1. Describe high level design

Show the main ***note app*** components and the logical interactions that will fulfill the requirements.

Front-end:

- The users interface consists in a web page responsive, which need to be accessible in mobile and computer browser.
- Components must have: A list of notes from the authenticated user that is shown in the home. A form to create new notes, a button to delete a note.
- In the list must have the following columns: id, title, content, actions.
- The first action available is to delete a note.
- The web page uses css, html and Javascript to create a good experience to the user, friendly and intuitive

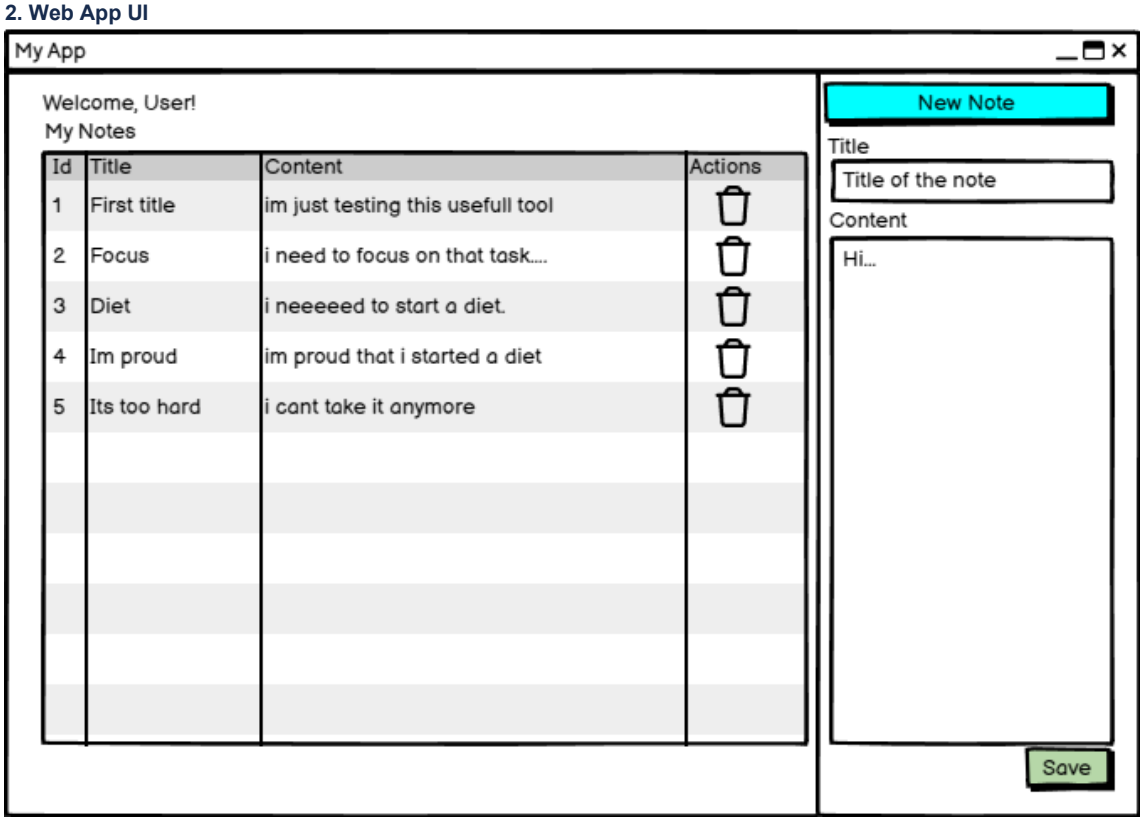
Interactions:

