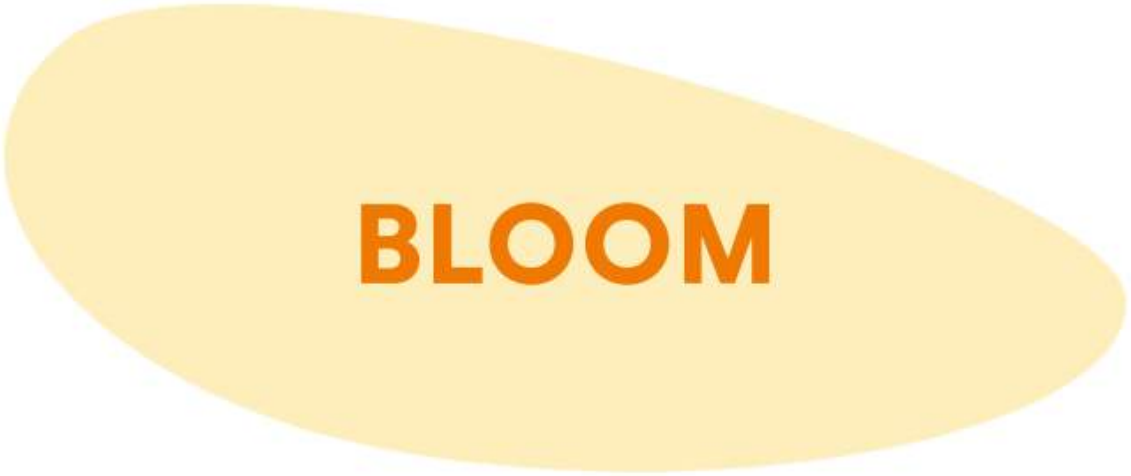
Product Biografie | bloom



Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

Inleiding

Opdrachtgever: Eva Valkenburg
Applicatie: Bloom

Wij gaan een Applicatie maken voor de opdrachtgever, die in samenwerking met kanker.nl & LangeLand Ziekenhuis Zoetermee is uitgebracht. De rechten voor de content heeft de opdrachtgever gekregen van de twee organisaties.

Reden van project
De reden dat de opdrachtgever aan dit project begon, is omdat zij persoonlijk betrokken is geweest bij deze ziekte. De moeder van de opdrachtgever heeft borstkanker gehad. Ze is toen veel met haar moeder gaan praten over het proces erna. Echter is er nog weinig instructie/aanpak na het behandingelingsproject te vinden. Het is wel tijdens het behandelingsproces duidelijk wat de aanpak is, maar daarna houdt het dan ook op. Daar is nog niet veel over bekend. Daarom kreeg Eva het idee om dit te gaan onderzoeken.

Doel app
Het doel van deze applicatie is het verbinden van mensen die de ziekte kanker hebben gehad of nog hebben. Deze mensen kunnen bij elkaar hun verhaal of juist steun vinden bij de andere gebruikers. De gebruikers kunnen dus met deze app een ‘buddy’ vinden. De gebruiker kan in de app ook aangeven welk soort kanker zij hebben gehad en hierdoor worden deze mensen dus gematched op basis van het soort kanker.

Doel opdracht
Aan ons is de opdracht om het bestaande online prototype (Adobe XD) te realiseren in een werkend prototype die echt te gebruiken kan worden door gebruikers. Bij het maken van deze opdracht zou er met de volgende dingen rekening gehouden moeten worden:

- Communicatie zowel met de opdrachtgever als met elkaar
- Het begrijpen van het design en dit realiseren tot de gecodeerde versie (productie)
- Het doornemen van alle documentatie die de opdrachtgever heeft gedaan
- Op basis van de beschikbare documentatie kijken of alles helder/duidelijk/compleet is

Het briefingsproces

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

Zoals eerder benoemd, heeft de opdrachtgever research gedaan. Deze research heeft zij gedaan met kanker.nl & LangeLand Ziekenhuis Zoetermee. Ze heeft haar hele proces bijgehouden op: Gitbook. Hierin kunnen wij dus een kijkje nemen naar het proces die de opdrachtgever gedaan heeft. Op basis hiervan kunnen wij ook een beter beeld krijgen van het hele proces.

Welke problemen worden er door de doelgroep ervaren?
subvraag: Waar lopen zij het hardst tegenaan?
subvraag: Op welke vlakken is er wel, of juist geen begeleiding?
subvraag: Wat zijn de pijnpunten van de doelgroep?
subvraag: Welke oplossingen zijn er al?

Beantwoording subvragen

MoSCoW

Must haves:

- Het kunnen inloggen & registreren van gebruikers
- Het vinden van een buddy (matching op basis van het soort kanker)
- Kunnen chatten met andere gebruikers
- Vinden van informatie (thema's)
- De gebruikers tips/hulp geven bij bepaalde emoties (angst, relatie, somberheid etc.)

Should haves:

- Het koppelen van een bitmoji/initialen aan je profiel
- De app toegankelijk maken voor elk device

Won't haves:

- Het maken van een CMS systeem waar alle informatie van de thema's te vinden is
- Het gebruik van een API om al d content te managen
- 2-step verificatie

Welke middelen zijn er beschikbaar?
Bij het maken van deze applicatie zijn er de volgende middelen beschikbaar:

- Een discord groep (met de opdrachtgever en de studenten),

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

- Een git omgeving,
- Een ‘Trello’ (projects) omgeving om goed overzicht te kunnen houden,
- Verschillende digitale prototype(s) (Adobe XD)
- Een Uitgebreide product biografie

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

Stakeholders

Tijdens het maken van de applicatie is het handig om verschillende stakeholders in gedachten te houden, deze stakeholders vormen namelijk een geheel voor wie wij ook rekeningen moeten houden voor het maken, deze stakeholder zijn als volgt:

Ziekenhuis

- Verpleegkundigen
- Chirurg
- Oncoloog
- Psycholoog
- Huisarts
- Arbo arts
- Fysiotherapeut

Prive omstandigheden

- Directe omgeving
    - Vrienden
    - Partner
    - Familie

Werk

- Werkgever
- Collega’s

Organisaties

- Untire
- kanker.nl

Overige

- Bestaanden oplossingen
- Patiënten in remissie

Planning

	Taken	Deadlines/Oplevering
Week 1	- Debriefing - MoSCoW inventarisatie - Concept aanscherpen (bv Micro-Interacties) - PvA - user scenarios (kritiek?) - customer journey (kritiek?)	- Debriefing - 21 mei: 1e prototype
Week 2	- Feedback verwerken 1e prototype - Verder bouwen - *userStories/kernFunctionaliteiten af* - Prototype testen	- 28 mei: 2e prototype
Week 3	- Feedback verwerken 2e prototype - Verder bouwen - *overige features* - Prototype testen	- 4 juni: 3e prototype
Week 4	- Feedback verwerken 3e prototype - Verder bouwen - *extra (coole) features (zoals bitmoji)* - Prototype testen - Check of alles erin zit	- 11 juni: 4e prototype
Week 5	- Feedback verwerken 4e prototype - Laatste dingen bouwen + testen - Laatste dubbel check of alles gedaan is	- 17 juni: Presentatie opdrachtgever - 18 juni: Beoordeling docent

Informatie die de lezers nodig hebben
Alle informatie die de lezers nodig hebben, die kunnen die informatie hier vinden

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

Concept aanscherpen
Hier vertellen wij het een en ander wat ons opvalt aan het bestaande prototype en wat ons lijkt te kunnen verbeteren. Met kritische blik

- start page: loader overbodig
- interactie: teruggaan naar vorige pagina, (swipe?) bolletjes klikken?
- 'inloggen' ipv 'starten
- registratie
    - progressive disclosure
    - animatie: blad erbij bij het verdere aanmaken van acc
    - terug pijltje onduidelijk
    - gender: te makkelijk. Misschien iets dat meer past bij huisstijl
    - type kanker: 'overig' op het einde.
    - wat is memoji®?
    - height van tekstballon langer maken. Lijkt klein

- home
    - skip ik onboarding als ik niet op 'start uitleg' klik?
    - wat is zoeklens? waar moet ik mn input zetten? feedback mist. Lens moet niet ingekleurt zijn.

- thema
    - iframe link openen
- buddy
    - filter locatie logisch
    - profile vragen alleen te maken bij "profiel", niet bij "registratie"
    - eerst naar chat wnr je op bericht klikt, dan het type venster

- chat
    - 1'tje onderaan weg wanneer verzoek is ingediend
    - wat doet 3 points rechtsboven in chatverzoek?
- profiel
    - huisstijl blijft hetzelfde, ook al is backgroundcolor van profile image anders
    - wat is 'onco specifiek' → bestaat niet? → 'ziekte-specifiek'
    - waar kan ik mijn account verwijderen? Extra knop onder "uitloggen" met extra nadruk?
    - eigen ingeving bij profielvragen?

Plan van aanpak
Hier vertellen wij hoe wij dit project gaan aanpakken, en hoe de communicatielijnen gaan lopen in de komende weken.

Communicatie met opdrachtgever
De communicatie met de opdrachtgever zou plaats vinden in Discord, hierin kunnen de studenten meteen een gerichte vraag stellen aan de opdrachtgever als zij die hebben. Ook kunnen de andere studenten de vraag zien en het antwoord, hiermee voorkomen we dubbele vragen wat voor de opdrachtgever ook minder tijd in beslag neemt.

Communicatie met elkaar
De communicatie met elkaar zou voornamelijk bestaan uit Discord & github, met deze tools kunnen we elkaar benaderen/spreken, Discord zou gebruikt worden voor het overleggen of code issues hiermee kunnen we voorkomen dat we vastlopen op een bepaald stuk voor gedurende tijd.

Github zou voornamelijk gebruikt worden om 'issues' in te schieten, hiermee krijgen we een beter beeld over wat er nog gedaan moet worden, en waar eventueel nog een aanpassing nodig is (bug, extra toevoeging etc.)

Afspraken die gemaakt zijn voor het project
Elke dag een ochtend meeting die om 9:30 gaat beginnen, en om 17.00 een andere meeting, hiermee kunnen we elkaar peilen van hoe ver iemand is of waar hij/zij tegen aan denkt te gaan lopen, hiermee proberen we te voorkomen dat we eigenlijk geen idee hebben wat de andere persoon aan het doen is of hoe ver de persoon met een bepaalde feature is.

Werkomgeving
- git
- GitHub

Code conventies
- beschrijvende functie -en variabele namen
- functies voorzien van beschrijving (comments) van wat meekrijgen (parameters), wat ze doen en wat ze teruggeven (returnen)
- arrow of regular functions?
- Prettier linting file
    - semicolon
    - 2 quotes
    - 2 Tabs
- ES 6 modules

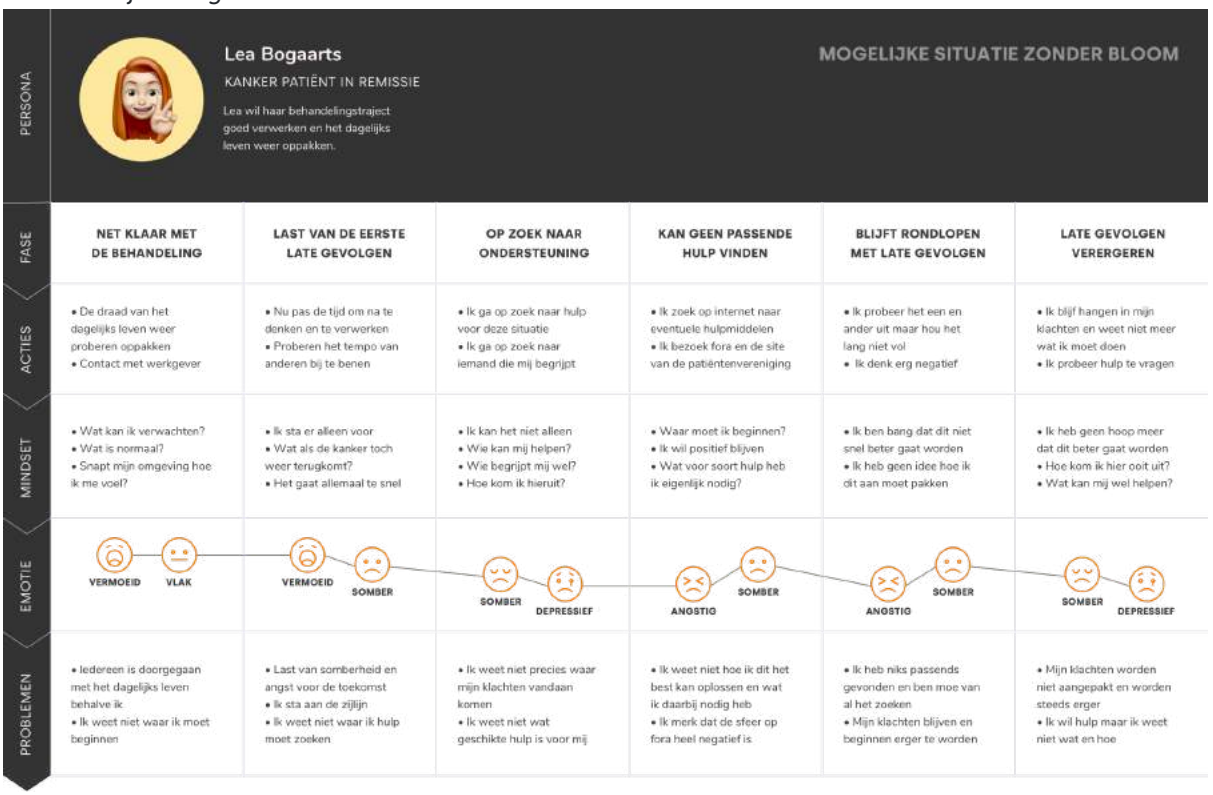
User scenarios
1. Als gebruiker, wil ik informatie over relevante thema's, zodat ik weet wat ik kan

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

verwachten na afloop van het behandelingstraject.

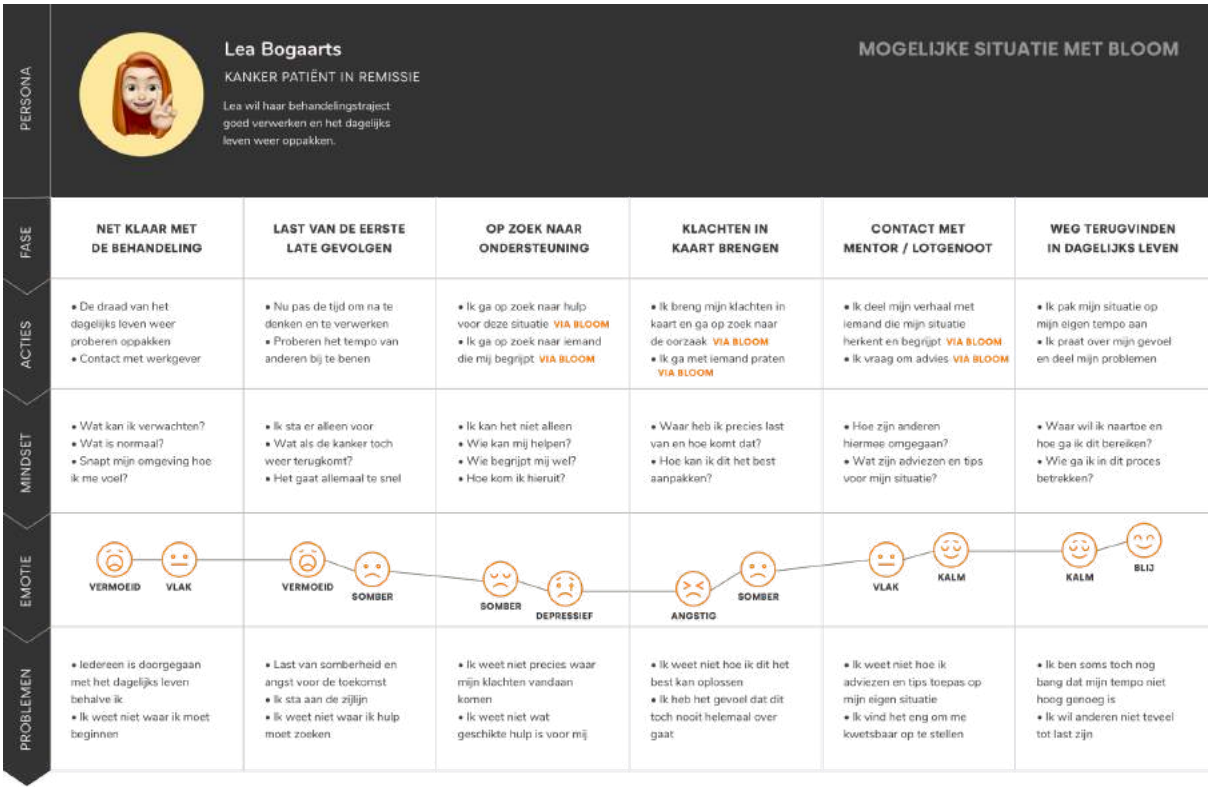
2. Als gebruiker, wil ik 1 op 1 contact met lotgenoten, zodat ik kan reflecteren op mijn ervaringen en mijn copingstrategieën kan versterken.

