The MongoExplorer

During the DWA server course you have been working with RoboMongo, a nice tool for developers to manipulate data and data structures in a Mongo database. Such database tools are often used in software development projects. You probably already worked with tools like SQL Server Management Studio or MySQL Workbench in projects using SQL databases. RoboMongo can be used in projects based on MongoDB.

RoboMongo is entirely written in C++. You can easily recognize this from it's design! ;-) Although C++ is a powerful programming language, it certainly has it's caveat, especially when it comes to the user interface. React doesn't offer all the technical possibilities C++ does, but it does offer much better support for design of state of the art user interfaces. The MongoExplorer should offer "RoboMongo like functionality" with a "user interface layer" that can be designed and configured technology independent.

Assignment:
Implement your own "MongoExplorer" completely written in JavaScript, using node.js, React and Redux.

Problem statement

During a software development project we'll usually need to store data somewhere. A tool that simplifies the creation of structures and data is a must have in any project, whether SQL or NoSQL. A well-designed user interface can make a difference. Although it is not our goal to create a cool user interface, React can help us to split logic and design. In this project we'll focus on logic. Our MongoExplorer however, should split logic and design in such a way that it will be easy to add a cool design at a later stage. Although we expect you to do your outmost in this area as well, you will not be graded on design aspects.

Our Goals for the MongoExplorer

A better RoboMongo, with a flexible user interface that ultimately should run on PC's, laptops, tablets and smartphones. So, whenever you implement functionality: MAKE IT GENERIC!!! Here is a list of "possible requirements":

1. React Document Explorer Component
   Implement a React document explorer. This component takes a json as an argument. During this stage the json argument can be a static structure. It dynamically builds an explorer showing the json structure as a tree. The component can be called as a "standard React component" from any other React component. Start with simple json structures. Gradually add more complex structures. Eventually the component should able to handle any kind of complex json structure. Test it extensively!

2. Server API
   Implement an API on the server using node.js. Your api should be able to display (1) all databases and collections, (2) all collections in a specific database and (3) all documents within a specific database/collection. Test it extensively!

3. React Component which brings 1 and 2 together
   Implement a React component that calls the api (stage 2) with the necessary arguments, fetches the returned json and calls the React document explorer component (stage 1) to show the json structure.

4. Cool stuff
   Add more functionality to you React document explorer component (stage 1): (1) collapse/expand nodes, (2) highlight active document, (3) add icons to nodes, and so on. Do you have idea other cool idea's? Add them to the list!

5. A working MongoExplorer
   Make sure you can show multiple React document explorers on one page. Add reload functionality that allows components to update each other. Implement a page holding twee document explorers. On the left all databases and collections are shown. If the user clicks on a collection, the document explorer on the right will show all documents within that collection. (like RoboMongo)

6. Multiple platform research
   Investigate how to make your MongoExplorer available on as many platforms as possible. What are the pre-requisites for installation? What can you do to make it simpler/easier? Is it possible to offer a stand-alone version? How? Is it possible to offer your MongoExplorer on smartphones? On tables? How? What are the pre-requisites? Do some research! You don't have to implement anything during this stage.

7. More ideas…?
   What interesting features, options, functionality can you add more? Try to come up with as many interesting ideas as you can. You do not have to implement them all! Which ideas do you like most? Implement as much as possible.