Afstudeerdossier

Project VoMo

Voorstroom Mobile

Inversable B.V.

Steven Verkuil – Stagebegeleider Inversable B.V. – Steven@inversable.com
Jeroen Bolhuis – Voorstroom – jeroen@voorstroom.nl

Saxion Deventer

Emiel Haarhuis – Afstudeerder – 452901@student.saxion.nl
Micheal de Louwere – Stagebegeleider Saxion Deventer – m.h.e.delouwere@saxion.nl

Deventer – 25 April 2022

Inhoudsopgave

1 Inleiding

Dit is het Afstudeerdossier voor de afstudeeropdracht van Emiel Haarhuis bij Voorstroom (deel van Inversable B.V). Hierin treft u een beknopte weergave van de werkzaamheden die tijdens de afstudeerperiode van 25 April 2022 tot en met 6 Februari 2023 zijn uitgevoerd.

Bij deze afstudeeropdracht is een app ontworpen en ontwikkeld. In dit dossier wordt het doorlopen proces beschreven en inzichtelijk gemaakt.

1.1 Leeswijzer

Deze leeswijzer dient ter verduidelijking van de structuur van dit afstudeerdossier. Er wordt kort beschreven welke informatie ieder hoofdstuk bevat.

Hoofdstuk 3 - Inversable en Voorstroom
In dit hoofdstuk wordt de achtergrond van Voorstroom en Inversable besproken. Daarnaast wordt hier een inleiding van de afstudeeropdracht gegeven, zowel met de beginsituatie als de gewenste eindsituatie.

Hoofdstuk 4 - Onderzoeksvragen
In dit hoofdstuk worden de specifieke voorwaarden voor de nieuwe app opgesteld, rekening houdend met zowel technische vereisten als gebruikersbehoeften. Er zal ook een hoofdvraag worden opgesteld met bijbehorende deelvragen. Deze deelvragen zijn gekoppeld aan de beroepscompetenties van het HBO-I 25-vlaksmodel.

Hoofdstuk 4.1 - Deelvraag 1: Aan welke eisen moet VoMo voldoen?
Het doel van de eerste deelvraag is het vaststellen van alle vereisten en de randvoorwaarden voor VoMo. Hierbij wordt er een requirementsanalyse gemaakt waarbij de vereisten van de app worden verwerkt in ‘User Requirements’, ‘Technical Requirements’ en de ‘Software Requirements’.

Hoofdstuk 4.2 - Deelvraag 2: Hoe ontwerp je een geschikte User Interface voor VoMo?
De opgestelde requirements van Deelvraag 1 worden vertaald worden naar een interface die gebruiksvriendelijk is en er mooi uitziet op een groot aantal verschillende apparaten en schermgroottes. Om hier zo goed mogelijk aan te voldoen is er een ‘Low-Fi’ en vervolgens een ‘High-Fi’ Ontwerp gemaakt. Hierbij worden ook de ontwerpkeuzes verder onderbouwd.

Hoofdstuk 4.3 - Deelvraag 3: Welk hybride framework is het meest geschikt voor het ontwikkelen van VoMo?
In dit hoofdstuk wordt besproken dat voordat er begonnen kan worden met het maken

van de VoMo app, er eerst een geschikt Hybrid Framework gekozen moet worden. Er zal een kort onderzoek gedaan worden door een Longlist op te stellen met potentieel geschikte Hybrid Frameworks en vervolgens een Shortlist met de meest geschikte Frameworks. De Shortlist zal geïnventariseerd en beoordeeld worden. Waarna er in overleg voor een Framework gekozen zal worden.

Hoofdstuk 4.4 - Deelvraag 4: Wat is de beste manier om VoMo technisch vorm te geven en te laten integreren met alle huidige systemen die gebruikt worden door VoMo?

De vierde deelvraag gaat over hoe de technische structuur van de Voorstroom Mobile Applicatie eruit gaat zien en hoe deze geïntegreerd wordt met de huidige systemen van Voorstroom. Er wordt gekeken naar de benodigde wijzigingen in de backend structuur om de nieuwe app te faciliteren. Daarnaast worden er een aantal 'best practices' in programmeerstijl benoemd om ervoor te zorgen dat de code van de app van voldoende kwaliteit is.

Hoofdstuk 5 – Conclusie en Reflectie

Een persoonlijke reflectie op het afstudeertraject en de opgeleverde producten.

Hoofdstuk 6 - Aanbevelingen

Dit hoofdstuk bevat een aantal aanbevelingen voor de verdere ontwikkeling van VoMo. Deze richten zich op alles wat nog gedaan of verbeterd moet worden voordat de app uitgebracht kan worden.

2 Afkortingen en Terminologie

Naam	Beschrijving
VoMo	Voorstroom Mobile, de nieuwe vervangende Hybrid Mobile applicatie die ontwikkeld wordt.
Hybrid Framework	Programmeer framework waarmee tegelijkertijd voor meerdere besturingssystemen kan ontwikkeld worden.
API	Application Programming Interface
iOS	Iphone Operating System
URL	Uniform Resource Locators
DM's	Direct Messages
API Endpoint	De URL waarmee een API-Call kan worden gedaan.
Requirement	Vereiste functionaliteit van Software.
REST	Representational State Transfer, een stijl voor het structureren van API-Calls.
JSON	Javascript Object Notation, een manier voor het structureren van de data in API-Calls.
JWT	JSON Web Token, een veelgebruikte authenticatiemethode.
Requirement	Vereiste functionaliteit van de nieuwe oplossing.
MoSCow	Must have, Should have, Could have en Won't have. Manier om prioriteit te geven aan Requirements.
DM	Direct Message.
CRM	Customer Relations Management.
UI	User Interface (gebruikersinterface)
UX	User Experience (gebruikerservaring)
Enterprises	Zeer grote (tech)bedrijven
AVG	Algemene verordening gegevensbescherming

3 Inversable en Voorstroom

Inversable B.V. is een bedrijf opgericht in 2017 door Steven Verkuil en Erwin Bisschop in Deventer, gehuisvest in een oude gasfabriek. Naast Voorstroom bieden zij ook andere producten aan, zoals Hanzenet, IntoAgri en de SEN-app.

Voorstroom is een software die sinds februari 2017 beschikbaar is, ontwikkeld om te helpen bij het opzetten en laten groeien van lokale duurzame energiebedrijven. Er zijn momenteel ongeveer 6.000 deelnemers aan 45 energiecollectieven die gebruik maken van Voorstroom.

3.1 Producten

Voorstroom biedt onder andere het Voorstroom CRM-systeem aan, speciaal ontworpen voor het beheren van energiecollectieven. Er is ook een bijbehorende Android en iOS-applicatie beschikbaar voor de particuliere deelnemers van deze energiecollectieven, waarmee bijvoorbeeld de opbrengst van de deelname in te zien is.

3.2 Opdrachtomschrijving

Echter, de huidige app dateert uit 2015. Bovendien is het gebouwd met React-Native, een ontwikkelingsframework dat Voorstroom niet langer wil gebruiken. Emiel's afstudeeropdracht bestaat uit het ontwerpen en ontwikkelen van een nieuwe applicatie die de oude vervangt. Hierbij is er ruimte voor het ontwerpen van de software en een gebruiksvriendelijke interface die past binnen de rest van de producten van Voorstroom en Inversable.

3.3 Beginsituatie

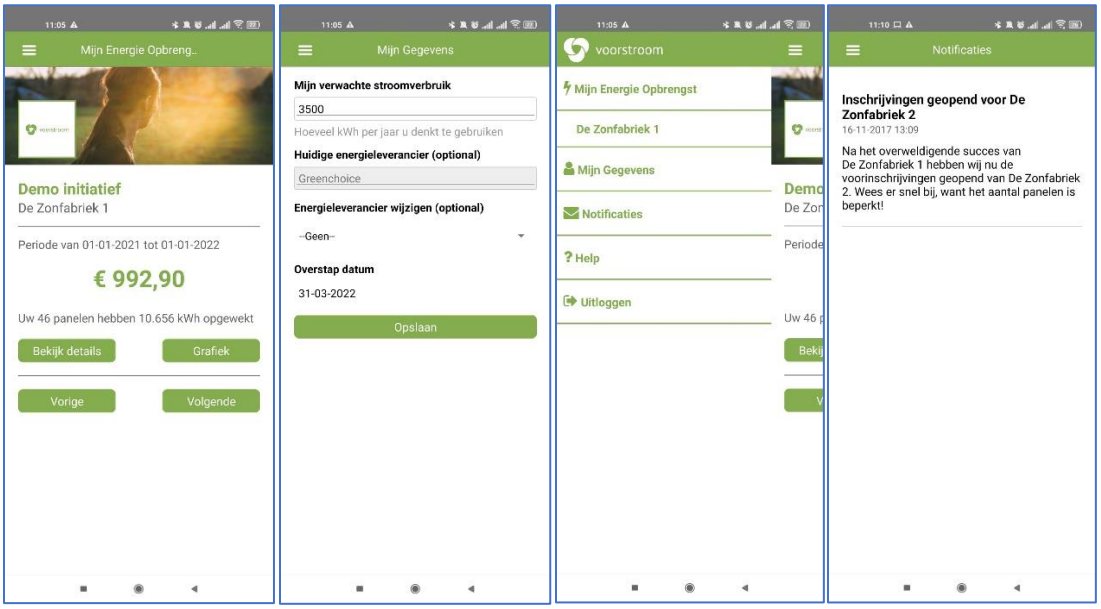
Voorstroom is een software die helpt bij het beheren van energiecollectieven. Het is ontwikkeld om lokale energiecollectieven te professionaliseren en te ondersteunen bij groei.

Een lokaal energiecollectief is een groep burgers of bedrijven die samen werken aan de productie, distributie en consumptie van duurzame energie in hun eigen gemeenschap. Dit kan bijvoorbeeld door het opzetten van een coöperatie, het aankopen van zonnepanelen of windturbines, of het ontwikkelen van een gemeenschappelijk energieopslagsysteem.

Hiervoor zijn verschillende softwareproducten beschikbaar, waaronder een Hybrid applicatie die particulieren kunnen gebruiken die zijn ingeschreven bij een energiecollectief. Met deze applicatie kan men bijvoorbeeld zien hoeveel zelf opgewekte stroom er is en hoeveel vergoeding er is, wijzigingen in de persoonlijke situatie doorgeven, en notificaties lezen die door de energiecoöperatie worden verstuurd.

De huidige applicatie is gemaakt met het React-Native framework, maar Voorstroom wil niet langer gebruik maken van dit framework.

Huidige Applicatie



3.4 Eindsituatie

Er moet een nieuwe Hybride applicatie worden ontwikkeld met een modern hybride framework, waarbij alle huidige functionaliteit behouden moet blijven en nieuwe optionele functionaliteit toegevoegd kan worden.

Een hybride framework is een type softwareontwikkelings-framework waarmee softwareontwikkelaars applicaties kunnen maken die draaien op meerdere platformen (zoals Android, iOS en Web) met *één* codebase.

Hierbij zijn een aantal gewenste verbeteringen:

- De UI en de UX van de applicatie moet ook worden herzien om de weergave van meerdere energiecollectieven op een gebruikersvriendelijke en overzichtelijke manier te ondersteunen. Er is ook een andere stagiair aanwezig die onderzoek doet naar de wensen van energiecoöperaties voor de applicatie, het is verstandig om hiermee samen te werken bij het bepalen van de requirements.

- De backend structuur van de applicatie is deels aanwezig, maar er moeten ook delen worden toegevoegd of uitgebreid en deze keuzes moeten wel onderbouwd zijn en de opgeleverde code moet van voldoende kwaliteit zijn. Een uitbreiding zou bijvoorbeeld zijn dat de gebruiker zelf zijn persoonlijke gegevens kan doorgeven waarbij ook rekening wordt gehouden met de AVG. De AVG (Algemene Verordening Gegevensbescherming) is een Europese wet die de bescherming van persoonlijke gegevens regelt voor organisaties die gegevens verwerken.

- Er is ook de wens om gebruikers (deels) toegang te geven tot hun eigen “file storage” bestaande uit noodzakelijke documenten voor de deelname in een energiecollectief. Dit vereist een aanpassing aan de huidige API. Ook wordt er gekeken naar een manier om gebruikers toegang te geven tot het “eloket” via DM's voor vragen en eventuele aanvragen. Het is nodig om de huidige backend structuur aan te passen voor de ontvangst van notificaties voor DM's.

3.5 Doelstellingen

Dit project heeft voor Voorstroom de volgende doelen:

1. Het bepalen van alle vereisten voor de nieuwe VoMo app. Dit wordt verwerkt in het document “Requirementsanalyse VoMo”.

2. Research naar hybrid frameworks. Het resultaat hiervan is een lijst met Hybrid Platformen die het meest geschikt zijn voor gebruik door Inversable, met hun belangrijkste eigenschappen.

3. Het maken van een UX/UI Design voor VoMo waarmee alle requirements op een mooie en gebruiksvriendelijke manier vormgegeven worden. Dit wordt gedocumenteerd in het document “UX/UI Design Document VoMo”.

4. Het uitvoerig testen van het nieuwe UX/UI Design voor VoMo met gebruikerstesten. Hierbij kan ook het UX/UI Design deels herzien worden. De uitslagen van de gebruikerstesten worden gedocumenteerd in het “User Testing VoMo Document”.

5. Het opstellen van een “Technisch Ontwerp VoMo”. Dit bevat het Technische Ontwerp voor de VoMo app en hoe dit integreert met de bestaande Voorstroom systemen.

6. Een prototype van de nieuwe VoMo applicatie.

3.6 Requirementsanalyse

Requirementsanalyse in software engineering is het opstellen van de vereisten van een nieuw (of gewijzigd) product. Hierbij wordt er rekening gehouden met mogelijke tegenstrijdige vereisten van alle betrokken stakeholders. [1]

3.6.1 Wat is een requirement?

Een requirement in Softwareontwikkeling beschrijft een gewenste functionaliteit van het product of dienst. Er zijn een aantal manieren om een requirement te documenteren.

Er is in dit geval gekozen voor het opstellen van User Stories en System Requirements. Deze zijn vervolgens omgezet in software requirements.

- User Stories zijn geschreven vanuit het perspectief van de eindgebruiker. Het specificeert wat de gebruiker verwacht dat de software kan doen.

- System Requirements zijn geschreven vanuit het perspectief van de ontwikkelaar, systeembeheerder of opdrachtgever. Dit specificeert vaak op welke manier de functionaliteiten geïmplementeerd zijn.

- Een software requirement is een verklaring van een functie of kenmerk dat een systeem of een product moet bezitten of een beperking waaraan het systeem of product moet voldoen.

  Dit kan bijvoorbeeld een functie of een prestatie-eis zijn of een beperking op een gebied zoals veiligheid, betrouwbaarheid of gebruiksgemak.

Er wordt ook een prioriteit aan de requirements gegeven. Bij dit project is ervoor gekozen om dit met MoSCoW te doen. Onder iedere requirement staat een korte verdere toelichting.

3.7 Projectteam

Het project wordt uitgevoerd door afstudeerder Emiel Haarhuis. Hij zal hierbij begeleid worden door Steven Verkuil van Inversable B.V. en Michael de Louwere van het Saxion.

Verdere betrokkenen zijn Jeroen Bolhuis (Business Developer Voorstroom) en Mark Versluis (Lead Developer).

Inversable

Naam	Inversable B.V.
Adres	Zutphenseweb 6B, Deventer
Postcode	7418 AJ
Telefoon	0640779277

Bedrijfsbegeleider

Naam	Steven Verkuil
Email	steven@inversable.com
Functie	Algemeen Directeur
Telefoon	0616841386

Afstudeerbegeleider Saxion Deventer

Naam	Micheal de Louwere
Email	m.h.e.delouwere@saxion.nl
Telefoon	+31880196308

Student

Naam	Emiel Haarhuis
Adres	Stoppelbergweg 26, Beekbergen
Postcode	7361TE
Email	452901@student.saxion.nl
Telefoon	0637272170

4 Onderzoeksvragen

In dit hoofdstuk zal in dieper detail ingegaan worden op welke specifieke voorwaarden er gesteld moeten worden aan de nieuwe app. Hierbij zal rekening gehouden worden met zowel de technische vereisten als de gebruikersbehoeften.

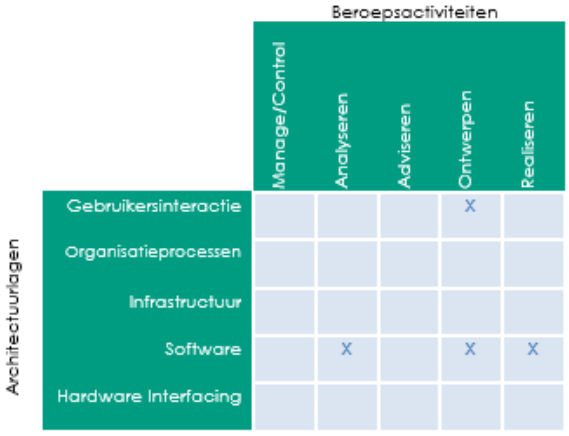
Daarnaast zal er in dit hoofdstuk de hoofdvraag met bijbehorende deelvragen opgesteld worden. Hierdoor kan gericht onderzoek worden gedaan en de antwoorden van de deelvragen zullen uiteindelijk leiden tot een antwoord op de hoofdvraag.

