Afstudeerverslag

AFSTUDEREN PROFITFLOW – 2023

Rosanne Stook 442502
SAXION | DEVENTER

Voorwoord

Hierbij presenteer ik met trots mijn afstudeerverslag voor de opleiding HBO-ICT. Dit verslag is het resultaat van een uitdagende maar inspirerende periode bij het bedrijf ProfitFlow in Deventer, waar ik mijn afstudeerstage heb gelopen. Gedurende mijn stageperiode heb ik gewerkt aan een applicatie waarmee zonnepaneelgebruikers en -installateurs in staat zijn om de prestaties van hun zonnepanelen te monitoren en te optimaliseren. De focus lag hierbij op het ontwikkelen van een gebruikersvriendelijke app die voldoet aan de eisen en wensen van beide partijen. Dit heeft geresulteerd in een innovatieve app die klanten en installateurs in staat stelt om eenvoudig contact met elkaar te onderhouden en eventuele problemen snel op te lossen.

Ik wil graag mijn dank uitspreken aan mijn stagebegeleider Joey Teunissen en het team van ProfitFlow voor hun ondersteuning en waardevolle feedback tijdens mijn stageperiode. Ook wil ik mijn dank uitspreken aan mijn docent Kevin Wilmink voor zijn begeleiding en begrip.

Ik hoop dat dit verslag bijdraagt aan de verdere ontwikkeling van de applicatie en de zonnepanelenindustrie als geheel.

Samenvatting

Dit afstudeerverslag beschrijft mijn afstudeerperiode bij ProfitFlow, waarin ik een eerste opzet voor een zonnepaneel-monitoring app heb ontwikkeld. Het doel van het project was het ontwerpen en implementeren van een gebruiksvriendelijke app voor het monitoren van zonnepaneelgegevens. Hierbij lag de focus op ontwerp, gebruikersinteractie, het uitvoeren van een gedegen gebruikersonderzoek en het implementeren van het ontwerp met een simpele werking waarbij een aantal externe API's in de app werden geïntegreerd.

Voor het onderzoek heb ik diverse methoden toegepast, waaronder interviews, enquêtes, het ontwerpen van wireframes, het ontwerpen van een design prototype en het testen van dit prototype op potentiële gebruikers, om inzichten te verkrijgen in hun wensen en eisen en de juiste functionaliteiten te definiëren. Uit dit onderzoek is uiteindelijk een ontwerp gekomen wat werd gerealiseerd in een mobiele applicatie.

Uiteindelijk is er een app uit gekomen waarmee gebruikers via een dashboard-achtig design hun zonnepaneel-gegevens kunnen bekijken, zowel real-time als historisch. Ook kunnen ze andere relevante informatie inzien, zoals weersgegevens en hun Co2 besparingen. Ook kunnen ze meldingen ontvangen en waarschuwingen inzien, omdat één van de belangrijkste doelen van de app was om het makkelijker te maken om problemen in te zien. Hierom kunnen gebruikers ook tickets sturen naar installateurs, die een andere versie van de app hebben, om zo in contact te komen met hun installateur over problemen.

Omdat het gaat om een eerste opzet is er nog heel veel aan de app te doen om hem klaar te stomen voor release. Het adviesgedeelte presenteert aanbevelingen voor toekomstige ontwikkeling. Deze omvatten onder andere het integreren van meerdere API's om verschillende soorten apparaten te ondersteunen, het verbeteren van het melding- en ticketsysteem en het genereren van geautomatiseerde rapporten voor klanten. Daarnaast wordt aandacht besteed aan technische optimalisatie, kostenbeheersing en het aanbieden van aanpasbare instellingen voor de gebruikers.

Dit afstudeerverslag biedt een gedetailleerde beschrijving van mijn afstudeerperiode, inclusief onderzoek, ontwikkeling en conclusies. Het vormt een waardevolle basis voor verdere ontwikkeling van EnergyFlow, zoals ik de app heb genoemd, als een gebruiksvriendelijke en uitgebreide monitoring app.

Table of Contents

Inleiding

Bedrijfscontext

ProfitFlow is een dynamisch IT-bedrijf gevestigd in Deventer dat zich richt op het optimaliseren van bedrijfsprocessen door middel van innovatieve softwareoplossingen. Sinds 2019 werkt ProfitFlow samen met eigenaren, managers, CTO's en producteigenaren om succesvolle software te lanceren. Ze hebben expertise in technologie en een sterk gevoel voor business, waardoor ze bijna elk vraagstuk in een succesvol digitaal product kunnen omzetten.

Het bedrijf heeft als missie om de uitdagingen van hun klanten op te lossen door middel van softwareoplossingen op maat, waarbij ze vooral gebruik maken van de nieuwste technologieën en inzichten. Ze richten zich vooral op de energie- en installatiebranche, waarbij ze met hun software producten de installatie van groene energie, zoals zonnepanelen, warmtepompen en laadpalen, ondersteunen. Hun belangrijkste project is momenteel OpusFlow, een all-in-one ERP systeem dat specifiek is ontworpen voor deze installateurs. Met OpusFlow hebben de installateurs toegang tot verschillende handige functionaliteiten, zoals facturatie, planningen, HR-beheer en projectoverzichten.

ProfitFlow is een vraag-gestuurde organisatie, wat betekend dat ze sterk inspelen op de behoeften van hun klanten. Het bedrijf is klein van omvang, maar groeit snel en heeft een sterke focus op innovatie en klantgerichtheid. Met hun sterke expertise en gevoel voor zaken zijn ze in staat om softwareoplossingen op maat te leveren die bijdragen aan de groei en efficiëntie van hun klanten.

Probleemcontext

De afgelopen jaren is de populariteit van groene energie sterk toegenomen vanwege het groeiende bewustzijn onder mensen van de impact van traditionele energiebronnen op het milieu en de zoektocht naar duurzame en milieuvriendelijke alternatieven zoals zonnepanelen, elektrische auto's en warmtepompen. Omdat zonnepanelen een van de meest populaire opties zijn onder deze apparaten, richt ik me in mijn afstudeerproject specifiek op dit type apparaat. Zonnepanelen zijn tegenwoordig overal te vinden en daarom is het belangrijk om gebruikers een gemakkelijke en efficiënte manier te bieden om hun zonnepanelen te monitoren. Momenteel hebben zonnepaneelgebruikers vaak niet op tijd inzicht in problemen met hun zonnepanelen, wat kan leiden tot vertragingen of onopgeloste problemen. Installateurs hebben ook vaak moeite om snel inzicht te krijgen in problemen. Bovendien verloopt het contact tussen gebruikers en installateurs vaak via onhandige kanalen zoals e-mail of telefoon. Om deze problemen op te lossen, kan er een gebruiksvriendelijke app ontwikkeld worden waarmee gebruikers hun zonnepanelen kunnen monitoren en installateurs gemakkelijk in contact kunnen komen met hun klanten. Buiten de scope van de opdracht is het ook het idee dat verschillende apparaten kunnen worden geregistreerd in de app, zodat gebruikers al hun apparaten kunnen monitoren. Dit kan bijdragen aan een bredere acceptatie en implementatie van groene energie.

Opdrachtcontext

De opdracht is om een eerste opzet van een applicatie te ontwikkelen waarmee zonnepaneel eigenaren en installateurs hun zonnepanelen kunnen monitoren en gemakkelijk met elkaar in contact kunnen komen om eventuele problemen op te lossen. De focus ligt op de front-end en gebruikersinteractie, maar er wordt ook een backend gemaakt die kan communiceren met verschillende API's van leveranciers zoals SolarEdge, SMA, Enphase of Solarman. Via de front-end wordt de zonnepaneeldata op een visueel aantrekkelijke manier weergegeven aan de gebruiker. Om tot een goed ontwerp en gebruikersinterface te komen, wordt er onderzoek gedaan naar de wensen en eisen van zowel installateurs als klanten. De bevindingen van het onderzoek worden vervolgens geïntegreerd in de applicatie en er wordt rekening gehouden met de manier waarop de gebruikers in contact willen komen met elkaar en welke informatie en functionaliteiten zij willen zien in de app. Uiteindelijk resulteert dit in een eerste versie van de applicatie, waarop later kan worden voortgebouwd.

Stakeholders

- **Zonnepaneeleigenaren:** Huiseigenaren die zonnepanelen hebben geïnstalleerd zijn directe stakeholders omdat zij de belangrijkste gebruikers van de applicatie zijn. Zij hebben baat bij het monitoren van hun zonnepanelen en het oplossen van eventuele problemen om hun investering te beschermen en te optimaliseren.

- **Zonnepaneelinstallateurs:** Ook zonnepaneelinstallateurs zijn directe stakeholders omdat ook zij de app veel zullen gebruiken. De applicatie kan hen helpen om de installaties van hun klanten gemakkelijker te monitoren en problemen sneller op te lossen, waardoor de kwaliteit van hun dienstverlening wordt verbeterd.

- **Energiebedrijven:** Het bedrijf waar de installateurs voor werken is een indirecte stakeholder omdat niet het gehele bedrijf de applicatie gebruikt maar zij wel baat hebben aan een goede werking van deze app. De applicatie kan bijdragen aan de populariteit en gebruik van groene energie door bewustwording en educatie te vergroten, efficiënt energiegebruik te bevorderen en klantbetrokkenheid te stimuleren. Het geeft voor gebruikers inzicht in de voordelen van groene energie over andere energiebronnen. Hierdoor kan de vraag naar groene energie toenemen, wat resulteert in meer klanten voor het energiebedrijf dat de app aanbiedt.

- **ProfitFlow:** Als ontwikkelaar van de app is ProfitFlow een directe stakeholder. Het bedrijf heeft een financieel en strategisch belang bij de ontwikkeling en het succes van de applicatie, omdat het kan bijdragen aan de groei en winstgevendheid van het bedrijf. Ook kan de ontwikkeling van deze applicatie bijdragen aan de positionering als innovatieve marktleider in de duurzame-installatie branche.

Onderzoek

Inleiding

In de wereld van duurzame energie is het monitoren van zonnepanelen van groot belang. Niet alleen zorgt dit voor een efficiënter gebruik van zonne-energie, het kan ook helpen bij het vroegtijdig opsporen en oplossen van eventuele problemen. Om deze reden is het ontwikkelen van een zonnepaneel-monitoring app voor zowel gebruiker als installateur een interessante optie. In het onderzoek dat is uitgevoerd wordt onderzocht hoe zo'n app kan voldoen aan de behoeften van zowel zonnepaneelbezitters als installateurs, terwijl deze zich onderscheidt van bestaande oplossingen.

Om tot een succesvolle app te komen is het essentieel om de wensen en eisen van beide gebruikersgroepen in kaart te brengen. Daarom zijn verschillende onderzoeksmethoden toegepast, waaronder interviews, enquêtes, available product analysis en gebruikerstesten. Op basis van deze resultaten zijn er wireframes en design prototypes gemaakt die zo veel mogelijk aan de wensen en eisen van beide partijen voldoen.

In dit afstudeerverslag wordt het onderzoek en de resultaten besproken. Hierbij wordt ingegaan op de behoeften van de verschillende gebruikers, de toegevoegde waarde ten opzichte van bestaande monitoring apps en de gewenste functionaliteiten en design elementen. Tot slot worden er aanbevelingen gegeven voor toekomstig onderzoek en voor de verdere ontwikkeling van de zonnepaneel-monitoring app.

Hoofd- en deelvragen

Het onderzoek bestond uit een hoofdvraag en een aantal deelvragen. Hier wordt beschreven wat deze vragen zijn en waarom zij belangrijk waren voor het onderzoek.

Hoofdvraag: Hoe kan een zonnepaneel-monitoring applicatie voldoen aan de behoeften van zowel zonnepaneelbezitters als installateurs, terwijl zij zich onderscheidt van bestaande oplossingen, met als doel het verbeteren van de gebruikerservaring en het optimaliseren van prestaties?

Deze hoofdvraag richt zich op het vinden van een oplossing die voldoet aan de behoeften van zowel zonnepaneelbezitters als installateurs en zich onderscheidt van de bestaande oplossingen. Het is belangrijk om te weten wat beide partijen nodig hebben en verwachten van de applicatie, zodat deze zo gebruiksvriendelijk en functioneel mogelijk kan worden gemaakt. Daarnaast is het van belang om te weten hoe de applicatie zich kan onderscheiden van de bestaande oplossingen om zo een meerwaarde te kunnen bieden aan zonnepaneelbezitters en installateurs.

Deelvragen:

1. Wat zijn de belangrijkste behoeften van de gebruikers van de app?

   Verschillende gebruikers hebben diverse behoeften die ze willen vervullen met behulp van de applicatie. Het is van essentieel belang om deze behoeften te begrijpen, zodat de app hierop kan worden afgestemd. Door inzicht te krijgen in de specifieke behoeften van de gebruikers, kan de app functionaliteiten en features bieden die precies aansluiten bij hun wensen en verwachtingen.

2. Hoe onderscheidt deze app zich van bestaande monitoring apps en op welke specifieke gebieden kan deze app een verbetering bieden ten opzichte van de concurrentie?

Om een succesvolle app te ontwikkelen is het belangrijk om te weten wat de huidige monitoring apps missen. Door te onderzoeken welke aspecten van bestaande apps niet goed werken of ontbreken kan de nieuwe app deze problemen oplossen en zich daarmee onderscheiden van de concurrentie.

3. Wat voor functionaliteiten willen zonnepaneelgebruikers graag in de app zien?

Het achterhalen van de gewenste functionaliteiten van zonnepaneelgebruikers is zeer belangrijk bij het ontwikkelen van een gebruikersvriendelijke app. Door te onderzoeken wat de gebruikers precies willen zien in de app kan er een app worden ontwikkeld die aan de wensen en eisen voldoet en daarmee beter aansluit bij de behoeften van de gebruikers.

4. Wat voor problemen zouden zonnepaneelinstallateurs met deze app graag willen oplossen?

Ook zonnepaneelinstallateurs hebben baat bij een gebruikersvriendelijke app. Door te onderzoeken welke problemen zij graag met behulp van de app willen oplossen kan er een app worden ontwikkeld die ook voor hen een meerwaarde heeft en hen helpt bij het oplossen van eventuele problemen.

5. Wat voor functionaliteiten willen zonnepaneelinstallateurs graag in de app zien?

Het achterhalen van de gewenste functionaliteiten van zonnepaneelinstallateurs kan helpen bij het ontwikkelen van een app die niet alleen aan de behoeften van zonnepaneelgebruikers voldoet, maar ook aan de behoeften van installateurs. Door ook hun wensen en eisen mee te nemen in het ontwerpproces kan de app hen helpen bij het oplossen van eventuele problemen en hun werkzaamheden efficiënter maken.

Methodes

Het doel van dit onderzoek was om inzicht te krijgen in de behoeften en wensen van zowel zonnepaneelgebruikers als -installateurs met betrekking tot een zonnepaneel-monitoring app en om te onderzoeken hoe deze app zich kon onderscheiden van bestaande oplossingen. Om deze vragen te beantwoorden werden verschillende onderzoeksmethoden gebruikt.

Allereerst werden er **enquêtes** uitgezet onder een grote groep zonnepaneelbezitters en een groep installateurs met vragen over functionaliteiten, design elementen en de gewenste informatie die ze willen kunnen zien in de app. Op deze manier kon worden bepaald welke functionaliteiten het belangrijkst waren om te ontwikkelen.

Vervolgens werden er bij een select aantal gebruikers **interviews** afgenomen om dieper in te gaan op hun wensen en behoeftes. Tijdens de interviews werden open vragen gesteld om hier zo veel mogelijk inzicht in te krijgen. De enquête en interview methoden zijn gebruikt om onderzoeksvraag 1, 3, 4 en 5 te beantwoorden. De vragen die werden gesteld in de enquêtes en interviews zorgden er voor dat de juiste informatie naar voren kwam om deze vragen te beantwoorden omdat deze specifiek ingingen op gewenste elementen en probleemoplossingen.

Na het analyseren van de resultaten van de enquêtes en interviews zijn er **persona's** voor beide type doelgroepen gecreëerd om beter in te kunnen zien wat hun behoeften zijn. Door een persona te creëren ga je niet alleen in op de behoeften rondom een app maar ook op de behoeften voor hun dagelijks leven en toekomstige doelen. Dit zorgt er voor dat er kan worden nagedacht hoe de app hen

kan helpen met deze bredere doelen en behoeften en heeft geholpen met het beantwoorden van onderzoeksvraag 1.

Naast de interviews en enquêtes werd er ook **available product analysis** uitgevoerd om te onderzoeken welke functionaliteiten bestaande monitoring apps bieden en welke aspecten van deze apps minder goed werken. Dit was vooral om onderzoeksvraag 2 te kunnen beantwoorden en om zelf een beter beeld te krijgen van bestaande oplossingen en verbeterpunten. Door deze analyse konden verbeterpunten worden geïdentificeerd voor de ontwikkeling van de nieuwe app. Hierbij werden verschillende bestaande apps beoordeeld en vergeleken met elkaar.

Na het verzamelen van informatie uit de voorgaande methodes zijn de resultaten genomen om **wireframes** te ontwerpen. De wireframes vormen als het ware het skelet van de applicatie en geven aan welke schermen gemaakt moeten worden en waar ongeveer belangrijke design elementen zoals knoppen en afbeeldingen moeten komen. Er werden wireframes gemaakt voor zowel het gebruikersgedeelte als het installateur gedeelte van de app.

Deze wireframes werden vervolgens gebruikt om een **design prototype** te maken. De schermen die werden aangegeven via de wireframes werden geïmplementeerd in dit design prototype om de schermen vervolgens af te maken met de gewenste design elementen, kleuren, iconen en afbeeldingen. Dit prototype werd gebruikt om **gebruikerstesten** uit te voeren om te onderzoeken of de functionaliteiten van de app aansloten bij de wensen en eisen van de gebruikers. Tijdens deze testen werden de gebruikers gevraagd om bepaalde taken uit te voeren in de app en werd hun feedback verzameld. Op basis van de feedback die uit deze testen werd verzameld konden verdere aanpassingen worden gemaakt aan het ontwerp van de app om deze nog beter te maken. De wireframes en de testen met het design prototype hebben verder gezorgd dat onderzoeksvragen 3 en 5 konden worden beantwoord omdat deze een stukje verder ingingen in de gewenste functionaliteiten van de app door ze door een realistische app te laten navigeren. Hierdoor kon er worden gekeken of er nog dingen misten of beter konden en of er al dingen waren in de app waar gebruikers juist tevreden mee waren.

Door het combineren van deze verschillende methoden werd een compleet beeld verkregen van de behoeften en wensen van zowel zonnepaneelgebruikers als installateurs en konden alle onderzoeksvragen worden beantwoord, waardoor de app zo veel mogelijk aan deze behoeften en wensen kon voldoen en optimaal kon functioneren.

Resultaten

De resultaten van alle onderzoeksmethoden zijn samengevat om tot een conclusie te komen voor het ontwerp van de app. **Deze samenvatting is gebaseerd op de uitgebreidere beschrijvingen die te vinden zijn in het bijgevoegde onderzoeksrapport en ik raad daarom ook aan het onderzoeksrapport er bij te houden voor gedetailleerdere uitleg en het beantwoorden van vragen die kunnen opkomen.**

Available product analysis

Voor de available product analysis zijn er 4 verschillende bestaande zonnepaneel-monitoring apps geanalyseerd, SolarEdge, SMA, Aurum en SEMS. Hierbij is er gekeken naar de functionele mogelijkheden, ontwerpelementen en methoden voor het weergeven van gegevens. Alle applicaties bieden monitoring- en beheermogelijkheden voor zonne-energiesystemen, gegevensanalyse tools en op afstand beheerfuncties. Daarnaast bieden de applicaties ook hun eigen unieke functionaliteiten. De ontwerpelementen zijn modern en gebruikersvriendelijk, met een vergelijkbaar kleurenpalet en geoptimaliseerd voor mobiele apparaten. De applicaties gebruiken ook een aantal vergelijkbare

methoden voor het weergeven van gegevens, waaronder lijngrafieken, staafdiagrammen en tabellen. De verschillen liggen voornamelijk in de branding en visuele identiteit van de app. Het lijkt erop dat bestaande producten zich veelal focussen op de functionaliteiten en gegevens die over het algemeen het belangrijkste zijn voor gebruikers om toegang tot te hebben. Er zijn echter ook verschillen in de nadruk op datavisualisatie versus gebruikersinvoer en -interactie. De keuze voor een benadering hangt vaak af van de doelgroep van de app.