Customer journey



De reis waar Leo doorheen gaat in de huidige probleemsituatie

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development



De reis waar Lea doorheen gaat met behulp van Bloom

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

Concept kritiek verbeteringen

- Start page loader
  Wij gaan op het beginscherm geen loading state maken, dit komt met name door het feit dat de app meteen zichtbaar hoort te zijn, en daarbij wordt er geen externe data ingeladen waardoor de app zou moeten laden.

- Interactie terug gaan naar de vorige pagina, swipe, bolletjes?
  Tijdens het testen was het niet duidelijk of we terug konden gaan met behulp van de bolletjes, qua swipe interactie etc, daarom willen wij er voor zorgen dat je ook terug kan navigeren met behulp van de bolletjes en eventueel swipen.

- Registratie
- Het terug pijltje bovenaan is onduidelijk. Deze zal anders vormgegeven worden. Denk aan links onderaan bijvoorbeeld.
- Het “Kies Gender” scherm is wat te “simpel” qua visueel ivm de rest van de app. We gaan proberen dit meer in de huisstijl van de app te maken.
- Bij het kiezen van het “type kanker” verwachten we dat de keuze “overig” op het einde staat in plaats van in het midden tussen alle andere keuzes. Dit is namelijk een keuze als je niet kan kiezen uit de gegeven keuzes.
- Het is onduidelijk bij het aanmaken van een profiel dat bij het kiezen van een avatar dat memoji onderstreept is? Is dit een link? Wij gaan hier een info knop van maken die in 1 zin uitlegt wat memoji betekent. (“*Memoji is een vorm van het weergeven van een karakter, wat erg persoonlijk is, maar niet privacygevoelig als bij een foto.*”).
- De hoogte van het tekstveld bij “Vertel iets over jezelf” is niet erg hoog. Je mag best wel wat invullen, omdat er staat dat je iets mag vertellen over jezelf. Dus het tekstveld zal tot bijna aan de knop “Volgende” komen. Wel zal de tekst meebewegen op het moment dat het tekstveld vol staat en je bent aan het typen.

- Home
- Het zoekveld op het beginscherm was voor ons niet meteen duidelijk dat het om een zoekveld ging, omdat er geen icoon instond, het was een rond bolletje, hierdoor was het dus niet duidelijk dat deze interactie beschikbaar was, wij willen dit gaan oplossen door het bijvoorbeeld duidelijker te maken mbv een icoon, animatie of meenemen in de zero-state
- Voor ons was het niet duidelijk zodra je niet op start uitleg drukt of je dan meteen de onboarding process overslaat, en dus meteen naar het index scherm gaat of dat je juist wel door het process heen gaat van onboarding.

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

- Thema
- Er wordt gebruikt gemaakt van externe (?) links bij de thema's bij de koppen: hulp/tips, omdat we er niet op kunnen klikken om te zien waar het naartoe ging, hadden wij het idee het het zou leiden naar een externe website waar je de informatie kan opzoeken. Als die informatie hetzelfde is kan het misschien aantrekkelijker zijn om er een soort uitklap systeem van te maken waarin de gebruikers de informatie te zien krijgen dus de basis, en als ze geïnteresseerd zijn de volledige pagina te open.

- Buddy
- De profielvragen waren alleen beschikbaar als je zelf naar je profiel ging, deze werden overgeslagen in het regristratie process, misschien is het beter om deze vragen meteen bij de registratie te vragen.

- Chat
- Het "1"tje dat onderaan bij het chat icoon in de navbar staat, mag weg wanneer er een chatverzoek is. Deze indicatie moet er alleen staan wanneer het gaat over het aantal berichten/chats die messages bevatten. Dus niet wanneer er een chatverzoek is ingediend.
- Het is onduidelijk wat de 3 puntjes rechtsboven in het chatverzoek doen? Dit is niet een must-have, maar als er tijd voor is, zullen we hier een dropdown van maken met de keuze uit "Rapporteren"/"Blokkeren".

- Profiel
- Wij kwamen er niet achter wat onco-specifiek is, wellicht is dit een jargon term die ons niet bekend is, maar we twijfelde daarom aan de naamgeving van de header of deze wel specifiek en helder is om te begrijpen, omdat we nog niet specifiek weten wat het woord precies inhoudt hebben we hier nog geen vervanging voor, maar we willen deze wel graag veranderen.
- Er is binnen de app niet de optie om je account te verwijderen, wellicht kan/is dit een handige feature om je account helemaal te verwijderen.
- De profielvragen zijn kant en klaar voor je geschreven, wellicht is het een idee om zelf een vraag te maken en deze in te vullen, hierdoor wordt de user experience wellicht iets meer persoonlijker.

Week 1

Taken (streven) deze week

Rowin	Ralf
Thema's	Connectie database (mongoDB)
Github inrichten	Connectie frontend framework POST methods naar backend Node.js server opvangen
SASS inrichten, gebruik van variabele, media queries, fonts	Projects/Issues management in GH opgezet

Login maken met Vue frontend en Nodejs backend

De omgeving gemaakt met nodejs, vue , expres, mongodb en Mongoose. Het was een hele uitzoekerij, maar deze resource heeft mij geholpen: https://bezkoder.com/vue-node-express-mongodb-mevn-crud/

Ik heb een data model gemaakt van hoe de user in de database komt te staan. Hij ziet er zo uit:

```
const userSchema = new Schema({
  firstName: String,
  surName: String,
  emailAddress: String,
  password: String,
  birthDate: String,
  town: String,
  gender: String,
  typeIllness: String,
  profileAvatar: String,
  about: String,
});
```

Connectie database

Connectie met de DB was als altijd even uitzoeken hoe dat ook alweer moest. Maar na een paar uurtjes was de connectie geslaagd.

Login met bestaande user uit DB

Ik ben gelijk aan de slag gegaan met het inloggen van een user. Dit lukte prima en hierdoor kon ik gelijk verder met de volgende stappen.

Vue frontend koppelen form submit data doorsturen naar server

Resource: https://bezkoder.com/vue-js-crud-app/

Het koppelen van form submits in vue naar nodeJS server gaat niet gemakkelijk.
Even later heb ik bodyarser toegevoegd en cors optionSuccesStatus: 200 en nu doet hij het wel. Gebruiker kan nu inloggen en account aanmaken. Ik heb dit op mij genomen, omdat ik al bekend was met DB connections en omdat ik de logica ervan begrijp. Mongoose werkt daarin erg fijn en userfriendly.

Dinsdag:

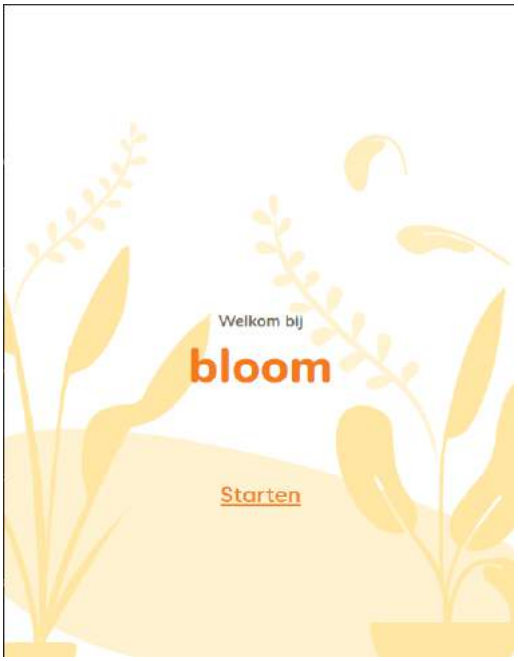
- Connectie mongo → gestoei met .env maar gelukt!
- Registreren van account gelukt op form submit
- login met bestaand account gelukt op form submit
- Bezig met vue connecten aan nodejs form submit → lastig!

Er was heel veel gedoe met de package vue server cli die mijn macbook niet kon vinden. Na veel stoeien en in stackoverflow rondneuzen lukte het! Blijkbaar was het dus een global package van npm die ik handmatig moest installeren. Dit Vue framework Vue is nieuw voor mij, dus het is nog erg wennen hoe alles werkt en waar bepaalde methods voor zijn.

Rowins onderwerpen

Ik heb mij voornamelijk bezig gehouden met het oprichten van de github repo in de eerste week, zo heb ik er voor gezorgd dat variable in elk component gebruikt kan worden. Denk hierbij aan kleuren, fonts, mediaqueries etc.

Daarnaast heb ik de eerste opzet gemaakt voor het begin scherm en de onboarding



Het beginscherm leidt de gebruiker door naar de onboarding, via de onboarding krijg je als het ware 3 states te zien waar je doorheen kan, als de gebruiker op volgende drukt krijgt hij/zij een andere card te zien. Dit heb ik opgelost om een mutation te maken in de Vuex store, hierdoor heb ik een dynamische functie kunnen maken. het gekleurde bolletje geeft aan waar de gebruiker is en hoeveel stappen hij/zij moet zetten.

Notes - meeting met Eva over onze kritiek op concept

- teruggaan swipe + terugknop.
- “Terug naar vorige stap” bij pijltje
- Onboarding is er alleen na de 1e keer inlog van account. Overigens kan je de onboarding wel skippen, door er een button bij te maken.
- Terugkomen van profielvragen bij profiel registreren. Mag, hoeft niet.
- Achtergrondkleur van memoji is gekoppeld aan de kleur van het profiel. → mag ook avatar en kleur kiezen (los van elkaar).
- Account verwijderen toevoegen
- Eigen ingeving profielvragen toevoegen.
- “Overigens” moet achteraan bij het type kanker kiezen bij registratie

Feedback 21 mei van Eva

Wat hebben we laten zien aan Eva?
We lieten ons prototype zien van dat je op 1 pagina in 1x alle invoervelden van de user aanmaakt en zo kan je een user aanmaken in de database. Ook liet Rowin de “begin kaartjes” zien dat je daar doorheen kon swipen.

Feedback Eva
Eva was erg verrast en vond het er al goed uitzien en werken. Hoewel de profielregistratie nog in Progressive Disclosure verwerkt moet worden en het user aanmaken echt goed werkt, was ze al goed tevreden.
Overigens attendeerde Eva ons erop dat de bodytext van de begin kaartjes het font Silka moest zijn. Bovendien vond Eva het er verzorgd uit zien, ook dat je een user kan aanmaken en dat je daarna kan inloggen met email en wachtwoord van die user. Echter zijn de buttons niet goed uitgelijnd bij de begin kaartjes, maar daar zijn wij nog erg mee bezig.
Met deze feedback kunnen we lekker aan de slag volgende week en ook nog even een goede sprint maken.

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

Week 2

Taken (streven) deze week

Rowin	Ralf
Vue en Node koppelen	Progressive Disclosure bij registreren
user scenario: thema's	Componenten maken in Vue
	Form data opvangen bij server

Componenten maken

Sommige elementen hoeven niet als component gemaakt te worden, want dat vind ik meer voor als een element meerdere malen gebruikt wordt. Dan is het handig om maar 1 keer het element te maken zodat je hem daarna eindeloos kan toepassen. Daarom heb ik ervoor gekozen om bijvoorbeeld radio buttons van het Type Kanker niet als component te maken, maar het grote textarea veld wel omdat we die nog een keer in de rest van de app maken (bij profielvragen).

Registreren | Progressive Disclosure

Ik ben bezig met het Progressive Disclosure maken van de registratie stappen. Dit moet in verschillende stappen. Ik heb dit 1 keer eerder gedaan met React, maar dat werkt net ietsjes anders, dus daarom heb ik een artikel gevonden die dit fijn uitlegt. Resource: https://medium.com/engineering-samlino/building-a-multi-step-form-with-vue-2bc861447c4a

Vuex.Store is een data management binnen Vue die alle data van bepaalde inputs bijhoudt.

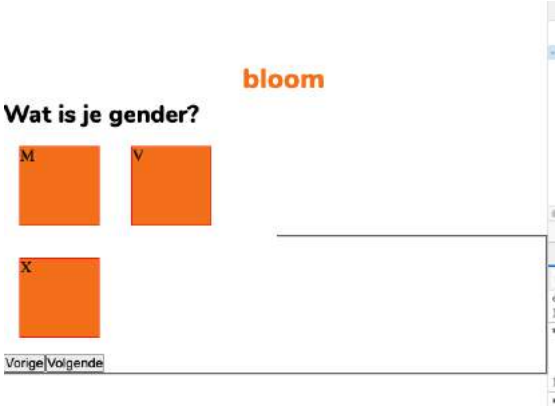
Het aantal bolletjes in het adobe klopt niet. Hij loopt 1 step vast bij bolletje 3. Bovendien moet je in de app inloggen met een e mailadres en wachtwoord, maar in het prototype maak je die bij het registratieproces nergens aan. Deze twee stappen moeten dus nog verwerkt designed en verwerkt worden.

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

Het steeds laten zien van andere components werkt nu goed. Ik maak gebruik van de v-if-else methode van Vue en er wordt een state bijgehouden dat er hele tijd een andere component wordt ingeladen.

```
<fieldset>
  <legend>
    <div v-if="state === 1">
      <h2>step 1</h2>
      <FirstStep />
    </div>

    <div v-else-if="state === 2">
      <h2>step 2</h2>
      <SecondStep />
    </div>

    <div v-else-if="state === 3">
      <ThirdStep />
    </div>

    <div v-else-if="state === 4">
      <FourthStep />
    </div>

    <div v-else-if="state === 5">
      <FifthStep />
    </div>

    <div v-else-if="state === 6">
      <SixthStep />
    </div>

    <div v-else-if="state === 7">
      <SeventhStep />
    </div>

    <div v-else-if="state === 8">
      <EightStep />
    </div>

    <div v-else-if="state === 9">
      <NinthStep />
    </div>

    <div v-else-if="state === 10">
      <Ready />
    </div>
    <button>Vorige</button>
    <button @click="setState()">Volgende</button>
  </legend>
</fieldset>
```

Hoe de gender-stap in registratie er nu uit ziet:



Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

Registreren | State bijhouden

Vuex store gebruiken

Ik wil eigenlijk OF per component het in de vuex store zetten, of in de index.vue het totale aan het einde ophalen.
Op deze manier werkt het en zo heb ik het ook geimplementeerd: https://vuex.vuejs.org/guide/forms.html#two-way-computed-property. In plaasts van v-model moet ik @value gebruiken.