De hoofdvraag luidt als volgt:
Hoe ontwerp en ontwikkel je een hybride applicatie voor Voorstroom.nl?

Met hierbij de volgende 4 deelvragen die gekoppeld zijn aan de beroepscompetenties van het HBO-I 25-vlaksmodel:

Hoofdvraag Hoe ontwerp en ontwikkel je een hybride applicatie voor Voorstroom.nl?	Software - Realiseren
Deelvraag 1 Aan welke eisen moet VoMo voldoen?	Software - Analyseren
Deelvraag 2 Hoe ontwerp je een geschikte User Interface voor Vomo?	Gebruikersinteractie - Ontwerpen
Deelvraag 3 Welk Hybride Framework is het meest geschikt voor het ontwikkelen van VoMo?	Software - Analyseren
Deelvraag 4 Wat is de beste manier om VoMo technisch vorm te geven en te laten integreren met alle huidige systemen die gebruikt worden voor Voorstroom.nl	Software - Ontwerpen

Hiermee wordt op de volgende manier het HBO-I 25-vlaksmodel ingevuld:



Figuur 1: HBO-i 25-vlaksmodel 2018

4.1 Deelvraag 1: Aan welke eisen moet VoMo voldoen?

Het doel van de eerste deelvraag is het vaststellen van alle vereisten en de randvoorwaarden voor VoMo.

In overleg met Jeroen Bolhuis en Mark Buis van Voorstroom wordt er een requirementsanalyse gemaakt. Hierbij worden de vereisten van de app verwerkt in ‘User Requirements’, ‘Technical Requirements’ en de ‘Software Requirements’.

Aan deze requirements worden vervolgens met MoSCoW prioriteit toegekend.

4.1.1 Requirements

User Stories

Must have	**Should have**	**Could have**	**Won’t have**

ID	**Requirement**	**MoSCoW**
U01	*Als gebruiker van VoMo wil ik een “Help en Info pagina” hebben waar ik contact-en-algemene informatie van Voorstroom en het collectief kan vinden.*	Must have
Toelichting U01 Het doel van deze pagina is het makkelijk maken om contactgegevens van zowel Voorstroom of het energiecollectief te vinden. Dit maakt het gemakkelijk voor de gebruiker om contact op te nemen als er zich problemen voordoen. In de huidige app zijn er maar beperkt contactgegevens van het collectief beschikbaar.		
Software Requirements U01 1. De app moet een pagina bevatten met contact- en algemene informatie over Voorstroom en het energiecollectief. 2. De pagina moet een contactformulier bevatten waarmee de gebruiker vragen of opmerkingen kan sturen naar Voorstroom of het energiecollectief. 3. De pagina moet een overzicht bevatten van relevante links naar bijvoorbeeld de website van Voorstroom, het energiecollectief, of andere informatiebronnen. 4. De pagina moet een zichtbaar en gemakkelijk te bereiken element hebben dat de gebruiker naar de "Help en Info pagina" kan leiden vanuit de rest van de app.		
U02	*Als gebruiker van VoMo wil ik mijn verwachte energieverbruik kunnen inzien en wijzigen.*	Must have
Toelichting U02 Om een inschatting te kunnen maken van de opbrengst voor de gebruiker is het noodzakelijk om hun verwachte energieverbruik per jaar te weten. Zonder kan in veel		

gevallen niet een correcte berekening gemaakt worden.

Software Requirements U02

1. De app moet een pagina bevatten waar de gebruiker hun verwachte energieverbruik per jaar kan invoeren.
2. De gebruiker moet in staat zijn om hun verwachte energieverbruik te wijzigen wanneer hun verbruikspatroon verandert.
3. De app moet de gebruiker waarschuwen als hun ingevoerde verwachte energieverbruik afwijkt van hun historisch verbruik of als het niet realistisch is in vergelijking met gemiddelde verbruiksniveaus voor huishoudens met vergelijkbare omstandigheden.
4. De app moet de gebruiker in staat stellen om hun verwachte energieverbruik te delen met anderen (bijv. met het energiecollectief) via een link of screenshot.

U03	*Als gebruiker van VoMo wil ik mijn huidige energieleverancier kunnen inzien en wijzigen met een overstapdatum.*	Must have

Toelichting U03
In het verleden werd er gebruik gemaakt van de postcoderoosregeling. Hierbij krijgt een participant een deel van de energiebelasting terug. De verantwoordelijke hiervoor is de energieleverancier, vandaar dat de gebruiker dit moet kunnen aangeven.

Oude projecten blijven nog lang van de postcoderoosregeling gebruik maken. Sommige projecten hebben een looptijd van 15 jaar. Bij nieuwe projecten is het soms niet nodig dat de energieleverancier gewijzigd kan worden.

Software Requirements U03

1. De app moet een pagina bevatten waar de gebruiker hun huidige energieleverancier en de overstapdatum ervan kan zien.
2. De gebruiker moet in staat zijn om hun energieleverancier te wijzigen en een overstapdatum op te geven wanneer ze van leverancier willen veranderen.
3. De app moet de gebruiker informeren over eventuele kosten die verbonden zijn aan het wijzigen van de energieleverancier.
4. De app moet de gebruiker in staat stellen om hun energieleverancier te delen met het energiecollectief.
5. De app moet de gebruiker waarschuwen als de opgegeven overstapdatum niet valt binnen de looptijd van het huidige contract met de energieleverancier.
6. De app moet de gebruiker in staat stellen om aan te geven of ze gebruik maken van de postcoderoosregeling.

U04	*Als gebruiker van VoMo wil ik een overzichtelijke weergave van mijn deelname aan een (of meer) energiecollectieven* *O Tonen opwekking in het verleden.* *O Tonen prognose en huidige opwekking.* *O Tonen cumulatieve opbrengsten.*	Must have

Toelichting U04
Dit is waarschijnlijk de voornaamste reden om de app te gebruiken. Gebruikers willen graag hun prognose, huidige opwekking en verkregen opbrensten inzien.

Software Requirements U04

1. De app moet een pagina bevatten waar de gebruiker hun deelname aan energiecollectieven kan bekijken.
2. De app moet een overzicht geven van de opwekking van de gebruiker in het verleden.
3. De app moet de huidige opwekking van de gebruiker tonen.
4. De app moet een cumulatief overzicht geven van de opbrengsten van de gebruiker.

U05	*Als gebruiker van VoMo wil ik het aantal Units in mijn bezit kunnen zien per project (en eventueel per Collectief).*	Must have

Toelichting U05
Het aantal Units is momenteel het aantal zonnepanelen. Maar het zou in de toekomst bijvoorbeeld ook voor windmolens gebruikt kunnen worden.

Software Requirements U05

1. De app moet het aantal Units dat de gebruiker bezit weergeven voor elk project en, optioneel, voor elk Collectief.
2. De Units moeten worden geassocieerd met het corresponderende project of Collectief.
3. De app moet de gebruiker in staat stellen om het totale aantal Units dat zij bezitten te bekijken.
4. De app moet de mogelijkheid bieden om Units te volgen voor verschillende soorten hernieuwbare energiebronnen, zoals zonnepanelen en windturbines.

U06	*Als gebruiker van VoMo wil ik mijn persoonlijke energieproductie van alle projecten waar ik aan meedoe in kunnen zien.*	Must have

Toelichting U06
Een gebruiker kan meedoen met meerdere projecten per collectief. De ‘persoonlijke energieproductie’ refereert naar de opbrengst van de zonnepanelen (units) die in bezit zijn van de gebruiker.

Software Requirements U06

1. De app moet een overzicht tonen van de persoonlijke energieproductie van de gebruiker voor alle projecten waaraan zij deelnemen.
2. De app moet de opbrengst van de zonnepanelen van de gebruiker (units) weergeven.
3. De app moet de mogelijkheid bieden om de persoonlijke energieproductie van de gebruiker per project te bekijken.

U07	*Als gebruiker wil ik graag mijn besparing per periode in kunnen zien. Dit bestaat uit:* *O Teruggave Energiebelasting* *O Korting op leveringstarief elektriciteit* *O Onderhoudsbijdrage*	Must have

Toelichting U07
Hoe precies de besparing per periode berekend wordt verschilt per project en collectief. In sommige gevallen bestaat dit uit meer-of-minder onderdelen.

Software Requirements U07

1. De applicatie moet in staat zijn om de teruggave van de energiebelasting te berekenen en te tonen aan de gebruiker.
2. De applicatie moet in staat zijn om de korting op het leveringstarief van elektriciteit te berekenen en te tonen aan de gebruiker.
3. De applicatie moet in staat zijn om de onderhoudsbijdrage te berekenen en te tonen aan de gebruiker.
4. De applicatie moet een overzicht kunnen tonen van de besparingen per periode, inclusief de teruggave van energiebelasting, korting op het leveringstarief en eventuele onderhoudsbijdrage.

U08	*Als gebruiker van VoMo wil ik graag notificaties van de energiecoöperaties via de app kunnen inzien.*	Must have

Toelichting U08
Energiecoöperaties kunnen de gebruiker notificaties sturen.

Deze bestaan uit bijvoorbeeld nieuwsbrieven, updates over het project of verzoeken tot informatie.

Software Requirements U08

1. De app moet een manier bieden om notificaties van energiecoöperaties te ontvangen.
2. De app moet een gebruiker in staat stellen om de notificaties te bekijken.
3. De app moet een overzicht bieden van alle ontvangen notificaties, met informatie zoals het onderwerp en de datum van verzending.
4. De app moet een manier bieden om notificaties te markeren als gelezen of ongelezen.
5. De app moet een manier bieden om notificaties te verwijderen.
6. De app moet de notificaties op een logische manier presenteren, bijvoorbeeld in chronologische volgorde of gegroepeerd per energiecoöperatie.

U09	*Als gebruiker van VoMo wil ik pushnotificaties kunnen ontvangen.*	Must have

Toelichting U09
Een pushnotificatie is een geautomatiseerd bericht aan de gebruiker. Deze wordt ook gestuurd als de applicatie niet geopend is.

Dit kan bijvoorbeeld als de gebruiker een bericht van hun energiecoöperatie heeft ontvangen. Het is ook handig om aan de gebruiker mede te delen dat er een update beschikbaar is voor de app.

Software Requirements U09

1. De app moet pushnotificaties kunnen sturen naar de gebruiker, zowel wanneer de app geopend is als wanneer deze gesloten is.
2. De app moet in staat zijn om automatisch berichten naar de gebruiker te sturen, bijvoorbeeld in reactie op een bericht van de energiecoöperatie of om aan te geven dat er een update beschikbaar is.
3. De app moet een manier bieden om pushnotificaties in te stellen of uit te schakelen, zodat gebruikers deze naar eigen wens kunnen beheren.

U10	*Als gebruiker wil ik kunnen inloggen om toegang tot de VoMo app te verkrijgen.*	Must have

Toelichting U10
In de app wordt er omgegaan met gevoelige gebruikersgegevens. Het is noodzakelijk dat deze goed afgeschermd zijn van andere gebruikers of kwaadwillenden.

Daarom moet de applicatie beveiligd worden met een loginsysteem, de gebruiker kan

inloggen met zijn email en wachtwoord.

Software Requirements U10

1. De applicatie moet een loginpagina bevatten waar de gebruiker zijn email en wachtwoord kan invoeren.
2. De applicatie moet de ingevoerde gegevens verifiëren tegen de opgeslagen gegevens in de database.
3. Als de ingevoerde gegevens correct zijn, moet de gebruiker toegang krijgen tot de app.
4. Als de ingevoerde gegevens onjuist zijn, moet de gebruiker een foutmelding ontvangen.
5. De applicatie moet beveiligd zijn tegen brute force-aanvallen op het inlogsysteem.
6. De applicatie moet een wachtwoordherstelfunctie bevatten voor gebruikers die hun wachtwoord zijn vergeten.
7. De gebruiker moet zich kunnen uitloggen van de applicatie.

U11	*Als gebruiker wil ik mijn persoonlijke gegevens in kunnen zien.*	Must have

Toelichting U11
Voor het correct deelnemen aan een energiecollectief en voor het correct functioneren van de app zijn er een aantal persoonlijke gegevens van de gebruiker nodig.

Het is noodzakelijk dat de gebruiker deze kan inzien en controleren.

Software Requirements U11

- De app moet een pagina hebben waar de gebruiker hun persoonlijke gegevens kan bekijken en wijzigen.
- De app moet de gebruiker de mogelijkheid geven om hun emailadres te wijzigen.
- De app moet de gebruiker de mogelijkheid geven om hun persoonlijke gegevens, zoals hun naam en adres, te wijzigen.
- De app moet de gebruiker de mogelijkheid geven om hun telefoonnummer te wijzigen.

U12	*Als gebruiker van VoMo wil ik de artikelen op de Kennisbank van Voorstroom.nl kunnen lezen.*	Should have

Toelichting U12
Het Nederlandse systeem met deelname aan lokale energiecollectieven is vrij complex.

De meeste gebruikers zullen niet precies weten hoe het allemaal werkt.

Daarom is de kennisbank van voorstroom.nl in het leven geroepen. In de kennisbank is allerlei informatie hierover te vinden. Het zou handig zijn als de gebruiker deze vanuit de app zou kunnen bekijken.

Software Requirements U12

1. De VoMo app moet een optie bieden om de Kennisbank van Voorstroom.nl te bekijken.

2. De Kennisbank moet toegankelijk zijn vanuit de VoMo app, zonder dat de gebruiker hiervoor naar een andere website of app moet gaan.

3. De VoMo app moet de Kennisbank op een overzichtelijke en gebruiksvriendelijke manier weergeven, zodat de gebruiker gemakkelijk de gewenste informatie kan vinden.

U13	*Als gebruiker wil ik een mooie en uitgebreide visualisatie van mijn opbrengsten en prognoses voor energiecollectieven.*	Should have

Toelichting U13
Dit is waarschijnlijk de voornaamste reden om de app te gebruiken. Gebruikers willen graag hun prognose, huidige opwekking en verkregen opbrensten inzien.

Het is een bonus als dit op een mooie manier gevisualiseerd kan worden.

Software Requirements U13

1. Een visualisatiemodule moet geïmplementeerd worden in de app om de opbrengsten en prognoses van de gebruiker te tonen.

2. De visualisatie moet duidelijk en overzichtelijk zijn, zodat de gebruiker snel en gemakkelijk inzicht kan krijgen in hun opbrengsten en prognoses.

U14	*Als gebruiker van VoMo wil ik al mijn relevante documenten via de App kunnen inzien.*	Should have

Toelichting U14
Voor deelname aan een energiecollectief zijn er een aantal documenten nodig:

O Participatieovereenkomst
O Facturen
O Inschrijvingen
O Aanvragen

Het zou handig zijn als de gebruiker deze direct vanuit de app in zou kunnen zien.

Software Requirements U14

1. De app moet een module bevatten waarbinnen de gebruiker al zijn relevante documenten kan inzien.

2. Deze module moet toegang bieden tot de participatieovereenkomst, facturen, inschrijvingen en aanvragen van de gebruiker.

3. De module moet een overzichtelijke weergave bieden van de verschillende documenten, zodat de gebruiker gemakkelijk kan navigeren en zoeken naar specifieke documenten.

4. De app moet de mogelijkheid bieden om de documenten te downloaden of te delen met anderen, indien gewenst.

U15	*Als gebruiker van VoMo wil ik mijn verdere persoonlijke gegevens in kunnen zien en deze kunnen wijzigen.*	Should have

Toelichting U15
Voor het correct deelnemen aan een energiecollectief en voor het correct functioneren van de app zijn er een aantal persoonlijke gegevens van de gebruiker nodig.

Het is handig als de gebruiker deze gegevens kan controleren en wijzigen in de app. Dat scheelt de gebruiker een keer contact opnemen met voorstroom of het energiecollectief.

Software Requirements U15

1. Implementeer een scherm waar de gebruiker zijn persoonlijke gegevens kan inzien.

2. Maak het mogelijk voor de gebruiker om zijn persoonlijke gegevens te wijzigen.

3. Valideer de ingevoerde gegevens voordat ze worden opgeslagen om te voorkomen dat er onjuiste of onvolledige informatie wordt opgeslagen.

4. Voeg een bevestigingsmelding toe voordat de gebruiker zijn gegevens wijzigt, zodat hij zeker weet dat zijn wijzigingen zullen worden opgeslagen.

U16	*Als gebruiker van VoMo wil ik via de app toegang tot het Eloket met de optie om DM's te sturen.*	Could have

Toelichting U16
Eloket is energieloket. Hier kunnen gebruikers in contact komen met bijvoorbeeld energiecoaches. Het energieloket is momenteel nog niet operationeel. Waarschijnlijk wordt deze requirement daarom niet geimplementeerd.

