer een document verwijderd wordt. De *onWrite trigger* observeert alle drie de soorten veranderingen laat het werk aan de ontwikkelaar om het onderscheid te maken als het nodig zou zijn.

3.3.2 Cloud Function koppelen en opzetten

Cloud Functions worden geschreven in JavaScript maar zullen voor dit onderzoek geschreven worden in TypeScript aangezien Cloud Functions hier een *predeploy hook* voor voorzien heeft die TypeScript-code omzet naar JavaScript vooraleer het gedeployed wordt naar Firebase. De keuze voor TypeScript is gemaakt vanwege de aanwezige kennis ervan die opgedaan werd tijdens de stage. De JavaScript-code zal draaien in een NodeJS-omgeving dus deze zal geïnstalleerd moeten worden. Node is ook nodig om de Firebase CLI te installeren zodat de Cloud Functions gedeployed kunnen worden. [36]

```
npm -g install firebase-tools
```

Om de Cloud Function te koppelen aan het juiste Firebase-project moet er worden ingelogd met een account dat toegang heeft tot het project.

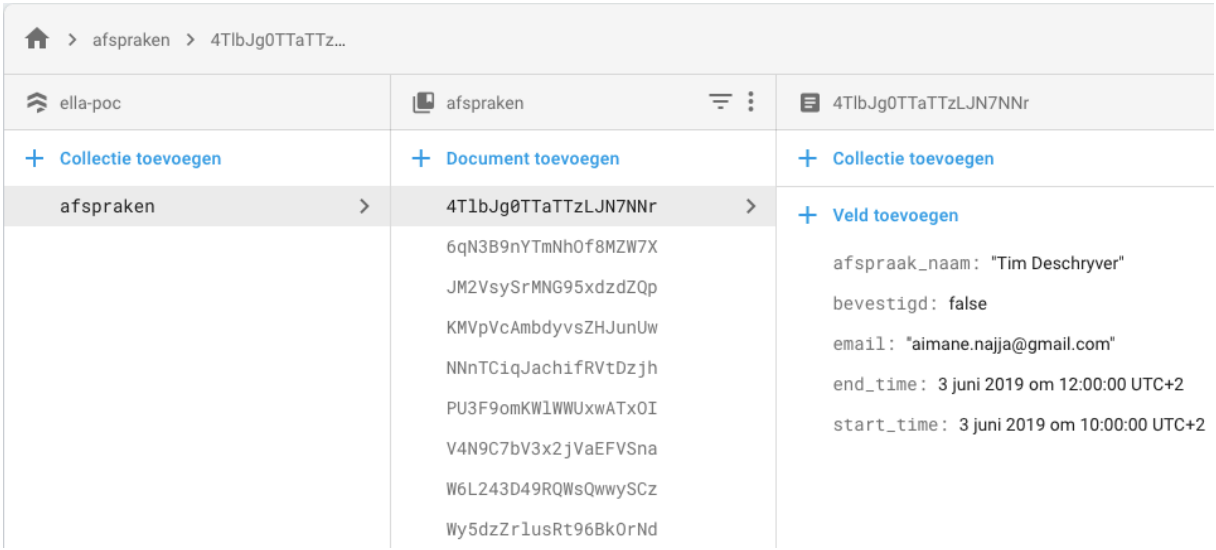
```
firebase login
```

Vervolgens kan met onderstaand commando een mappenstructuur gegenereerd worden. De CLI zal vragen voor welke Firebase-service de aanzet gedaan moet worden, voor welk project op het aangemelde account en in welke programmeertaal. En tot slot vraagt de CLI of TSLint kan helpen en of er *dependencies* geinstalleerd mogen worden.

```
firebase init
```

3.3.3 Cloud Message versturen na *onUpdate trigger*

Gemakkelijkheidhalve is er voor iedere afspraak een extra veld 'bevestigd' toegevoegd die standaard op *false* staat. De database ziet er uit zoals in onderstaande figuur.



Figuur 34: Database Ell@ PoC

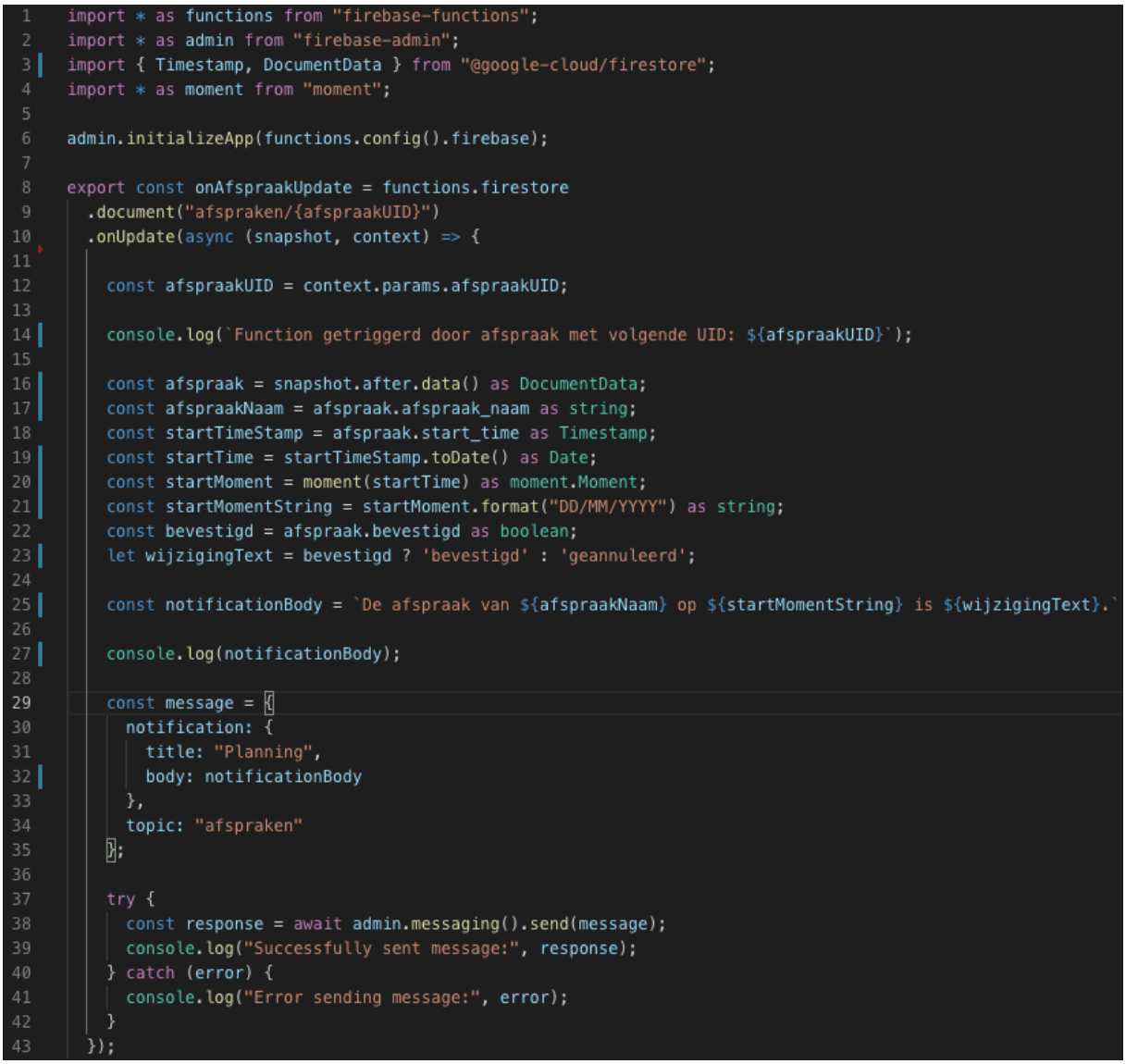
Er bestaat één collectie genaamd afspraken met daarin verschillende documenten gekenmerkt door een ID en elk document heeft momenteel vijf velden.