Uiteindelijk heb ik de data van alle components in de Vuex Store gestored.

```
<input
  type="text"
  id="firstName"
  name="firstName"
  :value="firstName"
  @input="updateFirstName"
  placeholder="Voornaam"
/>
<input
  type="text"
  id="surName"
  name="surName"
  :value="surName"
  @input="updateSurName"
  placeholder="Achternaam"
/>
```

1. Value en functie aanroepen vanuit input
2. Computed: haalt de data uit de store en render die in de inputs, zodat als user teruggaat, zijn data er “nog” instaat.
   methods: hier staan de functies die de inputs aanroepen wanneer ze “changen”. Deze functies roepen een andere functie in de Vuex store aan met de value van de input die ze meesturen.

```
computed: {
  ...mapState({
    firstName: (state) => state.user.firstName,
    surName: (state) => state.user.surName,
  }),
},
methods: {
  updateFirstName(e) {
    this.$store.commit("updateStateFirstName", e.target.value);
  },
  updateSurName(e) {
    this.$store.commit("updateStateSurName", e.target.value);
  },
},
};
```

3. De functies in de Vuex store assignen de values aan de keys in de state

```
// Step 1
updateStateFirstName(state, value) {
  state.user.firstName = value;
},

updateStateSurName(state, value) {
  state.user.surName = value;
},
```

4. De state.user met daarin *geen* keys, omdat die door de functies worden aangemaakt.

```
export default new Vuex.Store({
  state: {
    // Register user
    user: {},

    // Onboarding
    activeIndex: 0,
    Onboardingmax: 3,
  },
```

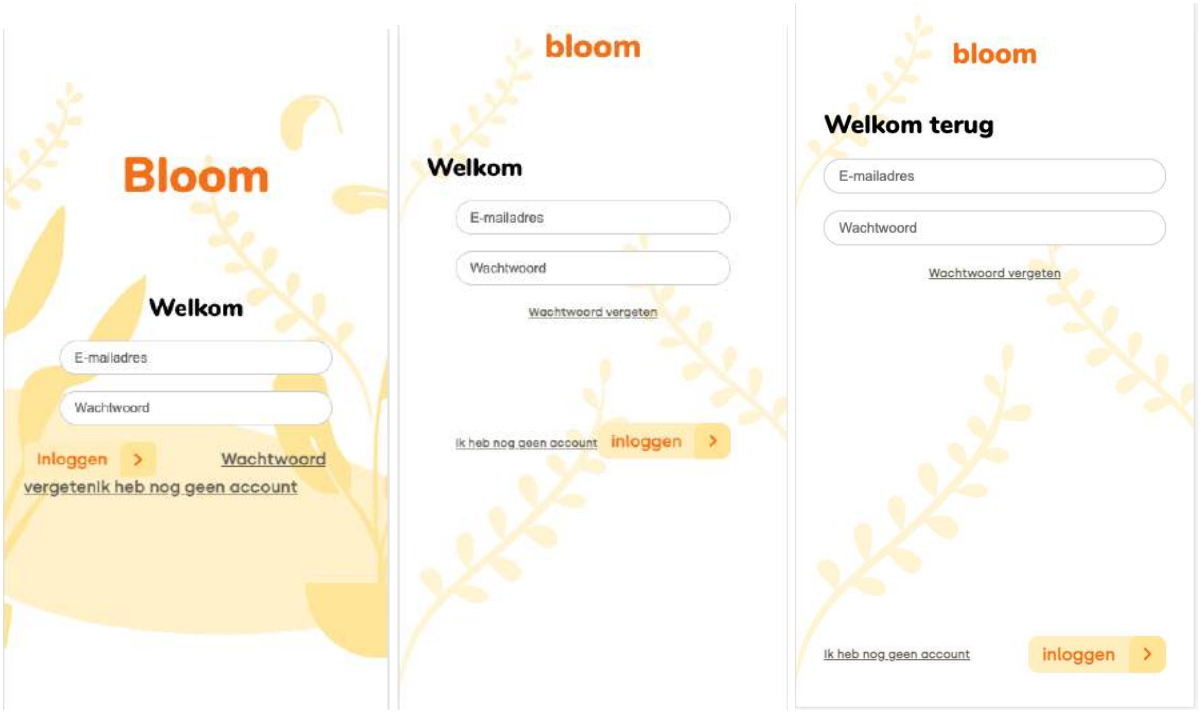
Er komen dus allemaal update functies in de mutations in de Vuex store voor elk data input, maar dat wil ik eigenlijk niet. Dat is niet DRY en ik vind het gewoon niet netjes. Ik ben van mening dat ik dit kan omschrijven naar 1 functie, zodat ik alleen de naam van de data key hoef mee te geven en dat ie op basis daarvan het assigned aan de data key.

Registreren | Form submit data naar server en feedback terugsturen naar cliënt

Dit lukte eerst niet lekker omdat ik moest werken met Axios met het heen en weer passen van data van client naar server en die moest de userdata teruggeven aan de client. Het terugpassen van de userdata lukte nog niet, waardoor ik geen session bijvoorbeeld kon starten. We gaan hier volgende week mee verder, maar dan samen.

Styling | Login

Ik ben bezig met de styling van het login scherm te maken net zoals het prototype. En het begon door de uren heen er steeds meer op te lijken.



SASS is nieuw voor mij , dus ik moet het eerst even doorhebben met de (pseudo) selectoren en dat alles in elkaar genest is. Soms gebruik ik als ik er echt niet uitkom een CSS→ SASS converter om te zien hoe dat dan eruit moet zien.

Styling alternatief voor memoji
Doordat memoji te lastig wordt, heb ik een alternatief bedacht om de initialen van de user aan te bieden in avatars met ieder een andere achtergrondkleur waardoor die achtergrondkleur wordt gekozen en die mee wordt genomen in de profielkleur van die user.

Alleen moeten de colors in de Vuex state geplaatst worden (of global Sass bestand) zodat als de data gerenderd is, dan weet de app welke kleur hij moet renderen bij de value “yellowTwo”.

Styling | Registratie - profiel avatar

Bitmoji(memoji) gaat gewoon niet werken. Het is een super coole feature, maar deze laten we sowieso in de backlog staan. Het heeft geen prioriteit. Wat het alternatief is, en ook al in het prototype van de opdrachtgever stond, is dat het verschillende kleuren

maakt met de initialen van de user daarin. Deze worden gegenereerd door wat de user als input meegeeft bij firstName en surName.

Idee | Animatie

Wij kregen het idee om ook de animatie tussen de routes en in het registratieproces bij Progressive Disclosure animatie toe te voegen. Dit is natuurlijk een vette feature wat niet prioriteit #1 is, maar wel de UX nog een extra sausje geeft. We hebben uitgezocht alvast hoe dit kan en dit zijn de resources:

- https://www.npmjs.com/package/vue-page-transition

Rowins onderwerpen

Ik ben voornamelijk deze week bezig geweest om het te kunnen koppelen van vue en express, dit koste me heel veel tijd omdat ik er geen goede tutorials van kon vinden, en ik kreeg error na error, ik kwam er achter dat vercel geen node server support. Maar we hadden wel onze app al hier op gedeployed. Dus heb ik er voor gekozen om van hosting te switchen naar heroku omdat ik zeker wist dat het hier wel zou moeten werken.

Door middel van een guide heb ik werkend kunnen maken, ik heb een vue.config.js file aangemaakt met daarin:

```
const path = require("path");

module.exports = {
  outputDir: path.resolve(__dirname, "../server/public"),
  devServer: {
    proxy: {
      "/api": {
        target: "http://localhost:5000",
      },
    },
  },
  css: {
    loaderOptions: {
      sass: {
        prependData: '@import "@/assets/scss/main.scss";',
      },
    },
  },
};
```

Ik heb de output (build) directory verplaatst naar de server public, aangezien we de express server gebruiken voor het serveren van de build pages, met behulp van deze commando hoef ik dat dus niet meer zelf te regelen, daarnaast heb ik een devserver ingesteld met een proxy, deze proxy is voor het gebruik van de interne api (express). Met behulp van deze code kan ik de server aanroepen en dus gebruik maken van onze express api.

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

```
app.use(express.static(__dirname + "/public/"));

// Handle SPA
app.get(/.*/, (req, res) =>
    res.sendFile(path.resolve(__dirname, "public/index.html"))
);
```

Vervolgens heb ik ingesteld dat als er geen pagina beschikbaar is dat hij de index.html uit de public map serveert, ook heb ik ingesteld dat expres de public map serveert. Vervolgens heb ik de server map, met behulp van de heroku cli gepushed, na het installeren werkte de app meteen, bij het inloggen van de user kreeg ik de volgende console log uit heroku:

```
2021-05-26T09:01:04.400275+00:00 app[web.1]: hallo
2021-05-26T09:01:04.446390+00:00 app[web.1]: Logged in with account:  {
2021-05-26T09:01:04.446394+00:00 app[web.1]:   _id: 60a4f02ae631e516640ab9cb,
2021-05-26T09:01:04.446394+00:00 app[web.1]:   firstName: 'Harry',
2021-05-26T09:01:04.446394+00:00 app[web.1]:   surName: 'Nak',
2021-05-26T09:01:04.446395+00:00 app[web.1]:   emailAddress: 'harrynak.nl',
2021-05-26T09:01:04.446395+00:00 app[web.1]:   password: '12345',
2021-05-26T09:01:04.446395+00:00 app[web.1]:   birthDate: '2001-12-16',
2021-05-26T09:01:04.446396+00:00 app[web.1]:   town: 'Amsterdam',
2021-05-26T09:01:04.446396+00:00 app[web.1]:   gender: 'Man',
2021-05-26T09:01:04.446396+00:00 app[web.1]:   typeIllness: 'Borstkanker',
2021-05-26T09:01:04.446396+00:00 app[web.1]:   profileAvatar: 'Blauwe foto',
2021-05-26T09:01:04.446397+00:00 app[web.1]:   about: 'Ik ben voetballer',
2021-05-26T09:01:04.446400+00:00 app[web.1]:   __v: 0
2021-05-26T09:01:04.446400+00:00 app[web.1]: }
```

Dit was dus een goed teken dat alles goed zou moeten gaan.

Vervolgens ben ik gaan kijken naar de volgende user scenario en dat was in dit geval het maken van thema's.

```
state: {
  themelist: [
    {
      id: 1,
      name: "Vermoeidheid",
      image: "1.jpg",
      intro:
        "De behandeling van kanker kost veel energie. Dat is logisch. Je lichaam heeft tijd nodig om te herstellen. Daarnaast moet je alle emoties en gebeurtenissen verwerk
      p1: "Meestal ben je na een paar maanden minder moe. Maar je kunt ook lang na de behandeling nog erg moe zijn. Ben je zes maanden na de behandeling nog steeds erg moe
      bulletlist: [
        "Je bent plotseling moe maar je weet niet waarom",
        "De vermoeidheid voelt alsof je uitgeput bent. Je bent al erg moe na een eenvoudig klusje.",
        "Bijvoorbeeld afwassen of stofzuigen. Je bent elke dag moe. De vermoeidheid gaat niet over. Ook niet als je even gaat zitten en uitrust.",
      ],
      quote:
        "Ik mezelf behoorlijk uitgeput. Dit zorgt voor vermoeidheid zowel lichamelijk als mentaal",
    },
    { id: 2, name: "Angst", image: "1.jpg" },
    { id: 3, name: "Somberheid" },
    { id: 4, name: "Mijn relatie" },
    { id: 5, name: "Geldzaken & werk" },
    { id: 6, name: "Lichaam & uiterlijk" },
  ],
```

Ik definieer elke thema in een array via de state in de vuex store, hierin kan ik deze data opslaan zodat ik het per component weer kan opvragen vanuit de url.

```
  getters: {
    themelist: (state) => {
      return state.themelist;
    },
    theme: (state) => (id) => {
      console.log(id);
      return state.themelist.find((theme) => theme.id === id);
    },
  },
});
```

Vervolgens heb ik 2 getters aangemaakt, eentje die alle themes returned en een die specifiek het thema ophaalt op basis van de id.

```
export default {
  computed: {
    themelist() {
      return this.$store.getters.themelist;
    },
  },
};
</script>
```

in themes.vue maak ik een computed property aan, en roep ik de getter vanuit de vuex store op, hierin krijg ik alle thema's terug die ik vervolgens kan loopen met:

```
<div class="container">
  <div v-for="(theme, index) in themelist" :key="index">
    <router-link :to="{ name: 'article', params: { id: theme.id } }">
      <article>
        <h3>{{ theme.name }}</h3>
      </article>
    </router-link>
```

In de loop, haal ik verschillende waardes uit de array, hiermee vul ik de router links aan, ik geef de param een id mee van het thema id, zodat deze route uniek is.

```
    path: "/article/:id",
    component: Themeslug,
    name: "article",
  },
];
```

In router.js geef ik aan dat er achter de article een id komt te staan als wildcard, hierdoor weet vue dat het iets moet doen met die id.

```
<script>
export default {
  computed: {
    article() {
      return this.$store.getters.theme(parseInt(this.$route.params.id));
    },
  },
};
</script>
```

in de slug van de thema's, heb ik gebruikt gemaakt van de andere getters functie, deze functie zorgt er voor dat ik een artikel als waarden terug krijg op basis van de url. Deze data kan ik weer gebruiken in een loop, om zo al mijn slugs te organiseren.

```
  />
  <h1>{{ article.name }}</h1>
  <div class="text">
    <p>{{ article.intro }}</p>
    <div>Hier komen toggles te staan</div>
    <p>{{ article.p1 }}</p>

    <h2 class="klachten">Herken je deze klachten?</h2>

    <ul>
      <li v-for="(theme, index) in article.bulletlist" :key="index">
        {{ article.bulletlist[index] }}
      </li>
    </ul>
  </div>
</article>
```

Feedback 28 mei van Eva

Wat hebben we laten zien aan Eva?

- login
- registratie
- thema's

Feedback Eva

1. Login

Lekker bezig, ziet er goed uit! Super strak

2. Registratie

Super nice! Achtergrond bewegend zou ik gaaf vinden, maar is natuurlijk nu geen prioriteit. Ik wil wel dat de opties van de "Type Kanker" stap breder kunnen zodat het "grid" wat beter is. (net zoals op de Adobe)

3. Thema's

kijk nog even goed naar de styling. Hij ziet er wel echt goed uit.

Week 3

Taken (streven) deze week

Rowin	Ralf
Documentatie updaten	Register & Login: post data naar server om account aan te maken/in te loggen
user scenario: themes	user scenario: buddies, buddie detailpage, chat (Sockets voor chatfunctie)
Maken van navbar	Bonus: profielData zien en kunnen aanpassen (updaten)
Pair coding	
Salting passwords	
Route guard	

Post form data naar server

Vorige week lukte het posten van data naar de server niet. Dus Rowin en ik zijn er samen naar gaan kijken. We zijn gelijk aan de slag gegaan met het post submitten naar de server, zodat de server iets terug returned naar de client. Na 2,5 uur worstelen, was het gelukt. Het ging namelijk eerst fout met de input component en de props meegeven aan data component. Daarna hebben we ook gelijk de Axios functie aangepast. Nu het inloggen gelukt is, wil ik ook het registreren voltooien. Dat was eigenlijk best eenvoudig, ik moest namelijk dezelfde axios functie schrijven en daarvan de data (die in de Vuex Store werd bijgehouden) bij de laatste stap submitten naar de server. Na 1,5 dag had ik dan eindelijk deze user scenario afgerond. ✅

Post submit functie | Login

```
async login(e) {
  let data = {
    emailAddress: this.emailAddress,
    password: this.password,
  };
  axios
    .post("/api/login", data, { headers: { "Content-type": "application/json" } })
    .then((response) => {
      if (response.status === 200) {
        console.log(response);
        this.$store.commit("updateUser", response.data);
        return this.$router.push("/themes");
      }
    })
    .catch((err) => {
      this.errors.push("Er is helaas geen account gevonden");
      console.log(err);
    });
},
onSubmit(e) {
  e.preventDefault();
  this.login();
},
```

Post submit functie | Register

```
onSubmit(e) {
  e.preventDefault();

  if (this.stepState === 9) {
    // get data from vuex store
    console.log("user data, ", this.$store.state.user);

    // POst submit to server
    axios.post("/api/register", this.$store.state.user, {
      headers: { "Content-type": "application/json" },
    });
  }
```

Registratie | Buttons bepalen dmv state

Ik wilde het voor elkaar krijgen dat ik het “Starten” knopje op stap 1 en de laatste (ready) pagina liet veranderen in de tekst “Volgende”. Dat werkte eerst niet en ik dacht dat ik helemaal moest worstelen met props. Maar toen kwam ik erachter dat er ook de mooie implementatie van Vue is: v-if en v-else.
Dit werkt erg handig als je verschillende elementen, in mijn geval buttons, wil laten zien met tekst erin, maar die tekst moet veranderen per state.

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

Hieronder zie je wanneer de state step 1 of step 9 is, dan laat die de button zien met “Starten” erin. Zo niet, dan staat er “Volgende” in.
v-if en v-else werken heel handig als je verschillende buttons wil laten zien met tekst erin, maar die afhangen van een bepaalde state. Zoals wat ik hier heb gedaan.

```
<button v-if="stepState !== 1 && stepState !== 9" @click="setState('next')">
  Volgende
</button>

<button v-else @click="setState('next')">Starten</button>
```

Ook wilde ik de “vorige” button op alle stappen behalve op stap 1 en laatste pagina laten zien, anders niet.

```
<button v-if="stepState !== 1 && stepState !== 9" @click="setState('prev')"></button>
```

Na deze stappen waren de features login en register helemaal af. Ook qua styling waren ze helemaal mooi en clean afgemaakt.