Software Requirements U16

1. De VoMo app moet een integratie hebben met het Eloket, waarbij de gebruiker toegang

	heeft tot het Eloket en de optie heeft om direct messages te sturen naar de energiecoaches.	
U17	*Als gebruiker wil ik dat het het energiecollectief de kleur en stijl kan kan kiezen (whitelabel).*	Could have

Toelichting U17
Een whitelabel is een product of dienst gemaakt door een bedrijf die andere bedrijven rebranden zodat het lijkt of het door hun zelf gemaakt is.

In dit geval zou het betekenen dat het Initiatief zelf de logo's en kleuren van de app kan kiezen.

Software Requirements U17

- De app moet in staat zijn om logo's en kleuren te accepteren die door het energiecollectief worden opgegeven.
- De app moet in staat zijn om deze logo's en kleuren te tonen op de juiste plaatsen in de interface.
- Er moet een manier zijn voor het energiecollectief om de logo's en kleuren te uploaden en te wijzigen.
- Er moet duidelijkheid zijn over welke elementen van de app aangepast kunnen worden met de whitelabel-functie en welke niet.
- Er moet voldoende documentatie zijn om te beschrijven hoe het energiecollectief de whitelabel-functie kan gebruiken.

U18	*Als gebruiker wil ik mij direct via de app kunnen inschrijven en registreren.*	Won't Have

Toelichting U18
Het zou voor de gebruiker makkelijker zijn als zij zich direct via de app kunnen inschrijven bij een energiecollectief. Momenteel moet de gebruiker zich eerst inschrijven bij een energiecollectief en word vervolgens een uitnodiging voor de voorstroom app gemaakt.

Dit is een requirement die iedereen erg graag zou willen, maar het is erg complex op allerlei vlakken. Voor nu wordt hier dus ook nog niet verder naar gekeken.

Software Requirements U18

1. De app moet een formulier hebben waarmee gebruikers zich kunnen inschrijven en registreren bij een energiecollectief.
2. De app moet een manier hebben om de ingevoerde gegevens van gebruikers te valideren en op te slaan.

3. De app moet een manier hebben om gebruikers te informeren over de status van hun inschrijving (bijvoorbeeld wanneer hun aanvraag is goedgekeurd).

U19	*Als gebruiker van VoMo wil ik dat de app beschikbaar is in meerdere talen.*	Won't Have

Toelichting U19
Voor niet (goed) nederlands sprekende gebruikers zou het handig zijn als de app ook in andere talen beschikbaar is. Momenteel is deze groep echter zo klein dat deze requirement nog geen prioriteit heeft.

Software Requirements U19

1. De app moet een taalkeuze-optie hebben zodat gebruikers de taal van de app kunnen selecteren.

2. De app moet de benodigde vertalingen hebben. Dit kan door middel van professionele vertalers of door het gebruik van machinevertalingen.

3. De app moet op een consistente manier gebruik maken van de geselecteerde taal, inclusief alle teksten, knoppen, labels en andere elementen.

4. De benodigde ondersteuning voor het beheren van meerdere talen toe in de back-end van de app moet beschikbaar zijn, zodat nieuwe talen gemakkelijk toegevoegd kunnen worden in de toekomst.

U20	*Als gebruiker wil ik dat mijn meterstanden automatisch uitgelezen worden.*	Won't Have

Toelichting U20
Momenteel moet de gebruiker zelf zijn verwachte energieverbruik in de app invullen. Het zou handig zijn als dit automatisch voorspelt kan worden vanuit de huidige meterstand van de gebruiker.

Ook zou het de app extra functionaliteit kunnen bieden door de gebruiker ook zijn huidige energieverbruik in de app te tonen.

Helaas is het op afstand uitlezen van Meterstanden erg complex en heeft deze requirement daarom momenteel geen prioriteit.

System Requirements

Must have	Should have	Could have	Won't have

SR1	*Als systeembeheerder wil ik dat de VoMo app de gegevens uit de huidige relationele database haalt waar ook het CRM-systeem in*	Must have

	zit.	

Toelichting SR1
Momenteel is er bij Voorstroom een relationele database waar zowel het CRM-systeem als de app gebruik van maakt. De nieuwe app moet van dezelfde database gebruik maken.

Software Requirements SR1

1. Het ontwikkelen van een interface tussen de VoMo app en de relationele database.
2. Het testen van de functionaliteit van de API om ervoor te zorgen dat de app de juiste gegevens opvraagt uit de database.
3. Het implementeren van beveiligingsmaatregelen om ervoor te zorgen dat alleen bevoegde gebruikers toegang hebben tot de gegevens in de database.
4. Het opstellen van documentatie voor het gebruik van de API, zodat systeembeheerders weten hoe ze de app moeten configureren om gebruik te maken van de database.

SR2	*Als systeembeheerder wil ik dat de Backend van Voorstroom uitgebreid wordt zodat individuele persoonsgegevens opgehaald kunnen worden door de VoMo app.*	Must Have

Toelichting SR2
Momenteel kunnen persoonsgegevens niet direct aangepast worden vanuit de app. Dit kan alleen via het Managementdashboard.

De backend moet uitgebreid worden zodat deze individuele persoonsgegevens wel opgehaald en aangepast kunnen worden.

Software Requirements SR2

- Implementeren van een API voor het ophalen van individuele persoonsgegevens.
- Implementeren van een API voor het aanpassen van individuele persoonsgegevens.
- Toevoegen van functionaliteit in de app om individuele persoonsgegevens te bekijken en te wijzigen.
- Toevoegen van beveiligingsmaatregelen om ervoor te zorgen dat alleen de juiste gebruiker toegang heeft tot hun eigen persoonsgegevens.
- Updaten van de documentatie en de trainingsmaterialen voor systeembeheerders om deze nieuwe functionaliteiten te beschrijven en te demonstreren.

SR3	*Als systeembeheerder wil ik dat gebruikers toegang hebben tot hun filestorage in de backend zodat zij hun relevante documenten via de*	Should Have

	app kunnen inzien.	

Toelichting SR3
Een deel van de Backend is gereserveerd voor bestandsopslag. Deze zijn momenteel alleen toegankelijk voor het initiatief via het managementdashboard.

Het zou handig zijn als de gebruiker ook hun individuele documenten kan inzien via de app, maar dit vereist wel eerst een wijziging aan de backend.

Software Requirements SR3

- De VoMo app moet toegang hebben tot de bestandsopslag in de backend om relevante documenten van de gebruiker te laten inzien.
- Er moet een functie worden toegevoegd aan de app om deze documenten op te halen en te tonen aan de gebruiker.

SR4	*Als systeembeheerder wil ik dat de Backend van Voorstroom wordt uitgebreid zodat berichten naar het Eloket gestuurd kunnen worden vanuit de app.*	Could Have

Toelichting SR4
Eloket is energieloket. Hier kunnen gebruikers in contact komen met bijvoorbeeld energiecoaches. Het energieloket is momenteel nog niet operationeel. Waarschijnlijk wordt deze requirement daarom niet geimplementeerd.

SR5	*Als energiecollectief wil ik zelf mijn whitelabel kunnen kiezen, en op een relevante manier getoond wordt aan de app gebruikers.*	Could Have

Toelichting SR5
Een whitelabel is een product of dienst gemaakt door een bedrijf die andere bedrijven rebranden zodat het lijkt of het door hun zelf gemaakt is.

In dit geval zou het betekenen dat het Initiatief zelf de logo's en kleuren van de app kan kiezen. Dit moet op een intuitieve manier voor hun te doen zijn. Bijvoorbeeld met een webpagina waar ze de kleuren kunnen kiezen en logo's kunnen uploaden.

4.2 Deelvraag 2: Hoe ontwerp je een geschikte User Interface voor VoMo?

De opgestelde requirements moeten vertaald worden naar een interface die gebruiksvriendelijk is en er mooi uitziet op een groot aantal verschillende apparaten en schermgroottes.

Om hier zo goed mogelijk aan te voldoen is er een ‘Low-Fi’ en vervolgens een ‘High-Fi’ Ontwerp gemaakt. Hierbij worden ook de ontwerpkeuzes verder onderbouwd.

4.2.1.1 Low-Fi Design

‘Low-Fi’ is een afkorting van ‘Low Fidelity’. Dit houdt in dat alleen de basale vormgeving zichtbaar is. In softwareontwikkeling worden ook vaak ‘Wireframes’ gebruikt. Een Low-Fi ontwerp lijkt echter wel veel meer op de uiteindelijke applicatie.

Er worden bijvoorbeeld grijstinten gebruikt om de verschillende kleuren aan te geven en er wordt placeholder tekst gebruikt voor het tonen van gegevens. [2]

4.2.1.2 High-Fi Design

‘High-Fi’ is een afkorting van ‘High Fidelity’. Dit houdt in dat dit de complete vormgeving van de applicatie weergeeft. Dit is hoe de eindgebruiker de applicatie zou moeten zien.

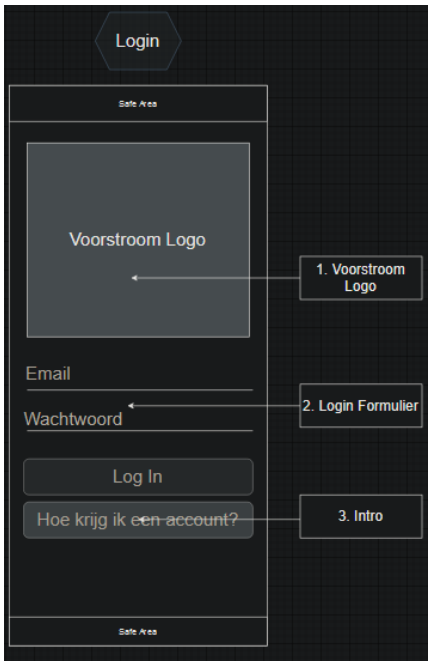
Het High-Fi Design wordt ook tijdens het ontwikkelen van de applicatie gebruikt. Er wordt geprobeerd dat het eindproduct zoveel mogelijk lijkt op het High-Fi ontwerp. [2]

4.2.2 Concept Low-Fi Ontwerp

4.2.2.1 Loginpagina

De loginpagina is de startpagina van de applicatie. De gebruiker moet namelijk eerst inloggen voordat allerlei gegevens opgehaald kunnen worden voor de rest van de app.

Figuur 1: Loginpagina



1. **Voorstroom Logo**

Het voorstroom logo staat groot en pontificaal op het loginscherm. Een kleiner en subtieler logo is geprobeerd, maar uiteindelijk is de voorkeur op het grote logo gevallen.

Dit is vooral gedaan om het scherm te vullen. Anders voelde de pagina erg leeg.

2. **Login Formulier**

Belangrijk functioneel component, de gebruiker moet ergens zijn inloggegevens kunnen invullen.

De gebruiker heeft als het goed is een emailadres bij het energiecollectief. Het energiecollectief verstuurd een uitnodiging waarmee de gebruiker dit emailadres aan de app kan koppelen en een wachtwoord selecteren.

Dit gebeurt buiten de app om, deze functionaliteit zit in het Managementdashboard.

In de app zijn er placeholders voor het Email en Wachtwoord geplaatst zodat het duidelijk is waar de gebruiker het moet invullen. De ‘Log In’ knop is in grayscale totdat de gebruiker iets heeft ingevuld bij het Email en Wachtwoord.

Na het inloggen wordt de gebruiker naar het Initiatief-Selectie, Project Selectie of het Hoofdscherm gestuurd.

3. **Intro**

Bij de eerste keer opstarten wordt de gebruiker een aantal slides getoond met instructies over het verkrijgen van een account. Registreren kan namelijk niet in de app.

De gebruiker kan opnieuw deze slides bekijken door op deze knop te drukken.

4.2.2.2 Initiatief-selectie

De gebruiker komt bij het ‘Intiatief-Selectie’ scherm als deze meer dan 1 initiatief aan zijn account heeft gekoppeld. De gebruiker kiest op dit scherm een initiatief uit om de details van te bekijken.

Veruit de meeste gebruikers hebben maar 1 initiatief aan hun account gekoppeld. De meeste gebruikers krijgen dit scherm dus niet te zien.

Figuur 2: Initiatief-Selectie

1. **Recyclerview met Initiatieven**

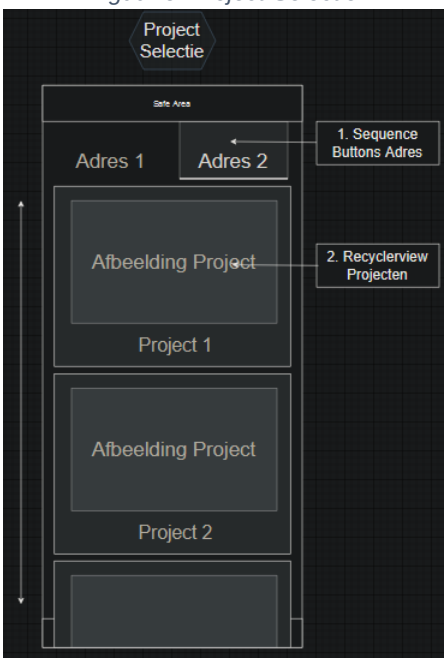
Er wordt een recyclerview getoond met de Initiatieven die de gebruiker aan zijn account heeft gekoppeld.

Bij ieder initiatief wordt alleen de naam getoond, het is nog niet mogelijk om meer gegevens op te halen.

4.2.2.3 Project-selectie

De gebruiker komt bij het ‘Project-Selectie’ scherm als deze meer dan 1 project aan zijn Initiatief heeft gekoppeld. De gebruiker kiest op dit scherm een Project uit om de details van te bekijken.

Figuur 3: Project-Selectie



Veruit de meeste gebruikers hebben maar 1 project aan hun account gekoppeld. De meeste gebruikers krijgen dit scherm dus niet te zien.

1. **Sequence Buttons Adres**

Bovenin de pagina zijn de adressen te zien de gebruiker heeft gekoppeld. Het is namelijk ook mogelijk om meer dan 1 adres per account te hebben. Bij het wisselen van adres update ook de Recyclerview met Projecten.

2. **Recyclerview Projecten**

Toont een scrollbare lijst met projecten die op het geselecteerde adres aanwezig zijn. Hierbij wordt

zowel de naam als een afbeelding van het project getoond.

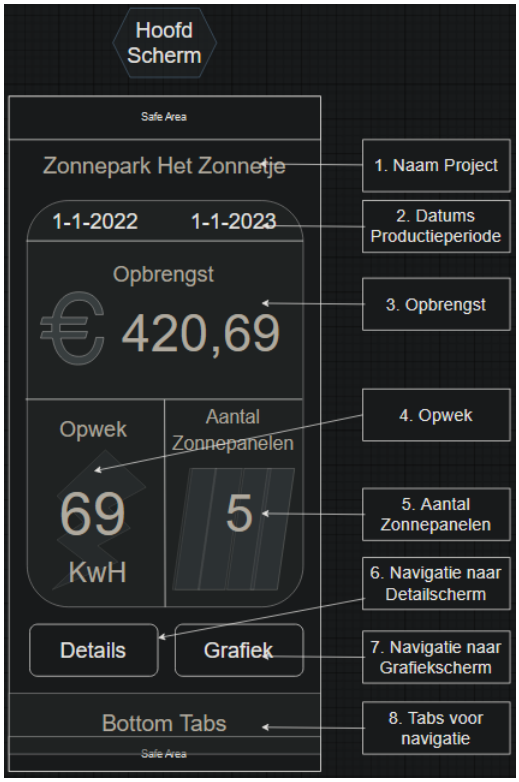
4.2.2.4 Hoofdscherm

De gebruiker beland op het hoofdscherm na het inloggen (en eventueel een initiatief en project selecteren). Het doel van deze pagina is de belangrijkste gegevens tonen aan de gebruiker.

1. **Naam Project**

Ieder project heeft een unieke naam. Vaak gerelateerd aan de locatie van het project. Bijvoorbeeld ‘Lochem Energie’ of ‘Opgewekt Houten’.

Figuur 4: Hoofdscherm



Omdat sommige gebruikers aan meer dan 1 project meedoen is het noodzakelijk om deze naam te tonen. Anders kunnen gebruikers moeilijk onderscheid tussen hun verschillende projecten zien.

2. **Datums Productieperiode**

De Opbrengst wordt over het algemeen uitgekeerd op basis van productieperiodes. Door groot de datum te tonen is het voor de gebruiker duidelijk naar welke productieperiode die aan het kijken is.

De hele rechthoek met productieperiode, opbrengst, opwek en aantal zonnepanelen is swipebaar naar links en rechts. Hiermee kan de gebruiker zowel naar de huidige productieperiode als die uit het verleden kijken.

3. **Opbrengst**

Waarschijnlijk de informatie waarin de gebruikers het meest geïnteresseerd zijn. De huidige opbrengst van een productieperiode. Daarom staat deze bovenaan en is deze het grootst.

