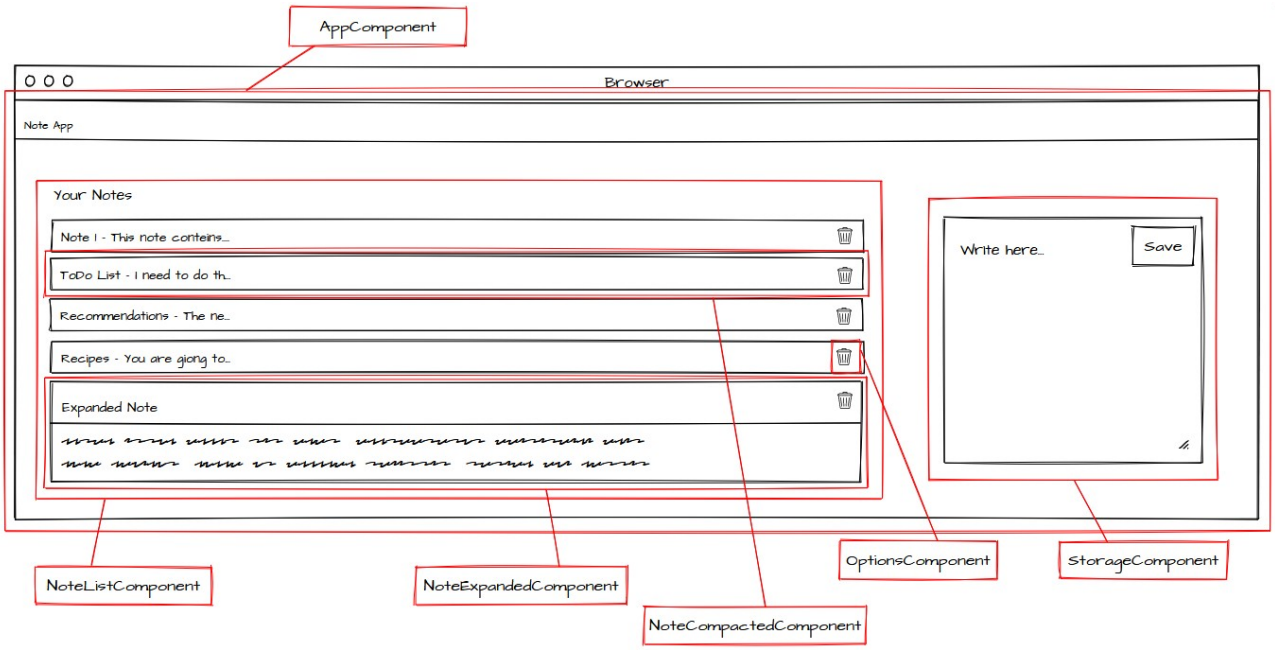
3. Web App Restful API System Design

3.1 Describe high level design

I am assuming that we have a component called AppComponent that is going to be the parent of the project and will contain the rest of components.

NoteListComponent → Will be the responsible of displaying all the notes, firstly they will be shown using NoteCompacted version. If we click on one of them it will change to a NoteExpanded version.
NoteCompactedComponent –> Display summary of note information and can interact with OptionsComponent in case we want to remove the note.
NoteExpandedComponent –> Display an expanded view with more information of the note and also will interact with OptionsComponent if we want to remove the note.
StorageComponent –> Component responsible of keep the new note and once it is clicke the button to save send it to the Backend.
OptionsComponent –> Element that will contain an icon to remove the note. This element will be inside the NoteExpandedComponent and NoteCompactedComponent.

3.2 Web App UI



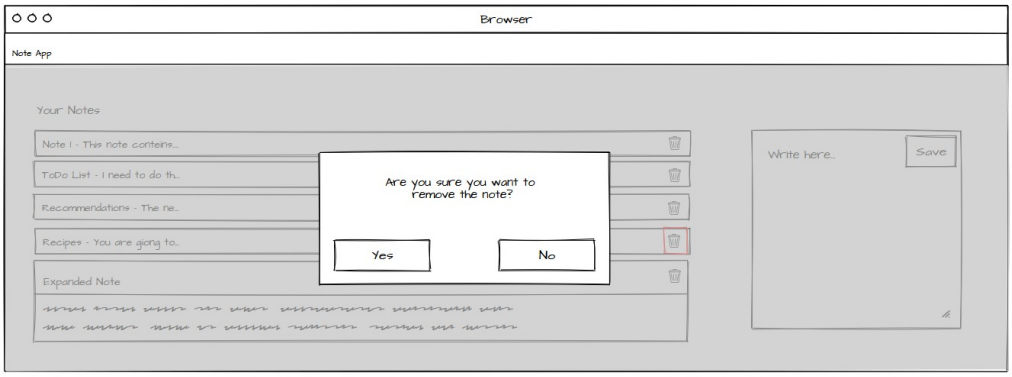
3.2.1 Consider what UI components are required and how these interact with the other components.

