Ontwikkelingen rondom de development van Body Buddy

Na de goedkeuring om verder te mogen met het project in kwartaal 4, ben ik gestart met het bouwen van Body Buddy met behulp van Ionic React. Hiervoor ben ik een beetje door de documentatie gaan bladeren, en heb ik een aantal tutorials gezocht om te zorgen dat ik zo trouw mogelijk aan het framework blijf, zoals het bedoeld is om te gebruiken.

Ik heb gebruik gemaakt van een video tutorial van Academind om de eerste componenten op te bouwen.

Toen ik eenmaal een enkel (hard-coded) recept kon ophalen, zie deze commit, besloot ik dat het een goed idee was om te bepalen welke database we gaan gebruiken voor dit project. Hierom besloot ik eerst een plan te maken:

- Ik wil met drie databases gaan testen:
    - Firebase
    - MongoDB
    - MariaDB
- Met deze drie databases wil ik een login/registreersysteem bouwen
- Met deze drie databases wil ik recepten kunnen ophalen uit de database en in een lijstweergave plaatsen.

Doordat het langer duurde dan verwacht om deze functionaliteiten in Firebase en in MongoDB werkend te krijgen, heb ik ervoor gekozen om MariaDB achterwege te laten, aangezien ik qua werking met Firebase en MongoDB tevreden ben.

Om beide databases te testen, leek het mij slim om een basis te bouwen en hier vervolgens drie verschillende branches voor te maken, zodat ik met iedere database kan testen en de uiteindelijke keus terug kan mergen in de development-branch.

Firebase

Zo gezegd, zo gedaan. Ik heb eerst een loginpagina en een registreerpagina aangemaakt (de bijbehorende commit 1 en commit 2), en ben vervolgens in de eerste subbranch met Firebase aan de slag gegaan. Hier heb ik als eerste de login en registratie gebouwd, wat verrassend soepel verliep. Hierbij heb ik gebruik gemaakt van deze tutorials en wat losse google zoekopdrachten om foutmeldingen op te lossen, want uiteraard werkt het nooit zoals in de tutorials.

Vervolgens heb k geprobeerd een koppeling te maken tussen de firebase.auth functionaliteit en de firebase.firestore functionaliteit. Dit laatste is de daadwerkelijke database waarin alle gegevens opgeslagen worden. De eerste functionaliteit wordt gebruikt om een login / registratie te maken en deze te voorzien van authenticatie. Om deze koppeling voor elkaar te krijgen heb ik gebruik gemaakt van deze tutorial. Als extra gegeven in de database heb ik een geboortejaar toegevoegd aan het registratieformulier.

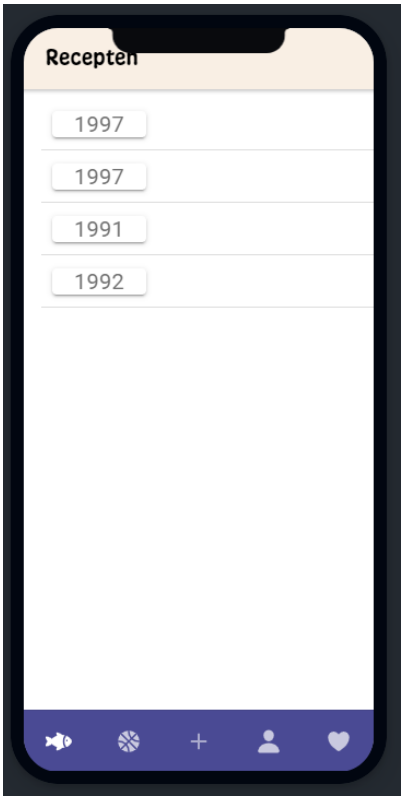
Als laatste wilde ik het geboortejaar in de al bestaande receptenpagina uitlezen. Het uitlezen van de database data verliep redelijk voorspoedig (dit lukte al vrij snel in de console van de developer tools), maar het converteren van de data naar ionic react componenten wilde maar niet lukken. Ik gebruikte een map functie, maar kreeg hierbij als foutmelding dat dit niet kan op een object. Dit klinkt logisch,