Als bij het authenticeren (zie hoofdstuk 3.1.2) het 'bevestigd'-veldje wordt gewijzigd naar *true* (of weer naar *false*), zoals in hoofdstuk 3.2.2 wordt uitgelegd, dan kan een Cloud Function aangeroepen worden. In dit onderzoek is al ondervonden hoe Cloud Messages verstuurd kunnen worden via de terminal maar het is ook mogelijk om deze door een Cloud Function te laten versturen.

De *onUpdate trigger* vuurt de *onUpdate*-methode (zie figuur 35) af en deze heeft een DocumentSnapshot en een EventContext als parameters. Uit de *context* kan bijvoorbeeld de ID van het document worden opgehaald maar de *snapshot* is voor dit onderzoek interessanter. De *snapshot* bevat twee parameters *before* en *after* die respectievelijk de data vóór de wijziging en de gewijzigde data bevatten. Ook hier is enkel de *after* van belang aangezien het voor het onderzoek voldoende is om te melden dat een afspraak bevestigd of geannuleerd is.

De data kan correct worden geconverteerd met behulp van TypeScript om een gedetailleerde notificatie te kunnen versturen. Zoals gezegd in hoofdstuk 3.2.3, bevat een notificatie een *title* en een *body*. Dit notificatieobject is onderdeel van de Firebase Cloud Message waar dus ook een topic aan meegegeven kan worden. Aangezien de Flutter-applicatie luistert naar Messages van de topic 'afspraken', moet aan de Message de *string* "afspraken" worden meegegeven.

Tenslotte wordt de message meegegeven aan de methode admin.messaging.send en zal, als alles goed is geconfigureerd, de notificatie ontvangen worden zoals in figuur 31.



```
import * as functions from "firebase-functions";
import * as admin from "firebase-admin";
import { Timestamp, DocumentData } from "@google-cloud/firestore";
import * as moment from "moment";

admin.initializeApp(functions.config().firebase);

export const onAfspraakUpdate = functions.firestore
  .document("afspraken/{afspraakUID}")
  .onUpdate(async (snapshot, context) => {

    const afspraakUID = context.params.afspraakUID;

    console.log(`Function getriggerd door afspraak met volgende UID: ${afspraakUID}`);

    const afspraak = snapshot.after.data() as DocumentData;
    const afspraakNaam = afspraak.afspraak_naam as string;
    const startTimeStamp = afspraak.start_time as Timestamp;
    const startTime = startTimeStamp.toDate() as Date;
    const startMoment = moment(startTime) as moment.Moment;
    const startMomentString = startMoment.format("DD/MM/YYYY") as string;
    const bevestigd = afspraak.bevestigd as boolean;
    let wijzigingText = bevestigd ? 'bevestigd' : 'geannuleerd';

    const notificationBody = `De afspraak van ${afspraakNaam} op ${startMomentString} is ${wijzigingText}.`

    console.log(notificationBody);

    const message = {
      notification: {
        title: "Planning",
        body: notificationBody
      },
      topic: "afspraken"
    };

    try {
      const response = await admin.messaging().send(message);
      console.log("Successfully sent message:", response);
    } catch (error) {
      console.log("Error sending message:", error);
    }
  });
```

Figuur 35: Cloud Function voor het versturen van een notificatie na een update in Cloud Firestore

