Web App Restful API System Design

Bruno Simm

The problem being solved involves a note web app, where a user can write a note, save a note, view a list of their notes, and delete a note. The user's notes are saved so that they are available via any web-capable device.

High Level Design

The proposed solution has the following high-level design that involves the following components:

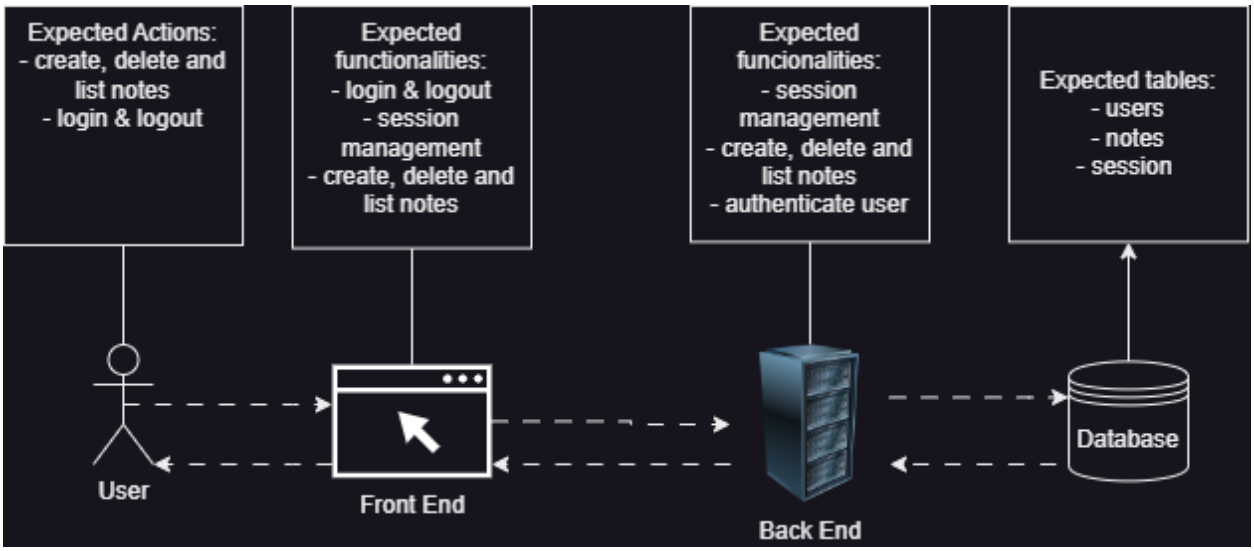

Figure 1 - High Level Design

- Front End
  - Handle login & logout functionalities to authenticate the user.
  - Handle the user session management, sending the session id to the backend.
  - Handle the process to create, delete and list notes from the authenticated user. Also apply validations when creating new notes.
- Back End
  - Handle user authentication in every request using the user session id.
  - Session management to create, validate and invalidate sessions.
  - Create, delete, and list notes considering authenticated user context.
- Database
  - Store users, notes, and sessions data.

Web App UI

The designed UI can be viewed in the following FIGMA LINK (Prototype interactions are also available).

