



Chargehub

Realisatie

Bachelor in de Toegepaste Informatica
Keuserichting APP

Sebastian Van Grieken

Academiejaar 2023-2024

Campus Geel, Kleinhoefstraat 4, BE-2440 Geel

Inhoudstafel

1 Inleiding.....	5
1.1 Stage bedrijf	5
1.2 Opdracht.....	5
2 Planning	6
3 Native applicaties	7
4 Multiplatform	8
4.1 Flutter	8
4.2 Kotlin.....	8
4.2.1 Compose multiplatform.....	9
5 Project Structuur	10
5.1 Firestore	10
5.2 Local database	11
5.3 Firebase Authentication	11
5.4 Data flow	11
5.5 ViewModel.....	12
5.6 UI	12
6 Application flow.....	13
6.1 Landing page	13
6.2 Registration.....	13
6.3 Logging in	Error! Bookmark not defined.
6.4 Wachtwoord vergeten	15
6.5 Biometrics	16
6.6 Home page.....	17
6.7 Quick reservation.....	18
6.8 Edit reservation	18
6.9 Kalender.....	19
6.10 Notificaties	19
7 Slot.....	20

1 INLEIDING

1.1 Stage bedrijf

De Afgelopen 14 weken heb ik het genoeg gehad om stage te mogen lopen bij Wisemen. Wisemen is een digital agency gelegen in Diepenbeek dat hun klanten helpt leads te boosten, bedrijfsefficiëntie te verhogen en customer experience te verbeteren. Ik ben hier aan de slag gegaan als Android developer.

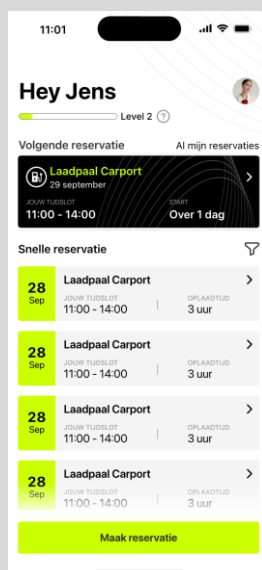


1.2 Opdracht

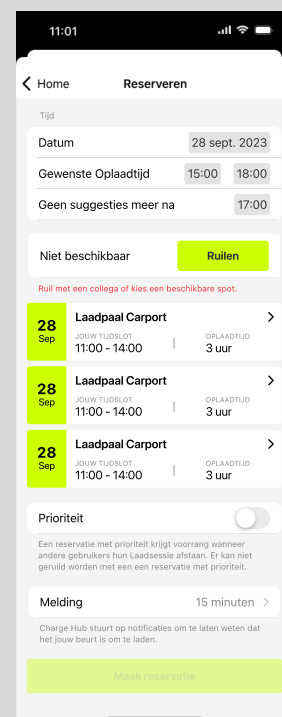
Aan het begin van mijn stage heb ik de opdracht gekregen om samen met een mede stagair een mobiele applicatie te maken. De applicatie zou binnen het bedrijf gebruikt worden om een reservatie te kunnen maken voor één van de drie laadpalen gelegen binnen het bedrijf. Vooraf was hier geen regeling voor een werd dit gedaan door onderling berichten te sturen. Buiten het makkelijker te maken voor werknemers om hun auto op te laden dient de applicatie ook als proof of concept, de applicatie moest namelijk gemaakt worden met kotlin-multiplatform, een technologie die nog onbekend was voor Wisemen. Ik en mijn collega moesten hiervoor grondig research doen en onze bevindingen documenteren voor mogelijks verder gebruik binnen het bedrijf. Meer uitleg over kotlin-multiplatform komt verder in dit document. Hier onder vindt u alvast een paar voorbeelden van de designs die ons gegeven zijn om te kunnen visualiseren hoe het project er uiteindelijk zou moeten uitzien.



Landing scherm



Home scherm



Reserveer scherm

2 PLANNING

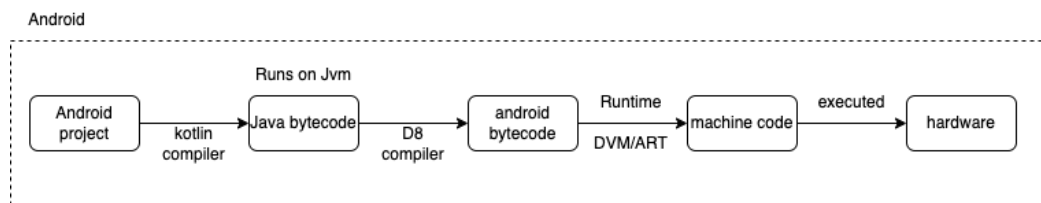
Voor we aan de applicatie zijn begonnen hebben we samen gezeten om een gezamenlijke planning te maken. We hadden besloten om de applicatie op te splitsen per scherm en waar nodig hebben we de functionaliteit en de UI ook nog opgesplitst. Doorheen het project hebben we zelf gekozen welke functionaliteit we gingen maken, met kort overleg natuurlijk. De planning zelf lag ook niet vast in steen, deze diende voor ons als lijndraad om op schema te blijven. De functionaliteiten die we besloten vooraf in te planning en die zeker in de applicatie moesten zitten waren: het landing scherm, de login en registratie schermen, het hoofd scherm en een scherm om je reservatie aan te passen. Wanneer deze functionaliteiten klaar waren zijn we verder gegaan met sprint per sprint, met overleg met onze begeleiders het werk in te plannen. Doorheen het project hebben we beiden ook individueel tijd ingepland om onderzoek te doen over kotlin multiplatform, of een specifieke technologie die hiermee te maken had. Het grootste deel van de technologieën hebben we dan ook beiden onderzocht. Op het eind van de stage hadden we beiden grootendeels individueel aan onze functionaliteiten gewerkt. Wanneer het nodig was hebben we elkaar ook wel geholpen.

