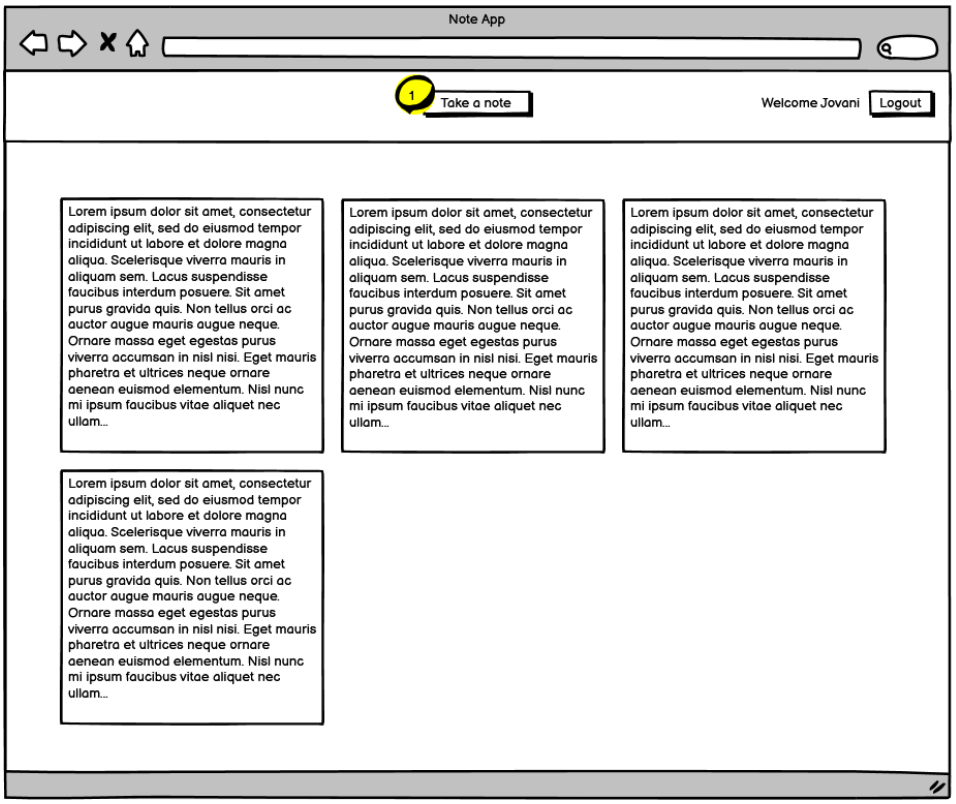
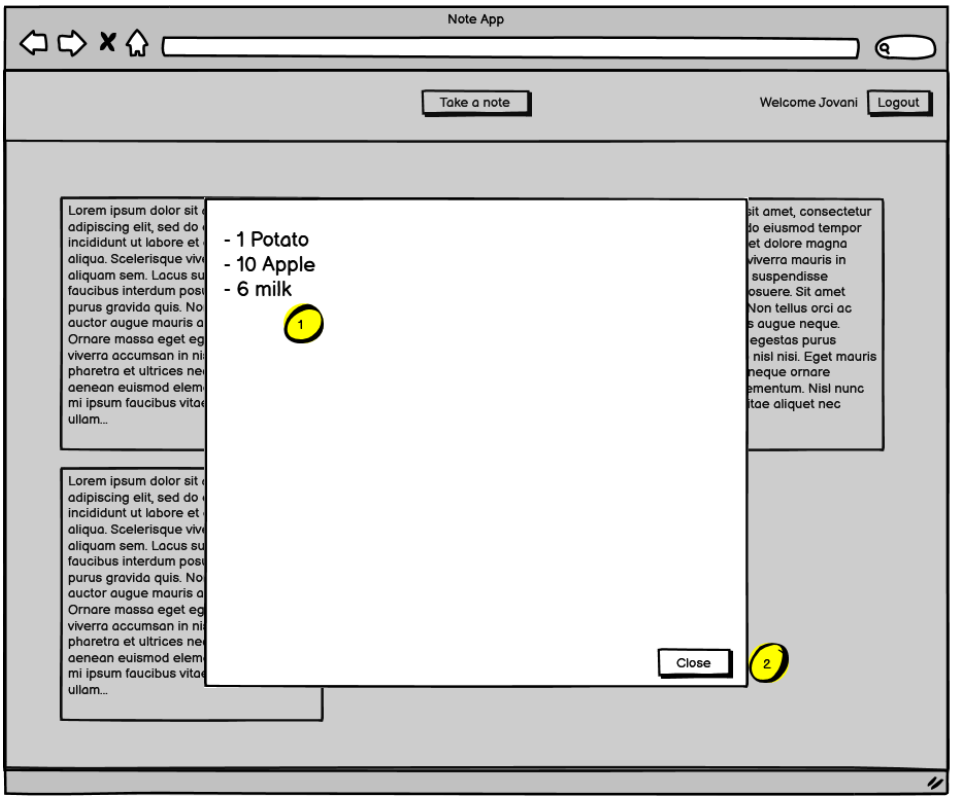
