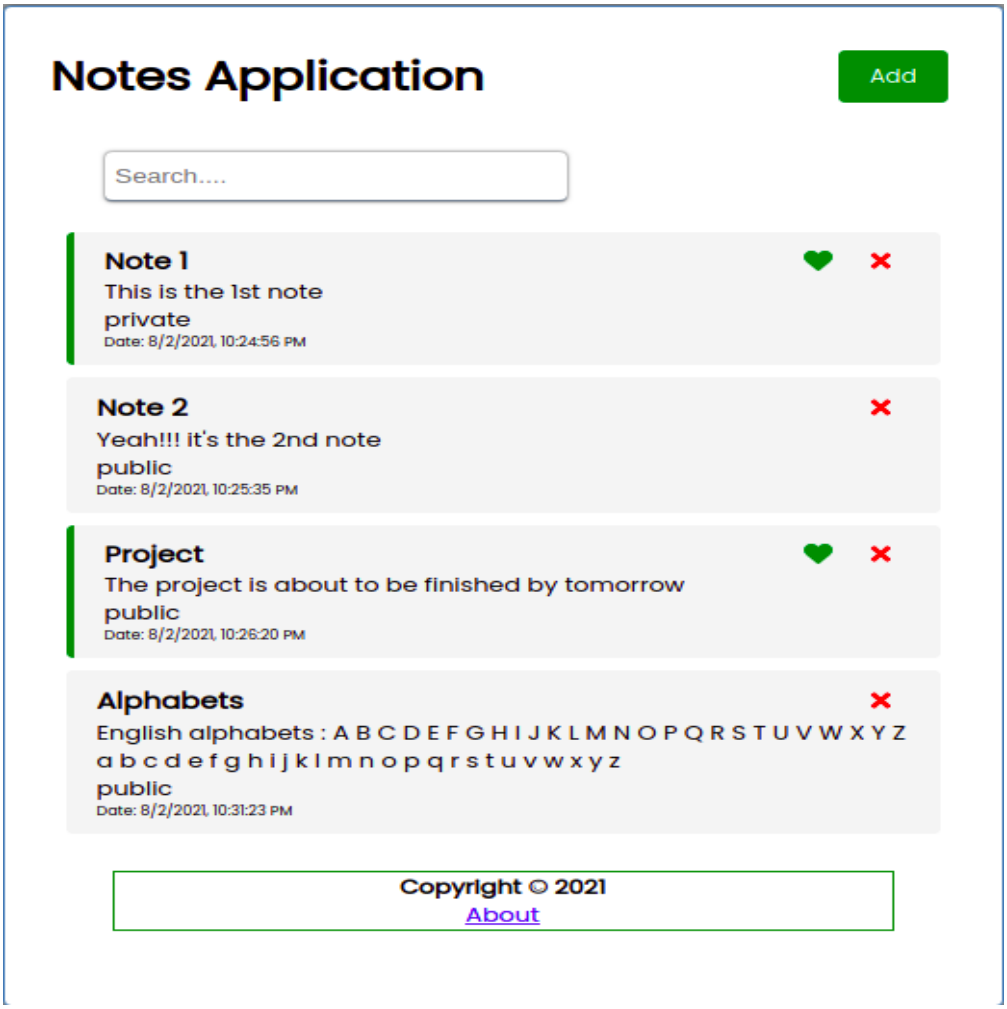
Notes Application

> The objective of this project is to create a simple web application , and perform a various tests on the application.

Application Interface

- As the name suggest " **Notes Application** ", we have created a simple application to write down notes and display them to the user within a simple interface created with **Angular** famework and **TypeScript** language, as well as a database to store the user's notes by using the **json-server** model
- **Angular** Angular is a TypeScript-based free and open-source web application framework led by the Angular Team at Google and by a community of individuals and corporations. Angular is a complete rewrite from the same team that built AngularJS.
- **TypeScript** is a programming language developed and maintained by Microsoft. It is a strict syntactical superset of JavaScript and adds optional static typing to the language. TypeScript is designed for the development of large applications and transcompiles to JavaScript.

Interface

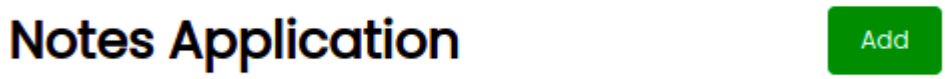


- As we see in the picture, the user interface contains a lot of sections, each one of them within its own component, so we can devide to application into three components :

- **Header Component**
- **Notes Component**
- **Footer Component**

Header Component

In the header component, we've the title of the application as well as a button with the text **Add** basically used to add new note (*more on the Notes Component*)