In NoteExpandedComponent we could have two UI components: one for the header that would contain the OptionComponent and would interact with it passing the note id, the second UI component would contain the content of the note in a different style.

We could have the titles of NoteCompactedComponent and NoteExpandedComponent as a separate UI component so in case we need to do changes in both we can modify only once css.

3.2.2 What (if any) validation is required?

To remove the note we would need some confirmation. To make use of this, I would create a Component (ConfirmationComponent) as a pop up to double check before doing the action. Here you can see an example of how it would be displayed:



Also we have to be cautious with the number of characters that it is allowed in a note. So in Frontend we could add a validation in case the characters are higher that a number we decided appear a pop up with the information and not allowing to write further

3.3 Data Model

Assuming we don't need to create any structure for the user database these would be the data model that I would create.

Note Table - SQL Database

IdNote - varchar(40), – Primary key

userId - varchar(255),

IdNoteText - varchar(255),

title – varchar(255),

creationDate- long

NoteText Table – NonSql Database

primaryKey: IdNoteText – string,

value: Would contain an object with the following data:

noteText – BinData

3.4 Restful API

GET noteapp/notes - Method to get all user's notes. We would use the session information to be able to get user information required to get all the notes essential information form the database. Once we get it we will return the list of essential.

GET noteapp/note/4444 - Method to get specific note information. Same as with the previous call we would use the session information to get the user id needed to request the note id from the sql information. In case the user has this note id in the database we would request the text from the nosql database and return it to frontend.

POST noteapp/delete/4444 – Method to delete a note for the user. After checking the user is correctly logged. We would remove from the nosql database the text information and the row from the sql database. After the remove we would send back a confirmation that it is correct.

POST noteapp/save – Method to add a new Note. We would receive blob information so we can store the note as binary data in the nosql db and the ids in the sql db. We would send a confirmation that everything went well back to frontend.

3.5 Web Server

I would create a rest controller to receive all the calls under *noteapp* context. Then I would use the annotation GetMapping or PostMapping depending of the call method. This controller would basically call to a service that would be the responsible to generate the correct response.

The service would manage all the business logic. For all the calls would check if we get a user from the session information we have received. Once we have the user information I would implement a method that fill the objective described in the previous exercise.

I would use two JPA repositories to connect with the sql and nosql databases and would create the proper model to parse it.

In case of adding a new note I could return a boolean but in case we are fetching all the notes or only one note I would return an Object or list of objects Note with the text, title and date of creation.

The notes are saved in a noSql database so it is easy to handle a flexible amount of notes and will be able to be more performative since a semi-structure data could be benefitial in case we add more note types and maybe we do not want to store the same information for different types of notes. Also the text is store in binaryData to be unreadable from the database.

3.6 Further questions to consider

3.6.1 What user login and session strategies could be used?

I would use a token-based authentication strategy so from that information I can get later user information needed in the requests. With this we can be sure that they have enough access to get the information they need and make the system more secure.

3.6.2 How would your UI solution facilitate extension and reuse?

Since the components are grouped by functionality we can reuse this same code in other platforms like tablet or mobile just changing the styles.

Also the option for example of creating a component for the delete operation as Options Components make possible for us in the future to add other functionalities to share the notes or other modifications with basic changes.

3.6.3 What as considered when deciding on a storage component?

I wanted to be easy to use and direct to I decided to directly add the functionality in the component and add it under AppComponent.

3.6.4 How might the capability for a user to share a note publicly affect the design?

We would need to add the button of the social network we want to share or a specific share icon in the OptionsComponent but with the current implementation should be simple.

3.6.5 What additional capabilities would be useful, and how would these affect the design?

The ability to edit notes that you have created → Will need to add an icon into the options and add that functionality into the NotedExpandedComponent for example.

Support different types of notes like check lists → We would need to create a component that implements the changes in the types and we would need to add it in StorageComponent and in NotedExpandedComponent.

Support import and export notes → We would need to add the options in the OptionsComponents

Implement a deleted notes in case you need to recover some notes → We should add a new component with this information and with different styling also this change would require to modify the backed to do not remove the notes and mark them as not visible so we can look for them.