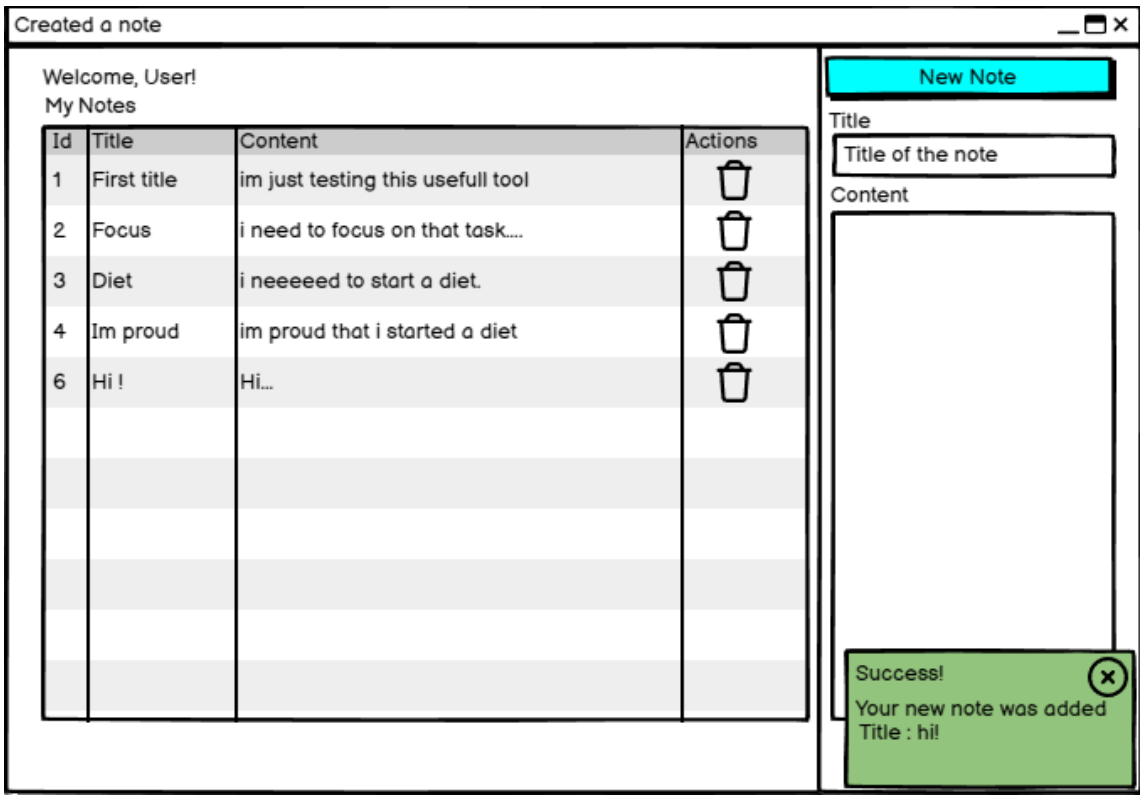
1 – User log in and is redirected to home. In the home we have a list with all notes, and form to save a new note.