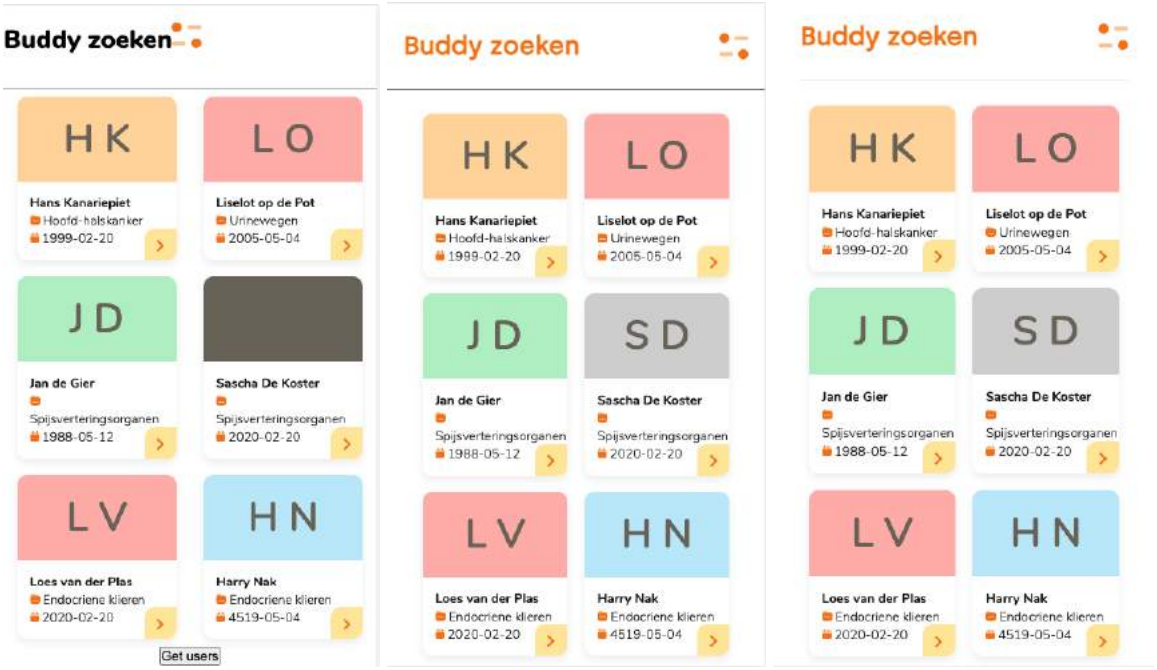
Profiel

Ik heb toch even een profiel aangemaakt, omdat ik toch al de data binnenkreeg en die wilde ik gewoon even renderen in de pagina, zodat ik (en vrijdag de opdrachtgever) het zeker weet dat er een user is ingelogd. Dit was vrij simpel, omdat ik de data in de Vuex store had gezet. Daardoor was het een niet heel complexe implementatie. Verder heb ik de ruwe opzet van de User Interface gemaakt, zodat het al een beetje lijkt op de UI van het prototype. Zie hieronder een impressie. Ik heb ook de iteratie die ik later deze week heb gemaakt hierachter gezet:

Chat | Buddies (alle users)

Om de chat te realiseren, moet ik eerst gebruikers aanmaken. Dat zal ik doen door middel van het registratie formulier. Daarna ga ik aan de slag met een GET request maken naar de server om alle users te krijgen, zonder de ingelogde user uiteraard. Daarna zal ik als ik naar een bepaalde user ga, het ID van de user in de URL zetten, waardoor ik met een axios GET request naar de server de specifieke data ophaal uit de database en terugstuur naar de client. Vervolgens render ik de data in de detailpagina.

De chat functie komt later. Dit zal gaan werken met sockets.io of misschien een andere tool, maar dat is voor later deze week. Voor nu moet ik dus eerst alle users renderen. Begint al aardig op het prototype te lijken:



Ik ben eerst de kaartjes gaan maken. Vervolgens ben ik de layout gaan stylen in flexbox. Toen dat klaar was, ben ik de data gaan ophalen uit de server door middel van een button die je onderaan ziet in de 1e afbeelding (links). Op die button zit een event die een axios GET request doet naar de server om alle users op te halen. Wanneer de users verzameld zijn, stuurt de server de user data terug naar de client, die hem in de data() zet van de pagina en vanuit daar rendert hij de pagina met de data die daarin staat. Hierna heb ik de button veranderd en de functie aangeroepen in de mounted(). Dit was nieuw voor mij, maar wel hoe ik het had willen hebben. Dus wanneer de pagina bezocht wordt, wordt de functie uitgevoerd en dus data opgehaald.

Connectie database met GET request

De connectie met de database is gelukt, dus we kunnen AXIOS post en get methods naar de server sturen, die vervolgens iets naar de db stuurt en we kunnen wat terugsturen weer naar de client. Dit was een erg sterke inhaalslag. We hebben hier samen namelijk wel even wat uurtjes aan gezeten.

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

Tussentijdse Bugs

- Ik merk dat bij elke page refresh, de Vuex store wordt weggegooid. Dus we moeten hier wat op verzinnen, zoals bijvoorbeeld LocalStorage ipv de Vuex Store. Of sessions. Rowin heeft dit opgelost met localstorage in combinatie met de Vuex Store.
- Ook de issue met features mergen naar master gaat gewoon helemaal niet goed. Master neemt deels merges over en sommige ook niet. Dit kost een hoop tijd. Heroku neemt niet de juiste PORT over waardoor de submits falen. Na vaak proberen ging het stuk, maar na 2 uur deed hij het weer.

Idee | Header dynamisch component

ik heb de header van alle pagina's consistent gestyled en hierdoor ziet de gehele app er stuk professioneler uit. Om die reden wil ik van de header een component maken om dynamisch op elke pagina in te laden met props voor de naam van de pagina en dat er ruimte is voor het icoon dat (meestal) rechts staat uitgelijnd.

Chat | Datastructuur

Om de structuur te bepalen van de chats moet ik dat eerst even goed bedenken. Ik heb al een idee in mijn gedachte en die wil ik graag voortzetten. Ik zie het namelijk zo voor me.

```
chats: [
  {
    participant: '109sje0xsnsla9sd9asjsfl',
    messages: [
      {
        sender: '109sje0xsnsla9sd9asjsfl',
        content: 'Hi, how are you?',
        time: '15 Mei, 12:17',
      },
    ],
  },
],
```

Dus iedere user heeft een chat key in zijn object staan. In die key zit een array met daarin allerlei objecten die de chats met verschillende users aanduiden. Per chat object zitten er participant en messages. De participant geeft aan welke user er ook in die chat zit, het userID. En messages is een array met message objecten met de keys: sender, content en time. De sender is het userID, content is het bericht dat is verstuurd

en time is de tijd waarop het bericht verstuurd is. Dit lijkt mij een erg logische structuur van het bijhouden van de chats.

Documentatie updaten

Helaas hoort het bijwerken van de documentatie er soms ook bij, ik heb de focus gelegd op het updaten van de README, ik heb er voor gekozen om dit op tijd bij te houden omdat het project steeds groter en groter wordt. Als dit last-minute nog gedaan zou moeten worden dan is het opeens een hele grote waslijn qua content/data die er in moet komen.

User scenario: themes

Vorige week had ik mijn user scenario gedeeltelijk af, alleen miste ik hier en daar nog de juiste content, toen ik verder in het prototype ging kijken merkte ik dat elk thema weer een andere heading kon hebben of juist geen inleiding van het thema. Deze content zo structureren zodat het er overal goed uit ziet duurde wel even. Ook heb ik de verschillende toggles weten te maken (klachten, tips & hulp) hiermee kan de gebruiker door de verschillende tabbladen heen en meteen de juiste data tevoorschijn krijgen.

In deze images ziet u de content van het tabblad: klachten en als de gebruiker op tips drukt krijgt de gebruiker andere content te zien.

Modules maken

Toen ik verder was gekomen met de user scenario, besefte ik me opeens dat we nog geen navbar hadden gemaakt die wel essentieel is vooral nu dat Ralf al dichterbij komt met het inloggen en registreren.

.

Ik heb hierbij gebruik gemaakt van props, ik gebruik de props om aan te geven welk icoon er oranje moet zijn (active) en welke niet, hierin heb ik dus verschillende mogelijke props:

home

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

buddies
messages
profile

Hiermee kan je dus aangeven welke er active is en er dus een andere class qua styling krijgt.

Pair coding

Ik heb deze week ook tijd vrij gemaakt om Ralf te kunnen helpen bij het maken van zijn features/branches, dit kost soms best veel uren en als je er dan ook alleen naar moet kijken kan het helemaal een woestijn worden.

Password salting

Toen we begonnen met de registratie/login feature ging we de data als plain tekst opslaan, dit deden we omdat we als eerste prioriteit hadden dat het zou werken, zodat we het konden laten zien aan Eva. Ik ben vervolgens terug in de tijd gegaan en mijn oude repo bekeken van project-tech waarin ik ook passwords eerder had gesalt.

```
const passwordHash = bcrypt.hashSync(req.body.password, saltRounds);

const userObject = {
    firstName: req.body.firstName,
    surName: req.body.surName,
    emailAddress: req.body.emailAddress,
    password: passwordHash,
    birthDate: req.body.birthDate,
    town: req.body.town,
    gender: req.body.gender,
    typeIllness: req.body.typeIllness,
    profileAvatar: req.body.profileAvatar,
    about: req.body.about,
};
```

Met behulp van bcrypt kunnen we gemakkelijk passwords salten en valideren, hierboven ziet u een image hoe het wordt opgeslagen, vervolgens wordt het opgeslagen in het database als het volgende:

Het wachtwoord is dus niet meer te lezen met het blote oog. Dit zorgt voor extra security maatregelen.

Route guard

Toen we de themapagina af hadden en het inloggen en registreren, was het de bedoeling dat de gebruiker meteen naar de gewenste pagina zou gaan en als het ware een soort state zou krijgen van: ingelogd. In het begin hadden we hier nog niks mee gedaan.

Dus konden gebruikers in een keer door naar de pagina die zij wilde bezoeken zonder ingelogd te zijn of zich te registreren, hiervoor heb ik de volgende oplossing voor bedacht: Route guard.

Route guard zorgt ervoor dat we bepaalde routes een meta data kunnen meegeven, hierin kunnen we dus bijvoorbeeld zeggen:

```
{
  path: "/profile-edit",
  name: "ProfileEdit",
  component: ProfileEdit,
  meta: {
    requiresAuth: true,
  },
```

We geven dus op dat deze route een requiresAuth nodig heeft.

```
router.beforeEach((to, from, next) => {
  if (to.matched.some((record) => record.meta.requiresAuth)) {
    if (!store.state.loggedIn) {
      next({
        name: "Login",
      });
    } else {
      next();
    }
  } else {
    next();
  }
});
```

Om het echt helemaal werken te maken, loopen we over de routes, en als de route de meta data bevat van requiresAuth, kijkt hij naar de state in de route of die true of false is, zodra het false is rendert vue het component login, en anders gebruiken we de next functie.

Vervolgens is het de bedoeling om de state in de store te updaten zodra een user inlogt of zodra een user zich registreert..

```
axios
  .post("/api/login", data, { headers: { "Content-type": "application/json" } })
  .then((response) => {
    if (response.status === 200) {
      this.$store.commit("updateUser", response.data);
      return this.$router.push("/themes");
    }
  })
```

Zodra de server een response status geeft van 200 wordt de state van de user loggedin naar true gezet.

Feedback Design review 1 - Koop

We hebben in dit gesprek eerst het project besproken, verteld wat onze kritische blik daarop was en wat de aanpassingen waren naar aanleiding daarvan en wat de houding van de opdrachtgever is.

Ons werd de vraag gesteld wat de meerwaarde van ons is als frontenders op het moment dat je een project krijgt waarvan het concept al grotendeels af is. Wij zijn er om het ontwerp en de UX te verbeteren. Ook vroeg Koop ons de vraag van Wat als er heel veel data is in de applicatie. Hoe functioneert hij dan? En hoe reageert de user daarop en hoe gaat hij/zij daar mee om? In een clickable prototype zoals Adobe XD kun je het prototypen maar niet realiseren omdat er nooit echte data in staat. Daarom is het ook aan ons om de app te testen, puur omdat er ook echte data in staat en dan kijken we of die ideeën daadwerkelijk goed werken. Vervolgens stelde Koop ons een aantal vragen, wat ons heeft gebracht tot 4 features waar wij een meerwaarde/bijdragen kunnen leveren aan het concept als frontenders, namelijk;

1. UI Stack. De loader bij het opstarten van de applicatie hoort daar niet vinden Rowin en ik, omdat er geen externe data dan wordt opgehaald. Waar moet die loader dan wel staan? Dit kunnen we testen op het moment dat de applicatie *veel* data bevat.

2. In het prototype staat de vraag ‘Waar woon je’ in het registratieproces, en als dropdown keuzes staan er maar een paar statische plaatsnamen. Maar misschien is het goed om eerst de GeoLocatie te vragen. En als dit wordt geweigerd, hoe handelen we dit dan af? Met behulp van een steden API kunnen we dit heel mooi afhandelen.

3. In het prototype staan 1-10 buddies maar wat zou er gebeuren als hier bijvoorbeeld 900 buddies zouden staan, hoe ga je het beste filteren? met de huidige filter is er nog steeds de kans dat je heel veel buddies terug krijgt vanuit het filter resultaat en vinden wij dat het huidige filter zich meer richt op de discoverability. Met wellicht de hulp van het zoeken op een naam zou deze experience al een stuk beter kunnen worden, dat de gebruiker d.m.v een tekstveld de naam kan opzoeken van zijn/haar buddy.

4. In het prototype kan je chatten, maar hoe gaat dat eruit zien en leven op het moment dat er veel data aanwezig is. Hoe laten wij dit zo soepel mogelijk verlopen.

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

Het is nu aan ons om suggesties te geven aan Eva, zodat zij daarop kan inspelen en haar mening over kan loslaten. Ook kan zij hierin meedenken en zo kunnen we samen op mooie oplossingen uitkomen.

Server | Mappenstructuur

Ik heb de mappenstructuur van de server aangepast, want het ene bestand (server.js) waar alles in stond werd een beetje overvol en ik houd van overzichtelijkheid, dus ik zal u even meenemen door de nieuwe mappenstructuur.

Ik heb zoals u hier ziet de server opgedeeld in de root en het mapje **app**. De app heb ik onderverdeeld in **config, controllers, helpers, models** en **routes**. In config staat alles van de database opgesteld, in models staat het datamodel van wat er in de database komt te staan. Dan hebben we in routes alle routes staan van de applicatie en de functies van de routes komen uit het mapje controllers en dan user.controller.js . Daarnaast staat er in het mapje helpers het db.helpers.js bestand waarin alle functies

staan die gaan over de database, zoals het opzoeken van een user met behulp van het userID. Dit is een stuk duidelijkere weergave van de server en netjes te overzien. Het moet alleen nog voorzien worden van code comments.

Chat | Datastructuur v2.0

Vandaag ben ik samen met Rowin de datastructuur voor de chat gaan bepalen. We hebben even in een call gezeten met elkaar en wat datastructuren over en weer gestuurd, maar we zijn toch tot de conclusie gekomen dat de datastructuur die ik gemaakt heb het handigst is. Maar er moet nog een aanpassing gebeuren en dat is dat participant moet veranderen in een object die bestaat uit **ObjectID, firstName** en **surName.** Omdat we anders steeds een GET request naar de server moeten doen, om erachter te komen van wie het bericht is. Zie de datastructuur hieronder:

```
chats: Array
  0: Object
    participant: Object
        id: "60bde60ffedeac4da052be25"
        firstName: "Gerrit"
        surName: "De slak"
        profileAvatar: "orange"
    messages: Array
      0: Object
          sender: "60bde60ffedeac4da052be25"
          content: "Hey Harry, ik ben Ger. Alles goed?"
          time: "17 mei 2021, 12:15"
```

Chat | Implementatie sockets

Het was een afgang die sockets. Uren hoofdpijn samen met Rowin. Het was een onwijze prestatie deze week maar het is echter niet gelukt. Volgende week gaan we deze feature proberen af te ronden.

Ik was aan de styling van de chat begonnen, maar daarna toch snel afgehaakt van de cursus, omdat ik toch erachter kwam dat ik eerst chat data moest ontvangen en renderen in de pagina. Toen kwam ik tot de conclusie dat mijn datastructuur niet helemaal lekker was. Om die reden heb ik samen met Rowin op het eind van de dag ernaar gekeken.