4 Prototype

4.1 Voorafgaand verloop

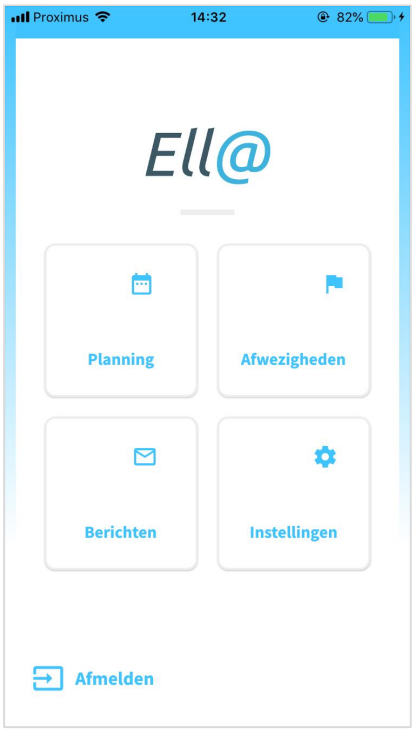
De Flutter-prototype ontwikkelen bracht de nodige tegenwind met zich mee. Bescheiden werden de eerste stappen gezet in de Windows-omgeving maar al kwam het besef dat met Windows niet op een fysiek iOS-toestel gedeployed kan worden. Hiervoor is macOS nodig. Als eerste alternatief werd gebruikgemaakt van een Android Simulator, maar ook dit bleek niet naar wens te werken wegens de traag werkende Hot Reload-functionaliteit. Een macOS-partitie creëren op een externe harde schijf en een macOS-VM gebruiken via VMWare waren de volgende twee pogingen, maar ook hier ging het nodige mis. De oude MacBook Air betekende uiteindelijk het licht aan het eind van de tunnel. Hiermee vorderde het onderzoek sneller en met grotere stappen richting een goed resultaat. MacOS heeft het voordeel dat er virtueel en fysiek op zowel Android- als iOS-toestellen gedeployed kan worden.

4.2 Installatie Flutter

Allereerst moet de Flutter SDK afgehaald worden van de Flutter-site en uitgepakt worden in een lokale map. Vervolgens moet de systeemvariabele 'PATH' worden aangevuld met het pad naar de bin-folder in de gedownloade Flutter SDK-map om van het commando 'flutter' gebruik te kunnen maken. Als bovenstaande stappen correct uitgevoerd zijn, moet het commando 'flutter doctor' aangeven wat nog eventueel in orde gebracht moet worden. Om de Flutter-apps te kunnen uitvoeren, is het aangeraden om Xcode en Android Studio te installeren omdat deze de nodige tools met zich meebrengen voor respectievelijk iOS en Android. Voor het deployen naar een fysiek iOS-toestel is een Apple-account vereist en moeten nog extra tools geïnstalleerd worden via Homebrew. [37] In dit onderzoek is gebruikgemaakt van VS Code als editor wegens een persoonlijke voorkeur.

4.3 Reconstructie Ell@-design

Als beginner die nooit eerder van de programmeertaal Dart gehoord heeft en voor de eerste keer met het Flutter-framework aan de slag gaat, kan Flutter op het eerste zicht ongestructureerd lijken. Angela Yu heeft, in samenwerking met het Flutter-team, een bijna allesomvattende cursus gemaakt die alle fundamentele concepten van Flutter behandeld. [38] Deze cursus bezorgt een beginner voldoende kennis over en ervaring in Flutter om zelf aan de slag te kunnen gaan. Angela Yu laat haar studenten mee ontwikkelen aan een tiental volwaardige Flutter-apps met daarbij de uitleg over de gebruikte concepten.



Figuur 36: Landingspagina

4.3.1 RoundedRectangleButton Widget

De eerste stap van het ontwikkelen van het prototype was het tonen van tekst op het scherm in het gewenste lettertype "Source Sans Pro".

Flutter beschikt over alle standaard Material kleuren en de kleur die gebruikt is over het merendeel van het prototype is Colors.lightBlueAccent, dat overeenkomt met een hexadecimale waarde van 40C4FF. Zoals bekend in vele programmeertalen staat deze waarde voor het Rood-Groen-Blauw-kleurenmodel.

Net zoals Flutter een groot aantal kleuren van Material ter beschikking stelt, brengt het ook een verzameling icoontjes met zich mee van Material. Het is ook mogelijk de Cupertino-icoontjes te gebruiken door middel van een verwijzing naar de betreffende *package* te vermelden in pubspec.yaml. Cupertino is de naam van het iOS-design en is vernoemd naar de stad waar het hoofdkantoor van Apple is gevestigd. Het Material-design is gemaakt door Google en is veelal gebaseerd op Android.

Eens het gelukt is om een icoon in te laden, moeten ze correct gepositioneerd worden en liefst binnen een afgerond vierkant. Voor het positioneren van elementen in Flutter bestaan er Layout Widgets die verdeeld zijn in twee categorieën: Single-Child en Multi-Child Layout Widgets. [39] Zowel de tekst als het icoontje moeten in een afgerond vierkant komen, dus is hiervoor een Multi-Child Layout Widget nodig. De icoontjes moeten in de verticale richting gepositioneerd worden dus een kolom is hiervoor het meest gepast.

Een Column Widget heeft standaard geen *padding* waarmee de inhoud van de kolom afgescheiden kan worden van de randen, maar Flutter heeft in dit geval een Padding Widget voorzien waarin de kolom gestoken kan worden om alsnog te voorkomen dat de tekst en het icoontje tegen de rand geplakt worden. Om de afgeronde rand te tekenen, kan de Padding Widget, waarin de Column Widget zit en dus ook de Text en Icon Widget, gewrapt worden door een Card Widget.

De Card Widget bevat een *shape* waarmee het figuur bepaald kan worden dat de Card visueel moet afbeelden. De Card Widget kan geen hoogte of breedte geven aan het vierkant, maar daarvoor bestaat de SizedBox Widget waarin de Card gewrapt kan worden.

Op deze manier ontstaat, zoals men in Flutter-termen noemt, een Widget Tree en om niet voor elk van de vier vierkanten dezelfde Widget Tree te moeten kopiëren, kan de Widget Tree uitgetrokken worden naar een aparte Widget. Op die manier voldoet Flutter aan OOP en is de code makkelijker te hergebruiken.

De Hero Widget (zie regel 28 in bijlage A) markeert de Widget die daarmee gewrapt wordt als kandidaat voor een Hero Animation. Dit type animatie zorgt ervoor dat Widgets die op meerdere schermen bestaan niet verdwijnen bij het navigeren, maar wel zichtbaar de overgang maken naar het volgende en ook terug naar het vorige scherm. In dit geval wordt de tekst “Planning” ook gebruikt als titel bovenaan het volgend scherm.