Enquête

Uit de enquête voor gebruikers blijkt dat er een sterke vraag is naar apps waarmee eigenaren van zonnepanelen de prestaties van hun zonnepanelen kunnen monitoren. 92% van de zonnepaneeleigenaren gebruikt namelijk een app om hun zonnepanelen te kunnen monitoren. Als we kijken naar hun wensen wil 98% inzicht krijgen in de energieproductie en 65% in energieverbruik van hun zonnepanelen. Het integreren van weergegevens in de energieproductie wordt als interessant beschouwd met 64%, maar is nog niet erg gebruikelijk in bestaande applicaties. Gebruikers waarderen een gebruikersvriendelijke interface en real-time gegevens aangezien zij dit aangeven met respectievelijk 81 en 82 procent. Daarnaast vinden ze geautomatiseerde rapporten, voorspellende analyse en aanpasbare meldingen belangrijk, maar zijn extra functionaliteiten zoals delen via social media en gamification niet gewenst. De voorkeur voor een neutraal en simplistisch kleurenschema ondersteunt de voorkeur van gebruikers met 70%, voor eenvoud en overzichtelijkheid. Grafieken zijn bijzonder populair met 77% en gebruikers willen graag gemakkelijk kunnen navigeren door middel van een navigatiebalk en het kunnen scrollen en swipen door pagina's.

Uit de enquête voor installateurs blijkt dat installateurs de gegevens van zonnepanelen doorgaans jaarlijks of maandelijks controleren en het in real-time kunnen zien van gegevens zoals productie, stroom en uitvoer, en waarschuwingen is voor hen belangrijk aangezien dit wordt aangegeven met respectievelijk 74, 48, 83 en 91 procent. Installateurs zijn positief over het personaliseren van de app naar de huisstijl van hun bedrijf (48%) en geautomatiseerde rapportage (61%). Er is minder behoefte aan virtuele training (0% wil het zeker, 30% wil het misschien). Installateurs willen graag contact houden via e-mail en telefoon, maar staan open voor nieuwe communicatiemiddelen aangezien 61% aangeeft een ticket systeem ook wel te zien zitten. Ze geven de voorkeur aan een neutrale en simplistische look voor de app met 65%, met mogelijkheid tot fel en kleurrijk (22%). Installateurs zijn niet erg kieskeurig wat betreft navigatie, maar de voorkeur gaat uit naar scrollbare en swipebare pagina's (70%) in combinatie met tabbladen en een navigatiebalk (61% en 40%).

Interviews

Uit interviews met gebruikers van zonnepanelen monitoring apps blijkt dat de helft de huidige apps wel eens als verwarrend ervaren en de aanpassingsmogelijkheden en analysemogelijkheden beperkt zijn. Deze gebruikers missen vooral de mogelijkheid om gemakkelijk in contact te kunnen komen met hun installateur via de app. Voor 75% van de geïnterviewden is de primaire motivatie voor het gebruik van de app is om geld te besparen door de energieproductie te optimaliseren en om milieubewuster te zijn. Wel is ongeveer 40% van de gebruikers bezorgd over privacy en gegevensbeveiliging, maar vinden communicatie met de installateur en aanpasbare meldingen belangrijker om in een app te hebben. Een gebruikersvriendelijke interface is bij 100% van de gebruikers van groot belang, met eenvoud, visuele aantrekkelijkheid, overzichtelijkheid en toegankelijkheid als belangrijke kwaliteiten. Personalisering is niet noodzakelijk, maar kan wel een goede toevoeging zijn. De mogelijkheid om in contact te komen met de installateur en een gebruiksvriendelijke interface zijn belangrijker.

Ook installateurs van zonnepanelen zijn geïnterviewd over hun ervaringen met huidige monitoring software en hun wensen voor een potentiële nieuwe applicatie. De helft gaf aan dat klantcontact via telefoontjes en e-mails momenteel niet optimaal is en graag een manier zouden willen hebben om dit

contact te optimaliseren. De andere helft vond de huidige manier niet per se een probleem maar stonden ook open voor nieuwe manieren. Allemaal benadrukten ze het belang van een gebruikersvriendelijke interface, meldingen sturen en rapporten genereren. Ze noemden verschillende soorten meldingen die nuttig zouden zijn, zoals waarschuwingen, betalingen en afspraken, en legden uit dat automatische rapporten efficiënter, consistenter en nauwkeuriger zijn dan handmatig opgestelde rapporten. Ze benadrukten dat de functionaliteit, bruikbaarheid en betrouwbaarheid van de app altijd voorop moet komen te staan.

Tussentijdse conclusie

Zonnepanelen monitoring apps worden steeds populairder bij eigenaren van zonnepanelen. Het is daarom belangrijk om een app te ontwikkelen die aan de wensen van de klant voldoet. Om te achterhalen wat de wensen van potentiële gebruikers zijn rondom design, functionaliteiten en gebruikersinteractie zijn verschillende onderzoeksmethoden gebruikt, waaronder een enquête en interviews. Het onderzoek naar de wensen van gebruikers heeft een aantal belangrijke inzichten opgeleverd. Uit de enquête en interviews is gebleken dat de meeste gebruikers op zoek zijn naar een app die eenvoudig te gebruiken is, met een gebruikersvriendelijke interface en een overzichtelijke weergave van de belangrijkste informatie zoals energieverbruik, productie en verdiensten. Ook het kunnen bekijken van historische gegevens en voorspellende analyses werden als belangrijk gezien. Deze data zal worden weergegeven op een dashboard scherm in verschillende vormen zoals real-time data, grafieken en kalenderweergave.

Ook installateurs zien graag een gebruikersvriendelijke interface met een duidelijk overzicht van de belangrijkste data. Zij zien graag of er problemen zijn bij gebruikers en hoe de zonnepanelensystemen van hun klanten presteren. De prestaties willen ze kunnen inzien door actuele en historische gegevens te kunnen bekijken over energieproductie, verdiensten, temperatuur en batterijopslag.

Een ander belangrijk inzicht is dat veel gebruikers van zonnepanelen monitoring apps wel eens problemen ervaren met hun zonnepanelen maar dit vaak pas laat of per toeval ontdekken. Gebruikers missen daarom een mogelijkheid om contact te leggen met de installateur en het ontvangen van waarschuwingen over het systeem. Dit probleem kan worden opgelost door het toevoegen van belangrijke meldingen over het systeem en een mogelijkheid tot beter contact met de installateur. Ook het ontvangen van rapporten, zowel automatisch als gemaakt door de installateur kan helpen met beter inzicht krijgen in de prestaties van de zonnepanelen. Omdat meldingen een grote rol spelen in het ontwerp van deze app zal er daarom een apart scherm komen met meldingen over zowel het systeem als over het contact met de installateur en eventueel andere type meldingen.

Installateurs ervaren deze problemen ook, maar dan vanaf de andere kant. Zij krijgen vaak laat te zien of te horen dat er problemen zijn, maar elke dag checken of er problemen zijn is te tijdrovend. Dit probleem kan worden opgelost door op één scherm direct te kunnen zien of er klanten zijn met problemen en welke problemen dat dan zijn. Installateurs kunnen ook specifieke meldingen en rapporten sturen om de gebruiker te helpen met inzicht krijgen en beide partijen kunnen gebruik maken van een ticket systeem waarmee afspraken kunnen worden gemaakt over onder andere reparaties of onderhoud.

Naast de gewenste functionaliteiten en informatie is ook gekeken naar de design elementen van de app. Gebruikers hebben aangegeven dat zij graag een neutraal en simplistisch kleurenschema willen met consistente ontwerpelementen. Ook moet de app eenvoudig te navigeren zijn, door middel van een navigatiebalk en de mogelijkheid om te kunnen swipen tussen pagina’s.

Met deze inzichten in gedachten kan er een wireframe en design prototype worden ontwikkeld die alle gewenste informatie, functionaliteiten en ontwerpelementen bevatten. Het prototype zal tevens gebruik maken van de ISO-9421-11 normen. ISO 9241-11 is een internationale norm die richtlijnen geeft voor bruikbaarheid en ergonomische ontwerpbeginselen voor interactieve systemen, waaronder software applicaties. De norm richt zich op het ontwerp van gebruikersinterfaces om ervoor te zorgen dat deze effectief, efficiënt en bevredigend zijn voor gebruikers (ISO, 2018). Dit prototype kan vervolgens worden gebruikt om nogmaals gebruikers te betrekken bij het onderzoek en de app verder te optimaliseren.

Wireframes

Na het verzamelen van de onderzoeksresultaten en het vaststellen van de gewenste functionaliteiten voor de app zijn er wireframes gecreëerd die als basis zullen dienen voor het definitieve ontwerp van de app. Het doel van deze wireframes is om een visuele representatie te creëren van de indeling van de verschillende schermen en functionaliteiten.

Alle wireframes en hun uitleg zijn te vinden in het onderzoeksrapport in de bijlage en zijn ook te bekijken via deze linkjes:

Gebruiker:
https://preview.uxpin.com/f7c3fedf4d79b0c829b36fa742be4040075f824b#/pages/161894046

Installateur:
https://preview.uxpin.com/aa1f75ebfd317df842ec0151f647fa636836a89c#/pages/162363977

**Mocht u na het klikken van de link, links géén lijst met schermen zien, klik dan links bovenaan op 'Sitemap'.*

Figuur 1: Voorbeeld wireframes; homepagina.

Design prototype testen

Uit de gebruikerstesten op het prototype van de app kwamen enkele conclusies naar voren. Over het algemeen was de feedback van de testers positief, hoewel er wel enkele verbeterpunten werden aangedragen. Het inlogscherm werd goed ontvangen en als eenvoudig en goed ontworpen beschouwd, maar sommige testers suggereerden om een optie toe te voegen voor het terugkrijgen van een vergeten wachtwoord via e-mail of sms. Over het homescherm waren de meningen verdeeld, maar de meeste testers vonden het mooi en interessant. Enkele testers gaven aan liever een visualisatie van de energieproductie te zien in plaats van de temperatuur in de weersvoorspellingstool. Het waarschuwingsgedeelte werd over het algemeen goed ontvangen, omdat het vaak ontbreekt in de huidige app van de testers. De meldingen werden goed functionerend beschouwd, maar sommige testers vonden de kleuren van de bolletjes niet altijd duidelijk en zouden individuele meldingen willen kunnen verwijderen. Het ticketscherm werd goed begrepen, maar sommige testers vroegen zich af of het de beste optie was voor contact met het bedrijf en zouden meer ticketopties willen zien. Ze begrepen de keuze echter wel nadat er werd uitgelegd dat andere opties niet handig of te duur zouden zijn. Op de profielpagina waren testers enthousiast over de indeling en het inzien van rapporten, maar sommigen begrepen de instellingen voor meldingen niet en zouden meer uitleg en personalisatie-opties willen zien. De navigatiebalk werd over het algemeen positief beoordeeld, maar sommige testers zouden graag willen kunnen swipen tussen schermen in plaats van bovenaan het scherm te moeten klikken om naar een ander scherm te gaan. De app bevat de belangrijkste functies die gebruikers nodig hebben en is over het algemeen goed ontworpen, maar er zijn nog enkele verbeterpunten mogelijk.

Link naar prototype: https://xd.adobe.com/view/e4d1dbc3-2c5a-47a2-827d-9ef42b8f6b40-b182/?fullscreen&hints=off

Conclusie

Op basis van het uitgevoerde onderzoek naar de wensen en behoeften van gebruikers en installateurs van een zonnepaneel-monitoring app en de resultaten van de gebruikerstesten op het design prototype kan er geconcludeerd worden dat een nieuwe zonnepaneel-monitoring applicatie kan voldoen aan de behoeften van zowel zonnepaneelbezitters als installateurs. Bovendien is bij de ontwikkeling van de applicatie rekening gehouden met de principes die in de ISO 9241-11 normen voor bruikbaarheid en gebruikersgericht ontwerp zijn vastgelegd.

Uit het onderzoek is gebleken dat gebruikers vooral op zoek zijn naar een app die eenvoudig te gebruiken is, met een gebruikersvriendelijke interface en een overzichtelijke weergave van de belangrijkste informatie zoals energieverbruik, productie en verdiensten. Daarnaast willen zij historische gegevens en voorspellende analyses kunnen bekijken en problemen met de zonnepanelen direct kunnen melden aan de installateur. Installateurs willen op hun beurt een gebruikersvriendelijke interface met een duidelijk overzicht van de prestaties van de zonnepanelensystemen van hun klanten en mogelijkheden om problemen te detecteren en op te lossen. Door zich te houden aan de ISO 9241-11 normen garandeert de applicatie bruikbaarheid, effectieve presentatie van informatie, foutpreventie en -verwerking, en gebruikersgerichte ontwerpprincipes. De app voldoet aan de normen door een gebruikersgerichte benadering te hanteren bij het ontwerp. Dit betekent dat de app rekening houdt met de behoeften, vaardigheden en verwachtingen van de gebruikers. Het presenteert informatie op een duidelijke en begrijpelijke manier, voorkomt en verwerkt fouten, en biedt een gebruikersvriendelijke interface. Daarnaast biedt de app feedback en responsiviteit aan gebruikers.

Het design prototype is ontwikkeld op basis van deze inzichten en heeft positieve feedback ontvangen tijdens de gebruikerstesten. Het inlogscherm, de meldingen en het ticketscherm werden over het algemeen goed beoordeeld en de grafieken en kalenderweergave werden als leuke toevoegingen gezien. Sommige testers gaven echter aan dat de navigatiebalk iets te afleidend was en dat zij graag zouden willen kunnen swipen tussen schermen in plaats van altijd bovenaan het scherm te moeten klikken om naar een ander scherm te gaan. Uit de gebruikerstesten op het prototype van de applicatie zijn nog een aantal andere suggesties naar voren gekomen die kunnen bijdragen aan de verbetering van de applicatie. Zo hebben testers aangegeven dat zij graag individuele meldingen willen kunnen verwijderen en meer uitleg en opties voor personalisatie van meldingen zouden willen zien. Ook zou de navigatie van de applicatie verbeterd kunnen worden door het toevoegen van swipen tussen schermen en het verminderen van de afleiding van de navigatiebalk.

Om zich te onderscheiden van bestaande oplossingen kan de nieuwe zonnepaneel-monitoring applicatie zich richten op een neutraal en simplistisch kleurenschema met consistente ontwerpelementen en een eenvoudige navigatie. Daarnaast kunnen specifieke functionaliteiten worden toegevoegd die nog niet worden aangeboden door andere oplossingen, zoals de mogelijkheid om rapporten automatisch te ontvangen en te laten genereren door de installateur, het kunnen aanpassen van de stijl van de app naar de huisstijl van het bedrijf, of het bedachte ticket systeem om problemen eerder op te lossen. Door aan te sluiten bij de ISO 9241-11 normen, zorgt de applicatie voor naleving van de beste praktijken in de industrie en gebruikersgerichte ontwerpprincipes.

Al met al biedt de nieuwe zonnepaneel-monitoring applicatie een geavanceerde en gebruikersvriendelijke oplossing voor zowel zonnepaneelbezitters als installateurs en kan deze zich onderscheiden van bestaande oplossingen door de combinatie van functionaliteiten, design en gebruikersinteractie. De applicatie biedt een overzichtelijke en duidelijke weergave van belangrijke informatie zoals energieproductie, verdiensten, temperatuur en batterijopslag. Het dashboard bevat een verscheidenheid aan weergaven zoals real-time data, grafieken en kalender, die het voor

gebruikers gemakkelijk maken om hun prestaties te monitoren en historische gegevens in te zien. Ook kunnen gebruikers makkelijk meldingen ontvangen over problemen met hun zonnepanelen en contact opnemen met de installateur via het ticket systeem.

Om de applicatie nog verder te optimaliseren zou het aan te raden zijn om nogmaals gebruikerstesten uit te voeren op een verbeterde versie van de applicatie. Zo kan er worden nagegaan of de voorgestelde verbeteringen in de smaak vallen bij de gebruikers en of er nog verdere verbeteringen nodig zijn. Ook zou het aan te raden zijn om de applicatie op grotere schaal te testen, bijvoorbeeld door het uitvoeren van een bètatest.

Over het algemeen genomen biedt de nieuwe zonnepaneel-monitoring applicatie een geavanceerde en gebruikersvriendelijke oplossing die voldoet aan de wensen van zowel zonnepaneelbezitters als installateurs en onderscheidt het zich van bestaande oplossingen. Met de toevoeging van de voorgestelde verbeteringen kan de applicatie nog verder worden geoptimaliseerd en kan deze een belangrijke rol spelen in het monitoren en onderhouden van zonnepanelen.

Aanbevelingen

Om het onderzoek naar de wensen van gebruikers en de optimalisatie van de zonnepaneel-monitoring applicatie verder te verbeteren zijn er enkele aanbevelingen te doen. Ten eerste is het aan te raden om de steekproefomvang van het onderzoek naar de wensen van gebruikers te vergroten. Hoewel de huidige enquête en interviews enkele belangrijke inzichten hebben opgeleverd kan een grotere steekproefomvang dan 13 personen meer representatieve en gedetailleerde resultaten opleveren.

Daarnaast kan er nog dieper worden ingegaan op de behoeften van de installateurs, bijvoorbeeld door middel van gerichtere interviews of focusgroepen. Dit kan helpen om de specifieke behoeften van installateurs nog beter te begrijpen en om de functionaliteiten van de app hierop af te stemmen.

Om de gebruikerstesten verder te verbeteren kan er ook meer focus worden gelegd op het werven van testers die representatief zijn voor de doelgroep. Hoewel alle testers zonnepaneelbezitters waren, kwamen deze wel allemaal uit de zelfde omgeving. Aangezien door het hele land zonnepanelen worden gebruikt kan de locatie invloed hebben op de behoeftes. Dit kan helpen om een beter beeld te krijgen van de ervaringen en behoeften van alle soort gebruikers en om de app verder te optimaliseren op basis van deze feedback.

Daarnaast blijft het originele idee dat er verschillende apparaten kunnen worden gekoppeld in de app zodat alle statistieken te zien zijn. Er moet dus nagedacht worden over hoe verschillende apparaten kunnen worden getoond en of het nodig is om te kunnen switchen of dat alles in één keer wordt laten zien bijvoorbeeld.

Wanneer dat wordt gedaan is het ook nodig om verder onderzoek te doen naar de integratie van de app met andere slimme apparaten en slimme huizen. Dit zou de app nog aantrekkelijker kunnen maken voor gebruikers die hun huis op een slimme manier willen automatiseren.

Tot slot kan het nuttig zijn om de app te blijven testen en ontwikkelen in samenwerking met gebruikers en installateurs, bijvoorbeeld door middel van een gebruikerscommunity of een feedbacksysteem. Dit kan helpen om de app voortdurend te verbeteren en af te stemmen op de behoeften en wensen van de gebruikers en installateurs.

Door deze aanbevelingen in acht te nemen kan het onderzoek naar de zonnepaneel-monitoring applicatie verder worden verbeterd.

Proces

In dit hoofdstuk ga ik in op het proces van mijn afstuderen. Ik zal onder andere beschrijven hoe ik de verschillende fasen van mijn afstudeerperiode heb aangepakt, hoe ik met scrum heb gewerkt tijdens de ontwikkelfase en welke meetings ik heb gehad om de voortgang te bespreken en bij te sturen. Door in te gaan op deze aspecten van het proces hoop ik een goed beeld te geven van de aanpak die ik heb gehanteerd en de resultaten die ik heb behaald.

Planning

Aan het begin van mijn afstudeerperiode heb ik een uitgebreide planning gemaakt waarin alle schoolweken van mijn afstuderen zijn opgenomen en zijn onderverdeeld in verschillende fases. Voor elke fase heb ik de specifieke taken uitgewerkt waaraan ik zou gaan werken en de documenten waarop ik me zou gaan richten. De afstudeerperiode is opgedeelde in verschillende begin- en eindfases, met sprints tussendoor waarin ik de app heb ontwikkeld. In de eerste fase, die liep van week 1 tot en met 3, heb ik gewerkt aan het plan van aanpak, het analyseren van mijn opdracht en heb ik me vooral gericht op het kennismaken met het bedrijf, mijn collega's en hun werkwijze. In fase 2, van week 3 tot en met 6, heb ik het definitief plan van aanpak afgemaakt, vooronderzoek gedaan en ben ik begonnen met het nadenken over een ontwerp van de front- en backend van de app. Week 6 tot en met week 14 zijn opgedeeld in 5 sprints van 2 weken, 10 weken in totaal. Tijdens deze sprints ben ik bezig geweest met het ontwikkelen van de app en stond de laatste sprint in het teken van het afmaken van de documentatie. Fase 4 en fase 5 stonden in het teken van het afronden van alle documenten en het voorbereiden op de verdediging.

