Mobile Operator Support

Afstudeerverslag

Door: Bart Krabbe (450807)
School: Saxion Hogeschool Enschede
Opleiding: HBO-ICT SE
Bedrijf: Voortman Steel Group B.V.
Bedrijfsbegeleider: Dominic Stremmelaar
Saxion begeleider: Theo Lansink
Datum:16-01-2022

Voorwoord

Voor u ligt de scriptie ’Mobile Operator Support’. Het onderzoek is uitgevoerd bij Voortman Steel Machinery (Onderdeel van Voortman Steel Group). Deze scriptie is geschreven in het kader van mijn afstuderen bij de opleiding HBO-ICT software engineering, aan de hogeschool Saxion in Enschede. Van Augustus 2021 t/m Januari 2022 ben ik bezig geweest met het analyseren, onderzoeken, ontwerpen en het bouwen van de applicatie en het schrijven van deze scriptie.

Nadat ik een plan van aanpak heb opgesteld en de hoofdvraag heb bedacht is het proces begonnen. Deze hoofdvraag heb ik opgedeeld in verschillende opeenvolgende deelvragen zodat zij mij helpen bij het beantwoorden van de hoofdvraag. Deze afstudeerperiode is op te delen in de volgende stukken respectievelijk: uitvoerig onderzoek, ontwerpen applicatie (zowel functioneel als technisch), realiseren van de applicatie en de applicatie testen bij eindgebruikers.

Bij dezen wil ik graag mijn begeleiders bedanken voor de fijne begeleiding en ondersteuning tijdens het afstuderen. Dominic Stremmelaar bij het Voortman Steel Machinery en Theo Lansink vanuit de opleiding.

Tevens wil ik ook mijn vrienden (tijdens thuiswerken online) en collega’s (als ik naar werk kon) bedanken voor de gezellige afstudeerperiode en samenwerkingen.

Ik wens u veel leesplezier toe,

Bart Krabbe

Begrippenlijst

Lijn / machinelijn / productielijn	Met een lijn wordt bedoeld op een reeks machines die verbonden zijn met rollenbanen en hierdoor een geheel process vormen om een staalproduct te creëren.
Operator	medewerkers die de machines bedienen
Hybride app	App voor zowel het web als voor native
AWS	Amazon web services
Native	Mobiele device zoals Android of iOS
CI/CD	Continuous Integration / Continuous Deployment
VSG / VSM / VSC	Voortman Steel Group / Voortman Steel Machinery / Voortman Steel Construction

1. Samenvatting

Deze afstudeeropdracht gaat over de Mobile Operator Support applicatie. Deze applicatie is een idee van Voortman Steel Group om operators van een productielijn een app te geven die inzicht biedt in de productie van staal bewerking en verwerkingsmachines. Ook moet deze applicatie notificaties tonen, met informatie over de status en machine, als de machines van status veranderen. Dit wordt gedaan om het probleem zo snel mogelijk te kunnen oplossen. Alleen voordat Voortman kan beginnen aan de applicatie moet er een keuze worden gemaakt tussen de verschillende hybride frameworks om een native en web applicatie te maken. Hiervan hebben ze geen voorkennis, dus moet dit worden onderzocht.

Voordat de verschillende frameworks met elkaar kunnen vergelijken, moet er eerst gekeken worden naar wat Voortman echt nodig heeft en wil hebben. Door meerdere interviews te houden met de medewerkers van Voortman en meerdere interne bestanden te hebben gelezen konden de requirements worden opgesteld. Uit deze requirements zijn vervolgens 21 eisen opgesteld waaraan de frameworks beoordeeld kunnen worden.

De gekozen frameworks om te onderzoeken zijn: React Native, Ionic, Flutter, Progressive Web Apps en Native (Android & iOS). Per framework is er veld en desk research gedaan om te ontdekken of ze zich aan de eisen kunnen houden. Uit dit onderzoek bleek dat theoretisch, Flutter het beste framework is en React Native het twee na beste framework is. Omdat het toch interessant is voor Voortman, is er voor gekozen om van 2 frameworks een implementatie te maken.

Hierdoor is er een implementatie gemaakt met Flutter en React Native. Deze implementaties testen meerdere eisen in de praktijk. Eisen die hierbij getest zijn, zijn bijvoorbeeld of de frameworks de biometrie van native devices kunnen gebruiken en of de frameworks grafieken en diagrammen konden maken en tekenen. Als de gemaakte apps vergeleken worden is de Flutter app beter, sneller en mooier geworden, ook kan Flutter zich beter houden aan de eisen. Dus is Flutter ook beter in de praktijk.

Dus kan er geconcludeerd worden dat Flutter het beste framework is voor Voortman om hun hybride applicatie mee te ontwikkelen.

2. Inleiding

Voortman Steel Machinery (VSM) maakt boor-, frees- en snijmachines voor de staalindustrie en verkoopt deze wereldwijd. Dit zijn vaak geen losse machines, maar hele productielijnen. Voortman is wereldwijd actief en onderscheid zich ten aanzien van concurrenten door uitstekende after sales en service diensten. Ze willen ervoor zorgen dat klanten het maximale uit de machines kunnen halen. Ze geven onder andere trainingen aan operators die de machines moeten bedienen. Hierdoor kunnen de operators zich focussen op maximale output van de machines. Nu wil Voortman het makkelijker maken voor deze Operators m.b.v. een applicatie voor zowel web als native, deze applicatie biedt hierbij extra inzicht in informatie over de machines en de statushistorie van de machines.

Voortman heeft alleen nooit een mobiele applicatie ontwikkeld en wil dit wel zelf doen. Hierom het onderzoek Mobile Operator Support. Dit onderzoek houdt in dat Voortman graag wil weten welk framework ze het beste kunnen gebruiken voor de applicatie.

Hieruit komt een groot onderzoek gerolt naar welk framework het best geschikt is voor Voortman. Het onderzoek vergelijkt de frameworks met elkaar. De twee frameworks die het beste uit het onderzoek komen worden verder getest door een proof of concept te maken met deze frameworks.
Bij deze proofs of concept wordt gekeken of het onderzoek overeenkomt met een kleine implementatie van de applicatie, zodat we uiteindelijk een conclusie kunnen trekken uit het onderzoek.

Eerst kaarten we het probleem verder aan, daarna wordt het probleem geanalyseerd en worden de requirements en eisen opgesteld. Dan komt een deel over het onderzoek dat naar de frameworks is gedaan en wordt hierna het functioneel en technische ontwerp besproken, het verslag eindigt met de implementatie en het advies over welk framework het beste is voor Voortman om te gebruiken.

3. Organisatieschets

Dit hoofdstuk behandelt wat Voortman Steel Group is en doet.

Figuur 1 Logo VSG

3.1 Voortman Steel Group

Voortman Steel Group B.V. is een metaalverwerkingsbedrijf die ook staalproces machines bouwt. Ook is dit het bedrijf waar dit onderzoek plaats gaat vinden. Dit hoofdstuk gaat dieper in op de geschiedenis van het bedrijf maar ook over de huidige situatie en hun visie voor de toekomst. (figuur 1: het logo van VSG)

Toen Gerrit Voortman in 1970 na zijn studie technische werktuigbouwkunde bij het bedrijf van zijn broers ging werken (Voortman & Co) ontwikkelde zich een mechanisatie bedrijf dat al snel groeide en de basis legde voor wat Voortman nu is. Naast de mechanisatie begint Voortman met het ontwerpen en bouwen van staalconstructies. Na enkele jaren splitst het bedrijf in twee divisies: Voortman Steel Construction (VSC) en Voortman Steel Machinery (VSM)

3.2 Voortman Steel Construction

Voortman Steel Construction is vanaf het ontstaan gefocussed op het produceren en het bouwen van staalproducten. Klanten kunnen hierbij orders plaatsen voor deze producten of zelfs Voortman inschakelen om staalconstructies in elkaar te zetten. Sinds enkele jaren richten ze zich ook op het ontwerpen en realiseren van bedrijfspanden, parkeergarages, industrie faciliteiten (denk hierbij aan de stalen constructies van bijvoorbeeld een olie verwerkingsbedrijf).

Het doel van VSC is complexe projecten realiseren met passie. Ook willen ze ervoor zorgen dat de collega's die aan projecten werken tot hun recht komen en met veel werkplezier kunnen rondkomen. "Fouten maken is niet erg, als we er met elkaar maar van leren. Durf te doen; alleen dan kom je verder!" (Voortman Steel Group, c)

Het bedrijf telt totaal meer dan 500 medewerkers en 6 internationale vestigingen. VSG is dus een groot bedrijf.

3.3 Voortman Steel Machinery

Deze divisie van Voortman gaat over de bouw en verkoop van productielijnen / metaalbewerkingsmachines. De klant kan bij Voortman komen om machines te bekijken en te bestellen. Vervolgens gaat VSM deze machines/productielijnen bouwen/customizen en deze installeren bij de klant. (Zie figuur 2 om verschillende machines te zien die Voortman aanbiedt aan klanten) Om de klantenservice hoog te

Figuur 2 Expohal waarin klanten de vele machines kunnen zien en voorproefjes krijgen wat de machines allemaal kunnen

houden houdt Voortman de software van de machines bij en kan de klant er zonder probleem gebruik maken van deze productielijnen. Het leuke is dat VSC eigenlijk een klant is van VSM, want ze willen zelf ook de beste machines voor hun eigen productie. Zodat zij het proces kunnen verkorten (efficiënter kunnen maken) en hierbij de beste in het veld zijn.

VSM kijkt naar de klant en wat hun precies willen hebben. Wat ze met deze producten gaan doen en of de machine wel het juiste middel is voor die taak. Zij willen niet alleen machines produceren en verkopen maar toegevoegde waarde leveren. Dit doen ze allemaal zodat ze klantwaardering hoog houden. Het liefst zouden ze problemen oplossen waarvan de klant nog niet wist dat hij ze had. En dan kijken ze ook naar interne klanten (van de klanten) bij dit proces. Ook willen ze continu ontwikkelen en blijven leren zodat ze de beste op de markt kunnen zijn. (Voortman Steel Group, s.d. b)

3.4 Toekomst van Voortman Steel Group

Momenteel is Voortman Steel Group veel bezig met een geheel geautomatiseerde productielijn voor het opslaan, herkennen, transporteren, lassen en assembleren van constructiestaal voor staalbouwers. Ze werken hier met een groep van high tech kennis- en ketenpartners uit het land. Het project waar ze met de groep aan het werken zijn heet ‘WELD4.0’. Dit project maak het mogelijk voor mkb-staalbouwers om hun fabriek geheel te automatiseren. (Voortman Steel Group, a)

Ze hebben sinds kort ook een automatisch las systeem ontwikkeld genaamd “The Fabricator”. Deze machine last dus verschillende staalproducten aan elkaar, zodat de productielijn nog efficiënter wordt. Dit was ook een wereldprimeur, want zoiets was nog niet gemaakt. (Voortman)

Figuur 3 The Fabricator (automatische lasrobot)

4. Probleemstelling

Voortman Steel Machinery (VSM) maakt boor-, frees- en snijmachines voor de staalindustrie en verkoopt deze wereldwijd. Dit zijn vaak geen losse machines, maar hele productielijnen. Voortman is wereldwijd actief en onderscheid zich ten aanzien van concurrenten door uitstekende after sales en service diensten. Ze willen ervoor zorgen dat klanten het maximale uit de machines kunnen halen. Ze geven onder andere trainingen aan operators die de machines moeten bedienen. Hierdoor kunnen de operators zich focussen op maximale output van de machines. Nu wil Voortman het makkelijker maken voor deze Operators m.b.v. een applicatie voor zowel web als native, deze applicatie biedt hierbij extra inzicht in informatie over de machines en de statushistorie van de machines.

4.1 Context

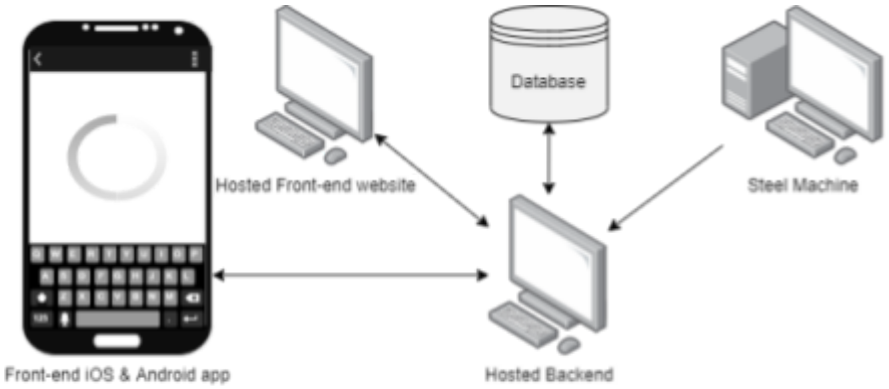
Voortman Steel Machinery ontwikkelen machines en productielijnen waarin staal wordt verwerkt. Nu werken deze machines vaak goed, maar soms kan er een probleem ontstaan. Dit kan door factoren komen waar de klant niks aan kan doen. Dus moeten de operators die bij deze productielijnen staan dit probleem oplossen. De operators zijn getraind om deze problemen op te lossen, maar het doorhebben dat er een probleem is en het vinden van dit probleem zijn hierbij het probleem. Ten eerste moet er iemand bij de machines in de buurt staan om de error mee te krijgen. Dan moet deze operator doorhebben dat er een fout aanwezig is in de lijn. Als de operator de machine heeft gevonden waarop de fout zit, moet de operator op het machine scherm kijken wat er aan de hand is en aan de hand van die informatie een oplossing verzinnen. Dit kost tijd en heeft een negatief effect op de efficiëntie van de machine/lijn en Voortman wil dit negatieve effect minimaliseren.

Voortman ziet kansen om deze behoefte te voorzien d.m.v. een hybride app. Zij hebben een aantal eisen die aan de app moeten voldoen, hiervoor moet onderzoek gedaan worden of bepaalde frameworks de app aan deze eisen kunnen laten voldoen. Een voorbeeld van een van deze eisen is hoe gemakkelijk het framework kan worden uitgebracht in een CI/CD. Nog een voorbeeld is bijvoorbeeld hoe goed het framework diagrammen en grafieken kan tekenen. In het hoofdstuk requirements wordt dit verder aangekaart.

Om een app te ontwikkelen kan Voortman vele technologieën en ontwikkelstrategieën kiezen. Voortman heeft nog nooit een hybride app gemaakt en heeft dus geen uitgangspunt voor frameworks die zij hiervoor gebruiken kunnen. Ook de tools vinden zij interessant, want ze willen uiteindelijk een framework die een goede ervaring voor de klant biedt en als ontwikkelaar een makkelijke en simpele programmeerervaring biedt.

Ook is er geen online plaats waar Voortman de data van de klanten en machines kan ophalen. Ze willen hierbij een cloud oplossing om de data meer beschikbaar te maken. Dit laten ze deels doen door het bedrijf Inawisdom, dit bedrijf is gespecialiseerd in AWS services en helpen het Voortman team om een AWS omgeving op te zetten, waarin ze alle data omtrent machines, bedrijven en werknemers opslaan en verwerken. Maar omdat er nog geen functionele backend is moet dit gedeelte nagebootst worden door een eigen draaiende backend op te zetten. Deze backend moet

gebruik maken van een SQL database om data op te kunnen slaan. (Zie figuur 4 voor een voorbeeld van de versimpelde architectuur van de volledige app)



Figuur 4 - Simpele fullstack architectuur

4.2 Hoofdvraag

Tijdens dit onderzoek staat het onderzoeken van frameworks om een hybride app te maken centraal, daarom luidt de hoofdvraag:

Welke framework implementatie is het meest geschikt voor Voortman voor het ontwikkelen van zowel native apps als een website die voldoet aan de opgestelde requirements en eisen?

5. Analyse

In dit hoofdstuk wordt geanalyseerd wie de stakeholders zijn, wat de context is, hoe de requirements worden opgesteld en welke requirements uiteindelijk zijn opgesteld.

5.1 Stakeholders analyse

De stakeholders in dit project zijn als volgt in volgorde van meeste belang en invloed tot minste belang en invloed:

Voortman Steel Group (Veel belang, veel invloed):

Voortman Steel Group heeft veel belang voor dit project, want zij willen een product verkopen aan de klant die meer inzicht biedt in machines. Hierbij is de invloed ook groot, want zij bepalen uiteindelijk wat de applicatie moet doen en kunnen.

Klanten van Voortman (Veel belang, veel invloed):

De klanten van Voortman zijn uiteindelijk de groep die de applicatie moet gebruiken, deze gaan ook betalen voor de app en hebben hierdoor veel invloed op wat de app moet kunnen.

De klanten van Voortman zullen uiteindelijk feedback bieden op de app. Ook zullen zij met klachten komen als de applicatie niet naar behoren werkt.

EVI team (Weinig belang, veel invloed):

Het EVI team is het team dat de mobile operator support applicatie moet gaan ontwikkelen. Deze maken de keuzes welke technologie er uiteindelijk gebruikt gaat worden. Hiervan zijn zij dus van grote invloed op mijn afstudeerstage, want zij kunnen extreem behulpzaam zijn voor het bieden van feedback op keuzes die gemaakt moeten worden en het geven van advies als het gaat over welke technologieën onderzocht kunnen worden.

5.2 Contextanalyse

Voordat er begonnen kon worden aan het afstuderen had Voortman samen met EVI enkele randvoorwaarden opgezet waaraan de app zich aan zou moeten houden.

5.2.1 Randvoorwaarden

Toen de opdracht geboden werd waren er 3 voorwaarden waaraan het framework/app zich aan zou moeten houden, dit zijn de randvoorwaarden. Deze kunnen ook omschreven worden als high-level-requirements, maar zijn gescheiden omdat deze voorwaarden nog belangrijker zijn dan de high-level-requirements.

- De app moet zoveel mogelijk vanuit een codebase werken (hiervoor dus uitzoeken welke het (beste past).
- De gebruiker moet kunnen inloggen met Biometrie op de mobiele applicatie (niet op web).
- De app moet notificaties kunnen sturen naar de mobiele applicatie (niet naar web).

5.2.2 High-Level-Requirements

Samen met Dominic (mijn begeleider vanuit VSM) zijn in de eerste ontmoetings gesprekken over de opdracht en afstudeerstage enkele andere requirements duidelijk geworden:

- Het framework moet makkelijk zijn te begrijpen en gebruiken voor softwareontwikkelaars.
- Voortman wil niet het wiel opnieuw uitvinden als het gaat om app ontwikkeling. Dit houdt in dat ze voor problemen met het gekozen framework naar de community kunnen kijken voor hulp met programmeerproblemen. Hierdoor zijn nieuwe minder gebruikte frameworks geen optie.
- De app moet zowel werken voor web als mobiel (Android en IOS).
- Een CI/CD omgeving moet kunnen worden opgezet voor deze app.

5.3 Requirements voor de app

Door gebruik te maken van de informatie die vergaard is uit het plan van aanpak (bijlage G) en de verschillende interviews die met het EVI team zijn gehouden kunnen de requirements worden opgesteld. De requirements zijn er voor het opstellen van de eisen aan het framework. Ook kunnen deze eisen uiteindelijk gebruikt worden om een proof of concept te maken. Er vonden meerdere interviews plaats om achter de requirements te komen. Van deze interviews zijn er twee gedocumenteerd (zie bijlage A & B), deze twee zijn gedocumenteerd omdat deze van groot belang waren op de requirements, waarbij de rest wel meer informatie ten opzichte van de applicatie biedt

dit niet genoeg was om deze te documenteren (denk hierbij aan dat ik een vraag stel aan een collega en dat we dat gaan discussiëren). Ook zijn er interne bestanden waar veel informatie uitgehaald kon worden. Vooral de functies die klanten nodig hebben en de systeemarchitectuur van de gehele chain stonden hier duidelijk beschreven. Deze bestanden mogen niet gedeelt worden. (Zie bijlage E voor de should and coulds)

In de diagram hieronder zijn de must requirements opgesteld. De functionele en non functionele requirements staan in dezelfde rij omdat dit ruimte bespaard, niet omdat deze gerelateerd zijn.

Must	Functionele requirements	Non Functionele requirements
1	De gebruiker moet kunnen inloggen (Biometrie)(Eye scanner)(Wachtwoord) op mobiel	De app moet zowel werken voor iOS, Android en web.
2	De gebruiker moet een productielijn kunnen selecteren na het inloggen	Er is een CI/CD omgeving mogelijk voor de applicatie, waarin de applicatie voor zowel web als app wordt getest en gereleased.
3	De app moet meerdere talen kunnen ondersteunen.	De app moet zoveel mogelijk vanuit een code-base werken.
4	Een gebruiker kan switchen tussen verschillende lijnen wanneer hij uit meerdere lijnen kan kiezen.	De app moet nog lange ondersteuning van de platforms hebben of het beeld geven dat deze niet binnen enkele jaren zal verdwijnen.
5	Als gebruiker wil ik dat ik op een knop kan drukken om notificaties te ontvangen voor de lijnen die ik geselecteerd heb.	De app moet worden ontwikkeld in een taal/framework waar al veel werk van bestaat. (Voortman wil hiervoor niet opnieuw het wiel uitvinden en riskeren dat het niet meer ondersteund wordt).
6	Als gebruiker wil ik dat ik op een knop kan drukken om notificaties te ontvangen voor de lijnen die ik geselecteerd heb.	De app moet binnen x tellen de verschillende overviews hebben geladen. (Vooral bij het tekenen en berekenen van de dag/week/maand overzichten kan veel rekenwerk bij komen kijken).
7	Als gebruiker wil ik een notificatie ontvangen wanneer een machine op mijn geselecteerde lijn van status veranderd (geen notificatie als naar de status running gaat).	De app moet met AWS SnS kunnen werken.
9	Als gebruiker wil ik aangeven voor hoelang ik notificaties wil ontvangen voor lijnen d.m.v. een tijd te selecteren voordat je op de knop drukt om notificaties te ontvangen.	De app moet gedownload kunnen worden vanuit de Playstore en Appstore.

10	Als gebruiker wil ik een overzicht van machines in een bepaalde lijn kunnen zien.	De mobiele versie van de applicatie moet notificaties kunnen ontvangen over de machinelijnen waarop zij zich hebben gesubscribed..
11	In het lijn overzicht met een teken aangeven of er een machine in de lijn zit die een storing heeft.	De app moet kunnen schalen aan de breedte en hoogte van het beeldscherm.
12	Als gebruiker wil ik mij op meerdere lijnen kunnen subscriben voor notificaties.	De app moet dezelfde functionaliteit kunnen bieden als de webversie ervan (behalve de native only functionaliteit, denk hierbij aan fingerprint, camera, push notifications). De meeste computers hebben geen fingerprint scanner en kunnen niet biometrisch inloggen. Hierbij kunnen we de applicatie dus niet hetzelfde maken.
13	Als gebruiker wil ik notificaties kunnen uitschakelen voor de verschillende lijnen.	
14	De app moet het access token in de lokale storage voor hoe lang deze geldig is.	
15	De app moet het refresh token achter de biometrie van de app zetten. (Dit token wordt gebruikt om de gebruiker te verifiëren bij de backend)	
16	De app zorgt ervoor dat de gebruiker wordt doorgestuurd naar Amazon Cognito (Een webservice van Amazon) wanneer de gebruiker wil inloggen.	
17	Een gebruiker kan inloggen in de app via Cognito.	
18	Een gebruiker kan na het al ingelogd hebben met Cognito een code / fingerprint / faceId gebruiken om de app te kunnen gebruiken voor de duur van de refresh token (30 dagen).	
19	Als de gebruiker geen biometrie heeft ingesteld dan moet deze inloggen via Cognito.	
20	De app zorgt ervoor dat als je de app gesloten hebt dat je opnieuw moet inloggen met of de biometrie/Cognito.	
21	De app moet rekening houden met	

	meerdere typen gebruikers. (Hierbij alleen rekening mee houden, zodat het niet te moeilijk is om dit te implementeren in de toekomst).	
22	Als gebruiker wil ik dat ik een menu heb waarin ik verschillende lijnen kan selecteren om te bekijken.	
23	Als de gebruiker wil ik een overzicht kunnen zien van een machinelijn, met de machines in volgorde van positie in de lijn.	
24	Als gebruiker wil ik in een machine specifiek zien, waarbij ik de historie van statussen in meerdere grafieken en diagrammen te zien krijg.	
25	Als gebruiker wil ik een plaatje, de naam, de huidige status, de duur van de status en de kleurcode van de machine zien bij de specifieke machine view.	