4. **Opwek**

De totale productie van de zonnepanelen van de gebruiker tijdens deze productieperiode. Iets minder interessant, maar wel leuk om te tonen. Dit getal wordt afgerond op hele getallen.

5. **Aantal Zonnepanelen**

Het aantal zonnepanelen van de gebruiker tijdens deze productieperiode. Het is vaak mogelijk om panelen bij te kopen. Dus soms kan een gebruiker tijdens een bepaalde

productieperiode meer zonnepanelen hebben dan een andere. Dit is dus informatie die wel getoond moet worden ondanks dat er meestal geen verandering zal zijn.

6. **Navigatie naar Detailscherm**

Op het hoofdscherm is niet voldoende ruimte om alle gegevens op een intuïtieve manier te tonen. De extra gegevens worden weergegeven op het ‘Details Productieperiode’ scherm. Hier kan met deze knop naar genavigeerd worden.

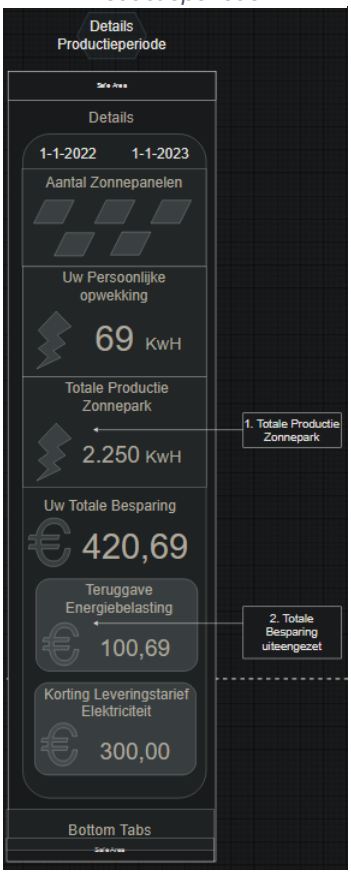
7. **Navigatie naar Grafiek**

Het is mogelijk om de opbrengst meer uiteengezet en per maand te tonen. Maar de enige manier om dit mooi te doen momenteel is met een grafiek. Deze grafiek wordt weergegeven op het ‘Grafiek Scherm’. De gebruiker kan hier naartoe navigeren met deze knop.

8. **Tabs voor navigatie**

Na het inloggen en het eventueel kiezen van een initiatief en project wordt de onderste tabbalk getoond. Hier staan een aantal icoontjes zodat de gebruiker tussen de verschillende schermen van de app kan wisselen.

Figuur 5: Details Productieperiode



4.2.2.5 Details Productieperiode

Dit scherm wordt getoond nadat de gebruiker op de details knop op het hoofdscherm drukt. Het doel van dit scherm is alle informatie gerelateerd aan de productieperiode tonen in een scrollbare lijst. Niet alle componenten op deze pagina worden hier beschreven. De meeste staan ook al in het hoofdscherm beschreven.

1. **Totale Productie Zonnepark**

Het is interessant voor de gebruiker om de totale productie van het zonnepark in te zien.

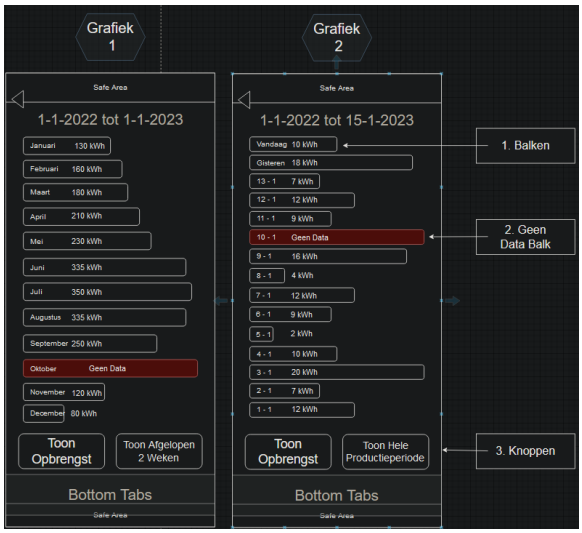
2. **Totale besparing uiteengezet**

De besparing voor de gebruiker bestaat vaak uit een aantal componenten. Meestal zijn het er minstens 2, maar het kunnen er veel meer zijn. Hier worden al deze kosten en opbrengsten getoond. Alles bij elkaar opgeteld wordt bovenaan weergegeven.

4.2.2.6 Grafiek

Dit scherm toont de grafiek voor de afgelopen 7/14 dagen en de hele productieperiode. De grafiek is getest in meerdere oriëntaties, deze oriëntatie met de horizontale balken is de meest logische. Bij het verticaal oriënteren ontstaat er veel lege ruimte, zeker bij telefoons met een groter scherm.

Figuur 5: Grafieken



4.2.2.7 Instellingen

Het instellingenscherm bevat momenteel maar 2 componenten. Deze zijn echter wel erg belangrijk. Ook is hier ruimte voor uitbreiding, alle toekomstige technische instellingen worden in dit scherm geplaatst.

Figuur 6: Instellingen

1. **Wissel van Initiatief**

Deze knop wordt alleen getoond aan gebruikers die meer dan 1 initiatief aan hun account gekoppeld hebben. Hiermee navigeert de gebruiker naar het ‘Initiatief-Selectie’ scherm om een ander initiatief te kunnen kiezen.

2. **Log uit Knop**

Hiermee kan de gebruiker uitloggen uit de app. Ze worden vervolgens weer de loginpagina getoond.

4.2.3 High-Fi Design

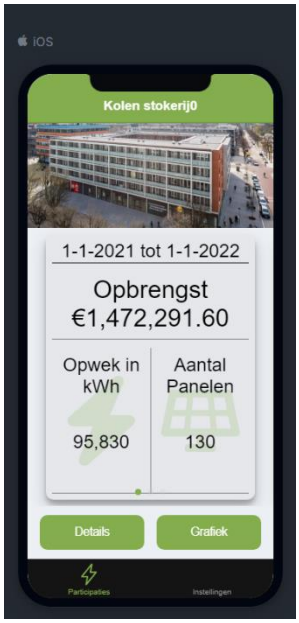
Er was het plan om een High-Fi prototype te maken, maar uiteindelijk is er besloten om dit niet te doen. Binnenkort krijgt Voorstroom toegang tot een UX/UI specialist die komt werken bij Inversable.

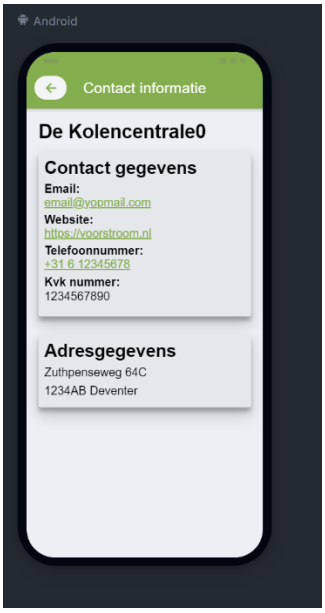
Er is daarom geen behoefte aan een losstaand High-Fi Design. Waarschijnlijk zou deze in de toekomst namelijk overbodig worden.

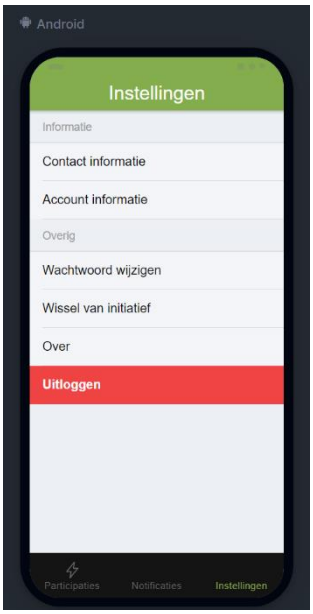
Desondanks is er tijdens de ontwikkeling van de app wel geprobeerd een functionele UI te maken die in de tussentijd gebruikt kan worden.

Hier zijn 4 screenshots te zien:

- Hoofdscherm
- Contactinformatie Voorstroom
- Contactinformatie Initiatief
- Instellingenscherm







4.3 Deelvraag 3: Welk Hybride Framework is het meest geschikt voor het ontwikkelen van VoMo?

Het doel van de derde deelvraag is het ontwerpen van een professionele en gebruiksvriendelijke User Interface voor VoMo.

Dit wordt op de volgende manier aangepakt:
Er worden concepten voor de vormgeving gemaakt. Deze worden onderbouwd in het functioneel ontwerp. Vervolgens wordt er intern feedback verzameld op deze concepten en wordt op basis hiervan de uiteindelijke vormgeving bepaald.

4.3.1 Stappen Onderzoek

Dit hoofdstuk bevat het onderzoek naar een geschikt Hybrid Framework voor VoMo.

Dit onderzoek wordt gedaan in de volgende stappen:

1. **Opstellen Longlist**

Er wordt eerst een lijst met mogelijke Hybrid Frameworks en de bijbehorende programmeertalen die gebruikt zouden kunnen worden voor VoMo.

Hierbij worden kort de belangrijkste voor-en-nadelen van ieder framework benoemd. [1] [2] [3]

2. **Opstellen Shortlist**

Uit deze longlist worden de meest kansrijke kandidaten geselecteerd. Er wordt gemotiveerd waarom deze kandidaten het best geschikt zijn.

3. **Vergelijken Shortlist**

Vervolgens worden de kandidaten van de Shortlist op een aantal punten met elkaar vergeleken. Deze punten zijn specifiek gericht op de eisen die Voorstroom heeft.

4. **Bepalen uiteindelijke keuze**

Vervolgens wordt een uiteindelijke keuze voor een framework gemaakt.

Er wordt voor ieder framework gereflecteerd op het onderzoek. Op basis hiervan worden aanbevelingen gedaan.

4.3.2 Opstellen Longlist

Ionic (met Angular of Vue)

Ionic is waarschijnlijk de meest bekende manier om Hybride applicaties te ontwikkelen. Het is ook een van de oudste beschikbare opties.

Voordelen	Nadelen
Relatief Lage Leercurve.	'Hot Reloading' is niet mogelijk.
Veel beschikbare plug-ins.	Soms incompatibele plug-ins
Live reload systeem	Zeer afhankelijk van plug-ins
Veel kant-en-klaar beschikbare UI componenten	Redelijke maar niet uitzonderlijke performance.
Ingebouwde Development Server	
Debugging tools beschikbaar	
IONIC CLI, erg krachtige command line interface beschikbaar.	
Relatief Populair	
Maakt gebruik van Angular	

React Native

Het open-source UI framework ontwikkeld door facebook.

Voordelen	Nadelen
Goede Performance	Minder goed geschikt voor applicaties met veel UI transities, interacties en verschillende soorten schermen.
Mogelijkheid om bestaande React Code te hergebruiken	Beperkte selectie aan Custom Modules
Relatief Populair	Relatief weinig kant-en-klare UI en Native functionaliteit
Uitgebreide Live Reload functionaliteit	Pittige leercurve
Ondersteund door node.js	Weinig ingebouwde navigatiecomponenten
Veel 'Social Plugins' beschikbaar.	Relatief lange ontwikkeltijden

Mobile Angular UI

Een combinatie van de Twitter Bootstrap en AngularJS waarmee HTML5 hybride applicaties gemaakt kunnen worden.

Voordelen	Nadelen
Combinatie van Angular en Bootstrap	JavaScript-only, geen Typescript
Grote beschikbaarheid UI Componenten	Onoverzichtelijk debuggen door combinatie AngularJS en HTML.
Open Source	Kleine ontwikkelaarscommunity
Maakt gebruik van Angular	Lage populariteit
	Wordt onderhouden door 1 persoon.

NativeScript (met Angular of Vue)

Nativescript is net als Ionic een framework die al lang bestaat voor hybrid development. Het is geschikt voor Android en iOS.

Voordelen	Nadelen
Uitstekende Native functionaliteit	Grote bestandsgrootte, zelfs onder Hybrid Frameworks.
Veel beschikbare 3rd Party Libraries	De kwaliteit van de open source plugins is inconsistent.
	Zeer lage populariteit
	Geen ondersteuning voor Webapplicaties.

Xamarin

Xamarin heeft een met C# gedeelde codebase. Het kan gebruikt worden om native Android, iOS en Windowsapplicaties met een native user interface te maken.

Xamarin is sinds 2016 een deel van Microsoft. De ondersteuning vervalt echter in November 2021. De vervanger hiervoor is .NET MAUI.

Voordelen	Nadelen
Ondersteund door Microsoft	Kleine Community
Ruime compatibiliteit: Android, Ios, tvOS, WatchOS, MacOS, en Windows	Jong platform (met daardoor meer bugs)
	Ondersteuning vervalt in November 2021
	API Lag
	Geen ondersteuning Web

Kendo UI

Kendo UI is een HTML5 UI framework voor het maken van websites en applicaties. Het is voornamelijk populair onder enterprises vanwege de prijs.

Er zijn kosten vooraf om überhaupt het framework te mogen gebruiken.

Voordelen	Nadelen
Veel klant en klare widgets voor jQuery, Angular, React en Vue	Betaald Framework
Widgets zijn eenvoudig aan te passen met CSS	Gericht op Enterprises.
Sterke Documentatie	
Uitgebreide Ondersteuning vanuit de ontwikkelaar	

Framework7

Framework7 is een gratis en open source mobile html framework. Het is alleen compatibel met Andoid en IOS.

Voordelen	Nadelen
Flexibel, er zijn een aantal varianten beschikbaar.	Meer gericht op het Apple ecosysteem en daardoor passen sommige thema's niet goed op Android.
Gratis en Open Source	Alleen ondersteuning Android en IOS.
	Beperkte Community
	Veel informatie alleen beschikbaar op Fora.

Apache Cordova (formerly PhoneGap)

Apache Cordova is geschikt voor het maken van hybrid web applications met behulp van HTML5 en JavaScript. Het is ontwikkeld Apache (valt onder Adobe).

Voordelen	Nadelen
Ondersteund door Apache	Matige Performance
Zeer ervaren Hybrid Development Platform (sinds 2011)	Beperkt aantal UI Widgets beschikbaar
Open Source	Beperkte Native functionaliteit
Relatief lage leercurve	

Flutter

Flutter is een open-source UI software development kit gemaakt door Google. De ontwikkelde applicaties zijn compatibel met Android, iOS, Web, Linux, Mac, Windows en zelfs Google Fuchsia.

Voordelen	Nadelen
Ondersteund en ontworpen door Google	Relatief Jong (December 2018)
Uitstekende performance (voor hybrid)	Gebaseerd op Dart. Dit is een nieuwe programmeertaal specifiek voor Angular.
Veel kant en klare widgets	Featureset is (nog) beperkt
Live en Hot Reload beschikbaar	Relatief grote applicatiegrootte
Zeer hoge populariteit	

4.3.3 Opstellen Shortlist

Van deze lijst met kandidaten zijn de vier kansrijkste geselecteerd.

De kandidaten zijn:

1. Flutter (met Dart)
2. React Native (JSX/Javascript + HTML)
3. Ionic (met Angular)
4. Nativescript (met Angular)

4.3.3.1 Motivatie Kandidaten Shortlist

Er is gekozen voor deze 4 kandidaten op basis van de volgende punten:

- **Platformondersteuning Android, iOS en Web:**

Alle kandidaten hebben op zijn minst ondersteuning voor Android, iOS en Web. Dit is een vereiste voor VoMo.

Extra platformondersteuning is een voordeel. Maar niet erg relevant voor VoMo.

- **Bestaande kennis binnen Inversable:**

Het is voor een kandidaat een voordeel als er al kennis en ervaring hiervan bij Inversable of Voorstroom aanwezig is. Het is echter geen vereiste.

- **Populariteit**

Het is een voordeel als een kandidaat een hoge populariteit heeft (gehad). Hierdoor is er meer informatie over online te vinden. Er is hierdoor ook een hogere kans dat personeel er al ervaring mee heeft.

- **Mogelijkheid tot hergebruiken bestaande code**

Bij Nativescript en Ionic is het misschien mogelijk om de bestaande Angular Web apps uit te breiden met een Native mobile versie. Als dit werkt scheelt dit ontwikkeltijd en een codebase.

4.3.4 Vergelijken Shortlist

Voor het vergelijken van Flutter, React-native, Ionic en Nativescript (Angular) worden op basis van de volgende punten met elkaar vergeleken;

- Ondersteunde platformen
- Populariteit
- Leercurve
- Bestaande kennis Inversable
- Mogelijkheid om bestaande Angular Web applications te migreren naar Native Mobile.

4.3.4.1 Ondersteunde Platformen

Score	Naam Kandidaat
1	Flutter + React Native
2	Ionic + Nativescript

Alle kandidaten voldoen aan de eis om ondersteuning te hebben voor:

- Android
- Ios
- Web

Flutter en React Native gaan nog een stapje verder en bieden ook ondersteuning voor Windows en MacOS.