Week 1 – 3: Plan van aanpak, kennismaking, analyse (*fase 1*)
Week 3 – 6: Definitief plan van aanpak, vooronderzoek, ontwerp (*fase 2*)
Week 6 – 14: Gebruikersonderzoeken, ontwikkelen, documenteren (*fase 3*)
Week 15 – 16:
Afronden, feedback verwerken (*fase 4*)
Week 17 - 19:
Voorbereiding verdediging, verdediging (*fase 5*)

Begin – Eind producten:

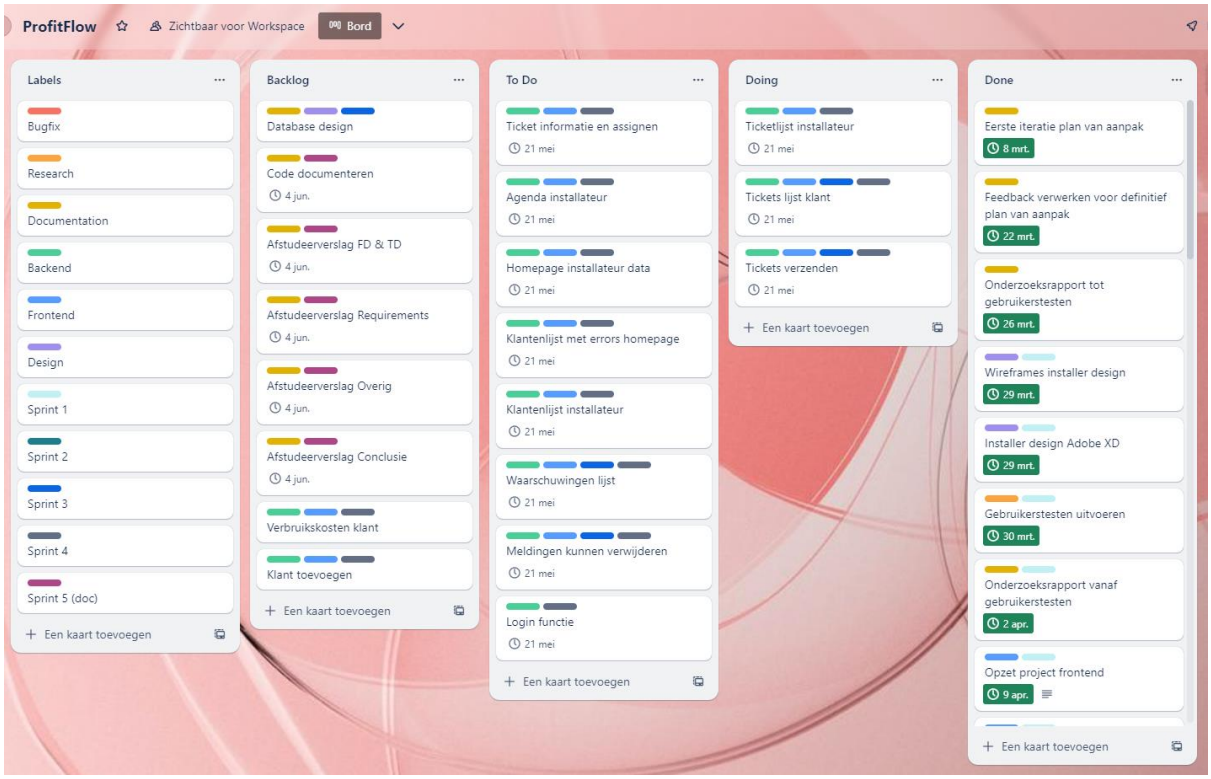
- Plan van aanpak
- Onderzoeksrapport
- Functioneel ontwerp
- Fase/sprint periode
- Technisch ontwerp
- Afstudeerverslag
- Zelfreflectie
- Verdediging

Week	Vanaf	Tot/met	Fase 1	Fase 2	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Fase 4	Fase 5
1	13 feb	17 feb									
2	20 feb	24 feb									
VAK	**VAK**	**VAK**	**VAK**	**VAK**	**VAK**	**VAK**	**VAK**	**VAK**	**VAK**	**VAK**	**VAK**
3	6 mar	10 mar									
4	13 mar	17 mar									
5	20 mar	24 mar									
6	27 mar	31 mar									
7	3 apr	7 apr									
8	10 apr	14 apr									
9	17 apr	21 apr									
10	24 apr	28 apr									
VAK	**VAK**	**VAK**	**VAK**	**VAK**	**VAK**	**VAK**	**VAK**	**VAK**	**VAK**	**VAK**	**VAK**
11	8 mei	12 mei									
12	15 mei	19 mei									
13	22 mei	26 mei									
14	29 mei	2 jun									
15	5 jun	9 jun									
16	12 jun	16 jun									
17	19 jun	23 jun									
18	26 jun	30 jun									
19	3 jul	7 jul									

Figuur 2: Strokenplanning voor afstudeerperiode gemaakt in week 1.

Scrum

Ondanks dat ik zelfstandig aan mijn opdracht werkte heb ik er toch voor gekozen om met scrum te werken. Scrum is gericht op het werken in korte sprints waarin er wordt gestreefd om de geplande taken af te maken. Ik heb er voor gekozen om in sprints van 2 weken te werken zodat ik genoeg tijd zou hebben om taken in te plannen en af te maken. Aangezien ik 10 weken ontwikkeltijd had, had ik 5 sprints in totaal. Daarnaast deed ik aan het begin en einde van elke sprint een sprint planning en sprint retrospective met mezelf. Ondanks dat ik niet in een team heb gewerkt tijdens mijn afstuderen vond ik het wel een fijne manier van werken voor mezelf. Met scrum werken geeft een mooi en duidelijk overzicht van de taken die gedaan moeten worden en wanneer deze gedaan moeten worden. Voor mezelf is het ook vaak een duwtje in de rug om bepaalde taken af te maken. Wanneer ik geen overzichtelijk planning maak van wanneer ik bepaalde taken wil doen raak ik vaak de weg en het overzicht kwijt en weet ik niet goed wanneer ik waarmee moet beginnen. Ook is het voor mij vaak van groot belang om terug te kijken om de voorgaande sprint. Ik vind het fijn om te weten wat ik goed heb gedaan en wat ik dus door kan zetten, en waar ik op moet verbeteren. Een retrospective is voor mij vaak een eye opener en een duidelijke afsluiter van een periode. Een sprint planning is voor mij dan weer een nieuw begin en een frisse start. Om mijzelf die overzichtelijke takenplanning te geven heb ik gebruik gemaakt van een Trello board. Op dit board staan alle taken in de vorm van tickets, met specifieke labels en uiterlijke einddatum. Deze tickets kon ik dan verslepen wanneer ik ze die sprint wou doen, wanneer ik er mee bezig was of wanneer ze klaar waren. Dit gaf mij elke sprint een goed overzicht van wat mij nog te doen stond en of ik nog op schema liep. In figuur 3 is te zien hoe het Trello board er bijvoorbeeld uitzag halverwege sprint 4.



Figuur 3: Trello board voor afstudeerperiode. Getoonde sprint is sprint 4.

Link naar Trello board:
https://trello.com/invite/b/AnRDH6dl/ATTI4760bb5b4a3290bf39cd5efe9107a369ECAA7111/profitflow

Aan het einde van elke sprint deed ik een sprint retrospective waarbij ik keek naar wat goed ging, wat beter kon en wat ik zou meenemen naar de volgende sprint. Hierdoor kreeg ik voor mezelf een goed overzicht over wat ik goed deed en wat ik beter kon doen. Ook gaf dit me meer motivatie om daar daadwerkelijk aan te werken omdat ik het aan mezelf kon uiten en kon opschrijven wat er dwars zat. Hier onder een voorbeeld van één van mijn retrospectives:

Sprint 3: 24-04-2023 - 07-05-2023

Samenvatting:

Deze sprint stond in het teken van het implementeren van de functionaliteiten van het klant gedeelte van de app. Ik zou de SolarEdge API gebruiken om real-time data te verkrijgen die ik in de app kon stoppen. Uiteindelijk is het verkrijgen van de benodigde API key erg lang uitgesteld waardoor er pas laat aan gewerkt kon worden. Het doel van mijn afstuderen is daardoor ook wat veranderd, aangezien er waarschijnlijk niet meer genoeg tijd is om meerdere API's te integreren. De focus ligt nu op het implementeren van alle functionaliteiten met één API. Uiteindelijk zijn er nog wel veel klant functionaliteiten geïmplementeerd na het verkrijgen van de key, en is er in de tussentijd gewerkt aan het afstudeerverslag, waardoor alle hoofdstukken die geschreven konden worden vóór het afmaken van de app nu al af zijn.

Wat ging goed:

- Ondanks vertraging toch nog veel kunnen implementeren, mede door doorzettingsvermogen om in het weekend te werken
- Niet stil gezeten, gewerkt aan afstudeerverslag
- Voor het eerst gewerkt met een externe API

Wat kon beter:

- Ik had strenger mogen zijn om eerder die API key te krijgen
- Had nog iets meer research mogen doen naar werken met Hasura en externe API's
- Weather data had al eerder geïmplementeerd kunnen worden, had niet hoeven wachten op de API key

Wat neem ik mee naar volgende sprint:

- Eerder communiceren met verantwoordelijken over belangrijke zaken zoals het verkrijgen van die API key en het uiten van mijn zorgen
- Me niet laten tegenhouden om door te werken ook al kom ik tegen obstakels en vertraging aan
- Doorzettingsvermogen om alsnog terug te komen op schema en mijn doel af te maken

Meetings

Tijdens mijn afstuderen heb ik me voorgenomen om een aantal verschillende vaste meetings op te stellen. Hier onder vallen ook meetings gerelateerd aan mijn scrumproces, wat ik vaak zelfstandig deed.

Wat	Wie	Wanneer
Sprint retrospective	Afstudeerder en eventueel bedrijfsbegeleider	Elke 2 weken aan het einde van de sprint
OpusFlow 2 weekly	Alle werknemers	Elke 2 weken op de laatste vrijdag
Voortgangsgesprek	Afstudeerder en schoolbegeleider	Eens per maand
Sprint planning	Afstudeerder en eventueel begeleider	Elke 2 weken aan het begin van de sprint

Tabel 1: Terugkerende meetings afstudeerperiode.

Mijn sprints duurde 2 weken, waarbij ik aan het begin van een sprint een sprint planning deed en aan het einde van een sprint een sprint retrospective. Bij de sprint planning vulde ik de backlog van mijn Trello board aan en bedacht ik me welke taken ik die sprint wou gaan doen. Bij de sprint retrospective keek ik terug op de sprint en bedacht en noteerde ik wat er goed ging, wat er beter kon en wat ik meenam naar de volgende sprint. Elke maand had ik een voortgangsgesprek met mijn begeleiders over waar ik op dat moment was en of het nog de goeie kant op ging en een aantal keer was er een OpusFlow 2 weekly waarin de voortgang van het bedrijf werd besproken en waar ik ook mijn eigen voortgang kon laten zien.

Kwaliteit

In dit hoofdstuk beschrijf ik wat ik heb ondernomen om er voor te zorgen dat ik de kwaliteit van mijn project heb kunnen waarborgen.

GitHub

Tijdens het ontwikkelen van de applicatie heb ik gebruik gemaakt van GitHub, een online platform voor versiebeheer en samenwerking bij softwareontwikkeling. Door de applicatie code regelmatig te commiten en te pushen naar GitHub heb ik een gedetailleerde geschiedenis van de ontwikkeling van de applicatie kunnen bijhouden en eventuele fouten in de code snel kunnen opsporen en corrigeren. Door gebruik te maken van branches heb ik er ook voor kunnen zorgen dat verschillende taken apart van elkaar konden worden gedaan zonder dat ze elkaar in de weg zaten. Zo kon ik bijvoorbeeld aan zowel de loginfunctie van de app werken terwijl ik op een andere branch nog aan de opmaak van (onder andere) de login pagina werkte. Om het overzichtelijk voor mezelf te houden gebruikte ik dan branches om beter te kunnen zien waar ik mee bezig was en wat al af was.

Definition of Done

Bij het aanmaken van een ticket op mijn Trello board heb ik elk ticket een Definition of Done (DoD) gegeven. Hiermee geef ik aan wanneer ik vind dat de taak voldaan is en naar de 'Done' rij mag worden geplaatst. Het heeft er ook voor gezorgd dat de ontwikkeling van de applicatie gestructureerd verliep en dat er geen taken werden afgerond voordat deze aan alle vereisten voldeden.

Code Quality

Het handhaven van een hoge kwaliteit van de code gedurende het hele ontwikkelingsproces is belangrijk voor het waarborgen van de leesbaarheid, onderhoudbaarheid en betrouwbaarheid van het project. Door een consistente codeerstijl te volgen, codedocumentatie toe te passen en de nadruk te leggen op code modulariteit en herbruikbaarheid, streefde ik ernaar een project te leveren die zo veel mogelijk voldoet aan de kwaliteitsnormen die vallen binnen de scope van mijn project.

Consistente Coderingsstijl: Een uniforme coderingsstijl is gebruikt in de hele codebasis. Door me te houden aan gevestigde praktijken, zoals consistente inkeping, naamgevingsconventies en opmaak, werd de code meer georganiseerd en gemakkelijker te begrijpen.

Codedocumentatie: Naast het schrijven van schone code speelt codedocumentatie een ook een rol bij het verbeteren van de kwaliteit van de code. Elke functie, klasse en belangrijk codeblok is gedocumenteerd om duidelijke uitleg te geven over hun doel, ingangen en uitgangen. Deze documentatie vergemakkelijkt het begrip van code voor ontwikkelaars die in de toekomst aan het project kunnen werken, waardoor snellere onboarding en soepelere samenwerking mogelijk zijn.

Modulaire project structuur: Bij het ontwerpen van de codebase is geprobeerd een modulaire aanpak toe te passen. De functionaliteit is onderverdeeld in kleinere, zelfstandige modules, waarbij elke module een specifieke verantwoordelijkheid heeft. Dit modulaire ontwerp maakte een betere codeorganisatie en verbeterde herbruikbaarheid mogelijk.

```
// This function sets the right icon according to the weather conditions and calculates the energy boost for each day of the week based on energy levels.
async function calculateBoost(){
  // Get the weather data for the client's city using the getWeatherData function and await the result.
  const city = await getClientCity()

  const energyArray = await getWeatherData(city)

  // Create an array of weekdays to display in the UI.
  const weekdays = ["Ma", "Di", "Wo", "Do", "Vr", "Za", "Zo"]

  // Create an object to map weather conditions to corresponding icons to display in the UI.
  const icons = {
    "snow": 'snowy',
    "rain": 'rainy',
    "fog": 'foggy',
    "wind": 'cloudy',
    "cloudy": 'cloudy',
    "partly-cloudy-day": 'partSunny',
    "partly-cloudy-night": 'partNighty',
    "clear-day": 'sunny',
    "clear-night": 'nighty'
  }

  // Set the energy thresholds for low, medium, and high energy levels.
  const mediumTreshold = 15
  const highTreshold = 25

  // Create an array to store the energy boost for each day of the week.
  const dayArray = []
  for (let i = 0; i < 7; i++) {
    let boost = 'low'
    if(energyArray[i].energy > mediumTreshold){
      boost = 'medium'
    } if(energyArray[i].energy > highTreshold){
      boost = 'high'
    }
    dayArray.push({day: weekdays[energyArray[i].day.getDay()], weather: icons[energyArray[i].condition], boost: boost})
  }

  // Return the array of data for each day of the week to display in the UI: day name, weather condition, boost amount.
  return dayArray
}
```

Figuur 4: Voorbeeld code; weatherboost calculate function.

Requirements

Dit hoofdstuk richt zich op het vaststellen en documenteren van de requirements van de applicatie. Requirements zijn belangrijke specificaties en criteria die bepalen hoe de app moet functioneren en welke prestaties en kwaliteitsattributen moeten worden bereikt. De requirements worden ingedeeld in twee categorieën: functionele en non-functionele requirements en zijn zorgvuldig opgesteld na onderzoek en overleg met het bedrijf. Functionele requirements beschrijven de specifieke functies, mogelijkheden en gedragingen die de app moet hebben. Non-functionele requirements richten zich daarentegen op de kwaliteit, prestatie, gebruiksgemak en andere aspecten die niet direct gerelateerd zijn aan specifieke functionaliteiten.

Functionele requirements:

Nr.	Beschrijving	ISO 25010	MoSCoW
F1	Een gebruiker moet kunnen inloggen	Functionele geschiktheid	Must
F2	Een gebruiker moet zijn omvormer kunnen toevoegen	Functionele geschiktheid	Should
F3	Een gebruiker moet zijn omvormer(s) kunnen inzien	Functionele geschiktheid	Must
F4	Een gebruiker moet per omvormer details in kunnen zien (verbruik, fouten, etc.)	Functionele geschiktheid	Must
F5	Een gebruiker moet historische details kunnen zien	Functionele geschiktheid	Must
F6	Een gebruiker moet weergegevens kunnen inzien via de app	Functionele geschiktheid	Must
F7	Een gebruiker kan tickets aanmaken voor installateurs om te kunnen zien	Uitwisselbaarheid	Must
F8	Een gebruiker kan meldingen ontvangen (waarschuwing, info, ticket)	Bruikbaarheid	Must
F9	Een gebruiker kan meldingen verwijderen	Bruikbaarheid	Must
F10	Een installateur kan meldingen versturen	Functionele geschiktheid	Must
F11	Een installateur kan tickets inzien van gebruikers	Uitwisselbaarheid	Must
F12	Een installateur kan tickets behandelen van gebruikers	Uitwisselbaarheid	Should
F13	Een installateur kan afspraken in zijn agenda in de app plaatsen	Functionele geschiktheid	Could
F14	Een installateur moet omvormers kunnen toevoegen aan klanten	Uitwisselbaarheid	Must
F15	Een installateur moet klanten kunnen aanmaken	Beveiligbaarheid	Must
F16	Een installateur moet alle geïnstalleerde omvormers kunnen inzien	Functionele geschiktheid	Must

F17	Een installateur moet per omvormer details in kunnen zien (verbruik, fouten, etc.)	Functionele geschiktheid	Must
F18	Een administrator moet gebruikers kunnen aamaken	Beveiligbaarheid	Could
F19	Een administrator moet gebruikers kunnen verwijderen	Beveiligbaarheid	Could
F20	Een administrator moet wachtwoorden kunnen resetten	Beveiligbaarheid	Could
F21	Een administrator moet wachtwoorden kunnen verwijderen	Beveiligbaarheid	Could
F22	Een administrator moet installateurs kunnen aanmaken	Beveiligbaarheid	Could
F23	Een administrator moet installateurs kunnen verwijderen/aanpassen	Beveiligbaarheid	Could

Tabel 2: Functionele requirements EnergyFlow.

Non-functionele requirements:

Nr.	**Beschrijving**	**ISO 25010**	**MoSCoW**
NF1	Wanneer er iets fout gaat aan de client side krijgt de gebruiker eenduidige instructies hoe verder te handelen	Bruikbaarheid	Should
NF2	De app moet een dashboard-achtig design hebben voor minimaal één scherm	Bruikbaarheid	Must
NF3	De backend wordt gemaakt met Hasura	Onderhoudbaarheid	Must
NF4	De frontend wordt gemaakt met React Native	Onderhoudbaarheid	Must
NF5	De applicatie wordt getest d.m.v. PlayWright	Betrouwbaarheid	Must
NF6	De applicatie moet publiek worden gehost	Overdraagbaarheid	Could
NF7	De applicatie wordt gemaakt met focus op mobile first	Bruikbaarheid	Must
NF8	De applicatie maakt gebruik van een Postgres database	Onderhoudbaarheid	Must
NF9	Minimaal één externe API is geïntegreerd in de app	Uitwisselbaarheid	Must

Tabel 3: Non-functionele requirements EnergyFlow.

Product

Functioneel ontwerp

De functionele documentatie beschrijft de functionaliteit en werking van de applicatie. Het biedt een gedetailleerd overzicht van de mogelijkheden en functies van de applicatie, de stappen die nodig zijn om deze te gebruiken, en de processen die plaatsvinden binnen de applicatie. Deze functionele documentatie bestaat uit verschillende onderdelen, waaronder sequence wireframes, verschillende diagrammen en een handleiding.