5.4 Eisen voor het framework

Nu de requirements van de applicatie duidelijk zijn moeten er eisen worden opgesteld waar het framework aan getest kan worden. Deze eisen moeten zowel duidelijk als testbaar zijn. De requirements van de app zijn nodig voor het opstellen van de eisen aan het framework. Denk hierbij aan het tekenen van grafieken en diagrammen, dit kan bijvoorbeeld op verschillende manieren worden gedaan als je gebruik maakt van verschillende frameworks.

De eisen zijn opgedeeld in hoofdstukken waarbij er later een score gegeven kan worden per onderdeel. Het hoofdstuk algemeen is toegevoegd voor mensen die het onderzoek niet helemaal willen doorlezen. In dit hoofdstuk staan kort de plus en minpunten genoemd.

5.4.1 Platforms van de app

1. Het framework is geschikt om een volwaardige oplossing te bieden aan gebruikers op iOS en Android.
2. Het framework is geschikt om native apps te combineren met web.
3. De app is geschikt voor de Appstore en de Playstore.

5.4.2 Mogelijk om met Amazon services te koppelen

4. Het framework moet samen kunnen werken met AWS SnS.
5. Het framework moet samen kunnen werken met AWS Cognito.

5.4.3 De maturiteit van het framework

6. Er moet zekerheid zijn dat het framework nog gebruikt wordt in de komende jaren.
7. Het framework moet mature zijn zodat er veel gebruik gemaakt kan worden van al bestaande tools/toepassingen/plug-ins/tutorials/etc...
8. De programmeertaal van het framework is gemakkelijk te leren.
9. Het framework moet gebruik kunnen maken van native functionaliteit (fingerprint scanner, push notifications en eventueel de face/eye scanner) van de mobiele telefoon.
10. Het framework biedt mogelijkheden voor vertaal opties in de app. (Plugins/Modules/etc..)
11. Het framework wordt regelmatig geüpdate/gefixt als er een beveiligingslek is gevonden. (Major bugs worden snel opgelost)

5.4.4 Vermogen van het framework

12. Het framework moet vanuit een api call binnen 3 seconden de data omzetten naar een dashboard.
13. Het framework kan diagrammen en grafieken tekenen om data te visualiseren.-
14. Het framework kan beveiligd data in de lokale opslag zetten. (Encryptie en Decrypten)-
15. Het framework biedt een mogelijkheid tot het versimpelen van de routing tussen pagina's. (Heeft modules/plugins die hierbij helpen)
16. De aanpasbaarheid van het framework. (Of de code makkelijk overgeschreven kan worden naar een ander framework/taal). (Denk hierbij aan Javascript naar Typescript)
17. Ondersteuning van het framework. (Welke telefoons en operating systems frameworks ondersteunen om te ontwikkelen en te deployen)
18. Aansluiting op andere frameworks/technologieën.

5.4.5 Devops van het framework:

19. Test mogelijkheden van het framework (Biedt het framework opties voor automatisch testen, het testen versimpelen, modules bieden voor testen, plugins , etc..).
20. Automatisering van het framework (Genereert het framework al veel boilerplate of moet je zelf alles vanaf de grond typen).
21. Devops mogelijkheden van het framework. (Biedt het framework mogelijkheden om automatische CI/CD en automatisch testen op te zetten en te runnen).

De volgende eisen zijn geen eisen aan de applicatie, maar zijn gemaakt om in het onderzoek een korte samenvatting te geven over de frameworks. Deze twee eisen komen niet terug in dit verslag.

5.4.6 Algemeen

22. Algemene voordelen van het framework.
23. Grote nadelen van het framework.

6. Onderzoek

Dit hoofdstuk behandelt het onderzoek dat is gedaan naar de gekozen frameworks. Eerst wordt er uitgelegd welke frameworks gekozen zijn voor dit onderzoek en waarom deze gekozen zijn. Daarna worden de bevindingen van de verschillende frameworks besproken. Tot slot worden de twee beste frameworks gekozen (beste frameworks uit het onderzoek), waar voor allebei een proof of concept wordt gemaakt, die de eisen testen aan een ontwikkelde applicatie, hierbij worden dus twee applicaties zelf gebouwd om deze eisen te testen. (Dit onderzoek komt uit bijlage C)

6.1 Frameworks

6.1.1 React Native

React native is een open source UI software framework ontwikkeld door Facebook om de trage mobiele versie te verbeteren, deze werd namelijk geprogrammeerd met HTML5. De eerste versie van Facebook op de mobiel was hierdoor instabiel en langzaam. Hier moest een oplossing voor komen. Een medewerker van Facebook Jordan Walke heeft een manier gevonden om UI elementen te renderen vanuit een Javascript thread. Dit werd de basis voor het React framework. Om deze technologie te verbeteren hebben ze een interne Hackathon gehouden, deze was specifiek voor het bouwen van native apps. Na maanden van ontwikkelen bracht Facebook de eerste versie van het React Native framework uit. Sindsdien is het een van de meest gebruikte technologieën om mobiele applicaties mee te schrijven. Ook is dit de taal die ik het meeste ken vanwege mijn React ervaring.

Door de populariteit en herkenbaarheid van het framework is React Native gekozen.

6.1.2 Ionic (Angular)

Ionic Angular is een opensource mobiele toolkit voor het bouwen van hoge kwaliteit, cross platform native en web ervaringen. Ook worden deze gemaakt door een enkele codebase. Ionic laat je je favoriete Javascript Framework kiezen om mee te werken, denk hierbij aan React, Angular, Vue en standaard Javascript. Het verschil met bijvoorbeeld React Native en Flutter is dat Ionic web-first is, dit betekent dat ze van native functies gebruik maken door middel van web technologie zonder het aanraken van platform UI. React Native of Flutter praten wel met de native API.

Persoonlijk heb ik nog niet heel veel ervaring met Ionic, maar genoeg om conclusies te kunnen maken over het framework. Ook gebruikt Voortman Angular voor hun eigen website. Hierdoor zijn er al developers bekend met Angular waardoor Ionic een aantrekkelijke keuze wordt. Door deze redenen wordt Ionic Angular onderzocht.

6.1.3 Flutter

Flutter is een portable UI toolkit. Dit zijn dure woorden voor dat de UI die flutter biedt voor meerdere platforms werkt. Ook is Flutter open source en gratis te gebruiken. Het wordt ontwikkeld door een team van Google, hier wordt flutter intern gebruikt voor applicaties zoals google pay en google ads. Het wordt dus door Google ontwikkeld. Google is een van de grootste Techbedrijven van het moment als het gaat over web/app development. Dit komt ook omdat zij zelf de Android OS onderhouden en updaten. Hierdoor heeft Google veel kennis over het ontwikkelen van applicaties.

Flutter kan gebruikt worden voor het maken van: websites, mobiele apps, desktop apps en zelfs embedded apps voor applicaties op bijvoorbeeld auto beeldschermen.

Flutter runt de app door gebruik te maken van hun eigen “high-performance rendering engine om widgets te tekenen”. Dit is anders dan andere frameworks die gebruik maken van web technologie. De taal waarin Flutter wordt geprogrammeerd is Dart.

Aan het begin van het project had ik geen ervaring met Flutter. Ik had er alleen nog maar van gehoord. Ook vanuit Voortman waren ze niet zo voor Flutter, want dit zou te nieuw zijn.

Ik heb toch de keuze gemaakt om Flutter mee te nemen in het onderzoek, omdat veel mensen van de opleiding gebruik maakten van Flutter en dat Flutter toch wel groot aan het worden is als het gaat over hybrid app development.

6.1.4 Progressive Web Apps

Progressive Web Apps is waarschijnlijk de toekomst van de webapp wereld. Dit is een website die je kan downloaden op de telefoon via de website. Vanuit deze app kan je dan gewoon alles doen wat een normale app ook zou kunnen doen. Vooral voor Android zijn deze mogelijkheden enorm. Apple doet hier heel moeilijk over. Zij willen niet dat er zomaar apps op mensen hun telefoon komen via andere sources dan hun eigen appstore (Anders kunnen ze geen geld verdienen aan de app en hebben ze geen controle meer over apps). Hierdoor zijn de meeste iOS functies beperkt of geblokkeerd. Dit houdt in dat er geen push notificaties (nog), geo-fencing, etc. zijn.

PWA’s bestaan eigenlijk uit frameworks als Ionic/React/Vue, deze zijn uitgebreid om met mobiele “websites” mobiele apps te kunnen maken. Natuurlijk kan een website gedisplayed worden op een mobiele telefoon, maar deze zou niet als app aanvoelen. Hier houden PWA dus rekening mee, zodat je gebruik kan maken van web met het gevoel van native.

PWA’s zijn interessant om te onderzoeken. Omdat de gedownloade apps altijd de laatste update hebben zonder te hoeven updaten vanuit de appstores, ook omdat er zoveel webtechnologieën gebruikt kunnen worden om PWA’s te kunnen maken. Ook wou Dominic (mijn begeleider) deze technologie onderzocht hebben.

6.1.5 Native (iOS / Android)

iOS en Android worden tegelijkertijd benaderd, omdat ze beide hetzelfde kunnen en doen. Het enige verschil met de vorige frameworks is dat deze talen alleen voor hun eigen operating systeem werken.

Android Java/Kotlin werd gemaakt om apps te maken voor Android devices. Het eerste device dat werkte op de Android OS werd rond september 2008 gereleased. Hiermee konden developers apps ontwikkelen voor deze telefoon met het gebruik van Java. Met tegenwoordig een wereldwijd aandeel van 71% van de mobiele operating systems is Android de grootste in het veld. iOS volgt Android met 28% van het aandeel. In meer eerste wereld landen ligt dit aandeel meer 50/50, maar door landen zoals India wordt Android enorm geholpen. iOS wordt tegenwoordig geprogrammeerd met Swift (vroeger met objective C), dit is een objectgeoriënteerde taal voor het ontwikkelen van applicaties voor iOS en OS X.

Android en iOS geven de beste performance en hebben de meeste native functionaliteit van alle frameworks. Maar deze performance werkt dus alleen op een eigen operating systeem, waardoor je

drie apps zou moeten maken i.p.v. een. Dit is dus een conflict met een van de randvoorwaarden. Toch vond het team native nuttig om te onderzoeken voor de ervaring en vergelijking met de frameworks.

6.2 Vooronderzoek

Voordat er onderzoek naar Flutter gedaan kon worden heb ik besloten om eerst ervaring op te doen met Flutter. Om ervaring op te doen heb ik een basisapplicatie ontwikkeld die gebruik maakt van fingerprint, opslag en request naar backends, ook kwam hierbij styling kijken. Deze ervaring is zeer waardevol, want anders kan je niet goed oordelen over deze frameworks.

Voor React Native en Ionic heb ik voor mijn specialisatie een applicatie moeten bouwen. In deze specialisatie moesten wij ook onderzoek doen naar frameworks. Hierdoor heb ik kennis en ervaring over deze frameworks. In het onderzoek zijn ook andere technologieën onderzocht, waaronder Xamarin en Nativescript. Ook PWA's kwamen hierbij kijken. Daarom weet ik het meeste over deze drie frameworks (React Native, Ionic en PWA's).

Native apps heb ik ook al gemaakt in mijn schoolperiode. Dit waren vooral basisapplicaties, maar ik vond dit een leuke programmeertaal en heb thuis ook nog wat appjes in elkaar gezet. Deze zijn alleen niet meer noemenswaardig vergeleken met applicaties die ik nu maak. Dus moest ik mijn native kennis wel weer opkrikken door enkele tutorials te kijken voordat ik de onderzoeksresultaten kon beoordelen.

Toen daarna het onderzoek begon heb ik eerst meerdere video's en onderzoeken bekeken die de verschillende frameworks beschrijven en vergelijken [3, 4, 5, 6, 7]. Dit is gedaan om de manier waarop deze informatiebronnen hun informatie vandaan halen en te achterhalen hoe zij de frameworks vergelijken met elkaar.

6.3 Onderzoek

In het onderzoek behandelen we de eisen die aan het framework zijn gesteld. Deze zijn opgedeeld in hoofdstukken om het wat tastbaarder te maken. De resultaten van de hoofdstukken bespreken we in dit hoofdstuk. Voor uitgebreider onderzoek zie bijlage A.

6.3.1 Platforms van de app

Om dit goed te beoordelen moet er gekeken worden of er volwaardige apps zijn gemaakt voor zowel native als web met het framework. Dit doe we door de gemaakt apps (apps die gemaakt zijn met het framework te bekijken) te bekijken en beoordelen of dit hoogwaardige apps zijn. Ook speelt de bekendheid van deze apps een rol.

Voor native iOS en Android apps zijn er veel apps om naar te kijken. De meeste professionele apps zijn gemaakt met iOS en Android, dit komt doordat deze "frameworks" zijn gemaakt voor de OS. Hierdoor krijgt de gebruiker de beste UI/UX ervaring. Alleen voor native moeten er meerdere applicaties gemaakt om alle platforms die Voortman wil benaderen te kunnen realiseren (aparte iOS

en Android app, ook een losstaande website). Hierdoor is gebruik maken van een native taal niet optimaal.

Ionic apps werken zowel op iOS, Android en web. Alleen zijn er nog niet veel grote bedrijven die Ionic gebruiken om een applicatie op deze platforms te realiseren. Dit komt deels door de populariteit van het framework, ionic is minder populair dan de andere frameworks (behalve PWA).

React Native, Flutter en PWA's werken op de benodigde platforms en ook nog op Windows en macOs. Hierdoor kunnen deze apps worden ingezet op meerdere systemen. PWA's kunnen niet worden geïnstalleerd via de appstores, flutter en react native wel.

React Native en Flutter bieden een nog groter aanbod aan platforms die je kan targeten voor je app. Alleen kon ik voor platforms als VR (react native)[17], Google Fuchsia (Google Fuschia is nog in een vroeg stadium, maar Google wil graag Flutter apps beschikbaar maken voor dit nieuwe operating system) en embedded systemen (Flutter) nog geen goede voorbeelden van applicaties vinden. Hierbij moet er geloofd worden wat er op het internet staat en wat de developers van de platforms hierover zeggen. Deze platforms bieden niks extra's voor Voortman, maar zijn wel een leuke bonus als de platforms groter zouden worden.

React Native en Flutter zijn de beste frameworks voor het benaderen van de meeste platforms. Ook de kwaliteit van de applicaties leidt hier niet onder, denk aan facebook (React Native) , google pay en eBay Motors (Flutter). Ook wordt dit deel hoger beoordeeld dan de rest, omdat Voortman het belangrijk vindt dat het moet werken op meerdere platforms (mobiel en web) vanuit dezelfde codebase.

6.3.2 Mogelijkheden om met Amazon services te koppelen

Alle platforms bieden mogelijkheden om een koppeling te leggen met AWS (Amazon web services), hierdoor zijn ze allemaal geschikt, hoewel de frameworks die vooral webtechnologieën gebruiken voorliggen, want deze koppelen wat makkelijker met deze services. Dit komt omdat AWS een webservice is en zelf.

Flutter is redelijk nieuw, hierdoor is de koppeling met AWS moeilijker op te stellen vergeleken met de andere frameworks. Dit kan zorgen voor vertraging in ontwikkeling, maar het is nog steeds mogelijk om Flutter en AWS te koppelen.

PWA's zijn gebaseerd op web technologieën, waardoor ze snel en makkelijk kunnen koppelen met AWS, alleen kunnen PWA's slecht notificaties ontvangen. Dit komt omdat de native functionaliteit niet opengesteld worden door de besturingssystemen van de telefoons (iOS, Android). Omdat PWA's geen notificaties kunnen ontvangen op iOS zal er een groot deel van de functionaliteit van de applicatie wegvallen voor PWA's. Hierdoor kan de Voortman app niet gerealiseerd worden door middel van een PWA framework. Deze is iets minder belangrijk, want Voortman gebruikt maar een service van AWS zeker, dat is Simple Notification Service (SNS). Dit programma regelt de push notifications richting mobiele devices. Er is nog geen zekerheid dat AWS Cognito (soort inlogsysteem) gebruikt gaat worden, ze zijn nog met alternatieve oplossingen bezig.

6.3.3 Maturiteit van het framework

De meest mature frameworks om mobile first te programmeren zijn de native iOS en Android programmeer “frameworks”. Deze hebben de meeste tooling tot hun beschikking, Ook zijn dit de oudste “frameworks” die gaan over native app development. De ontwikkeltools die deze frameworks bieden helpen ontwikkelaars professionele apps op de markt te zetten. Het probleem met native development is dat de talen alleen niet op elkaar lijken. Hierdoor moet je meerdere teams inhuren om dezelfde applicatie te bouwen. Wel zijn de talen makkelijk om te leren en zijn er veel mensen die deze talen al kunnen.

Flutter scoort in deze categorie vooral laag op zekerheid en leeftijd. De zekerheid komt omdat Google wel vaker projecten start er veel tijd in stop en daarna stopt met het ontwikkelen van dit project (Denk aan Google Hangouts). Wel is het zo dat Google bezig is met Fuchsia (een operating systeem, i.p.v. windows / mac) en hierop runnen ook Flutter apps. Ook intern is Google veel bezig met Flutter. Doordat Google Flutter veel gebruikt en Flutter enorm populair is onder developers gaan de meeste mensen er van uit dat Flutter hier is om te blijven. Door Flutters populariteit haalt Flutter veel grond in als het aankomt op libraries en tools. Hierdoor lijkt het frameworks ouder dan dat het is, want de community en Google zijn druk met nieuwe functionaliteit en functies. Helaas loopt Flutter nog wel achter vergeleken met de andere frameworks (behalve vergeleken met PWA’s).

Flutter is ook een makkelijke taal om te leren als je gewend bent om objectgeorienteerd te programmeren. Voordat ik begon aan het Flutter onderzoek heb ik een basisapplicatie gemaakt, ik begreep Dart vrijwel direct en de Flutter objecten waren bijna click and drag. Het gedrag van de objecten was makkelijk om te programmeren. Ik ben zeer tevreden over Flutter en ga dit framework zeker vaker gebruiken.

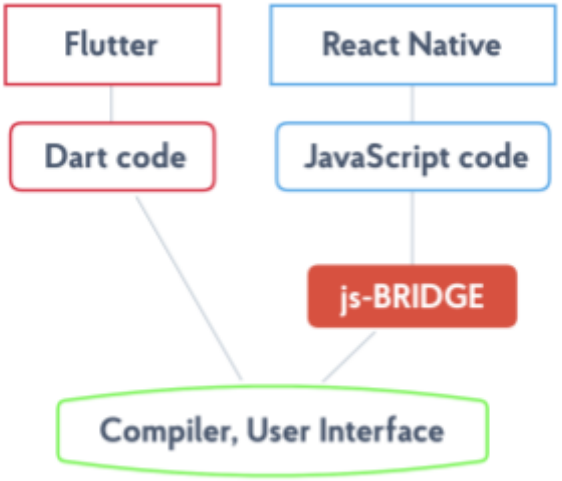
Ionic, React Native en PWA zijn allemaal zeer mature. Dit komt omdat ze zijn gebaseerd op webtechnologie en webtechnologie bestaat al langer dan native. Wel zijn Ionic en React Native frameworks gebouwd om de connectie met native te maken, dit heeft PWA niet. Alle drie de frameworks worden geprogrammeerd met een soort Javascript, HTML en CSS. (Hierop zijn veel varianten omdat de technologieën zo lang bestaan. Denk hierbij aan SCSS i.p.v. CSS of TypeScript i.p.v. Javascript).

Het voordeel dat React Native over Ionic en PWA’s heeft is dat React Native het populairste en meest gebruikte framework is. [18] Hierdoor zijn er meer mensen die verstand van React Native hebben en hierbij dus libraries en tools maken. Ook kunnen ontwikkelaars die React kunnen als taal overspringen naar React Native zonder al te veel moeite, want de talen lijken sprekend op elkaar.

Maturiteit in de zin van compleetheid is wel belangrijk voor Voortman, ze willen namelijk niet opnieuw het wiel uitvinden. Maar de meeste onderzochte frameworks (behalve PWA) houden zich hier goed aan. Maar op het gebied van maturiteit winnen React Native en Native iOS/Android het. Deze frameworks en tools bestaan al zo’n lange tijd dat je veel informatie, voorbeelden, tools en libraries tot je beschikking hebt vergeleken met minder gebruikte frameworks als Ionic en Flutter. Deze combinatie van extra’s zorgt voor een soepelere en snellere ontwikkelervaring.

6.3.4 Vermogen van het framework

Alle platforms bieden mogelijkheden voor het tekenen en berekenen van diagrammen en grafieken, ook kunnen alle frameworks werken met encryptie en decryptie. Flutter en Native (iOS/Android) kunnen deze berekeningen het snelste omzetten naar UI components omdat de code niet via een bridge gaat, maar direct met de eigen engine gerenderd worden. (connectie tussen de taal en de mobiel die het omzet naar native code, zie figuur 5)



Figuur 5 Javascript bridge vs no bridge

React Native, Ionic gebruiken allebei zo'n bridge. Hierdoor is de performance minder, dit zal bijna niet merkbaar zijn voor de meeste applicaties. Zodra vermogen een probleem wordt kan je de applicatie beter in Native code schrijven, want deze biedt de snelste performance voor mobiel.

PWA's runnen in een webbrowser. Daarom is een brug van javascript naar native niet nodig. Hierdoor zijn PWA's erg snel, maar gaat de UI/UX er aan ten onder, want tot nu toe worden er alleen maar web objecten gerenderd.

Dus kwa vermogen van het framework zijn Flutter en Native (iOS/Android) het beste. Dit komt vooral door de snelheid die zij bieden, deze snelheid is het grootste verschil met React Native op dit gebied. De snelheid van de apps gaat waarschijnlijk in het begin weinig uitmaken voor de Mobile Operator Support applicatie, want deze begint als een lichte applicatie zonder animaties. Wel wordt er veel data verwerkt om grafieken en diagrammen te tekenen, dit kan dus wel ten koste gaan van de snelheid. In de toekomst kan de applicatie zwaarder worden en maakt de snelheid meer uit.

6.3.5 Devops van het framework

Devops was een nieuw onderwerp voor mij, hierdoor ben ik geen expert in het opzetten/onderzoeken van devops.

Voor devops kunnen alle frameworks werken met bijvoorbeeld Codemagic, dit is een website/tool die het verwerken van devops omgevingen automatiseert. Hierbij hoef je weinig werk meer te verrichten en kan het ook gratis gebruikt worden. Ook GitLab CI/CD of GitHub Actions kunnen gebruikt worden voor het opzetten van een CI/CD omgeving. Voor de meeste frameworks waren er veel tutorials die lieten zien hoe een CI/CD omgeving zal werken. Voor een framework zoals een PWA hoeft een CI/CD niet heel uitgebreid te zijn. Deze moet bijvoorbeeld alleen testen runnen en code pushen naar een live website. Vergeleken wat een Flutter of React Native moet doen die naar vele platforms de code moeten sturen. Deze moeten een AAB (Android App Bundle), een IPA (iOS App store package) en een website genereren. Daarna moeten deze applicaties naar de juiste platforms gestuurd en geconfigureerd worden. Om uit deze categorie een beste kandidaat te halen heb ik ook gekeken naar de testmogelijkheden en automatisering van code. Hierbij scoren de frameworks

gemiddeld en de native talen wat hoger. Dit komt vooral doordat deze talen langer bestaan en er meer voorbeelden zijn van tools en testomgevingen.