De FlatButton Widget (zie regel 20 in bijlage A) zorgt er, in tegenstelling tot de RaisedButton of andere Buttons, voor dat de Widgets niet van lay-out veranderen, op uitzondering van een beetje *padding* in sommige gevallen. Het is uiteraard de bedoeling dat de FlatButton ervoor moet zorgen dat er iets gedaan moet worden wanneer de gebruikers op het vierkant klikken. Dit was ook mogelijk geweest met een GestureDetector Widget, die zelfs voor meerdere types van aanraking verschillende handelingen toelaat, maar de FlatButton geeft visuele feedback na het klikken zodat het duidelijk is dat er een aanraking is geweest. In dat geval moet er genavigeerd worden naar het betreffende scherm en aangezien de *onPressed*-functie als *property* is gedefinieerd voor de RoundedRectangleButton Widget, kan voor ieder vierkant een andere actie worden uitgevoerd. Er hoeft dus niet altijd genavigeerd te worden. Er kan dus, bij wijze van spreken, ook een geluid worden afgespeeld bij het klikken op het vierkant.

Als alle knoppen gemaakt zijn en elke knop heeft een verschillende tekst en een verschillend icoontje, dan kan door middel van Row en Column Widgets de positionering gefixt worden. Daarna kan dat menu van vier knoppen worden uitgetrokken naar een aparte Widget, genaamd HomeScreenMenu (zie bijlage B).

4.3.2 Logo Widget

De volgende stap is het inladen van de logo. Om de logo niet iedere keer op te halen over het netwerk, wordt ze best als *asset* in het project gestoken. In Flutter kunnen afbeeldingen op verschillende manieren opgehaald worden, maar voor dit onderzoek wordt gebruikgemaakt van de AssetImage als ImageProvider. Hiervoor moet de locatie van de afbeelding of van de hele map met afbeeldingen worden gespecificeerd in pubspec.yaml zodat de AssetImage de afbeelding kan fetchen.

```
4   class Logo extends StatelessWidget {
5
6     @override
7     Widget build(BuildContext context) {
8       return Column(
9         mainAxisSize: MainAxisSize.min,
10        children: <Widget>[
11          Image(
12            image: AssetImage('images/ella.png'),
13            width: 120.0,
14          ), // Image
15          CustomPaint(
16            painter: LinePainter(),
17            size: Size.square(50.0),
18          ), // CustomPaint
19        ], // <Widget>[]
20      ); // Column
21    }
22  }
23
```

Figuur 37: Logo Widget

De CustomPaint Widget staat in voor het tekenen van de grijze lijn onder het logo.

De twee Widgets horen onder elkaar te komen dus mogen ze in een Column Widget gestoken worden. Tenslotte mag dit stuk code ook uitgetrokken worden naar een aparte Widget, genaamd Logo. Op deze manier hoeft er niet steeds gelezen te worden dat een LinePainter Widget gebruikt is om het logo te tekenen als men de structuur van de pagina wilt bekijken.

4.3.3 Afmelden Widget

Tenslotte rest er voor de landingspagina enkel nog de knop linksonder om af te melden. Deze knop bestaat uit een Icon en een Tekst Widget dir horizontaal gepositioneerd worden door een Row Widget. Deze Row wordt gewrapt in een FlatButton zodat er op alle Widgets onder de Row gedrukt kan worden om af te melden. Ook hier kan er een aparte Widget uitgetrokken worden, genaamd Afmelden (zie bijlage C). In dit geval hoeft de *onPressed*-methode niet meegegeven te worden aan de Afmelden Widget omdat er iedere keer afgemeld moet worden en genavigeerd moet worden naar de loginpagina.

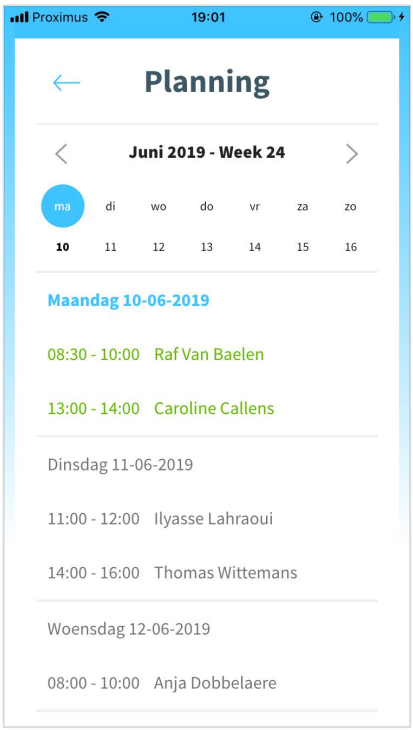
4.3.4 HomeScreen Widget

Als alle Widgets gemaakt zijn die thuishoren op de landingspagina, kunnen ze verzameld worden in één Widget, genaamd HomeScreen (zie bijlage D). Om de Widgets op schermen met verschillende afmetingen min of meer hetzelfde te positioneren is er in de HomeScreen Widget gebruik gemaakt van de Expanded Widget. De landingspagina kan verticaal opgedeeld worden in drie delen: de Logo Widget, de HomeScreenMenu Widget en de Afmelden Widget. Elk van deze Widgets moet in een Expanded Widget gestoken worden zodat de verdeling, die aan deze Expanded Widgets gegeven wordt met het *flex property,* behouden blijft in alle verschillende schermhoogtes. Het id, die in de HomeScreen gedefinieerd is, kan worden gebruikt om te navigeren naar dit scherm na het registreren of inloggen.

4.3.5 AfsprakenScreen Widget

Nu bekend is dat het maken van een scherm niet meer is dan het samen nemen van Widgets in Multi-Child Layout Widgets, zoals de Row en Column Widget, gaat er niet meer dieper worden ingegaan op de bouwstenen van een specifiek scherm.

Op de afsprakenpagina is duidelijk te zien dat het met Flutter mogelijk is om figuur 21 na te bouwen. Deze pagina wordt ook, net als de landingspagina, verticaal opgedeeld in drie delen, namelijk de titel, de weekkalender en de afspraken. In tegenstelling tot de landingspagina, waar elk van de drie Widgets in een Expanded Widget werden gewrapt, wordt hier enkel de AfspraakDagen Widget in een Expanded Widget gestoken. Dit maakt dat hoe groter het scherm in de hoogte is, hoe meer afspraken er tegelijk op het scherm zichtbaar zijn zonder te scrollen. De titel en de weekkalender blijven dezelfde hoogte behouden op alle schermen.