		26 feb 24'	4 mrt 24'	11 mrt 24'	18 mrt 24'	25 mrt 24'	1 apr 24'	8 apr 24'	15 apr 24'	22 apr 24'	29 apr 24'	6 mei 24'	13 mei 24'	20 mei 24'	27 mei 24'
Taak omschrijving	geplande tijd	week 1	week 2	week 3	week 4	week 5	week 6	week 7	week 8	week 9	week 10	week 11	week 12	week 13	week 14
Onboarding	1 week														
Project setup	1 week														
Landing page android	1 dag														
Landing page ios	1 dag														
Login page android/ios	~3 dagen														
Paswoord reset page android/ios	~3 dagen														
Registration page android/ios	2 week														
Home page android/ios	2 week														
Quick reservation	1 week														
Calendar	~4 weken														
Biometric authentication	1 week														
Notifications	~2 dagen														
Edit reservation	2 weken														
research															
documentation															
		gepland	extra tijd	doorlopend											

Overzicht van wat we samen hebben kunnen bereiken

3 NATIVE APPLICATIONS

Om te begrijpen wat ik in de rest van het document ga uitleggen ga ik je eerst wat moeten uitleggen over android native. Als we een mobiele applicatie maken hebben we vandaag de dag daar meerdere keuzes voor: native, Flutter, React native en sinds kort Kotlin multiplatform, meer daarover komt later. Android native en iOS native is de manier om een android applicatie te maken met de tools die google en apple ondersteunen. In het verleden werden android applicaties geschreven in java samen met xml. Sinds JetBrains in 2017 kotlin heeft uitgebracht is dit een tweede manier om applicaties te maken. Kotlin is ook al snel de voorkeur geworden over java voor android development.



Stappen van code naar applicatie die draaid op hardware

Om de werking van kotlin multiplatform later in dit document beter te kunnen begrijpen lijkt het mij een goed idee om even te stoppen bij hoe android native applicaties werken. Wanneer je een applicatie hebt gemaakt met kotlin, hoe werkt deze dan op de gebruikers hun toestellen? Om het begrijpelijker te maken ga ik dit proces omgekeerd uitleggen, en dus beginnen we bij de cpu. Cpu's hebben meerderer architecturen zoals x86 en ARM. Beide gebruiken machine code in binair formaat om te draaien, maar deze machine code ziet er voor elke cpu architectuur er anders uit. Je kan dus voor x86 en ARM cpu's andere machine code hebben die hetzelfde doet. Om tot deze machine code te komen



gebruikt elk platform zijn eigen compiler om platform specifieke bytecode om te zetten. Als we dan nog één stap terug gaan dan zien we dat voor android, android bytecode afkomstig is van java bytecode die ook gecompileerd is. En dan de laatste stap in de ketting, java bytecode komt uit de compiler als we het android project geschreven kotlin compilen. Dit proces is voor iOS vergelijkbaar met als verschil dat zij een tussenstap overslaan, een iOS project geschreven in swift wordt rechtstreeks naar IOS bytecode gecompileerd waarnaar het naar native machine code wordt gecompileerd.

4 MULTIPLATFORM

Multiplatform is een concept dat in theorie het efficiënter zou moeten maken voor developers om een applicatie op de markt te krijgen voor android en iOS. In praktijk loopt het niet altijd zo vlot en zijn er trade offs voor elke oplossing. Wanneer je een applicatie wilt maken zit je met een bottleneck. Je wilt je applicatie op beide platformen krijgen, maar dit kost dubbel de mankracht. Hierdoor is zich men beginnen afvragen: is het mogelijk om een applicatie te maken voor beide platformen die je maar één keer moet schrijven, dit vermindert de nodige mankracht wat op zijn beurt ook de kosten vermindert. Hiervoor zijn vandaag de dag drie mogelijkheden: flutter, react native en kotlin multiplatform.

4.1 Flutter

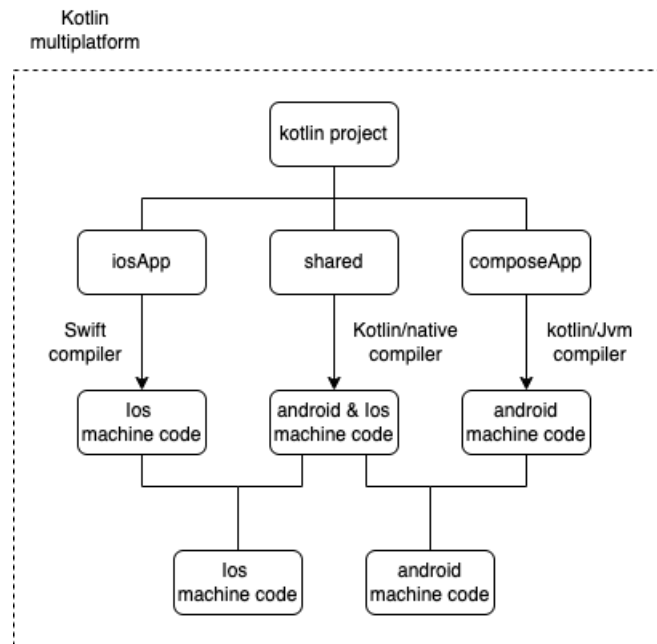
Flutter is een project van google dat in 2017 op de markt kwam. Het was bedoeld als een alternatieve manier om mobiele applicaties te maken. Flutter is ontworpen om het makkelijker te maken om applicaties te maken voor meerdere platformen, waaronder android en iOS. Dit wordt gerealiseerd door gebruik te maken van dart, een programmeer taal die gecompileerd kan worden naar native machine code. Door gebruik te maken van dart maakt flutter het mogelijk om een applicatie één keer te schrijven en op beide platformen te kunnen draaien. Natuurlijk heeft elk voordeel zijn nadeel. Als je flutter wilt gebruiken moet je een heel nieuwe taal leren. Wanneer je al ervaring hebt met native devopment met kotlin en swift moet jezelf en/of je team een volledig nieuwe taal leren. Performance is niet altijd iets waar developers naar kijken en dat hoeft in een groot deel van de applicaties ook zeker niet, maar wanneer je werkt met een applicatie waarvan de performance wel belangrijk is dan ga je met flutter tekort komen als je gaat vergelijken met native. Als laatste ga je ook extra werk moeten doen om platformspecifieke functies te gebruiken. Momenteel heeft flutter een goed assortiment aan libraries, maar dit wilt niet zeggen dat je altijd en kant-en-klare oplossing gaat vinden.