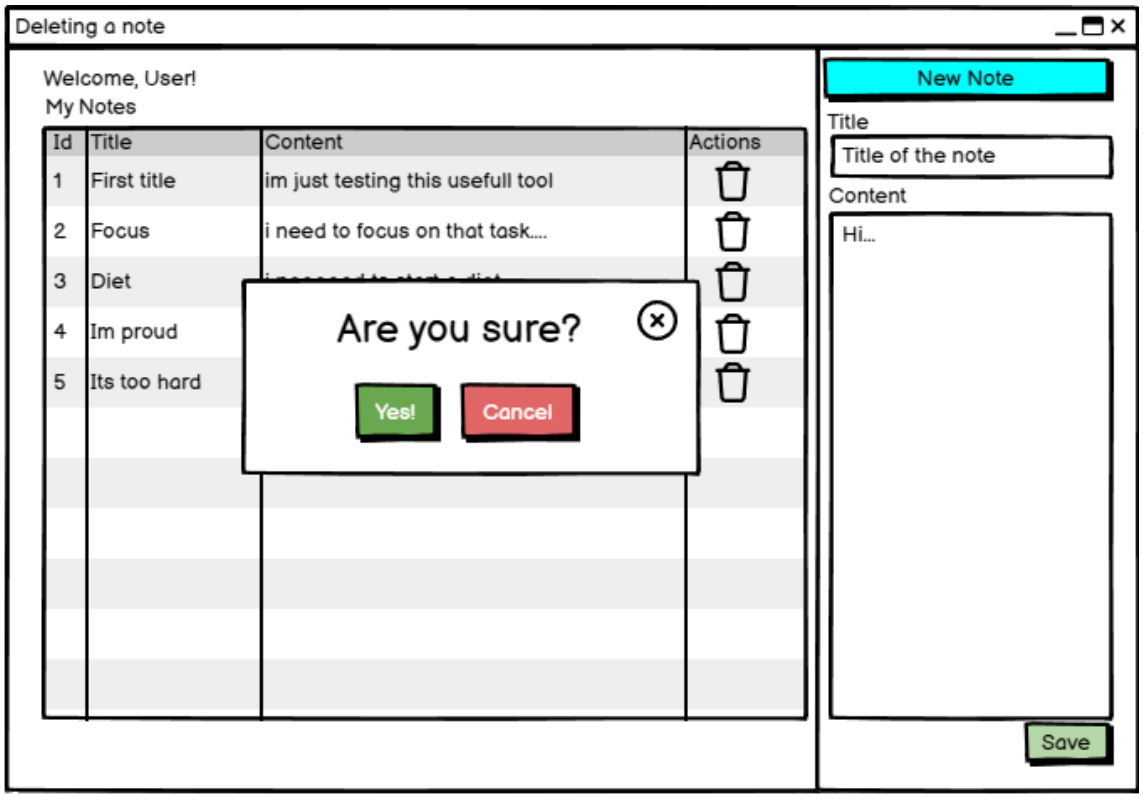
2 – User is logged and want to add a new note. He types the title and the content, then click save. The app will send to the backend api, the api will receive the data and validate, then store in the data base. The list component must update and show the new note.

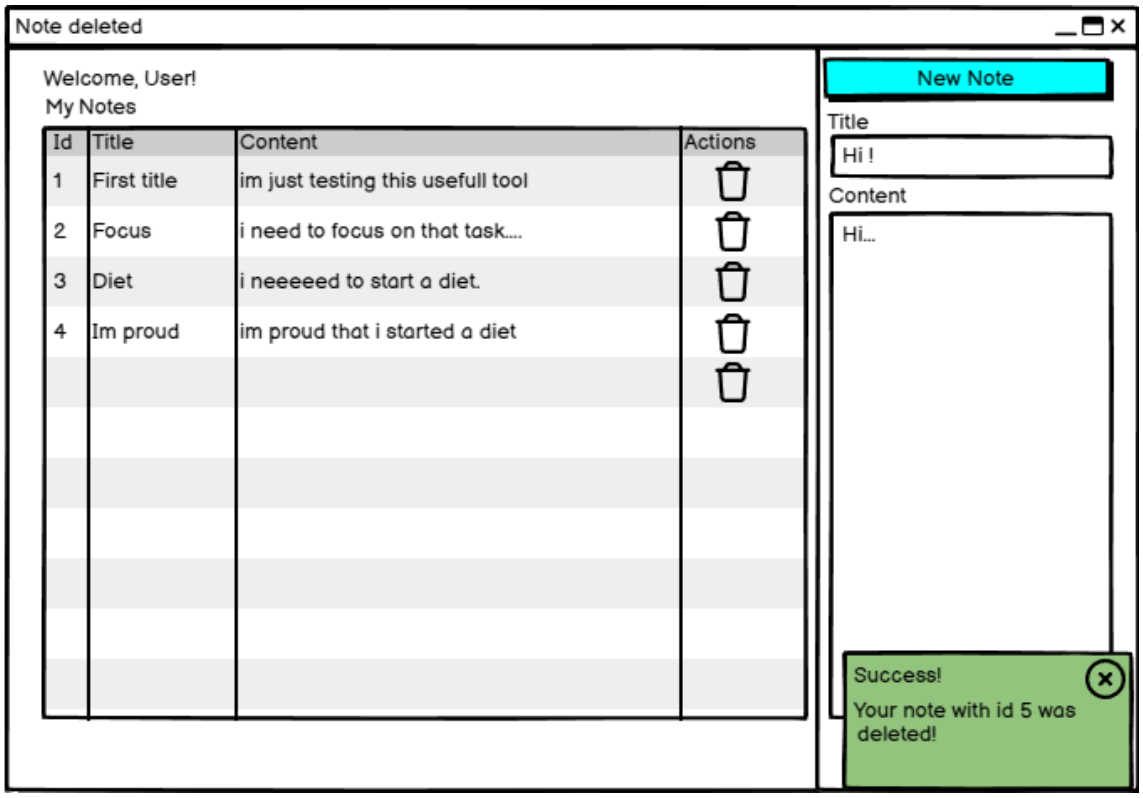
3 – User is logged and want to delete a note. He click at the icon trash can, a modal opens asking if he's sure, if he clicks yes then the note is deleted, if he clicks cancel, then the modal closes. If he clicks at the X button, the modal closes. If he delete a note, then a notification must be sent to de app, confirming the act.