4.3.4.2 Populariteit

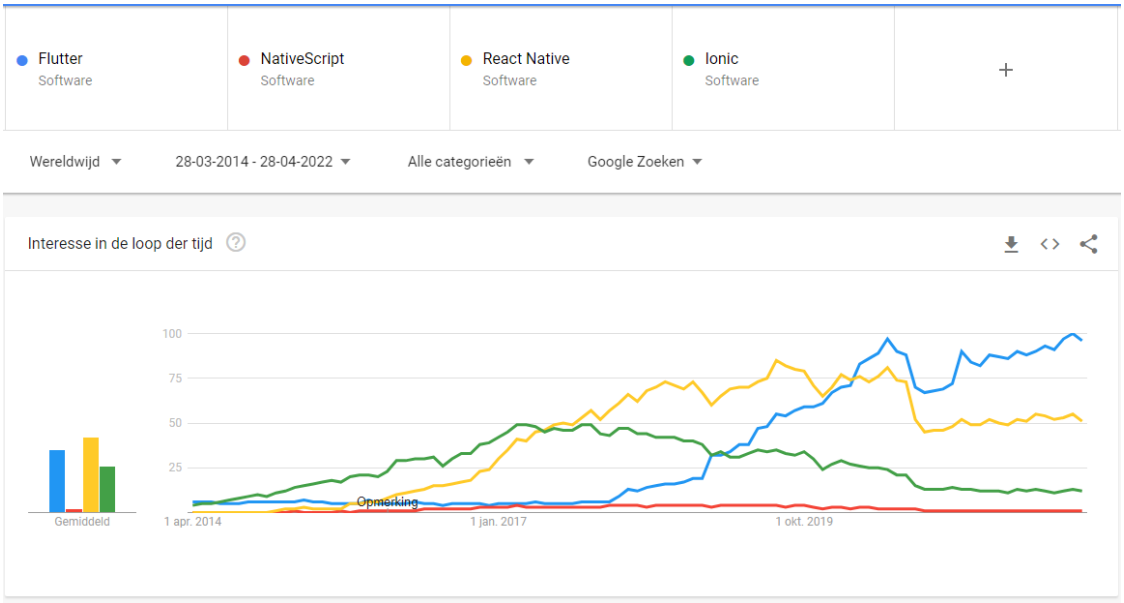
Score	Naam Kandidaat
1	Flutter
2	React Native
3	Ionic
4	Nativescript

Momenteel is Flutter het populairste Hybrid development platform.

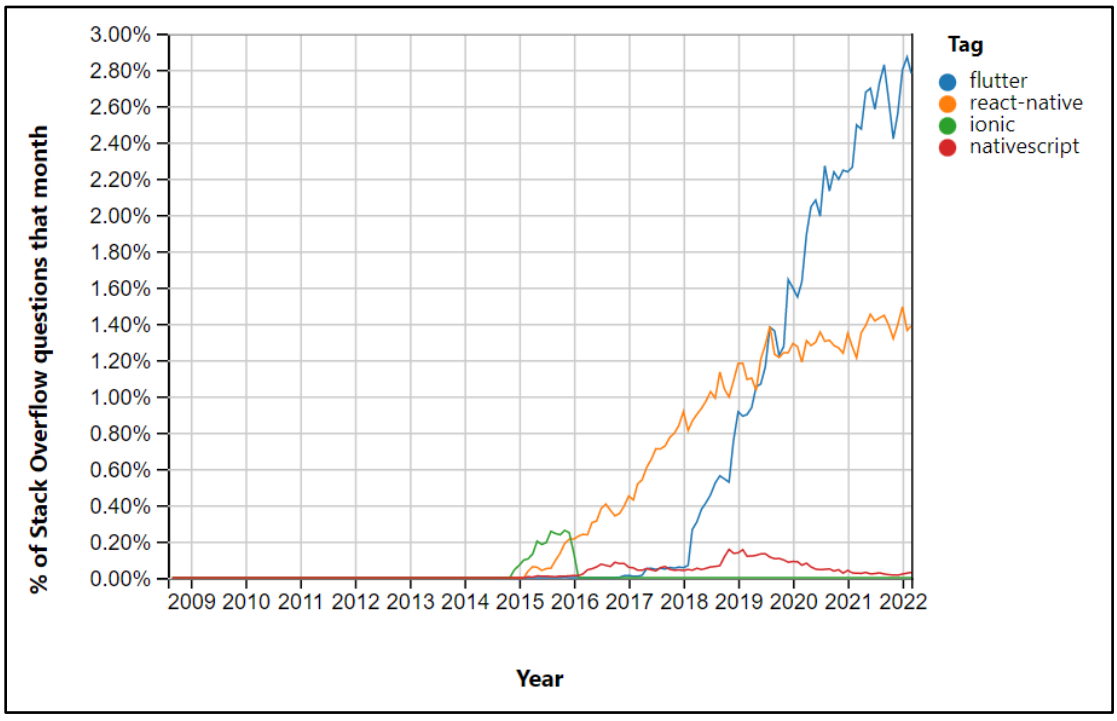
In het verleden was React Native het meest populaire platform, en in een verder verleden was Ionic het meest populaire platform.

Nativescript heeft nooit een hoge populariteit gehad.

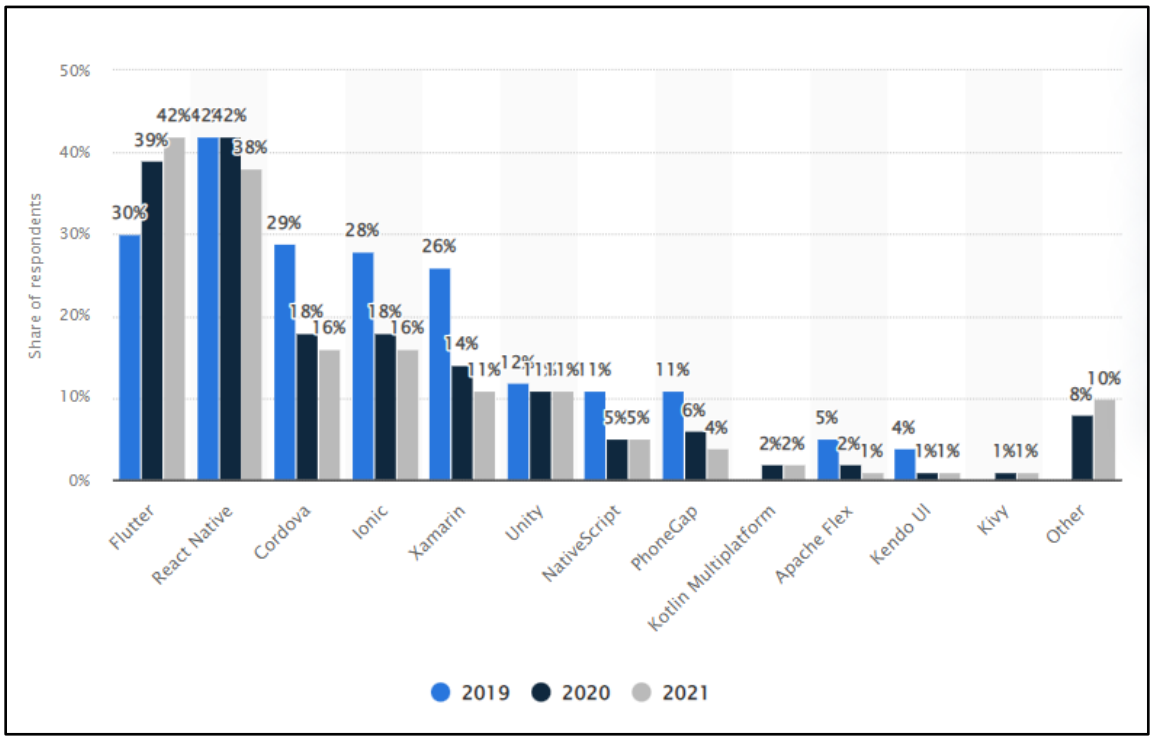
Figuur 1: Google Trends Hybrid Frameworks [4]



Figuur 2: Stack Overflow Activiteit [5]



Figuur 3: Cross-Platform Mobile Frameworks – Software Developer Survey Statista [6]



4.3.4.3 Bestaande Kennis Inversable

Score	Naam Kandidaat
1	Ionic + Nativescript
2	React Native
3	Flutter

Binnen Inversable zijn er al een aantal Angular Web apps gemaakt. Dus er is zeker al kennis in het gebruik van Angular.

Er is minder kennis van React Native, de huidige versie van de Voorstroom Mobile Applicatie is hiermee gemaakt.

Er is nog niet eerder met Flutter gewerkt binnen Inversable.

4.3.4.4 Angular Web Application Migreren naar Native

Score	Naam Kandidaat
1	Ionic + Nativescript
2	React Native + Flutter

Bij Ionic en Nativescript is het mogelijk om Angular Web applications te migreren naar een native Android/Ios app. Dit kan echter wel behoorlijk wat tijd kosten.

4.3.5 Conclusie Hybrid Frameworks

Dit hoofdstuk bevat voor ieder framework de belangrijkste conclusies van het onderzoek. Hier wordt ook een aanbeveling gedaan voor de uiteindelijke keuze van het framework.

Ionic (Angular)

Ionic was in het verleden een super populaire manier om hybride applicaties te ontwikkelen. Het is ook een van de oudste beschikbare opties.

De populariteit is de laatste jaren wel wat aan het teruglopen, waarschijnlijk door grote nieuwe concurrenten zoals React Native en Flutter. Ook zitten een aantal plugins vast achter een enterprise licentie. Er is echter ruim voldoende functionaliteit beschikbaar in de open source (community) versie.

Ionic met Angular heeft een aantal voordelen specifiek voor Inversable. Er is al veel kennis van Angular aanwezig. Ook is het misschien mogelijk om de nieuwe VoMo app toe te voegen aan de bestaande codebase voor de Angular Web app van Voorstroom. Maar hier is niet zeer veel documentatie over te vinden.

Aanbeveling:
De mogelijkheid om de bestaande Angular Web apps te hergebruiken is een gigantisch voordeel voor Ionic, en dit is daarom ook de eerste keus voor de VoMo app.

Er zitten wat nadeeltjes aan het platform, met name de teruglopende populariteit en sommige gepaywallde plugins. Dit zijn echter geen onoverkoombare problemen. De mogelijkheid voor het hergebruiken van bestaande code is zonde om links te laten liggen.

Flutter

Flutter is momenteel het meest populaire Hybrid Development platform. Dat is niet zonder reden; het heeft een relatief eenvoudige leercurve, veel QOL-features en voor Hybrid zeer goede performance. Verder is het platform Open Source en ondersteund door Google.
Het heeft echter geen specifieke voordelen voor Voorstroom en Inversable.

Aanbeveling:

Flutter lijkt een prima kandidaat voor het maken van de VoMo app. Door de hoge populariteit is er heel veel online te vinden. Ook is er veel kans dat toekomstige ontwikkelaars ervaring met Flutter hebben.

Helaas zijn er voor Voorstroom wel wat nadelen waardoor dit framework niet de eerste, maar de tweede keus is.

Het framework kan alleen gebruikt worden met Dart, een programmeertaal specifiek voor Flutter. Deze is niet moeilijk te leren, maar momenteel heeft bijna niemand binnen Voorstroom hier ervaring mee.

Ook is het niet mogelijk om de Native versie toe te voegen aan de bestaande Codebase van Voorstroom. Dit betekent dat er een extra Codebase onderhouden moet worden.

React-Native

React was en is een zeer populair Hybrid development platform. Het wordt ondersteund door facebook. Het heeft uitgebreide functionaliteit en relatief goede performance.
Het framework heeft wel een groot nadeel. Het staat bekend om de langere ontwikkeltijden. Hierdoor wordt het ook afgeraden voor startups en junior ontwikkelaars.

Aanbeveling:

React Native wordt niet aangeraden om de VoMo app te maken. Het zou prima kunnen, maar er zijn betere opties.

Het enige specifieke voordeel voor Voorstroom is dat de huidige Voorstroom Mobile app al in React Native gemaakt is. Er is ook een beetje kennis hiervan aanwezig binnen Voorstroom.

Het nadeel is de langere ontwikkeltijd en dat het niet mogelijk is om de native versie toe te voegen aan de bestaande Angular Web app voor Voorstroom. Dit betekent dat er een extra Codebase onderhouden moet worden.

Nativescript

Nativescript lijkt op papier een hele geschikte optie, het heeft nagenoeg dezelfde functionaliteit als de veel populairdere frameworks. Toch is deze minder geschikt. Het kan gebruikt worden met een aantal programmeertalen waaronder Angular.

Ook is het mogelijk om de Native versie te combineren met de bestaande Angular Web apps.

Aanbeveling:

Tijdens het onderzoek is naar voren gekomen dat het Nativescript framework uitgebreide functionaliteit heeft, maar het heeft wel een kleine gebruikersbasis en naast de officiële documentatie is er online weinig over het framework te vinden.

Dit is een behoorlijk risico. Er is een kans dat er een probleem wordt aangetroffen en er absoluut niks online erover te vinden is. Tegenover dit risico heeft het framework heeft verder niet echt voordelen ten opzichte van Ionic (en de andere frameworks) en lijkt daarom minder goed geschikt voor VoMo.

4.4 Deelvraag 4: Wat is de beste manier om VoMo technisch vorm te geven en te laten integreren met alle huidige systemen die gebruikt worden voor Voorstroom?

Het doel van de vierde deelvraag is bedenken hoe de Voorstroom Mobile Applicatie er technisch uit gaat zien en hoe deze integreert met alle huidige systemen die gebruikt worden voor Voorstroom.

Hierbij wordt niet alleen gekeken naar de nieuwe applicatie, maar ook naar alle wijzigingen die in de backend structuur gemaakt moeten worden om de nieuwe app te faciliteren.

Dit wordt op de volgende manier aangepakt:

Er wordt in kaart gebracht welke systemen er nu bij Voorstroom in gebruik zijn en welke gegevens de Voorstroom Mobile app hieruit moet halen. Vervolgens wordt een technisch ontwerp gemaakt waarmee de wijzigingen aan de backend gemaakt kunnen worden. Vervolgens kan hier een prototype van de Voorstroom Mobile app ontwikkeld worden.

4.4.1 Waarborgen codekwaliteit

Tijdens de ontwikkeling van de applicatie werd het duidelijk dat de code waarschijnlijk van onvoldoende kwaliteit zou zijn. Voorstroom wil graag de applicatie verder blijven ontwikkelen, goede codekwaliteit heeft dus een hoge prioriteit.

Om dit te waarborgen wordt er gebruik gemaakt van ‘Merge Requests’ waarbij andere ontwikkelaars (Tygo + Mark) commentaar kunnen geven op de code. Een ‘Merge Request’ is simpelweg een verzoek om de nieuwe code toe te voegen aan de bestaande code, op deze manier wordt dus alle nieuwe code altijd door 2 mensen bekeken.

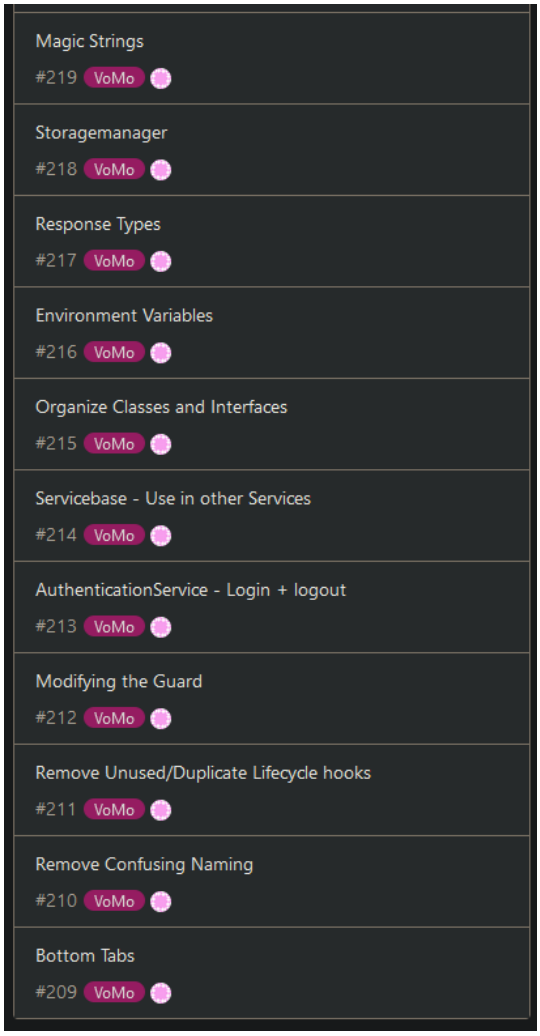
Tijdens het begin van de afstudeeropdracht is er niet met dit systeem van mergerequests gewerkt. Helaas was de basis van de applicatie hierdoor niet van voldoende kwaliteit. Er is een volledige code review van de hele applicatie gedaan en in grote lijnen waren er de volgende problemen ontdekt:

- Bestandsstructuur
- Naamgeving
- Complexiteit loginsysteem
- Algemene leesbaarheid code

Er is hier vervolgens een lijst met Gitlab Issues van gemaakt. Uiteindelijk zijn al deze problemen opgelost.