Handleiding

Wanneer de app is geïnstalleerd en opgestart is als eerste het keuzescherm zien. Het keuzescherm bevat twee knoppen: klant en installateur. De knoppen leiden naar het bijbehorende inlogscherm.

Als klant is in te loggen met de volgende gegevens:

- **Email:** joey@opusflow.io
- **Wachtwoord:** 123

Als installateur is in te loggen met de volgende gegevens:

- **Email:** rosanne@opusflow.io
- **Wachtwoord:** 123

Klant:

Wanneer er succesvol is ingelogd, komt de gebruiker op het homescherm van de app terecht. Hier heeft de gebruiker toegang tot diverse gegevens met betrekking tot uw zonnepaneelsysteem. Het homescherm biedt een overzicht van real-time informatie, zoals uw actuele energieverbruik, energieproductie en opbrengsten. Bovendien kan de gebruiker deze gegevens ook bekijken in een historisch perspectief. Met behulp van de productie- en consumptiegrafieken heeft de gebruiker de mogelijkheid om gedetailleerde informatie te bekijken voor specifieke tijdsperiodes, zoals per dag, week of jaar. Op de homepagina is ook een handige agenda-tool waarmee de gegevens per dag bekeken kunnen worden te vinden. Daarnaast heeft de gebruiker toegang tot andere relevante gegevens, zoals het aantal systeemwaarschuwingen en de weersvoorspelling. De pijltjes op het scherm geven een indicatie van de verwachte energieproductie voor die specifieke dag. Tot slot wordt op de homepagina ook aangegeven hoeveel CO2-uitstoot er is bespaard dankzij de zonnepanelen en hoeveel bomen er geplant kunnen worden met die besparing. Op deze manier biedt de app een uitgebreid overzicht van belangrijke gegevens en functionaliteiten die verband houden met het zonnepaneelsysteem.

Als naar links wordt geveegd of op het bel-icoontje in de navigatiebalk wordt getikt, verschijnt het meldingenscherm. Op deze pagina worden alle ontvangen meldingen van het account weergegeven. Deze meldingen kunnen betrekking hebben op waarschuwingen, algemene informatie of de status van tickets. Elke melding wordt weergegeven met een prullenbak-icoontje. Door hierop te klikken, wordt de specifieke melding geselecteerd. Bovenaan de lijst bevindt zich een selectievakje waarmee alle meldingen tegelijk geselecteerd kunnen worden. Zodra een of meerdere meldingen zijn geselecteerd, verschijnt er een knop waarmee alle geselecteerde meldingen verwijderd kunnen worden. Op deze manier kan de gebruiker eenvoudig meldingen beheren en opruimen.

Als er nogmaals naar links wordt geveegd of op het sleutel-icoontje in de navigatiebalk wordt getikt, verschijnt het ticketscherm. Op deze pagina worden alle aangemaakte tickets van het account weergegeven. Tickets kunnen worden aangemaakt wanneer er contact moet worden gezocht met een installateur voor reparatie, onderhoud of informatie. Bovenaan de lijst met tickets bevindt zich een knop waarmee een nieuw ticket aangemaakt kan worden. Deze knop leidt de gebruiker naar een scherm waar alle benodigde informatie voor het nieuwe ticket kunt ingevuld kan worden. De gebruiker kan het type ticket kiezen, een bericht schrijven en een gewenste datum doorgeven. Zodra het ticket wordt verzonden, wordt het toegevoegd aan de ticketlijst met de status 'Open'. Installateurs ontvangt het ticket en kunnen het behandelen. De status van het ticket in de lijst zal worden bijgewerkt zodra het ticket wordt geaccepteerd of afgewezen. De gebruiker ontvangt ook een melding met betrekking tot de statusverandering van het ticket. Op deze manier biedt de ticketpagina een overzichtelijke weergave van alle aangemaakte tickets en houdt de gebruiker gemakkelijk contact met de installateur.

Als er nogmaals naar links wordt geveegd of op het persoon-icoontje in de navigatiebalk wordt getikt, verschijnt het profielscherm. Op deze pagina heeft de gebruiker toegang tot persoonlijke gegevens en knoppen naar andere persoonlijke secties, zoals "Mijn apparaten", "Mijn rapporten" en "Instellingen". Bij "Mijn apparaten" kunt de gebruiker zien welke apparaten geregistreerd zijn in de app. Momenteel kunnen dit zonnepaneelsystemen zijn, maar in de toekomst kunnen ook andere apparaten zoals warmtepompen en laadpalen worden toegevoegd. Mijn rapporten" toont alle gegenereerde rapporten die de gebruiker maandelijks ontvangt. Hier kan de gebruiker de historische rapporten bekijken en relevante informatie over de energieproductie en consumptie inzien. Bij "Instellingen" heeft de gebruiker de mogelijkheid om verschillende persoonlijke instellingen aan te passen, zoals de taal van de app, meldingsvoorkeuren en persoonlijke gegevens. Ten slotte kan de gebruiker via de profielpagina uitloggen door op de "Uitloggen" knop onderaan het scherm te klikken.

Let op: Op dit moment zijn de knoppen voor "Mijn apparaten", "Mijn rapporten" en "Instellingen" nog niet functioneel.

Installateur:

Als een installateur is ingelogd, krijgt hij toegang tot het homescherm. Op deze pagina worden verschillende gegevens weergegeven, zoals de datum, het aantal klanten en het aantal openstaande tickets. Daarnaast kan de installateur op de homepagina zien of er klanten zijn die problemen ervaren met hun zonnepaneelsysteem. Indien een klant een waarschuwing ontvangt met betrekking tot hun zonnepaneelsysteem, wordt deze waarschuwing ook weergegeven op de homapagina van de installateur. Op deze manier biedt de homapagina een overzicht van relevante informatie en helpt het de installateur om op de hoogte te blijven van eventuele problemen bij klanten.

Als naar links word geveegd of op het personen-icoontje in de navigatiebalk wordt getikt, komt de installateur op het klantenscherm. Hier is een lijst van alle geregistreerde klanten van het installateursbedrijf te zien. De installateur kan zoeken op naam of klantnummer, en door op een klant te klikken, krijgt hij toegang tot de gedetailleerde informatie van die klant. Op de detailpagina van een klant worden belangrijke real-time gegevens weergegeven, zoals energieproductie, energieverbruik, spanning en temperatuur. Hij kan ook de historische energieproductie van de klant bekijken in een grafiek en zien of er waarschuwingen zijn voor die klant. Boven de lijst van klanten op de klantenpagina bevindt zich een knop om klanten toe te voegen. Als de installateur hierop klikt, wordt hij naar een scherm geleid waar hij de gegevens van een nieuwe klant kunt invullen, zoals naam, omvormer nummer, e-mail en telefoon. Zodra deze gegevens zijn ingevuld, kan hij op de "Toevoegen" knop klikken om de klant aan de lijst toe te voegen.

Op deze manier biedt de klantenpagina een overzicht van alle geregistreerde klanten en kan de installateur gemakkelijk belangrijke real-time gegevens en waarschuwingen voor elke individuele klant bekijken. Bovendien kan hij nieuwe klanten toevoegen aan de lijst voor verdere beheerdoeleinden.

Wanneer vanuit de klantenpagina naar links wordt geveegd of op het sleutel-icoontje in de navigatiebalk wordt geklikt, komt de installateur op het ticketscherm terecht. Op deze pagina kunt u alle openstaande tickets van klanten bekijken. Wanneer een klant een nieuw ticket aanmaakt, wordt dit toegevoegd aan de lijst en kunt u als installateur de details van het ticket bekijken door erop te klikken. Binnen het ticket kunt u zien wie het ticket heeft aangemaakt, wat het specifieke probleem is en welke gewenste datum er is opgegeven. Indien nodig, kunt u als installateur de datum aanpassen. Bovendien heeft u de mogelijkheid om uzelf of een andere geregistreerde installateur aan het ticket toe te wijzen. Wanneer het ticket wordt goedgekeurd, ontvangt de toegewezen installateur een afspraak in zijn of haar agenda. Het is ook mogelijk om een ticket af te wijzen als het niet binnen uw verantwoordelijkheidsgebied valt. Wanneer een ticket wordt goedgekeurd of afgewezen, zal de status van het ticket veranderen en ontvangt de betreffende klant een melding met betrekking tot het ticket en de genomen beslissing. Op deze manier biedt de ticketpagina een overzicht van alle openstaande tickets van klanten en kunt u als installateur efficiënt de details van elk ticket beheren, toewijzen en behandelen.

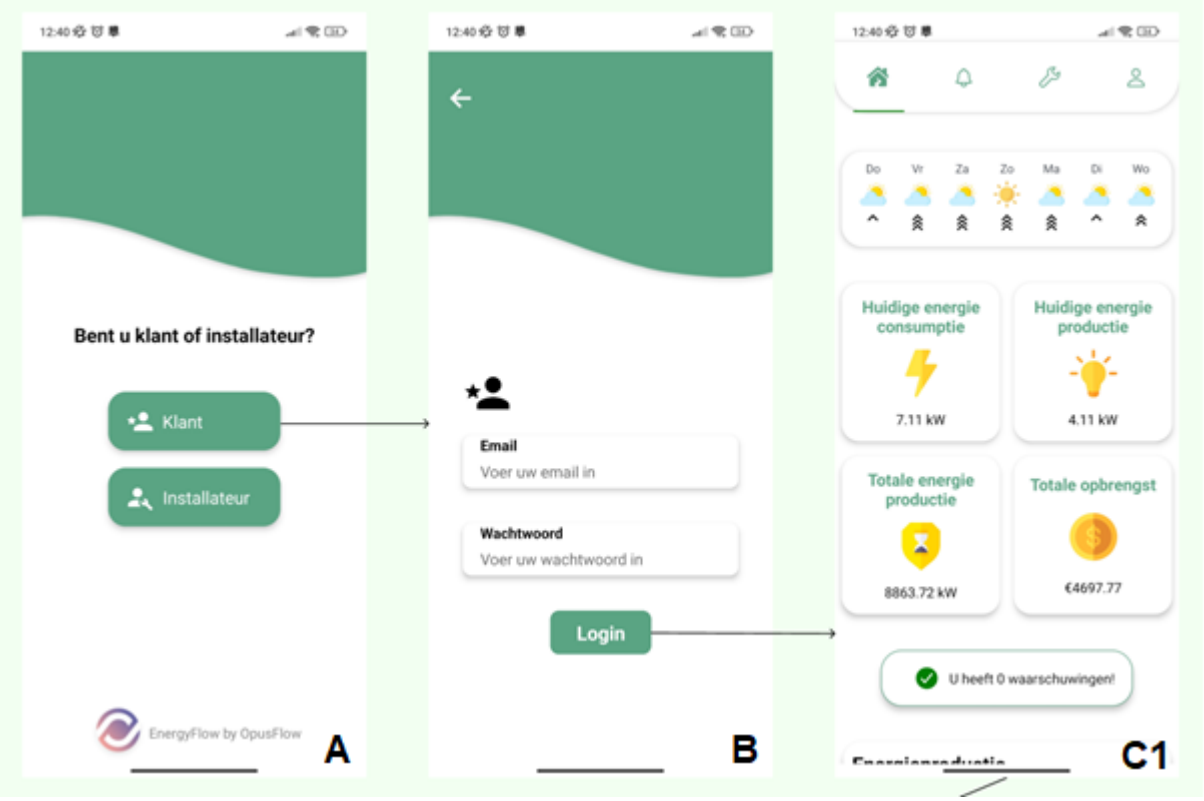
Wanneer er nogmaals naar links wordt geveegd of op het persoon-icoontje in de navigatiebalk tikt, komt de installateur op het profielscherm terecht. Hier heeft hij toegang tot zijn afsprakenagenda. In deze agenda kan hij alle afspraken per dag bekijken die zijn ingepland voor hem. Naast de afsprakenagenda, kan de installateur ook via de homepagina naar de instellingen gaan om persoonlijke voorkeuren aan te passen. Daarnaast is er een uitlogknop beschikbaar onderaan het scherm om uit te loggen uit het account.

Let op: Op dit moment werken de knoppen voor instellingen en afspraken nog niet naar behoren.

Sequence wireframes

In deze functionele documentatie wil ik graag de flow binnen de app visualiseren aan de hand van screenshots waarbij pijlen worden gebruikt om de gebruikersstroom te verduidelijken. Door middel van deze screenshots wil ik een duidelijk beeld geven van hoe gebruikers door de verschillende schermen en functies van de app kunnen navigeren.

Het geheel is te vinden in deze Figma file:
https://www.figma.com/file/Dvlj9ATOo3JJvZhNAgY1t6/EnergyFlow?type=design&node-id=0-1&t=AVMEyoBD21iIzmFE-0



Figuur 5: Keuzescherm, klant inlogscherm, klant homescherm van EnergyFlow.

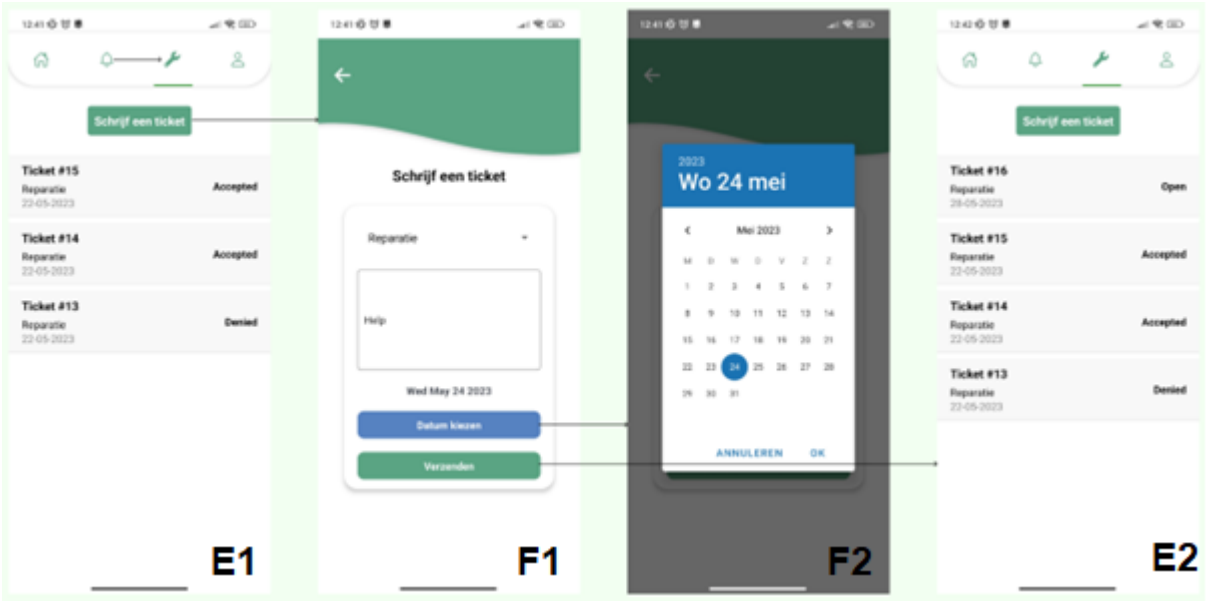
Het keuzescherm, inlogscherm en een deel van het homescherm. De gebruiker kan via het keuzescherm (A) kiezen welke rol hij heeft en welk deel van de app hij dus kan bekijken na het inloggen. In dit geval kan hij via het inlogscherm (B) inloggen als klant en de gegevens van zijn persoonlijke zonnepaneelsysteem inzien op het homescherm (C1). Wanneer er naar beneden wordt gescrold in het homeschermkomt de gebruiker in de schermaanzichten van figuur 6.

Figuur 6: De grafieken op het klant homescherm van EnergyFlow in verschillende stadia.

Het onderste gedeelte van het klant homescherm. Te zien is dat de grafieken een menu bevatten waar de tijdsperiode mee kan worden aangepast. Wanneer er vanuit C2/C3 in figuur 6 nog meer naar beneden wordt gescrold zijn ook zijn de agenda tool en de Co2 tool te zien (C4) waarbij er in de agenda op een bepaalde dag is geklikt om de statistieken van die dag te bekijken (C5).

Figuur 7: Klant notificatiescherm van EnergyFlow.

Wanneer er links wordt geswiped vanuit het homescherm is het klant notificatiescherm van de app te zien. Te zien is dat de klant meldingen kan inzien en verwijderen, zowel individueel (D2) als allemaal tegelijk (D3). Wanneer er op het prullenbak icoontje wordt geklikt wordt die individuele melding geselecteerd maar alle meldingen kunnen ook tegelijk worden geselecteerd wanneer op de checkbox bovenaan de lijst wordt geklikt. De verwijderknop verwijderd alle geselecteerde meldingen (D4).

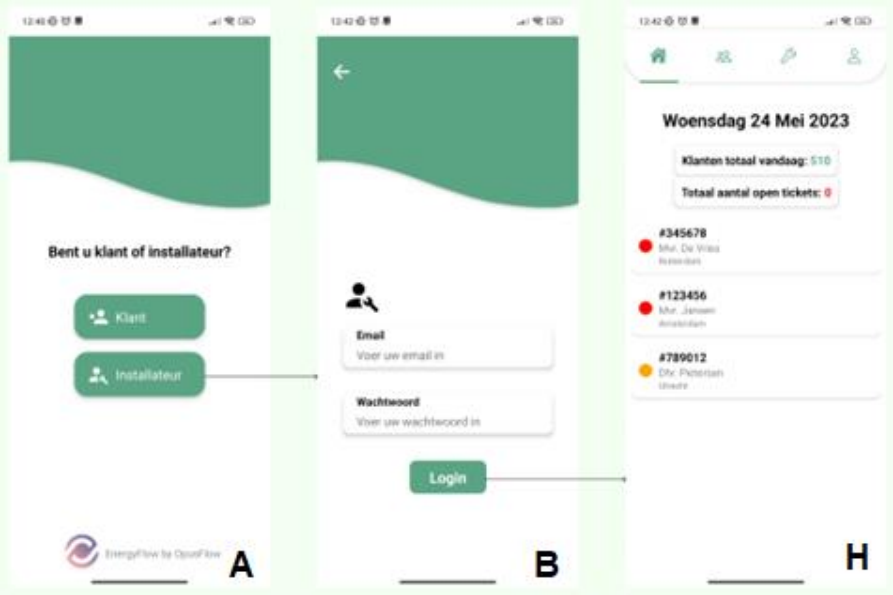


Figuur 8: Het klant ticketscherm en ticket aanmaken scherm van EnergyFlow.

Wanneer naar links wordt geswiped vanuit het notificatiescherm is het klant ticketscherm van de app te zien (E1). Te zien is dat de klant een lijst van al zijn tickets met hun status kan zien. Wanneer op de bovenste knop wordt gedrukt kan de klant een nieuw ticket schrijven met data zoals type afspraak, informatie en gewenste datum (F1). De gewenste datum kan worden aangepast door op de datum knop te klikken (F2). Wanneer het nieuwe ticket wordt verzonden komt deze direct met een 'Open' status in de lijst te staan (E2).

Figuur 9: Klant profielscherm van EnergyFlow.

Wanneer vanaf het ticketscherm naar links wordt geswiped is het profielscherm te zien (G1). De klant kan hier een aantal persoonlijke gegevens inzien en kan vanuit de profielpagina navigeren naar 'Mijn apparaten', 'Mijn rapporten' en 'Instellingen'. De klant kan ook uitloggen via de 'Uitloggen' knop. De begroeting bovenaan het scherm veranderd per dagdeel (morgen, middag, avond, nacht).



Figuur 10: Keuzescherm, installateur loginscherm en installateur homescherm van EnergyFlow.

Het keuzescherm, inlogscherm en homepagina van de installateur. De gebruiker kan via het keuzescherm (A) kiezen welke rol hij heeft en welk deel van de app hij dus kan bekijken na het inloggen. In dit geval kan hij via het inlogscherm (B) inloggen als installateur om zo toegang te krijgen tot de app. Op het homescherm van de installateur (H) zijn een aantal gegevens te zien, zoals aantal klanten en aantal open tickets. Ook kan de installateur zien of en welke klanten dan problemen hebben.

Figuur 11: Installateur klantenpagina van EnergyFlow.