Hierbij is Native (iOS/Android) de beste optie voor devops vanwege de automatisatie van de code en tests. De rest van de frameworks komen hier vlak achterna, maar deze zijn wat ingewikkelder om op te zetten, want je moet meer bestanden genereren en testen. Want de website is niet hetzelfde als de mobiele applicatie bij de frameworks. (Denk aan native only functionaliteit)

6.4 Resultaat

Onderdeel	**Framework→ Haalbare punten ↓**	**React Native**	**Ionic**	**Flutter**	**PWA**	**Native**
Platforms	15	13	10	15	10	1
Koppeling AWS	5	5	5	4	4	5
Maturiteit	10	8	8	5	3	10
Vermogen	10	6	5	9	3	10
Devops	10	8	8	8	7	10
Totaal	50	40	33	41	27	36

Uit het onderzoek is komen rollen dat Flutter en React Native de beste kandidaten zijn als het op de eisen aan komt, ook is er rekening gehouden met de randvoorwaarden van het project. Hierdoor lijkt Native(iOS/Android) hoog te scoren, maar het gaat niet gebruikt worden omdat er vanuit een codebase gewerkt moet kunnen worden. Wel is Ionic nog steeds een goede optie omdat het ontwikkeld kan worden met Angular. Angular wordt al gebruikt door Voortman. Ook wordt er al een mvp ontwikkeld in Ionic. Hierbij is het dus niet meer interessant om hiervan ook nog een proof of concept te maken, want deze is er al deels. Daarom zijn Flutter en React Native de beste keuze om verder te onderzoeken. Dat React Native en Flutter de beste talen zijn om een hybride applicatie te maken is geen verrassing. Veel online forms en onderzoeken komen ook uit op dit resultaat. Flutter en React Native zijn momenteel het grootst. [8, 9, 10, 11]

7. Functioneel ontwerp

In dit hoofdstuk wordt het functionele ontwerp van de proof of concept opgesteld. Ook kunnen de MUST requirements en framework eisen hieraan gekoppeld worden. Deze MUST requirements zijn alleen niet heel belangrijk voor de opdracht, want de proof of concept gaat over hoe het framework zich kan houden aan de opgestelde eisen. Niet alle eisen kunnen worden getest met een gebouwde applicatie, de andere eisen worden theoretisch onderzocht en of ondervonden. (Denk hierbij aan de koppeling met AWS, er is geen AWS account tot mijn beschikking waardoor ik deze niet kan testen). Je kan bij Amazon namelijk geen student account aanmaken om deze tools uit te testen, want het studentenprogramma van Amazon is een beetje een vergeten achterstallige applicatie. [19]

Schermen	**Eisen**	**Functionele Requirements**
Login pagina	7, 8, 9, 12, 14, 15	1, 12, 13, (14 & 15), 16, (17), 18
Lijnen overzichtspagina (meerdere lijnen van een bedrijf)	7, 8, 9, 10, 12, 14, 15	2, 3, 4, 5, (6), 9, (10 & 11), 20
Lijn overzichtspagina (een lijn met hierin machines)	7, 8, 12, 13, 15	8, 9, 21, 23
Machine overzichtspagina	7, 8, 12, 13, 15	8, 22, 23

Onderdelen	**Eisen**	**Functionele Requirements**	**Non-Functionele Requirements**
CI/CD omgeving	(3), 19, 20 , 21	1	1, 2, (9)

7.1 Login Pagina

De loginpagina is de pagina waar de gebruiker als eerste terecht komt als deze de app of website opent. (Figuur 6) De loginpagina test het framework op ontwerp, styling, http-calls, biometrie, lokale opslag en routing. Ook de leerbaarheid van het framework wordt hier getest.

Deze pagina moet vertaalt kunnen worden, waarbij het framework rekening moet houden bij de taal waarin het device is ingesteld. Dit wordt getest door meerdere talen toe te voegen aan het systeem. De gekozen talen zijn Engels, Nederlands en Russisch geworden.

Figuur 6 Login Pagina

7.2 Lijnen overzichtspagina

Op de Lijnenoverzicht pagina staan alle lijnen van een specifiek bedrijf die aan een account zijn gekoppeld. Deze Lijnen worden getoond in een scrollbaar lijstje. Deze lijst bevat onder andere plaatjes van de machines die in deze lijn staan. Ook staat de status van de lijn in het algemeen aangegeven door de kleur om de lijn kaartjes. (Figuur 7) Deze lijnen worden opgehaald door een http-call naar de backend.

Gebruikers kunnen op een van lijnen klikken om naar het lijn overzichtspagina te gaan. Als gebruikers op een specifieke machine klikken in de lijst worden ze doorgestuurd naar de machine overzichtspagina van de geklikte machine.

In het lijnenoverzicht staat ook rechtsboven een optie om van taal te wisselen. Hier krijg je een dropdown menu om van taal te veranderen.

Ook kun je linksonder op het notificatie symbool klikken om het notificatie menu te openen (Figuur 8) deze is hier simpel aangegeven omdat het alleen getest moet worden of je de notificaties aan en uit kan zetten.

Figuur 7 Lijnen Overzicht

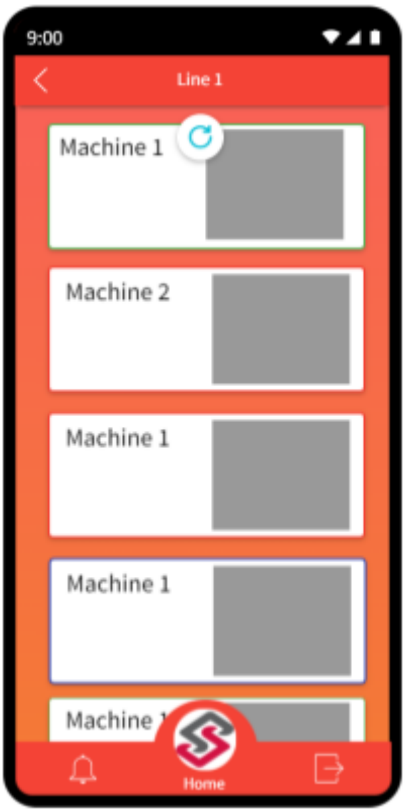
Figuur 8 Notificaties

7.3 Lijn overzichtspagina

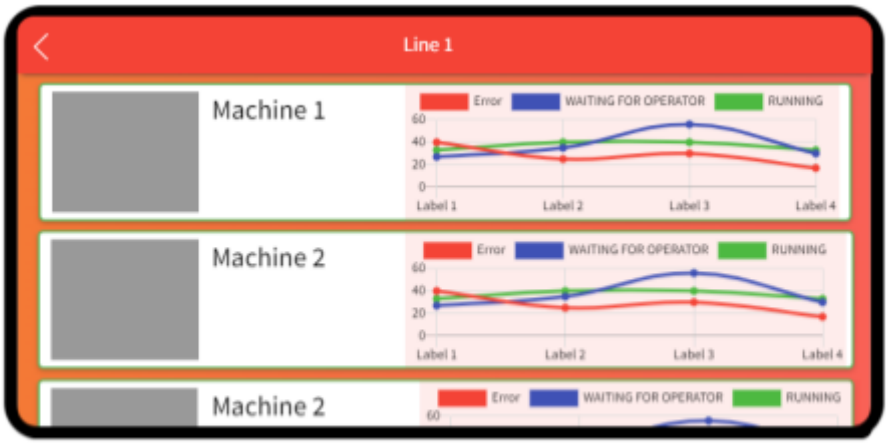
Op de lijn overzichtspagina kunnen gebruikers een lijst met machines zien. In deze lijst staan de huidige statussen van de machine, ook kunnen ze in dit scherm zien: de naam van de machine, naam van de lijn de huidige status de duur van de status en de vorige status. (Figuur 9)

Gebruikers kunnen vervolgens op deze machines klikken om naar het machine overzichtsscherm te gaan.

Als het beeldscherm breed genoeg is kan de website/app een balk laten zien met daarin een geschiedenis balk van statussen van de machines. (Figuur 10)



Figuur 9 Lijn overzicht



Figuur 10 Lijn overzicht Horizontaal

7.4 Machine overzichtspagina

Als gebruikers op de machine overzichtspagina beland zijn krijgen ze hier meerdere diagrammen en grafieken te zien, met hierin informatie over de huidige status en vorige statussen. Deze kunnen zowel worden getoond in bijvoorbeeld duratie van de status of het aantal statussen. (Figuur 11)

Figuur 11 Machine Overzicht

7.5 CI/CD omgeving

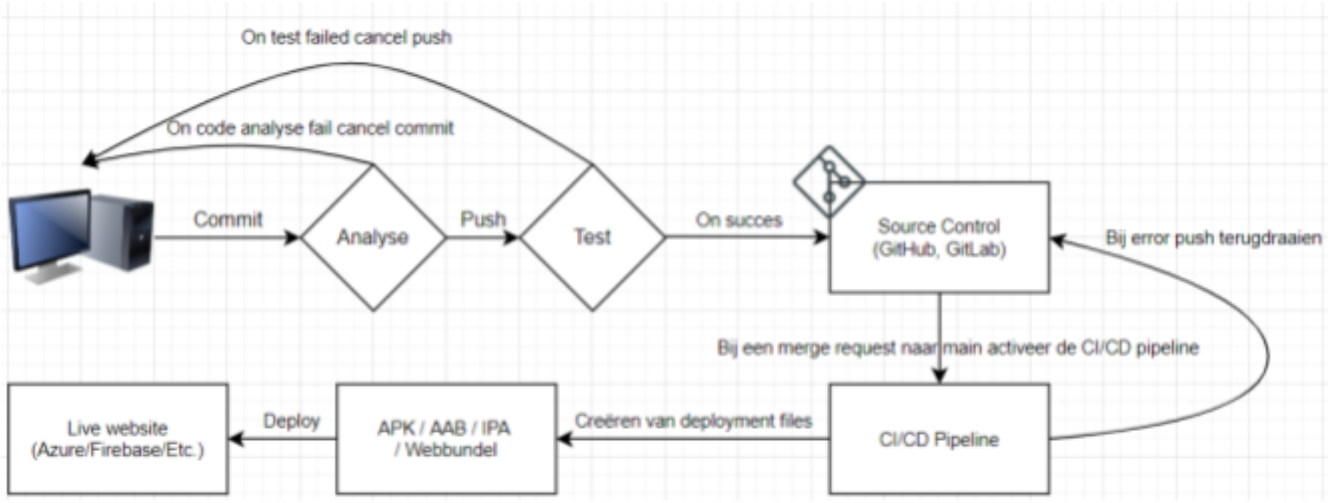
We willen met een CI/CD omgeving het mogelijk maken om de code automatisch te pushen naar de app stores en websites. Deze omgeving ziet er als volgt uit. Een ontwikkelaar commit zijn code, deze code wordt geanalyseerd door een tool / programma, vervolgens als deze goedgekeurd is kan de ontwikkelaar de code pushen naar de branch.

Bij een push naar een branch wordt de code getest. Zodra deze code goed door de test komt wordt de code naar de branch gepusht. Zodra de gebruiker deze branch wil samenvoegen naar de main/master branch wordt er een merge request gedaan.

Tijdens de merge request starten er pipelines die de app testen en builden nadat deze gemerged is. Hierbij wordt gekeken of de gemergde code door de tests heen komt. (Met de tests wordt bedoeld: Unittests, widget tests en intergratietests) Zodra deze tests succesvol zijn afgerond moet de pipeline bezig gaan met de deployment bestanden.

Voor nu genereren we een Webbundel en APK. Dit doen we omdat een IPA genereren geld kost. (100 euro per jaar voor Apple). Deze APK kan voor nu met de hand geupload worden naar de app stores om de chain af te maken. Je pushed namelijk eens in de twee á vier weken updates naar gebruikers vanwege de lange controletijd van de app stores, dit process gebeurt niet vaak genoeg om dit te hoeven te automatiseren.

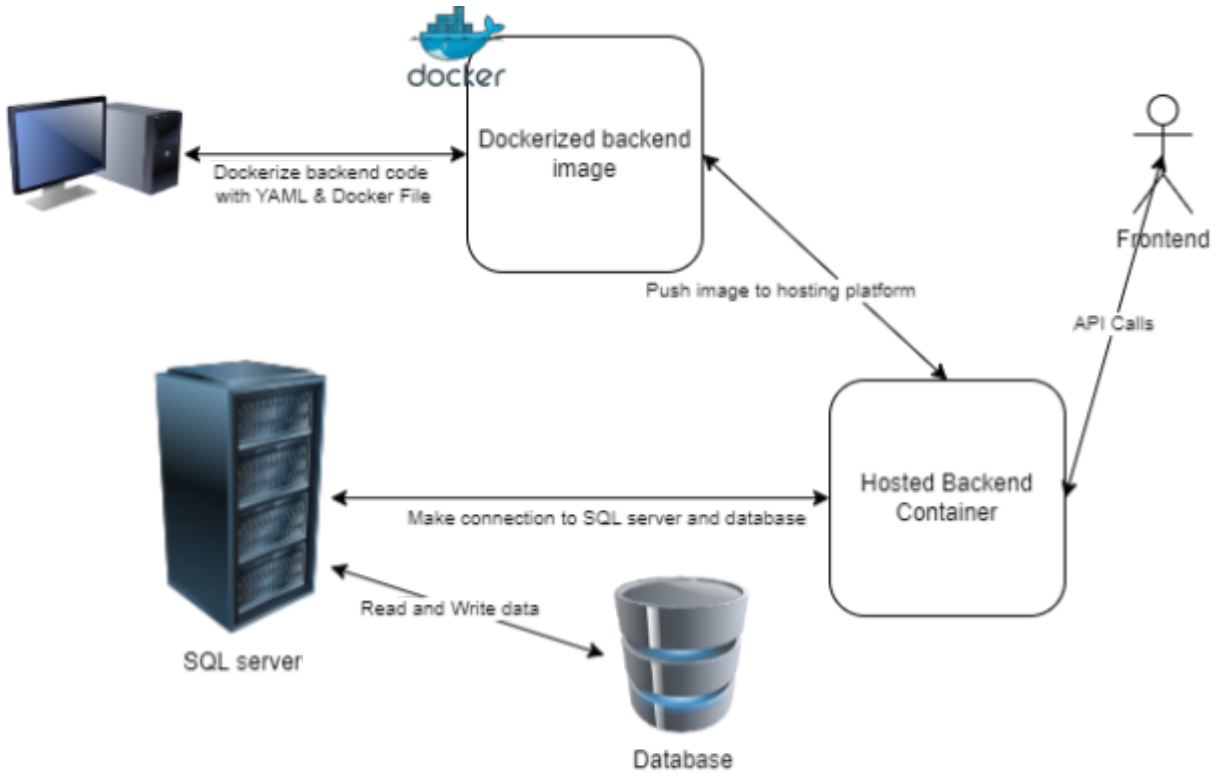
De website kan wel live worden geupdate omdat hier geen verdere controle opzit. De webbundel wordt hier vervolgens naar een hosting platform gestuurd en geconfigureerd. (Azure/Firebase/Etc.)
(Zie figuur 12 voor de CI/CD omgeving frontend)



Figuur 12 CI/CD omgeving

7.6 Backend Omgeving

Omdat het pas duidelijk werd dat halverwege het project ik toch geen backend en data tot mijn beschikking zou hebben moest er toch een backend gemaakt worden. Deze backend is simpel gebleven zodat er meer tijd over was voor het onderzoek en de proof of concepts. Wel is besloten om deze een CI/CD te geven om dit concept wat beter te begrijpen. (Figuur 13 voor de gehele backend structuur)



Figuur 13 Backend structuur

8. Technisch ontwerp

In dit hoofdstuk worden de technische keuzes van de applicatie besproken.

8.1 Frontend techniek

8.1.1 Frameworks

Uit het onderzoek bleek dat Flutter en React Native de beste frameworks zijn om de Voortman applicatie te ontwikkelen, daarom worden van deze frameworks proof of concepts gemaakt om de eisen te testen aan een gebouwde applicatie. Eerst is de Flutter applicatie aan de beurt, omdat er niemand met flutter ervaring op de afdeling zit, zou ik hier de meeste tijd aan kunnen besteden.

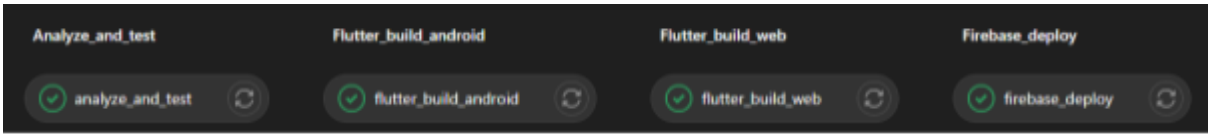
8.1.2 CI / CD

Als er nieuwe code gepushed wordt naar de versie controle (Git), wordt deze code geanalyseerd en getest, vervolgens als de code daarna gemerged wordt naar de main branch moeten er een aantal zaken gebeuren. Deze zaken worden geregeld door de CI/CD omgeving. Voortman gebruikt zelf voor hun website AWS voor CI/CD, maar aangezien deze niet beschikbaar is voor dit project en CI/CD bijna overal hetzelfde doet/werkt hebben wij hier vrije keuze in.

Voor het opzetten van een CI/CD omgeving hebben we drie zaken nodig. Een Git platform (versie controlesysteem), enkele tests om de code aan te testen en een hosting site om de code te kunnen hosten.

8.1.2.1 Versie Controle Systeem

Voor het versie controlesysteem is er gekozen met GitLab, onder andere omdat deze ook gebruikt is in vorige projecten en omdat dit een fijne omgeving is voor het automatiseren van een CI/CD omgeving. Omdat een omgeving als GitLab / GitHub / AWS Code Commit hetzelfde kunnen en doen, vond ik GitLab een fijn programma om te gebruiken. GitLab is gratis om te gebruiken en je krijgt gratis CI/CD minuten voor pipelines, hierdoor hoef je zelf geen machine op te zetten die de stappen volgt. Dit versnelt het CI/CD proces aanzienlijk.



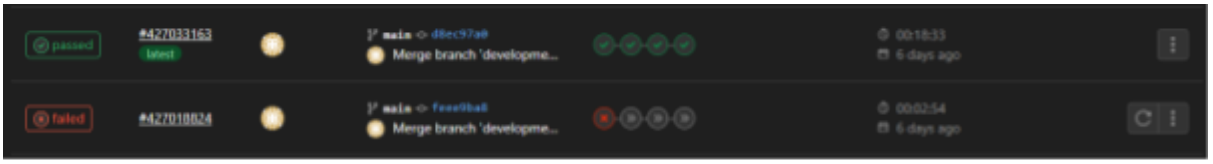
Figuur 23 CI/CD Stappen

De CI/CD is vervolgens zo opgezet in de GitLab (Figuur 23), eerst wordt de code geanalyseerd en getest (dit is om slechte en slordige code op te vangen), vervolgens wordt de code gebuild voor android (checken of de apk correct gebouwd kan worden), daarna wordt de code gebuild voor web (check of de webbuild het doet) en dan wordt deze webbundel gedeployed naar een firebase applicatie (check of de site geupload kan worden naar een hosting platform). Zie bijlage D voor de GitLab yml file, waarin alle stappen beschreven staan in yml formaat.

8.1.2.2 Analyseren & Testing

In deze stap analyseren we de code. Deze vangt af dat we bijvoorbeeld onnodige libraries en dependencies gebruiken in de code. Dit kan ruimte besparen op de uiteindelijke applicatie waardoor deze stap best belangrijk is. Ook vangt dit slordige code af, zoals errors en warnings.

Vervolgens zet deze stap een omgeving op waarin de applicatie getest kan worden. De tests worden dan uitgevoerd (unittests, widget tests en integratietests). Zodra deze tests slagen kan de applicatie verder met de volgende stap. Als de applicatie niet door deze stap heen komt wordt de merge geblokkeerd. (Zie figuur 24)



Figuur 24 GitLab geslaagde pipeline en gefaalde pipeline

8.1.2.3 Code Build Android

De code wordt na het testen gebuild voor Android, in deze stap wordt een apk gegenereerd die na een succesvolle build gebruik kan worden op mobiele android devices. Er is specifiek gekozen om Android wel te builden en iOS niet, omdat voor iOS een Apple dev account nodig was om een IPA te genereren, ook is hier een mac voor nodig die niet tot mijn beschikking is. Hier was geen beschikking voor dus kon dit niet geregeld worden. Wel is zo'n account het enige dat nodig is om ook een iOS build te maken. Nadat de Android build succesvol is afgerond wordt de webbundel gemaakt.

8.1.2.4 Web Build

De web build zorgt ervoor dat de code wordt omgezet naar een webbundel, deze webbundel kan vervolgens worden gedockerized indien nodig. Deze webbuild zet de applicatie om in javascript, html en css bestanden. Deze bestanden kunnen lokaal gedraaid worden met node of online gedraaid worden door een hosting platform als Firebase.

8.1.2.5 Web Deployment to Firebase

Eerder in dit hoofdstuk is verteld over dat Firebase is gekozen als hosting platform, dit hosting platform host gratis de website. Om voor Firebase CI/CD op te zetten moet Firebase wel geconfigureerd zijn in het project. Dit betekend dat je eerst in het project een aantal bestanden moet aanmaken (of je voert een commando uit "firebase init"). Daarna moet je inloggen met je Firebase account om je token te leveren aan de gitlab.yml file. Daarna kan automatisch de webbundel worden gepushed naar de Firebase servers, deze servers regelen automatisch dat de nieuwe versie van de webbundel wordt gebruikt voor de live website.

Zie figuur 25 voor een gefaalde deployment, waardoor de merge naar main nog steeds geblokkeerd wordt.

Figuur 25 Gefaalde deployment naar firebase

8.2 Backend techniek

Omdat de backend in korte tijd gemaakt moest worden en deze niet heel veel functionaliteit nodig had heb ik besloten om het onderzoek/keuze kort te houden.

8.2.1 Frameworks

De keuzes tussen de verschillende backend frameworks was snel gemaakt. Dit werd namelijk .NET.
Dit komt door drie redenen, namelijk:

- Voortman gebruikt zelf de taal C# voor VACAM (de software die de machines aanstuurt). .NET wordt geprogrammeerd in C#. Hierdoor kan ik veel hulp krijgen bij problemen met C#.

- De backend die nog niet beschikbaar is voor mij (niet af) wordt geprogrammeerd in .NET. Hierom is het handig om dezelfde backend te hanteren.

- Ik heb zelf veel ervaring met het opzetten van een backend omgeving in .NET. Voor mijn vorige project moest namelijk een volwaardige backend worden geschreven en dit hebben wij gedaan in het .NET framework.

Dus door de bovenstaande redenen heb ik gekozen voor het .NET framework om de backend in te programmeren. Om dit te programmeren gebruikt Voortman Visual Studio Code en doe ik dit ook, want het is een erg fijn programma en het is net als .NET gemaakt door Microsoft en sluit hierdoor goed aan op het framework.

Het doel van deze backend is om machinelijnen en machines op te slaan, ook accounts worden bijgehouden in de backend, de backend is wel minimaal gehouden omdat er geen tijd gepland was hiervoor. Daarom zijn sommige functies en methodes makkelijker gemaakt dan dat je het in de praktijk daadwerkelijk zou maken.

8.2.2 Backend Hosting

Nadat de backend lokaal gedraaid kon worden, was het voor het testen op mijn eigen device en iOS devices nodig om de backend te hosten, want deze konden geen lokale verbinding maken met de backend en database. Dit kwam doordat bepaalde nodige poorten werden geblokkeerd door het Voortman netwerk. Hierdoor moest de backend gehost worden. Hierdoor zou ik ook makkelijker thuis kunnen werken aan de applicatie.

De keuze moest worden gemaakt tussen Firebase (Google Cloud), Azure en AWS. AWS en Firebase (Google Cloud) zijn zo'n beetje de grootste hosting platforms op het internet. [12]. Azure was naar voren gekomen omdat een collega had gehoord dat studenten een budget krijgen voor het gebruiken van Azure, dit klopt inderdaad.

Hosting platform ->	Azure	Firebase (Google Cloud)	AWS
Kosten	100 Euro budget voor studenten p.m.	Free tier (geen creditcard nodig)	Free tier (wel creditcard nodig)
Ontwikkeld door	Microsoft	Google	Amazon
Compatible met .NET	Ja (Kan docker containers runnen)	Nee (Ondersteunt NodeJS)	Ja (Kan docker containers runnen)
CI/CD mogelijkheden	Veel	Veel	Veel
Mijn ervaring met het hosting platform	Geen	Weinig	Geen

Omdat ik geen vrijwel geen ervaring had met alle drie de hosting tools en ze allemaal goede CI/CD tools leveren is er toch gekozen voor Azure. Deze keuze is vooral gemaakt omdat het van hetzelfde bedrijf is waarin de backend geschreven is (Microsoft), ook is het budget ruim genoeg om een backend op te zetten en later ook een SQL server en database hiermee te koppelen (in het hoofdstuk database techniek hier meer over). Ook het hebben van een credit card was een minpunt, want deze had ik niet en kon ik ook niet aan komen met mijn bank. Rabobank wil dat je minimaal 3 maanden achter elkaar X euro aan loon binnenkrijgt voor een creditcard en omdat mijn bijbaan op een laag pitje is gezet door deze opdracht zit ik niet meer stabiel boven de X euro in de maand.

Het opzetten van een Azure account is geen probleem, je kan de schoolmail gebruiken om een account te creëren, vervolgens vraag je studenten status aan en heeft Saxion of Azure een standaardaccount met een 100 euro limiet geboden. Een limiet is normaal niet mogelijk, maar voor studenten wel.

Om de backend applicatie te laten runnen in Azure had het een docker container nodig. Deze Docker container was mogelijk, omdat de computers waarop gewerkt wordt bij Voortman Windows pro runnen, deze versie maakt het mogelijk om Docker te gebruiken.