Het was achteraf gezien veel beter als deze problemen aan het begin van de stageopdracht waren opgelost. Doordat de basis van de app eigenlijk niet goed was duurde het ontwikkelen van nieuwe functionaliteit veel langer.

Gelukkig is er uiteindelijk wel aandacht voor gekomen en is de kwaliteit van de code een van de sterkere punten van de applicatie!



4.4.1.1 Kwaliteit Ontwerp

Het nieuwe ontwerp van de code is zo opgezet dat er rekening wordt gehouden met veiligheid, schaalbaarheid en uitbreidbaarheid. In dit hoofdstuk wordt de schaalbaarheid en uitbreidbaarheid van VoMo aangetoond. Veiligheid is bij deze app van groot belang, daarom is daarvoor een apart ‘Security’ hoofdstuk gemaakt.

Uitbreidbaarheid

Onder uitbreidbaarheid wordt verstaan hoe eenvoudig het is om de app in de toekomst verder uit te breiden. Om de toekomstige uitbreidingen zo eenvoudig mogelijk te maken moet er van Voorstroom volgens de volgende principes gebruik gemaakt worden.

Herbruikbare Componenten

Ionic maakt gebruik van Angular. Een op componenten gebaseerd web application framework. [2]

Een van de mogelijkheden van Angular is het hergebruiken van gemaakte componenten. Een component is een stukje van de content die op een pagina getoond wordt. Een hele pagina is zelfs ook een component.

Als er een vermoeden is dat een bepaald onderdeel op meerdere plekken aan de gebruiker getoond moet worden, is het verstandig om hier een component van te maken.

Dit kan in sommige gevallen ook gedaan worden om de leesbaarheid van de code te verbeteren. Kleinere en simpelere componenten die met elkaar verbonden zijn is over het algemeen beter leesbaar dan enkele hele grote componenten.

Gebruik Pagina’s

Pagina’s zijn in principe ook componenten. In sommige Angular en Ionic applicaties worden deze ook niet anders behandeld.

Bij VoMo is er besloten om dit wel te doen. De motivatie hiervoor is de betere overzichtelijkheid in de structuur van de app. Het nadeel hiervan is echter wel meer componenten, modules en dus code in de app aanwezig is.

Servicebase

Bij VoMo is er een onderdeel van een andere Voorstroom Angular applicatie hergebruikt. Het Managementdashboard is een webapplicatie waarmee een collectief al zijn participanten beheerd.

Aangezien dit Managementdashboard met dezelfde Back End communiceert als VoMo is hier een groot onderdeel die hergebruikt kon worden.

De servicebase is een stuk code die als uitbreiding voor een Angular Service dient. De Servicebase kan alle handelingen die noodzakelijk zijn voor het ontvangen en versturen van API-Calls.

Hierdoor hoeft deze code niet in de individuele Services gedefinieerd te worden. Met als resultaat beter leesbare en makkelijker uit te breiden Services.

Schaalbaarheid

Onder Schaalbaarheid wordt verstaan hoe geschikt de app is om op grote schaal uit te brengen. Als er met bepaalde principes geen rekening wordt gehouden ontstaan er problemen als de app op grote schaal wordt uitgerold. Dit hoofdstuk behandeld een aantal van deze principes.

Efficiëntie API-Calls

De app gebruikt API-Calls voor het ophalen van alle data die aan de gebruiker getoond wordt. Er moet hier echter wel een afweging gevonden wanneer welke data wordt opgehaald.

Het is niet nodig om altijd alle gegevens op te halen. Dit is een onnodige belasting op de Backend, maakt de app zwaarder en zorgt ervoor dat de app meer data gebruikt.

Bij iedere API-Call wordt de afweging gemaakt wanneer en hoe deze wordt aangeroepen. Bijvoorbeeld pas wanneer de gebruiker naar een specifieke pagina navigeert.

Bij sommige Calls maakt het erg weinig uit. Het ophalen van productieperiodes is bijvoorbeeld erg licht voor de Backend, dus deze worden allemaal opgehaald zodra de gebruiker naar het hoofdscherm navigeert.

Maar de opbrengsten die bij deze productieperiode horen zijn daarentegen weer erg zwaar voor de backend om te produceren. Deze worden dus alleen opgehaald voor de specifieke productieperiode waar de gebruiker naar kijkt.
Pas als een gebruiker begint te swipen naar een nieuwe productieperiode wordt een nieuwe Call gedaan om de opbrengsten van die periode te berekenen.

Ondersteuning verschillende Devices

VoMo is gemaakt met Ionic en ondersteund hierdoor zowel Android, Ios als Web. Helaas kunnen niet alle Android en iOS versies worden ondersteund. Het zou de functionaliteit

van de app erg limiteren als deze ook op de originele Iphone uit 2007 zou moeten kunnen werken.

Het is wel wenselijk om zoveel mogelijke versies van Android en iOS te ondersteunen. Hiermee wordt voorkomen dat klanten de app niet kunnen installeren omdat hun smartphone te oud is.

Voor IOS is de ondersteuning redelijk. Er is ondersteuning terug t/m IOS 13, de oudste ondersteunde Iphone daarmee is de 6s. Hiermee is er meer dan 90% van het IOS-marktaandeel in Nederland gedekt. [3]

Voor Android is de ondersteuning zeer goed. Er is ondersteuning terug t/m Android 5.0, hierdoor is er ondersteuning voor meer dan 99% van het Android marktaandeel in Nederland. [4]

Compactheid Assets

Een groot deel van de uiteindelijke bestandsgrootte van de app is afhankelijk van de hoeveelheid Assets die deze bevat. Daarnaast zijn hybrid applicaties altijd iets groter dan hun native tegenhangers.

Om te voorkomen dat de app onnodig groot wordt moet er rekening worden gehouden met de soort assets die gebruikt worden, de hoeveelheid hiervan en de grootte.

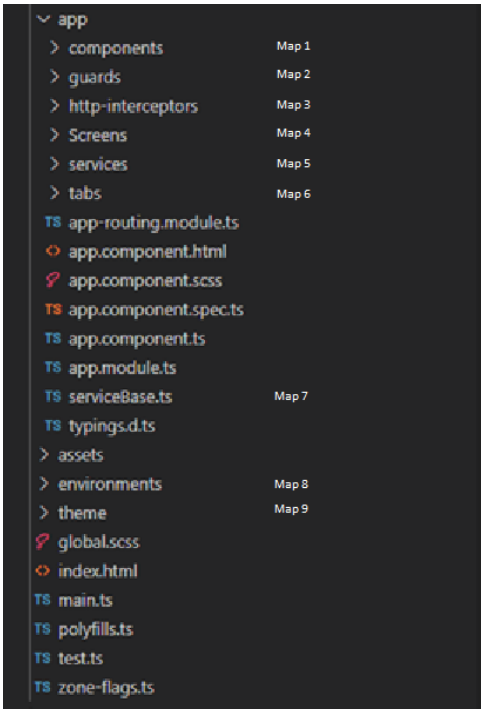
Assets zijn voornamelijk Afbeeldingen, hoewel het er misschien mooi uit ziet is het niet nodig om allemaal 4k afbeeldingen voor bijvoorbeeld logo’s te gebruiken. Vaak is het ook makkelijker om een afbeelding pas op te halen zodra een gebruiker naar een pagina navigeert. Die gebeurt bijvoorbeeld bij de projectfoto die bovenin het hoofdscherm getoond wordt, die wordt gewoon met een API-Call opgehaald en hoeft daardoor niet in de assets bewaard te worden.

4.4.2 Bestandsstructuur Code

Dit hoofdstuk beschrijft de bestandsstructuur van de app. Het is een belangrijke factor in de leesbaarheid en toekomstige uitbreidbaarheid van de app.

De relevante mappen en bestanden zijn gemarkeerd in de afbeelding. Deze gemarkeerde mappen en bestanden worden hier toegelicht. Er wordt eventueel verwezen naar andere hoofdstukken die nog meer toelichting kunnen geven.

Figuur 4: Complete Bestandsstructuur



Map 1: Components

Bevat alle (herbruikbare) Ionic componenten. Alle componenten die hierin staan worden op meerdere pagina's in de app gebruikt.

Bekijk het hoofdstuk: 'Kwaliteit Ontwerp – Herbruikbare Componenten', voor meer uitleg over Componenten.

Map 2: Guards

Deze map bevat de code gerelateerd aan de Guards in de app. Guards belemmeren gebruikers met het bezoeken van bepaalde webpagina's voordat er aan een conditie is voldaan. Momenteel zijn er maar twee van deze condities:

1. Het dwingen dat de gebruiker inlogt. Anders werkt een groot deel van de functionaliteit niet.
2. Het tonen van een aantal introductieslides de eerste keer dat de gebruiker de app opent.

Voor meer informatie over Guards, bekijk het hoofdstuk: 'Security - Guards'.

Map 3: Http-interceptors

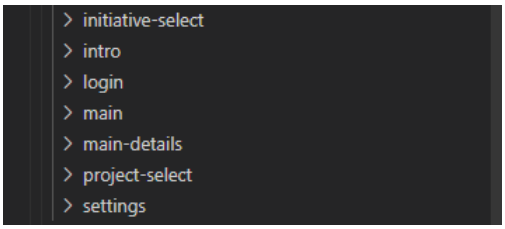
Deze map bevat de code gerelateerd aan het meesturen van de 'headers' bij de verschillende API-Calls. Deze headers zijn noodzakelijk voor de authenticatie van deze API-Calls.

Een voorbeeld van een API-Call met de toegevoegde header is te vinden in het hoofdstuk: 'Voorstroom API – Voorbeeld Api Call'.

Map 4: Screens

Deze map bevat een heel aantal andere mappen. Iedere mapje bevat een scherm die de app toont aan de gebruiker.

Figuur 5: Bestandsstructuur Schermen



Schermen zijn in Angular ook Componenten. In sommige Apps en Websites worden de pagina's ook niet los gedefinieerd, maar staan deze ook onder Componenten.

Bij deze app is er besloten om de schermen wel los te definiëren. Een motivatie hiervoor is te vinden in: [‘Kwaliteit Ontwerp – Gebruik pagina’s’.]

Map 5: Services

Angular Services zijn objecten die maar een keer geïnstantieerd tijdens het gebruik van de applicatie. Vaak worden ze gestart bij het opstarten.

Figuur 6: Bestandsstructuur Services

Services bevatten methoden met data die onderhouden wordt gedurende het gebruik van de applicatie. Deze data is continue beschikbaar en kan met behulp van componenten aan de gebruiker getoond worden. [1]

Map 6: Tabs

Er is gekozen voor Tabs als methode van navigatie in VoMo. Deze worden onderin het scherm getoond. Deze map bevat alle logica die hiervoor noodzakelijk is.

Bijvoorbeeld dat de tabs getoond worden op alle pagina’s en dat de huidige pagina een andere kleur heeft in het tab-balkje om deze te onderscheiden.

Map 7: Servicebase.ts

Wat gerecyclede code-onderdelen uit het ‘ManagementDashboard’. De Servicebase zorgt voor alle additionele logica die nodig is om API-Calls te kunnen doen naar de backend.

De eerdergenoemde Services importeren deze Servicebase.

Er is meer uitleg te vinden over de Servicebase in het hoofdstuk: [Waarborgen Codekwaliteit – Kwaliteit Ontwerp – Servicebase].

Map 8: Environments

Bevat de omgevingsvariabelen voor de applicatie. Hierin kunnen een aantal dingen geconfigureerd worden. Bijvoorbeeld de URL waar de Backend gehost staat.

Er is zowel een versie van de environmentvariabelen voor de productieomgeving en de ontwikkelomgeving, deze zijn namelijk verschillend.

Map 9: Theme

Bevat de Ionic thema bestanden. Dit zijn scss-bestanden die gebruikt worden voor de verschillende thema's in de app. Dit is een standaard functie van Ionic maar dient uiteraard wel handmatig geconfigureerd te worden.

Er is hierbij gebruik gemaakt van de Voorstroom kleuren die ook al in het Managementdashboard gebruikt worden.

Het is eventueel mogelijk om de app verschillende thema's te laten hebben afhankelijk van het systeemthema (donker of licht), en hierbij ook onderscheid te maken tussen de Android, iOS en Webversie.

4.4.3 Voorstroom API

API staat voor 'Application Programming Interface'. API's laten een product of dienst communiceren met andere producten of diensten, zonder precies te weten hoe deze zijn geïmplementeerd.

API's kunnen gezien worden als een soort van contract. Als de ene partij een verzoek op een bepaalde manier stuurt, dan is dit hoe de andere partij reageert. [6]

Meestal wordt er bij een API gebruik gemaakt van endpoints. Dit zijn URL's waarmee de verschillende contracten onderscheiden worden.

Bijvoorbeeld:

- **{Locatie Backend}/signin** -> Wordt gebruikt voor het inloggen.
- **{Locatie Backend}/logout** -> Wordt gebruikt voor het uitloggen.

Of meer complex:

- **{Locatie Backend}/api/my/users/projects**
  -> Wordt gebruikt om de projecten van een gebruiker op te halen

Figuur 7: Voorbeeld API Respons


```
{
   "Id": "3df68dd6-15e2-400e-ae61-f6d5053ca406",
   "FirstName": "Emiel",
   "LastName": "Haarhuis",
   "City": "Deventer",
   "Street": "Zutphenseweg",
   "HouseNumber": "3",
   "Postcode": "7418AJ",
   "Projects": [
        {
            "Id": "841747f1-8ac0-4353-96b6-921214b1a386",
            "Name": "Zonnepark het Zonnebloempje",
            "FeaturedPictureUrl": "https://kempenenergie.nl/wp-30.jpg"
        }
    ]
}
```


4.4.3.1 Backend VoMo

Er is al een uitgebreide API voor Voorstroom in gebruik. Deze Backend is verantwoordelijk voor het managementdashboard en de oude VoMo app.

Er had gekozen kunnen worden om de API-Endpoints van de oude VoMo app te hergebruiken. Uiteindelijk is er gekozen om alleen de endpoints voor het inloggen en uitloggen te hergebruiken, deze zijn namelijk erg gecompliceerd om te vervangen. Wijzigingen hieraan zou betekenen dat de inlogprocedure voor bestaande software van Voorstroom ook mee zou moeten veranderen.

Alle andere endpoints zijn opnieuw gemaakt. Hiervoor zijn de volgende redenen:

- Kwaliteit en structuur
- Documentatie
- Leeftijd
- Gebrek aan gewenste nieuwe functionaliteit

De bestaande endpoints hebben ook geen documentatie. Met wat aanpassingen hadden deze endpoints wel hergebruikt kunnen worden, maar dit zou ten koste gaan van de structuur en de leesbaarheid van de code van de backend.

Het opnieuw schrijven van de endpoints geeft de optie om de structuur goed op te zetten en hierbij rekening te houden met nieuwe features.

4.4.3.2 Backend Implementatie nieuwe API-endpoints

Er is gekozen om de nieuwe API-endpoints duidelijk te scheiden van zowel de oude endpoints van de app en de endpoints van het managementdashboard.

Ook wordt er geprobeerd om de REST (Representational State Transfer) architectuurstijl aan te houden. [7]

4.4.3.3 Naamgeving URL's

Een URL (Uniform Resource Locator) of webadres is een gestructureerde naam die verwijst naar gegevens online. Het meest bekende voorbeeld hiervan zijn websites, webbrowsers tonen de URL vaak in de adresbalk bovenin de pagina. [8]

Maar deze URL's worden ook voor API's gebruikt. Hier staan twee voorbeelden van de Voorstroom API.

Voorbeeld oude Endpoint:

- **{Locatie Backend}/api/MyProject**

Voorbeeld nieuwe Endpoint:

- **{Locatie Backend}/api/my/users/projects**

Deze endpoints zijn voor het ophalen van alle projecten waarin een gebruiker participeert. Deze URL is op de volgende manier tot stand gekomen:

{Locatie Backend}/api/my/users/projects

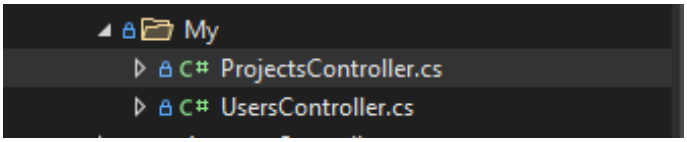
Rood	Verplicht. Geeft de locatie op het wereldwijde web aan waar de backend wordt gehost.
Oranje	Geeft aan dat het dit een api endpoint is.
Groen	Bedacht voor de scheiding van bestaande endpoints met de nieuwe app endpoints. Er is gekozen voor 'my' omdat het kort is en de app is gericht op een enkele participant.
Blauw	REST Syntax. Projecten vallen onder een gebruiker, dus wordt eerst users in de url gezet.
Grijs	Laatste woord is de daadwerkelijke informatie die opgehaald wordt. In dit geval de projecten waarin de gebruiker participeert.