behalve dat de gegevens uit de database een array was, met daarin 4 objecten voor 4 database entries. Uiteindelijk bleek ik ergens anders per ongeluk te hebben aangegeven dat er een object moest worden verwacht (typescript verlangt dit van je), wat niet overeenkwam met wat er werd meegegeven vanuit de functie. Toen dit eenmaal opgelost was, kreeg ik een foutmelding dat een JSX element (wat React gebruikt) geen leegte (void) terug kan krijgen, terwijl ik wel een return statement had waarin ik de Ionic Componenten teruggaf. Echter bleek het zo te zijn dat ik die componenten in een React Fragment moest plaatsen, zodat er slechts 1 ding teruggegeven wordt. En aangezien dit in een map functie stond, moest ik een fragment aanmaken die boven het aangemaakte lijst-component stond. Dit eenmaal gedaan hebbende, kreeg ik geen fouten meer te zien in de editor, dus dacht ik dat ik klaar was. Echter bleek in de console nog wel waarschuwingen te komen, namelijk dat iedere item uit de map functie een eigen key moet hebben. Ook dit dacht ik al gedaan te hebben, ik had op iedere IonItem (zie afbeelding verderop) een key toegevoegd met de index van het databaseitem. Echter bleek na een hoop googlewerk dat ik de key niet op het item moest plaatsen, maar op het fragment. Hier kwam ik achter door de code van deze github issue door te nemen. My mind was blown.

Hieronder is de code te zien waar ik door de 'recepten' loop en voor ieder 'recept' een paar Ionic componenten inlaadt. Door alle problemen die ik had en de tijd die ik er aan gespendeerd heb, ben ik vergeten goed bij te houden welke websites en code mij geholpen hebben om dit op te lossen. Hierdoor heb ik alleen de url naar de code voor de laatste foutmelding hierboven geplaatst.

```
return <> {

    <IonList> {

        recipes.map( (anObjectMapped: any, index: any) => {
            console.log(anObjectMapped[0]);
            return <React.Fragment key={index}> {
                <IonItem>
                    <IonCard>
                        <IonTitle>{anObjectMapped[1]}</IonTitle>
                    </IonCard>
                </IonItem>

            } </React.Fragment>

        })

    } </IonList>

} </>
```

Figuur 1 – Code loopen door 'recepten'



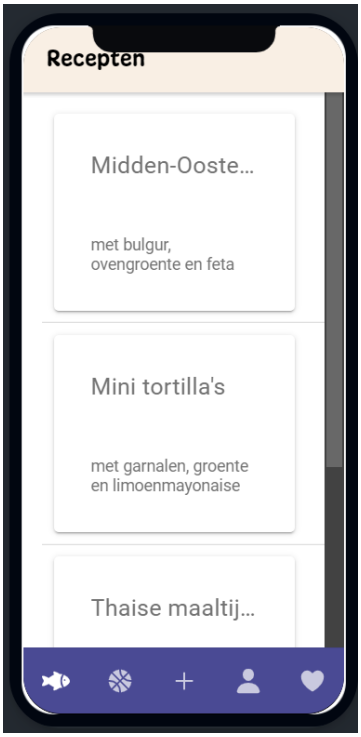
Figuur 2 - eindresultaat loopen door 'recepten'

Het voordeel van de problemen die ik had, is dat deze problemen waarschijnlijk niet meer voor gaan komen bij het implementeren van de andere twee databases. Een groot deel van de code zou ik zo kunnen hergebruiken, als alles voorspoedig verloopt…

MongoDB

Om de gebruiker te registreren, moet ik eerst een connectie maken met mongoDB. Hiervoor heb ik deze tutorial gebruikt. Hier heb ik uiteraard mijn eigen draai aan gegeven om het toe te passen op onze gebruikerssituatie. De tutorial maakt echter gebruik van Angular, een framework waar ik zeker van gehoord heb, maar zelf nog nooit mee gewerkt heb. Hierom ben ik verder gaan zoeken naar een react + mongoDB implementatie, en kwam deze videotutorial tegen. Deze tutorial is op het moment van schrijven nog geen maand oud, dus is in ieder geval actueel. De tutorial gaat best uitgebreid in op de aspecten die we nodig hebben om te werken met MongoDB, en ook op dingen die ik later nodig ga hebben, zoals filtering. De code hiervoor heb ik geimplementeerd in de mongoDB branch, zodat deze al klaar staat mochten we deze branch gaan terugmergen naar de development branch. Echter moeten de filtertermen wel worden aangepast naar termen die wij gaan gebruiken. In de tutorial worden POST, PUT en DELETE ook behandeld. Een goede tutorial om te bewaren, mochten we MongoDB gaan gebruiken!