2. Web App UI


My App
Welcome, User!
My Notes
Id
Title
Content
Actions
1
First title
im just testing this usefull tool
2
Focus
i need to focus on that task....
3
Diet
i neeeeed to start a diet.
4
Im proud
im proud that i started a diet
5
Its too hard
i cant take it anymore
New Note
Title
Title of the note
Content
Hi...
Save

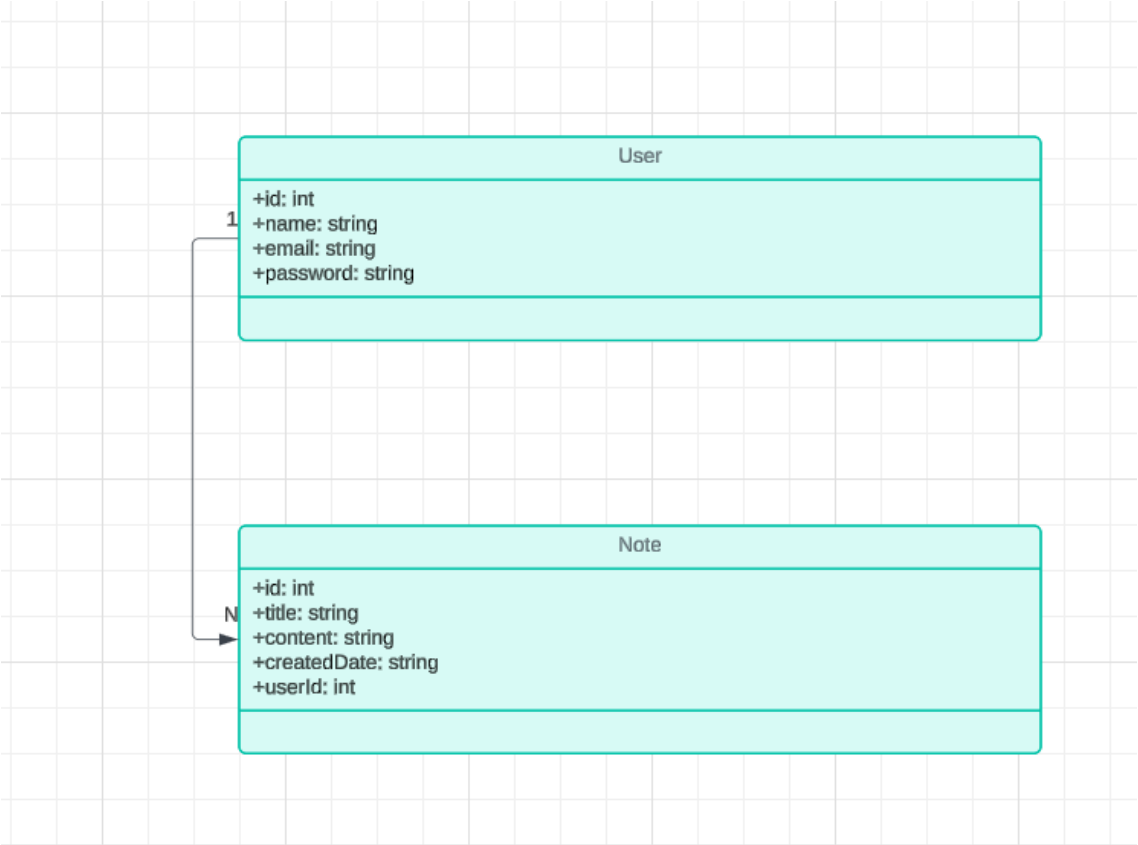

Created a note
Welcome, User!
My Notes
Id
Title
Content
Actions
1
First title
im just testing this usefull tool
2
Focus
i need to focus on that task....
3
Diet
i neeeeed to start a diet.
4
Im proud
im proud that i started a diet
6
Hi !
Hi...
New Note
Title
Title of the note
Content
Success!
Your new note was added
Title : hi!