Figuur 38: Afsprakenpagina

4.3.6 StatefulWidgets vs. StatelessWidgets

Voorheen werd er enkel gebruikgemaakt van StatelessWidgets. Voor de AfsprakenScreen Widget (zie figuur 39) is voor het eerst nood aan een StatefulWidget.

```
10  class AfsprakenScreen extends StatefulWidget {
11    static const String id = "afspraken_screen";
12
13    @override
14    _AfsprakenScreenState createState() => _AfsprakenScreenState();
15  }
```

Figuur 39: AfsprakenScreen Widget

De HomeScreen Widget is een StatelessWidget omdat al de Widgets in de Widget Tree van HomeScreen ook StatelessWidgets zijn. Dit wil zeggen dat alle Widgets onder HomeScreen nooit visueel zullen veranderen door interactie van de gebruikers. De ontwikkelaar kan uiteraard een andere implementatie geven aan de methodes die uitgevoerd worden als op de knoppen gedrukt wordt op de HomeScreen Widget, maar deze implementatie zal dan pas werken als een nieuwe versie van de app gedeployed wordt.

De AfsprakenScreen Widget bevat bijvoorbeeld de weekkalender. In figuur 38 staat de kalender op week 24. Als een gebruiker op de pijltjes rechts of links klikt, dan zal de tekst moeten veranderen naar respectievelijk week 25 of week 23. Het jaar en de maand moet uiteraard ook aangepast worden naar de maand en het jaar van de eerste dag van de week. (Dit was een vereiste van de Ell@-applicatie die tijdens de stage ontwikkeld werd.) Een StatefulWidget heeft een *createState*-methode die de staat van de Widget retourneert. De _AfsprakenScreenState Widget (zie bijlage E) is verantwoordelijk voor het initieel tekenen van de Widgets op het scherm en voor het opnieuw tekenen van de Widgets wanneer de staat veranderd wordt. De staat wordt aangepast telkens wanneer de methode *setState* wordt aangeroepen in de StatefulWidget. Bijvoorbeeld wanneer er op het pijltje rechts geklikt wordt, worden nieuwe afspraken geladen in hetzelfde scherm. Er wordt dus niet echt genavigeerd tussen schermen.

4.3.7 LoginScreen Widget

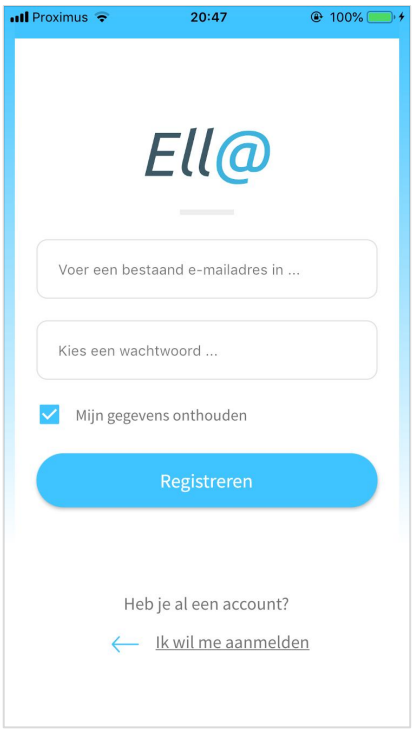
Om te voorkomen dat ongewenste bezoekers toegang hebben tot de database die gekoppeld is aan de applicatie, is er een loginpagina (zie figuur 40) voorzien. Met behulp van de *package* shared_preferences is het mogelijk om het e-mailadres en het wachtwoord te onthouden als daarvoor de checkbox aangevinkt is. Ook dit scherm is verticaal opgedeeld in drie delen: het logo, de tekstvakken (inclusief de checkbox en de knop) en de tekst onderaan voor het navigeren naar de registratiepagina. Door dezelfde verhouding te gebruiken tussen de drie delen, met name het eerste deel waarin het logo zit, als in figuur 36, komt het logo op exact dezelfde plaats te staan als vanuit de loginpagina genavigeerd wordt naar de landingspagina en omgekeerd.

4.3.8 RegistrationScreen Widget

Achteraf gezien zou van de LoginScreen en de RegistrationScreen Widget (zie figuur 41) één gezamenlijke widget gemaakt kunnen worden waarbij, op basis van een boolean- of een enumwaarde, de teksten en de achterliggende functies aangepast worden. Nu zijn het twee aparte Widgets waarbij, met behulp van een Hero Widget over de hele pagina, de indruk wordt gegeven dat er niet genavigeerd wordt naar een ander scherm.

Er kan standaard met eenderwelk e-mailadres geregistreerd worden zolang de @ en de punt de tekst verdelen in drie delen. Eventueel zou er meer controle gevoerd kunnen worden met een Firebase Authentication trigger. [40]

Figuur 40: Loginpagina



Figuur 41: Registratiepagina

4.4 Discussie

Het Flutter-team heeft op hun site, voor ontwikkelaars die van andere technieken en andere platformen willen overschakelen naar Flutter, voor elke overstap een pagina toegewijd om de gelijkenissen in kaart te brengen. [41] Dit maakt weeral dat men voor vragen over Flutter vanuit verschillende hoeken terechtkan op de site van Flutter.

De eerste keer dat een Widget Tree gemaakt wordt, kan afschrikkend zijn als men niet weet hoe men Widgets moet hergebruiken. Ondanks dat de stabiele versie van het Flutter-framework momenteel haar eerste verjaardag nog moet vieren, is er al een grote community van Flutter-developers actief die alle vragen omtrent Flutter van een hulpzaam antwoord weten te voorzien.