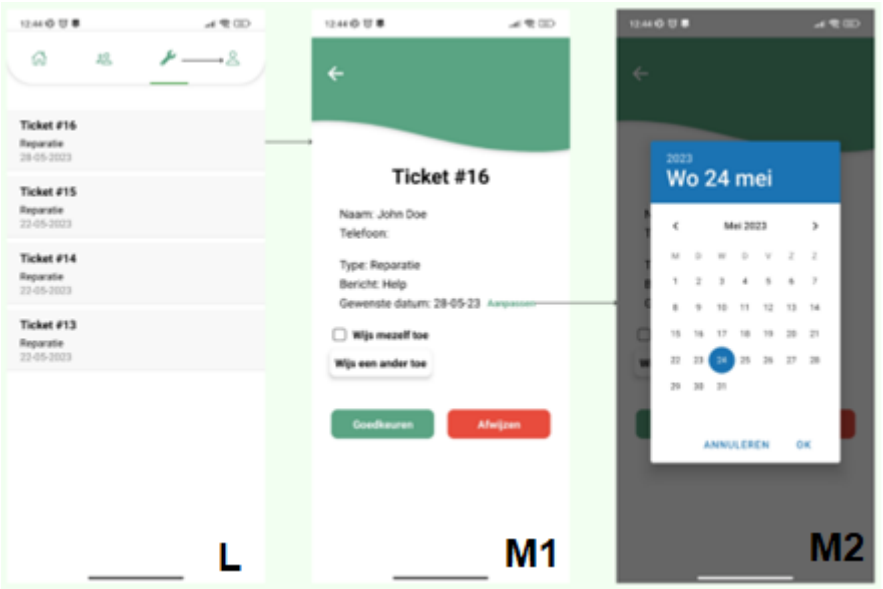
Wanneer naar links wordt geswiped vanuit het homescherm is het klantenscherm van de installateur te zien (I1). Hier zijn alle geregistreerde klanten te zien en kan er via de zoekbalk gezocht worden naar klanten via klantnummer of naam (I2). Een installateur kan ook een klant toevoegen en komt dan op de schermen van figuur 12.

Figuur 12: Klant toevoegen scherm en toegevoegde klant in klantenlijst van EnergyFlow.

Het toevoegen van klanten. Een installateur kan via het klant toevoegen scherm (J1) een klant met de benodigde informatie toevoegen, zoals naam, omvormer nummer, email en telefoon (J2). Wanneer deze is toegevoegd komt deze gelijk in de klantenlijst te staan (I3). Wanneer een installateur op een klant klikt in de lijst komt deze op de schermen in figuur 13.

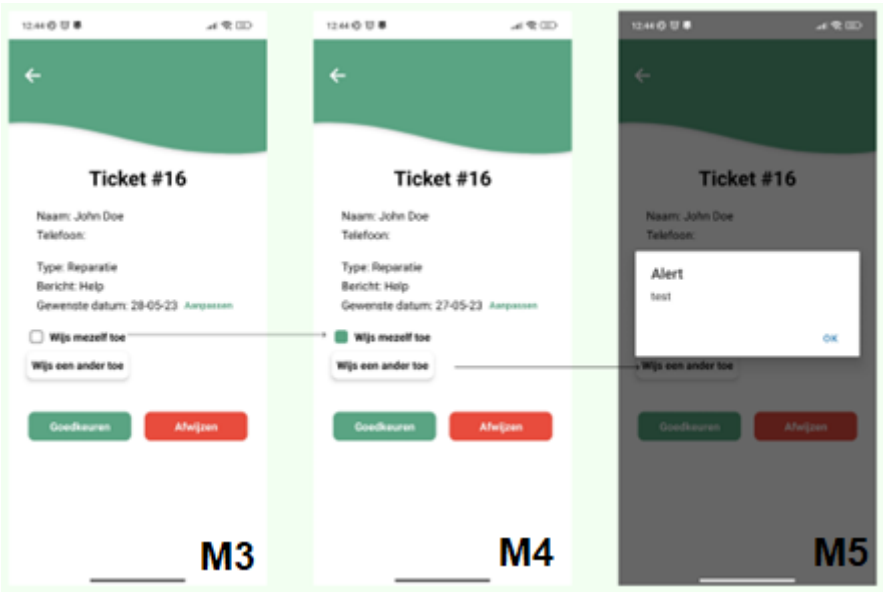
Figuur 13: Klant detailscherm voor een installateur van EnergyFlow.

Het klant detailscherm. Hier kan de installateur belangrijke details van het zonnepaneelsysteem van de klant inzien, zoals productie, consumptie, temperatuur en spanning (K1). Ook kan de installateur zien of er waarschuwingen zijn en hoe het systeem het over-time doet via de grafiek (K2).



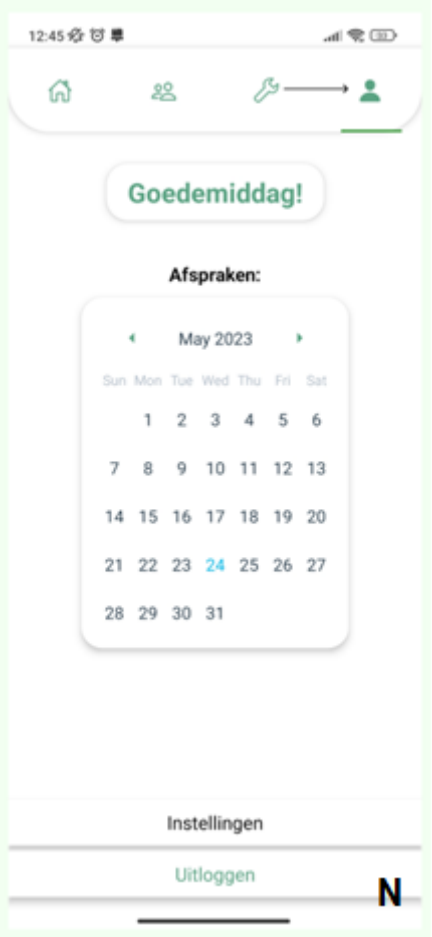
Figuur 14: Installateur ticketscherm en ticket detailscherm en het kiezen van een datum van EnergyFlow.

Wanneer naar links wordt geswiped vanuit het klantenscherm is het ticketscherm van de installateur te zien. De installateur kan hier alle open tickets zien en er op klikken (L). Wanneer er op een ticket wordt geklikt gaat de installateur naar de detailpagina van het ticket (M1). Hier kan de installateur alle details van het ticket en de aanmaker zien. De installateur kan de gewenste datum aanpassen door op de aanpassen knop te klikken en een nieuwe datum te kiezen (M2).



Figuur 15: Installateur ticket detailscherm en het toewijzen van installateurs van EnergyFlow.

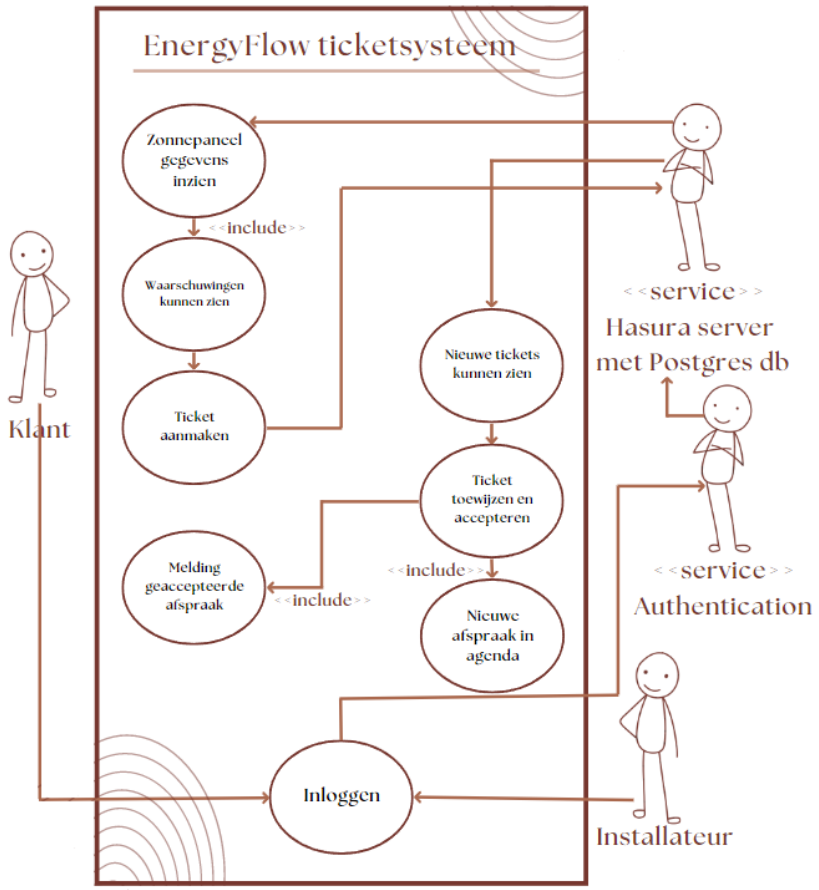
Ook kan de installateur een installateur toewijzen aan het ticket (M3). Hij kan zichzelf toewijzen door op de checkbox te klikken (M4) of hij kan een andere installateur aanwijzen door op de knop te klikken en een installateur uit de lijst te kiezen (M5). De lijst bestaat nog niet maar de popup wel. Wanneer een ticket wordt goedgekeurd of afgewezen veranderd de status van het ticket en krijgt de aanmaker een melding.



Figuur 16: Installateur profielpagina van EnergyFlow.

Wanneer naar links wordt geswiped vanuit het ticketscherm is het profielscherm van de installateur te zien (N). Hier kan de installateur zijn afspraken, gemaakt via de tickets, zien in de agenda tool. Ook kan de installateur zijn instellingen hier aanpassen en uitloggen via de ‘Uitloggen’ knop.

Ticket systeem Use Case diagram



Figuur 17: Use case diagram van het ticketsysteem van EnergyFlow.

Deze use case diagram dient als een schematische weergave van de interacties tussen verschillende gebruikerstypen (zonnepaneel eigenaren en zonnepaneel installateurs) en het ticketsysteem binnen de applicatie. Het is van belang om een van de belangrijkste functionaliteiten van de applicatie en de stroom van interacties binnen het systeem te begrijpen.

Het diagram illustreert de volgende processen:

Gebruikerslogin: Zowel zonnepaneel eigenaren als installateurs hebben toegang tot het systeem via een loginproces. Eenmaal ingelogd, kunnen ze interageren met de verschillende functies die de app biedt.

Datatoegang: Zodra een gebruiker is ingelogd, hebben ze toegang tot de gegevens van hun zonnepanelen. Dit stelt hen in staat om de status en prestaties van hun panelen te bekijken, en eventuele problemen te identificeren.

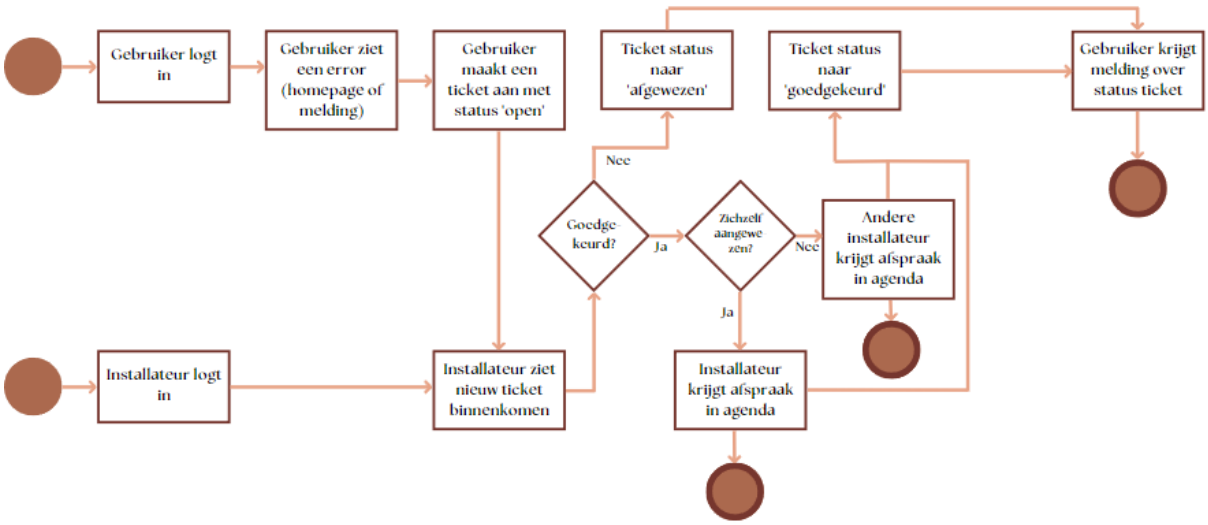
Ticketcreatie: Als er een probleem wordt geïdentificeerd, kan de eigenaar van het zonnepaneelsysteem een ticket aanmaken. Dit ticket, dat een beschrijving van het probleem bevat, wordt vervolgens naar de bijbehorende installateur gestuurd.

Ticketbeheer: De installateur heeft het vermogen om alle nieuwe tickets te bekijken. Ze kunnen een ticket accepteren en toewijzen aan zichzelf of een andere installateur. Dit triggert verdere acties in het systeem.

Afspraak maken: Wanneer een ticket wordt geaccepteerd, krijgt de installateur een afspraak in zijn of haar agenda. Dit zorgt voor een gestructureerd proces van taakbeheer en planning.

Notificatie: De eigenaar van het zonnepaneel ontvangt een notificatie wanneer zijn ticket is geaccepteerd. Dit zorgt voor transparantie en constante communicatie tussen de gebruikers van de applicatie.

Het gebruik van dit diagram is belangrijk voor de documentatie van de applicatie, omdat het een duidelijk en beknopt overzicht geeft van de gebruikersinteracties van een van de belangrijkste functionaliteiten binnen het systeem. Bovendien helpt het bij het begrijpen van de gevarieerde functionaliteiten en de workflow van de applicatie. Hier onder volgt ook nog een flowdiagram van het ticketsysteem.



Figuur 18: Flow diagram van het ticket systeem van EnergyFlow.

Technisch ontwerp

De technische documentatie van mijn project biedt een beschrijving van de architectuur, structuur, gebruikte tools en API-documentatie die zijn toegepast bij de ontwikkeling van de applicatie. Het doel van deze documentatie is om een duidelijk inzicht te bieden in de technische aspecten van het project en om andere ontwikkelaars en belanghebbenden te informeren over de keuzes en aanpak die zijn geïmplementeerd. Het bekijkt stap voor stap de technische lagen van het project en gaat steeds dieper in op de structuur.

Gebruikte tools

Dit hoofdstuk gaat over de keuze van de hulpmiddelen die zijn gebruikt tijdens de ontwikkeling van de applicatie. Het bouwen van een succesvolle app vereist een zorgvuldige afweging van de tools die tijdens het ontwikkelingsproces worden gebruikt. In dit hoofdstuk gaan we in op de redenen achter de selectie van deze tools en hun betekenis voor een efficiënte en hoogwaardige app-ontwikkeling.

___

React Native: Ik heb React Native gekozen omdat het een populair en breed geaccepteerd framework is voor het bouwen van cross-platform mobiele applicaties. De keuze lag voor mij bij React Native of Flutter, aangezien dit de enige hybrid frameworks zijn waar ik ervaring mee heb en ik niet nog een geheel nieuw framework wou gebruiken. De doorslaggevende factor voor het kiezen van React Native was uiteindelijk toch omdat React (Native) ook veel binnen het bedrijf wordt gebruikt. Het stelt me verder in staat om eenmaal code te schrijven en deze te implementeren op zowel iOS- als Android-platforms, wat de ontwikkelingstijd en -kosten vermindert. Bovendien maakt React Native gebruik van JavaScript en React, bekende en veelgebruikte technologieën. Dit betekent dat er gemakkelijk bronnen, documentatie en ondersteuning gevonden kan worden binnen zowel mijn bedrijf en de ontwikkelaarsgemeenschap.

Hasura met GraphQL: Voor bepaalde backend logica gebruik ik Hasura met GraphQL om snel een schaalbare en efficiënte API te bouwen voor de applicatie. Dit was een requirement vanuit het bedrijf omdat ze dit voor al hun applicaties gebruiken. GraphQL's querytaal zorgt er voor dat alleen de benodigde gegevens opgevraagd kunnen worden, waardoor onnodige netwerkoverhead wordt verminderd en de prestaties worden verbeterd. Hasura vereenvoudigt het proces van het maken van GraphQL API's door automatisch het schema, de resolvers en de datatoegangslaag te genereren vanuit mijn PostgreSQL database.

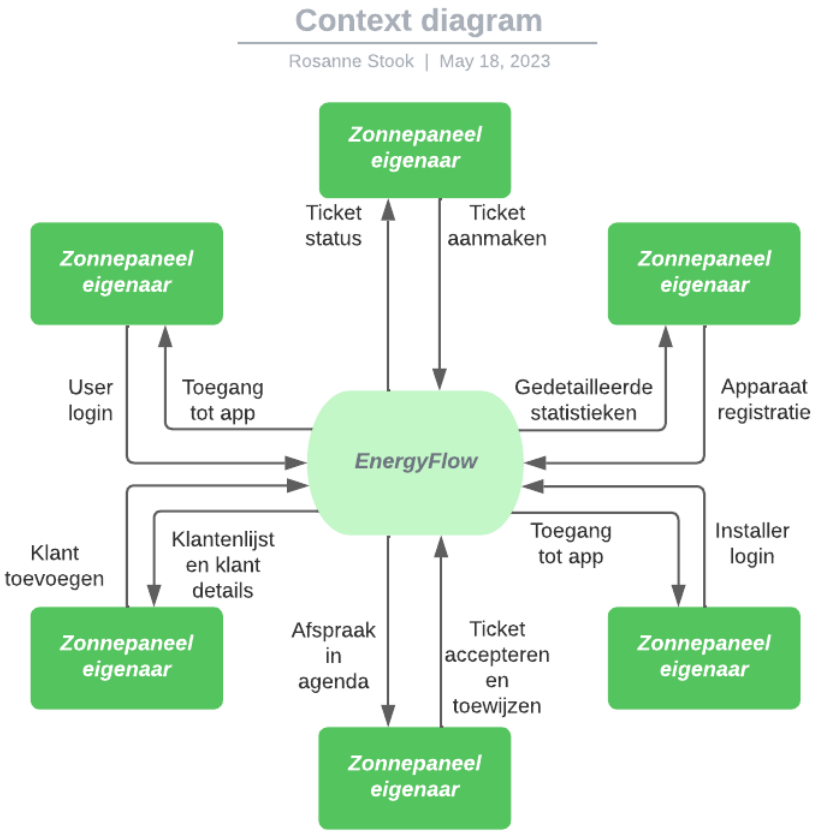
PostgreSQL: PostgreSQL is gekozen als mijn database management systeem omdat het robuust en betrouwbaar is, en veel gebruikt wordt in (web)ontwikkeling. Het biedt schaalbaarheid en prestatievermogen, waardoor het geschikt is voor het verwerken van grote hoeveelheden gegevens. PostgreSQL wordt gebruikt als database systeem binnen ProfitFlow en is daarom een requirement.

Expo: Ik heb Expo gebruikt voor het ontwikkelings-, bouw- en implementatieproces voor mijn React Native-applicatie. Door Expo te gebruiken, kreeg ik toegang tot vooraf gebouwde componenten en API's die de ontwikkeling versnellen en hoge kwaliteit garanderen. De meegeleverde ontwikkeltools, zoals de visuele editor en de simulator zijn erg handig om de applicatie te testen en te debuggen.

Apollo: Apollo is een library voor een uitgebreide set tools en diensten voor het bouwen en beheren van GraphQL API's. De door Apollo geleverde client library integreert gemakkelijk met populaire frameworks en is gebruikt om de GraphQL functies van de Hasura server te integreren met de applicatie.

Clerk: Clerk is een veelzijdige authenticatieoplossing die specifiek is ontworpen om ontwikkelaars te helpen bij het implementeren van veilige en gebruiksvriendelijke gebruikersauthenticatie in hun applicaties. Ik heb gekozen voor Clerk vanwege de vele voordelen die het biedt, zoals het feit dat het een robuust beveiligingsframework is dat is gebaseerd op moderne beveiligingsstandaarden en -protocollen waardoor je kunt vertrouwen op een veilige authenticatie-ervaring voor gebruikers, inclusief wachtwoordhashing, tweefactorauthenticatie en bescherming tegen veelvoorkomende beveiligingskwetsbaarheden. Ik gebruik Clerk om zijn functionaliteiten in mijn frontend te kunnen gebruiken om zo de authenticatie en autorisatie af te handelen. Ook kan ik met Clerk een rollensysteem toevoegen, waardoor ik gemakkelijk een onderscheid kan maken tussen installateur en klant.

