**Stackoverflow application**

**Architecture of front-end development**

The front-end part of the application is written in Angular. Angular is a popular front-end framework for building web applications. The architecture of Angular front-end consists of several components, modules, services, and directives, which work together to create dynamic, responsive, and interactive web applications.

The key features of Angular architecture:

* Modules: Angular applications are modular, meaning they are composed of multiple small, self-contained modules that can be easily combined to create a larger application. Each module is a container for a set of components, directives, services, and other related code. An Angular application can be thought of as a puzzle where each piece (or each module) is needed to be able to see the full picture.
* Components: Components are the the most basic UI building block of an Angular app. Each component is responsible for managing a specific part of the user interface, and can contain HTML templates, styles, and logic for that part of the interface.
* Services: Angular services are objects that get instantiated just once during the lifetime of an application. Services are used to provide functionality that can be shared across multiple components and they can handle data retrieval, caching, authentication, and other common tasks.
* Directives: Directives are used to extend the functionality of HTML elements. They can be used to add behaviors, modify the appearance of elements, or create reusable UI components.
* Templates: Angular templates are used to define the structure and content of the user interface. Templates are written in HTML and can be combined with Angular directives and expressions to create dynamic and interactive user interfaces.
* Dependency Injection: Angular uses a powerful dependency injection system to manage the creation and sharing of objects across components and services. This makes it easy to write modular, testable code.

**Routing**

Angular routing is a powerful feature that enables developers to create single-page applications with multiple views. With routing, developers can define routes that map to specific components, and specify the behavior of the application when a user navigates to a particular route. This allows users to navigate within the application without having to reload the entire page.

To enable routing in an Angular application, we use the built-in RouterModule module, which provides the necessary tools for defining and configuring routes. A route definition consists of a path, which is the URL path that maps to the component, and a component, which should be displayed when the user navigates to that path.

Angular also supports nested routes, which means that a component can have its own child routes. Developers can also use route parameters to pass data to a component via the URL path.

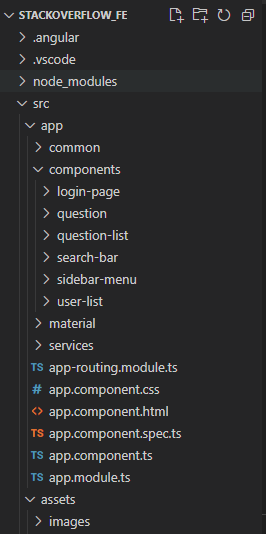
To create links that navigate to different routes within the application, we can use the Router Link directive in the HTML template. Additionally, Angular provides route guards, which are used to protect routes and determine if a user can access them based on conditions like login status or permissions.

**My application**

For my application I have mostly used Angular Material Components that I have combined to form my pages. I have created a basic login page, with a form for username and password of the user. When the user logs in, it is redirected to the main page of the application that contains a top search bar (that is used to search questions) and a side menu with the options for “users” and “questions”. Here, I have used the nested routes because when the user selects an option, the side menu and the search bar remain the same and only the part where I display the information is rendered.

The content that corresponds to the “questions” menu is formed by a filter bar (where the user can filter questions by tag, users or to show only his questions) and a list of cards that have basic information about the questions, like title, tags, a little text and score. The user can also publish a question by clicking a button that opens a form with all the required fields. Clicking a question will trigger a route change and will change the content that is displayed. Now, the user can see the full question, with a bigger photo and if he is the author, he is able to edit/delete the question. Below the question is a list of answers and by pressing a button, a form for inserting a new answer will be displayed. Only the users that own an asnwer can edit or delete it.

The ”users” option from the menu only displays the registered users from the system. Only the moderator can ban certain users.



These are the current packages in my application:

- common: contains only simple classes representing the dto’s that we are working with to display the objects retrieved from the back-end

- components: angular material components adapted with the help of css to form the pagesd

- material: composed of one module that has all the imports for the angular material components; the module is imported into every component to prevent duplicate code (like importing the same angular material component into multiple components from the app)

- services: all the services used for making the back-end requests

- assets: the “database” for the used images