Als aanvulling op de Flutter-site is het Medium-platform een aanrader om interessante artikels te ontdekken over Flutter. [42] Ontwikkelaars delen er hun ervaringen over het implementeren van complexere technieken en schrijven vaak een handleiding over hoe ze tot een bepaald resultaat zijn gekomen of delen een link naar het resultaat zelf. De Flutter-community heeft zelfs een eigen ruimte gekregen op het Medium-platform waarop alle Flutter-artikels verzameld zijn. [42]

Voor hele specifieke problemen kan er hulp gevraagd worden op Gitter. Ook daar heeft Flutter een eigen kanaal gekregen waar ontwikkelaars elkaar helpen elkaars problemen op te lossen. Tot drie keer toe is voor dit onderzoek een oplossing geboden vanuit Gitter voor Flutter. Leden van het Flutter-team scrollen met regelmaat zelf door het kanaal en zien de community, realtime als het ware, het Flutter-framework onder de knie krijgen.

Het is voor het stagebedrijf zeker interessant omdat dit prototype in vier (!) dagen tijd is ontwikkeld door één ontwikkelaar. Vervelende kwesties met betrekking tot het plotseling verschuiven van elementen op het scherm komen minder voor dan bij andere programmeertalen. Er kan zich met andere woorden geen onverwacht gedrag voordoen als men weet waar men mee bezig is. Het kost slechts €10 om een bootcamp te volgen waarmee men zoals gezegd voldoende bagage heeft om van start te kunnen gaan als beginnende Flutter-developer.

Tijdens de stage werd gevraagd door de UX-designer of het niet mogelijk is om op zijn minst een gevoel van navigatie te voorzien in de PWA, in tegenstelling tot enkel het vernieuwen van de gegevens zonder transitie. Persoonlijk lijkt dit, ook met behulp van Angular, onbegonnen werk in HTML en CSS terwijl in Flutter hier hoogstwaarschijnlijk een Widget voor zou bestaan die kan zorgen voor de gewenste animatie.

Wat ook interessant is om te onderzoeken, is de *package* flutter_redux die het reduxpatroon in Flutter integreert. Voor webframeworks als VueJS, React en Angular bestaat respectievelijk VueX, Redux en NgRx voor het managen van de *state* van een applicatie. Naarmate een applicatie groter en ingewikkelder wordt, is het nodig om orde op zaken te houden en daar zou het reduxpatroon een handje bij kunnen helpen.

Conclusie

Werken in team vergt meer inspanning dan men denkt. Communicatie wordt vaak onderschat maar is een van de belangrijkste pijlers in het tot stand brengen van een succesvol project. Als communicatie gepaard gaat met ownership en toewijding van alle teamleden dan verdwijnen alle problemen als sneeuw voor de zon. Dit is hoofdzakelijk de conclusie van de stage binnen Ons.

Datzelfde ownership was ook aanwezig in het onderzoeksgedeelte omdat hier gestreefd werd naar het opleveren van een volwaardig eindresultaat in de hoop het stagebedrijf te overtuigen om in zee te gaan met de onderzochte technologie.

De voldoening van het eindresultaat en het overwinnen van de uitdagingen die aangegaan zijn bij het formuleren van de onderzoeksvraag, hebben een echte programmeur in mij naar boven gehaald waar ik voorafgaand aan deze stage nog van mezelf vond dat ik tekortkwam als programmeur. Ik heb geleerd op mezelf te zoeken naar oplossingen in verschillende hoeken en bij te leren over technieken, concepten en technologieën die mij interessant leken of die mij konden helpen realiseren wat ik wilde bereiken. In deze zoektocht merkte ik dat met de juiste ingesteldheid en voldoende wilskracht iedereen kan worden en zijn wie hij wil. Daarmee bedoel ik dat men zichzelf moet zien als de persoon die ze willen worden waarna de handelingen vanzelf komen.

Al vroeg in mijn stageperiode kreeg ik het besef dat de oplossing voor de verzorgenden binnen Landelijke Thuiszorg in mijn handen lag. Ik was dus geen huis-, tuin- en keukenprogrammeur maar ik moest wel degelijk een realistische oplossing bieden voor een realistisch publiek. Het fascineert mij dat wij als informatici de vaardigheden hebben om oplossingen te bieden voor eenderwelk technisch probleem en zie dan ook geen vuiltje aan de lucht dat de Ell@-app een groot succes en een hele verlichting zal worden voor haar gebruikers.

Bibliografie

[1] Cegeka, „kantoren - Cegeka,” Cegeka, [Online]. Available: https://www.cegeka.com/nl-be/kantoren. [Geopend 26 april 2019].

[2] Cegeka, „Jaarverslag - Cegeka,” Cegeka, [Online]. Available: https://annualreport.cegeka.com/. [Geopend 26 april 2019].

[3] Cegeka, „Offering - Cegeka,” Cegeka, [Online]. Available: https://www.cegeka.com/nl-be/it-experts/offering. [Geopend 26 april 2019].

[4] Cegeka, „Over Ons - Cegeka,” Cegeka, [Online]. Available: https://www.cegeka.com/nl-be/over-ons. [Geopend 26 april 2019].

[5] M. Swinnen, „Monique Swinnen, gedeputeerde Vlaams-Brabant,” [Online]. Available: http://www.moniqueswinnen.be/ell-landelijke-thuiszorg. [Geopend 3 maart 2019].

[6] L. Thuiszorg, „e-zorg,” Landelijke Thuiszorg, [Online]. Available: http://www.e-zorg.be/Portals/37/Users/004/16/6916/doc/FAQ%20Ell@%20v8.0.pdf. [Geopend 26 april 2019].

[7] Wikipedia, „Microsoft Visual Studio,” Wikipedia, [Online]. Available: https://nl.wikipedia.org/wiki/Microsoft_Visual_Studio. [Geopend 28 april 2019].

[8] Microsoft, „Visual Studio,” Microsoft, [Online]. Available: https://devblogs.microsoft.com/visualstudio/join-us-april-2nd-for-the-launch-of-visual-studio-2019/?utm_source=vs_developer_news&utm_medium=referral. [Geopend 28 april 2019].

[9] TutorialsTeacher, „TutorialsTeacher,” TutorialsTeacher, [Online]. Available: https://www.tutorialsteacher.com/webapi/what-is-web-api. [Geopend 9 6 2019].