Context diagram



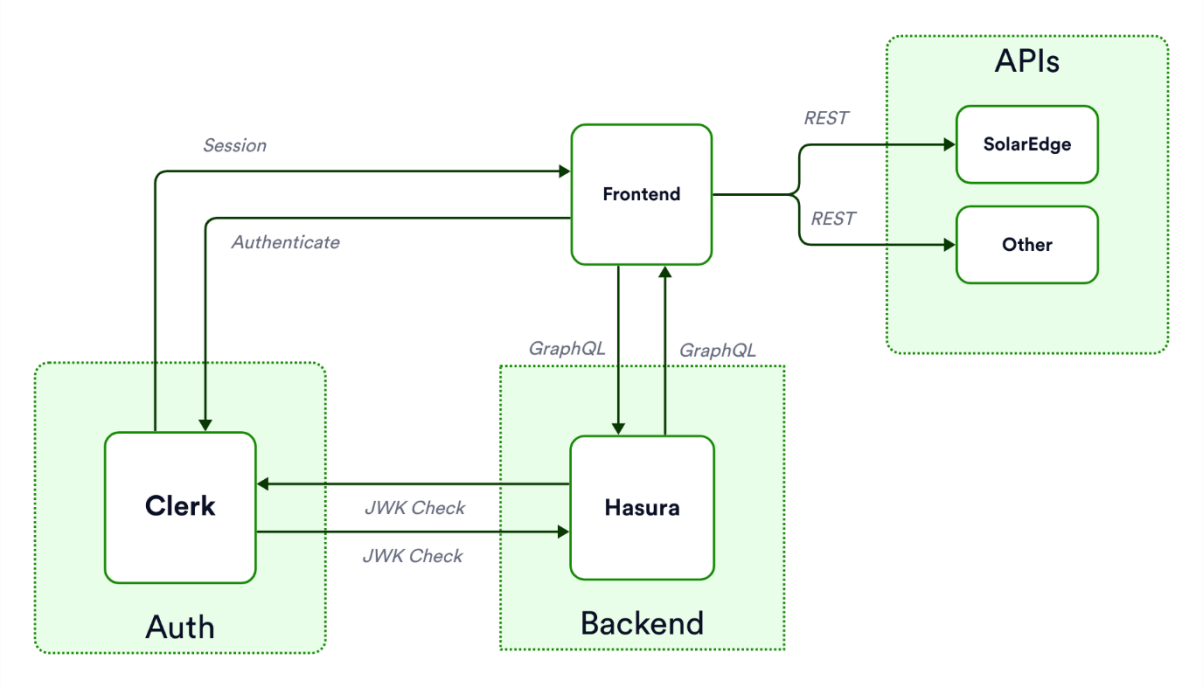
Figuur 19: Context diagram van EnergyFlow.

Het contextdiagram illustreert de overkoepelende structuur van de Energyflow app en is een hulpmiddel bij het analyseren en begrijpen van de applicatie. Het biedt een overzicht van de interacties tussen het systeem en externe entiteiten, in dit geval de twee type gebruikers.

Voor eigenaren van zonnepanelen biedt het systeem verschillende functionaliteiten, waaronder gebruikerslogin, het indienen van een nieuw ticket en apparaat registratie. Deze functionaliteiten stellen eigenaren in staat om toegang te krijgen tot de app, problemen te rapporteren en gedetailleerde statistieken van hun zonnepaneelsystemen te bekijken. Dit stelt eigenaren in staat om essentiële informatie over hun zonnepanelen te verkrijgen en de voortgang er van te volgen.

Voor zonnepaneelinstallateurs omvat het systeem functionaliteiten zoals login, ticketacceptatie en -toewijzing en klantbeheer. Door gebruik te maken van deze functies kunnen installateurs toegang krijgen tot de app, afspraken maken en beheren en de gegevens van hun klanten bijhouden. Dit stelt installateurs in staat om hun taken efficiënt te beheren en een gestroomlijnde dienstverlening aan klanten te bieden.

Architectuur diagram



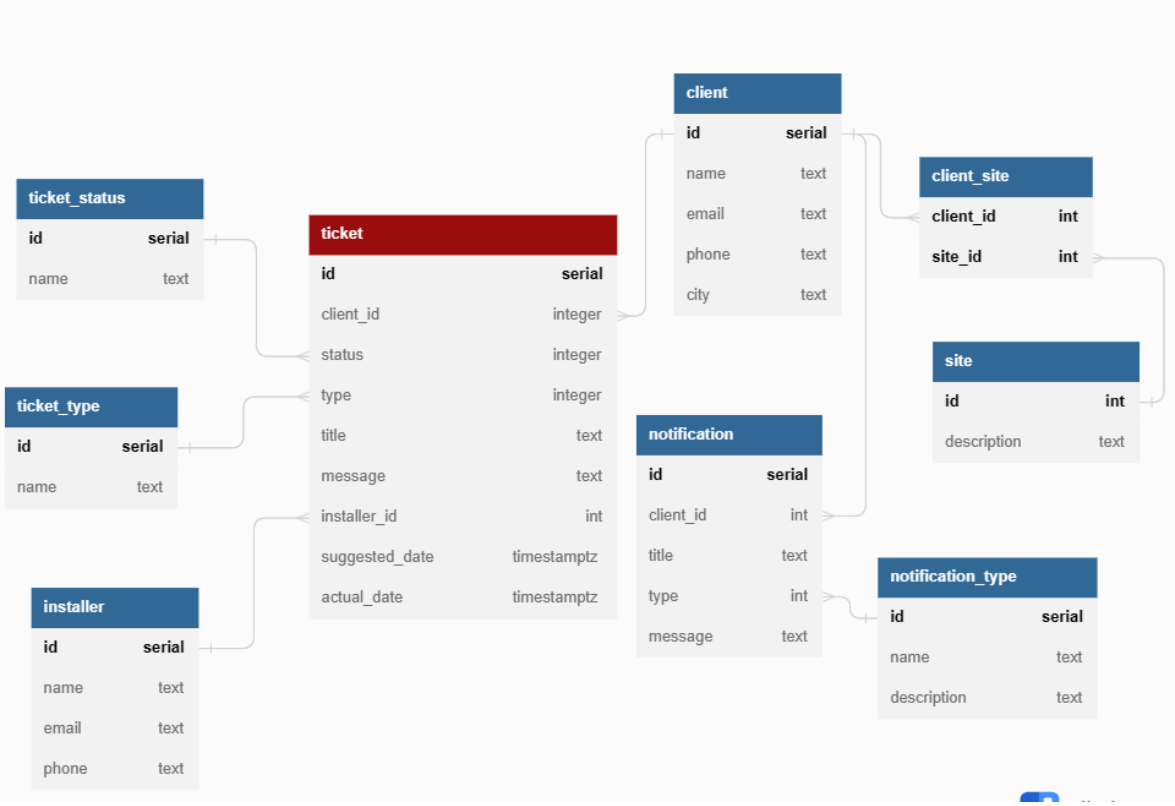
Figuur 20: Architectuur diagram van EnergyFlow.

Het architectuurschema van EnergyFlow bestaat uit verschillende componenten die elk een specifieke functie binnen het systeem vervullen.

Hasura, als onderdeel van de Backend-sectie, faciliteert de communicatie met Clerk, die verantwoordelijk is voor het afhandelen van de authenticatie. Hasura communiceert met Clerk om een JSON Web Key (JWK)-controle uit te voeren, om de geldigheid van authenticatietokens te waarborgen. Clerk communiceert op zijn beurt met Hasura voor hetzelfde doel, waardoor een wederzijdse interactie tussen de twee componenten ontstaat. Clerk heeft ook interactie met het Frontend-gedeelte om sessies te beheren. Door deze interactie kan de Frontend gebruikerssessies opzetten en onderhouden in de gehele applicatie. Bovendien communiceert de Frontend met Clerk voor gebruikersauthenticatie, zodat gebruikers veilig toegang hebben tot de functies en middelen van het systeem.

Wat betreft het ophalen en bewerken van gegevens speelt Hasura een belangrijke rol. Het communiceert met de Frontend sectie voor GraphQL operaties, waardoor efficiënte en flexibele data queries en updates mogelijk zijn. Op dezelfde manier communiceert de Frontend ook met Hasura voor GraphQL-bewerkingen, waardoor een soepele gegevensstroom tussen de gebruikersinterface en de backend ontstaat. Verder communiceert de Frontend-sectie met de API's-sectie voor RESTful-interacties. Dankzij deze interacties kan de Frontend gegevens en diensten gebruiken die door externe API's, zoals SolarEdge, worden geleverd. Dankzij deze integratie kan de applicatie gegevens van deze API's benutten en in functionaliteiten opnemen.

Database diagram



Figuur 21: Database diagram van EnergyFlow.

Het database diagram is een belangrijk onderdeel van de applicatie architectuur, en geeft een overzicht van de database structuur en de relaties tussen entiteiten. Het diagram toont een doordacht ontwerp met verschillende tabellen die elk een specifiek doel dienen. De database draait via de Hasura server en is hier volledig vanuit de grond gemaakt om te werken voor mijn EnergyFlow app. De frontend can communiceren met de Hasura server om data op te halen of te bewerken die uit deze database komen.

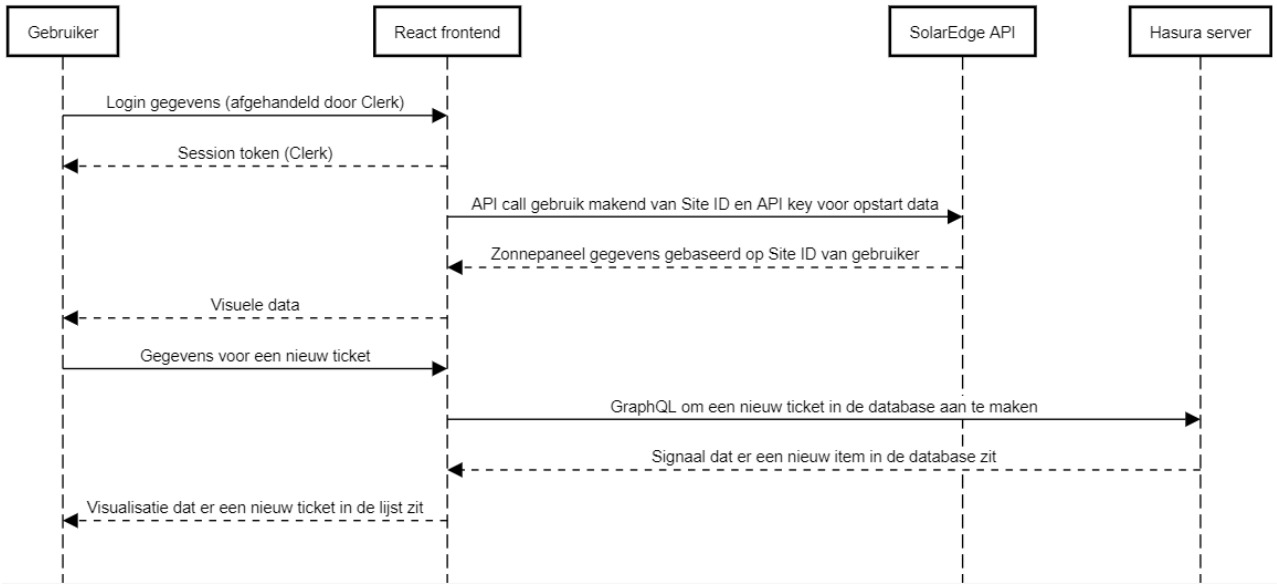
De ticket tabel is een centraal onderdeel van de database, waarmee zonnepaneel eigenaren tickets kunnen aanmaken en problemen kunnen rapporteren aan hun zonnepaneel installateurs. De tabel bevat velden zoals titel, type, bericht en datum, zodat alle relevante informatie wordt vastgelegd en bijgehouden. Ook de aanmaker wordt bijgehouden via het client_id en zodra een installateur wordt aangewezen wordt ook deze bijgehouden zodat er een afspraak kan worden geplaats in de agenda van de installateur.

De klantentabel bevat informatie over de zonnepaneelgebruikers van de app, waaronder hun naam, e-mail, telefoonnummer en woonplaats. Deze tabel biedt een gecentraliseerde locatie voor het beheer van klantgegevens, zodat installateurs van zonnepanelen snel en gemakkelijk toegang hebben tot klantinformatie wanneer dat nodig is. De clientsite tabel bevat informatie over de zonnepaneel systemen die klanten bezitten. Deze tabel zorgt ervoor dat alle relevante informatie over zonnepaneelsystemen wordt vastgelegd en bijgehouden, zodat installateurs deze systemen gemakkelijker kunnen beheren en onderhouden.

De tabellen ticket type en ticket status bevatten respectievelijk informatie over de soorten tickets die kunnen worden aangemaakt en de status van elk ticket. Deze tabellen zorgen ervoor dat alle tickets goed worden bijgehouden en beheerd, waardoor een gestroomlijnde aanpak voor het beheer van problemen van klanten ontstaat.

De meldingentabel tenslotte bevat informatie over klantmeldingen, waaronder de klant-ID, de titel, het type en het bericht. Deze tabel zorgt ervoor dat alle relevante informatie over klantmeldingen wordt vastgelegd en bijgehouden, waardoor meldingen ook custom worden uitgestuurd en worden verstuurd bij de juiste dingen.

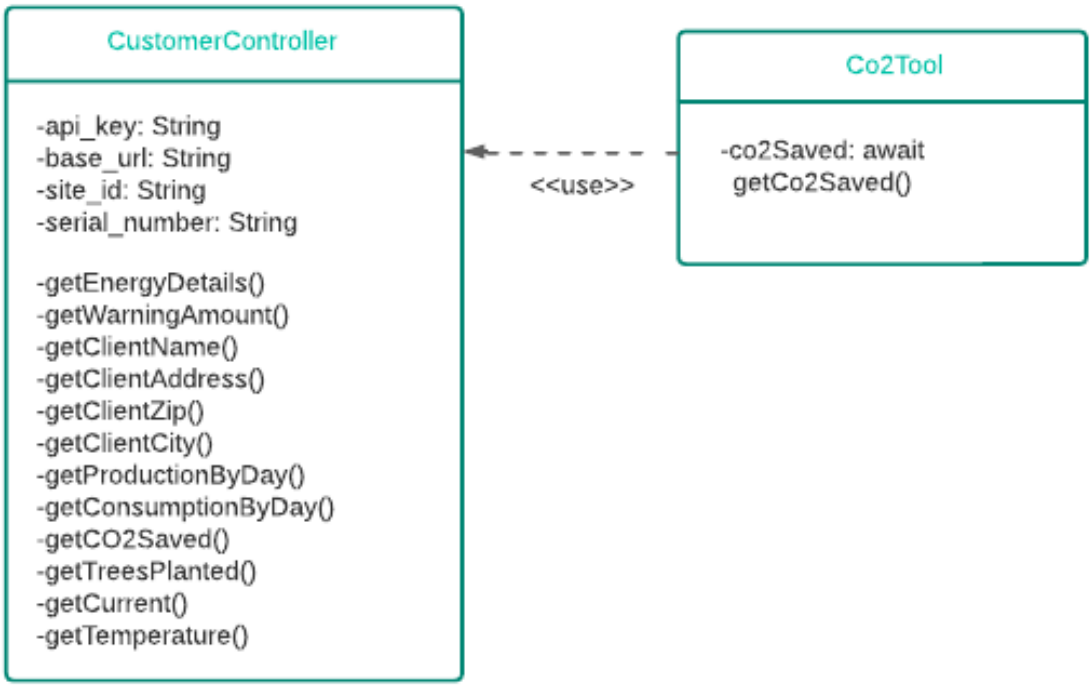
API en Hasura Sequence diagram



Deze sequence diagram illustreert de interacties tussen verschillende componenten bij het gebruik van de applicatie en de externe API's. Het proces begint wanneer de gebruiker inloggegevens verstrekt aan de React frontend van de applicatie, die vervolgens deze gegevens afhandelt via de Clerk-authenticatie.

Na succesvolle verificatie ontvangt de gebruiker een sessietoken. Met dit sessietoken kan de gebruiker verzoeken om specifieke gegevens op te halen van de SolarEdge API. De React frontend maakt een API-call naar de SolarEdge API met behulp van de site-ID van het ingelogde account en een API-key om gegevens het zonnepaneelsysteem te verkrijgen. De SolarEdge API reageert met de benodigde zonnepaneelgegevens, die de gebruiker vervolgens visueel kan bekijken. Bovendien stelt de applicatie de gebruiker in staat om bepaalde interactieve taken uit te voeren via de Hasura server met aangesloten database. Om bijvoorbeeld een nieuw ticket aan te maken geeft de gebruiker de nodige gegevens aan de React frontend, die via een GraphQL-verzoek naar Hasura een nieuw ticket aanmaakt in de database. Zodra de Hasura server het nieuwe ticket in de database heeft aangemaakt, stuurt het een signaal terug naar de React frontend om aan te geven dat er een nieuw item in de database is. De React frontend informeert vervolgens de gebruiker over het nieuwe ticket, zodat het zichtbaar is in de lijst met tickets.

Controller Class diagram

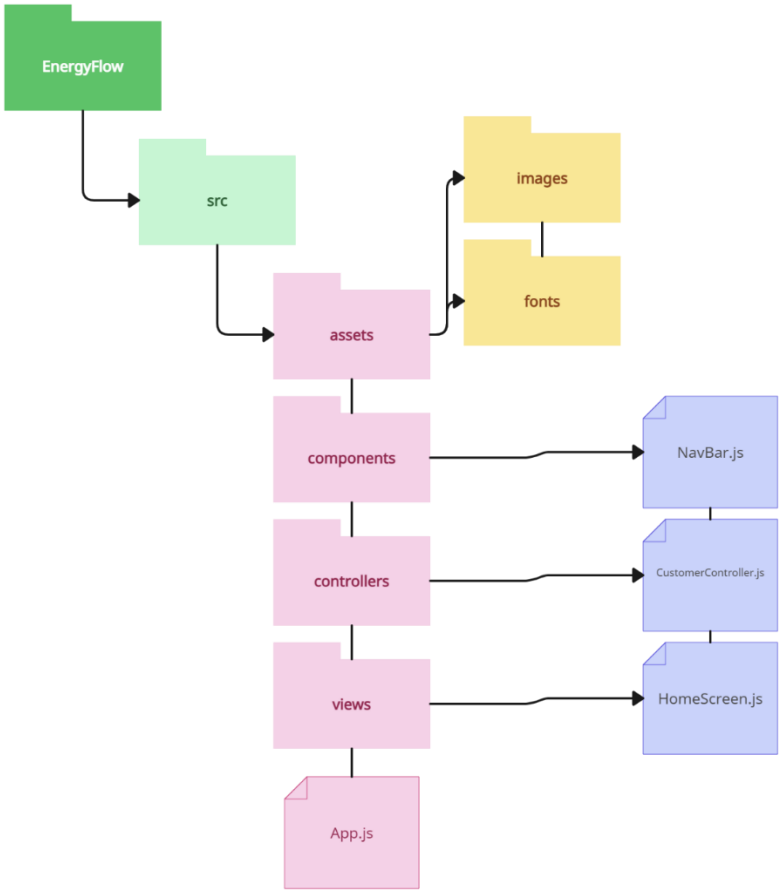


Bovenstaande class diagram illustreert een van de belangrijkste classes van de applicatie, de customer controller. Deze class bevat alle functies die API calls doen naar de SolarEdge API. Via deze controller wordt op een gestructureerde manier alle relevante data van de SolarEdge API verzameld en kunnen de functies vanuit de gehele app worden aangeroepen door de functies te importeren. Ook worden in deze controller belangrijke variabelen aangemaakt die nodig zijn om de API te kunnen benaderen, zoals de base URL en een API key.

Het gebruik van controllers zorgt er ook voor dat de app makkelijk uit te breiden is voor wanneer er meerdere API's moeten worden toegevoegd. Er kunnen dan meerdere controllers worden aangemaakt, waarbij elke API zijn eigen controller krijgt. Op deze manier wordt er onderscheidt tussen elke API gemaakt en kan er gemakkelijk, door bijvoorbeeld het toevoegen van een bedrijfsnaam aan een account, gecheckt worden bij welk bedrijf het account hoort en om zo de juiste functie aan te roepen. Het splitsen van controllers wordt momenteel al gedaan, aangezien ik gebruik maak van een aparte Weather Data API (VisualCrossing) en deze zijn aparte controller heeft.