4.2 Kotlin

Kotlin multiplatform is een poging van JetBrains om cross-platform applicaties te maken te vereenvoudigen. Kmp* maakt gebruik van een compiler om kotlin code te compilen naar native machine code, zoals flutter dit ook doet. Kmp maakt gebruik van een unieke folder structuur en zijn compiler om het developen van een applicatie voor beide platformen gemakkelijker te maken. Wanneer je een kmp applicatie wilt maken heb je de keuze om het volledige project met kotlin te maken, of de UI native te schrijven maar alles dat met data te maken heeft in kotlin te maken. Ik en mijn collega hebben ervoor gekozen om de UI van onze applicatie native te schrijven. De reden voor deze keuze wordt verduidelijkt bij het volgende deel.

Zoals je kunt zien op het onderstaande diagram bestaat het project uit 3 folders: iOSApp, composeApp en shared. De shared folder maakt gebruik van de kotlin/native compiler om kotlin code naar native machine code om te zetten. Deze folder bevat al de code die instaat voor het handelen met data, en is volledig geschreven in kotlin. De iOSApp en composeApp worden gebruikt om de UI code in op te slaan. Deze folders maken gebruik van swift voor de iOS UI en kotlin voor de android UI. Verder kan hier ook code in terug vinden om native functies te gebruiken waarvan het ander platform geen oplossing voor heeft. Zoals je kunt zien worden de native folders ook apart gecompileerd met hun eigen native compiler. De voordelen van deze werking is dat je native performance krijgt. Ook kan je gebruik maken van je bestaande kennis van kotlin en android om applicaties te schrijven.



Folder structuur van een kmp project met native UI

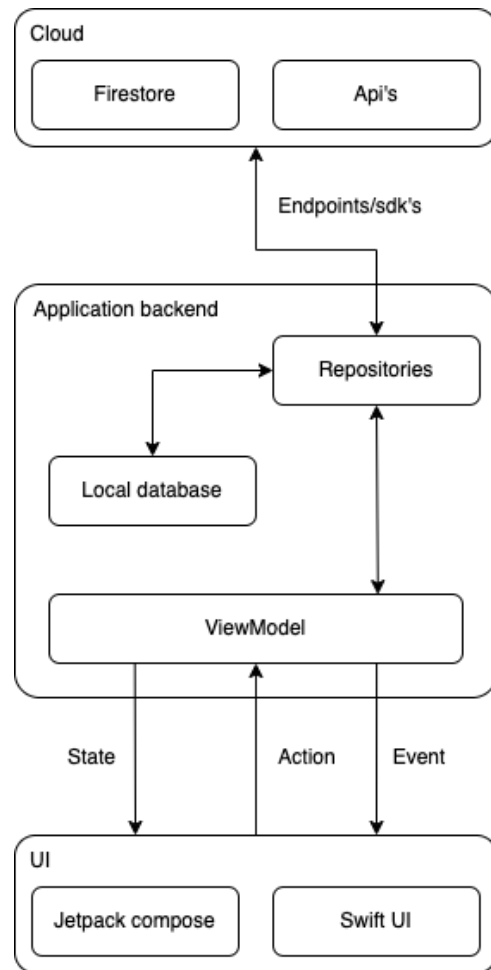
4.2.1 Compose multiplatform

Compose multiplatform is een UI library voor kmp projecten. In een native android applicatie maak je gebruik van jetpack compose om het maken van UI te versimpelen. Jetpack compose is een toolkit met vooraf gedefinieerde componenten die je kunt gebruiken om je UI op te bouwen. Compose multiplatform zoals het woord zegt heeft hetzelfde nut maar kan gebruikt worden in een kmp project. Het voordeel hiervan is dat je geen swift code meer moet schrijven, maar volledig met kotlin code kunt werken. Een groot nadeel en de reden waarom ik en mijn collega ervoor gekozen hebben om dit niet te gebruiken is dat dit nog een vroege fase van ontwikkeling zit en daardoor hierover zeer weinig informatie te vinden valt.