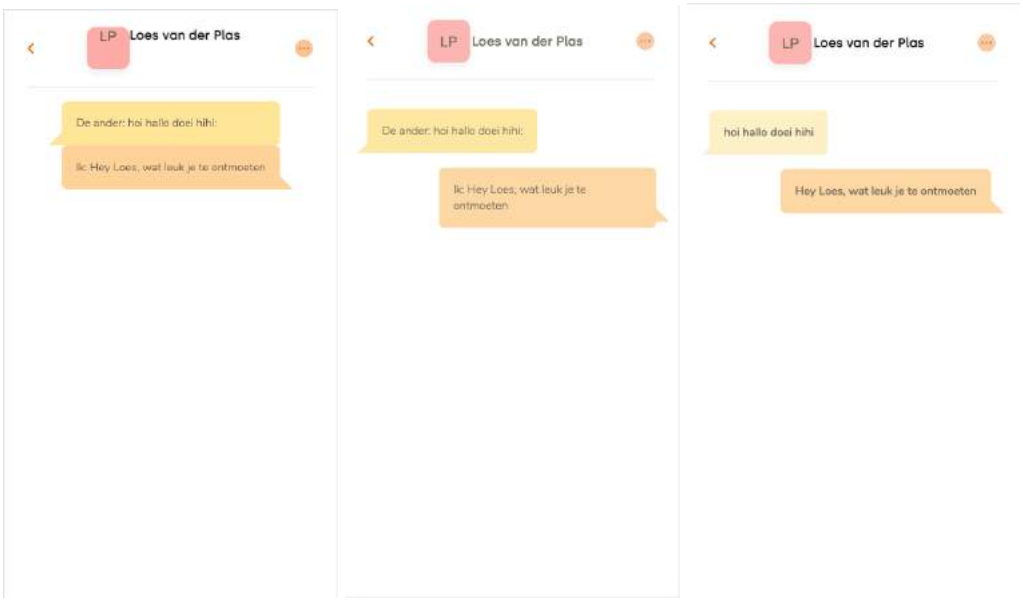
Bronnen:

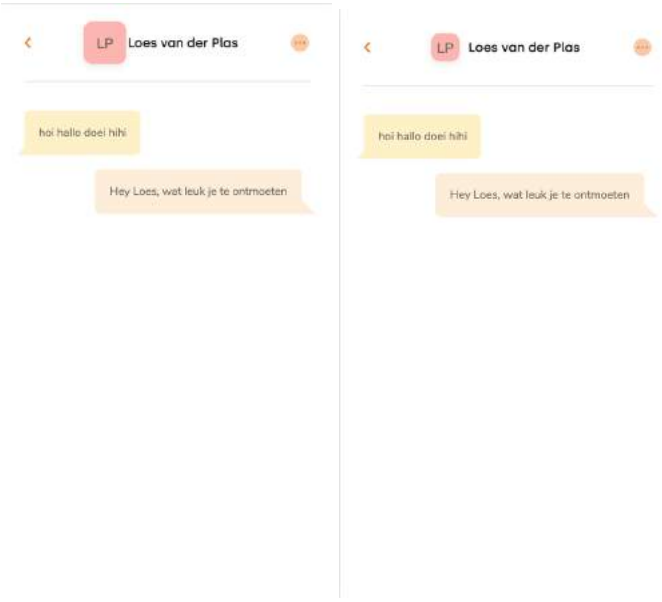
- https://medium.com/js-dojo/build-a-real-time-chat-app-with-vuejs-socket-io-and-nodejs-714c8eefa54e

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

Chat | Styling

De styling van de chats was nog een lastig puntje, omdat ik het driehoekje van het tekstballon perfect wilde hebben. Dat was dus eerst even puzzelen, maar al snel had ik gevonden hoe ik dat het beste kon doen. Het was mij echter onduidelijk waar de kleuren van de tekstballonnen vandaan kwamen. Of beter gezegd: waar die op gebaseerd waren. Nadat ik het aan Eva gevraagd had, bleek het dus te zijn dat lichtgeel voor de persoon is met wie je praat en lichtoranje is de kleur die jij bent. net als bij andere social media platformen waar je kan chatten is hier visuele hiërarchie toegepast. Hieronder zie je het proces van het stylen van de chat detailpagina:





Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

Feedback 4 juni van Eva

Wat hebben we laten zien aan Eva?

- login -> helemaal af qua styling en functie
- registratie -> helemaal af qua styling en functie
- thema's -> helemaal af qua styling en functie
- buddies
- buddie detail
- chat
- chat detail
- profile

Feedback Eva

1. Login
    - De error state van email/wachtwoord/account is niet uitgewerkt in design maar mag hetzelfde zijn als de andere die wel in het design staat.

"Voor de rest zien alle schermen er fantastisch uit en werkt erg fijn!"

We hebben ook het principe waar we het deze week over hebben gehad tijdens de design review voorgelegd en toen kwam Eva met iets bijzonders:
Bij de vraag "Waar woon je?" tijdens het registratieproces wilt Eva in plaats van een dropdown (wat wij dachten in het design te zien) een zoekbalk waar je de naam van je woonplaats kan intikken en dat er dan met autofill plaatsnamen gegenereerd worden waaruit je dan kan kiezen. Dit is erg gaaf en hiervoor hebben we dan wel een API nodig die de plaatsnamen geeft, maar dat is denken wij geen probleem.

Week 4

Taken (streven) deze week

Rowin	Ralf
Vuex store wordt gecached/opgeslagen.	user scenario: Sockets voor chatfunctie
PWA	db history chat
Bug fixing	de gehele chat feature afmaken

Echter lopen we met sockets iets achter, maar dat is wat je tegenkomt tijdens het developen in de praktijk. Dingen lopen nooit altijd zoals je het gepland had. Maar we proberen dat deze week in te halen.

Sockets voor chatfunctie

We hebben echt een hele tijd met de sockets zitten worstelen, maar het kwam eigenlijk op een bug neer die heel eenvoudig was. We hadden een http server aangemaakt, maar de app luitsterde (app.listen) naar de PORT in plaats van dat de http server dat deed. Daarom hadden we steeds een error in de console staan. Door dit dus aan te passen naar dat de socket server hiernaar luistert, werkte het eindelijk.
Hierna begon ik eerst met het joinen van de chatroom. Daarna het emitten van een event (chatbericht) en daarna het zetten in de database.
Ik heb eerst een hele tijd gestoeid met de datastructuur, maar die is uiteindelijk een chat object geworden die losstaat van de user objects. U zult onderaan lezen hoe die datastructuur er nu uit ziet.
Uiteindelijk heeft deze implementatie 3 / 4 dagen geduurd. Maar ik ben er wel echt wijzer van geworden. Om sockets werkend te krijgen in Vue is eerst even goed puzzelen, maar eindstand is het gelukt. Want ik ben van hardcoded data met de hand in de database naar zelf data in de database implementeren via sockets gegaan.

Hier staan de socketevents clientside:

```
mounted() {
  this.socket.emit("joinRoom", {
    userID: this.$store.state.user._id,
    roomID: this.$route.params.id,
  });

  this.socket.on("roomData", ({ room, participant }) => {
    // Store participant
    let participantData = participant;
    this.participant.push(participantData);

    let userID = this.$store.state.user._id;
    // Check if this is a chatRequest, then put request to true
    let requestState = room.request.accepted;
    let requestCreater = room.request.creater;

    // check if chatCreatorID is the same as this user --> render chat
    if (requestCreater == userID) {
      // set view to viewCreater: false
      this.viewCreater = true;
    }

    if (requestState == true) {
      this.requestAccepted = true;
    }

    // Store messages history
    let messages = room.messages;
    this.messagesHistory.push(messages);
  });

  this.socket.on("newMessage", (messageObject) => {
    // console.log("New Message: ", messageObject);
    this.messages.push(messageObject);
  });
},
```

Hier staan de socketevent “listeners” serverside:

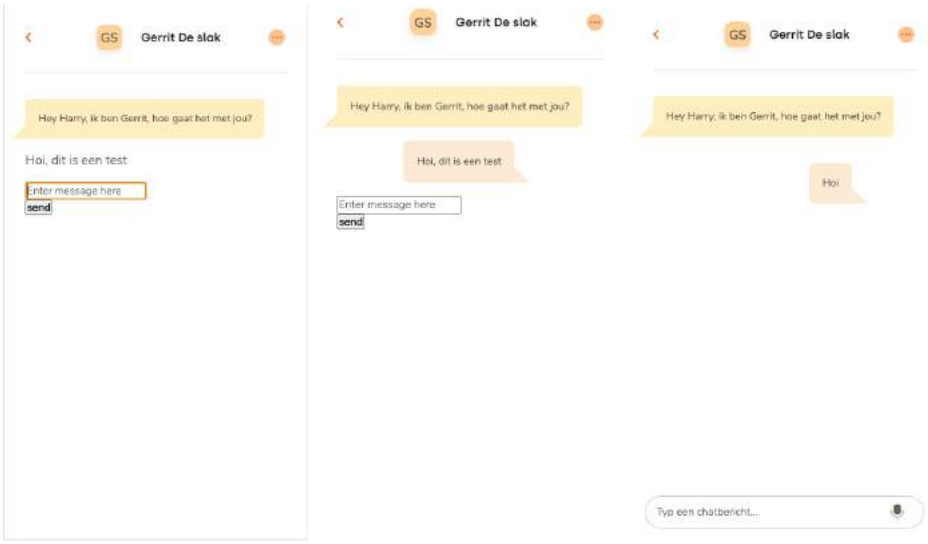
```
exports.ioEvents = async (socket, server) => {
  socket.on('joinRoom', async ({ userID, roomID }) => {
    joinRoomHandler(socket, server, userID, roomID);
  });

  socket.on('chat-message', ({ roomID, sender, content, time }) => {
    newMessageHandler(server, roomID, sender, content, time);
  });

  socket.on('disconnect', () => {
    leaveRoomHandler(socket, server);
  });
};
```

Hier staan de socketevents serverside:

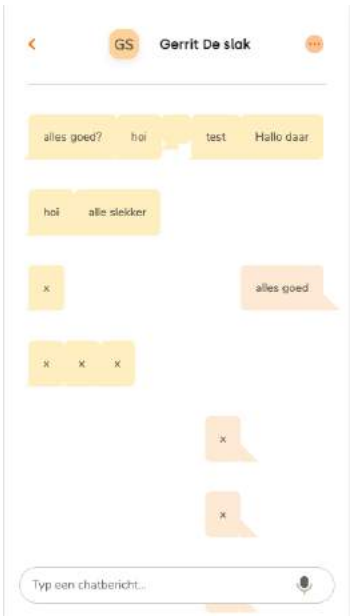
```
async function joinRoomHandler(socket, server, userID, roomID) {
  socket.join(roomID);
  roomGlobal = roomID;
  globalUser = userID;

  // Data of a chatroom
  const roomData = await findOneChat(roomID);

  // Participant data
  const participantID = roomData.participants;

  for (let i in participantID) {
    let partiUser;

    if (participantID[i] !== globalUser) {
      const userData = await findOneUser(participantID[i]);

      partiUser = userData;

      let wholeObject = {
        room: {
          request: {
            creater: roomData.request.creater,
            accepted: roomData.request.accepted,
          },
          participants: roomData.participants,
          _id: roomData._id,
          messages: roomData.messages,
        },
        participant: {
          firstName: partiUser.firstName,
          surName: partiUser.surName,
          profileAvatar: partiUser.profileAvatar,
          id: partiUser.id,
        },
      };

      server.to(roomID).emit('roomData', {
        room: wholeObject.room,
        participant: wholeObject.participant,
      });
    } else {
      // console.log('Not the same');
    }
  }
}

/**
 * New chat message handler
 *
 * @param {String} server - socket server (IO)
 * @param {String} roomID - ID of the chatroom
 * @param {String} sender - ID of the sender of the message
 * @param {String} content - Content of the message
 * @param {String} time - Time of the message
 *
 * @return {Object} messageObject - Object with all content of the new message
 */

function newMessageHandler(server, roomID, sender, content, time) {
  const messageObject = {
    sender: sender,
    content: content,
    time: time,
  };

  // 2. Put data obj in chatObj in db
  updateChatMessages(roomID, messageObject);

  // Return the new message to chatroom
  server.to(roomID).emit('newMessage', messageObject);
}

/**
 * Handles when user leaves chatroom
 *
 */

function leaveRoomHandler(roomID) {
  console.log('A user disconnected');
}
```

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

Omdat ik natuurlijk de realtime chats implementeer, maar ook de chathistory, zorg ik ervoor dat eerst de chathistory wordt gerenderd en daaronder de nieuwe chatmessages. In het prototype staat geen verzendbutton, die is niet nodig omdat het genoeg is met een enter. Maar ik bedacht me moet er geen verzendbutton zijn als de enter buttons niet meer werken op telefoons. Dan is een verzendbutton wel een goede fallback. Hieronder zie je de styling proces van de typbalk van de chat:



-

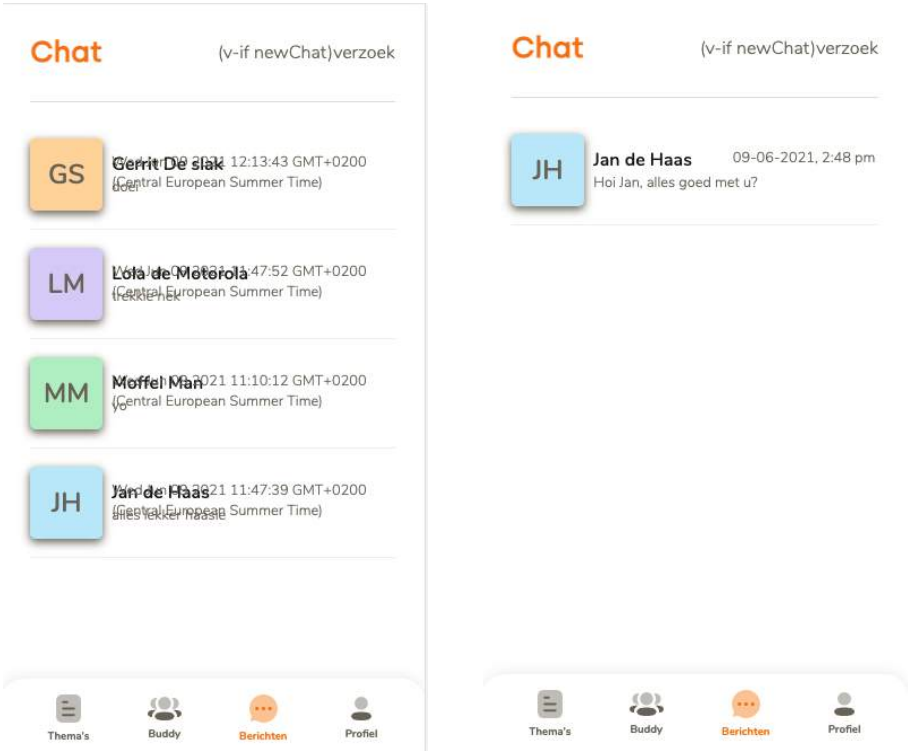


Chat | tijd van het bericht

De tijd van ieder chatbericht werd niet mooi opgeslagen vond ik en daarom moest er een betere notatie komen. Ik heb hiervoor gebruik gemaakt van [moment.js.](moment.js) Dit is een super mooie package die de tijd omrekent naar het formaat dat ik wil hebben. Hieronder zie je hoe ik dit gedaan heb:

```
send(e) {
  e.preventDefault();
  let date = new Date();
  let formattedDate = moment(date).format("DD-MM-YYYY, hh:mm a");

  let chatObject = {
    roomID: this.$route.params.id,
    sender: this.$store.state.user._id,
    content: this.newMessage,
    time: formattedDate,
  };

  this.socket.emit("chat-message", chatObject);
  this.newMessage = null;
  // this.$refs.newMsg.focus();
},
```

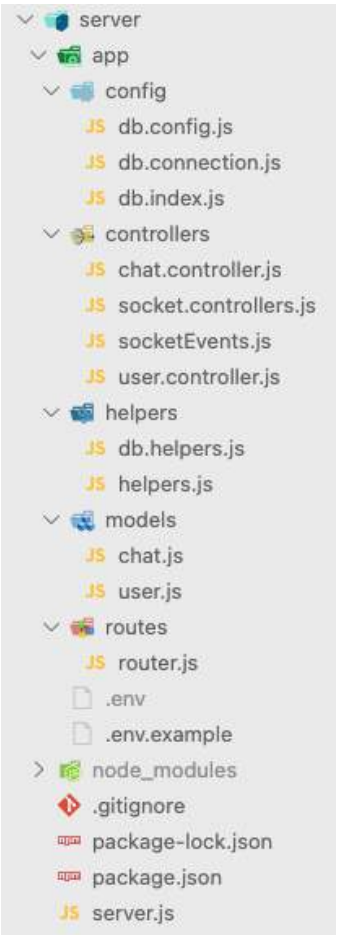
Links was eerst, rechts is nu



Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

Server | Mappenstructuur v2.0

Ik heb weer de hele mappenstructuur aangepast. Hierbij heb ik gelet op de socket events en alle benamingen van functies en variabelen serverside. Nu staan de socketevents en listeners los van elkaar en in aparte bestanden (zie hierboven bij “Sockets voor chatfunctie”). Hierdoor is de serverside super overzichtelijk en is de server.js echt clean. Hieronder laat ik de mappenstructuur zijn en de server.js




```
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors');
const path = require('path');
const dbConnection = require('./app/config/db.connection.js');
const router = require('./app/routes/router.js');
const app = express();
const http = require('http').createServer(app);
const io = require('socket.io')(http, {
  cors: {
    origin: 'http://localhost:8080',
  },
});

const eventHandler = require('./app/controllers/socketEvents.js');

io.on('connection', (socket) => {
  eventHandler.ioEvents(socket, io);
});

app.use(cors());

app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());

app.use(router);

app.use(express.static(__dirname + '/public/'));

// Handle SPA
app.get(/.*/, (req, res) =>
  res.sendFile(path.resolve(__dirname, 'public/index.html'))
);

// set port, listen for requests
const PORT = process.env.PORT || 5000;
http.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}.`);
});
```

Chat datastructuur v3.0

De datastructuur van de chats zijn aangepast. Ik heb de request key object toegevoegd, want op deze manier kunnen de chatverzoeken goed worden afgehandeld, zowel op de frontend als op de backend. Wanneer er een chat wordt aangemaakt, staat request.accepted op false en is de chat dus nog niet door de ander geaccepteerd. Ook is request.creater de organisator van de chat. Op het moment dat de ander de chat accepteert komt request.accepted op true te staan en zal de UI veranderen van chat request naar

normale chat detailpagina. Maar op het moment dat de ander kiest voor afwijzen, zal het chat object in de database gelijk verwijderd worden. Hier is echter door de opdrachtgever nog niet nagedacht over feedback voor de organisator van de chat. Ik zat zelf te denken om een soort alert bij de organisator achter te laten van “John heeft je chatverzoek afgewezen.”. Zo ziet de chatobject er nu uit:

Vuex store wordt gecached/opgeslagen

Er was een probleem waar wij de hele tijd tegen aan liepen, en dat was dat de vuex store na een browser refresh leeg werd gemaakt, hierdoor werden sommige states of data types automatisch weer leeg, en kregen we heel vaak de errors dat er iets niet gevonden kon worden omdat het simpelweg empty was.