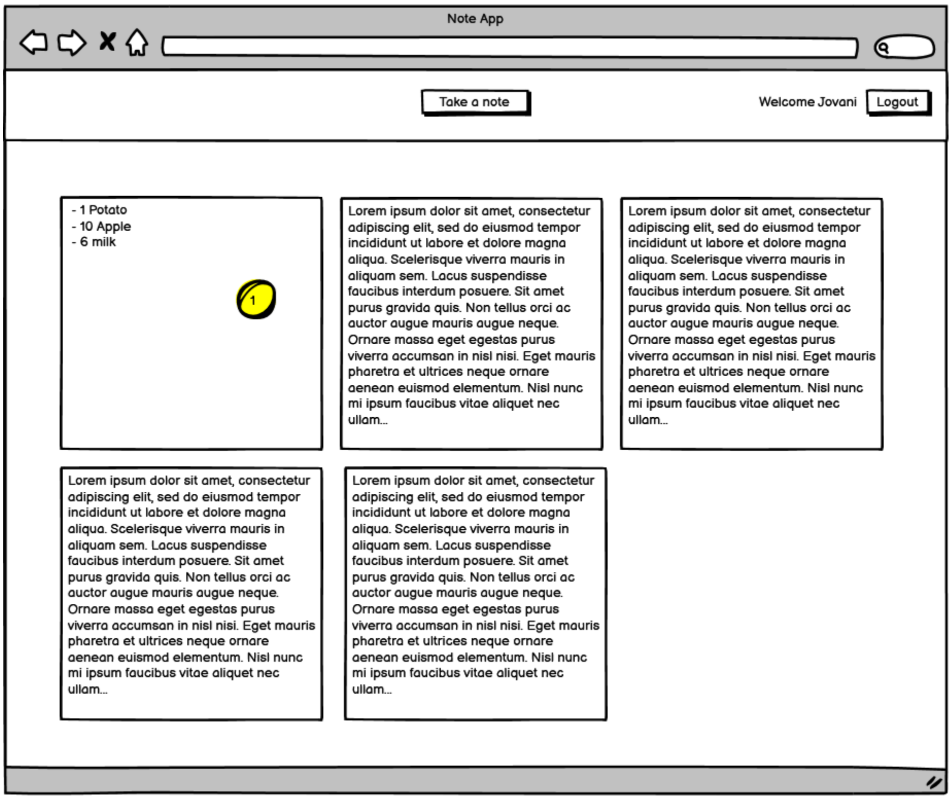
3. Web App Restful API System Design