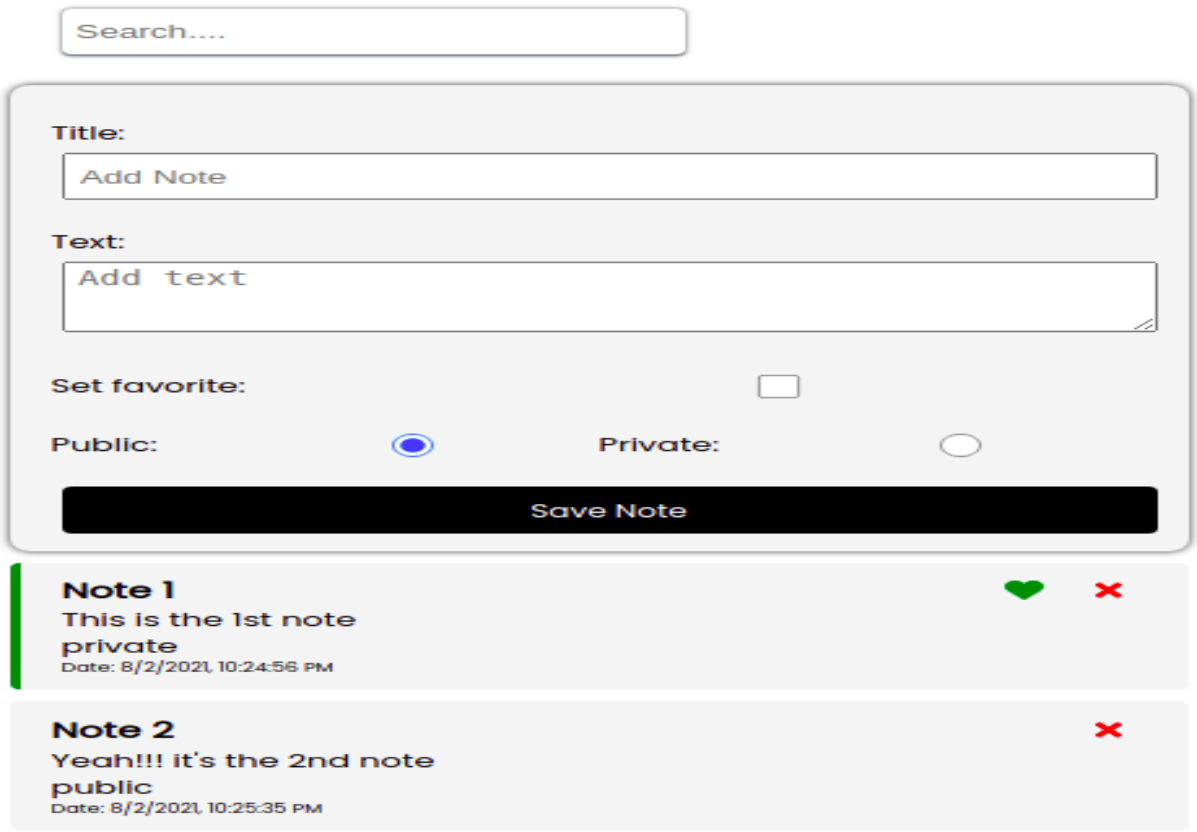


Footer Component

We don't have a lot in the footer, just a simple "copyright" © text and a link to navigate to the ***About Page*** that does the same as the footer, just a simple text with a link to **Go back home** (__More on the Routing section)

Notes Component

- Notes Component or as we can call it the **main** component, it's the the one who's responsable about showing the existing notes, add a new note, search for a particular note, ...



- As we can figure out from the picture above, in this component we've a lot of sibling items, each within its own component:
    - Seach Component
    - Form Component
    - Notes Items Component
- `Seach Component` : is this one we've an input field used to filter the displayed notes based on their title and its match with what we're loocking for.
- `Form Component` (*`add-note-component`* in our code):
    - this component contains the form that's responsable of adding new notes. By default it's hidden, to show/hide it we use the *Add/close* button in the header. It contains all field needed to create a note such as *Title, Text, Favorite, Type (public/private),Date, ...*, and a button to save the note in the DB.
    - Before sending any note t the DB, firstly we make a check to verify it,*has a title and a text set*, also to make sure that it match the **Note** interface that we've declared in the ***Notes.ts*** file.

> Note: the *Date* property of the note is generated automatically using the `Date()` Object.

- `Notes Items Component`: this is the one who his job to shows us the notes that we've in the DB.

Testing

- As we mentioned earlier, the main objective it to perform tests against our application to simulate some kind of the end user actions and activities. In our case we perfom some **E2E** (*End to End*) testing using a tool called **Cypress**
- **E2E Testing**: refers to a software testing method that involves testing an application's workflow from beginning to end. This method basically aims to replicate real user scenarios so that the system can be validated for integration and data integrity.
- **Cypress**: Cypress is a purely JavaScript-based front end testing tool built for the modern web. Cypress is a more developer-friendly tool that uses a unique DOM manipulation technique and operates directly in the browser. Cypress also provides a unique interactive test runner in which it executes all commands.
- The tests that we're interessted on are :
    1. The verification of the content loaded: if there are any missing element.
    2. The verification of the routing proccess: we try to visite other pages and check if every thing wents correctly as we want it to be.
    3. Check that we've the same number of notes as in the DB.
    4. Check that we can add notes.
    5. Check that we can delete notes.
    6. Verify that we can change the favorite property of a note.