5 PROJECT STRUCTUUR

Onafhankelijk van het project waar je aan werkt of de programmeer taal die je gebruikt is het altijd goed om een duidelijke structuur te hebben in je project. Hiervoor zijn een best practices gemaakt. Dit zijn regels die je kunnen helpen om je project op een handelbare manier in te delen. Deze regels volgen is geen vereiste, maar maakt het makkelijker voor jezelf en anderen om je code te verwerken. Daarom hebben mijn collega en ik hier ook gebruik van gemaakt. Android heeft hier zelf ook een library voor gemaakt om het gebruiken van je data in de UI gebruiksvriendelijker te maken. Zoals je kunt zien beginnen we bij de database. Deze kan eender welke database zijn, maar wij hebben gekozen om met firestore te werken. De database gebruiken doen we dan via de repository even als de lokale database. Om dan de data aan de UI te geven maken we gebruik van Android's viewModel. Hoe dit werkt leg ik uit in een later deel. En als laatste hebben we dan de UI, datgene wat de gebruiker uiteindelijk ziet. Hiervoor gaan we jetpack compose voor android en swiftUI voor iOS gebruiken.



Structuur van functionaliteit

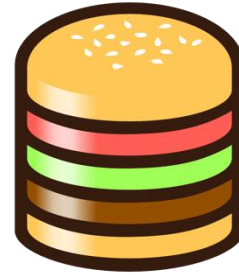
5.1 Firestore



Firestore is een document based database wat onderdeel is van google's firebase. Firebase heeft nog veel meer andere services zoals authentication, analytics, crashlytics en nog veel meer, maar voor ons project hebben we gebruik gemaakt van firestore en firebase authentication. De document based aard van firestore maakt dat het heel flexibel is met hoe het data opslaat. Firestore werkt met collecties en documents. Wanneer je al ervaring hebt met sql based databases, kun je deze collecties en documents vergelijken met tabels en rows respectievelijk. Firebase heeft sdk's voor veel platformen waaronder android en iOS. Dit maakt het gebruik van firestore dan ook zeer gemakkelijk. In kmp werkt dit iets anders, hiervoor hebben we gebruik gemaakt van een third-party library. Deze handelt als een koepel rond de standaard libraries en zorgt ervoor dat we deze kunnen gebruiken in de gedeelde code en niet in de native code, wat ons dubbel werk bespaart.

5.2 Local database

Een lokale database wordt in een mobiele applicatie gebruikt als eerste aanspreekpunt om data op te vragen. Het nut hiervan is dat het vele malen sneller is, de locale database is dan ook zeer handig wanneer je slecht bereik hebt. De locale database en firestore worden regelmatig gesynct, dit maakt dat je de applicatie toch nog kunt gebruiken wanneer je weining tot geen bereik hebt, en vanaf dat je terug bereik hebt kan firestore ook geüpdatet worden waardoor beide databases terug gelijk lopen. Voor de local database hebben we gebruik gemaakt van sqldelight. Voor native android development wordt aangeraden om room te gebruiken als local database. Room gebruikt sqldelight under the hood. Sqldelight was ook erg handig omdat dit direct met kmp werkt.



5.3 Firebase Authentication



Omdat we al gebruik maakten van firestore was de keuze om firebase authentication te gebruiken snel gemaakt. Authenticatie is voor elke developer een moeilijke taak, er zijn vandaag de dag veel goede keuzes voor services die authenticatie aanbieden, je zou het ook zelf kunnen schrijven. Authenticatie handelt ook met gevoelige data en moet dus waterdicht gemaakt zijn. Gelukkig is de firebase authenticatie service opnieuw zeer makkelijk om te gebruiken, en veiligheid kan gegarandeerd worden.

5.4 Data flow

We zouden rechtstreeks in onze viewModel firestore en api's kunnen aanroepen, maar dit wordt vaak niet gedaan. Bij programmeren gebruiken we vaak het separation of concerns principe. Dit wilt zeggen dat je je code gaat opsplitsen in files waarbij elke file een duidelijke en afzonderlijk probleem oplost. Wanneer we met een application backend werken doen we dit ook en gaan we een repository gebruiken. Deze kan dan aangeroepen worden door de viewModel. In dit geval heeft de repository functies gedefinieerd met één duidelijke opdracht: data ophalen of versturen naar de database, en deze functies worden dan gebruikt op andere plaatsen in de code. Op deze manier hebben we het aansturen van api's en de cloud database aan een specifiek deel van de code toegewezen, zo weten we altijd waar we moeten zoeken en voorkomen we dat we code dupliceren.

5.5 ViewModel

Native android applicaties maken gebruiken van een viewModel om de UI gescheiden te houden van de data die gebruikt wordt. Het viewModel concept is iets dat in iOS niet gebruikt wordt. Dit is natuurlijk een probleem want dan zouden we om data naar onze UI te brengen voor android en iOS en andere oplossing moeten vinden. Gelukkig was er een zeer slimme man uit Nederland die ons probleem al heeft opgelost, die man heeft hiervoor een library gemaakt. Deze library zorgt ervoor dat we viewModels kunnen maken zoals we dat in native android zouden doen, en vervolgens kunnen we die dan ook in onze iOS UI gebruiken.

5.6 UI

Voor UI hebben we gekozen om met native UI te werken, jetpack compose voor android en swiftUI voor iOS. Zoals ik in een vorig deel al heb uitgelegd, vonden wij de documentatie voor compose multiplatform niet voldoende om met zekerheid dit te gebruiken. UI libraries voor mobile development maken gebruik van componenten, dit zijn UI elementen die de library vooraf heeft gedefinieerd. Deze componenten zijn de basis, en meestal genoeg om je volledige UI met op te bouwen.

6 APPLICATION FLOW

In de komende delen ga ik ik applicatie tonen, en laten zien wat ik en mijn collega samen hebben kunnen realiseren. Bij delen waar ik zelf aan gewerkt heb zal ik proberen meer uitleg te geven en ook even stil te staan bij de moeilijkheden en het proces van het maken van die feature.