[10] S. Overflow, „Developer Survey Results,” Stack Overflow, 2019. [Online]. Available: https://insights.stackoverflow.com/survey/2019?utm_source=Iterable&utm_medium=email&utm_campaign=dev-survey-2019#technology-_-most-popular-development-environments. [Geopend 28 april 2019].

[11] Microsoft, „DevBlogs,” 29 4 2015. [Online]. Available: https://devblogs.microsoft.com/visualstudio/build-2015-news-visual-studio-code-visual-studio-2015-rc-team-foundation-server-2015-rc-visual-studio-2013-update-5/. [Geopend 9 juni 2019].

[12] P. Bright, „Ars Technica,” 14 4 2016. [Online]. Available: https://arstechnica.com/information-technology/2016/04/visual-studio-code-editor-hits-version-1-has-half-a-million-users/. [Geopend 9 juni 2019].

[13] D. Gavigan, „Medium,” Medium, 2 4 2018. [Online]. Available: https://medium.com/the-startup-lab-blog/the-history-of-angular-3e36f7e828c7. [Geopend 9 juni 2019].

[14] A. Sterkowitz, „Youtube,” [Online]. Available: https://www.youtube.com/watch?v=VAkio68d51A. [Geopend 29 april 2019].

[15] V. Abdullayev, „Youtube,” [Online]. Available: https://www.youtube.com/watch?v=TacjOypXPXg. [Geopend 29 april 2019].

[16] T. Deschryver, „GitHub - contributors to NgRx-platform,” [Online]. Available: https://github.com/ngrx/platform/graphs/contributors. [Geopend 9 juni 2019].

[17] Azure, „Wat is DevOps?,” Azure, [Online]. Available: https://azure.microsoft.com/nl-nl/overview/what-is-devops/. [Geopend 28 april 2019].

[18] Cypress, „Cypress.io,” Cypress, [Online]. Available: https://www.cypress.io/. [Geopend 29 april 2019].

[19] C4model. [Online]. Available: https://c4model.com. [Geopend 9 juni 2019].

[20] Strato, „Strato,” Strato, [Online]. Available: https://www.strato.nl/sitebuilder/progressive-web-apps-wat-zijn-de-voordelen/#. [Geopend 25 maart 2019].

[21] Google, „Flutter,” Google, [Online]. Available: https://flutter.dev/. [Geopend 5 mei 2019].

[22] Flutter, „Github,” Google, [Online]. Available: https://github.com/flutter/samples/blob/master/INDEX.md. [Geopend 14 mei 2019].

[23] Flutter, „Flutter Cookbook,” Google, [Online]. Available: https://flutter.dev/docs/cookbook. [Geopend 14 mei 2019].

[24] F. Packages, Flutter, [Online]. Available: https://pub.dev/flutter. [Geopend 9 juni 2019].

[25] U. Packages, Flutter, [Online]. Available: https://flutter.dev/docs/development/packages-and-plugins/using-packages. [Geopend 9 juni 2019].

[26] „Flutter Scoring,” Flutter, [Online]. Available: https://pub.dev/help#scoring. [Geopend 9 juni 2019].

[27] F. Team, „Flutter Packages,” Flutter, 4 juni 2019. [Online]. Available: https://pub.dev/packages/local_auth. [Geopend 9 juni 2019].

[28] P. Kumar, „GitHub,” [Online]. Available: https://github.com/iampawan/FlutterAuthFaceID-FingerPrint. [Geopend 9 juni 2019].

[29] P. Kumar, „YouTube,” 9 februari 2019. [Online]. Available: https://www.youtube.com/watch?v=S1ta90cTxBA&t=23s. [Geopend 9 juni 2019].

[30] Firebase, „Firebase Authentication,” Google, [Online]. Available: https://firebase.google.com/docs/auth. [Geopend 9 juni 2019].

[31] „Flutter Firebase Auth,” Google, [Online]. Available: https://pub.dev/packages/firebase_auth. [Geopend 9 juni 2019].

[32] „Cloud Firestore,” Firebase, [Online]. Available: https://firebase.google.com/docs/firestore. [Geopend 9 juni 2019].

[33] „RTDB vs Firestore,” Firebase, [Online]. Available: https://firebase.google.com/docs/firestore/rtdb-vs-firestore. [Geopend 9 juni 2019].

[34] J. Birch, „FlutterDoc,” Medium, 6 november 2018. [Online]. Available: https://flutterdoc.com/loading-data-from-firestore-with-flutter-c42c520f6ee5. [Geopend 9 juni 2019].

[35] Firebase, „YouTube,” Google, 9 maart 2017. [Online]. Available: https://www.youtube.com/watch?time_continue=23&v=vr0Gfvp5v1A. [Geopend 10 juni 2019].

[36] C. Google, „Cloud Functions for Firebase,” Google, [Online]. Available: https://codelabs.developers.google.com/codelabs/firebase-cloud-functions/?utm_campaign=featureoverview_education_general_en_04-03-18&utm_source=Firebase&utm_medium=yt-desc#3. [Geopend 10 juni 2019].

[37] „MacOS install,” Flutter, [Online]. Available: https://flutter.dev/docs/get-started/install/macos. [Geopend 10 juni 2019].

[38] A. Yu, „AppBrewery,” AppBrewery, [Online]. Available: https://www.appbrewery.co/p/flutter-development-bootcamp-with-dart/. [Geopend 10 juni 2019].

[39] „Layout Widgets,” Flutter, [Online]. Available: https://flutter.dev/docs/development/ui/widgets/layout. [Geopend 10 juni 2019].

[40] „Firebase Authentication triggers,” Firebase, [Online]. Available: https://firebase.google.com/docs/functions/auth-events. [Geopend 10 juni 2019].