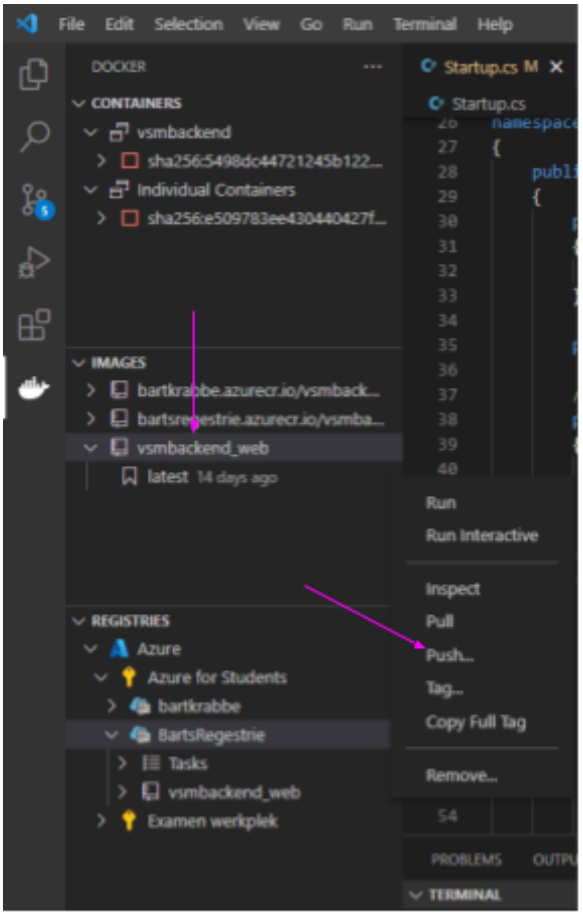
Deze docker container bestond uit twee delen een SQL image met daarin de database en een image met daarin de .NET backend API. Deze twee images werkte lokaal met elkaar, maar toen deze container naar Azure werden gestuurd deden ze het niet. Na in de logs te hebben gekeken bleek het dat de SQL server docker container niet gepushed konden worden naar Azure (logisch achteraf, want anders hoeven mensen niet voor SQL servers te betalen), hierdoor moest ook een SQL server "gekocht", opgezet en geconfigureerd worden.

Dit probleem is opgelost door van het resterende budget een SQL server en database aan te schaffen bij Azure. Daarna moest er nog een deel van de backend en Docker code herschreven worden om rekening te houden met een ander SQL server adres en logica.

8.2.3 CI/CD

In plaats van de code elke keer handmatig te hoeven uploaden kon Azure een CI/CD oplossing bieden. Hiervoor moesten er drie onderdelen worden aangemaakt, namelijk: Een App Service (hierin draait de applicatie uiteindelijk), een Container Registry (een opslag voor docker containers), en een image (de applicatie) om in de docker container te draaien.

Deze zijn uiteindelijk zo opgezet dat als de image wordt geupload naar de container registry vanuit Visual Studio Code. Azure zorgt dat de registry de laatste versie van de image gebruikt en deze levert aan de App Service. Deze App Service krijgt vervolgens een adres van Azure en naar dit adres kunnen API calls naar gestuurd worden en hebben we de backend live staan. (Deze is nu nog live te vinden op: https://bartkrabbe.azurewebsites.net/swagger/index.html, de backend kan traag zijn als hij een tijdje niet gebruikt wordt, maar dit probleem wordt opgelost zodra je hem weer een keer gebruikt. (Figuur 15)



Figuur 14 Visual Studio Code and Docker automatisation

Om deze image gemakkelijk te kunnen uploaden gebruiken we de Visual Studio Code Docker plugin. Deze plugin zorgt ervoor dat als je een docker image bouwd met "*docker compose build*" dat deze image dan getoond wordt in het Docker plugin menu (zie figuur 14).

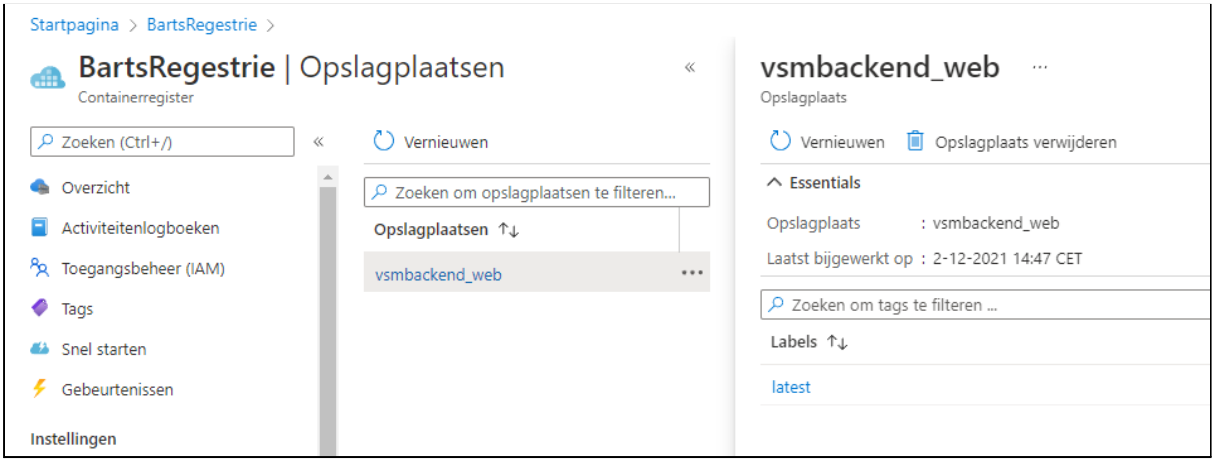
In deze plugin kan je ook een Azure account toevoegen, als je een account het gekoppeld kan je een image pushen door op de rechtermuisknop te drukken op de laatste image, op push te drukken, daarna de juiste container registry te selecteren en op enter te drukken.

Vanuit dit menu zou je de applicatie ook lokaal kunnen laten draaien vanuit een docker container. Dit kan gebruikt worden om snel nieuwe backend functies / calls te testen op de huidige database.

Als je de container pushed naar je account of in mijn geval een container registry, komt hij daarin te staan als de laatste versie van de image. Deze image kan vervolgens gebruikt worden door de App Services van Azure, waarop de backend gehost wordt. Deze App service (Figuur 15) staat zo ingesteld dat zodra de nieuwste versie van de image wordt geupload, de app service opnieuw opstart om zo de nieuwste versie van de backend te draaien. (Zie figuur 16)

Figuur 15 Azure App Service

De backend wordt niet getest in het CI/CD proces, hiervoor is gekozen omdat de backend functionaliteit beperkt bleef, waardoor de calls niet complex genoeg waren om daadwerkelijke tests voor te schrijven. Wel worden fouten zoveel mogelijk afgevangen in de backend, want de juiste status codes moeten wel teruggestuurd worden naar een call. Hierbij check ik al het token, of objecten null zijn en of bepaalde objecten kloppen. Het proces van het controleren van de data was dermate genoeg voor de backend om niet ook nog tests te schrijven.



Figuur 16 Azure Container Registry

8.2.4 Entity Framework

Om de data van de database te kunnen lezen en schrijven, gebruiken developers meestal SQL queries deze queries bevatten bepaalde logica om data te verwerken in de database. Een voorbeeld hiervan zou kunnen zijn :

```
SELECT *
FROM Voortman.Machines
WHERE [Machine_Name]='V807'
```

Deze query's kunnen lang en complex worden als je diepere gelinkte data hebt, of de data aan veel specificaties moet voldoen. Dit kan worden versimpeld door gebruik te maken van Entity Framework. Dit framework dat bovenop .NET wordt gebruikt, zorgt ervoor dat de data in databases net als objecten in C# gedragen, in plaats van te queryen roep je een lijst aan met daarin objecten. Deze objecten kan je vervolgens weer op dit object data oproepen. Het volgende plaatje is een voorbeeld van hoe je met entity framework data uit de database kan halen:

```
public Account GetAccountById(int id)
{
    return _context.Accounts.FirstOrDefault(acc => acc.AccountId == id);
}
```

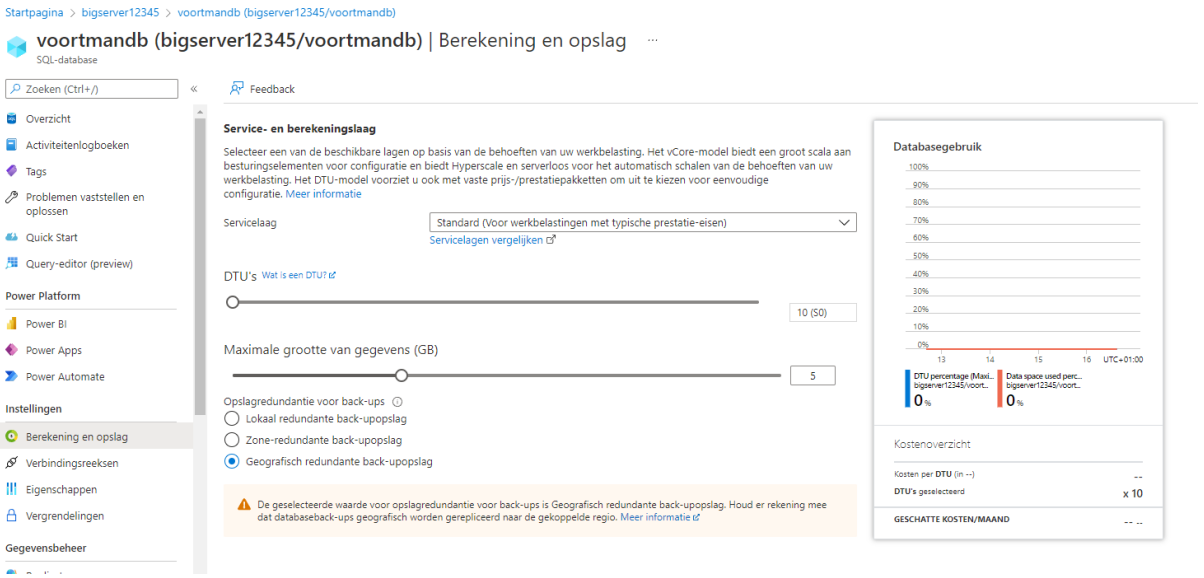
Uiteindelijk doet entity framework onder water zelf SQL query's, maar hier hebben gebruikers geen last van en kunnen ze gemakkelijker omgaan met data uit SQL databases.

8.3 Database techniek

Voor het opzetten van een database is gekozen voor een SQL database, dit komt omdat de data relationeel (zie figuur 20) is en dus SQL de beste keuze is. Ook wordt er gekozen om deze database te hosten in Azure, omdat hier ook de backend draait en om meer ervaring op te doen met het platform.

Voordat de SQL database benaderd kan worden moet er een SQL server worden ingericht, deze viel nog binnen het studentenbudget van Microsoft (15 euro), dus is deze gebruikt. De SQL server is ingesteld op minimaal gebruik, daarom kan de backend af en toe wat langzaam zijn, omdat de SQL server dan in de "sleep mode" gaat. Dit gebeurd als de SQL server een tijdje niet meer gebruikt is. (Figuur 27)

Deze server en database werken naar behoren met entity framework en zijn makkelijk te beheren vanuit het menu. De hele backend draait vanuit Azure en is hierdoor benaderbaar via elk device. De gehele app stack die draait in Azure is als volgt: Een SQL server en database, een container registry met daarin images van de backend, een App Service die gebruik maakt van deze container registry en laatste image. (Zie figuur 17)



Figuur 27 SQL Server instellingen

Figuur 17 Azure Overzicht

8.4 Grafieken en diagrammen tekenen

Voor de Voortman applicatie moesten er twee soorten grafieken getekend worden. Een pie chart (figuur 18) en een event stream bar diagram (figuur 19).

8.4.1 Pie Chart

De pie chart wordt gebruikt om aan te tonen hoeveel procent van de tijd welke status heeft. (Dus 30% is RUNNING, 50% ERROR, etc..) Ook kan deze piechart de tijd die deze statussen hebben vergelijken. (I.p.v. % aantal uur dat de machine die status heeft op deze dag).

Figuur 18 Pie Chart

8.4.1.1 Flutter

Voor Flutter zijn er twee libraries gebruikt om pie charts te tekenen. Eerst is er gebruik gemaakt van de aangeraden package namelijk flutter_charts [13] en later pie_charts [14] gebruikt om de piechart te maken, deze libraries werden het meest gebruikt voor het tekenen van grafieken en diagrammen. De pie_charts plugin was iets beter ingesteld op piecharts vergeleken met flutter_charts die niet zo'n mooie styling had.

8.4.1.2 React Native

Voor de React Native applicatie is react-native-chart-kit [20] gebruikt, deze biedt een mooie en goede oplossing om de charts te tekenen. Deze heeft veel weg van Chart.js (een veelgebruikte librarie voor charts en diagrammen), waardoor de koppeling met react native gemakkelijk gaat.

8.4.2 Event Stream Bar

De event stream bar houdt in dat de statussen van de machine van een dag worden getoond in een horizontale "streep" deze streep bevat verschillende blokken/kleuren om aan te tonen welke status wanneer gebeurd is. (In het voorbeeld van figuur 19: van 14:00 tot ~20:30 was deze machine in error). Door deze grafiek/diagram kunnen leidinggevenden kijken of de machines wel productief draaien en als ze niet productief zijn verhaal kunnen halen bij de operators.

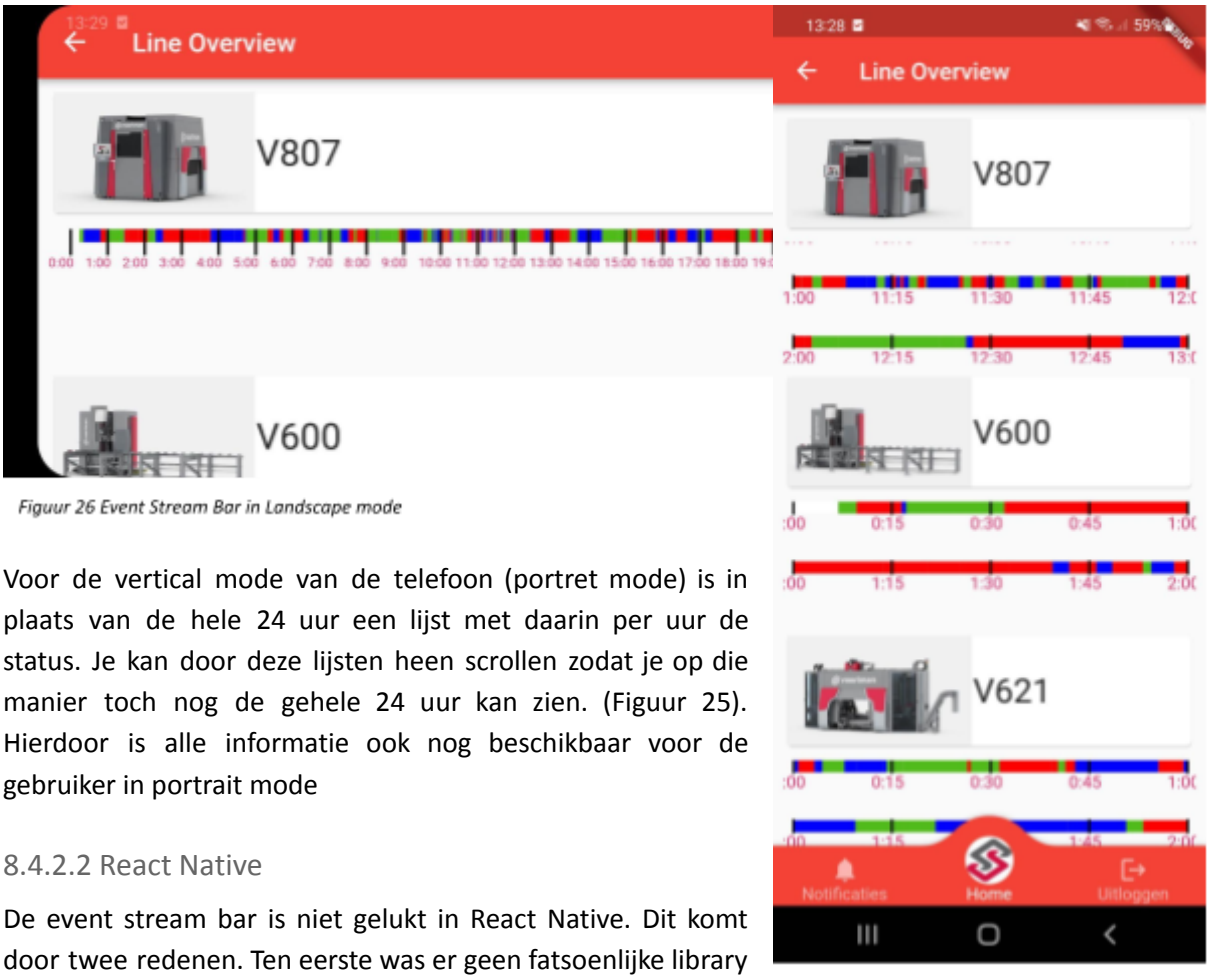
Figuur 19 Event Stream Bar

8.4.2.1 Flutter

De event stream bar met Flutter was helaas niet goed mogelijk met libraries [13][15][16]. Meerdere libraries zijn bekeken, maar deze boden allemaal geen oplossing voor een event stream bar. Dit komt waarschijnlijk door twee zaken:

- Flutter is nog redelijk nieuw, waardoor libraries nog niet compleet zijn.
- De event stream bar is iets unieks, hiervan heeft het team geen voorbeelden kunnen vinden online. Figuur 19 is daarom ook zelf gemaakt.

Uiteindelijk heb ik ervoor gekozen om dit probleem op te lossen door zelf de tijden en blokken op een canvas te tekenen, hiervoor waren wel berekeningen nodig om de ruimte tussen elke status te berekenen. De ruimte tussen de statussen was ook nog een probleem, want een balk van 24 uur zat eigenlijk al te veel data in om deze data goed te visualiseren, waardoor statussen weg vielen door de krappe ruimte. Dit is opgelost door de draaiing van de telefoon te gebruiken om horizontaal meer ruimte te creëren voor de event stream bar (Figuur 24).



Figuur 26 Event Stream Bar in Landscape mode

Figuur 27 Event Stream bar in portrait mode

Voor de vertical mode van de telefoon (portret mode) is in plaats van de hele 24 uur een lijst met daarin per uur de status. Je kan door deze lijsten heen scrollen zodat je op die manier toch nog de gehele 24 uur kan zien. (Figuur 25). Hierdoor is alle informatie ook nog beschikbaar voor de gebruiker in portrait mode

8.4.2.2 React Native

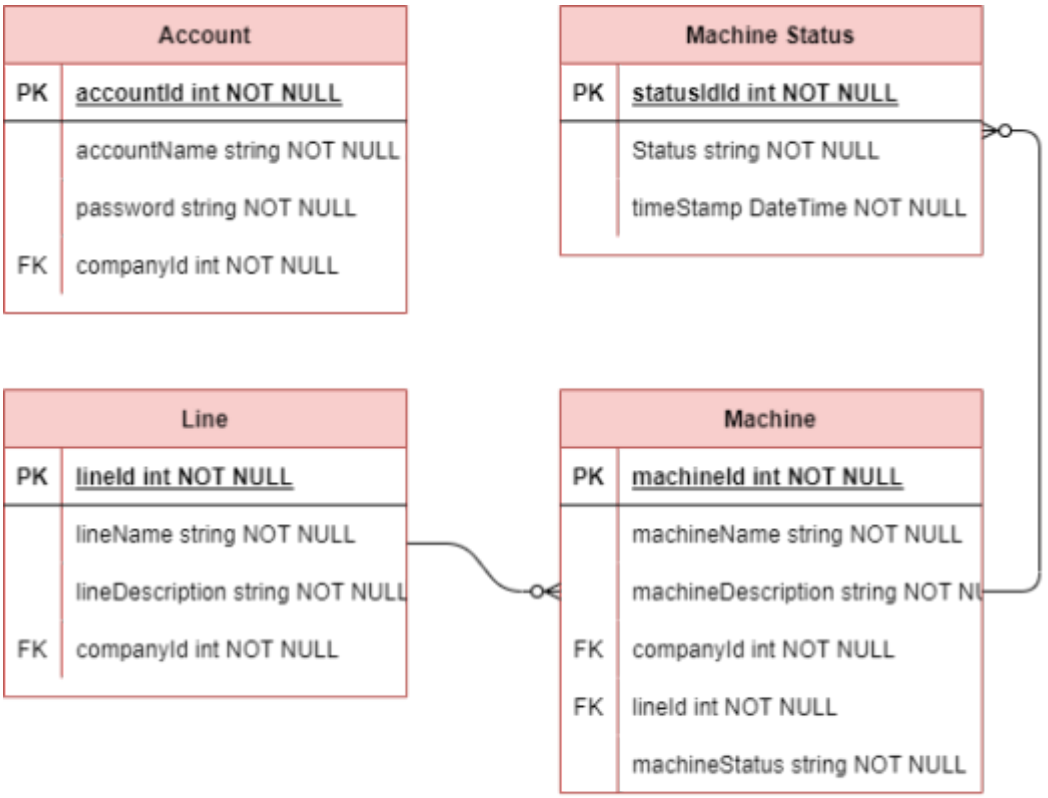
De event stream bar is niet gelukt in React Native. Dit komt door twee redenen. Ten eerste was er geen fatsoenlijke library die mij een oplossing kon bieden voor dit probleem en ten tweede kwam er tijd tekort door erge ziekte in de periode waarin deze functie gemaakt kon worden, waardoor zelfs geen handmatige functie is geïmplementeerd die werkte. Wel zijn er meerdere canvas libraries uitgeprobeerd om dit probleem op te lossen, maar dit is niet werkend gekregen.

8.5 Databasestructuur

Om de frontend te kunnen testen op de eisen hebben we minimaal 4 objecten (figuur 20) nodig in de backend en de SQL database, namelijk:

- Account, een account is nodig om een JWT token te ontvangen bij het inloggen (met gebruikersnaam en wachtwoord). Door dit token kunnen we de gebruiker verifiëren in de backend en de juiste rechten te geven.
- Line, als een gebruiker is aangemeld bij een bepaald bedrijf, krijgt de gebruiker toegang tot de bepaalde machinelijnen van het bedrijf. Een zo'n line bevat de naam van de line, het id van de line, de beschrijving van de line en een lijst met nul of meerdere machines met hetzelfde line id. In de daadwerkelijke uitwerking van deze opdracht zou een line overkomen met een line in het echt.
- Machine, een machine zou in het echt ook "gekoppeld" zijn aan een fysieke machine. Een machine bevat een id, een line id, een company id, een naam, een beschrijving en een lijst met hierin de verschillende statussen die de machine beleeft heeft.

- Machine Status, een machine status bevat een id, een status (in string format) en een timestamp. Een status geeft aan wat er met de machine aan de hand was op een bepaald tijdstip.



Figuur 20 Databasestructuur

8.6 Backend Specificatie

Deze sectie behandelt de backend specificatie van de Voortman backend. Deze backend wordt gehost op Azure: https://bartkrabbe.azurewebsites.net.
(https://bartkrabbe.azurewebsites.net/swagger/index.html voor een visualisatie van de backend in swagger documentatie)

Endpoints	Methode	Parameters	Omschrijvingen
/account/login	POST	gebruikersnaam en wachtwoord	Inloggen met gegevens, JWT token terug
/account	GET	JWT bearer token	Het account object
/line	GET	JWT bearer token	Alle lijnen die gekoppeld zijn aan de gebruiker
/line/machines/{lineId}	GET	line Id	Lijst met machines die in deze line staan

/line/machines/{lineId}/{machineId}	GET	line id, machine id	Een specifieke machine in een specifieke line, met daarbij de gehele historie van deze machine
/line/{lineId}/{machineId}	POST	line id, machine id, status, timestamp	Een status toevoegen aan een machine
/line/{lineId}/{machineId}	GET	lineId, machineId	Gets all the statuses from a single machine (history)
/line/{lineId}/{machineId}/{date}	GET	line id, machine id, date	Gets all the statuses from a certain machine at a certain day
/line/{lineId}/{machineId}/{date}/random	POST	line id, machine id, date	Voegt een heleboel statussen toe aan een specifieke machine in een specifieke line
W.I.P /ws/{companyId}	GET	companyId	Zet een websocket connectie op tussen de frontend en de backend, zodat de gebruiker de live data updates mee kan krijgen

9 Implementatie van Flutter en React Native

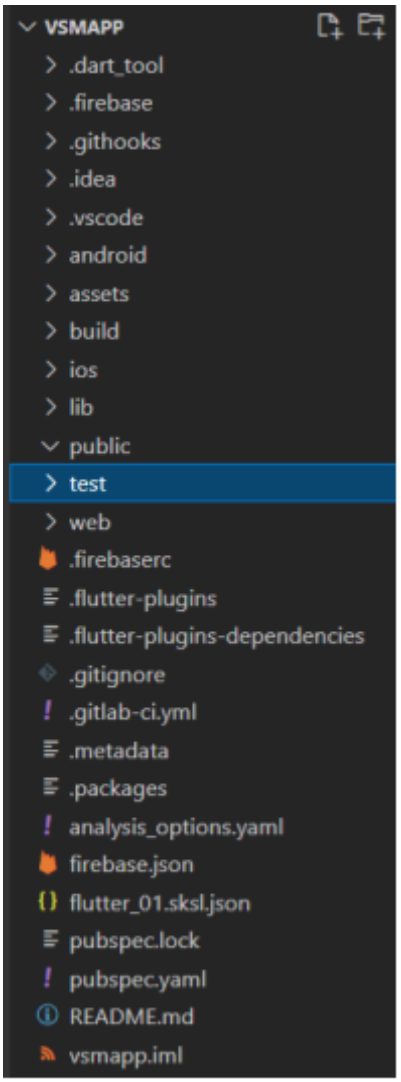
In dit hoofdstuk worden de implementaties van Flutter, React Native, Firebase en de backend besproken. Deze implementaties zijn allemaal (behalve Firebase) gemaakt in Visual Studio Code.