Tot slot wil ik benadrukken waarom ik er voor heb gekozen om verschillende functies te maken per type data (bijvoorbeeld client city, name, address). De eerste functie, getEnergyDetails, is een grote functie die meerdere gegevens kan ophalen in één call. Uit deze call kan bijvoorbeeld de productie én consumptie van een zonnepaneelsysteem worden opgehaald per dag, week of jaar. Echter moet deze functie dan weer worden ontleed in de code van een component of view. Hoewel dit in principe geen probleem is, vind ik het persoonlijk fijner om de functies te splitsen en direct de data op te kunnen halen zoals in het Co2Tool component.

Folderstructuur diagram



Figuur 22: Folderstructuur diagram van EnergyFlow.

Het folderstructuur diagram is een onderdeel van de applicatiearchitectuur en biedt een duidelijk en georganiseerd overzicht van de bestandshiërarchie. Het diagram toont het ontwerp met verschillende mappen die elk een specifiek doel dienen.

De hoofdmap, EnergyFlow, dient als de hoofdmap van de applicatie en biedt een centrale locatie voor alle bestanden en mappen. Binnen de src-map zijn er verschillende submappen, waaronder assets, componenten, controllers en views, elk met een specifiek doel. De map assets bevat afbeeldingen en fonts, die essentiële bronnen zijn voor de opmaak van de applicatie. De map components bevat bestanden van React-componenten die belangrijke bouwstenen zijn van de gebruikersinterface van de applicatie. De map controllers bevat bestanden van controllers, zoals CustomerController, die verantwoordelijk zijn voor het beheer van de gegevens en de bedrijfslogica van de applicatie. Tot slot bevat de map views bestanden van schermen die verantwoordelijk zijn voor het renderen van de gebruikersinterface.

Het diagram van de folderstructuur laat een goed ontworpen en uitgebreide bestandshiërarchie zien, die ervoor zorgt dat alle relevante bestanden en mappen georganiseerd en gemakkelijk toegankelijk zijn. Deze structuur maakt efficiënte ontwikkeling en onderhoud van de applicatie mogelijk en zorgt voor een gestroomlijnde aanpak bij het bouwen en beheren van de functionaliteit van de applicatie.

Hasura server documentatie

In het kader van het verzamelen, toevoegen en aanpassen van data die niet direct via externe API's beschikbaar zijn, heb ik gebruik gemaakt van Hasura in combinatie met GraphQL. Deze technologieën zijn geïmplementeerd om een scala aan functionaliteiten in mijn project mogelijk te maken.

Bijvoorbeeld, één van de primaire toepassingen was het verkrijgen van klantgegevens. Via Hasura en GraphQL kon ik een verzoek opzetten om alle klantgegevens op te halen uit de database, wat belangrijk is voor het beheer van klantrelaties binnen de applicatie. Bovendien zijn klantmeldingen toegankelijker gemaakt via deze technologieën, wat het systeem in staat stelt om inzicht te krijgen in klantbehoeften en -verzoeken. Hasura en GraphQL zijn ook ingezet voor het toevoegen en aanpassen van gegevens, waaronder tickets, klanten en meldingen. Een installateur kan bijvoorbeeld een ticket goedkeuren of afwijzen, wat resulteert in een verandering van de status van dat ticket.

Het bouwen van een database met de benodigde tabellen was een soepele ervaring dankzij Hasura. Zoals te zien is in Figuur 3, heeft Hasura het eenvoudig gemaakt om een gestructureerde database op te zetten die alle vereiste gegevens bevat. Daarnaast kon ik met behulp van GraphQL methodes opstellen die de benodigde gegevens ophaalden of aanpasten. Deze methodes werden vervolgens geïmporteerd en gebruikt in het React-project. Voorbeelden van deze methodes zijn te zien in de sectie 'Code Examples' hieronder.

Tot slot, voor een uitgebreid overzicht van de mogelijkheden die de Hasura server biedt en de bijbehorende vereisten, verwijs ik naar deze link. Het documenteert de brede reeks mogelijkheden die Hasura en GraphQL bieden voor geavanceerde datamanipulatie en -opvraging voor mijn data structuur.

Link naar Hasura server documentatie:
https://graphdoc.io/doc/Iqs5w4u1kcHX7l2z/

Code examples

In dit gedeelte wil ik graag een paar belangrijke delen van de codebase bespreken en laten zien hoe ze zijn geimplementeerd. Het gaat over user authentication, data mutation en API integratie.

User authentication

In onderstaande code worden enkele belangrijke functies van het gebruikersauthenticatieproces geïmplementeerd.

De code begint met een useEffect hook, een belangrijk onderdeel van de React functional components, die wordt uitgevoerd wanneer de waarde van isSignedIn of organizationList verandert. Deze hook bepaalt wat er gebeurt nadat een gebruiker succesvol is ingelogd. Wanneer een gebruiker is ingelogd (isSignedIn is true), zoekt de code in de organizationList naar een specifieke organisatie met een bepaald id. Als deze organisatie wordt gevonden, betekent dit dat de ingelogde gebruiker een installateur is, en wordt hij/zij omgeleid naar de 'MainInstaller' pagina. Zo niet, dan wordt aangenomen dat de gebruiker een klant is en wordt hij/zij naar de 'MainClient' pagina geleid. Deze functie illustreert hoe Clerk gebruikt wordt om de gebruiker te identificeren als klant of installateur, en zo toegang te geven tot de juiste deel van de app.

De tweede functie, onSignInPress, wordt uitgevoerd wanneer een gebruiker probeert in te loggen. Als de app niet volledig geladen is (isLoaded is false), gebeurt er niets. Wanneer de app wel geladen is, wordt er geprobeerd om de gebruiker in te loggen via de signIn.create methode van Clerk. Deze methode ontvangt een object met het e-mailadres (identifier) en het wachtwoord van de gebruiker. Als dit succesvol is, wordt de gebruiker ingelogd en wordt er een nieuwe sessie aangemaakt met de setActive methode. Dit illustreert hoe Clerk wordt gebruikt om de gebruiker te zoeken in de Clerk-database, en om een sessie aan te maken zodat de gebruiker ingelogd blijft, zelfs wanneer de app opnieuw wordt opgestart. extra beveiligingslaag door te voorkomen dat een gebruiker kan inloggen met ongeldige inloggegevens.

```
// This effect runs when the dependencies 'isSignedIn' or 'organizationList' change.
useEffect(() => {
  // Check if the user is signed in.
  if (isSignedIn) {
    // Find the organization with the specified ID in the organizationList.
    const isInstallateur = organizationList?.find((organization) => organization.organization.id === "org_2QHO2tIuv7uO4xMEnep1ezF4Rr

    // If the organization with the specified ID exists in the organizationList,
    // navigate to the 'MainInstaller' screen.
    if (isInstallateur) {
      navigation.push('MainInstaller');
    } else {
      // If the organization with the specified ID does not exist in the organizationList,
      // navigate to the 'MainClient' screen.
      navigation.push('MainClient');
    }
  }
}, [isSignedIn, organizationList]);

// This function is triggered when the user presses the sign-in button.
const onSignInPress = async () => {

  // If the required data is not loaded yet, return and do nothing.
  if (!isLoaded) {
    return;
  }

  // Attempt to sign in using the provided email address and password.
  try {
    const completeSignIn = await signIn.create({
      identifier: emailAddress,
      password,
    });

    // Set the active session using the created session ID.
    await setActive({ session: completeSignIn.createdSessionId });

    // Reset any error message.
    setError("");
  } catch (err) {
    // If an error occurs during sign-in, set an error message.
    setError("Incorrect email or password");
  }
};
```

Figuur 23: Voorbeeld code; user authentication voor EnergyFlow.

API integratie

Externe API's (in dit geval één) worden geintegreerd door hun functies aan te roepen vanuit controller files. Hier word een bepaalde URL aangeroepen om de gewenste data op te halen. Deze data wordt dan later opgeroepen en omgezet zodat deze in de UI van de app kan worden gebruikt.

In onderstaande code is bijvoorbeeld de getEnergyDetails functie te zien. Deze maakt een API-aanvraag naar een URL die is samengesteld op basis van verschillende parameters, waaronder de tijdseenheid (timeUnit) en het aantal dagen (totalDays). Afhankelijk van de meegegeven parameters, wordt een specifieke starttijd (startTime) en eindtijd (endTime) voor de aanvraag bepaald. Ook is een API-key en site-ID nodig om de URL te bereiken. Een site-id is een uniek nummer voor een zonnepaneelsysteem. Ik heb zo'n site-ID gekregen zodat ik échte real-time data kon gebruiken voor mijn project. Na het aanroepen van de URL, wordt de ontvangen data omgezet in een JSON-formaat.

De verkregen data wordt dan gebruikt in een component met behulp van de useEffect hook. De getEnergyDetails functie wordt aangeroepen en zodra de beloofde data terugkeert, worden de noodzakelijke details geëxtraheerd. In dit geval, waar de "YEAR" parameter wordt gebruikt, wordt de totale jaarproductie berekend door de waarden van de productie samen te voegen en om te zetten naar kWh. De berekende waarde wordt dan opgeslagen voor latere weergave.

```
// Gets the energydetails of a site from the SolarEdge API
function getEnergyDetails(timeUnit = "DAY", totalDays = 0) {
  // Get the current date and time.
  const now = new Date();

  // Calculate the date one month ago from the current date.
  const oneMonthAgo = new Date(now.getFullYear(), now.getMonth() - 1, now.getDate());

  // Format the current date to the desired format (YYYY-MM-DD).
  const formattedDate = now.toISOString().slice(0, 10);

  // Format the one month ago date to the desired format (YYYY-MM-DD).
  let formattedMonthAgo = oneMonthAgo.toISOString().slice(0, 10);

  if (timeUnit === "YEAR") {
    // If the time unit is set to "YEAR", calculate the date two years ago from the current date.
    const manyYearsAgo = new Date(now.getFullYear() - 2, now.getMonth(), now.getDate());
    formattedMonthAgo = manyYearsAgo.toISOString().slice(0, 10);
  }

  if (timeUnit === "DAY" && totalDays !== 0) {
    // If the time unit is set to "DAY" and totalDays is provided, calculate the date "totalDays" ago from the current date.
    const xDaysAgo = new Date(now.getFullYear(), now.getMonth(), now.getDate() - totalDays);
    formattedMonthAgo = xDaysAgo.toISOString().slice(0, 10);
  }

  if (timeUnit === "HOUR") {
    // If the time unit is set to "HOUR", set the start time to the current date.
    const xDaysAgo = new Date(now.getFullYear(), now.getMonth(), now.getDate());
    formattedMonthAgo = xDaysAgo.toISOString().slice(0, 10);
  }

  // Construct the URL to fetch the energy details based on the specified parameters.
  const url = baseUrl + `/site/${siteId}/energyDetails?meters=PRODUCTION,CONSUMPTION&timeUnit=${timeUnit}
  &startTime=${formattedMonthAgo}%2023:00:00&endTime=${formattedDate}%2023:00:00&api_key=${apiKey}`;

  // Perform the fetch request to retrieve the energy details.
  return fetch(url, {})
    .then(response => response.json());
}
```

Figuur 24: Voorbeeld code; energydetails verkrijgen van SolarEdge API.

```
  // Fetch energy details for the time unit "YEAR" and calculate the total production.
  getEnergyDetails("YEAR").then(res => {
    const energyDetails = res.energyDetails;
    const meters = energyDetails.meters;
    const production = meters[1];

    // Calculate the total production by summing up all the production values.
    const totalProduction = production.values.reduce((prev, current) => {
      return prev + current.value;
    }, 0);

    // Convert the total production from Wh to kWh and set the state.
    const kwh = totalProduction / 1000;
    setTotalProduction(Number(kwh.toFixed(2)));
  });
}, []);
```

Figuur 25: Voorbeeld code; energydetails verwerken naar totale jaarproductie..

Data mutatie

Binnen de app wordt Hasura samen met GraphQL gebruikt voor het beheren van de database en het uitvoeren van bewerkingshandelingen, zoals het aanmaken van tickets. De code hier onder laat een voorbeeld zien van het aanroepen van een GraphQL-mutatie om een nieuw ticket aan te maken.

De CREATE_TICKET mutatiequery definieert de structuur van de mutatie en de variabelen die nodig zijn om het ticket aan te maken. Deze query accepteert variabelen zoals client_id, message, status, suggested_date, title, en type. De insert_ticket_one mutatie wordt uitgevoerd en het aangemaakte ticket wordt teruggegeven met de id als respons. Vervolgens wordt de useMutation hook gebruikt om de createTicket functie en de bijbehorende gegevens en laadstatus te initialiseren. Binnen de onCompleted callback functie wordt er genavigeerd naar de "TicketsClient" pagina nadat de mutatie is voltooid.

Wanneer de onSend functie wordt uitgevoerd, wordt de createTicket functie aangeroepen met de benodigde variabelen om het ticket aan te maken. Hier worden waarden zoals client_id, message, title, suggested_date, en type ingesteld op basis van de specifieke logica van de app. Zodra de createTicket functie wordt aangeroepen, wordt de GraphQL-mutatie uitgevoerd en worden de variabelen naar de Hasura-server verzonden. De Hasura-server verwerkt de mutatie en slaat de nieuwe ticketgegevens op in de database.

```
const CREATE_TICKET = gql`
  mutation CREATE_TICKET(
    $client_id: Int!,
    $message: String!,
    $status: Int = 1,
    $suggested_date: timestamptz = "now()",
    $title: String!,
    $type: Int = 1
  ) {
    insert_ticket_one(object: {
      client_id: $client_id,
      title: $title,
      message: $message,
      type: $type,
      status: $status,
      suggested_date: $suggested_date
    }) {
      id
    }
  }
`;
```

Figuur 26: Voorbeeld code; GraphQL importatie.

```
// Event handler for sending the ticket
const onSend = () => {
  createTicket({
    variables: {
      client_id: 1,
      message: description,
      title: "",
      suggested_date: date,
      type: getTicketTypeId(selectedType)
    }
  });
};
```

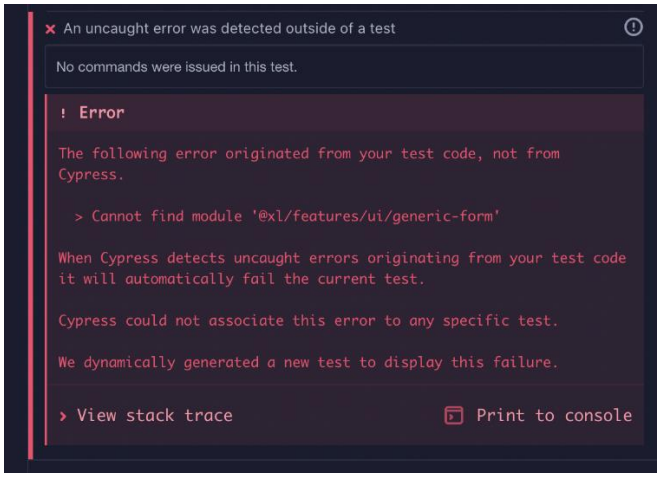
Figuur 27: Voorbeeld code; aanmaken van een ticket met nieuwe data om op te sturen naar Hasura server.

Testen

Aanvankelijk was het mijn intentie om een aantal front-end testen uit te voeren om de betrouwbaarheid van de app te verzekeren. Ik had van te voren bedacht om PlayWright te gebruiken, omdat dit binnen het bedrijf veel wordt gebruikt. Toen ik merkte dat dit niet heel soepel liep en dat er ook andere opties waren heb ik het ook geprobeerd met onder andere Cypress, Detox, Jest en enkele kleinere open-source libraries. Echter, tijdens de poging om deze testtools te implementeren, liep ik tegen onverwachte problemen aan. Bij een aantal installatiepogingen (PlayWright en Cypress voornamelijk) waren er steeds een eindeloos aantal aan packages en libraries nodig. Dit leidde uiteindelijk juist tot meer installatiefouten zonder dat ik kon vinden wat er fout was en hoe het wel moest. De installatie tutorials van beide tools heb ik precies gevolgd, maar elke keer miste ik weer iets. Bij andere installaties kreeg ik bij het proberen te runnen van testen errors zonder duidelijke foutmelding. Hierdoor wist ik niet goed waar het aan lag en of ik deze tool überhaupt wel voor mijn applicatie kon gebruiken. Ook kon ik erg weinig informatie over de fouten vinden bij verschillende bronnen op het internet.

Het is belangrijk om te benadrukken dat, ondanks deze tegenslagen, heel veel uren zijn besteed aan het proberen te overwinnen van deze problemen. Ik heb verschillende installatieprocedures geprobeerd, alternatieve libraries overwogen, en uitgebreid online onderzoek gedaan naar mogelijke oplossingen. Echter, vanwege de aanhoudende moeilijkheden, het niet beschikbaar hebben van mensen met ervaring die me zouden kunnen helpen en de naderende deadline voor de indiening van dit afstudeerverslag, heb ik uiteindelijk moeten besluiten om de front-end testen niet uit te voeren.

In plaats daar van heb ik er voor gekozen om handmatige testen uit te voeren door middel van een handmatig test plan. Ik heb een testpersoon uitgekozen die een potentiële gebruiker van de app zou kunnen zijn, om te testen of de functionaliteiten en flow van de app naar behoren werkt. Ook heb ik de feedback van de tester per test case opgenomen zodat deze meegenomen zou kunnen worden bij toekomstige ontwikkeling. Ik heb er voor gekozen om de testen op deze manier te doen omdat het me niet meer zou lukken om met hulp automatische testen te implementeren. Toch wou ik iets van testen doen, en kreeg ik de tip om handmatige testen uit te voeren. Ik denk dat dit een goede keuze was omdat ik op deze manier alsnog de kwaliteit van de app heb kunnen analyseren en heb ik extra feedback over de app mogen ontvangen die erg waardevol kan zijn. **Het testplan is te bekijken in het testplan document in de bijlagen.**



Figuur 27: Een van de vele errors bij het installeren van test software; Cypress

Conclusie

Dit hoofdstuk gaat in op de bevindingen en inzichten die ik tijdens mijn afstudeerperiode heb opgedaan. Ik reflecteer op het proces, analyseer het product en presenteer de eindconclusies. Via deze samenvatting geef ik een kort overzicht van de belangrijkste elementen die aan bod kwamen en trek ik algemene conclusies over de volledige scope van het project.

Het begint met het bespreken van het proces, waarbij ik inga op de gekozen competenties, hoe ik deze heb voltooid en reflecteer ik op hoe het project is verlopen, de samenwerking met het bedrijf en hoe ik als individu ben gegroeid door dit afstudeerproject. Vervolgens analyseer ik het product, identificeer ik de sterke en zwakke punten ervan en belicht ik de belangrijkste inzichten die ik tijdens het onderzoek en de ontwikkeling van het product heb opgedaan. Ten slotte presenteer ik de conclusie uit het adviesrapport.

Conclusie opdracht

Aan het einde van mijn afstudeerperiode staat de app op een punt waarin bijna alle must-requirements succesvol zijn geïmplementeerd en gerealiseerd. **In het bijgeleverde requirementsanalyse document is te zien welke requirements zijn afgekomen en welke niet.** De app biedt momenteel een aantal kernfunctionaliteiten die gericht zijn op het inloggen, inzien van zonnepaneel-data, beheren en verzenden van tickets en het ontvangen en versturen van meldingen.

De gebruikers hebben de mogelijkheid om in te loggen in de app om de gegevens van hun zonnepaneel-systeem te bekijken, inclusief details zoals verbruik, productie, foutmeldingen en historische gegevens. Daarnaast kunnen ze weergegevens inzien en meldingen ontvangen, waaronder waarschuwingen, informatieberichten en tickets. Installateurs hebben toegang tot de app om de details van hun klanten te kunnen zien en tickets van gebruikers te kunnen zien en behandelen. Ze kunnen ook klanten toevoegen en afspraken in hun agenda plaatsen door tickets te accepteren en toe te wijzen.

Wat betreft de non-functionele requirements heeft de app een bruikbaarheidsgericht ontwerp met een dashboard-achtige lay-out. De backend van de app is ontwikkeld met Hasura, wat bijdraagt aan de onderhoudbaarheid van het systeem. De frontend is gebouwd met behulp van React Native, waardoor het gemakkelijker is om het systeem te onderhouden en uit te breiden. Daarnaast is minimaal één externe API geïntegreerd, waardoor de app kan communiceren en gegevens kan uitwisselen met externe systemen.