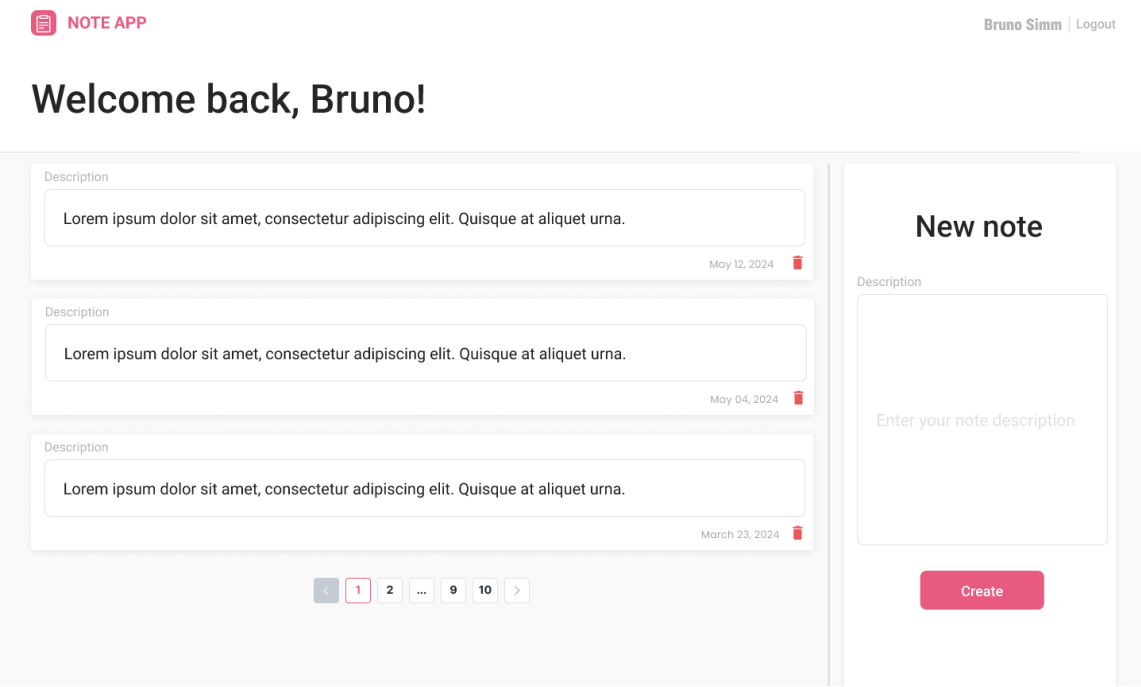

Figure 2 - UI Design

The above UI have 4 different components:

- Home
- CurrentUser
- NewNote
- NoteList

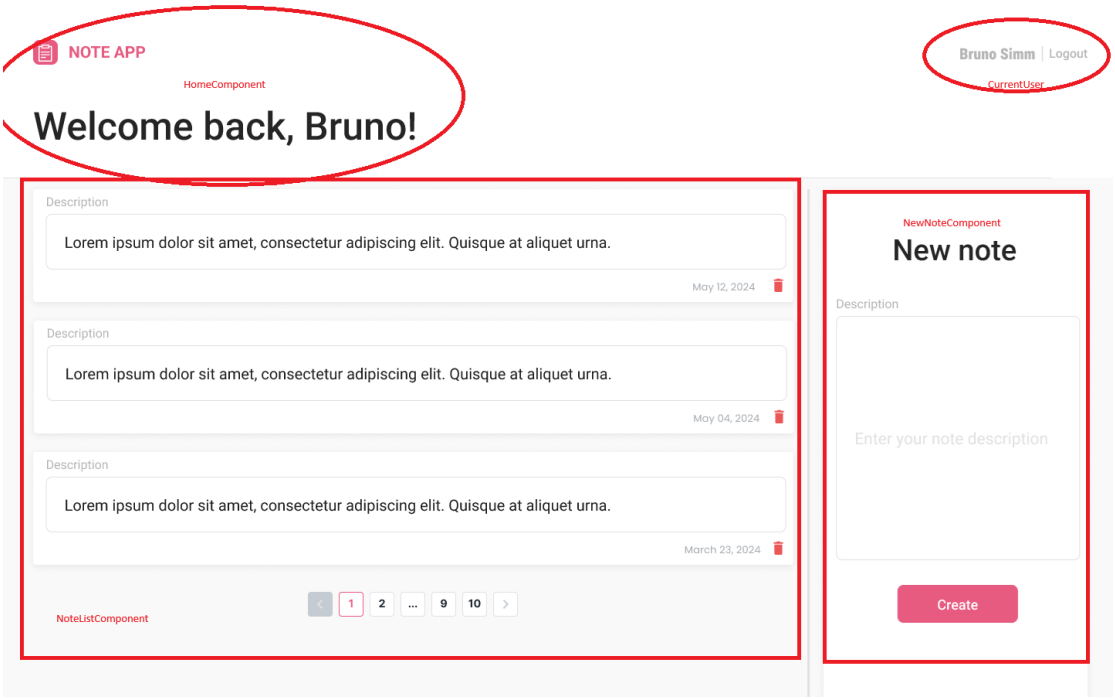

Figure 3 - UI Components (in red, the separated components)

Home Component

The “Home” component group all other components and corresponds for the welcome message, the app name in the top left corner.

