**THE RANKING GAME**

**Applicatie Opzet**

Er is een flow om een spel op te zetten (door de quiz master), om een spel in te vullen (door een deelnemer), om een spel te spelen (door een deelnemer) en om het spel te begeleiden (door de quiz master).

“Vul de namen en e-mail adressen in van de deelnemers van het spel. De naam die ze nu krijgen, wordt ook gebruikt om de ranking vragen in te vullen, en om de quiz te spelen. Het e-mail adres wordt gebruikt om de deelnemers uit te nodigen, en toegang te verlenen tot de quiz.”

Het zou optioneel kunnen zijn om in te loggen (ik heb al een naam en wachtwoord combinatie), maar als het spel een unieke code heeft, is deze code misschien ook voldoende om het spel af te schermen. Misschien eerst een opzet maken met login, ook omdat dat de meest leerzame opzet is…?

Home *startpagina, hier bepaal je of je een spel wilt opzetten, of wilt invullen, of wilt spelen*

Spel opzetten *deze elementen moeten in een ‘stepper’ worden gepresenteerd*

Deelnemers toevoegen *Wie is quiz master, wie is deelnemer (rol is belangrijk)*

Vragen toevoegen

Configuratie pagina…? *Deadline? Op jezelf stemmen? Inloggen?*

Quiz open zetten “*E-mails zijn verstuurd, veel succes!”*

Spel invullen *een groot formulier met alle vragen (wordt opgeslagen op veld verlaten)*

Spel spelen

home

setup

add-participants

add-questions

configuration

completed

rank

play

Hoe ziet de backend er uit, en welke endpoints bestaan daar?

Ik heb een game-controller, onder “/game”, waar ik een “/create” en een “/get/:id” en een “/get-all” kan aanroepen?

TODO

In Setup Component:

* De overgang tussen stappen mooier maken met een animatie (alleen bij openen een collapse van het component maken?)
* De formulieren afmaken
* Navigatie mogelijk maken afhankelijk van de validiteit van de formulieren
* Een edit-modus en view-modus van de formulieren maken (en een edit-knop toevoegen in inactive mode?)

Algemeen:

* Aparte modules maken voor Setup, Rank, Play en Shared
* Prefix veranderen van app- naar ranking-

De TOD’s worden vanaf nu bijgehouden in GitHub, op het Project bord. Leuke bijkomstigheid aan issues bijhouden via GitHub, is dat deze terug te zien zijn in mijn ‘contributions’ overzicht, waardoor het lijkt alsof ik aanzienlijk actiever ben op GitHub.

**CORS**

Ik loop tegen problemen met CORS errors; de FE draait op localhost:4200, en de BE draait op localhost:3000, en dat levert CORS issues op. De error is te zien in de console van de browser tools, en niet in de network calls.

Ik dacht het probleem om te lossen met een NestJS instelling in main.ts:

**const** app = **await** NestFactory.create(AppModule, { **cors**: **true** });

Maar dit is niet echt een oplossing die ik graag in de applicatie wil toepassen, want de CORS regel bestaat voor een reden, en om deze compleet te negeren voelt niet goed.

Na het hoofdstuk van NestJS over Middleware doorgenomen te hebben, dacht ik een betere oplossing gevonden te hebben in de vorm van een CORS Middleware class, zoals in deze blog wordt beschreven:

<https://www.thecodecampus.de/blog/creating-cors-middleware-nest-js/>

Dit zou een mooie oplossing geweest zijn, als Angular niet zelf met een alternatieve oplossing kwam, en dat is om gebruik te maken van het proxy.conf.json bestand. Hierin geef ik aan dat alle “/api” calls moeten worden vervangen door “http://localhost:3000”, en daarmee treedt de CORS error niet meer op. Dus uiteindelijk is het probleem opgelost in de FE, en niet in de BE.

**2020-10-16**

Ik ben bezig met het opslaan van data naar de BE, en loop tegen verschillende vragen aan. Wil ik bij iedere stap een post call naar de BE doen? En is vanuit de FE iedere child-component verantwoordelijk voor deze call? Mijn eerste idee is dat dit wel het geval is, maar dan heb ik bijvoorbeeld voor de game-setup wel erg veel calls te maken: 3 stappen, waarbij eigenlijk voor iedere stap zowel een post (opslaan van data) als een get nodig is (bijvoorbeeld wanneer de quiz-master terugkomt om de quiz bij te werken). Met drie stappen en 2 calls per stap, zijn dat al 6 verschillende endpoints die ik moet aanroepen. Dat is zowel op de FE als de BE een hoop controllers en code…

Biedt GraphQL hier een gepaste oplossing, en zo ja, hoe zou deze opzet er dan uitzien? Ik geloof dat ik nog altijd een verschil moet maken tussen een post en een get, ofwel een mutation en een query. Alleen mutations, met geen data wanneer niets is ingevuld, zou volgens mij niet werken (ik ben namelijk bang dat een leeg object voor een eerste mutation, eventueel bestaande data zou kunnen overschrijven waardoor deze data verloren gaat; als dit niet het geval is, en een lege mutation verandert niets, zou dit een krachtige oplossing kunnen zijn). Maar zelfs als de mutation en query apart gedefinieerd moeten worden, is GraphQL voordeliger dan de traditionele aanpak. Omdat ik namelijk met de query kan bepalen welke gegevens ik wil ophalen (bijvoorbeeld alleen questions, of alleen participants), heb ik in totaal nog maar 2 endpoints nodig. Alle sub-onderdelen worden namelijk in hetzelfde object (ofwel endpoint) opgeslagen.