Bijlagen

A. RoundedRectangleButton Widget

B. HomeScreenMenu Widget

C. Afmelden Widget

D. HomeScreen Widget

E. _AfsprakenScreenState Widget

A. RoundedRectangleButton Widget

```
class RoundedRectangleButton extends StatelessWidget {
  final String text;
  final IconData icon;
  final Function onPressed;

  RoundedRectangleButton({@required this.text, @required this.icon, @required this.onPressed});

  @override
  Widget build(BuildContext context) {
    return SizedBox(
      height: 150.0,
      width: 150.0,
      child: Card(
        shape: RoundedRectangleBorder(
          side: BorderSide(color: Colors.grey.shade200, width: 2.0),
          borderRadius: BorderRadius.all(Radius.circular(10.0)),
        ), // RoundedRectangleBorder
        child: FlatButton(
          child: Padding(
            padding: EdgeInsets.symmetric(vertical: 25.0),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              crossAxisAlignment: CrossAxisAlignment.end,
              children: <Widget>[
                Icon(icon, color: Colors.lightBlueAccent),
                Hero(
                  tag: text,
                  child: Text(
                    text,
                    style: TextStyle(
                        color: Colors.lightBlueAccent,
                        fontFamily: "Source Sans Pro",
                        fontWeight: FontWeight.w700,
                        fontSize: 19.0), // TextStyle
                    ), // Text
                  ), // Hero
                ], // <Widget>[]
              ), // Column
            ), // Padding
          onPressed: onPressed,
        ), // FlatButton
      ), // Card
    ); // SizedBox
  }
}
```

B. HomeScreenMenu Widget

```dart
class HomeScreenMenu extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Column(
      mainAxisSize: MainAxisSize.min,
      children: <Widget>[
        Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            RoundedRectangleButton(
              icon: Icons.date_range,
              text: "Planning",
              onPressed: () async {
                var calendarData = await CalendarModel()
                    .getWekenMetDagen();

                Navigator.pushNamed(
                    context, AfsprakenScreen.id,
                    arguments: calendarData);
              },
            ), // RoundedRectangleButton
            SizedBox(
              width: 10.0,
            ), // SizedBox
            RoundedRectangleButton(
              icon: Icons.flag,
              text: "Afwezigheden",
              onPressed: null,
            ), // RoundedRectangleButton
          ], // <Widget>[]
        ), // Row
        SizedBox(height: 10.0),
        Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            RoundedRectangleButton(
              icon: Icons.mail_outline,
              text: "Berichten",
              onPressed: null,
            ), // RoundedRectangleButton
            SizedBox(
              width: 10.0,
            ), // SizedBox
            RoundedRectangleButton(
              icon: Icons.settings,
              text: "Instellingen",
              onPressed: null,
            ), // RoundedRectangleButton
          ], // <Widget>[]
        ), // Row
      ], // <Widget>[]
    ); // Column
  }
}
```

C. Afmelden Widget

```
class Afmelden extends StatelessWidget {
  final _auth = FirebaseAuth.instance;

  @override
  Widget build(BuildContext context) {
    return FlatButton(
      padding: EdgeInsets.all(0),
      child: Padding(
        padding: EdgeInsets.symmetric(vertical: 10.0, horizontal: 15.0),
        child: Row(
          mainAxisSize: MainAxisSize.min,
          children: <Widget>[
            Icon(
              Icons.input,
              color: Colors.lightBlueAccent,
              size: 30.0,
            ), // Icon
            SizedBox(width: 10.0),
            Text(
              "Afmelden",
              style: TextStyle(
                  color: Colors.lightBlueAccent,
                  fontFamily: "Source Sans Pro",
                  fontWeight: FontWeight.w700,
                  fontSize: 21.0), // TextStyle
            ), // Text
          ], // <Widget>[]
        ), // Row
      ), // Padding
      onPressed: () async {
        await _auth.signOut();
        Navigator.popUntil(
          (context),
          ModalRoute.withName(LoginScreen.id),
        );
      },
    ); // FlatButton
  }
}
```

D. HomeScreen Widget

```dart
class HomeScreen extends StatelessWidget {
  static const String id = "home_screen";

  @override
  Widget build(BuildContext context) {
    return Container(
      decoration: kborderDecoration,
      child: Padding(
        padding: EdgeInsets.all(10.0),
        child: Container(
          color: Colors.lightBlueAccent,
          child: SafeArea(
            child: Scaffold(
              backgroundColor: Colors.white,
              body: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: <Widget>[
                  Expanded(
                    flex: 8,
                    child: Align(
                      alignment: Alignment.bottomCenter,
                      child: Logo(),
                    ), // Align
                  ), // Expanded
                  Expanded(
                    flex: 16,
                    child: HomeScreenMenu(),
                  ), // Expanded
                  Expanded(
                    flex: 3,
                    child: Afmelden(),
                  ), // Expanded
                ], // <Widget>[]
              ), // Column
            ), // Scaffold
          ), // SafeArea
        ), // Container
      ), // Padding
    ); // Container
  }
}
```

E. _AfsprakenScreenState Widget

```dart
class _AfsprakenScreenState extends State<AfsprakenScreen> {
  List<CalendarDate> wekenMetDagen = [];
  CalendarModel calendarModel = CalendarModel();

  vorigeWeek() async {…
  volgendeWeek() async {…
  @override
  void initState() {…
  @override
  Widget build(BuildContext context) {
    return Container(
      decoration: kborderDecoration,
      child: Padding(
        padding: EdgeInsets.all(10.0),
        child: Container(
          color: Colors.lightBlueAccent,
          child: SafeArea(
            child: Scaffold(
              backgroundColor: Colors.white,
              body: Padding(
                padding: EdgeInsets.symmetric(horizontal: 20.0),
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.center,
                  children: <Widget>[
                    TitleComponent(
                      text: 'Planning',
                    ), // TitleComponent
                    PlanningComponent(
                      wekenMetDagen: wekenMetDagen,
                      calendarModel: calendarModel,
                      vorigeWeek: () async {
                        await vorigeWeek();
                        setState(() {});
                      },
                      volgendeWeek: () async {
                        await volgendeWeek();
                        setState(() {});
                      },
                    ), // PlanningComponent
                    AfspraakDagen(
                      wekenMetDagen: wekenMetDagen,
                    ), // AfspraakDagen
                  ], // <Widget>[]
                ), // Column
              ), // Padding
            ), // Scaffold
          ), // SafeArea
        ), // Container
      ), // Padding
    ); // Container
  }
}
```