CurrentUser Component

The current user component displays the current authenticated user and handles the logout functionality.

NewNote Component

The “NewNote” component it’s a fast way to create a new note in the home screen. The required field “description” should have a validation to avoid sending it empty to the backend. After hitting the “Create” button the “NoteList” component should be updated with new note.

NoteList Component

The “NoteList” component list and paginate notes from the authenticated user. Every time a new page is requested the frontend should fetch data from the backend and then display it.

Data Model

The solution will need 3 entities to solve the problem, Users, Notes and Session table.

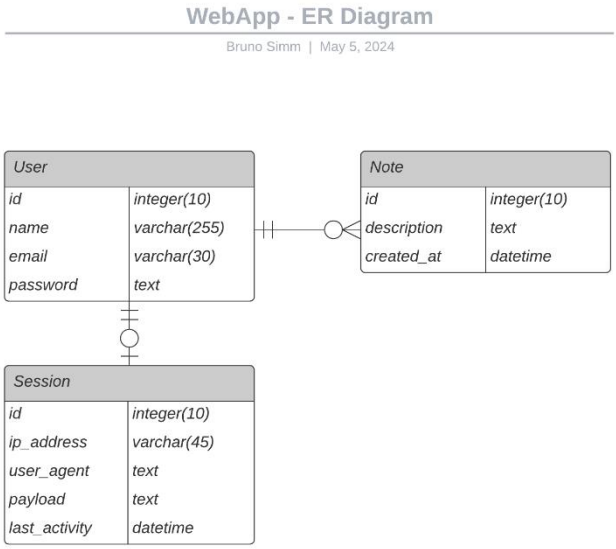

Figure 4 - ER Diagram

The **User entity** stores data related to the user and have relationships with Note and Session tables. The User has a *"OneToMany"* relationship with Notes and a *"OneToOne"* (the session cannot exist without a User) relationship with the Session table.

The **Note entity** stores data related to notes and have relationships with the User table. The Note has a *"ManyToOne"* relationship with the User.

The **Session table** stores data related to user sessions and have relationships with the User table. The Session has a *"OneToOne"* relationship with the User table.

RESTful API

Endpoints

The note resources have the URL "/notes" and can be accessed using the following routes:

- **GET /notes**: This endpoint retrieves the paginated list of notes belonging to the authenticated user. User can send the page and size parameters to filter the paginated data.
- **POST /notes**: Used for creating a new note. The client sends the note content in the request body, and the server saves it for the authenticated user.
- **DELETE /notes/{id}**: Deletes the note identified by {id}. Only the owner of the note can delete it.

Authentication

All endpoints require authentication to ensure that only authorized users can access or modify their notes. This aligns with the assumption that users must be logged in to use the note app.

Authentication must be implemented using session cookies and should be a required parameter in every request.

Error Handling

The API should handle errors and provide informative error messages. For example, if a user attempts to delete a note they don't own, the server should respond with a 403 Forbidden status code or if any request comes without “session_id” the server should respond with a 401 Unauthenticated error status code.

Common error scenarios that should be handled include invalid authentication credentials, malformed requests without “session_id”, and attempts to access nonexistent resources.

Web Server

To implement the Restful API, it is possible to use the following stack: Spring Boot for the backend, React for the frontend, and a relational database.

Actions

Create Note

To create a new note the backend must validate that the note content its not empty. Eventually if the note content has a max length requirement it should be validated in the creation action.

Delete Note

To delete a note the backend must validate that the selected note belongs to the authenticated user. If the note doesn’t belong to the authenticated user, then it should throw an exception with the status code 403 (Forbidden).

List Notes

The list note action must only return notes that belongs to the authenticated user and should be paginated.