6.1 Landing page

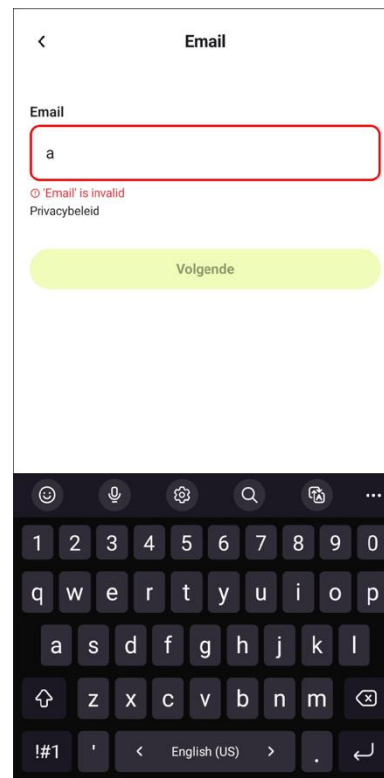
De landing page is de eerste pagina die je te zien krijgt als je de applicatie opent. Over deze pagina valt niet veel te vertellen, buiten dat je hier kan kiezen of je wilt inloggen of wilt registreren.

6.2 Registration

Als we dan in logische volgorde verder gaan, komen we op de eerste pagina van het registratieproces. Op deze pagina begin je met je email in te geven. Zoals je kan zien is het email adres ook voorzien van validatie. De gebruiker wordt op de hoogte gebracht dat het email adres niet valide is en de knop om verder te gaan kan ook niet meer gebruikt worden.



Landing scherm



Registratie: email scherm

Wanneer we dan een geldig email adres invoeren kunnen we naar de volgende pagina gaan. Op deze pagina kunnen we onze naam en password ingeven. Daarna kunnen we ook de privacy policy lezen en aanvinken dat we hiermee aakkoord gaan. Ook op deze pagina krijgen we validatie en kunnen we niet verder voor deze goed is.

Als we dan terug verder gaan, kunnen we op de volgende pagina de vin nummer van onze wagen ingeven. Deze wordt gebruikt om te kunnen verbinden met een api die de wagen status kent.

De paginas die hierop volgen vereisen geen input meer van de gebruiker, maar deze dienen voor een klein beetje info voor ze de app kunnen gebruiken, en te bevestigen dat ze meldingen willen.

Wanneer we dan op de laatste pagina terecht komen en al de info hebben gelezen, of niet, dan kunnen we verder klikken en komen we terecht op de home page. Wanneer we naar de home page navigeren gaan we tegelijkertijd een account aanmaken met de data die je hebt ingevuld. Hierbij gaan we eerst een firebase authentication account maken. Hier is alleen een email en paswoord aan verbonden. Het paswoord wordt ook niet door ons behandeld maar volledig door firebase. Dit garandeert de veiligheid van je gegevens. Hierna kunnen we ook de rest van de gegevens opslaan in een user document.

Registratie: vin nummer ingeven

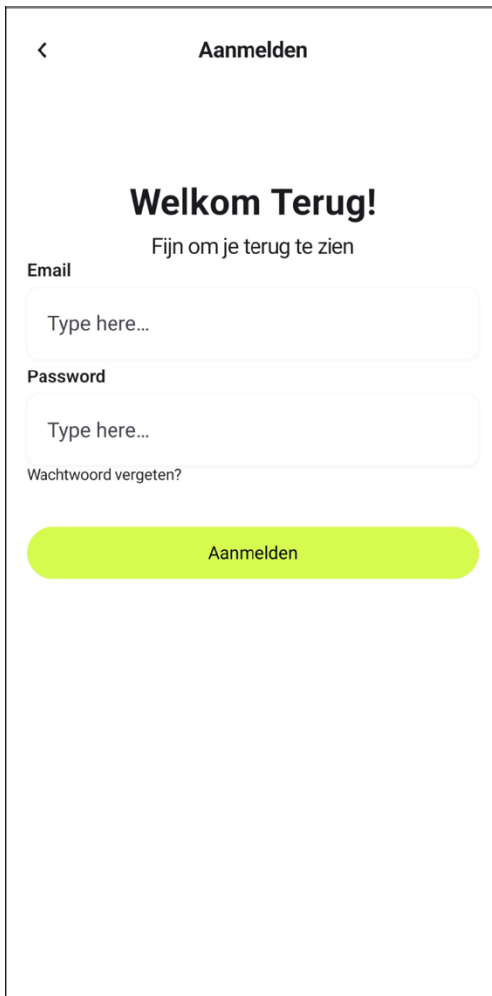
Registratie: algemene gegevens

6.3 Inloggen

De volgende volgende pagina die we gaan bespreken is er één waar ik volledig aan heb gewerkt, en dat is de login pagina. Deze is vrij simpel, voor deze pagina gebruik ik een custom component voor het email en wachtwoord veld. Dit staat ons toe om extra functionaliteit toe te voegen, zoals custom error stijl. De Firebase dependency maakt het gemakkelijk om authenticatie toe te voegen aan een project. Wanneer je je gegevens ingeeft worden deze opgeslagen in de state van het scherm. Wanneer je dan op de aanmeld knop klikt, wordt de login functie van firebase authenticatie gebruikt samen met de state om in te loggen. Hierna kunnen we eender waar in de applicatie ook zien wie ingelogd is.