Toen kwam ik op het idee om het wellicht in de localstorage op te slaan met de vuex-persistedstate deze package zorgt er dus voor dat de vuex store automatisch wordt opgeslagen in de localstorage en automatisch weer neerzet, een echte must have om te hebben voor ons goal.

```
plugins: [
  createPersistedState({
    storage: {
      getItem: (key) => ls.get(key),
      setItem: (key, value) => ls.set(key, value),
      removeItem: (key) => ls.remove(key),
    },
  }),
],
```

Ik heb vervolgens de plugin in geregistreerd in de index.js van de store en hier 3 functies gemaakt voor het ophalen van de items, het updaten van de items, en het verwijderen van de items. Wat LS precies inhoudt ga ik hieronder verder op in.

Toen de localstorage opgeslagen werd was dit plain text, nu kan localstorage soms een ding zijn qua privacy of persoonlijke data, vandaar dat ik ervoor heb gekozen om de localstorage met behulp van LS te encrypten, hierdoor wordt het dus voor de gebruikers en andere heel onduidelijk om te zien wat er precies in staat. Dit is dus gedaan vanuit een security gevoel.

Zoals u hier kan zien, is er in de localstorage een vuex attribuut en een secure_ls_metadata, deze metadata is nodig om de localstorage te kunnen decrypten. Hierdoor hebben wij verder kunnen werken aan de app zonder enige problemen meer, het werkte meteen lekker en werkte zoals behoren zoals wij dat wilde.

PWA

Toen we met dit project begonnen was het idee meteen om van de website een PWA te maken, met behulp van een PWA kunnen gebruikers pagina's cachen, op het moment dat de gebruiker geen internet heeft kan hij/zij alsnog gebruik maken van onze applicatie omdat deze pagina gecached staat op het device.

Wij kunnen dus onze dienst voortzetten zelfs als de gebruiker offline is (m.u.v de chat), hiervoor gebruiken wij de vue built-in tooling voor PWA.

```
/* eslint-disable no-console */

import { register } from "register-service-worker";

if (process.env.NODE_ENV === "production") {
  register(`${process.env.BASE_URL}service-worker.js`, {
    ready() {
      console.log(
        "App is being served from cache by a service worker.\n" +
          "For more details, visit https://goo.gl/AFskqB"
      );
    },
    registered() {
      console.log("Service worker has been registered.");
    },
    cached() {
      console.log("Content has been cached for offline use.");
    },
    updatefound() {
      console.log("New content is downloading.");
    },
    updated() {
      console.log("New content is available; please refresh.");
    },
    offline() {
      // eslint-disable-next-line max-len
      console.log("No internet connection found. App is running in offline mode.");
    },
    error(error) {
      console.error("Error during service worker registration:", error);
    },
  });
}
```

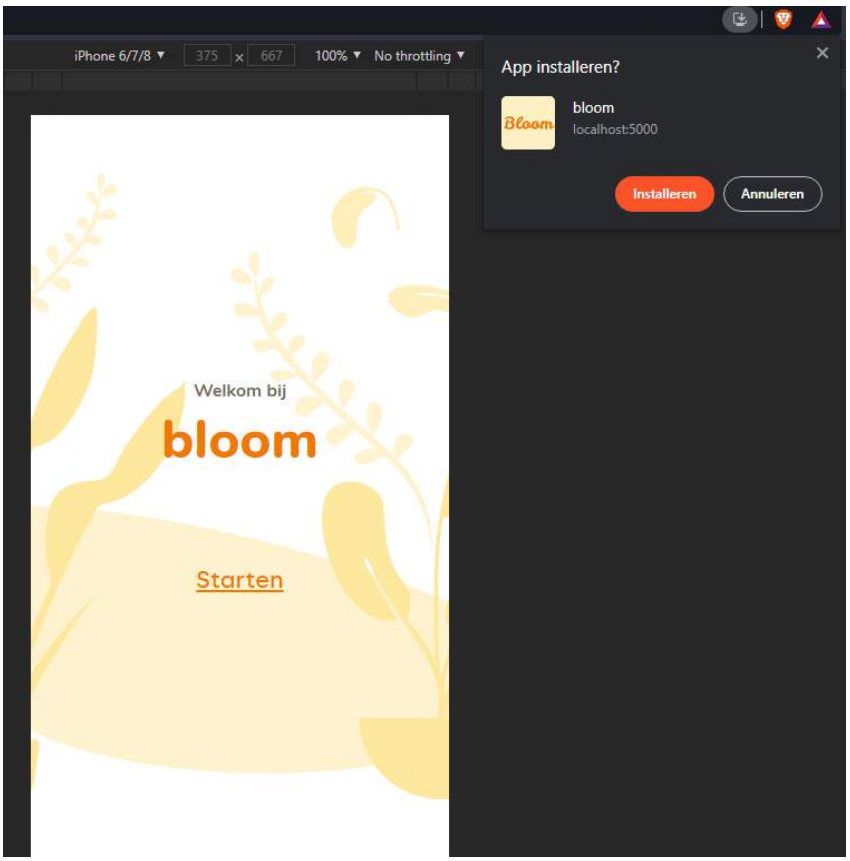
Er was echter nog een paar problemen, zoals dat vue de service-worker niet kon herkennen of vinden, na een tijdje zoeken bleek dat de service-worker in de root moest staan, met deze aanpassing liep de service worker als een trein🚄.

Voor ons was het ook de wens op de PWA zo aan te passen zodat het echt bij bloom hoort en dat de gebruiker ook echt de feeling heeft dat hij/zij het product gebruikt dat afkomstig is van bloom zelf.

Daarom heb ik de volgende aanpassingen gemaakt:

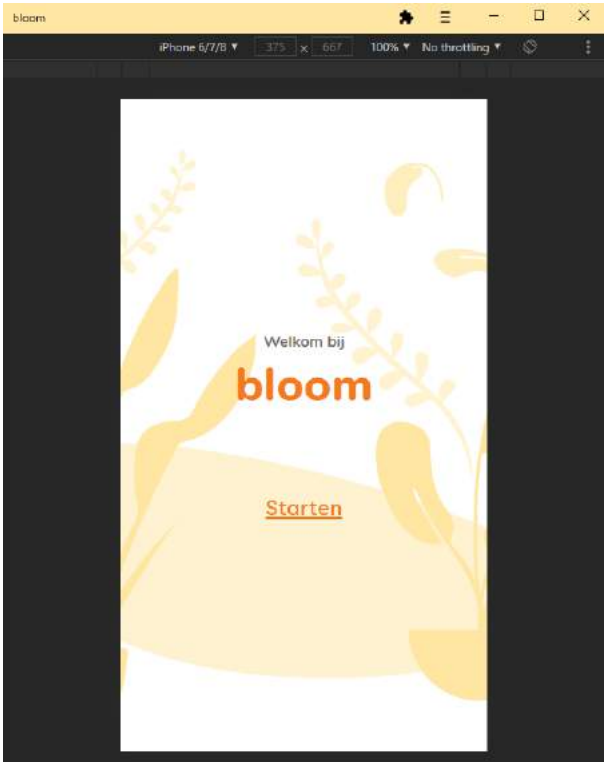
```
pwa: {
  manifestOptions: {
    name: "bloom",
    short_name: "bloom",
    start_url: "./",
    theme_color: "#fef1c5",
    msTileColor: "#ffffff",
    appleMobileWebAppCapable: "yes",
    appleMobileWebAppStatusBarStyle: "black",
    icons: [
      {
        src: "./img/icons/favicon-32x32.png",
        sizes: "32x32",
        type: "image/png",
      },
      {
        src: "./img/icons/favicon-16x16.png",
        sizes: "16x16",
        type: "image/png",
      },
      {
        src: "./img/icons/android-chrome-512x512.png",
        sizes: "512x512",
        type: "image/png",
      },
      {
        src: "./img/icons/android-chrome-192x192.png",
        sizes: "192x192",
        type: "image/png",
      },
    ],
    // configure the workbox plugin
    workboxPluginMode: "InjectManifest",
    workboxOptions: {
      // swSrc is required in InjectManifest mode.
      swSrc: "./registerServiceWorker.js",
      // ...other Workbox options...
    },
  },
},
```

In vue.config kunnen we de pwa aanpassen, daarin heb ik dus het nodige toegevoegd denk hierbij aan: images, theme colors etc. Toen ik dit aangepast had ik geprobeerd om de vue te builden en te runnen op de server, zodat ik hem kon installen en dit was het resultaat:

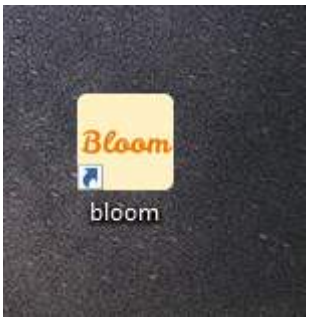


Hier kunnen we app installeren.

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development



De shell wordt geopend zodra die geïnstalleerd is



We hebben een Bloom icoon gekregen op onze desktop pagina, als we deze openen krijgen we meteen de shell van de website voor ons.

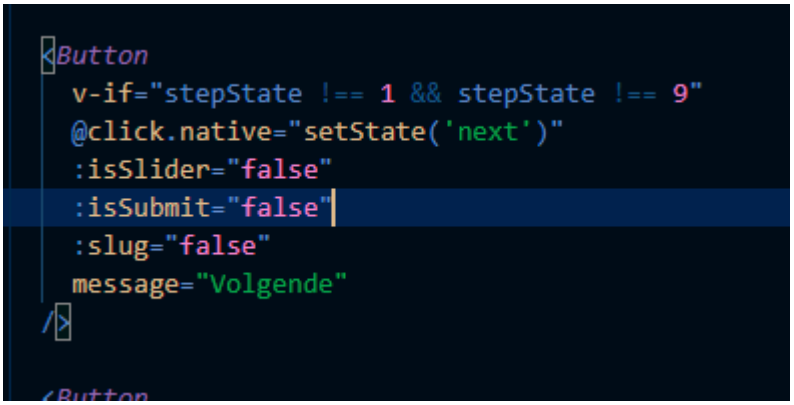
Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

Hier ziet u hoe de applicatie geïnstalleerd kan worden op een mobile device

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

Bug fixing

Deze week stond ook in het teken van het bug fixen, terwijl Ralf verder bezig is met de use case van het chatten, had ik het gevoel dat het beter zou zijn als ik de huidige bugs zou kunnen tackelen of wellicht sommige processen beter kan oplossen dan hoe ze er momenteel er voor staan enkele voorbeelden:



Zo kwam ik er bijvoorbeeld achter dat ik @click.native kan gebruiken met een geïmporteerde component om vervolgens een andere methode in hetzelfde bestand kan aanroepen.

Zo was er ook een image die Eva ons afgeleverd had die in de hoogte was genomen, ipv in de breedte, in het begin had ik dit niet door totdat ik toevallig ging kijken of alles klopte, ik heb vervolgens meteen contact opgenomen met Eva om te vragen of zij wellicht ook nog andere images beschikbaar had voor dit thema. Deze image kreeg ik binnen enkele minuten meteen opgestuurd.

pair coding

Omdat Ralf zo druk was met zijn use case, kwam hij er soms niet altijd lekker uit of wilde hij simpelweg even overleggen met hoe we het zouden doen, ik heb vervolgens een paar uur lang met Ralf in een call gezeten om de problemen te kunnen tackelen, met uiteindelijk een mooi resultaat: namelijk dat het werkte :)

Feedback Design review 2 - Koop

We hebben in deze sessie besproken van wat de error state zou kunnen zijn van de chats. Ralf gaat hier verder mee aan de slag en zal dit verwerken. Verder hebben we het gehad over wat onze meerwaarde als frontend developers is in dit project. Het is goed om je dat als frontender altijd af te vragen. Wat zijn jouw talenten en wat maakt jou als frontender zo uniek. Dit was best een inspirerende talk waar wij echt iets aan hebben gehad. Vervolgens hebben we het gehad over de micro interacties bij de chat bijvoorbeeld op wanneer de chat verstuurd is maar nog niet is aangekomen. Daar kan al een hele mooi micro interactie bij die precies vertelt over de staat van het chatbericht. Hierover kunnen Rowin en Ralf gaan sparren om zo'n micro interactie volgende week te implementeren. Echter is het wel de laatste week, dus of daar tijd voor is moeten we eerst nog zien omdat we andere prioriteiten op de planning hebben staan.
Ook hebben we het over het filteren gehad van de buddies pagina. Filteren is een cruciale stap die we als frontenders kunnen zetten om de UX nog meer te verbeteren en dus die meerwaarde te bieden die we als frontender kunnen realiseren. Rowin zal hier verder mee aan de slag gaan.

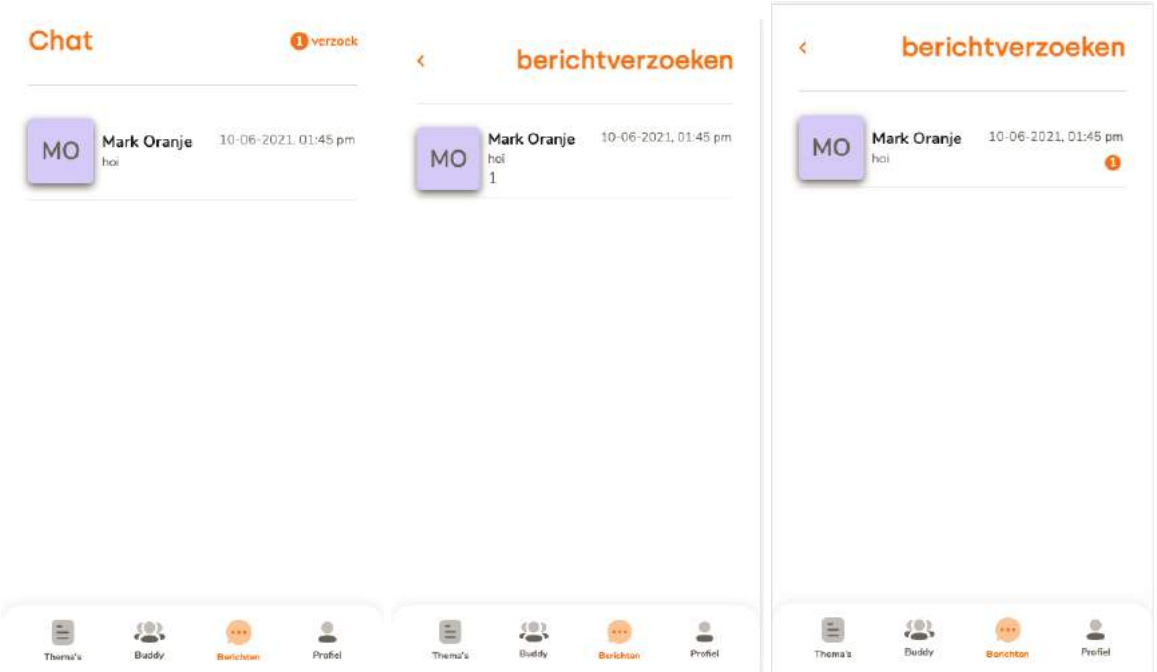
Chats | Checken of chat bestaat

Eerst heb ik de logica geschreven om te kijken of een chat bestaat. Als de chat bestaat, return dan dat chatID zodat de user daar naartoe kan gaan. Als de chat niet bestaat, maak dan een nieuwe chat aan. Echter moest ik de chatfunctie heel vaak testen, want het liep zo vaak stuk op het bijvoorbeeld accepteren van de chats. De functie die checkt of er een chat bestaat staat hieronder:

```
async function checkChatExist(userID, partID) {
  const chats = await getChatsById(userID);

  if (chats.length) {
    // Global variable - existing chatID
    let existingChatID;

    // If user has chats, take look at participants key in each object
    for (let i in chats) {
      let chatParticipants = chats[i].participants;

      let check = chatParticipants.includes(partID);

      if (check) {
        // Sets global variable to existing Chat ID
        existingChatID = chats[i]._id;
      } else {
        existingChatID = false;
      }
    }
    return existingChatID;
  }
  return false;
}
```