1 Describe high level design

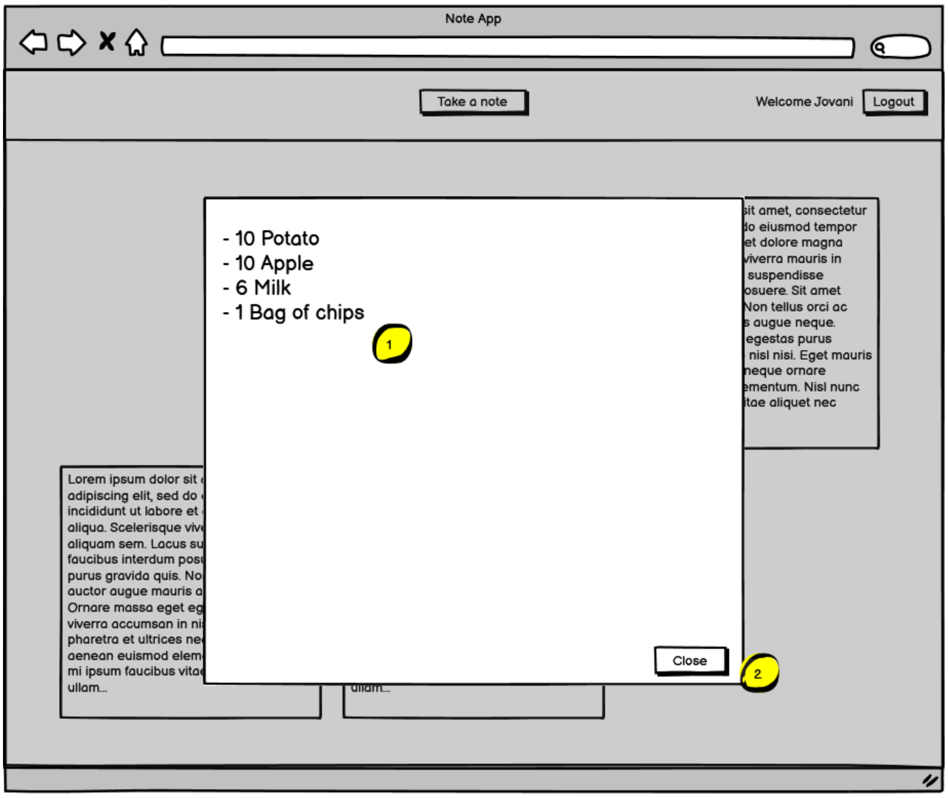
The Note app would work as shown in the screenshots below:





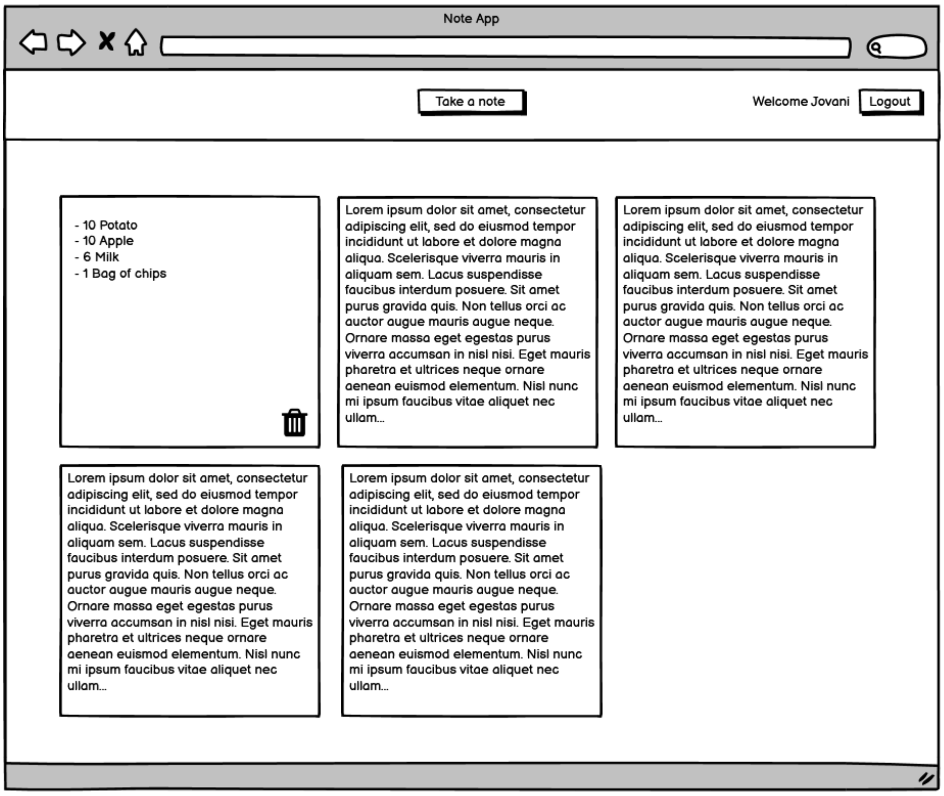


1 Click in the note to edit it



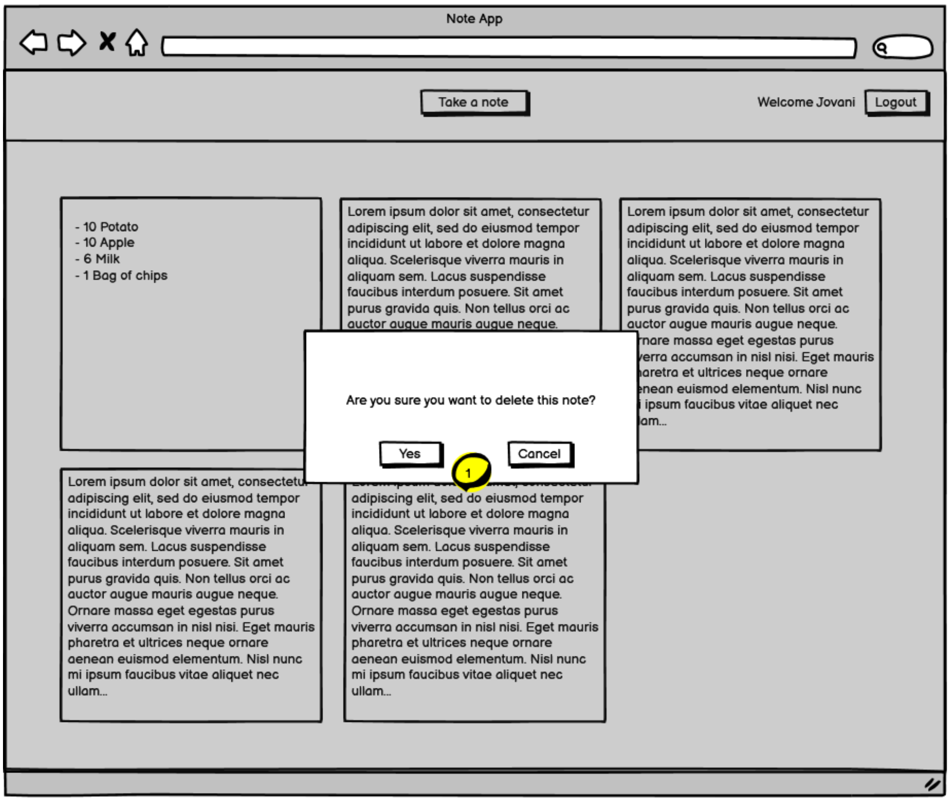
1 Edit the note

2 Click close or outside de card to save the note



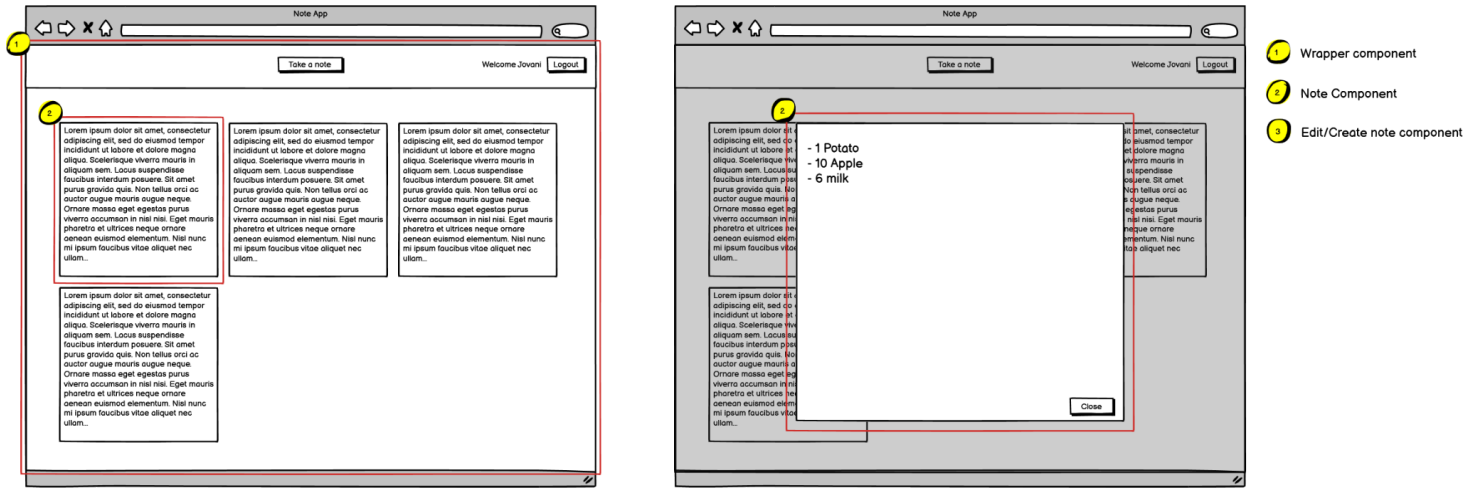
1 Hover over the note to the delete icon appear