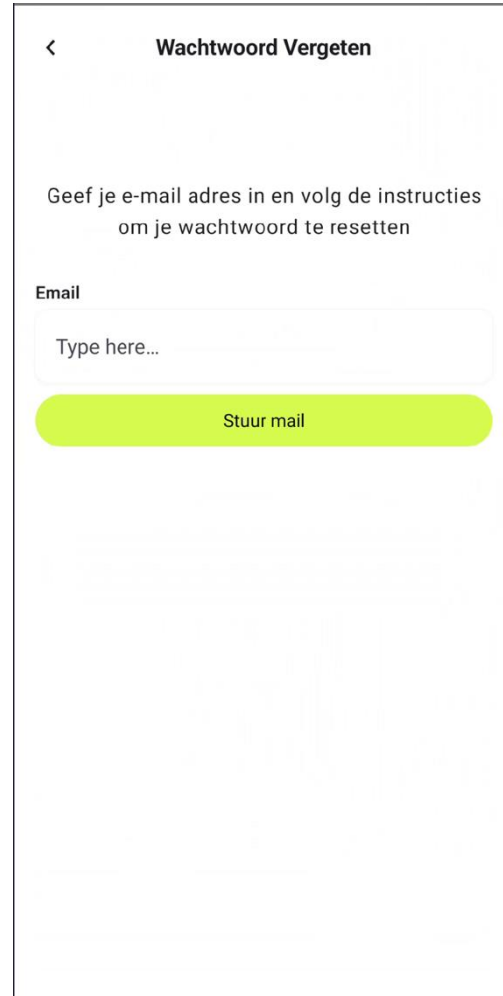
6.4 Wachtwoord vergeten

Als je goed hebt gekeken op het login scherm zag je een tekst onder het paswoord veld. Deze dient voor het geval dat je je paswoord bent vergeten. Het paswoord vervangen is ook een simpel proces: Je geeft je email adres in en klikt op de knop. Hierna krijg je een mail van firebase waarmee je je paswoord kunt aanpassen. Firebase maakt het ook hier gemakkelijk om deze functionaliteit in je applicatie in te werken. De email die gestuurd wordt om een wachtwoord te veranderen is namelijk al automatisch gemaakt, deze kan natuurlijk nog aangepast worden.



The login screen mockup features a white background with a light gray header bar. On the left of the header is a back arrow, and on the right is the title 'Aanmelden'. Below the header, the text 'Welkom Terug!' is centered in a bold font, followed by the subtitle 'Fijn om je terug te zien'. There are two input fields: 'Email' and 'Password', both with placeholder text 'Type here...'. Below the password field is a link 'Wachtwoord vergeten?'. At the bottom is a large, rounded, light green button labeled 'Aanmelden'.

Login scherm



The 'Wachtwoord Vergeten' screen mockup has a white background with a light gray header bar. On the left is a back arrow, and on the right is the title 'Wachtwoord Vergeten'. Below the header, the text 'Geef je e-mail adres in en volg de instructies om je wachtwoord te resetten' is centered. There is one input field labeled 'Email' with placeholder text 'Type here...'. Below the input field is a large, rounded, light green button labeled 'Stuur mail'.