Chats | Verzoeken indienen

Ik heb bij het stylen van de chatverzoeken goed gelet op dat 1'tje wat bovenaan hoort te staan. Wanneer er dus 1 chatverzoek is, geef hij dat aan met het cijfer 1 en het woord "verzoek" erachter. Op het moment dat dit meer dan 1 is, staat er "verzoeken". De styling progressie van de chatverzoeken ziet er zo uit (zie hieronder):



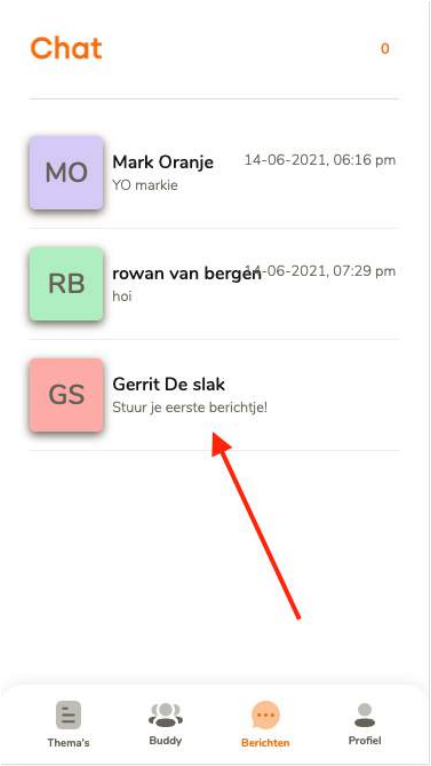
Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

UI Stack | Partial State | Chat

Chat detail

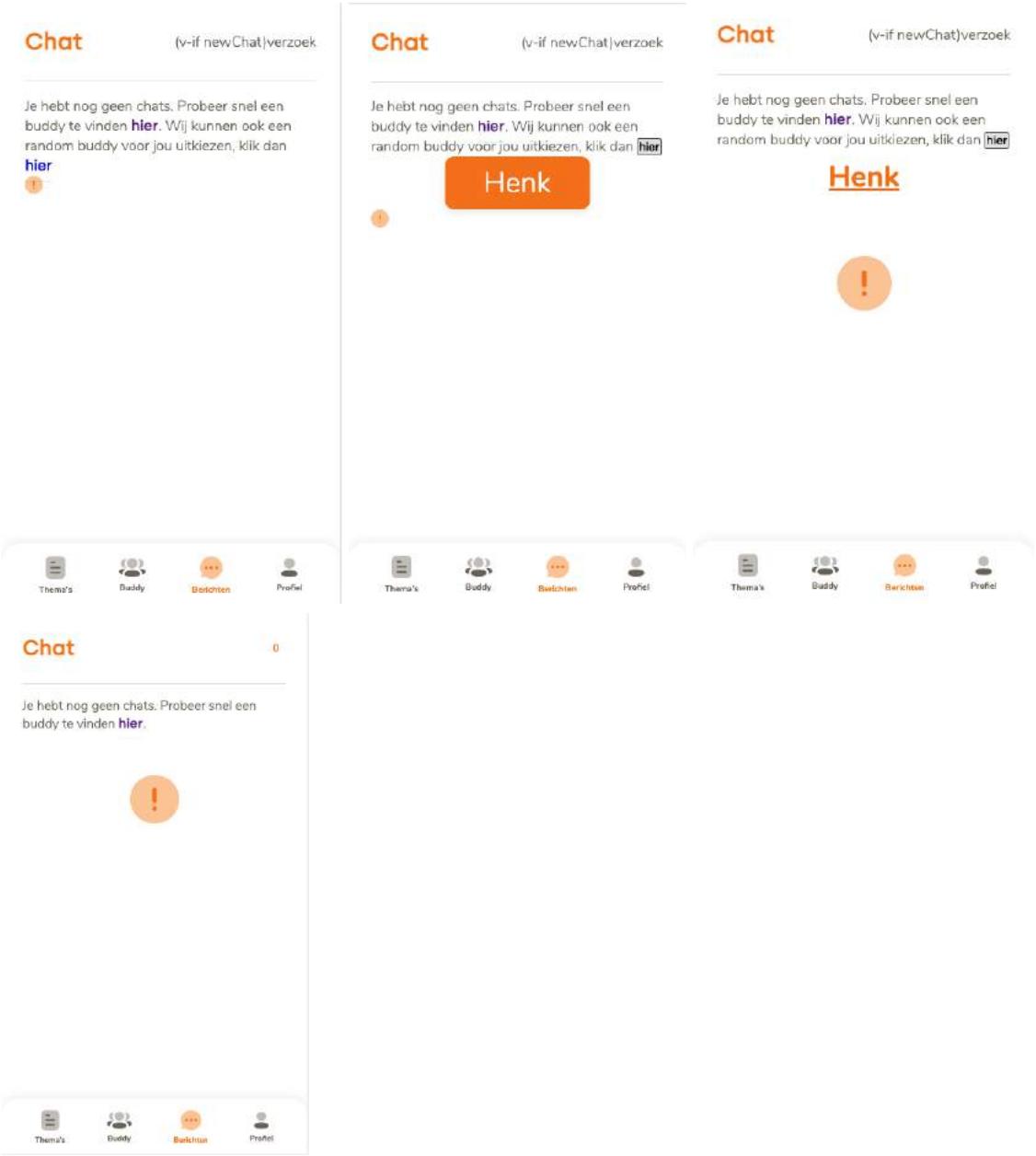
Op het moment dat een chat nog een chatverzoek is, verandert de UI in een pagina die aan de gebruiker vraagt om het chatverzoek te accepteren of te weigeren. Zo ziet dit scherm eruit:

Op het moment dat een gebruiker een chatverzoek indient bij een andere gebruiker en plots weer uitlogt, is er geen bericht gestuurd bij het chatverzoek. Dit wordt dan aangepast in de chat overview pagina met “Stuur je eerste bericht!”. Zie hieronder :



Chat overview

Empty/partial state maken voor chat overview
Wanneer er geen data aanwezig is van de chats, dan weergeeft hij dit in het scherm. Dit is een partial state, dus het zet de gebruiker aan tot actie nemen (om toch een paar chats op te bouwen). Echter vond Eva dit aan het eind deze week tijdens de feedback sessie op vrijdag niet goed, want ze vindt dat je een serieuze band moet opbouwen met een persoon en dan is het raar als je wel nonchalant een random buddy kan uitkiezen. Bovendien vond ze het wel goed dat ik dit scherm gemaakt heb dus alleen de referentie naar de buddies pagina is al erg sterk. Om die reden heb ik de random functie weggehaald. Ziet er nu zo (als afbeelding 4) uit.



Refactor | Code comments

Ik heb alle functies in de code van het hele project voorzien van code comments. Wij vinden dit erg belangrijk, want zo weten wij allebei wat de functies doen en andere developers weten dan ook hoe de functies werken. Ik heb gebruik gemaakt van de standaard JSDocs. Dit is een bepaalde, in mijn ogen, overzichtelijke manier van code

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

comments schrijven. Je ziet duidelijk het verschil tussen i) beschrijving, ii) parameters en iii) return waardes. Hieronder is een voorbeeld hoe ik dat gedaan heb:

```
/**
 * Checks if chat already exists, otherwise create chat environment
 *
 * @param {String} userID - ID of the current user
 * @param {String} participantID - ID of the participant
 *
 * @return {String} newChatID - ID of the new chatroom
 *
 */
```

Feedback 11 juni van Eva

Vraag aan Eva:

- Is de typbalk in de chat het laatste icon een mic of pijltje? Daar maak je verschil in in je design, hoe zit dat? *Nou het zit zo, het werkt eigenlijk net als bij WhatsApp. Als je nog niet de typbalk geselecteerd hebt, zie je een mic staan. Daar kan je dus voice meesturen. Maar op het moment dat je dus wel de typbalk geselecteerd hebt en dus aan het typen bent, veranderd die mic icoon in een verzend icoon.*

Wat laten we zien:

- Chat feature: realtime, chatverzoeken.
  → *Ik vind het heel gaaf hoe jullie dit gebouwd hebben en dat het ook nog werkt! En ook nog realtime. Super vet en ziet er leuk en strak uit. De chatverzoeken zijn ook 1 op 1 als in mijn ontworpen design. Cool!*

- Error state | chat overview
  → *Ik vind dit super nice, alleen zou ik de random functionaliteit wel weghalen, want dit is een serieus onderwerp en de gebruikers hebben echt behoefte aan een gesprek met iemand waarbij ze zich comfortabel voelen. Bij een random gebruiker kan dat gewoon verkeerd uitpakken. Mijn doel in de app is ook dat gebruikers op zoek gaan naar de juiste buddy.*

- Progressive Web App
  → Eva vond dit een geweldige feature en ze vond het er ook echt gaaf uit zien. Wel vind ze het favicon wat wij designed hebben iets minder, omdat we er een 3e font nu bij hebben, maar wij vonden het juist veel mooier staan dan het normale font wat zij gebruikt (Nunito), want dit lijkt volgens ons veel meer op een app die je echt kan installeren. Hieronder zie je de 2 favicons van hoe Eva het zou willen hebben (1e) en hoe wij het zouden willen hebben (2e):

Week 5

Taken (streven) deze week

Rowin	Ralf
Vue Tour onboarding	Microinteractions
Styling	Readme: uitleg functies/methods/folderstructure
	Transitie tussen de registratiestappen
	Styling
	Bugs + issues oplossen

Ook gaan we deze week vooral kijken naar de openstaande issues. We willen die bug/refactor dingen aan gaan pakken.

Styling | Register | stap 6

Ik heb de styling van stap 6 van het registreren verbeterd. Omdat deze week in het teken staat van alle puntjes op de ‘i’ zetten, begon ik hiermee. In de details wijkte dit een beetje af van het design. Daarom heb ik ervoor gekozen om nog een goede iteratieslag erop los te laten. Hieronder zie je het proces van hoe het was en hoe het nu is.

Styling | Register | Button alignen

Ook op de button component die we inladen bij het registratieproces heb ik even wat styling op losgelaten want hij stond links, terwijl in het design staat hij rechts. Ik heb hier gebruik gemaakt van de if else methodes van Vue in de template zelf en op basis daarvan de buttons gestyled qua positionering. Hieronder laat ik zien hoe ik dat gedaan heb: (link)

```
<div class="footerBegin" v-if="stepState === 1">
  <router-link to="/login" class="login">Ik heb al een account</router-link>

  <Button
    v-if="stepState !== 1 && stepState !== 9"
    @click.native="setState('next')"
    message="volgende"
  >
    ></Button
  >

  <Button
    v-else
    @click.native="setState('next')"
    message="starten"
    :isSubmit="true"
  ></Button>
</div>

<div class="footer" v-else-if="stepState !== 1 && stepState !== 9">
  <Button @click.native="setState('next')" message="volgende"> ></Button>
</div>

<div class="footer" v-else>
  <Button @click.native="setState('next')" message="starten" :isSubmit="true"></Button>
</div>
```

Hier zie je het visuele effect van deze aanpassing:

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

Bug | Na registratie naar buddies

Na het registreren wordt je automatisch ingelogd. Op het moment dat ik dan naar de buddies pagina ging, kreeg ik een error dat te maken had met ObjectID. Eerst dacht ik dat dit te maken had met Mongoose, maar achteraf scheen het een fout te zijn veel eerder in het proces.

```
async getAllUsers() {
  let currentUserID = this.$store.state.user._id;
  let url = `${window.location.origin}/api/users/${currentUserID}`;

  axios
    .get(url)
    .then((response) => {
      // Iterate over each obj and put in array
      let arrayUsers = this.users;
      arrayUsers.push(response.data);
    })
    .catch((err) => {
      console.log("err");
      // this.errors.push("Er zijn helaas geen users gevonden");
    });
},
```

Het scheen natuurlijk dat er bij de buddies pagina de userID die in de Vuex store zit wordt meegestuurd. Maar tijdens het registreren worden alleen de input waardes in de Vuex gezet. Er komt dus geen id in te staan, dus je kan dan nooit als je vanuit het registreren naar de buddies pagina gaat het ID meegeven, omdat dat ID undefined is. Ik heb er daarom voor gezorgd dat wanneer de user een account heeft aangemaakt en in de db wordt gezet, dat het het hele user object returnt naar de client en in de Vuex wordt gezet. Hieronder zie je hoe ik de user aanmaak in de database en het user object return naar de client:

```
async function handleRegister(req, res) {
  // Salt the plain password
  const passwordHash = bcrypt.hashSync(req.body.password, saltRounds);

  const userObject = {
    firstName: req.body.firstName,
    surName: req.body.surName,
    emailAddress: req.body.emailAddress,
    password: passwordHash,
    birthDate: req.body.birthDate,
    town: req.body.town,
    gender: req.body.gender,
    typeIllness: req.body.typeIllness,
    profileAvatar: req.body.profileAvatar,
    about: req.body.about,
  };

  console.log('User registered - data: ', userObject);
  let newUser = await createUser(userObject);

  // Set global userID to userID
  let globalUserID = newUser._id.toString();
  anotherGlobalUserID  = globalUserID
  setGlobal(globalUserID);

  return res.json(newUser)
}
```

Nu ik wil het ID kon meesturen met de axios post vanuit de client, doe ik dat niet meer. Want ik houd het userID bij op de server, dus dan hoef ik dit niet meer clientside te regelen. Bovendien is de axios post URL nu een stuk netter en “semantischer” Waarom semantischer? Omdat ik de URL eigenlijk met de currentUserID gebruikte als parameter en dat is niet helemaal de bedoeling van de axios URL. Nu is het zo:

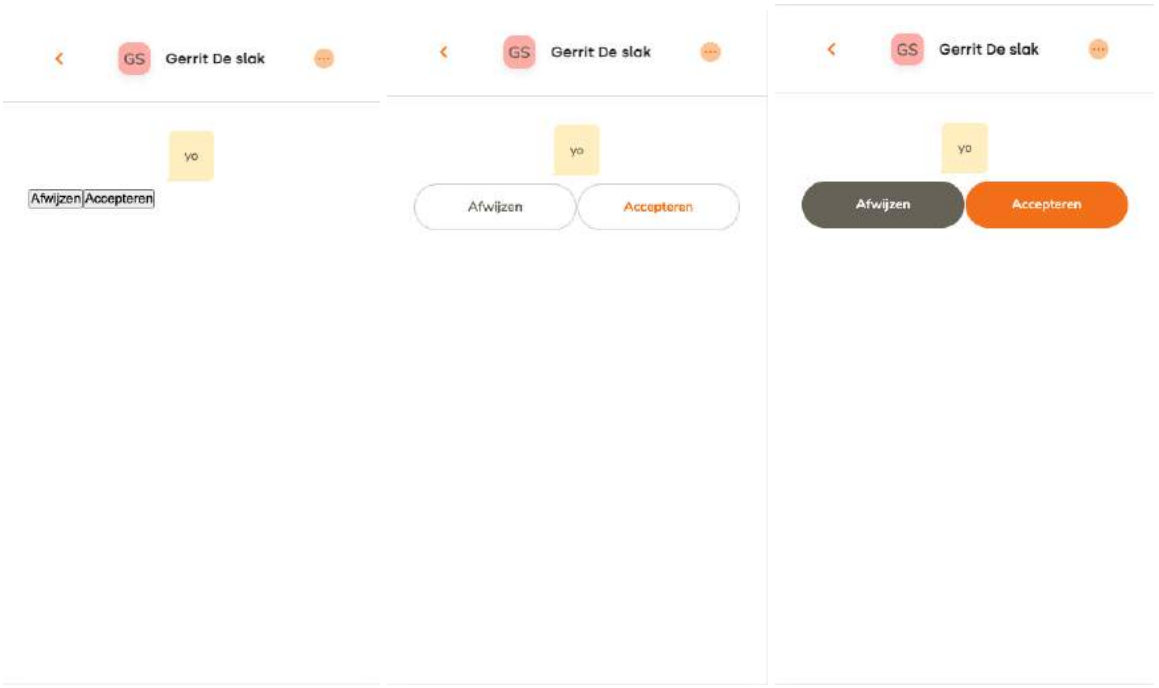
```
  async getAllUsers() {
    let url = `${window.location.origin}/api/users`;

    axios
      .get(url)
      .then((response) => {
        // Iterate over each obj and put in array
        let arrayUsers = this.users;
        arrayUsers.push(response.data);
      })
      .catch((err) => {
        // this.errors.push("Er zijn helaas geen users gevonden");
      });
  },
```

Readme

Ik heb de readme voor groot gedeelte aangevuld. Hierin staan nu de screenshots van hoe het UI eruit ziet, de features beschreven en een opzet gemaakt voor hoe ik het PWA en andere technieken ga uitleggen.