**2020-11-07**

Ik bedenk me net dat ik veel issues heb met het navigeren (denk aan opslaan van data, valideren van formulieren) die veroorzaakt worden door de ‘edit’ knop, en die navigatie onvoorspelbaar maakt. Dus waarom haal ik die knop er niet gewoon uit? Dan kan navigeren alleen van stap naar stap via de ‘previous’ en ‘next’ knoppen, en hoef ik alleen op ‘next’ een formulier te valideren en de data naar de BE op te sturen. Dan kan ik op de ‘next’ knop ook een loading icon maken (een spinner) die aangeeft dat data daadwerkelijk wordt opgestuurd. Door de ‘edit’ knoppen weg te laten, neem ik enige vrijheid van de gebruikers weg (al is het aantal stappen beperkt, dus de impact is niet heel erg groot) maar is de validatie wel veel robuuster.

*Edit:* eigenlijk hoeft de ‘edit’ knop niet weg, als hij hetzelfde werkt als de ‘previous’ knop, zolang deze niets valideren of naar de BE sturen… Dan is alleen de ‘next’ belangrijk voor validatie, en opsturen van data naar de BE. En dan moet het met de ‘edit’ knop niet mogelijk zijn om naar een volgende stap te gaan, enkel naar eerder voltooide stappen. Of kan ik controleren wanneer ‘edit’ een stap terug gaat (wat altijd mag), of een stap vooruit gaat (wat alleen mag na validatie en opslaan van data)?

Daarnaast realiseer ik me dat in de configuratie de optie hoort of de quizmaster wel of niet meespeelt. Dan moet ik de BE modellen opnieuw aanpassen (quizmaster wordt gewoon een speler en heeft niet een property “participates”).

**2021-03-05**

De add-players en add-questions pagina’s moeten worden opgesplitst; de html wordt te groot, en daarom moeten er aparte componenten komen voor view-players en edit-players. Dan kan ik wellicht de edit-players gebruiken om een BE call te maken op onInit, want het lijkt er nu op dat alle componenten in de stepper (ook degenen die niet actief zijn) in de onInit komen en daarmee potentieel al data van de BE ophalen.

Overigens moeten de componenten nog absoluut modules worden. Absurd dat setup nog geen module is…

**2022-03-13**

Opzet voor de Express BE:

Server

Controllers

Setup

Rank

Play

Middleware

Auth

Models

node\_modules

Routes

Setup

Rank

Play

Authenticatie zit er nu nog niet in, maar wordt zeker belangrijk als je gaat spelen (Rank en Play), want dan wil je niet dat onbevoegden aan het spel deelnemen. Ook bij Setup is authenticatie belangrijk, omdat niet zomaar iedereen de opzet van het spel mag veranderen.

De routes zijn nu mooi opgedeeld (alles onder /api, en dan onder /game, waarbij de gameId altijd beschikbaar lijkt te zijn). Wat nog moet gebeuren:

* De controllers moeten worden toegevoegd die de logica oppakken wat nu in de routes nog gebeurt

Bestaande endpoints:

* GET: api/game (*niet nodig?*)
* GET: api/game/:gameId (*bestaat in BE*)
* POST: api/game/create
* GET: api/game/:gameId/setup (*bestaat niet in FE*)
* GET: api/game/:gameId/setup/players (*bestaat in BE*)
* POST: api/game/:gameId/setup/players
* GET: api/game/:gameId/setup/questions (*bestaat in BE*)
* POST: api/game/:gameId/setup/questions
* GET: api/game/:gameId/play (*bestaat niet in FE*)
* GET: api/game/:gameId/rank (*bestaat niet in FE*)

Interessante tutorials met express en node:

* <https://www.toptal.com/express-js/nodejs-typescript-rest-api-pt-1>
* <https://javascript.plainenglish.io/create-a-typescript-express-server-in-minutes-7d34c306f60>
* <https://wanago.io/2018/12/03/typescript-express-tutorial-routing-controllers-middleware/>
* <https://auth0.com/blog/node-js-and-typescript-tutorial-build-a-crud-api/>

Nog te doen in de Express BE:

* Een apart bestand maken voor het configureren van alle routes (ik wil niet alles in het app.ts bestand plaatsen)
* Een nieuwe setup maken voor het add-players gedeelte (wat is het verschil tussen een Player en een Quizmaster? Het BE model komt niet overeen met het FE model, omdat een call getPlayers alleen Players moet terugsturen)