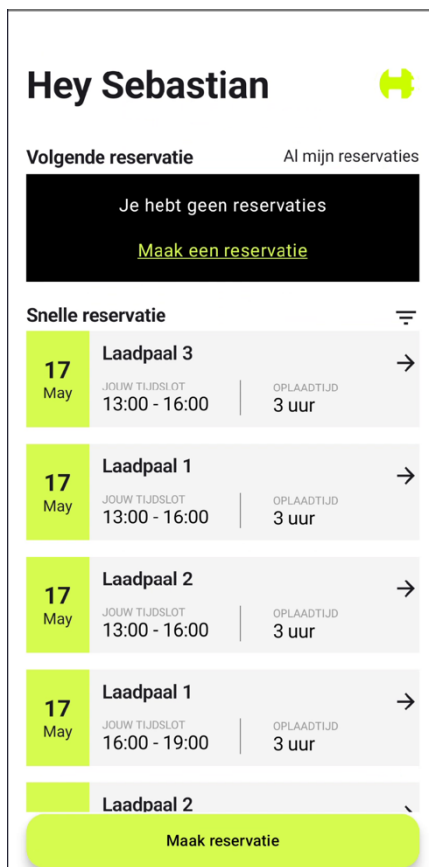
Wachtwoord vergeten

6.5 Biometrische authenticatie

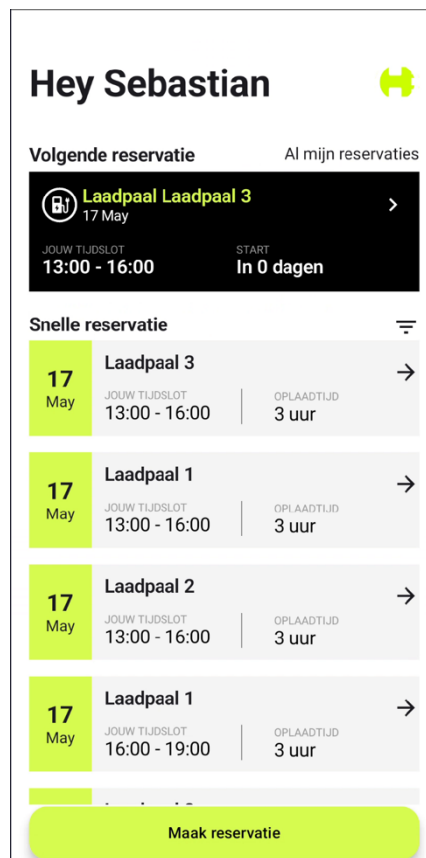
Biometrics is een feature waar ik research naar maar spijtig genoeg niet meer de tijd had om volledig te implementeren. Met biometrische authenticatie wordt bedoeld het inloggen met vinger afdruk of face id in het geval van iOS. Bij is iets meer involved als simpel inloggen. Wanneer we registreren of inloggen maken we gebruik van de firebase authentication service, deze stelt een hoop functies voor die je in je applicatie kan gebruiken, maar al de logica is al gemaakt en alles wordt afgehandeld door firebase, dit verzekerd de veiligheid van je data. Wanneer je wilt inloggen met biometrics moet je de vorige ingelogde gebruiker zijn/haar gegevens lokaal op het toestel opslaan, dit is omdat biometrics geen unique id verbonden hebben, dit wilt zeggen dat als je meer dan 1 vinger afdruk of gezicht heb in je gsm, wij niet weten welke gebruikt wordt om te authenticeren. Gelukkig heeft android en iOS hier een oplossing voor, bijden hebben een library om te handelen met sensitive data lokaal, dit is dan ook één van de weinige features die we niet gezamenlijk kunnen doen maar individueel moeten programmeren. Wanneer we biometrische authenticatie willen implementeren moeten we de vorige ingelogde gebruiker zijn gegevens gaan opslaan. Als je dan je jezelf dan probeert te authenticeren, dan kunnen we de gegevens van deze vorige ingelogde gebruiker gebruiken om opnieuw in te loggen. Verder ben ik nog op een mogelijk irritatie puntje getuimd. Het is namelijk zo dat wanneer je meerdere biometrische registratie mogelijkheden hebt op je gsm, neem nu bijvoorbeeld jij en je vriendin/vrouw, dan ga je handmatig je gegevens moeten ingeven. Waarom is dit, wanneer je een vinger afdruk in je gsm registreert, dan houdt je gsm niet bij tot wie die vinger afdruk hoort. Dit wil zeggen dat als je meerdere vinder afdrukken in je gsm hebt geregistreerd, dat je gsm niet weet wie zich registreert.

6.6 Home page

De home page is de centrale pagina waar je het meest op kan zien. Om te beginnen wordt je naam getoont met daarnaast een foto, op de moment is dit een placeholder. Daaronder zie je je eigen opkomende reservatie, zoals je ziet heb ik op de moment nog geen opkomende reservatie. Daaronder vindt je dan een lijst van mogelijke reservaties die je kan boeken. Als laatst hebben we een knop waarmee we een reservatie kunnen maken. Als we dan een reservatie zouden maken dan kan je die ook gemakkelijk op deze pagina zien.



Home scherm



Eerstvolgende reservatie

6.7 Quick reservation

Wanneer je een flexibel persoon bent kan je gebruik maken van de quick reservation feature. Wil je een laadpaal reserveren dan kun je op één van de reservaties klikken op je home screen, dit brengt je naar de volgende pagina. Hier kun je simpelweg op reserveren klikken, dit brengt je terug naar het home scherm en je reservatie is gemaakt.

6.8 Edit reservation

Wanneer je om eender welke reden op een specifiek tijdstip wilt reserveren dan is dat ook mogelijk. Als je op jouw reservatie klikt zie je de `bewerken` knop, wanneer je hier op klikt kom je op de pagina om je reservatie aan te passen. Bovenaan zie je de datum en tijd van je reservatie, als je je tijd zou willen aanpassen bijvoorbeeld, dan kun je op de tijd klikken en deze aanpassen. Daaronder kun je een klein overzicht zien van wanneer je reservatie is. Verder kun je ook de beschikbaarheid van je reservatie zien voor wanneer je je tijdslot zou aanpassen en kun je ook zien hoever voor je reservatie je een herinnerings melding krijgt.

The image displays three sequential screenshots of a mobile application interface for managing charging reservations.

- Left Screenshot (Reservatie maken):** Shows a confirmation screen titled "Wil je deze laadsessie reserveren?". It features a card for "Laadpaal Carport" on "17 May" with a reserved time slot of "16:00 - 19:00". Below the card are two buttons: a grey "Bewerken" (Edit) button and a green "Reserveren" (Reserve) button.
- Middle Screenshot (Reservatie aanpassen):** Shows the "Reserveren" (Reserve) screen. It includes a "Tijd" (Time) section with fields for "Datum" (Date) set to "28 Sept. 2003" and "Gewenste oplaadtijd" (Desired charging time) set to "16:00 - 19:00". A "Preview" section shows a calendar view with existing reservations for "Pieter Mullen" (Tesla Model X) at 07:00-10:00 and 19:00-20:00, and a new reservation for "Jij" (You) (Tesla Model X) at 16:00-19:00. At the bottom, there are toggle switches for "Laadpaal PLACEHOLDER vrij" (Charging station PLACEHOLDER free), "Beschikbaar" (Available), and "Prioriteit" (Priority).
- Right Screenshot (Reservatie aanpassen):** Shows the "Reserveren" (Reserve) screen with a focus on the reservation details. It displays a time slot of "16:00 - 19:00" and lists other reservations for "Pieter Mullen" (Tesla Model X) at 19:00-20:00. Below this, there are sections for "Laadpaal PLACEHOLDER vrij" (Charging station PLACEHOLDER free), "Beschikbaar" (Available) with a checkmark, "Prioriteit" (Priority) with a toggle switch, and "Melding" (Notification) set to "15 Minuten" (15 Minutes). A green "Maak reservatie" (Make reservation) button is at the bottom.