Met bovenstaande tutorial heb ik dus een API gebouwd met MongoDB, waarin ik een aantal dummy recepten heb staan. Met de code van de firebase branch kon ik er simpel voor zorgen dat de databasedata in de app in kaartjes kwam te staan, dit kostte niet veel tijd. Het eindresultaat voor de receptenpagina ziet er nu zo uit:



Figuur 3 - eindresultaat receptenpagina met MongoDB

Hierna moest ik nog authenticatie implementeren, om gelijkwaardig oefenmateriaal te kunnen vergelijken. Hiervoor heb ik MongoDB Realm gebruikt, wat ook terugkwam in de tutorial. Echter heb ik de tutorial zelf niet gebruikt, omdat deze niet op authenticatie in ging.

Voor de authenticatie heb ik gebruik gemaakt van de MongoDB Documentatie, welke een uitgebreide sectie heeft over authenticatie. Specifiek heb ik deze pagina gebruikt over email + password authenticatie (hetgeen wat mij het meest voor de hand liggend lijkt voor onze app), deze pagina om iets meer inzicht te krijgen in hoe ik mongoDB Realm met react kan verbinden en deze uitgeschreven tutorial van mongoDB zelf om me aan best practices te houden, samen met de bijbehorende github repository om bepaalde implementaties te bekijken (niet alles staat even duidelijk uitgelegd, naar mijn mening). Uiteindelijk heb ik in enkele uren voor elkaar gekregen om gebruikers aan te melden (helaas niet met een bevestigingsemail) en hen te kunnen laten inloggen.

Deskresearch

Nu ik beide databases getest heb, kom ik erachter dat beiden een fijne manier van werken hebben. Firebase heeft de back-end al voor je klaar staat, wat erg fijn is, maar hierdoor loop je wel tegen beperkingen aan. Zo is Firebase vooral goed te gebruiken bij kleine applicaties (aldus MongoDB (https://www.mongodb.com/firebase-vs-mongodb) en een developer die van firebase op mongodb realm is overgestapt (https://developer.mongodb.com/community/forums/t/firebase-vs-realm/9929/4)), maar bedrijven die Firebase gebruiken, zoals 9Gag en Twitch, spreken dit tegen naar mijn mening.

Wat wel overeenkomt op diverse websites, is dat MongoDB veiliger is dan Firebase. Voor PLE ben ik ook bezig met het bouwen van een applicatie, en een movement die zich bezighoud met de ontwikkeling van diabetesgerelateerde applicaties gebruiken ook MongoDB. Aangezien het hierbij om medische gegevens gaat, lijkt me dat de database uiterst veilig moet worden beschouwd. Dit bevestigt dus ook dat MongoDB als veilig beschouwd wordt, minstens veiliger dan Firebase. Firebase is van Google, dus het privacyvraagstuk is dan sowieso iets waar rekening mee gehouden moet worden.

Daarnaast is veel informatie te vinden over de zogenoemde MERN-stack (MongoDB, Express, React en NodeJS), dit is een veelgebruikte stack en mochten we vast komen te zitten in de ontwikkeling kunnen we dus zowel op internet, als binnen school (we hebben in het tweede jaar ook een opdracht moeten doen met de MERN-stack) gemakkelijk om hulp vragen. Deze specifieke informatie voor React en Firebase lijkt minder verkrijgbaar te zijn dan MERN.

Uiteindelijk is tijd ook een factor om rekening mee te houden, wat dat betreft is het backend as a service idee van firebase een stuk logischer om te gebruiken. Echter is MongoDB Realm ook een backend as a service, en door de eerder gevolgde tutorial staat die back-end ook helemaal klaar. MongoDB heeft Realm overgenomen en wat aanpassingen gedaan. Ook is er een uitgebreide documentatie, waardoor er zeer veel informatie en toepassingen over te vinden zijn.

Conclusie

Met name door de veiligheid (en privacyvraagstukken van firebase) is mijn voorkeur uitgegaan naar MongoDB. Daarnaast is het gebruik van de MERN-stack en bijbehorende tutorials een grote doorslaggever in dit onderzoek. Uiteindelijk moet er rekening gehouden worden met het feit dat ik nog weinig kennis heb van de backend, en aangezien we een goed product neer willen zetten, is alle tijd die ik kwijt ben aan het uitvogelen van de backend, tijd die we niet terug kunnen krijgen in de ontwikkeling van het product zelf.