4.4.3.4 Structuur en Implementatie Backend

De endpoints worden in de backend gedefinieerd als een Controller. Om het overzichtelijk te houden staan de nieuwe endpoints in het mapje ‘My’ hetzelfde woord dat in de nieuwe URL’s wordt gebruikt.

De naamgeving van het bestand zelf geeft aan wat er zich in de controller bevindt.

Figuur 8: Bestandsstructuur Controllers



Hieronder is de ProjectsController te zien met de functie voor het ophalen van Projecten. De andere Controllers hebben een vergelijkbare structuur.

Figuur 9: Projectscontroller

```
[Authorize]
[Route("api/my/[controller]")]
[ApiController]
public class ProjectsController : BaseApiController
{
    private readonly IProjectRepository _projectRepository;
    private readonly IProjectProductionPeriodRepository
_projectProductionPeriodRepository;

    public ProjectsController(
        ZonDbContext db,
        IAccountRepository accountRepository,
        IInitiativeRepository initiativeRepository,
        IFeatureAccessService featureAccessService,
        IMapper mapper,
        IProjectProductionPeriodRepository projectProductionPeriodRepository,
        IProjectRepository projectRepository)
        : base(db, accountRepository, initiativeRepository, featureAccessService,
mapper)
    {
        this._projectRepository = projectRepository;
        this._projectProductionPeriodRepository = projectProductionPeriodRepository;
    }


    [HttpGet("{id}")]
    public async Task<ActionResult<MyProjectModel>> GetProjectById(string id)
    {
        var project = await this._projectRepository.GetSimpleEntityById(id);

        if (project == null)
        {
            throw new ErrorResultException(HttpStatusCode.NotFound, "Not found");
        }

        await this.CheckRights(project.Initiative);

        var mappedResponse = this._mapper.Map<MyProjectModel>(project);

        return Ok(mappedResponse);
    }
```

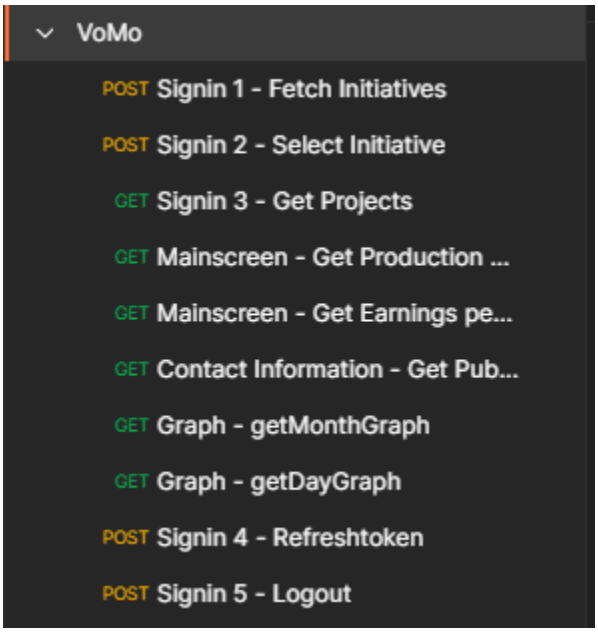
Bovenin op regel 2 is de route te zien van dit API-Endpoint. ‘api/my/[controller]’. In dit geval wordt de controller ‘/users/projects’.

In de tweede functie onderaan is te zien dat een ‘MyProjectModel’ gereturned wordt. Dit is de response die uiteindelijk door VoMo ontvangen wordt.

4.4.3.5 Lijst met API-endpoints

In deze paragraaf staan de huidige API-Endpoints waarvan de app gebruikmaakt. Hieraan worden in de toekomst waarschijnlijk nog meer Endpoints toegevoegd. Deze Endpoints zijn gedocumenteerd met behulp van Postman.

Figuur 10: Api-Endpoints



Signin
Onder de categorie 'Signin' staan alle Api-Endpoints die te maken hebben met het in-en-uitloggen.

Die procedure is relatief complex omdat er naast de logingegevens (wachtwoord en emailadres) ook een initiatief en vervolgens een project uitgekozen dient te worden.

Ook is er een Endpoint voor het verversen van de authenticatietoken voor een extra laag beveiliging.

Mainscreen
De Endpoints onder de categorie 'Mainscreen' zijn verantwoordelijk voor het leveren van de gegevens die op het hoofdscherm getoond worden. Dit zijn de productieperiodes en de desbetreffende opbrengst in Euro's en kWh.

Contactinformatie
Het Endpoint die verantwoordelijk is voor het ophalen van de contactinformatiegegevens van Voorstroom en het initiatief.

Graph
De Endpoints die gebruikt worden voor het ophalen van de maand-en-dag grafiek. Op deze grafieken wordt de opbrengst over die periode getoond.

4.4.3.6 Voorbeeld API-Call

De API-Calls die gebruikt worden voor VoMo bestaan uit 2 onderdelen:

- Headers
- Request-en-Respons Body

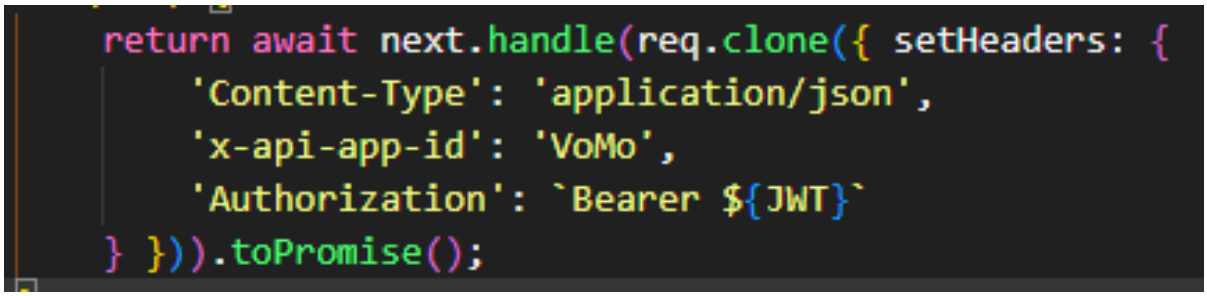
4.4.3.7 Header

API-headers zijn een klein beetje extra informatie bij iedere API-Call die gemaakt wordt. Deze extra informatie wordt meta-data genoemd. [9] Vaak vertellen ze iets over:

- De Request en Respons Body
- Autorisatie
- Respons Caching
- Respons Cookies

Bij VoMo worden ze gebruikt voor de Request-en-Respons Body, Autorisatie en voor het bijhouden van logs.

Figuur 11: Headers



Hieronder zijn de headers te zien die VoMo aan de meeste uitgaande API-Calls toevoegt.

- **'Content-Type': 'application/json'**

  Hier wordt verteld dat de Request-Body geformatteerd is op basis van JSON.

- **'x-api-app-id': 'VoMo'**

  Hier wordt ‘VoMo’ meegegeven. Dit wordt in de backend bewaard en hierdoor kan gezien worden dat deze call door de VoMo app is gedaan.

  Er kan hier bijvoorbeeld ook ManDash staan, dan is er duidelijk dat de call door het Managementdashboard is gedaan.

- **'Authorization': `Bearer ${JWT}`**

  Dit is een header die wordt gebruikt voor autorisatie. Het ${JWT} betekend dat de code hier de JWT-token moet plaatsen.

  Een JWT-token is een vorm van een API-key. Een soort wachtwoord waardoor de Backend weet dat het om een geautoriseerde gebruiker gaat. Een gebruiker krijgt een unieke JWT-token bij het inloggen. [10]

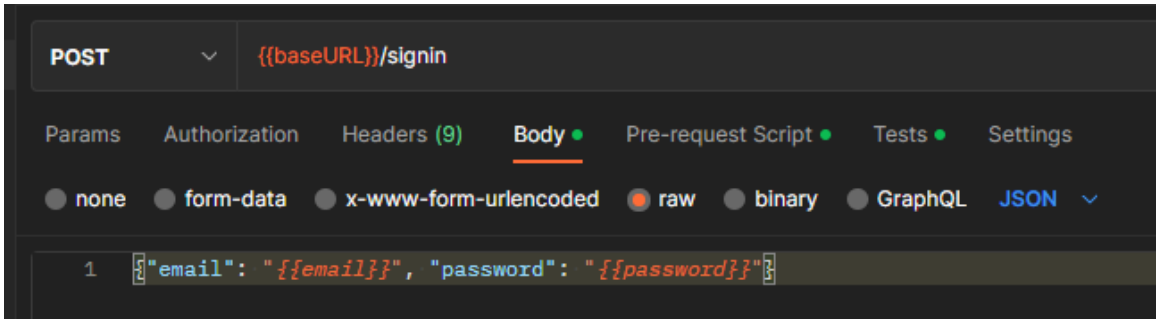
Bij het uitloggen wordt de JWT-token vernietigd. Extra uitleg over JWT-tokens in het hoofdstuk Security.

4.4.3.8 Request-en-Respons Body

De request-en-respons body's bevatten de daadwerkelijke data die van de backend naar de app wordt gestuurd en vice versa.

Zoals de naam al aangeeft is een request-body de inhoud die verstuurd wordt en is een respons body de inhoud die ontvangen wordt.

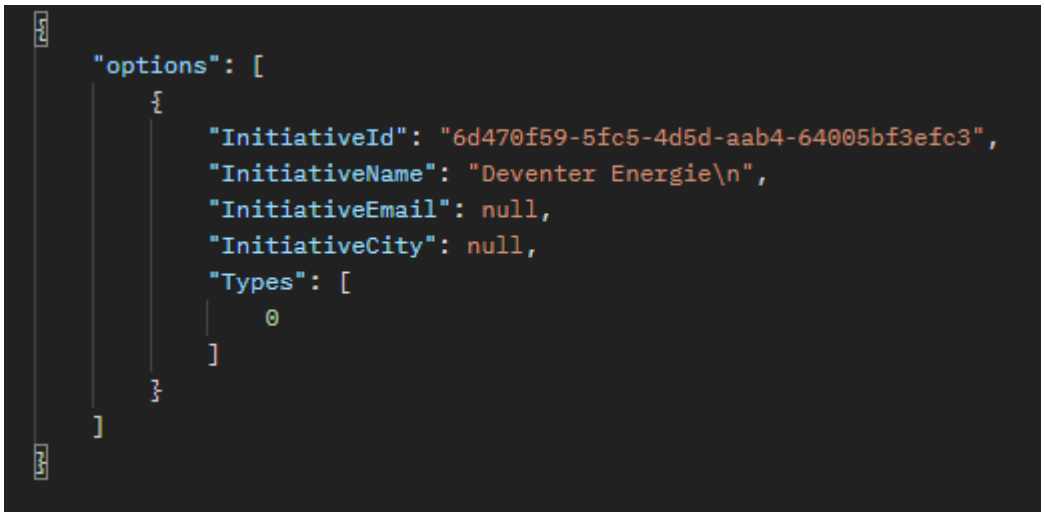
Figuur 12: API-Call Inloggen



Hier is de API-call die gebruikt wordt voor het inloggen. De app stuurt het emailadres en het wachtwoord naar de backend toe.

Het email en wachtwoord staan dus in de request-body, hier wordt gebruik gemaakt van de JSON-syntax.

Figuur 13: Voorbeeld Response Body Inloggen



Dit is de respons die ontvangen wordt na het inloggen. Het inloggen bij Voorstroom is iets complexer. Er moet namelijk bij het inloggen ook een initiatief geselecteerd worden. Dus de gebruiker krijgt eerst een lijst met initiatieven om uit te kiezen.

Deze Respons-Body is ook in de JSON-syntax, het bestaat uit een lijst met initiatieven. In dit geval staat er echter maar 1 initiatief in die lijst.

4.4.4 Security

Security is een belangrijk deel van softwareontwikkeling. Bij deze app worden er omgegaan met gevoelige gebruikersgegevens zoals hun volledige naam, woonplaats en adres.

Dit hoofdstuk licht een aantal van de interessante Security keuzes verder toe.

4.4.4.1 JWT-token

Tokens zijn stukjes data die precies genoeg informatie bevatten om de identiteit van de gebruiker te kunnen bepalen. Hierna kunnen ze gebruikt worden voor het autoriseren van acties. [11]

Tokens zijn essentieel in het autorisatie en authenticatieproces van (web)applicaties.

Er zijn een aantal standaarden voor deze ‘access tokens’, een veelgebruikte is de JSON Web Token (JWT). De basisstructuur hiervan past bij de typische JWT-structuur.

Figuur 14: Voorbeeld JWT-Token


```
{
  "iss": "https://YOUR_DOMAIN/",
  "sub": "auth0|123456",
  "aud": [
    "my-api-identifier",
    "https://YOUR_DOMAIN/userinfo"
  ],
  "azp": "YOUR_CLIENT_ID",
  "exp": 1489179954,
  "iat": 1489143954,
  "scope": "openid profile email address phone read:appointments"
}
```

Het is belangrijk om in gedachten te houden dat een JWT-token toegang tot dezelfde gegevens verleend die een gebruiker zou kunnen zien als deze zou inloggen. Er zijn een aantal methoden om dit beveiligingsrisico te beperken. De belangrijkste hiervan is de beperkte levensduur van een JWT-token. De levensduur is meestal een paar uur of een aantal dagen.

Nadat deze is verlopen kan die niet meer gebruikt worden en moet de gebruiker opnieuw authenticatie tonen voor een nieuwe JWT-token.

4.4.4.2 Wat is een refreshtoken?

Een refreshtoken kan gebruikt worden om een nieuwe JWT-token op te halen zodra de oude verlopen is. Hierdoor hoeft de gebruiker niet opnieuw in te loggen.

Een refreshtoken heeft daarom een veel langere levensduur hebben dan een JWT-token.

4.4.4.3 Beveiliging refreshtoken

Omdat net als een jwt-token een refreshtoken kan uitlekken moet hierbij ook nog een aantal beveiligingsstappen worden genomen. [11]

De belangrijkste bij de VoMo is het gebruik van refreshtoken rotatie. Zodra een jwt-token verloopt en de refreshtoken wordt gebruikt krijgt de gebruiker zowel een nieuwe jwt als een refreshtoken.

De oude refreshtoken is hierna niet meer valide, dit beperkt het risico van een uitgelekte refreshtoken. Verder vervalt een refreshtoken ook als de gebruiker opnieuw uit-of-inlogt.

4.4.4.4 Guards

Guards zijn een onderdeel van de Angular Routing en hebben vaak als doel het voorkomen van ongeautoriseerde toegang tot pagina’s.

Angular Route Guards bieden de mogelijkheid om toegang tot een route te verlenen of in te trekken gebaseerd op condities. Deze condities kunnen zelf ingesteld worden. [12]

Bij VoMo is er een guard voor een aantal introductieslides en voor het inloggen.

Figuur 15: Guards

```
async canActivate(): Promise<boolean> {
    const hasSeenIntro = await this.hasSeenIntro();
    if (!hasSeenIntro) {
        this.router.navigateByUrl('/intro', { replaceUrl: true });
        return false;
    }

    const token = await this.authService.receiveCheckedJWT();
    if (!token) {
        this.router.navigateByUrl('/login')
        return false;
    }
    return true
}
```

4.4.5 Notificatieprovider

Een pushnotificatie is een geautomatiseerd bericht van een app naar een gebruiker ongeacht of de gebruiker de app open heeft of niet. [13]

Pushnotificaties kunnen voor een aantal dingen gebruikt worden. In het geval van VoMo wordt het gebruikt om berichten van de energieleverancier en meldingen over updates te sturen.

Voor het versturen van pushnotificatie is een notificatieprovider nodig.

4.4.5.1 Wat is een Notificatieprovider?

De taak van een notificatieprovider is het maken en bezorgen van berichten naar de gebruikers volgens een zelf ingestelde dienstregeling.

Er zijn verscheidene Notificatieproviders, de keuze hiervoor is afhankelijk van het platform waarop de app wordt gebruikt.

Voor pushnotificaties op Android wordt ‘Firebase Cloud Messaging’ (FCM) gebruikt. Voor IOS moet de ‘Apple Push Notification Service’ (APNS) worden gebruikt.

4.4.5.2 Keuze Notificatieproviders

Naast FCM en APNS zijn er ook nog notificatieproviders van andere partijen. Deze sturen dan een notificatie naar FCM en APNS die het vervolgens naar de app sturen.

In dit geval lijkt het onlogisch om nog een 3e notificatieprovider te gebruiken, dit maakt het onnodig complex. Deze 3e partij notificatieproviders bieden vaak extra functionaliteit. Maar aangezien bij VoMo worden alleen relatief simpele notificaties gestuurd biedt dit weinig toegevoegde waarde.

In de huidige app wordt er ook gebruik gemaakt van FCM en APNS.

4.4.6 Teststrategie

Er is weinig ervaring binnen Voorstroom met Front-end testen. Er is daarom besloten om de standaard Angular-en-Ionic teststrategie aan te houden. [14]

Het doel van deze testen is om problemen met de code te ontdekken, niet om uit te vinden of de code correct is. Dit is een subtiel maar belangrijk verschil.