Reservatie maken

Reservatie aanpassen

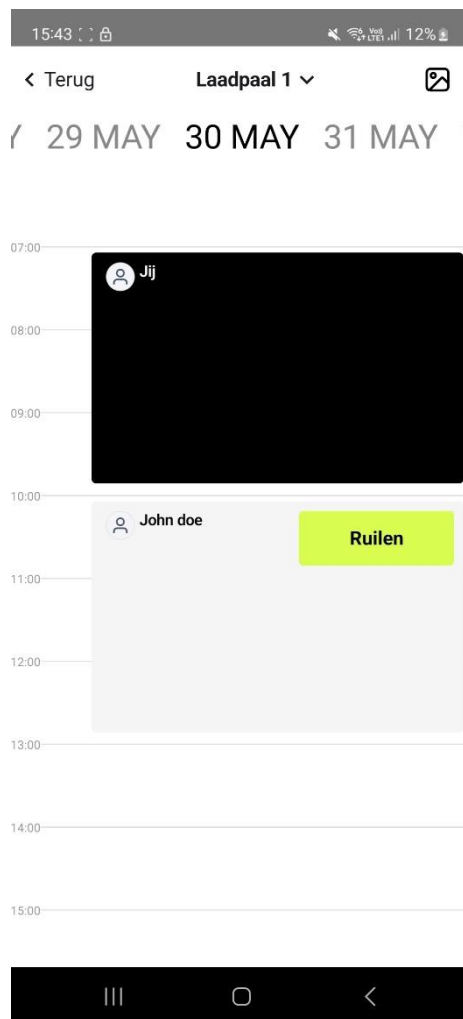
Reservatie aanpassen

6.9 Kalender

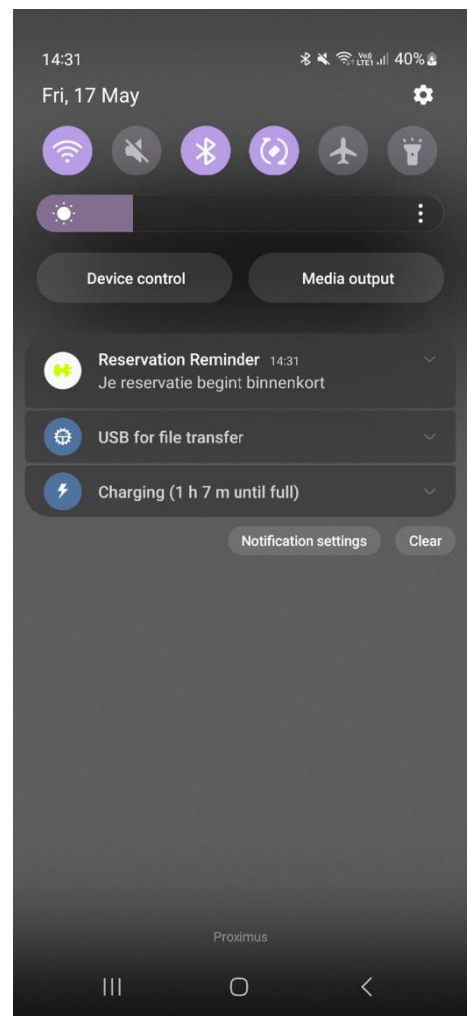
De Kalender is nog een pagina waar ik zelf aan gewerkt heb. De kalender biedt een betere visuele manier om de reservaties te bekijken. Bovenaan zie je voor welke laadpaal je de kalender ziet, hier kan je op klikken om een andere laadpaal te kiezen. Daaronder vindt je een scrollbare carousel met datums die je kunt selecteren, door een datum te selecteren spring je in de kalender naar die specifieke datum. De Kalender en de datums kunnen oneindig ver scrollen. De kalender toont al de reservaties van de geselecteerde laadpaal, de kalender begint automatisch op de huidige datum en het huidige uur.

6.10 Notificaties

Notificaties zien iets klein en iets wat mensen ook vaak vanzelfspreken vinden. In onze applicatie krijg je een notificatie als herinnering dat je een laadpaal hebt gereserveerd. Notificaties zijn OS specifiek, dit is dan ook één van de weinige functionaliteiten die we individueel hebben moeten maken.



Kalender scherm



Notificatie voor komende reservatie

7 SLOT

Ik ben erg blij als ik terugkijk op mijn stage bij Wisemen. Ik heb de mogelijkheid gekregen om te mogen werken in een geweldige werkomgeving. Met de hulp van twee top begeleiders die altijd hielpen wanneer het nodig was. Ik heb niet alleen veel technisch bijgeleerd, maar ook mijn algemene soft skills zijn verbeterd.

Ik heb mijn stage begonnen met weinig tot geen kennis over android development. Ik ben daarom ook begonnen met een onboarding project. Deze alleen was al zeer leerrijk. De onboarding heeft mij geleerd hoe ik een android applicatie maak die productie waardig is, met de juiste technieken, en op de Wisemen manier.

Hierna ben ik kunnen beginnen aan mijn stage opdracht, en als iemand die graag bijleert was dit en top opdracht. Het proces om een applicatie te maken heeft mij zeker geholpen om betere code te schrijven. Daarnaast heb ik ook geleerd om naar het grotere beeld te kijken, en dieper in te gaan op datgene dat ik wil maken.

Ik ben super blij met de kans die ik bij Wisemen heb gekregen. Hoewel ik de specifieke android development kennis misschien niet meer ga gebruiken in mijn toekomst, weet ik zeker dat ik heel veel algemene kennis meeneem. Ik zie er alvast erg naar uit om mijn kennis en skills te blijven ontwikkelen in mijn verdere studies en later mijn carrière.