Design / Implementation decisions

Migration

For the data migration, I opted for a mongo database, where I could quickly set up an Atlas instance and use it to connect easily to my database. I initially started looking into Neo4j, but it felt like it would be more efficient to use a tool I am familiar with, on my time window. I added some extra nodes to the data you provided on the challenge description, to demonstrate that my solution works dynamically with any amount of nested data.
I made sure to drop the collection each time, to avoid overlaps, if ran consequently.

Potential improvements:

- Use Neo4j. I think it’s cool technology and I would like to look more into it if I had the time.

Front end

I decided to build the front end of the project with Vanilla JS. I have been affiliated with Angular 2 (v.12) in my current job, which would definitely prove to be a bad decision to setup for a smaller project like this, and the time window I had to work on the project was not allowing me to setup a Vue.js or React project (it would take me some extra time, since I haven’t revisited the libraries recently), so I opted for vanilla.
Since I also didn’t have much familiarity with the D3.js library, I borrowed a working example I found online that suited my needs and worked my way up from there.
I did some small cleanup of the D3 logic that I borrowed and adjusted it to my needs.

Potential improvements:

- Use Typescript. I initially setup a TS/Webpack project, but the old version of the D3 code was giving me much trouble to set up on a short time.
- Linter: Again, the old code version of the D3 I used was taking long to get implemented properly.

Back end

The back end is the heart of the project. I mainly focused on implementing the conversion functions (backend/convert.js) that efficiently transform the linear data structure you provided in the challenge description, to a nested tree structure that D3 is happy with.
I decided to make the data conversion on the back end, just before serving the data on the endpoint (/data), so that the front-end is just responsible for rendering what it gets and thus, the conversion is made inside the server.js file.
I also added an error handler in the server.js file, some unit tests for the data conversion functionality and a linter.

Overall, I really enjoyed the challenge, as it got me thinking about efficient implementations and using tools that I really like, both on the front and the back end.
I will be very glad to discuss more about this project with you and answer any questions you might have.

Thank you very much for the opportunity,

Nikos Argyriadis