Hoewel de app op een goed niveau staat met betrekking tot de must-requirements, zijn er nog enkele could- en should-requirements die niet zijn afgerond. Deze requirements waren niet-essentieel maar kunnen wel waarde toevoegen aan de app in termen van functionaliteit en mogelijkheden. In het vervolgtraject kan er aandacht worden besteed aan het implementeren van de overgebleven requirements en het verbeteren van de zwakkere punten van de app, want hoewel de app over een aantal belangrijke sterke functionaliteiten beschikt zijn er ook nog wel een aantal zwakkere punten waar aandacht aan besteed moet worden. Het ticket- en meldingsysteem kan efficiënter en soepeler en uiteraard moet er voor gezorgd worden dat meerdere API’s geïntegreerd worden met de app. In het adviesrapport wordt uitgebreider besproken wat beter kan in de app en waar de meeste prioriteit op ligt.

Over het algemeen is de app op dit punt een solide basis voor verdere ontwikkeling en verbetering. Het voldoet aan de belangrijkste functionaliteiten en biedt een gebruiksvriendelijke gebruikerservaring

Advies

In de bijlage is mijn adviesrapport te vinden. Hierin bespreek ik welke toevoegingen en aanpassingen de app kan gebruiken in de toekomst. De conclusie uit dat rapport heb ik in dit verslag geplaatst. **Voor details over het advies raad ik aan het adviesrapport te lezen.**

Een belangrijke aanbeveling is het integreren van meerdere API's, waardoor de app verschillende soorten apparaten van verschillende bedrijven kan ondersteunen. Bij elke gebruiker moet worden gecontroleerd bij welk bedrijf zij horen, zodat de juiste API's kunnen worden geïntegreerd om toegang te krijgen tot specifieke gegevens en functionaliteiten van die apparaten. Daarnaast moet het huidige systeem verbeterd worden, waarbij de site ID dynamisch wordt bepaald op basis van de gebruikersgegevens. Op deze manier kan relevante informatie worden verkregen zonder dat er een statische site ID nodig is in de URL's. Dit zorgt voor een meer flexibele en schaalbare aanpak bij het toegankelijk maken van de gegevens van verschillende gebruikers. Een geavanceerd meldingssysteem is ook een belangrijke aanbeveling. Gebruikers moeten meerdere soorten waarschuwingen kunnen ontvangen, zoals systeem statussen, kritieke foutmeldingen en algemene informatie. Dit stelt gebruikers in staat om snel te reageren op problemen en mogelijke fouten op te lossen, waardoor de betrouwbaarheid en efficiëntie van hun groene apparaten wordt verbeterd. Het archiveren van tickets is een functie die klanten in staat stelt om een overzicht te behouden van hun vorige communicatie met installateurs. Door tickets te archiveren, kunnen klanten eerdere tickets eenvoudig raadplegen, wat de communicatie en opvolging van problemen met installateurs vergemakkelijkt. Voor installateurs is het belangrijk dat ze tickets kunnen beheren door ze te accepteren, af te wijzen of toe te wijzen aan zichzelf of andere installateurs. Dit geeft hen controle over de toewijzing van verantwoordelijkheden en zorgt ervoor dat tickets efficiënt worden afgehandeld. Daarnaast moeten installateurs een overzicht hebben van al hun klanten en in staat zijn om waarschuwingen van de apparaten van die klanten in te zien, waardoor ze proactief kunnen reageren op mogelijke problemen. Om klanten te voorzien van geautomatiseerde rapporten, moeten maandelijkse rapporten worden gegenereerd waarin de prestaties van hun apparaten worden samengevat. Deze rapporten kunnen belangrijke statistieken, zoals energieproductie en besparingen, bevatten. Hierdoor kunnen klanten de prestaties van hun apparaten eenvoudig volgen en evalueren. Het aanbieden van instellingen, zoals taalvoorkeuren, meldingsvoorkeuren en wachtwoordbeheer, draagt bij aan een gepersonaliseerde gebruikerservaring. Gebruikers moeten in staat zijn om de app aan te passen aan hun voorkeuren en behoeften, waardoor ze de app optimaal kunnen gebruiken. Bovendien moet de app de mogelijkheid bieden om de stijl aan te passen aan de huisstijl van installateurs. Dit omvat het aanpassen van kleuren, logo's en andere visuele elementen, waardoor de app een professionele uitstraling krijgt die consistent is met het merk en de identiteit van het installatiebedrijf.

Om ervoor te zorgen dat de app schaalbaar en goed presterend is, is het van belang om de technische infrastructuur te optimaliseren. Dit omvat het implementeren van bijvoorbeeld cachingmechanismen en een schaalbare database. Door deze maatregelen kan de app blijven functioneren en presteren, zelfs bij een groeiend aantal gebruikers en gegevens. Bij het ontwikkelen en implementeren van de app is het ook belangrijk om kostenoverwegingen in acht te nemen. Het opstellen van een realistisch budget en het anticiperen op ontwikkelingskosten, infrastructuurkosten, onderhouds- en ondersteuningskosten, licenties en andere relevante kostenfactoren is van groot belang om een succesvolle en kostenefficiënte implementatie te waarborgen.

Met deze aanbevelingen kunnen verdere ontwikkelingen en verbeteringen worden doorgevoerd in EnergyFlow met de ondersteuning van een groter ontwikkelingsteam. De huidige opzet van de app vormt een redelijke basis, en met de voorgestelde aanpassingen kan de app worden uitgebreid om uiteindelijk het gewenste doel te bereiken.

Conclusie proces

Software Engineering – Analyseren

Een belangrijke competentie binnen het vakgebied van software engineering is het vermogen om grondig te analyseren en te begrijpen wat er nodig is om een succesvol project te realiseren. Voor het aantonen van deze competentie op niveau 3 heb ik mij gericht op het uitvoeren van een requirementsanalyse, het opstellen van requirements en het uitvoeren van een risicoanalyse.

Voorafgaand aan het ontwikkelingsproces van de app heb ik een uitgebreid onderzoek uitgevoerd. Dit omvatte het in kaart brengen van de behoeften en verwachtingen van de gebruikers door middel van verschillende onderzoeksmethoden, waaronder interviews, enquêtes en gebruikerstesten. Door middel van deze analyses ben ik in staat geweest om de functionele en non-functionele requirements van de app te identificeren en te documenteren. Daarnaast heb ik de requirements voor het project opgesteld. Hierbij heb ik samengewerkt met personen binnen het bedrijf om er voor te zorgen dat de requirements volledig, consistent en begrijpelijk waren. Om potentiële risico's tijdens het requirementsproces te identificeren en beheren, heb ik ook een risicoanalyse uitgevoerd. Dit omvatte het identificeren van mogelijke risico's, zoals onduidelijke of inconsistente requirements, scope creep en veranderende gebruikersbehoeften. Door de impact en waarschijnlijkheid van deze risico's te beoordelen, kon ik prioriteit geven aan de belangrijkste risico's. De risicoanalyse bied een waardevol inzicht in potentiële uitdagingen.

Deze activiteiten op het gebied van requirementsanalyse, het opstellen van requirements en risicoanalyse hebben bijgedragen aan het aantonen van de competentie "Software Engineering - Analyseren" op niveau 3. Door het uitvoeren van gedegen analyses en het identificeren en beheren van risico's, ben ik in staat geweest om een sterke basis te leggen voor het verdere ontwikkelingsproces.

Software Engineering – Realiseren

Misschien wel de belangrijkste competentie binnen software engineering is het vermogen om ontwerpen daadwerkelijk om te zetten in werkende applicaties. Voor het aantonen van deze competentie op niveau 3 heb ik me gericht op het realiseren van het ontwerp dat gebaseerd is op de behoeften en voorkeuren van de gebruikers. Dit heb ik bereikt door het ontwikkelen van een functionele applicatie.

Met behulp van moderne ontwikkeltools en frameworks ben ik erin geslaagd om het ontwerp van de app om te zetten in een werkende mobiele applicatie. Ik heb gebruik gemaakt van relevante programmeertalen en technologieën, zoals JavaScript en React Native, om een goed gestructureerde en schaalbare codebase te creëren. Door mijn programmeervaardigheden toe te passen, kon ik de benodigde functionaliteiten implementeren, zoals gebruikersauthenticatie, gegevensopslag en interacties met de gebruikersinterface. Om echte en actuele gegevens in de app te integreren, heb ik gebruikgemaakt van Hasura en een externe API. Hasura heeft me in staat gesteld om eenvoudig een backend-infrastructuur op te zetten en te beheren, waardoor ik efficiënt met gegevens kon omgaan. Daarnaast heb ik een externe API geïntegreerd om externe gegevensbronnen te raadplegen en gegevens op te halen die relevant zijn voor de functionaliteit van de app.

Het realiseren van het ontwerp in de vorm van een werkende mobiele applicatie, het implementeren van echte gegevens via Hasura en een externe API hebben bijgedragen aan het aantonen van de competentie "Software Engineering - Realiseren" op niveau 3. Door deze activiteiten heb ik bewezen dat ik in staat ben om effectieve softwareoplossingen te bouwen die voldoen aan de vereisten en verwachtingen van de gebruikers.

Gebruikers Interactie – Ontwerpen

Een competentie binnen gebruikersinteractie is ontwerpkeuzes te kunnen maken die aansluiten bij de behoeften en verwachtingen van de gebruikers. Voor het aantonen van deze competentie op niveau 3 heb ik me gericht op het omzetten van de resultaten van mijn uitgebreide onderzoek naar een ontwerpprototype dat nauw aansluit bij de wensen van de gebruikers. Vervolgens heb ik het ontwerp getest met potentiële gebruikers en het ontwerp op basis van die resultaten verder aangepast.

Op basis van mijn onderzoek naar de behoeften en voorkeuren van de gebruikers, heb ik een ontwerpprototype gecreëerd dat zoveel mogelijk overeenkwam met hun verwachtingen. Dit omvatte het visualiseren van de gebruikersinterface, het definiëren van de interactiestromen en het creëren van een gebruiksvriendelijke gebruikerservaring. Ik heb rekening gehouden met de principes van gebruikerservaring (UX) en user interface design (UI) om een ontwerp te creëren dat zowel functioneel als esthetisch aantrekkelijk is. Om de effectiviteit van het ontwerp te valideren, heb ik tests uitgevoerd met potentiële gebruikers. Deze tests hebben me waardevolle inzichten gegeven over hoe gebruikers met de app omgingen, welke elementen duidelijk waren en welke verbeteringen nodig waren. Door deze feedback te verzamelen, kon ik het ontwerp aanpassen en verfijnen om beter aan te sluiten bij de verwachtingen en het gedrag van de gebruikers.

Het proces van het testen van het ontwerp met potentiële gebruikers en het aanpassen ervan op basis van hun feedback is belangrijk geweest bij het aantonen van de competentie "Gebruikersinteractie - Ontwerpen" op niveau 3.

Gebruikers Interactie – Realiseren

Een andere belangrijke competentie binnen gebruikersinteractie is het ontwerp dat voortkomt uit gebruikersonderzoek en -testen daadwerkelijk te kunnen realiseren in een functionele mobiele applicatie. Voor het aantonen van deze competentie op niveau 3 heb ik me gericht op het implementeren van het ontwerp in een mobiele applicatie, waarbij ik alle wensen en eisen van de gebruikers heb meegenomen en geïmplementeerd.

Het ontwerp dat voortkwam uit het gebruikersonderzoek en de gebruikerstesten is vertaald naar een werkende mobiele applicatie met een functionele gebruikersinterface. Hierbij heb ik ervoor gezorgd dat alle functionaliteiten, interacties en visuele elementen in overeenstemming waren met de wensen en eisen van de gebruikers. Tijdens het ontwikkelingsproces heb ik een aantal feedbackloops georganiseerd om ervoor te zorgen dat de implementatie voldeed aan de verwachtingen van de gebruikers. Het realiseren van het ontwerp door middel van implementatie in een mobiele applicatie, waarbij ik alle wensen en eisen van de gebruikers heb meegenomen en geïmplementeerd, heeft bijgedragen aan het aantonen van de competentie "Gebruikersinteractie - Realiseren" op niveau 3.

Zelfreflectie

Het schrijven van deze zelfreflectie betekend dat het einde van mijn afstuderen in zicht is en dat het tijd is om terug te kijken op mijn afstudeerperiode en om te kijken naar wat goed ging, wat minder goed ging en of mijn doelen zijn behaald. Om te beginnen wil ik het graag hebben over die doelen. Aan het begin van mijn afstuderen had ik een aantal doelen, zoals natuurlijk het behalen van deze afstudeerstage maar ook het groeien in een aantal persoonlijke aspecten. Ik wou graag leren werken in een professionele omgeving zonder corona restricties, uitvogelen of ik een groter of kleiner bedrijf fijner zou vinden, mijn communicatie en programmeervaardigheden verbeteren maar vooral ook uitvinden waar mijn hart nou eigenlijk echt lag. Voordat ik begon met deze stage wist ik nog niet goed welke kant ik na deze studie op wou gaan. Ik wist dat ik niet de backend kant op wou gaan, al zou ik full-stack niet zo erg vinden. Een frontender worden klonk niet verkeerd, maar werd ik blijkbaar toch ook weer niet helemaal mega enthousiast van. Na mijn vorige stage wist ik dat ik graag een afstudeerstage wou doen met een project waar ik echt gelukkig van zou worden, anders zat er voor mij eigenlijk weinig waarde in. Ik wou graag iets doen met ontwerp en interactie, maar tijdens mijn studie is dit erg weinig aan bod gekomen. Pas tijdens het laatste deel vóór deze stage, specialisatie, kreeg ik de kans hier iets mee te doen d.m.v. het vak Design Essentials. Ik ging dan ook effectief op zoek naar een stage waar ik dit kon toepassen. Die kans kreeg ik bij deze stage, maar de vraag was nog wel of ik hier genoeg mee kon doen om een complexe opdracht in te leveren waarbij ik ook nog eens gebruikersinteractie competentie(s) op niveau 3 kon aantonen. Ik wist dat dit niet makkelijk ging zijn en heb dan ook hard gewerkt om een onderzoek op te stellen dat hopelijk liet zien dat er genoeg werk zit achter het gebruikersinteractie gedeelte. Ik was bang dat mijn onderzoek niet uitgebreid en dus niet relevant genoeg zou zijn, maar met 175 antwoorden op mijn gebruikers enquête en 23 antwoorden op mijn installateur enquête en natuurlijk alle andere stappen die ik heb ondernomen vind ik dat het onderzoek zich zeker wel uitgebreid mag noemen. Ik vond het lastig om te laten zien dat zo'n onderzoek veel heeft bijgedragen aan het ontwerp van de app, omdat je naar mijn idee niet gemakkelijk kan laten zien hoeveel denk- en overweegwerk er achter zit ten opzichte van fysiek resultaat zoals een complexe backend. Ik heb geprobeerd de resultaten van het onderzoek zo veel mogelijk om te zetten naar een ontwerp. Ik vind dat dit redelijk goed gelukt is, ik heb veel van de belangrijkste wensen kunnen toevoegen aan het ontwerp van de app. Daarnaast heb ik me natuurlijk ook gefocust op het implementeren van de applicatie. Na het onderzoek gaf ik mezelf 8 weken om de app te ontwikkelen. Het belangrijkste gedeelde van deze opdracht was natuurlijk het ontwerp, en ik heb me daarom ook aan het begin vooral gefocust om dit ontwerp te implementeren. Ik gaf mezelf steeds één sprint om een deel van de app te doen. Sprint 1: ontwerp klant, sprint 2: ontwerp installateur, sprint 3: functionaliteiten klant, sprint 4: functionaliteiten installateur. De eerste helft ging erg goed, ik implementeerde als een speer het ontwerp voor beide type gebruikers. Tijdens de tweede helft was het het idee dat ik de functionaliteiten zou implementeren met behulp van verschillende API's. Echter werd het verkrijgen van überhaupt de eerste API key van SolarEdge een paar keer uitgesteld vanwege ziekte of andere dingen die in de weg kwamen te zitten. Daarnaast was het nog erg zoeken hoe ik dit überhaupt ging doen, aangezien het werken met externe API's en met Hasura nog helemaal nieuw voor mij was. Hierdoor kon ik pas laat in sprint 3 beginnen met het implementeren van functionaliteiten. Ik heb toen de keuze gemaakt om het bij één API te houden, omdat ik graag een zo compleet mogelijke eerste opzet wou laten zien in plaats van een halfslachtig product dat wél meerdere API's heeft geïntegreerd. Ondanks de vertraging heb ik me daarna wel opgepakt. Tijdens momenten dat ik niet kon werken aan de app heb ik gewerkt aan mijn verslag en andere documentatie. Ik had in eerste instantie gepland dit op andere momenten te doen, maar ik realiseerde me gelukkig snel dat dat ook eerder kon. Toen ik eenmaal de API key in handen had en doorhad hoe Hasura werkte kon ik de functionaliteiten redelijk snel implementeren. Helaas wou ik in sprint 4 een betere API key ontvangen om een aantal functionaliteiten te verbeteren, maar wederom

door ziekte (niet van mijn kant) kon dit niet doorgaan. Toch ben ik wel redelijk blij met wat ik heb kunnen implementeren. Ik heb een aantal nieuwe technieken kunnen toepassen en heb alles volledig zelfstandig gedaan. Tijdens mijn opleiding was alles altijd in groepjes, dus dit is de eerste keer dat ik een full-stack applicatie volledig in mijn eentje heb ontwikkeld. Omdat ik er toch voor had gekozen om met scrum te werken heb ik mijn hoofd koel kunnen houden door steeds een overzicht te hebben van mijn taken en niet de weg kwijt te raken. Het eindresultaat is niet helemaal wat ik aan het begin voor ogen had, maar ik weet dat dit niet volledig mijn schuld is. Daarnaast had ik waarschijnlijk ook een iets te onrealistisch beeld van wat ik had kunnen neerzetten. Ik ben daarom redelijk trots op wat ik wél heb kunnen neerzetten. Toch zijn er ook echt wel wat dingen die ik zelf beter had kunnen doen. Zo had ik eerder naar die API keys mogen vragen zodat ik deze direct beschikbaar had gehad tijdens mijn ontwikkelfase, en ik had me eerder mogen verdiepen in de nieuwe technieken zoals Hasura en GraphQL, zodat hier minder tijd aan verloren was gegaan. Ook een aantal niet-technische dingen hadden beter gekund. Hoewel ik dit keer niet bang was om feedback te vragen had ik wel meer mijn hoofd er bij kunnen houden om mijn documenten eerder op te sturen voor tussentijdse feedback momenten. Ik ben dit een aantal keer vergeten op tijd te doen, wat me waardevolle feedback heeft gekost. Dit is niet alleen niet handig maar ook gewoon niet professioneel en laat het doen lijken alsof het me niet boeit, terwijl ik het wel erg belangrijk vind. Voor de volgende keer schrijf ik duidelijker voor mezelf op wanneer ik iets precies inlever. Wel vind ik dat mijn communicatie en motivatie is verbeterd ten opzichte van de vorige stage. Ik communiceerde gemakkelijker met mensen om me heen en was niet meer zo bang om vragen te stellen. Ook heb ik goed doorgewerkt gedurende mijn hele stageperiode en heb ik dingen niet lopen uitstellen. Dit neem ik zeker mee naar het leven na mijn studie. Ook weet ik dankzij deze stage waar ik graag wil zijn na mijn studie. Ik werk liever in een kleiner bedrijf en wil graag te werk gaan met UI en UX, maar wil ook het technische aspect kunnen doen. Ik ga nog even moeten nadenken waar dit kan en wat ik precies wil, en of ik gelijk wil werken of dat ik nog een specialisatie doe, maar ik weet in elk geval welke kant ik uit wil gaan. Ik ben erg blij dat ik mijn afstuderen bij ProfitFlow heb mogen doen en dat ik de kans heb gekregen een opdracht te mogen doen die in mijn straatje valt. Ik hoop dat het genoeg gaat zijn, maar ik weet in elk geval voor mezelf dat ik er genoeg tijd en energie in heb gestopt en dat ik trots mag zijn op mijn prestaties. De afstudeerperiode bij ProfitFlow was een rollercoaster, maar ik ben blij dat ik in het karretje heb durven stappen en de rit heb kunnen afmaken.

Figuren en tabellen

Bibliography

ISO. (2018). *ISO 9241-11:2018(en) Ergonomics of human-system interaction.* ISO.