9.1 Flutter implementatie

Dit deelhoofdstuk gaat over de implementatie van de Flutter applicatie voor Mobile Operator Support. De Flutter web versie hiervan is te vinden op: https://biggestwebsite.web.app. De ontworpen app wordt meegeleverd als een APK bestand, ook zijn er screenshots gemaakt van de uitvoering van de app (Zie Figuur 31)

9.1.1 Projectstructuur

Dit deel gaat over de structuur van de Flutter frontend applicatie en er wordt uitgelegd wat het doel is van de bestanden en folders. De structuur van de Flutter applicatie is zoals figuur 21. Alleen de bestanden en folders die interessant zijn worden behandeld.



Figuur 21 Projectstructuur Flutter

9.1.1.1 Android & ios

In deze folders staan de build tools voor Flutter om de respectievelijke builds te maken voor de mobiele operating systemen. In deze folders staan dan ook bestanden met bijvoorbeeld de rechten die de app nodig zou hebben. In ons geval zouden deze permissies: biometrie, lokale opslag en toegang tot het internet voor API calls zijn.

9.1.1.2 .firebase / .firebaserc / firebase.json

In deze bestanden en folders staat de logica voor zowel het connecten met Firebase om push notificaties te kunnen ontvangen en om een build te kunnen versturen voor de Firebase hosted frontend van Flutter.

9.1.1.3 .gitlab-ci.yml

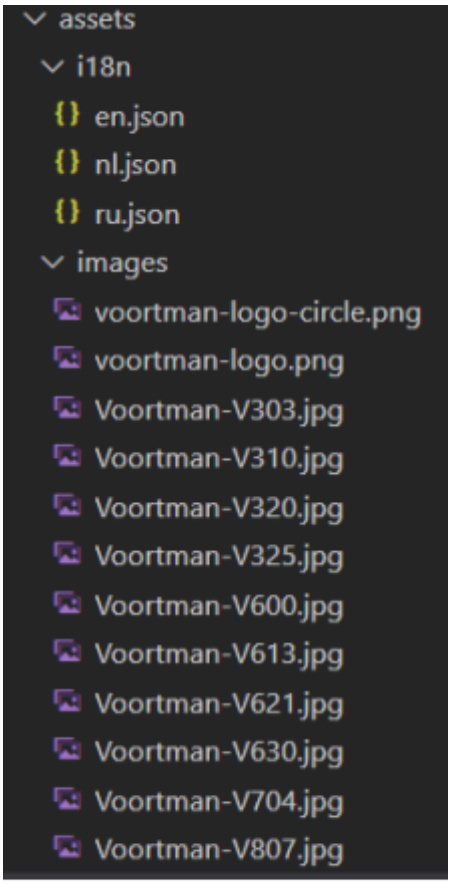
In dit bestand staat de logica waarmee GitLab een CI/CD pipeline kan maken en uitvoeren (Zie Bijlage D). Eerst wordt er een image gebruikt waarop Flutter en Dart staan, dan worden de aangeven stages (stappen) achter elkaar uitgevoerd. Deze stappen bevatten de logica voor het opschonen en testen van de code, het builden van een apk voor android en de laatste twee stappen van dit bestand builden en deployen de webapp naar Firebase.

9.1.1.4 Pubspec.yaml

In dit bestand staan alle dependencies van het project. Bijvoorbeeld de localstorage (voor lokale storage), http (voor het maken van API calls), flutter_local_notifications en de meerdere charts plugins.

9.1.1.5 Assets

Deze folder bevat de assets die in de applicatie gebruikt worden.



Figuur 22 Projectstructuur Flutter (assets)

9.1.1.5.1 i18n

Deze folder heet i18n, omdat de plugin die voor vertalen gebruikt wordt ook i18n heet. Deze folder wordt dus gebruikt om de verschillende talen van de app in op te slaan. Er staan momenteel 3 talen in deze map: Engels, Nederlands en Russisch. Engels en Nederlands zijn logische keuzes, maar Russisch is gekozen omdat het een ander alfabet heeft, zodat je sneller kan zien dat de app vertaald wordt.

9.1.1.5.2 Images

In deze folder worden de plaatjes van de machines bewaard om in de app te gebruiken. Dit zijn de huidige machines die Voortman allemaal verkoopt.

9.1.1.6 Lib

In de lib folder staat de daadwerkelijk logica voor de applicatie, van schermen, componenten en objecten. Deze folder is daarom ook het meest interessant. Daarom bespreken we bij dit deel elke subfolder. (Zie figuur 23)

9.1.1.6.1 custom_widgets

In deze folder zitten de hergebruikte componenten van de applicatie. Dit zijn de navigatiebar aan de onderkant van het scherm, een kaart component met daarin een machine lijn en een kaart component met een machine hierin. De laatste 2 worden voornamelijk gebruikt in een lijst formaat, waardoor het handig is om hiervan een apart object te maken, zodat deze code duidelijker en makkelijker te hergebruiken is.

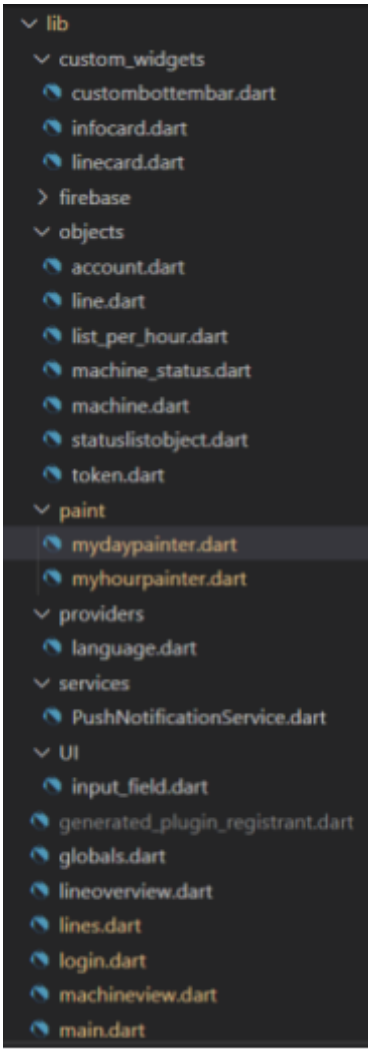
9.1.1.6.2 Objects

Zoals de folders naam beschrijft zit deze folder vol met objecten die gebruikt worden door de applicatie. Denk hierbij aan een Account, deze bevat informatie zoals gebruikersnaam en wachtwoord. Dit wordt gebruikt om de transfer van data makkelijker te maken tussen de schermen en de datalevering voor API calls.

9.1.1.6.3 Paint

In deze folder staan de paint componenten. Deze componenten zorgen ervoor dat de Event Stream Bar wordt getekend. Deze bar was een complex probleem om op te lossen, want er moest veel rekenwerk worden toegepast op de data. Om een voorbeeld te geven hoe dit probleem is opgelost pakken we de myhourpainter.dart erbij (Bijlage I voor 2 onderdelen van dit component).

Eerst wordt er een canvas aangemaakt met de aangegeven grootte. Vervolgens wordt 24 keer (een dag opgedeeld in uren) de paintList() methode aangeroepen. Deze methode tekent de daadwerkelijke rechthoeken op het canvas. Deze methode loopt door de statussen die in dat uur zijn meegegeven. Per status gaat hij dan de afstand berekenen die getekend moet worden op de canvas. Dit doen we door te berekenen hoeveel ruimte we hebben per seconde (canvas width / 60 /60). Als we deze variabel hebben kunnen we de tijd tussen statusveranderingen meten en dit vermenigvuldigen met de variabel. Dit geeft ons de 2e x coördinaat (de eerste hebben we al, namelijk waar de vorige status is gestopt).



Figuur 23 Projectstructuur Flutter

Er zijn 3 excepties bij deze regel, namelijk:

- Als er geen status wordt meegegeven in het eerste uur, gaan we er vanuit dat de machine uit staat en wordt er een witte streep getekend, totdat de methode in een ander uur een status tegenkomt.
- Als er geen statusverandering in bijvoorbeeld het 2e uur zit, maar wel in het 1e en 3e uur, dan trekt hij de status van het 1e uur door totdat hij bij de statusverandering van het 3e uur aankomt.
- Als de methode bij de laatste statusverandering in de lijst komt tekent hij deze status tot het einde van de 24 uur.

Als hij de correcte X coördinaten heeft berekend roepen we de canvas.drawRect() aan en geven deze de coördinaten, de kleur (met de methode getPaint(), deze methode returned de kleur die past bij de status. Dus rood bij error) en de hoogte van de rechthoek mee.

Als de methode klaar is met het tekenen van alle rechthoeken missen we nog de streepjes die de kwartier/uren aangeven. Deze methode wordt onderaan aangeroepen. We berekenen hierbij eerst hoeveel ruimte er tussen elk verticale streepje moet zitten (dat om de 5 minuten hoort te zitten). Zo'n streepje zou 12 keer voor moeten komen in een uur (60/5=12), dus wordt er een for loop gebruikt die om de 5 minuten dit streepje tekent. Dit werd alleen wat onduidelijk, omdat er dan veel streepjes kwamen te staan. Daarom is er gekozen om elke 15 minuten een streepje te tekenen, want dit maakt de grafiek leesbaarder. Welk streepje dit is wordt berekend door de huidige index op te vragen en te checken of modulo 3, 0 wordt. (bij 0,3,6,9,12) Hierbij wordt ook de text getekend die aangeeft hoe laat elk streepje is dus bij index 0 zou er 16:00 staan en bij index 3 16:15.

9.1.1.6.4 Providers

Dit bestand geeft aan de applicatie welke talen beschikbaar zijn om op te klikken als je op de globe klikt in het hoofdmenu. (Zie figuur 31)

9.1.1.6.5 Services

In dit bestand staat de logica om push notificaties binnen te krijgen en te verwerken.

9.1.1.6.5 Resterende bestanden

De resterende bestanden van deze folder zijn de verschillende schermen (behalve globals en generated_plugins). Deze schermen omvatten de volledige applicatie. In de main.dart wordt de navigatie van de applicatie bepaald. In het login scherm wordt de login logica verwerkt. Bij de lines, line overview en machine overview, worden de respectievelijke schermen getoond met de data die hierbij hoort.

9.1.2 De eisen

Flutter heeft zich goed opgehouden tegenover de geteste eisen. De eisen worden als volgt doorgelopen en gescoord.

9.1.2.1 Biometrie:

De biometrie is goed gelukt in Flutter. Na eerst wat problemen te hebben met wat interne errors vanuit Android (Bug in Android als het gaat om inloggen en dan cancelen, waardoor de biometrie het niet doet). Dit lag niet aan de Flutter applicatie, want alle applicaties die ik getest heb met biometrisch inloggen crashen hierdoor.

9.1.2.2 API Calls

De API calls worden gemaakt naar de backend die draait op Azure. Hiervoor moet wel de http library voor gebruikt worden (ontwikkeld door FLutter), ook moet het device goedkeuren dat we het internet gebruiken voor de applicatie. De calls zijn onder andere gebruikt voor het inloggen, opvragen van de machine lijnen en de data van bepaalde machines. Deze worden netjes verwerkt door Flutter tot objecten zodat hier beter mee gewerkt kan worden. Dus Flutter kan goed en makkelijk API calls maken.

9.1.2.3 Lokale Opslag

De lokale opslag wordt gebruik om de bearer token in op te slaan, dit token krijg je bij het inloggen van de applicatie en is nodig voor het maken van bepaalde API calls (want je wil niet dat iemand buiten je systeem API calls kan maken en data kan ophalen). Dit token wordt vervolgens gebruikt door de hele applicatie en zou het dus handig zijn om het lokaal op te slaan, hierdoor krijg je ook de mogelijkheid om in plaats van in te loggen met credentials in te loggen met biometrie, zodra de gebruiker geverifieerd is door de biometrie wordt het token opgehaald en kan deze worden gebruikt.

Flutter geeft dus goed de mogelijkheid om de lokale storage te gebruiken.

9.1.2.4 Push Notificaties

Flutter kan push notificaties ontvangen. Voor een voorbeeld van een push notificatie zie onderaan bijlage F. Deze notificaties zijn met Firebase geïmplementeerd (later meer over).

9.1.2.5 Ondersteuning van verschillende talen

Flutter kan gebruik maken van de i18n library die de applicatie de optie geeft om andere talen te ondersteunen. Deze talen kan je vervangen door op de globe in het hoofdmenu te drukken en een taal te selecteren. Je moet alleen zelf wel de vertalingen maken, dit is voor een kleine applicatie die niet veel talen hoeft te ondersteunen geen probleem, alleen als je een grote applicatie creëert die “alle” talen moet ondersteunen kan je beter voor betaalde vertaal opties kiezen (Google biedt dit ook [21]). Voor de Voortman applicatie zijn is i18n van Flutter goed en dus ondersteund Flutter meerdere talen.

9.1.2.6 Grafieken en Diagrammen

Voor de grafieken en diagrammen was er voor Flutter niet heel veel keuze tussen goede libraries. De meeste libraries boden niet niet wat nodig was, of de styling was achterhaald. Voor de piechart is uiteindelijk voor een pie_chart library gekozen die simpele styling en animaties toevoegt aan de grafiek en voor de Event Stream Bar is er gekozen om hem zelf te ontwikkelen via het Canvas dat Flutter biedt, omdat er geen library was die deze bar kon tekenen. Dit tekenen was complex om te begrijpen, maar het resultaat is er wel. Flutter is dus nog niet zo ver als het gaat om diagrammen en grafieken, hierin is Flutter een mindere keuze.

9.1.2.7 Styling

De overal styling van de applicatie is met Flutter makkelijk te ontwikkelen. De meeste componenten die je gebruikt hebben vanuit zichzelf al goede styling, maar Flutter zorgt ervoor dat je je tijd niet verdoet met styling die niet zou werken en biedt per component de styling tags aan die nuttig zijn.

Denk hierbij aan dat je de content wilt centreren. Dit kan door de styling van een container aan te passen en hierin je content stoppen, maar bij Flutter is er een component genaamd Center die dit automatisch regelt. Hiervan heb je nog veel meer voorbeelden, waardoor de styling van Flutter snel en effectief is (Figuur 31).

9.1.2.8 Testing

De basis die Flutter biedt voor testing is breedt, Flutter biedt zijn eigen test framework. Dit framework boots de schermen na (zonder ze visueel te tekenen) en runt je tests op dit gebootste device. De tests kunnen bestaan uit widget tests, unit tests en meer. Flutter is dus vanuit zichzelf een super framework om tests mee te kunnen maken.

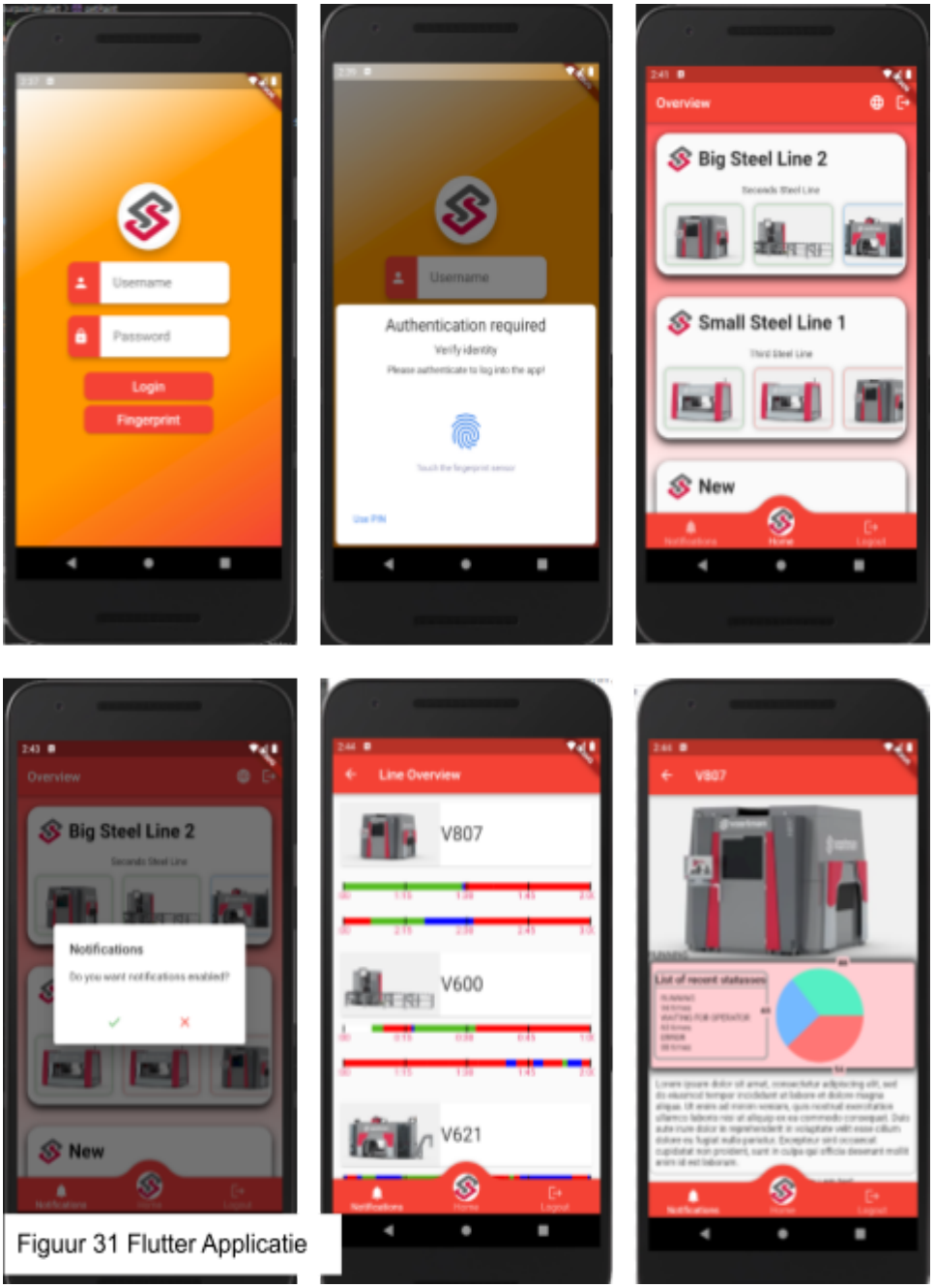
9.1.2.9 CI/CD

Flutter biedt goede opties als het gaat om CI/CD, ze voegen namelijk commando’s toe als analyse, die ervoor zorgt dat de code niet wordt volgestopt met ongebruikte libraries. Ook biedt Flutter vanuit

zichzelf build commands om de app te bouwen voor zowel Android, iOS en web. Flutter is dus een goed Framework om CI/CD op te zetten.

9.1.3 Visual Studio Code

Visual Studio Code biedt plugins om het developen makkelijker te maken en de Flutter plugin slaat de spijker op zijn kop. Deze plugin biedt veel functionaliteit, namelijk: Het opzetten en opstarten van emulators vanuit Visual Studio Code, buildtools die ervoor zorgen dat je van alle soorten componenten kan slepen/genereren/verwijderen(zonder dat je de rest van de code weer opnieuw moet formatteren), ook biedt deze plugin een soort linter, deze linter zorgt ervoor dat de code super leesbaar wordt (dit is ingesteld als je Ctrl+S drukt). Dit zijn maar een paar functies die deze plugin te bieden heeft, maar dit moest benoemd worden, want dit heeft het programmeren van de Flutter app veel beter gemaakt.



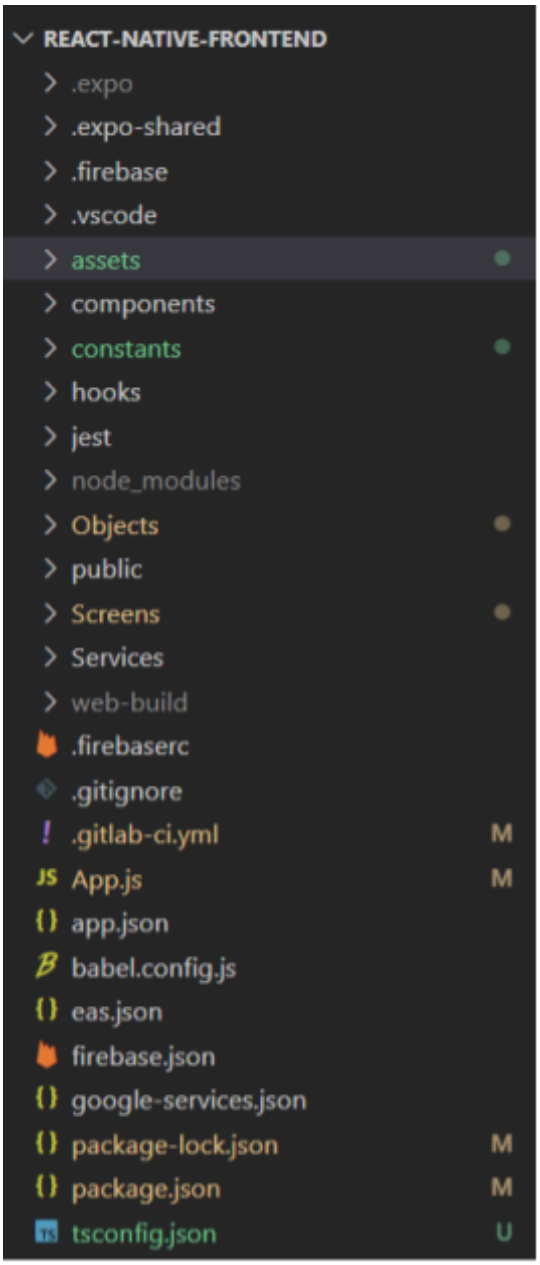
Figuur 31 Flutter Applicatie

9.2 React Native implementatie

Dit deelhoofdstuk gaat over de implementatie van de React Native applicatie voor Mobile Operator Support. De React Native web versie van deze applicatie is te vinden op: https://reactnativebiggestwebsite.web.app/. De ontworpen app wordt meegeleverd als een APK bestand, ook zijn er screenshots gemaakt van de uitvoering van de app (Zie Figuur 32)

9.2.1 Projectstructuur

In dit deel gaat het over de projectstructuur van de React Native applicatie. (Zie Figuur 33)



Figuur 33 React Native Projectstructuur

9.2.1.1 Assets

Deze folder (Figuur 34) bevat de assets die in de applicatie gebruikt worden.

9.2.1.1.1 i18n

Deze folder heet i18n, omdat de plugin die voor vertalen gebruikt wordt ook i18n heet. Deze folder wordt dus gebruikt om de verschillende talen van de app in op te slaan. Er staan momenteel 3 talen in deze map: Engels, Nederlands en Russisch. Engels en Nederlands zijn logische keuzes, maar Russisch is gekozen omdat het een ander alfabet heeft, zodat je sneller kan zien dat de app vertaald wordt.

9.2.1.1.2 Images

In deze folder worden de plaatjes van de machines bewaard om in de app te gebruiken. Dit zijn de huidige machines die Voortman allemaal verkoopt.

9.2.1.3 Components

Deze map (zie figuur 34) bevat onder andere de tests die geschreven zijn (map __tests__), ook zijn hier andere componenten in geschreven die niet gebruikt worden in de applicatie. Deze componenten zijn geschreven om getest te worden en om tests mee te oefenen.

9.2.1.4 Constants

Constants is de folder met daarin de constante waardes die door de hele app gebruikt kunnen worden.