Styling | Chatverzoek buttons

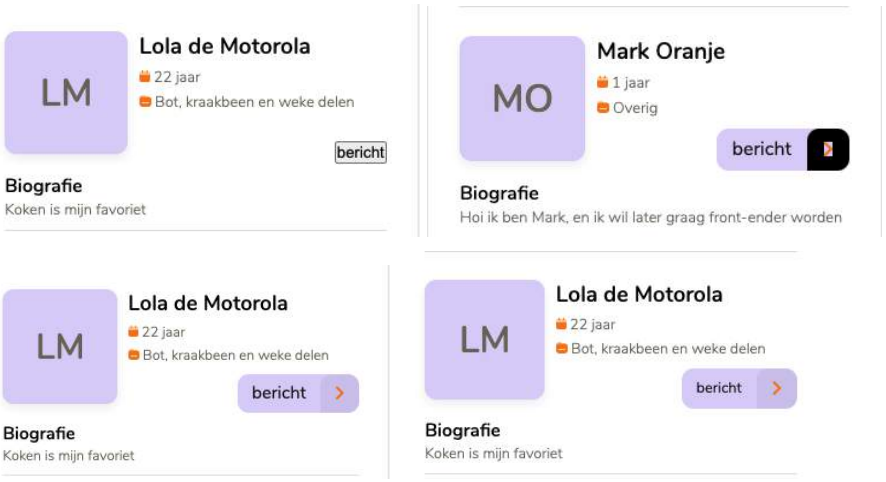
Ik heb de styling van de chatverzoek buttons aangepast. Echter staat in het design dat er natuurlijk geen rekening is gehouden met pseudo classes als wanneer de gebruiker eroverheen hovert, als button actief is of de focus heeft. Hier heb ik juist wel aan gedacht en de huisstijl heb ik meegenomen in mijn ontwerp. Hieronder zie je eerst hoe de buttons er als eerst uit zagen (niet vormgegeven). Daarnaast in het midden zie je de vormgegeven buttons en daarnaast staan de buttons met de pseudo classes hover/active/focus met alle 3 dezelfde styling per button:



Styling | Buddy detail bericht button

Als alle buttons gestyled zijn kan er niet een button “ongestyled” overblijven. Daarom heb ik ervoor gekozen om de button ook te stylen in de huisstijl. Wat deze button zo uniek maakt, is dat het de kleur overneemt van de profile avatar van de gebruiker. Dus

wanneer de gebruiker paars als profile avatar heeft, zal deze button ook paars worden. Dit doe ik weer met de dynamische class die ik op de button render. Hieronder zie je het proces van de button (van links naar rechts):



Chat overview | Volgorde van meest recente chats

Ik heb een feature gemaakt die de volgorde aanpast van de chats op de overview pagina van meest recent (bovenaan) naar minst recent (onderaan). Dit is de meerwaarde die ik als frontender kan geven aan de chat feature. Waarom? Eerst had ik de chat overview met alle chats die op volgorde staan zoals ze in de database staan: die het eerst zijn gemaakt bovenaan en dan de andere onderaan. Voor een gebruiker is dit niet erg fijn, want hij/zij moet dan steeds zoeken wanneer hij/zij het laatst heeft gechat met iemand. De gebruiker verwacht namelijk, net zoals bij WhatsApp, dat de chat waar het laatst in is gesproken bovenaan komt te staan. Toen ik hierover nadacht vond ik het eigenlijk wel een cruciale stap om te nemen om die UX te versterken. Dat is echt een meerwaarde die ik als frontender kan bieden bij de chat functionaliteit

Om dit te doen, moet ik de volgorde van de chats indelen op tijd. Dus de tijd van het laatst gestuurde bericht, daarop moet gesorteerd worden. De timestamp van het bericht moet dus eerst goed zijn. Eerst stond de tijd al geformatteerd in de database, maar dat bleek toch geen groot succes te zijn, want wanneer het op de client was aangekomen, moest ik het weer omvormen naar normale tijdsaanduiding en dat dan weer naar secondes omrekenen. Dat was erg slordig. Daarom koos ik ervoor om de timestamp van ieder chatbericht aan te geven in secondes. Dat is super nauwkeurig en dan weet je precies op welke datum en tijdstip het bericht verstuurd is. Op de client kijk ik naar de timestamps en op die waardes ga ik ze sorteren. Echter heb ik deze functie ***3 keer*** opnieuw moeten schrijven, omdat ik steeds vastliep of de app ging stuk. Toen de

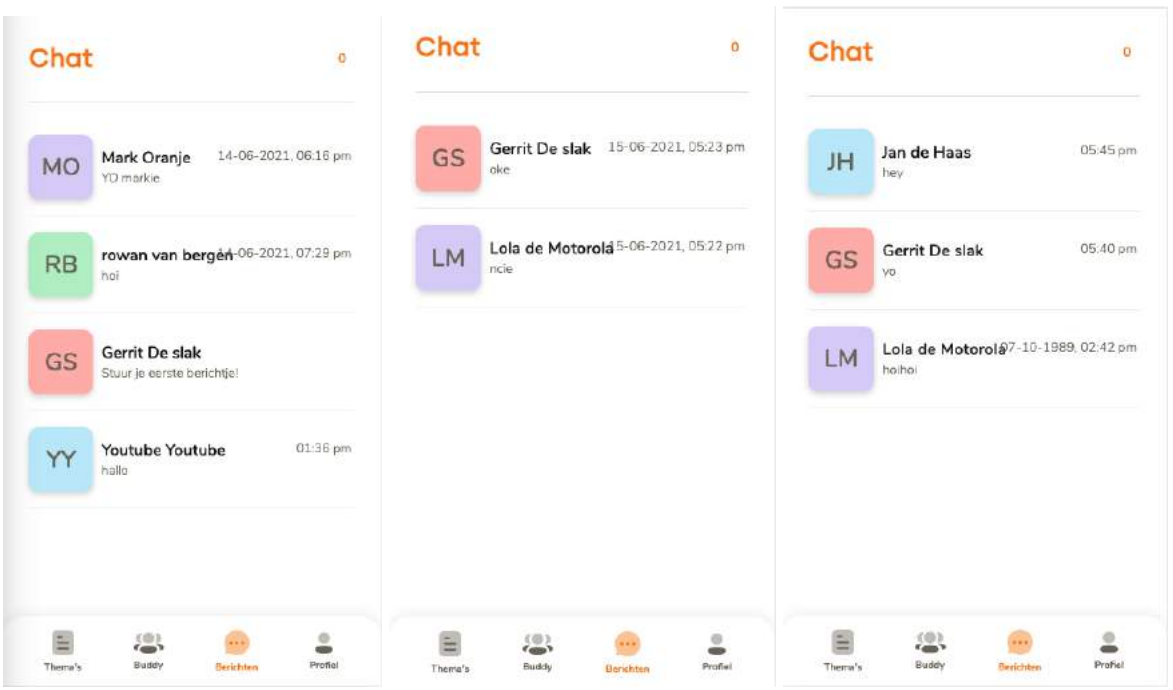
laatste keer ging ik eerst alle stappen uitschrijven en dan elke stap coderen. Ik kreeg bij iedere keer dat ik het opnieuw schreef een andere blik erop.

```
// 6. Sort the array
allChats.sort(function (a, b) {
  return b.timeSort.message.time - a.timeSort.message.time;
});
```

Nu was ik eigenlijk klaar. Maar ik vond het eigenlijk wel toepasselijk om nog een meerwaarde te bieden aan de gebruiker; de gebruiker de tijd van elk laatst gestuurd chatbericht te geven in normale tijd in plaats van secondes. En om de gebruiker te vertellen dat als er een bericht is gestuurd op de dag dat hij/zij kijkt op de chat overview, dat er dan alleen een tijdsaanduiding staat, anders staat er ook de datum bij. Dit doe ik door middel van een check op de client, zie hieronder:

```
// Check if time is the same as today, then remove date
convertTime(timestamp) {
  let timeStampMsg = timestamp * 1000;

  let todayHours = new Date().setHours(0, 0, 0, 0);
  let chatTimeHours = new Date(timeStampMsg).setHours(0, 0, 0, 0);

  if (todayHours === chatTimeHours) {
    // Message is from today
    // format to correct
    let formattedDate = moment(timeStampMsg).format("hh:mm a");
    return formattedDate;
  } else {
    // Message is from not today
    // format to correct
    let formattedDate = moment(timeStampMsg).format("DD-MM-YYYY, hh:mm a");
    return formattedDate;
  }
},
```

Hoe chat overview er door de iteraties uitziet:



Echter zit er nog wel een bug in die een beetje afhangt van de use case. Op het moment dat je iemand voor het eerst wil chatten, dan kom je op de detailpagina van de nieuwe gemaakte chat. Als je dan ervoor kiest om weg te gaan uit dit scherm, is er geen message achterlaten en is de messages key in het chat object in de database leeg. Op het moment dat de andere gebruiker dan op de chat overview komt, gaat de applicatie stuk. Dit is echter nog een nare bug, maar vanwege tijdgebrek en prioriteiten, hebben we geen tijd om dat nu nog te fixen. Er zijn wel een aantal manieren hiervoor om het op te lossen. Zoals bijvoorbeeld als de gebruiker bij het aanmaken van de chat geen bericht stuurt, dat de database zelf een bericht erin zet.

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

Vue ontour boarding

Een feature die we nog niet hadden was die onboarding, zodra je op de homepage komt word je als het ware begeleid door de app heen.

De gebruiker wordt bij de eerste keer inloggen dus begeleid, hierboven ziet u een afbeelding met de eerste state namelijk thema, de gebruiker kan m.b.v de knoppen verder gaan of de tour overslaan.

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

Als de gebruiker verder in het proces is, dan kan hij/zij ook terug naar de vorige stap

Error prevention /feedback

De gebruiker kan nu tijdens het registeren ook meteen zien of er een input klopt of niet, elke stap wordt gecontroleerd voordat de gebruik bij de eindstap is.

Hieronder heb ik de feedback van vorige week van Eva verwerkt, ik had eerst in gedachten om de resultaten te weergeven in de header met: ik wil contact met 5 mensen. Ik heb er nu dus een losse paragraaf ingezet zodat deze onderaan verschijnt zodra er een filter optie actief is.

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

Reflecties

Aan het eind van het project reflecteer je systematisch op je werk en het proces. Aan de hand van de vak-rubrics schrijf je welke vakken wel of niet aan bod zijn gekomen en waarom. Zo krijg je een goed beeld van je eigen niveau, mogelijke aandachtspunten in techniek, interactie en/of aspecten van het design-proces waar je je nog op kan verbeteren.

Reflectie | Rowin

Nadat de 5 korte weken alweer voorbij zijn gevlogen, is het ook tijd om te reflecteren op het project.

Toen we het project kozen dacht ik van tevoren dat het niet zoveel werk zou zijn om de basisstructuur te kunnen realiseren, dit bleek snel echter al een misverstand te zijn, omdat we er ook voor gekozen hebben om een tech stack te gebruiken die we van tevoren nog niet eerder gebruikt hebben. Namelijk express als back-end en vue als ons front-end framework, hierdoor komen er dus soms wat implementatie problemen, denk hierbij aan het online krijgen via heroku.

Na soms wat intense en vermoeiende weken vind ik dat we best trots mogen zijn op wat we eindelijk neergezet hebben.

Wat ik voornamelijk heb geleerd tijdens deze weken is eigenlijk best wel veel om op te noemen denk hierbij aan:

- het bijleren van het framework Vue
- Deployment via Heroku met node en vue
- Node als back-end gebruiken.

Ook ben ik er achter gekomen dat alle vakken van de minor ons geholpen hebben tijdens dit project, omdat er gewoon heel veel dingen inzitten die we geleerd hebben vanuit de minor, denk hierbij aan socket-io, pwa.

Ik vond de samenwerking met Ralf prettig en leerzaam, zo hebben we elke dag elkaar wel minstens 1x gebeld in discord en soms ook wel vaker dan 3x op een dag om de bugs/progressie te bespreken met elkaar. Deze meetings hielpen ons heel erg tijdens het process, Kortom heb ik veel geleerd van deze 5 weken.

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

Reflectie | Ralf

Daar ben ik dan, vijf weken voltijd programmeren, sprints indelen, overzicht behouden, testen, ***veel*** bugs oplossen en elke ochtend een stand-up met Rowin. Wat was dit een leerzaam en leuk project waar ik erg trots op mag zijn!

Aan het begin van dit project vond ik het leuk, maar ik dacht dat het een beetje dezelfde structuur zou worden als van Project Tech (blok Tech). Maar dat bleek dus niet zo te zijn. Puur omdat we gebruik hebben gemaakt van een front end framework en een backend was dit super nieuw voor mij. Daarbij heb ik ook super veel geleerd van dit project, zoals:

- Nieuw frontend framework (Vue) geleerd in korte tijd
- Backend en frontend laten runnen op 1 commando
- Rowin's codeertechnieken
- Leren zoeken op internet naar bugs en technieken
- SASS
- Veel beter begrip van client en server
- Werking Axios POST en GET

Over de minor-vakken in dit project gesproken zijn dat wel een aantal geweest. Zo is CSSttR aan bod geweest met het gebruiken van pseudo selectors in de styling. Dit was namelijk echt een aspect dat ik geleerd heb tijdens dat vak en wat ik hier super mooi kon toepassen. Ook het DRY schrijven van de styling code is iets wat ik heb geïmplementeerd. Van WAFS heb ik geleerd de plain JS functies te schrijven, vooral op de frontend. Ook heb ik geleerd bij WAFS altijd iets te returnen uit een functie om een waterfall effect te voorkomen. Verder heb ik gebruik gemaakt van sockets (socketIO), wat ik geleerd heb bij het vak RTW. Dit is een realtime functionaliteit die ik geïmplementeerd heb. Ik wilde deze functionaliteit sowieso implementeren, juist omdat ik die techniek voor het eerst had aangeleerd bij RTW en ik er eigenlijk wel erg enthousiast over was geraakt. Bovendien zit de manifest en serviceworker in het project, wat we bij PWA geleerd hebben, erin. Echter heeft Rowin hier voor gezorgd.

De samenwerking met Rowin was erg aangenaam, want hij communiceerde super fijn en altijd snel. Ook werkte Rowin gestructureerd en overzichtelijk, wat ik fijn vind omdat ik zelf ook zo werk. Rowin en ik hadden dagelijks minimaal 2 meetings dat soms uit liep tot 4 meetings als er weer plots een bug boven water verscheen. Ook merkte ik dat we altijd voor elkaar klaarstonden en hulp boden waar het nodig was.

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development

Ik ben erachter gekomen dat ik sommige dingen te ‘makkelijk’ had ingeschat, maar die juist veel tijd in namen. Dat is een mooie leerles die in de praktijk altijd wel terecht komt. Dingen lopen nooit helemaal zoals je ze gepland hebt. Daarom keken we uiteindelijk aan het begin van elke week wat onze sprint die week zou opleveren op vrijdag en of dat realiseerbaar was.

Bovendien is er een leerles waar ik persoonlijk heel erg ben achtergekomen. Dit is dat ik met het coderen van een nieuwe feature gelijk al tijdens het coderen het wil optimaliseren. En hiermee kwam ik dus steeds mijzelf tegen, want als ik dit deed werd de code wel is onoverzichtelijk, te lastig en ging zelfs de app stuk. Daarom heb ik het principe **Build First, Optimise Later** geleerd. Hier heb ik dus geleerd om eerst lekker de functionaliteit uit te schrijven en dat die dan werkt. En dat ik daarna pas die functionaliteit ga optimaliseren en overzichtelijk maken.

Link prototype: https://bloom-hva.herokuapp.com/

Link repository: https://github.com/RowinRuizendaal/Bloom

Link Design Rationale:
https://github.com/RowinRuizendaal/Bloom/blob/master/README.md

Ralf Zonneveld
Rowin Ruizendaal
Minor Web Design & Development