2 Click in the trash icon to delete the note



1 Click "Yes" to confirm or "Cancel" to cancel

2 Web App UI



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

We would have a Wrapper component that would be initialised at the moment we login to the app, followed by the note component. Once you click to take a note, a popup will be displayed on the screen with the option to add content to the note, you can click close or click outside the window to save the note. Once the note is saved the dom will be updated and will rearrange the notes automatically with the latest note at first position.

To delete a note, the user would hover over the note to see the trash icon appear, after clicking on the trash icon, a popup would appear asking *“Are you sure you want to delete this note?” the user can choose between “Yes”, to delete the note, or “Cancel”, to cancel the deletion.*

2.2 What (if any) validation is required?

Validation is required both in the front-end and the back-end. Validating Max characters in the note, as well if the number of the note that will be deleted/ updated or retrieved exists in the DB. Validate If the user is logged to the app to perform the actions.

3 Data Model

The data model would be as follow:
NOTE_KEY - type=Varchar(40) - PK - nullable=false
USER_ID - type=Number - nullable=false
NOTE_CONTENT - type=Text(20000) - nullable=false
MODIFIED_DATE - type=timestamp - nullable=false
Considering that the User table would be already being created.

4. Restful API
4.1 How would the web app get the user’s notes?
The frontend would make a GET request to the backend, that would do a query in the Database passing the USER_ID as parameter, retrieving all the notes of the user.

4.2 How would the web app save a user note?
The frontend would do a POST request to the backend, passing the USER_ID and NOTE_CONTENT, the backend would handle the request saving the content to the database with the USER_ID as

parameter and will send a response with the NOTE_KEY, USER_ID, NOTE_CONTENT and MODIFIED_DATE of the note.

4.3 What are the URL for the note resource(s)? And 4.4 and verbs to expose the actions?
GET - ““notes/v1/get-note/{NOTE_KEY}” - to get all notes
GET - “notes/v1/get-all-notes/{USER_ID}” - to get all notes
POST - “notes/v1/save-note” - to create notes
DELETE - “notes/v1/{NOTE_KEY}” - to delete the note
PUT - “notes/v1” - to update the note

5. Web Server
5.1 consider how each action will be implemented

```java
@RestController
@RequestMapping("notes/v1")
@RequiredArgsConstructor
public class NotesController {

    private final NotesService notesService;

    @GetMapping("/get-note/{noteKey}")
    public ResponseEntity<NotesDTO> findNoteById(@PathVariable(value = "noteKey") String
noteKey){
        NotesDTO note = notesService.findById(noteKey);
        return new ResponseEntity<>(note, HttpStatus.OK);
    }

    @GetMapping("/get-all-notes/{userId}")
    public ResponseEntity<NotesDTO> listAllNotes(@PathVariable(value = "userId") Long userId) {
        NotesDTO notes = notesService.listAllNotes(userId);
        return new ResponseEntity<>(notes, HttpStatus.OK);
    }

    @PostMapping("/save-note")
    public ResponseEntity<NotesDTO> createNote(@Valid @RequestBody NotesDTO notesDTO) {
        NotesDTO notes = notesService.save(notesDTO);
        return new ResponseEntity<>(notes, HttpStatus.CREATED);
    }

    @DeleteMapping("/{noteKey}")
    public void deleteNote(@PathVariable(value = "noteKey") String noteKey) throws
ResourceNotFoundException {
        notesService.deleteNote(noteKey);
    }

    @PutMapping
    public ResponseEntity<NotesDTO> updateNote(@Valid @RequestBody NotesDTO notesDTO) throws
ResourceNotFoundException {
        NotesDTO note =  notesService.save(notesDTO);
        return new ResponseEntity<>(note, HttpStatus.OK);
    }
```

5.2 what (if any) business logic is required?

First, the user needs to be a valid user and logged to the platform to perform the actions. Then, in order to delete the note, this note has to exist in the database, if not, a notification needs to be sent by the front end. A maximum of 20000 characters per note can be sent to the backend. (this can be checked by adding a tag to the DTO).

5.3 How are the notes saved?

The notes are saved in the relational database as type text.