9.2.1.5 Jest

Jest is het testing framework dat gebruikt is in het React Native project. In dit bestand wordt Jest geconfigureerd.

9.2.1.6 Node Modules

In de node modules folder staan alle npm modules die nodig zijn om dit project te runnen / builden / testen. Deze folder wordt automatisch gegenereerd en gevuld (met packages en libraries).

9.2.1.7 Objects

In deze folder heb ik de verschillende objecten die door de app gebruikt kunnen worden, zoals LineOverViewCard. Deze kan doordat dit een object/component is meerdere keren gebruikt worden door de app heen.

9.2.1.8 Screens

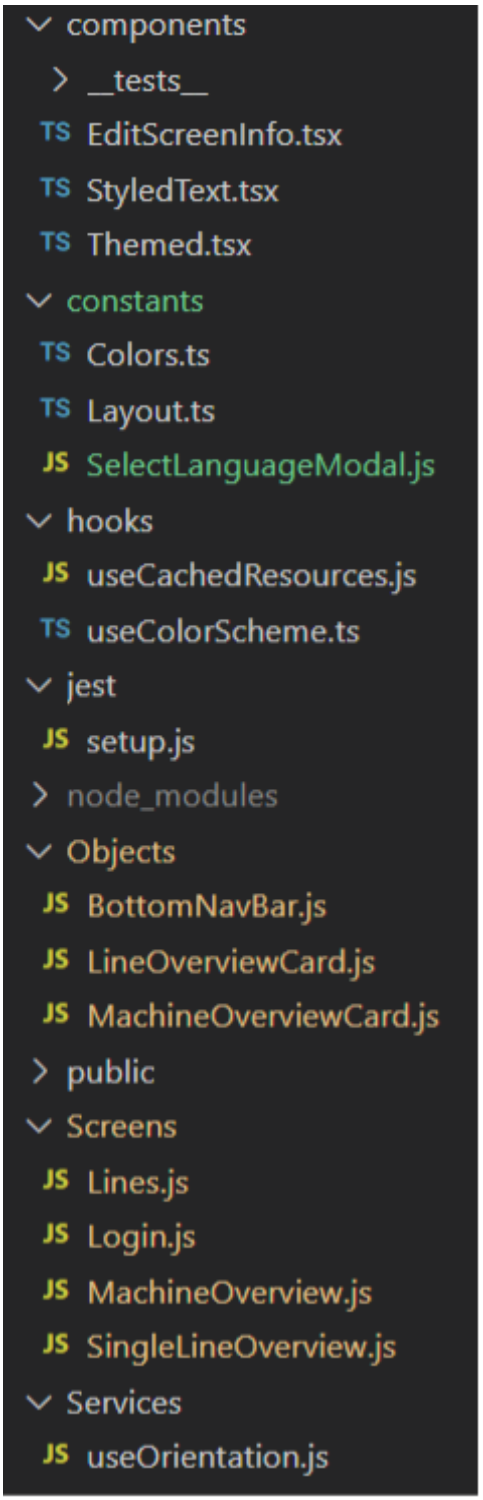
Deze folder bevat de verschillende implementaties van de ontworpen schermen. Deze schermen zijn gemaakt met Javascript.

9.1.1.10 .gitlab-ci.yml

In dit bestand staat de logica waarmee GitLab een CI/CD pipeline kan maken en uitvoeren (Zie Bijlage D). Eerst wordt er een image gebruikt waarop Flutter en Dart staan, dan worden de aangeven stages (stappen) achter elkaar uitgevoerd. Deze stappen bevatten de logica van het opschonen en testen van de code tot het builden van een apk voor android. De laatste twee stappen van dit bestand builden en deployen de webapp naar firebase.

9.1.1.11 .firebase / .firebaserc / firebase.json / google-services

In deze bestanden en folders staan de logica voor zowel het connecten met Firebase om push notificaties te kunnen ontvangen en om een build te kunnen versturen voor de Firebase hosted frontend van React Native.



Figuur 34 RN Project Structuur

9.2.1.12 App.json

Dit is het root bestand van het project (Officieel de eerste pagina die je bezoekt als je de app opent). In dit bestand worden de vertaal library (i18n), notificaties en navigatie geïnstantieerd.

9.2.2 De eisen

React Native heeft zich redelijk goed opgehouden tegenover de geteste eisen. Flutter kan namelijk gebruik maken van de biometrie van de telefoon,

9.2.2.1 Biometrie:

De biometrie is ook goed gelukt in React Native. Na expo libraries geïmporteerd te hebben kon ik (met toestemming van de devices) de biometrie aanroepen. Dit proces is hetzelfde als dat het bij Flutter gaat, dus hier scoren ze hetzelfde op.

9.2.2.2 API Calls

De API calls worden gemaakt naar de backend die draait op Azure. Wel moet het device goedkeuren dat we het internet gebruiken voor de applicatie. De calls zijn onder andere gebruikt voor het inloggen, opvragen van de machine lijnen en de data van bepaalde machines. Deze worden netjes verwerkt door React Native tot objecten JSON objecten. Dus React Native kan goed API calls maken.

9.2.2.3 Lokale Opslag

Voor de lokale opslag moet de expo plugin async storage worden gebruik. Met deze storage kunnen we de bearer token op slaan, dit token krijg je van de backend bij het inloggen van de applicatie en is nodig voor het maken van bepaalde API calls (want je wil niet dat iemand buiten je systeem API calls kan maken en data kan ophalen). Dit token wordt vervolgens gebruikt door de hele applicatie en zou het dus handig zijn om het lokaal op te slaan, hierdoor krijg je ook de mogelijkheid om in plaats van in te loggen met credentials in te loggen met biometrie, zodra de gebruiker geverifieerd is door de biometrie wordt het token opgehaald en kan deze worden gebruikt.

React Native geeft dus goed de mogelijkheid om de lokale storage te gebruiken.

9.2.2.4 Push Notificaties

Expo biedt mogelijkheden voor het ontvangen van push notificaties. Wel is dit proces moeizamer dan dat Flutter dat is. Want voor Expo moet je meerdere libraries gebruiken voor hetzelfde doel als dat Flutter dat met 1 library kan. Ook moeten de notificaties via Expo verstuurd worden, dit voegt weer een complexere laag toe aan de applicatie. Hierdoor is React Native minder efficiënt dan Flutter.

9.2.2.5 Ondersteuning van verschillende talen

React Native kan ook gebruik maken van de i18n library die de applicatie de optie geeft om andere talen te ondersteunen. Deze talen kan je vervangen door op de globe in het hoofdmenu te drukken. Voor de Voortman applicatie zijn is i18n van React Native net zo goed als de i18n Flutter applicatie.

9.2.2.6 Grafieken en Diagrammen

Voor de grafieken en diagrammen was er voor React Native genoeg keuze tussen goede libraries, maar de meeste van deze libraries boden hetzelfde of niet wat nodig was. Voor de piechart is uiteindelijk voor de chart_kit library gekozen die simpele styling aan de grafiek en voor de Event Stream Bar is er geen oplossing gemaakt. Dit komt door ziekte in de periode van het ontwikkelen van dit deel van de applicatie. Wel wordt er door mij verwacht dat deze grafiek mogelijk is met React

Native. Hierdoor kan er geen eerlijke beoordeling worden gegeven voor React Native op dit onderdeel dus slaan we dit over.

9.2.2.7 Styling

De styling van React Native was voor de mobiele applicatie goed te doen, maar toen ik hem op het web geprobeerd had bleek de styling voor het web van geen kant te kloppen. Dit was wel een groot verschil met Flutter (Flutter zag er bijna hetzelfde uit op mobiel vergeleken met web). Ook de schaduw van de kaartjes en menubalk was moeilijk te regelen met React Native. Dit komt omdat de schaduw van het element onder de box wordt getekend i.p.v. ernaast. Hierdoor moet je al 2 Views in elkaar stoppen om de schaduw te kunnen regelen. Je hebt ook nog 2 verschillende schaduw elementen eentje voor de website en eentje voor de app.

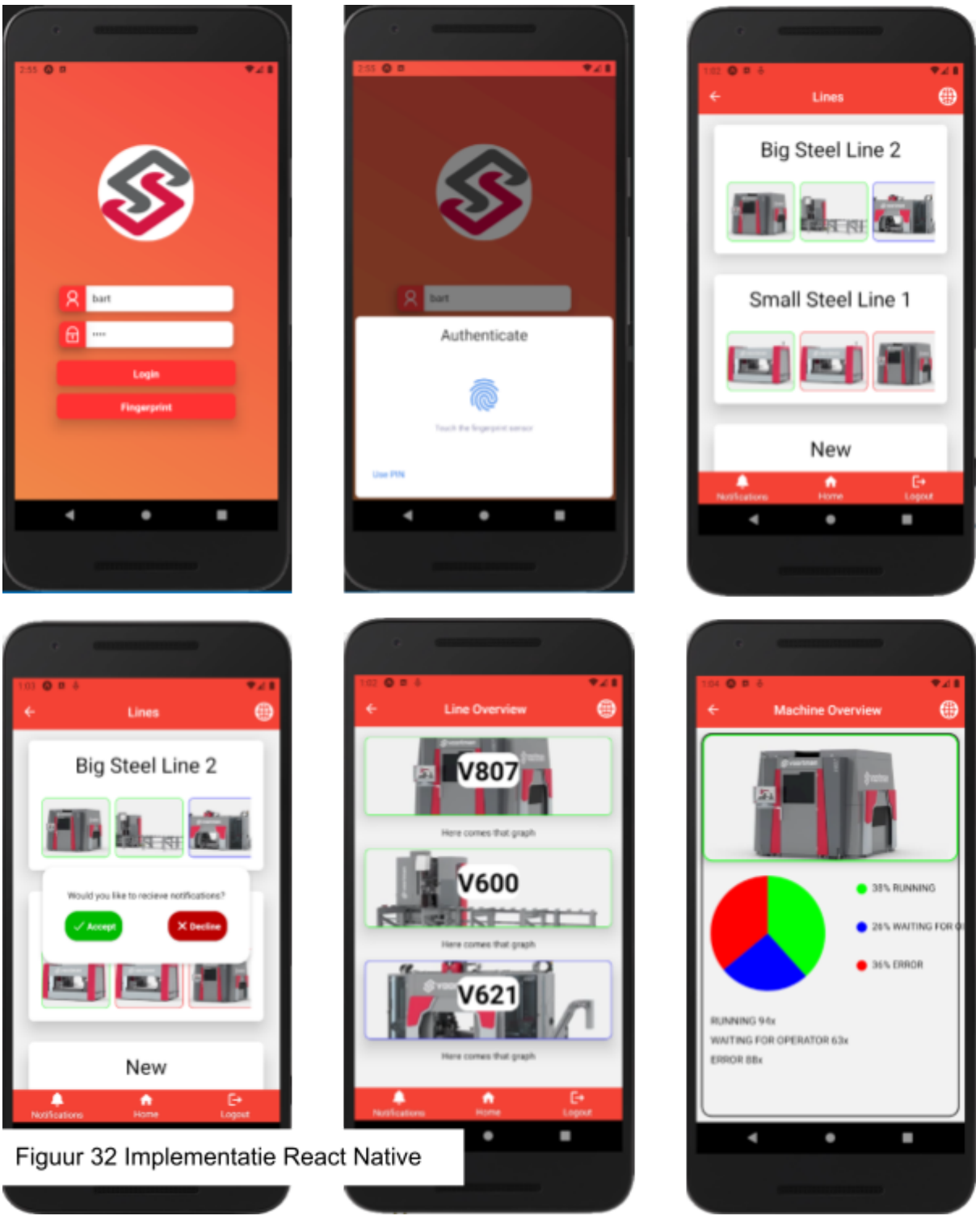
Voor de rest kon ik de app stylen met dezelfde technologie als hoe je een website zou stylen, dus het proces van de mobiele app stylen ging voor de rest gemakkelijk. Hierdoor krijgt React Native een iets slechtere score dan dat Flutter heeft.

9.2.2.8 Testing

De tests worden gemaakt met Jest, dit is een test framework dat veel talen kan testen. De Jest die worden uitgevoerd in React Native testen de widgets en functies. Ook kunnen deze testen data nabootsen en gebruik maken van de vertaal optie van i18n. Doordat Jest al veel gebruikt (meer opties voor testen, meer voorbeelden, meer implementaties) is dit framework beter dan de tests die flutter biedt.

9.2.2.9 CI/CD

React Native biedt net zoveel mogelijkheden met het automatiseren van de CI/CD, aangezien de tests geregeld kunnen worden met Jest, de builds met Expo en de deployement met Firebase zou ik zeggen data React Native goede CI/CD tools tot zijn beschikking heeft.



Figuur 32 Implementatie React Native

9.3 Firebase

In dit deelhoofdstuk wordt de implementatie van Firebase besproken, eerst wordt besproken hoe beide applicaties worden gehost op firebase en daarna hoe de push notificaties worden verstuurd naar de applicatie.

Om dit op te zetten was voor beide projecten de Firebase CLI nodig (dit is een download van Firebase), deze command line interface is nodig om tokens te genereren voor het pushen naar het hosting adres.

9.3.1 Web Hosting

Voor zowel werkende notificaties en een gehoste front-end (voor web) is er een firebase implementatie gemaakt. Dit geldt voor beide frameworks, waarbij Flutter gehost is op https://biggestwebsite.web.app/ en React Native op https://reactnativebiggestwebsite.web.app/. Deze websites worden beide bijgewerkt bij een push van de code naar de master/main branch van de codebase. Zodra de tests en builds slagen pushed de codebase (GitLab) met een token de build richting Firebase. Deze built wordt dan door Firebase automatisch gehost en vervangt de oude update.

Er zijn Firebase packages voor beide applicaties gedownload die bestanden toevoegen aan het project, die het automatiseren van de hosting mogelijk maken.

9.3.1 Push notificaties

Voor push notificaties is ook Firebase gebruikt om deze notificaties te testen voor Android. Dit werd gedaan door een project in Firebase op te zetten en hier push notificaties aan toe te voegen. Deze notificaties kunnen zowel handmatig worden verstuurd of via scripts in Firebase. (Perfect voor Voortman vanwege de automatie).

Om notificaties te kunnen versturen moet er data worden meegestuurd met de notificaties en moeten de apps staan geregistreerd bij Firebase. Dit wordt gedaan door het proces hierboven. Voor de stappen (die ook automatisch gedaan kunnen worden) die nodig zijn voor push notificaties zie bijlage F.

9.4 Backend implementatie

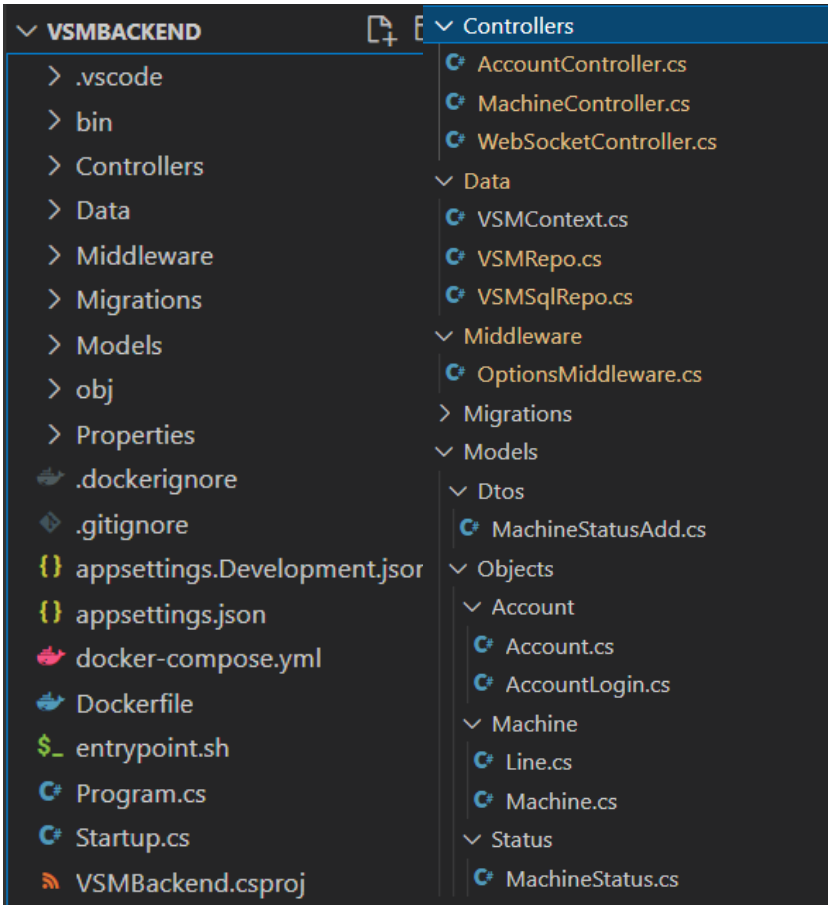
Dit deelhoofdstuk gaat over de backend die ontwikkeld is om de frontend te kunnen gebruiken met data vanuit een database.

9.4.1 Projectstructuur

Dit deel gaat over de structuur van de backend en welke bestanden wat doen. De structuur van de backend is figuur 29.

9.4.1.1 .vscode / bin / obj / properties

Deze folders en bestanden worden automatisch gegenereerd door Visual Studio code en het .NET framework. Hiermee wordt in dit project voor de rest niks mee gedaan.



Figuur 29 Projectstructuur Backend (.NET)

9.4.1.2 Controllers

In de controllers folder staan de methodes die uiteindelijk resulteren in de API endpoints. Deze methodes zien er als volgt uit (Zie ook wel figuur 30): Eerst komt er documentatie waarmee Swagger (een library) deze documentatie kan omzetten naar een webpagina met hierbij uitleg bij de verschillende methodes. (Voor deze website ga naar https://bartkrabbe.azurewebsites.net/swagger/index.html).

Na deze documentatie komt eerst een Authorize tag, deze tag checkt of bij de call naar deze endpoint een bearer token (in ons geval) is meegestuurd. Zo niet dan wordt de gebruiker/call afgewezen met een 401 status code (unauthorized).

Vervolgens komt het adres van de endpoint en wat voor soort endpoint het is (GET, POST, PUT, etc..). Om figuur 30 als voorbeeld te geven is hiervan het adres een GET {host}/account, omdat deze methode in het account controller bestand staat is het root adres /account en omdat er HttpGet staat is het een GET methode.

In de methode kan dan vervolgens worden verwezen naar bepaalde methodes en functies die intern de data ophalen en zo de gebruiker hierop kunnen voorzien.

Deze methodes moeten bij elk mogelijk codepad een return functie geven met hierbij een ActionResult (zoals aangegeven staat in methode naam). Dit ActionResult is een statuscode en een message ineen. Hierdoor weet de frontend wat de reactie is van de backend. Dit zorgt er ook voor dat de app een stuk schoner mooier en completer is, omdat je letterlijk elk pad afloopt.

```
/// <summary>
/// Gets the credentials from the account
/// </summary>
/// <remarks>
/// Sample request:
///
///     GET /api/account
///
/// </remarks>
/// <returns>User Credentials</returns>
/// <response code="200">Users credentials</response>
/// <response code="401">If the bearer token is not suplied or is not accepted</response>
/// <response code="404">If the account is not found</response>
[Authorize]
[HttpGet]
[ProducesResponseType(StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status401Unauthorized)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
public ActionResult<Account> GetAccount()
{
    int id = GetAccountId();
    var acc = _repository.GetAccountById(id);
    if (acc == null)
    {
        return NotFound();
    }
    return Ok(acc);
}
```

Figuur 30 API endpoint methode

9.4.1.3 Data

In de data folder staan de bestanden die de datastructuur bepalen voor in de database (De context), en een repo en sql repo. Het context bestand is nodig om Entity Framework door te geven wat er in de database moet komen te staan en om de resterende bestanden de datastructuur mee te geven.

Het repo bestand is een interface bestand met hierin alle methoden die gemaakt mogen worden naar de backend. Een interface bestand is handig voor als je meerdere typen repo's hebt. Denk hierbij aan een gemockte database i.p.v. een repo die met een echte database zou connecten.

Het SQL repo bestand implementeert het interface en vult alle methodes in met logica die data uit de database halen. Deze methoden worden vervolgens door Entity Framework omgezet (achter de schermen) om een SQL database te benaderen.

9.4.1.4 Middleware

Deze folder en het bestand hierin zijn gemaakt om de applicatie te kunnen laten runnen op localhost op chrome zonder dat je last krijgt van pre-flight errors. Een preflight wordt namelijk gestuurd door Google voordat je een call maakt om te kijken wat voor resultaten Google kan ontvangen. Dit moet opgevangen en verwerkt worden, zodat de backend niet crasht en de front-end geen data krijgt. Dit bestand lost dit probleem dus op.

9.4.1.5 Migrations

Deze folder wordt gegenereerd door Entity Framework. Deze folder staat vol met SQL commando's en structuur bestanden. Deze worden gegenereerd als de ontwikkelaar een nieuwe feature wil toevoegen aan de database (een nieuw object bijvoorbeeld). Je voegt het object toe aan het context bestand (eerder besproken), vervolgens roep je het commando *ef migrations add {name}* aan. Als het commando is aangeroepen maakt Entity Framework een geupdate database structuur file aan. Deze staat echter nog niet in de database, want de update moet nog worden gepushed. Dit wordt gedaan door het commando *ef database update* aan te roepen. Hierbij update hij de database (indien er verbonden kan worden met de database).

9.4.1.6 Models

Deze folder staat vol met objecten die door de backend gebruikt worden, denk hierbij aan een account object (met hierin gebruikersnaam, wachtwoord, etc..).

9.4.1.7 appsettings.json

In dit bestand staan variabelen zoals de database connectionstrings, JWT token keys en Logging informatie.

9.4.1.8 Dockerfile & docker-compose.yml

Deze bestanden staan vol met de logica die ervoor zorgen dat de backend gedockerized kan worden, waarna deze in een container gedraaid kan worden (zoals is gedaan met Azure).

9.4.1.9 Program / Startup / VSMBackend

Deze bestanden zorgen ervoor dat de backend weet welke plugins, tools en packages hij gebruiken moet. In de Program en Startup bestanden worden deze plugins en tools geconfigureerd. (Swagger bijvoorbeeld). Ook wordt er in deze bestanden een connectie opgezet met de database die staat gespecificeerd in het appsettings.json bestand.

9.4.2 Testing

Omdat de backend oorspronkelijk niet nodig was, zijn er voor de backend geen tests geschreven. Wel wordt er door aan elke API endpoint methode een ActionResult return verwacht, deze zorgt ervoor dat je zelf kritisch naar je code kijkt om te zien wat de methode allemaal kan teruggeven. Hierdoor kan worden geconcludeerd dat deze manier van "testing" genoeg is voor deze backend.

10. Conclusie

In dit onderzoek is gezocht naar een antwoord op de vraag: "*Welke framework implementatie is het meest geschikt voor Voortman voor het ontwikkelen van zowel native apps als een website die voldoet aan de opgestelde requirements en eisen?".* Hiervoor is zowel theoretisch onderzoek (veld en deskresearch) als praktijk onderzoek gedaan (implementatie).

Uit de resultaten van het onderzoek is gekomen dat Flutter en React native het beste scoren op de opgestelde eisen. Deze eisen zijn zo opgesteld dat de frameworks zoveel mogelijk getest worden op alle requirements die Voortman voor de Mobile Operator Support heeft. Echter was alleen onderzoeken niet genoeg om een resultaat en advies te bieden. Dus is de uitkomst van dit onderzoek verder getest door een kleine implementatie te maken van de Mobile Operator Support applicatie.

Hiervoor is eerst een design gemaakt, zodat de applicaties zoveel mogelijk op elkaar lijken. Hierdoor kan je de frameworks het beste vergelijken. Als je de twee implementaties met elkaar vergelijkt door de verwerkte eisen te bekijken en te beoordelen (Bijvoorbeeld biometrisch inloggen) komt Flutter er bovenuit.

Terwijl Flutter een redelijk nieuw framework is in de wereld van hybride frameworks, is het toch een heel mature en solide framework om applicaties mee te bouwen. Vergeleken met React Native is het makkelijker om Flutter apps te stylen en te bouwen. Flutter komt met kwalitatief betere componenten, omdat een professioneel team (gesteund door Google) aan deze componenten werkt. In tegenstelling tot React Native waarbij vooral de community plugins en componenten onderhouden en maken. Flutter loopt nog wel achter op het totaal aantal components, dit komt omdat de React Native community al langer bestaat. Flutter is ook beter voor Voortman omdat ze beter uit de geteste eisen gekomen zijn. Vaak is Flutter hierbij net wat handiger of sneller om deze functies te implementeren.