Als er geprobeerd wordt om te bewijzen dat de code correct is, is het waarschijnlijker dat er het ‘happy path’ door de code wordt gevolgd.

Als er opzoek gegaan wordt naar problemen is er meer kans om de bugs te vinden die zich daar verstopt hebben.

Standaard wordt door Ionic ‘Unit Testen’ en ‘End-to-end Testen’ gegenereerd. Deze worden in dit hoofdstuk toegelicht.

4.4.6.1 Unit Testen

Unit tests voeren een enkele unit (Component, Pagina, Service, Pipe etc.) geïsoleerd uit. Deze unit is dus afgesloten van de rest van het systeem. Deze isolatie wordt bereikt door het injecteren van ‘Mock Objecten’ in plaats van de normale dependencies van de app.

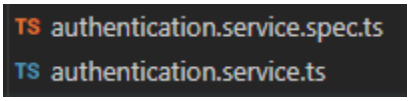
Er wordt aangeraden om ‘Jasmine’ te gebruiken voor het creëren van ‘Mock Objecten die de plaats van de reguliere dependencies innemen.

4.4.6.2 Spec-bestanden

Unit tests worden bewaard in '.spec' bestanden. Er is 1 '.spec' bestand per entiteit (Component, Pagina, Service, Pipe etc.). Ze staan naast het bestand dat zij moeten testen en hebben verder een identieke naam.

Bijvoorbeeld:

- Authentication.service.ts (Service)
- Authentication.service.spec.ts (Unit Test)



Ieder '.spec' bestand bevat een 'describe' functie die de volledige test definieert. Genest daarbinnen zitten andere 'describe' functies die grote delen van functionaliteit beschrijven.

Binnen een 'describe' functie kunnen 'it' funcites zitten die individuele test cases definieren.

Figuur 16: .SPEC bestand

```
describe('Calculation', () => {
  describe('divide', () => {
    it('calculates 4 / 2 properly' () => {});
    it('cowardly refuses to divide by zero' () => {});
    ...
  });
```

Zoals hier te zien hebben de 'describe' en 'it' calls een tekst label. In goed geschreven testen vormen de 'describe' en 'it' calls een volledige zin die de test beschrijft.

4.4.6.3 Pagina's en Componenten

Pagina's zijn gewoon Angular Componenten. Pagina's en Componenten worden allebei getest volgens de Angular Component Testing Guidelines.

Een Angular component bestaat uit een HTML Template en een Typescript class. De component ontstaat als deze twee goed met elkaar samen werken. Dat is het doel van deze testen.

Om dit te testen moet het component zijn Host-element nagemaakt worden in de Browser zijn DOM (Domain Object Model), vervolgens kan onderzocht worden wat de component class zijn interactie is met de DOM. [15]

4.4.6.4 Services

Angular Services zijn vaak in te delen in twee categorieen:

1. Utility Services die berekening en andere complexe operaties doen.
2. Data Services die vooral API-Calls doen en eventuele verdere data manipulatie.

De methode om een ‘Utility Service’ of een ‘Data Service’ te testen verschilt.

Utility Services

De aanbevolen methode voor Utility Services is om de service eerst te instantieren en handmatige ‘mocks’ te injecteren voor de dependencies die de Service heeft. Op deze manier kan de service volledig in isolatie getest worden. Dit lijkt erg op de manier waarop pagina’s en componenten getest worden.

Data Services

Als er http-operaties plaatsvinden in de service wordt er waarschijnlijk gebruik gemaakt van Angular zijn HttpClient Service. In dit geval wordt het aangeraden om de Angular ‘HttpClientTestingModule’ te gebruiken.

Hiermee kan een ‘neppe’ backend opgezet worden zodat de Data Service in volledige isolatie getest kan worden zonder dat de daadwerkelijke Backend nodig is.

4.4.6.5 End-to-end Testen

Het doel van ‘end-to-end’ testen is om te controleren of de applicatie werkt als een geheel. Vaak wordt hierbij verbinding gemaakt met live data.

Unit testen focussen zich op stukjes code in isolatie en kunnen dus op laag niveau de applicatie logica controleren. ‘End-to-end’ testen richten zich op de User stories of de standaard gebruiksscenario’s.

Het doel van de ‘end-to-end’ testen is om problemen te ontdekken als alle individuele stukken van de applicatie samenkomen. Vaak komen hierdoor problemen met de algemene architectuur van de applicatie aan het licht.

4.4.6.6 End-to-end Teststructuur

Het doel van ‘end-to-end’ testen is om te controleren of de applicatie werkt als een geheel. Vaak wordt hierbij verbinding gemaakt met live data.

Unit testen focussen zich op stukjes code in isolatie en kunnen dus op laag niveau de applicatie logica controleren. ‘End-to-end’ testen richten zich op de User stories of de standaard gebruiksscenario’s.

Het doel van de ‘end-to-end’ testen is om problemen te ontdekken als alle individuele stukken van de applicatie samenkomen. Vaak komen hierdoor problemen met de algemene architectuur van de applicatie aan het licht.

De standaard testapplicatie bestaat uit 4 bestanden:

Protractor.conf.js	Een Protractor configuratiebestand.
Tsconfig.e2e.json	Specificeert de specifieke Typescript configuratie voor de testapplicatie.
Src/app.po.ts	Een paginaobject die de methodes bevat voor het navigeren door de applicatie en het manipuleren van elementen op de pagina.
Src/app.e2e-spec.ts	Een testscript. Hier kunnen er meerdere van zijn.

4.4.6.7 Paginaobjecten

Paginaobjecten kapselen de HTML in voor een enkele pagina in een Typescript class, deze Typescript class werkt als een API dat het testscript gebruikt om de applicatie te gebruiken.

Het inkapselen van de HTML zorgt ervoor dat de testen veel beter leesbaar zijn. Het maken van goede paginaobjecten is de truc om goede en onderhoudbare end-to-end testen te maken.

5 Conclusie en Reflectie

In tegenstelling tot veel studieprojecten en beroepspraktijk, is een afstudeertraject iets dat ik alleen moet uitvoeren. In het begin van het afstudeertraject had ik hier wel moeite mee. Het hielp niet mee dat Voorstroom welgeteld 1 Software Engineer in dienst had en dat er eigenlijke geen enkele vorm van documentatie over de bestaande code was.

Hierdoor verliep het eerste deel van de afstudeeropdracht moeizaam. Achteraf heb ik tijdens deze periode ook een aantal fouten gemaakt. Er is veel tijd in de code gestopt terwijl achteraf gezien het logischer was om mij meer op het opstellen van requirements en het ontwerp te stoppen. Meer tijd om Angular en Ionic te leren was ook verstandig geweest.

Ik heb het nu geleerd door het meteen in de praktijk toe te passen, maar het nadeel daarvan was dat in het begin een gedrocht van een app is ontstaan die veel werk nodig had om beter opgezet te worden.

Hierdoor heeft het afstuderen ook een kwartiel vertraging opgelopen. Ik had achteraf waarschijnlijk het wel zonder gekund, maar dan zou ik zowel ontevreden over de documentatie als de app zelf zijn. Dat is nou niet bepaald een fijne manier om je verdediging in te gaan.

Maar uiteindelijk zijn de problemen verholpen! Over de code van de app zelf heb ik niks dan goede dingen te vertellen, het zegt al genoeg dat er nu door een andere medewerker van Voorstroom verder aan wordt gewerkt. Er is ook echt de intentie om deze uit te brengen. De app wordt dan waarschijnlijk door een paar honderd mensen gebruikt.

De documentatie vind ik uiteindelijk ook meer dan redelijk. Het schrijven van het afstudeerdossier was alleen wel pittig. Bij het ontwikkelen van een app schrijf je een heel ander afstudeerdossier dan bijvoorbeeld een onderzoek. Uiteindelijk is de structuur wat onconventioneel maar desondanks erg duidelijk.

Ik wil mijn reflectie eindigen met een bedankje aan het team van Voorstroom en Inversable. Ik ben er de afgelopen 3 kwartielen met plezier geweest. Ondanks dat het hard werken was heb ik mij er erg vermaakt, ik had meestal zin om naar kantoor te gaan. Er was altijd wel iets te beleven en de koffie was een stuk beter dan op het Saxion.

Ik ben aan de ene kant opgelucht dat ik klaar ben met afstuderen, maar ik vind het ook jammer, ik ben er met veel plezier geweest!

6 Aanbevelingen

Dit hoofdstuk bevat een aantal persoonlijke aanbevelingen voor de verdere ontwikkeling van VoMo. Deze richten zich op alles wat nog gedaan of verbeterd moet worden voordat de app uitgebracht kan worden.

Deze aanbevelingen zijn gegroepeerd per deelvraag.

6.1 Deelvraag 1: Aan welke eisen moet VoMo voldoen?

Aan de requirements voor VoMo is voor weinig toe te voegen. Er is al een redelijk compleet beeld over hoe de app er uit moet gaan zien en welke functionaliteit deze in de toekomst moet hebben.

Er is vooral een aanbeveling om kritisch te blijven kijken naar welke functionaliteit echt nodig is. Onnodige feature creep kost niet alleen veel ontwikkelingstijd, het zorgt ook voor een grotere leercurve voor nieuwe gebruikers. Deelnemers aan energiecollectieven zijn over het algemeen wat ouder, een vriendelijke leercurve heeft voor deze groep waarschijnlijk meer waarde dan een app met uitgebreide functionaliteit.

6.2 Deelvraag 2: Hoe ontwerp je een geschikte User Interface voor VoMo?

Binnenkort kan Voorstroom gebruik maken van een UX/UI specialist die in dienst is bij Inversable. Dit is handig, want de Interface die de app biedt is momenteel functioneel en duidelijk, maar niet per se erg mooi. Er zijn twee punten waarbij uitbreiding dit zou kunnen verbeteren.

1. Bruikbaarheid kleinere schermen

Sommige delen van de interface zijn minder geschikt om te gebruiken op een klein scherm. Voornamelijk de grafieken zouden eigenlijk op een andere manier vormgegeven moeten worden.

Dit probleem is deels aanwezig door de ruime ondersteuning van oudere Android en iOS versies. Smartphones uit 2013 zien er heel anders uit dan tegenwoordig, maar worden wel door VoMo ondersteund.

Er zou gekozen kunnen worden om een aantal oudere versies uit te sluiten, maar dit heeft als nadeel dat sommige gebruikers de app helemaal niet meer kunnen gebruiken. Het aanpassen van de UI is waarschijnlijk een betere manier, maar kost uiteraard wel veel meer ontwikkelingstijd.

2. Thema en Kleuren

Momenteel is er alleen een wit-thema beschikbaar met daarin verwerkt de typische Voorstroom kleuren. Smaken verschillen, maar ik denk niet dat het kwaad kan als er kritisch gekeken wordt of het verstandig is om deze kleuren te gebruiken voor de app.

De hoofdkleur van Voorstroom is een soort neongroen (door sommige stagiaires liefkozend ‘babykotsgroen’ genoemd). Om met deze kleur een mooi uitziende interface te maken is erg moeilijk.

Daarnaast is er geen donker thema beschikbaar terwijl tegenwoordig er wel veel mensen gebruik maken van Dark Mode. Daarnaast is het maken van een donker thema niet heel veel werk.

6.3 Deelvraag 4: Wat is de beste manier om VoMo technisch vorm te geven en te laten integreren met alle huidige systemen die gebruikt worden voor Voorstroom.nl?

Over de structuur en code van de app zelf heb ik niet veel aan te merken. Ook de integratie met de Voorstroom backend verloopt prima. Er is geen belemmering hierin om de app uit te brengen.

Desondanks heb ik wel de aanbeveling om de Backend beter te documenteren. Aan het begin van het afstuderen was er geen enkele documentatie hiervan en dit maakte de softwareontwikkeling moeilijk en tijdrovend.

De tijd die gestopt wordt in het documenteren en opschonen van de backend wordt waarschijnlijk teruggewonnen door het sneller kunnen ontwikkelen in de toekomst.

7 Bibliografie

[1] Wikipedia, "Requirements Analysis," 4 7 2022. [Online]. Available: https://en.wikipedia.org/wiki/Requirements_analysis. [Accessed 10 11 2022].

[2] Nick Babich, "Prototyping 101: The difference between low-fidelity and high-fidelity prototypes and when to use each," Adobe Blog, 29 11 2017. [Online]. Available: https://blog.adobe.com/en/publish/2017/11/29/prototyping-difference-low-fidelity-high-fidelity-prototypes-use. [Accessed 10 11 2022].

[3] "The 10 Best Hybrid App Frameworks in 2022," Mobile App Daily, 12 10 2022. [Online]. Available: https://www.mobileappdaily.com/best-hybrid-app-frameworks. [Accessed 15 11 2022].

[4] Jscrambler, "12 Frameworks for Hybrid Mobile Apps," Jscrambler, 15 7 2022. [Online]. Available: https://blog.jscrambler.com/12-frameworks-for-mobile-hybrid-apps. [Accessed 15 7 2022].

[5] Saurabh Barot, "Best Hybrid App Development Framework in 2022," A GlowID IT Solutions, 14 9 2021. [Online]. Available: https://aglowiditsolutions.com/blog/best-hybrid-app-development-framework/. [Accessed 15 11 2022].

[6] "Google Trends," Google, 15 11 2022. [Online]. Available: https://trends.google.com/trends/explore?date=today%205-y&q=flutter,nativescript,react%20native,ionic. [Accessed 15 11 2022].

[7] "Stack Overflow Trends," Stack Overflow, 15 11 2022. [Online]. Available: https://insights.stackoverflow.com/trends?tags=flutter%2Cionic-framework%2Creact-native%2Cnativescript. [Accessed 15 11 2022].

[8] "Cross-platform Mobile Frameworks used by Software Developers worldwide from 2019 to 2021," Statista, 7 2021. [Online]. Available: https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/. [Accessed 15 11 2022].

[9] "What is Angular," Google, 28 2 2022. [Online]. Available: https://angular.io/guide/what-is-angular. [Accessed 11 11 2022].

[10] "Mobile & Tablet iOS Version Market Share Netherlands," GlobalStats statcounter, 11 11 2022. [Online]. Available: https://gs.statcounter.com/ios-version-market-share/mobile-tablet/netherlands#monthly-202110-202210-bar. [Accessed 11 11 2022].

[11] "Mobile Android version Market Share Netherlands," Globalstats StatCounter, 11 11 2022. [Online]. Available: https://gs.statcounter.com/android-version-market-share/mobile/netherlands#monthly-202110-202210-bar. [Accessed 11 11 2022].

[12] Chinmayee Deshpande, "Introduction To Angular Service and its Features," simplilearn, 8 8 2022. [Online]. Available: https://www.simplilearn.com/tutorials/angular-tutorial/angular-service. [Accessed 14 11 2022].

[13] Unknown, "What is an API?," Red Hat, 2 6 2022. [Online]. Available: https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces. [Accessed 7 11 2022].

[14] Unknown, "What is a REST API," Red Hat, 8 3 2020. [Online]. Available: https://www.redhat.com/en/topics/api/what-is-a-rest-api. [Accessed 7 11 2022].

[15] Unknown, "URL," Wikipedia, 6 10 2022. [Online]. Available: https://en.wikipedia.org/wiki/URL. [Accessed 7 11 2022].

[16] Unknown, "API Headers - What are they? Examples & More [Explained]," Apipheny, [Online]. Available: https://apipheny.io/api-headers/. [Accessed 7 11 2022].

[17] Apipheny Team, "What's an API Key? API Key Meaning Explained for Beginners," Apipheny, [Online]. Available: https://apipheny.io/what-is-an-api-key/. [Accessed 7 11 2022].

[18] Dan Arias and Sam Bellen, "What Are Refresh Tokens and How to Use Them Securely," auth0, 7 10 2021. [Online]. Available: https://auth0.com/blog/refresh-tokens-what-are-they-and-when-to-use-them/. [Accessed 10 11 2022].

[19] Nwose Lotanna Victor, "Angular Basics: CanActivate - Introduction to Routing Guards," Progress Telerik, 8 3 2022. [Online]. Available: https://www.telerik.com/blogs/angular-basics-canactivate-introduction-routing-guards. [Accessed 15 11 2022].

[20] Sasha Langholz, "What is a push notifications service and how does it work?," One Signal, 12 1 2022. [Online]. Available: https://onesignal.com/blog/what-is-a-push-notifications-service-and-how-does-it-work/. [Accessed 10 11 2022].

[21] "Testing," Ionic Framework, [Online]. Available: https://ionicframework.com/docs/angular/testing. [Accessed 14 11 2022].

[22] "Basics of testing components," Angular, [Online]. Available: https://angular.io/guide/testing-components-basics. [Accessed 14 11 2022].

[23] "Feature Creep," Wikipedia, 3 11 2022. [Online]. Available: https://en.wikipedia.org/wiki/Feature_creep. [Accessed 14 11 2022].