Deleting a note
Welcome, User!
My Notes
Id
Title
Content
Actions
1
First title
im just testing this usefull tool
2
Focus
i need to focus on that task....
3
Diet
4
Im proud
5
Its too hard
Are you sure?
Yes!
Cancel
New Note
Title
Title of the note
Content
Hi...
Save


Note deleted
Welcome, User!
My Notes
Id
Title
Content
Actions
1
First title
im just testing this usefull tool
2
Focus
i need to focus on that task....
3
Diet
i neeeeed to start a diet.
4
Im proud
im proud that i started a diet
New Note
Title
Hi !
Content
Hi...
Success!
Your note with id 5 was deleted!

3. Data Model

Describe how a note will be modelled
consider the required Properties



The note model is build by

- Id Integer auto increment not null
- Title varchar(50)
- Content varchar(255)
- Created_Date datetime default getdate()
- User_id Integer not null (foreign key)

4. Restful API

Describe the Restful API required to fulfill the note app.
how would the web app get the user's notes?
how would the web app save a user note?
what are the URL for the note resource(s)?
and verbs to expose the actions?

- This rest api will have a note controller, service, repository, mapper and Entity.
- The web app will do requests to the backend application using restful, with the verbs get, post and delete, patch or put is not necessary at this point.
- Controller: Its the entry point that can handle the https requests.

- Service: Its Where we implement our business rules and Interact with the repositories.
- Repositories: Its the data layer, where we persist data and Interact with the database.
- Entity: Here we define the attributes we have in this Entity.
- Mapper: I like to use this class, because its a good practice to never return the entities in our controller class, so i always do a mapping from the entity to the dto, or when needed to the dto to entity.
- The backend has a context-path which is "/api/v1" so all requests must have this path before the resources path.
- The resource note request mapping is "/note" so all the requests calls that refers to a note, need to start with "/note"

Controller:

1 – Create a note:

POST /api/v1/note (returns 201 status created)

Json:{ "title": "Title",

"content": "Content",

"createDate": "2024-06-10T01:51:04.679Z",

"user": 1 }

Getting the user's notes:

GET /api/v1/note/{userId}/{page}/{pageSize} (returns a page with the 10 notes, ordered by note id, desc, with pageSize 20 by default.)

Deleting a user note:

DELETE /api/v1/note/{noteId} (returns 204 no content)

5. Web Server

Describe how the webserver implements that Restful API:
consider how each action will be implemented
what (if any) business logic is required?
how are the notes saved?

- To get the users note, we have an endpoint with a path variable id which represents the user.id, and path getAll, so the full endpoint is "GET /api/v1/note/{userId}/{page}/{pageSize} ").Returns a List<NoteDTO> (for better performance its paginated), which contains for each note the following attributes:
  - Integer id

    - String title
    - String content
    - LocalDateTime createDate

- To save a user note, we have an endpoint with path “POST /api/v1/note” with a @RequestBody in Json format. Returns a response entity status created (201). This body contains the following attributes:
    - String title
    - String content
    - LocalDateTime createdDate (by default is the now() time)
    - Integer userId
- To delete a user note, we have an endpoint path “DELETE /api/v1/note/{id}”. The id in path variable must be the id of the note. Returns a response entity no content (204).

The business logics are:

- If the app don’t send the right information in the path variable we answer a notFoundException.
- The note Title must have the max of 50 characters, not null.
- The note Content must have the max of 255 characters, not null.
- The note UserId must be not null.
- When a Note is deleted, the attribute active is changed to false, so its never truly deleted.