Beide frameworks leveren geen goede oplossing voor de event stream bar grafiek. Waardoor ze in dit aspect allebei een slechte keuze voor Voortman zijn.

Dus vanuit zowel het theoretische spectrum (veld en deskresearch) en het praktische spectrum (implementatie) is Flutter het beste framework om een implementatie te maken voor de Mobile Operator Support applicatie, dit komt omdat de Flutter app de opgestelde eisen beter kan implementeren dan React Native.

11. Reflectie

Onderzoek:
Het onderzoeken van de frameworks was een lastig proces, ik ben niet van de verslagen schrijven en moest nu een enorm onderzoek doen. Ik verloor hier wel motivatie door, ook al wist ik van tevoren dat het een groot onderzoek zou worden. Dit was een uitdaging voor mij persoonlijk en heb hierdoor mijzelf gepushed om meer onderzoekend te denken.

Daarna moest er ook nog verslag worden gedaan, wat ik ook een moeilijk proces vindt. Dit komt vooral omdat ik weinig opschrijf van wat in mijn hoofd zit. Ik kan alles makkelijk vertellen / uitleggen aan een persoon, maar het opschrijven hiervan en het in een logische volgorde zetten klikt nog niet zo. Daarom ben ik blij dat Theo mij daarbij heeft kunnen helpen.

Backend
Ik had aan het begin van de stage meer moeten doorvragen naar de backend, hierdoor ben ik later in de problemen geraakt, omdat deze er nog niet was. Deze had ik daarom ook niet meegenomen in mijn plan van aanpak. Toen deze backend gemaakt moest worden heb ik gebruik gemaakt van mijn kennis van backend opzetten in .NET. Hierdoor was er snel een backend opgezet, maar kon ik deze niet bereiken via mijn eigen mobiele device. Ik moest hiervoor een hosting platform zoeken.

Nadat ik een keuze had gemaakt tussen de hosting platforms, moest ik Azure gaan uitvogelen. Dit proces heeft mij best lang geduurd, maar is het wel waard geweest. Ik weet nu hoe ik een backend en SQL server draaiend kan krijgen in Azure.

Frontend
Voor het frontend deel van de applicatie bouwen ben ik zeer tevreden. Ik heb een 2 nieuwe talen moeten leren, namelijk Flutter en Dart. Dit proces heeft even geduurd, maar toen ik eenmaal doorhad hoe een Flutter applicatie werkt, vond ik het een fijne tool. Ook heb ik door dit proces besloten om zelf thuis apps te ontwikkelen met Flutter. Dit komt deels door het gemak van applicaties bouwen met Flutter.

Ook het CI/CD proces opzetten was een nieuwe ervaring in dit project. Ik had nog nooit eerder zo'n omgeving opgezet. Laat staan dat ik er iets van wist. Ik heb tijdens dit project geleerd hoe ik een website kan hosten met Firebase en hoe dit geautomatiseerd kan worden met GitLab. Ook heb ik geleerd hoe ik pipelines kan opzetten en deze bepaalde zaken te laten doen als je bijvoorbeeld alleen naar main pushed. (Bijvoorbeeld alleen bouwen als je naar main pushed).

Wel was het hele proces van werken bij Voortman zeer leerzaam.

Resultaat
Ik ben dik tevreden met de Flutter app die ik heb neer gezet. Dat Flutter door deze afstudeerstage mijn nieuwe favoriete hybride programmeertaal is zou al genoeg moeten zeggen over hoe goed Flutter is. Ook de Event Stream Bar diagram/grafiek ben ik zeer tevreden mee, want deze heb ik volledig zelf bedacht. De React Native app schiet ook goed op. Tijdens de kerstvakantie werd ik alleen ziek (buikgriep) en zat hierdoor aan mijn bed vast, waardoor ik in de week dat ik de React

Native app zou stylen en ontwikkelen niet zo productief was. Hierdoor is de React Native app naar mijn mening niet volledig af, maar ik ga nog aan deze app werken voor de presentatie om deze styling te fixen.

12. Bronnen

1. Voortman Steel Group. (s.d. c). *Voortman Steel Construction*. Opgeroepen op september 01, 2021, van voortmansteelgroup.com: https://www.voortmansteelgroup.com/voortman-steel-construction

2. Voortman Steel Group. (s.d. d). *Cultuur en waarden*. Opgeroepen op september 03, 2021, van voortmansteelgroup.com: https://www.voortmansteelgroup.com/cultuur-en-waarden

3. Academind. (2019, 14 augustus). *Which one is best for you? Flutter, React Native, Ionic or NativeScript?* [Video]. YouTube. https://www.youtube.com/watch?v=PKRXbLnfXXk

4. PeterCodes. (2021, 30 augustus). *Built a TodoApp on React Native, Flutter and Native. Which one is better?* [Video]. YouTube. https://www.youtube.com/watch?v=kjvfqfNadcM

5. Fireship. (2021, 4 oktober). *React Native vs Flutter - I built the same chat app with both* [Video]. YouTube. https://www.youtube.com/watch?v=X8ipUgXH6jw

6. Martin, S. (2020, 23 september). *React Native vs. Flutter vs. Ionic - Better Programming*. Medium. https://betterprogramming.pub/react-native-vs-flutter-vs-ionic-46d3350f96ee

7. AllBright.io. (z.d.). *ionic vs react native vs flutter*. https://allbright.io/blog/ionic-reactnative-flutter-comparison/

8. Aparna. (2019, 19 november). *The 10 Best Hybrid App Frameworks in 2021*. MobileAppAaily. https://www.mobileappdaily.com/best-hybrid-app-frameworks

9. OPTASY. (2020, 10 mei). *What's the Best Hybrid App Development Framework in 2020?* Medium.https://medium.com/@OPTASY.com/what-hybrid-app-development-framework-should-you-use-for-your-projects-in-2020-top-3-999662b30cce

10. Barot, S. (2021, 14 september). *Best Hybrid App Development Framework in 2021*. Aglowid IT Solutions. https://aglowiditsolutions.com/blog/best-hybrid-app-development-framework/

11. Jason, K. (2021, 1 december). *5 Best Hybrid App Development Frameworks For 2021*. DEV Community.https://dev.to/jason_kane7/5-best-hybrid-app-development-frameworks-for-2021-151l

12. Bernheim, L. (2019, 7 november). *20+ Largest Web Hosting Companies in 2021*. HostingAdvice.Com.

https://www.hostingadvice.com/how-to/largest-web-hosting-companies

13. Flutter. (z.d.). *charts_flutter | Flutter Package*. Dart Packages. https://pub.dev/packages/charts_flutter

14. ayushpgupta.com. (z.d.). *pie_chart | Flutter Package*. Dart Packages. https://pub.dev/packages/pie_chart

15. Flutter. (z.d.-b). *flutter_charts | Flutter Package*. Dart Packages. https://pub.dev/packages/flutter_charts

16. Morioh. (z.d.). *Top 10 Amazing Flutter Chart Libraries*. https://morioh.com/p/c214eaec953b

17. Viro Media. (z.d.). ViroReact React Native AR/VR. https://viromedia.com/viroreact

18. Martin, S. (2021, 16 december). *React Native vs. Flutter vs. Ionic - Better Programming*. Medium. https://betterprogramming.pub/react-native-vs-flutter-vs-ionic-46d3350f96ee#:%7E:text=Popularity%20and%20Community&text=According%20to%20Statista%2C%20React%20Native,be%20at%20the%20top%20position.

19. Amazon Inc. (z.d.). *AWS Educate*. Amazon Web Services, Inc. https://aws.amazon.com/education/awseducate/

20. Starikov, H. S. (z.d.). *GitHub - indiespirit/react-native-chart-kit: 📊React Native Chart Kit: Line Chart, Bezier Line Chart, Progress Ring, Bar chart, Pie chart, Contribution graph (heatmap)*. GitHub. https://github.com/indiespirit/react-native-chart-kit

21. Google Inc. (z.d.). *Cloud Translation |*. Google Cloud. https://cloud.google.com/translate/

13. Bijlagen

A. Interview over de aanleiding van de opdracht met Dominic

Wie ben je en wat doe je bij Voortman?

Dominic Stremmelaar

Product manager Cloud bij Voortman Steel Group (VSM), product manager van de cloud diensten die ze willen aanbieden aan de klanten. Hij is voor de rest verantwoordelijk voor het ontwikkelteam van de cloud.

In kort woorden uitleggen wat de huidige situatie is?
Machines draaien op software (VACAM). Klanten willen graag inzicht wat in de machines gebeurd, maar ze willen ook graag buiten de machines inzicht krijgen hierover. Momenteel stoppen ze hiervoor een USB in de IPC en kopiëren ze de interne database. Maar dit is niet schaalbaar.

Daarnaast krijgt Voortman van de klanten verzoeken om deze data beschikbaar te maken (statistieken, de huidige status, hoeveel is er al gezaagd, hoelang gaat een boor mee). Machines kent deze concepten (Hoeveel balken er p/s worden verwerkt) ook nog niet, alleen gefocust op bijv. een gat boren op een specifieke locatie.

Wanneer werd het project bedacht?
Vorig jaar bedacht. Als een SaaS dienst voor inzicht in de machines huidige status, vorige status en een overview. MOS bedacht ook in het begin, maar het project wat ik doe is afgelopen maanden naar voren gekomen.

In kort uitleggen wat de gewenste situatie is?
De klant wil meer inzicht hebben in de productie machines. Zodat ze maximaal staal kunnen produceren en verwerken.

Begin 2022 een MVP waardig product klaar hebben om aan klanten te verkopen.

Wanneer ben jij aan het project gekoppeld?
1 januari 2021.

Kan jij uitleggen wat jouw doel is in dit MOS project?
Begeleiden, keuzes maken tussen architectuur ontwerpen.

Hoe lang schat jij de loopduur van het project in?
Software as a service loopt eeuwig door. Ze willen altijd nieuwe services bieden voor klanten, waardoor Voortman meer centjes kan verdienen. Ziet het ook als product niet als project.

Wanneer wordt er gewerkt aan de backend?
Eind oktober de backend af. SnS draait dan dus ook

Is de backend klaar als ik deze nodig heb voor het project?
Ja

Moet de app in Ionic worden gemaakt (Of is dit snel gekozen)?
Is deels onderzocht, paar frameworks naast elkaar gehouden. Voor nu Ionic gekozen omdat ze daar makkelijk op terug kunnen komen. Grootste deel van de front-end kan worden gebruikt in andere delen.

Hebben jullie al bij klanten gevraagd over deze app?
Ja VSC vraagt hier al naar. Het product is ook best logisch en nodig, dat klanten dit sowieso gaan gebruiken.

Wordt het een betaalde service of een gratis service voor de klanten?
Betalen als het bedrijf voor de service, dan kan het bedrijf accounts toevoegen.

Waarom wouden jullie een afstudeerder bij dit project?
Voortman werkt graag met afstudeerders

Wat verwachten jullie dat ik ga bijleveren aan dit project?
Een PoC van die is gebouwd en gecontroleerd is met de Multi criteria analyse

Verwachten jullie alleen front-end werk en onderzoeken?
Ja verwachten niet dat ik dat ik te maken krijg met de backend behalve calls maken hierna. Eventueel nog iets kwa AWS SnS doen.

Als ik een goed project oplever dat werkt, wordt dit dan ook gebruikt, of meer als een voorbeeld?
Het PoC wordt wel gebruikt, maar er zal waarschijnlijk niet verder worden gebouwd op mijn app zelf.

In een eerder gesprek had je het over XCODE, maar wat wordt hiermee gedaan dan? (Dit is een IDE, heb je dit nodig om Apple apps te pushen naar de app store?)

Alleen voor het pushen naar de App Store

B. Vragen om achter de functionele requirements te komen

Interviewer, Bart Krabbe (**B**)

Geïnterviewde, Dominic Stremmelaar (**D**)

Geïnterviewde, Kelvin Ten Vregelaar (**K**)

App:

- Moet de app ook horizontaal gedraaid kunnen worden? (Of alleen de website)

> ***Kelvin*** *– indien nodig, afhankelijk van grootte van mobiel. Ook hangt er vanaf welke data gedisplayd wordt, sommige schermen misschien wel handig om te kunnen draaien*
>
> ***Dominic*** *– Voorziet niet dat het nog horizontaal nodig is.*

- Moet er rekening mee worden gehouden voor mensen met handicaps, dingen zoals tekst oplezen als een push notificatie binnenkomt, colorblind mode, etc… (toegankelijkheid)

> ***Kelvin*** *– Colorblind mode niet echt. Ervoor zorgen dat de kleuren niet verwarrend worden in eerste plaats.*
>
> ***Dominic*** *– kleurenblindheid modus, voor de rest niet echt rekening houden met mensen met een handicap. Wel gebruik maken van icoontjes. Tekst bij de error.*

- Verschillende talen ondersteunen?

> ***Kelvin*** *- Het liefst wel ja.*
>
> ***Dominic*** *- Ja de app moet meerdere talen kunnen ondersteunen want Voortman zit in meerdere landen met andere talen: Nederland, Russisch, Engels, Frans, etc..*

Inloggen:

- Welke gegevens heeft een gebruiker nodig om in te loggen voor de eerste keer?

- Hoe kan een gebruiker inloggen na de eerste keer? (Welke gegevens, Welke manieren?)

- Hoe moeten de inloggegevens worden opgestuurd om een *token* te ontvangen?

- Hoe werkt dit inloggen (Incognito?)

> ***Bart*** *– Het inloggen is mij nog niet helemaal helder. Wat ik zelf verwacht is dat je voor het eerst op de app kwam. Op de sign up knop drukt (verdwijnt na 1x inloggen). (Zoals figuur 1 & 2)*
>
> ***Kelvin*** *- De signup knop krijg je niet in de app.*

***Dominic** – Geen inloggen op de manier hoe jij (Bart) het dacht, geen signup met wachtwoord en gebruikersnaam.*

***Bart** – Helemaal niet?*

***Kelvin** – Je kunt niet zelf een account aanmaken.*

***Bart** – Oh nee, zo bedoelde ik sign up niet. Ik bedoelde meer dat je een username en token van het bedrijf zo meegeven om te signuppen.*

***Dominic** – Een soort enrollment?*

***Bart** – Ja enrollment.*

***Kelvin** – Ja dit hoeft niet, want je logt gewoon in, en wordt je doorgestuurd naar Cognito, naar externe URL.*

***Bart** – Dus Cognito logt je in?*

***Kelvin** – Cognito logt je in en als het je eerste keer is eenmalige wachtwoord invoeren en daarna inloggen met gebruikersnaam en wachtwoord, als je wil inloggen met Cognito. Pas als het inloggen gelukt is stuurt Cognito je terug naar de app, met een session_id die je kan gebruiken*

***Bart** – Het login doet er helemaal niet meer toe, dus inloggen met biometrie is niet meer nodig?*

***Kelvin** – Wat mijn idee is ,is dat we geen username en password gebruiken in de app, gaat allemaal via externe browser. Maar je krijgt 2 tokens een accestoken (tijdelijk geldig om data vanuit de backend op te halen) en een refreshtoken (om een nieuwe accestoken op te halen, langere tijd geldig). De refreshtoken locken we dan achter de biometrie en safe enclave (Als een gebruiker dit wil), zodat we nog wel gebruikers kunnen authentiseren*

***Bart –** En anders moet je weer opnieuw inloggen met cognito. **K** - Ja*

***Dominic** – Waarvan ik zou verwachten dat je de refresh achter een pincode kan zetten en dan kan vervangen door een vingerprint bijvoorbeeld.*

***Kelvin** – Je hoeft beide niet meer, er zijn libraries hiervoor die de biometrie kunnen aanroepen van de telefoon en via hier dus het token ophalen.*

***Dominic** – en als er geen biometric is gezet.*

***Kelvin** – Gewoon opnieuw inloggen met Cognito.*

***Dominic** – inlogcode in de app en dan die kunnen vervangen door de biometrics. Kan.*

***Kelvin** – Het belangrijkste van de app is dat we alleen het refreshtoken lokaal opslaan (Als de gebruiker hierop toestemt) en deze locken achter de biometrie van het device.*

***Bart** – Kan je dan met dat sessionId calls maken naar de back-end?*

***Kelvin** – Ja/Nee je kan call maken naar de backend om je accestoken op te halen die je dan weer kan gebruiken om data op te halen voor dat sessionId.*

***Bart** – Het opnieuw inloggen van gebruikers kan dus zelf ingesteld worden, in cognito (30 dagen).*

- Moeten gebruikers kunnen uitloggen? (Of wordt dit automatisch gedaan bij het sluiten van de app?)

***Bart –** Kan je uitloggen als gebruiker uitloggen via een knop?*

***Dominic** – Ja.*

***Bart –** Ik bedoel ook wel een beetje in trend van, als je de app op de achtergrond hebt, of de app sluit, opnieuw opstart, etc..*

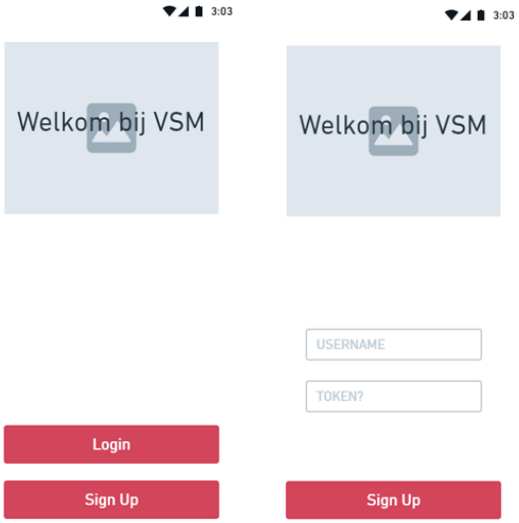
***Kelvin** - De app is vergrendeld achter biometrics zodra je de app even sluit.*

***Dominic –** Het is wel veilig, anders pas na 15 min weer inloggen.*

***Bart –** Ik weet ook niet hoe dat zit met de regelgeving van de appstore?*

***Bart –** Het is natuurlijk wel een goede beveiliging.*

***Dominic –** Het ding is dat de app geen weet heeft van wie de app opent als er geen inlog systeem (of iets in die richting) opzit om een gebruiker te authentiseren.*



Gebruikers:

***Bart** – Zijn er meerdere typen gebruikers~?*

***Dominic** – Ja*

***Kelvin** – Meer zien als meerdere rollen in de app zoals een admin die mensen kan toevoegen etc..*

***Bart** – Oke duidelijk*

***Kelvin** – Hogere rol is meer recht in de app.*

Machine Lijnen:

***Bart** – Hoe selecteert een gebruiker een lijn?*

***Dominic** – Het zou kunnen gebeuren dat een gebruiker uit meerdere lijnen moet kiezen, maar deze usecase is niet veel voorkomend.*

***Bart** – Ik dacht er ook al zelf aan dat als een gebruiker maar recht heeft op 1 lijn dat deze dan automatisch wordt getoond i.p.v. een menu met 1 keuze?*

***Kelvin –** ja zo dachten wij er ook wel over na*

***Bart** – Zit deze logica al in een inlogtoken verwerkt?*

***Dominic & Kelvin –** Nee deze token weet dat niet, dit regelt de back-end aan de hand van dat sessionId.*

***Kelvin –** De backend weet aan de hand van jou session Id wat voor rechten/machines/etc.. jij als gebruiker hebt.*

***Bart -** Hoe switcht een gebruiker van lijn?*

***Bart –** Je kan subscriben op meerdere lijnen, maar je kan ook teruggaan naar het lijn menu.*

Notificaties:

***Bart -** Na het inloggen een pop-up met tot hoelang je notificaties wil ontvangen? (Zie whimsical)*

***Dominic –** Ja dat klopt.*

***Bart –** Meldingen naar computers kan ook vanaf AWS is dit ook de bedoeling voor de websites via chrome popups bijvoorbeeld?*

***Dominic –** Vooralsnog niet, maar misschien als de app later een pwa wordt wel.*

***Kelvin –** Relatief nutteloos, want wanneer sta je nou met een computer naast een machine.*

- Hoe werkt meldingen ontvangen met AWS SnS (had gelezen dat je device ID en etc.. nodig had, dus dit automatisch doen?) Zelf

- Welke info wordt er in de notificaties gezet? Zelf

Bart - Moeten de notificaties wijzen naar de machine in de app (Als je op de push notificatie drukt automatisch doorverbonden worden naar de machine pagina.)

Dominic – *Zou wel handig zijn, maar dan eerder naar het overzicht van alle machines en niet specifiek naar 1 machine.*

Kelvin – Notificatie moet dan de specifieke lijn meegeven.

Bart - *Hoe stacken de notificaties? (Als je een notificatie van een machine krijgt (ID = 1) en machine (ID = 2) gaat ook stuk, wordt de melding dan overschreven, of komen ze op elkaar. Vervolgens als machine ID hetzelfde is, wat gebeurd er dan?)*

Dominic – *Dan worden ze onder elkaar gezet. Zodra er meer dan x notificaties zijn worden deze opgevouwen door de OS naar een meldingcentrum zoals whatssapp dat ook heeft.*

Kelvin – *Dat is eigenlijk wel een goede vraag, want stel je krijgt in eens 8 meldingen. En je hebt de meldingen alleen in de notificatie staan. Kan je niet zien wat er aan de hand is in de app.*

Dominic – *Als je meerdere lijnen hebt kan je in dat menu een error teken bij de lijnen zetten, zodat je alsnog weet als je de app opent welke lijn stuk is.*

Dominic – *Als er dan wat gebeurd op de andere lijn krijg je een melding in de app eventueel.*

Bart – *Je kan dus op 2 lijnen tegelijkertijd subscriben??*

Dominic – *Daar ga ik wel vanuit ja. Dat je vanuit het lijn menu aan kan geven van welke lijnen je meldingen wil ontvangen.*

Bart – *Kunnen gebruikers de notificaties uitschakelen?*

Dominic – *Dit lijkt mij wel handig.*

Bart – *Worden meldingen vaker gestuurd wanneer een machine langer stilstaat en er is al een melding gestuurd?*

Kelvin – *De machine ‘berichten’ wanneer de status veranderd. Hierdoor krijg je daarvan een melding dus krijg je niet meer notificaties van 1 status, zelfs als deze een langere tijd stilstaat.*

Wel kunnen wen dit in de back-end afvangen door in de back-end iets van een timer te implementeren die kijkt of de status van de machine is veranderd.

Machine Overzicht:

Bart – *Kom je na het inloggen bij het machine overzicht? Nee bij het lijn overzicht menu.*

Bart – *Welke informatie moet je in dit overzicht zien?*

Dominic – *Afbeelding van machine, huidige status, duur van deze status, kleurcodering. Naam van de machine. (Eventueel toekomst, waiting for operator, welke actie deze moet uitvoeren).*

Bart – *Moet deze lijst sorteerbaar zijn?*

Dominic – *Nee, niet perse. De sequence van die lijst wordt ingesteld door de klant. Dit wordt waarschijnlijk de volgorder van de machines in de lijn.*

- Waarop moet deze lijst sorteerbaar zijn? (Dit hoeft dus niet en de onderstaande ook niet)

- Als een gebruiker uitlogt en opnieuw inlogt moet de app dan de sorteerrichting onthouden.

- Moet een gebruiker machines kunnen filteren?

Bart - *Moet er op de machines geklikt kunnen worden om naar de specifieke machine te gaan?*

Dominic – *3 schermen, machine overview, machine specificatie en machine vergelijking.*

- Welke andere opties moet het overzicht kunnen bieden (Menu opties, etc..).

Machine Specifiek:

Bart - *Wanneer een operator op de machine view klikt, wat krijgt hij te zien?*

Dominic – *Kijk naar de user stories op JIRA (Taartdiagram die in procenten de status van de machine laat zien, Tijdsbalk van de statussen op de balk. Statussen in de lijst. Hoe vaak de machine deze status heeft gehad, tijdsduur van de status, plaatje van de machine, de naam van de machine.)*

Bart - *Moet de historie op die pagina exact worden getoond of kan het via een speciale knop (Scrollview naar beneden etc.. (Op de website zou dat dus anders zijn))* ***D –*** *Als het past 1 pagina zo niet 2.*

Bart - *Wat zijn de opties voor een historie pagina, (Per dag, per uur, per uur van de dag, etc…) Per week, per maand.*

Dominic – *Ja, als het kan de maand, of zelfs vanaf het begin. Maar dit is veel data en rekenwerk.*

Bart - *Welke statistieken zijn interessant om hier neer te zetten.*

Dominic – *Deze informatie is nog niet beschikbaar, maar willen we wel graag bieden aan de klant. We willen hierbij Tonnage gedraaid, aantal bewerkingen en aantal producten kunnen laten zien. Type profiel, aantal meter per profiel.*

Bart – *Dus vanuit die data kan je dit weer verkopen aan de klant.* ***D -*** *Ja*

Toekomst:

Bart – *Welke functionaliteit is nodig voor de toekomst van de app.*

Dominic – *Soort task scheduling, bepaalde operator een taak toegewezen kan worden die langs of op de lijn moeten gebeuren, zoals onderhoud.*

Meer mogelijkheden bieden voor analyse van de verschillende lijnen. Buffers tussen de machines in beeld kunnen brengen. Plaatje van de lijn kunnen tonen.