7. Check that the searching process is working and gives us good results.

and for each one of those test we basically create a new Cypress method to make that easy.

1. Verify the content

- In this test we simply check for the existing of the Add button within the header component

```
 //  1st simple
  it("has add button", () => {
    cy.contains("Add").should("have.length",1);
});
```

2. Routing Process

- With this one we try to visite the **About** page and verify it's content as well as the URL

```
// Visit the about page
  it("Visit routes", () => {
    cy.get("a").click();
    // confirm that we visit the right URL
    cy.url().should("contain", "about");
    cy.contains("About page").should("have.length", 1);
    cy.contains("Version").should("have.length", 1);
  });
```

- To config routing we use `RouterModule` and `Routes` models, and setup the paths

```
const appRoutes : Routes = [
  { path: "", component: NotesComponent }, // config the home page
  { path: "about", component: AboutComponent} // config the about page
]
```

3. Notes in DB

- Basically we've 4 notes stored in the DB, but it's possible that we add some notes during tests or things like that, so to avoid any error we configure our test to confirm that we've at least 4 notes

```
// check N° of notes we've in DB
  it("Number of  notes in DB", () => {
    cy.notesNumber(initial_notes_number);
  })
```

- The `.notesNumber()` Command:

```
  // notesNumber() defintion (in support/Commands.js)
Cypress.Commands.add("notesNumber", (expected) => {
  cy.get(".notes-item").should("have.length.at.least", expected);
})
```

4. Add notes

- Here we try to added some notes with a random title and text (*), and confirm that they've been added successfully.

> (*) : to generate random titles and texts, we use **Chance.js**

```
it("Add a new Note", () => {
    // a loop to add 3 Notes
    let i = 0;
    while (i < 3) {
      type = types[i % 2];
      favorite = (i % 2) ? true : false;
      cy.addNote(chance.name(), chance.sentence({words: (Math.random() * 10
| 0) + 1}), favorite, type);
      initial_notes += 1;
      cy.notesNumber(initial_notes);
      i++;
    }
```

- The `.addNote()` Command:

```
//  Add a note
Cypress.Commands.add("addNote", (title, text, favorite, type) => {
  cy.get(".btn").click(); // get the "Add" button && click it
  cy.get("input[name='title']").type(title); // Add the title
  cy.get("textarea[name=text]").type(text); // Add the text

  if (favorite)
    cy.get("input[name='favorite']").click(); // Add to favorite

  cy.get(`input#${type}`).click(); // Set the type
  cy.get("input[name='submit']").click(); // Save the Note

  //   close the form
  cy.get("button#add").click();
})
```

5. Delete note

- To delete a note, we define a `.deleteNote()` command that take the index of the note we want to delete as well as the number of notes in order to decriment it.

```
  //  delete a note
  it("Delete a Note", () => {
    cy.pause();
    cy.deleteNote(4, initial_notes);
  })
```

- The `.deleteNote()` command:

```
Cypress.Commands.add("deleteNote", (index, nNotes) => {
  cy.get(".remove").then(
    e => {
    e[index].click();
    nNotes--;
  });
})
```

6. Change the favorite of the 1st note

- A sample test where we try to change the favorite property for the first note.

```
// change the fovorite of a note
Cypress.Commands.add("changeFavorite", () => {
  cy.get(".notes-item").first().dblclick()
})
```

7. Searching

- In the Search test, we create a `.search()` command that takes a text and a number as arguments, and compare the results of the search with the number that we expected to have.

```
  //  Searching.....
  describe("Search testing: Search for notes", () => {

    it("Where title: Note", () => {
      cy.search("note", 2);
    })

    it("Where title contains 'e' ", () => {
      cy.search("e", 4);
    })

    it("Where title contains 'p' ", () => {
      cy.search("p", 2);
    })
```

```
    it("Where title: random text ", () => {
      cy.get("#search")
      .type(
        chance.string({length: 6, alpha: true, numeric: true
      }), 0);
      // cuz no notes will match that string
      expect(0).to.eq(0);
    })
  })
```

```
//  Search testing...
Cypress.Commands.add("search", (text, expected) => {
  cy.get("#search").type(text);
  cy.get(".notes-item").then(e => {
    expect(e.length).to.be.at.least(expected);
  })
})
```

- Some search tests results:

Notes Application

Add

iYhuB

Copyright © 2021
About

Notes Application

Add

p

Project
The project is about to be finished by tomorrow
public
Date: 8/2/2021, 10:26:20 PM

Alphabets
English alphabets : A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z
public
Date: 8/2/2021, 10:31:23 PM

Copyright © 2021
About

Notes Application

Add

note

Note 1
This is the 1st note
private
Date: 8/2/2021, 10:24:56 PM

Note 2
Yeah!!! it's the 2nd note
public
Date: 8/2/2021, 10:25:35 PM

Copyright © 2021
About