Vragen om achter de Non functionele requirements te komen.

Bart – *Welke programmeertaal heeft voorkeur?*

Kelvin – *Angular wordt al mee gewerkt binnen het bedrijf voor websites. En voor de backend wordt C# gebruikt.*

Bart – *Hoe belangrijk is de single codebase?*

Dominic – *Zo kwalitatief goed mogelijke code maken en zo makkelijk mogelijk onderhoudbare code. Hiervan gaan wij uit dat single codebase het makkelijkste is.*

Kelvin – *Bespaart heel wat tijd, maar soms is het niet geoptimaliseerd voor 1 platform.*

Performance is ook niet een groot punt, want de app is niet zo zwaar.

Bart – *Is het dan ook belangrijk om vanuit bv Microsoft Visual Studio te werken?*

Kelvin – *Geld hetzelfde voor als de vorige vraag. Is het makkelijkste om samen in een omgeving te programmeren.*

Bart – *Want je kan ook gewoon vanuit notepad++ werken.*

Dominic – *Ligt aan de ontwikkelaar zelf.*

Kelvin – *Andere argumenten wegen zwaarder dan welke IDE we gaan gebruiken.*

Bart - *De app werkt met Cognito? Waarom gebruiken we Cognito?*

Bart – *Hier hadden wij het gisteren nog over, Cognito wordt gebruikt om users te authentiseren.*

Bart – *De app moet AWS SnS ontvangen. Ik weet dat voor Android met google services moet werken.*

Dominic – *Voor iOS moet je met Apple notification services werken.*

Bart – *Moet ik dit zelf instellen?*

Kelvin – *Niet helemaal. Google en APS misschien instellen, AWS niet.*

Dominic – *Wat ik nog wel wil hebben is dat we een bericht kunnen sturen naar AWS SNS met daarin de tijden waarin we notificaties willen ontvangen.*

Kelvin – *Dit kan waarschijnlijk niet in SnS.* ***B –*** *Volgens mij kan je je aanmelden en afmelden. (That's it).*

Kelvin – *Met een tooltje dit zelf regelen in de back-end.*

Dominic – *Valt niet binnen jouw opdracht.*

Bart – *Welke tests moet de app aan voldoen?*

Kelvin – *Devops waardig.*

Bart - *Tot in hoeverre willen jullie niet het wiel opnieuw uitvinden (kwa software)?*

Dominic – *Niks zelf uitvinden, alles moet online staan. We willen gebruik maken van veel gebruikte technologieën en willen niet voorop lopen in de mobiele app wereld.*

Bart - *Tot in hoeverre moet de app kunnen scalen?*

Dominic – *Unlimited.*

Bart – *Ligt ook niet aan de front-end, vooral aan de backend.*

Kelvin – *Iedere gebruiker heeft de app geinstalleer, dus front-end hosting valt mee, alleen voor de website moet er wat geregeld worden. Maar dit doet de AWS service zelf.*

Bart – *De app moet automatisch aanpasbaar zijn aan de beeldschermen. Moet de mobiel dan ook zijwaarts gehouden kunnen worden?*

Dominic – *Ja vind ik lastig****.***

Bart – *Een mobiel schuin houden levert weinig ruimte op, maar een tablet heeft allebei de richtingen meer ruimte.*

Dominic – *Niet doen, anders moet je 2x stylen.*

Kelvin – *Misschien dat je de tijdbalken wel beter kan lezen met een gedraaid scherm.*

Dominic – *Operators houden de mobiel toch niet zijwaards, dus niet nodig.*

Bart – *En voor tablets dan?*

Dominic – *Daar hebben wij nog niet over na gedacht.*

Bart – *Moet de app anders gestyled worden voor iOS vergeleken met Android.*

Kelvin – *Bij voorkeur wel, maar kwa gemakt wordt er vaak voor gekozen om het niet te doen.*

Dominic – *Menu moet onderin.*

Bart – *Swipen voor het menu zou ook kunnen*

Dominic – *Menu knop, logout knop, logo in het midden.*

Bart – *Is de app anders op de website i.p.v. de app.*

Dominic – *Nee.*

Bart – *Hoelang moet een operator maximaal wachten voordat zijn scherm is geladen? (10 seconden is de standaard max, maar na 2 seconden stoppen veel mensen al met wachten)*

Dominic – *Het liefst maximaal 1 seconde.*

Kelvin – *Maar omdat gebruikers in staalfabrieken staan, kan de internetverbinding slecht zijn. Hierdoor kan het langer duren voordat ze data binnen kunnen halen.*

Bart – *Dus de data pakketjes zo klein mogelijk houden en deze naar de gebruikers te sturen.*

Dominic – *Wel een opmerking, ik zou de app graag kunnen uitbrengen naar een testomgevingen, want wil dat de werknemers van de sales afdeling altijd de laatste updates hebben voor de testapp zodat we dit ook kunnen verkopen aan klanten. Alleen even uitzoeken op welke manier zo'n testapp handig is.*

Bart – *En geen APK installeren i.p.v. de app store? (is dit voor updates?, misschien ook wel af te vangen in de front-end).*

Dominic – *Nee gewoon via de playstore.*

Bart - *Als iOS ervoor zorgt dat PWA's push notificaties kunnen ontvangen willen jullie dan switchen?*

Dominic – *Als de technologie dan gebruikt wordt door meerdere grote tech-bedrijven, kunnen we gaan overwegen om over te gaan. Maar dit proces kan wel langer dan een jaar duren.*

C. Onderzoek Frameworks

Voor het onderzoek wat gedaan is naar de frameworks zie het bijgeleverde bestand genaamd “Framework Onderzoek”.

D. GitLab CI/CD Flutter

```
image: cirrusci/flutter:latest

cache:
  paths:
    - .pub-cache/global_packages

stages:
  - analyze_and_test
  - flutter_build_android
  - flutter_build_web
  - firebase_deploy



analyze_and_test:
  stage: analyze_and_test
  rules:
    - if: '$CI_COMMIT_BRANCH == "main"'
  before_script:
    - export PATH="$PATH":"$HOME/.flutter-sdk/.pub-cache/bin"
    - flutter pub get
  script:
    - flutter analyze
    - flutter test
  tags:
    - shared

flutter_build_android:
  stage: flutter_build_android
  rules:
    - if: '$CI_COMMIT_BRANCH == "main"'
  before_script:
    - flutter clean
  before_script:
    - export PATH="$PATH":"$HOME/.flutter-sdk/.pub-cache/bin"
    - flutter pub get
  script:
     - flutter build apk
```

```
  artifacts:
    paths:
      - build/app/outputs/apk/release/app-release.apk
  tags:
    - shared

flutter_build_web:
  stage: flutter_build_web
  rules:
    - if: '$CI_COMMIT_BRANCH == "main"'
  before_script:
    - export PATH="$PATH":"$HOME/.flutter-sdk/.pub-cache/bin"
    - flutter pub get
    - flutter clean
  script:
     - flutter build web --release
  artifacts:
    paths:
      - build/web
  tags:
    - shared

firebase_deploy:
  image: node:latest
  stage: firebase_deploy
  rules:
    - if: '$CI_COMMIT_BRANCH == "main"'
  script:
    - npm install -g firebase-tools
    - firebase deploy --token [redacted]-m "Deploying the best new feature ever."
  tags:
    - shared
```

E. Requirements

Should

De styling moet automatisch aangepast worden op de grootte van de app (Waarbij er geen rekening wordt gehouden dat de mobiel zijwaarts is).

De app maakt gebruik van iconen en tekst waar nodig is om informatie

te verduidelijken.

De app zorgt ervoor dat de kleuren goed uit elkaar te houden zijn

De app verbindt de gebruiker automatisch door als er maar 1 lijn is.

Als gebruiker wil ik deze grafieken en diagrammen kunnen filteren op datum en status.

Als gebruiker wil ik meerdere machines met elkaar kunnen vergelijken, zodat ik duidelijkheid kan creëren aan welke machine het probleem zou kunnen liggen.

Als gebruiker wil ik statistieken zien van de machines, hoeveel deze geproduceerd heeft, hoeveel meter, balken, tonnage.

De app moet makkelijk kunnen worden overgeschreven naar een ander ondersteunend programma als het niet meer ondersteund zou worden.

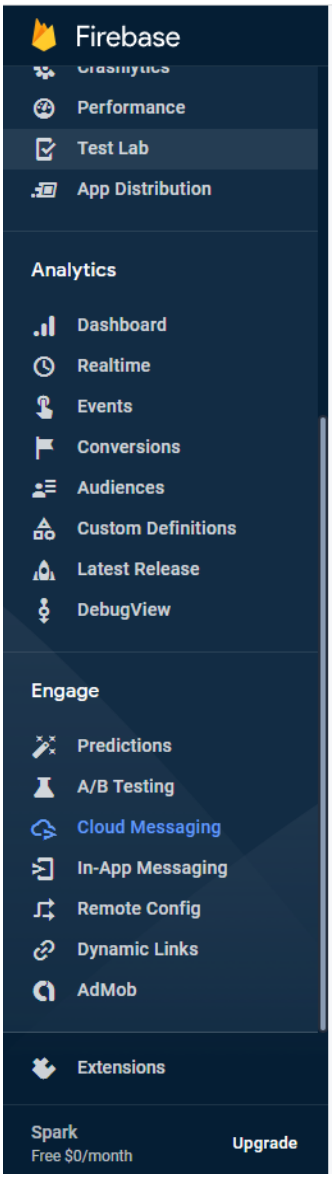
Could

Als administrator zou ik tasks willen opleggen aan operators rondom de lijnen. Dus onderhoudt van de banen, misschien wel het verplaatsen van bepaalde machines etc..

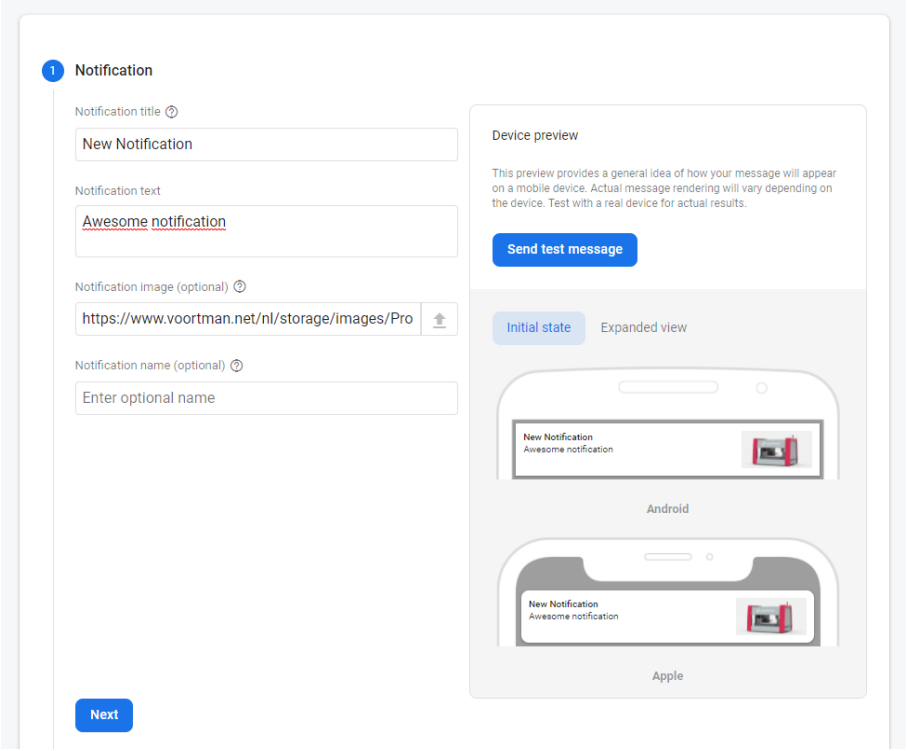
Als gebruiker zou ik een beter beeld willen krijgen van het totaal plaatje van de machines en banen. Hierbij een live plaatje tekenen wat er gebeurd op de baan.

F. Firebase push notifications

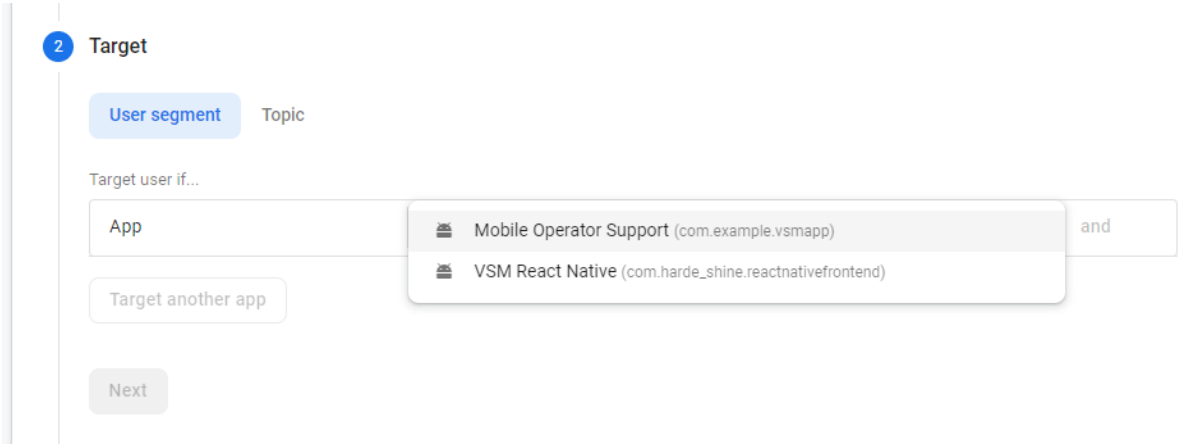
Voor het opzetten van een push notificatie moet er eerst naar de Cloud Messaging optie genavigeerd worden (Blauwe optie in het plaatje hier rechts ->). Vervolgens kan er met de knop creëer new notificatie een nieuwe notificaties gemaakt worden.

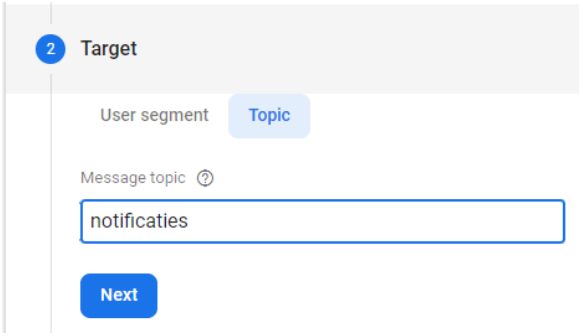


Dan krijg je een menu voor je met opties, deze opties (Titel, beschrijving etc..) worden ingevuld met de data die moet worden getoond in de notificatie, er wordt ook en voorbeeld gegeven voor hoe de notificatie eruit komt te zien. (zie plaatje hieronder)

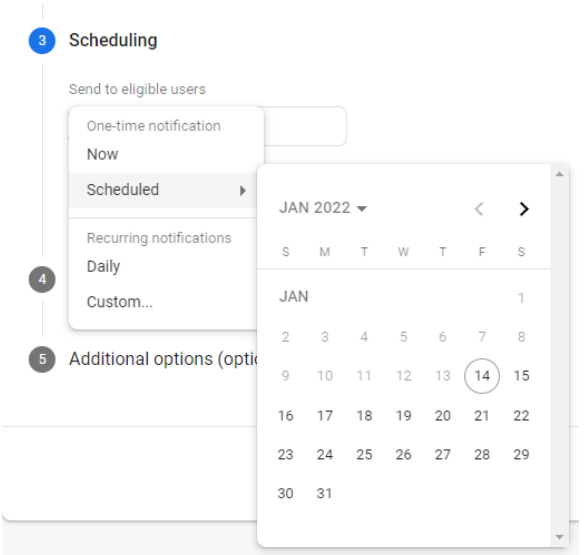


Vervolgens kan aangegeven worden naar welke specifieke applicatie de notificatie gestuurd moet worden, deze optie wordt niet gebruikt omdat we meer geïnteresseerd zijn in het versturen naar topics, want hierop kunnen gebruikers zich aanmelden. Dit aanmelden is nodig om de gebruikers uiteindelijk andere notificaties te kunnen geven als ze op verschillende productielijnen zouden staan. (Zie plaatjes hieronder)

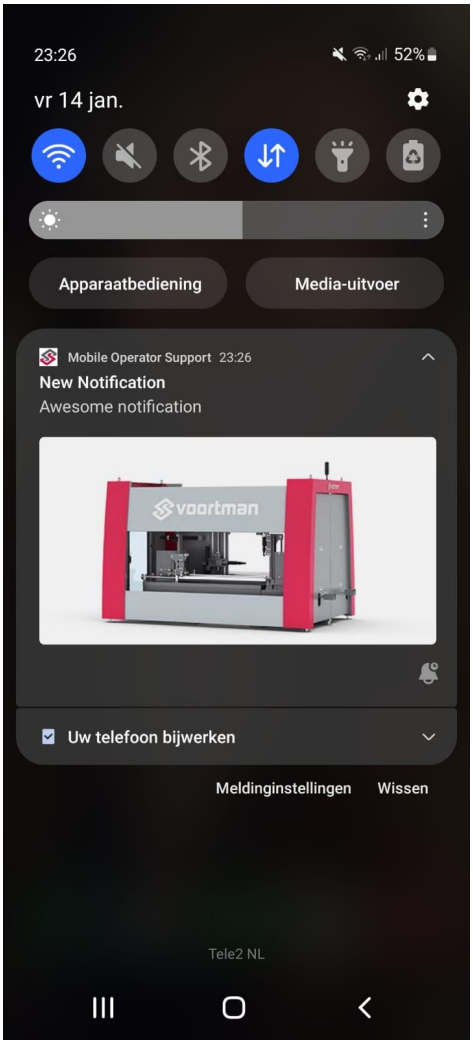


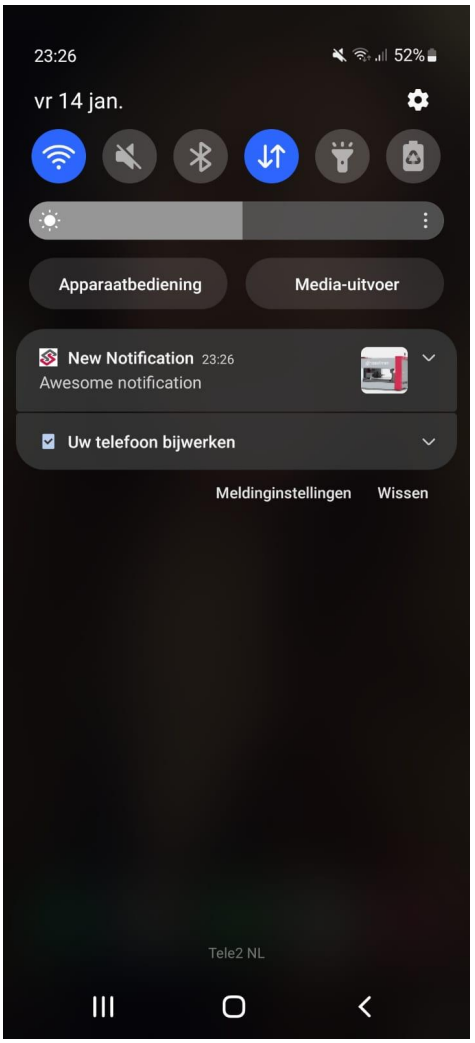


Daarna kan nog worden aangegeven wanneer de notificatie verstuurd moet worden. Voor onze use case is de “Now” optie perfect, want wij willen zo snel mogelijk een notificatie om de gebruiker te melden dat een machine stil zou staan.



Om te bewijzen dat ook daadwerkelijk notificaties binnenkomen heb ik via de gemaakte applicaties een screenshot gemaakt op mijn eigen telefoon van de push notificatie die hierboven is opgesteld. Je kan deze notificatie zowel groot maken als klein laten. Als je op deze notificatie klikt start de telefoon de applicatie op naar waar de notificatie is heen gestuurd. (De Voortman applicatie)





G. Plan van aanpak

Voor het plan van aanpak zie het bijgeleverde bestand genaamd “Plan Van Aanpak”.

H. GitLab CI/CD React Native

```
image: patwoz/expo-cli:latest

# Cache downloaded dependencies and plugins between builds.
# To keep cache across branches add 'key: "$CI_JOB_NAME"'
cache:
  paths:
    - .pub-cache/global_packages

stages:
  - test
  - build_web
  - firebase_deploy

jest-tests:
  stage: test
  before_script:
    - npm ci
  script:
    - npm test

reactnative_build_web:
  stage: build_web
  before_script:
    - npm ci
  script:
    - expo build:web
  artifacts:
    paths:
      - web-build
  tags:
    - shared

firebase_deploy:
  stage: firebase_deploy
  script:
    - npm install -g firebase-tools
```

```
                        -      firebase      deploy      --token
"1//09FxSBHM4flZaCgYIARAAGAkSNwF-L9IrSgJAox68A05lbSaW4ZTnWVvCo
zE0SBqYPzk-MPkgYmK0pnyMPBo5zKkeRpPzO4JNSr8"
  tags:
   -  shared
```

I. MyHourPainter methods

```
void paintList(Canvas canvas, Size size, ListPerHour listPerHour, double height, String previousStatus)
{
 double h = 10;
 double previousX = 0.0;
 // Space per second
 double spacePerSecond = size.width / 60 / 60;
 var startOfDay = DateTime(2021, 10, 29, listPerHour.hour);
 // For each status in that hour
 for (var i = 0; i < listPerHour.list.length; i++) {
  var paint = getPaint(listPerHour.list[i].status);
  var duration = 0;
  // If this is the first loop create a new difference and timestamp
  if (i == 0) {
   double dura = listPerHour.list[i].timeStamp.difference(startOfDay).inSeconds * spacePerSecond;
   canvas.drawRect(Offset(previousX, 0) & Size(dura, h), getPaint(previousStatus));
   previousX += dura;
  }
  if (i == listPerHour.list.length - 1) {
   var endOfHour = DateTime(2021, 10, 29, listPerHour.hour + 1);
   duration = listPerHour.list[i].timeStamp.difference(endOfHour).inSeconds * -1;
   canvas.drawRect(Offset(previousX, 0) & Size(duration * spacePerSecond, h), paint);
  } else {
     duration = listPerHour.list[i].timeStamp.difference(listPerHour.list[i + 1].timeStamp).inSeconds *
-1;
   canvas.drawRect(Offset(previousX, 0) & Size(duration * spacePerSecond, h), paint);
  }
  previousX += ((duration * spacePerSecond));
 }

 int currentHour = listPerHour.hour;
 double spaceBetweenSeconds = size.width / 60 * 5;
 for (var i = 0; i <= 12; i++) {
  if (i % 3 == 0) {
   canvas.drawRect(
     Offset(i * spaceBetweenSeconds, height - 1) & Size(2, h + 2),
```

```
        Paint()
          ..color = const Color(0xff000000)
          ..style = PaintingStyle.fill);
      // Draw the timestamp:
      paintHoursText(currentHour, i, canvas, spaceBetweenSeconds, height);
    } else {
      if (i == 12) {
        paintHoursText(currentHour, i, canvas, spaceBetweenSeconds, height);
      } else if (i == 0) {
        paintHoursText(currentHour, i, canvas, spaceBetweenSeconds, height);
      }
      // canvas.drawRect(
      //     Offset(i * spaceBetweenSeconds, height) & Size(1, h),
      //     Paint()
      //       ..color = const Color(0xff000000)
      //       ..style = PaintingStyle.fill);
    }
  }
}

void paintHoursText(int currentHour, int i, Canvas canvas, double spaceBetweenSeconds, double height) {
  String time = "";
  if (i == 12) {
    time = (currentHour + 1).toString() + ":00";
  } else if (i == 0) {
    time = (currentHour).toString() + ":00";
  } else {
    time = currentHour.toString() + ":" + (i * 5).toString();
  }
  TextSpan span = TextSpan(text: time, style: const TextStyle(color: Colors.pink));
  TextPainter tp = TextPainter(text: span, textAlign: TextAlign.left, textDirection: TextDirection.ltr);
  tp.layout();
  tp.paint(canvas, Offset((i * spaceBetweenSeconds - 16), height + 10));
}
```