Prodyna Coding Challenge Readme

1. Installation guide

Application can be downloaded from git repository. It is required that user have installed npm with Node.js and Angular CLI:

<https://github.com/milosvejnovic83/prodyna-test>

<https://www.npmjs.com/get-npm>

<https://cli.angular.io/>

After application is dowloaded or cloned and required applications are installed to local machine user should go to project directory and run „npm install“ in command prompt in order to get all required node\_modules.

Once that application is installed it can be runed by „ng s“ in command prompt and it can be accessed from the browser on the following link:

<http://localhost:4200/>

2. Problem and solution

Application should consume fake API in order to get list of posts. Titles should be displayed in a list and body of post body for each title should be displayed on click event. Filtering of titles is implemented based on user input. New form should be available in order to create a new post and also available posts can be deleted. Application should implement load spinner when page is loading for any service that is called and lazy load infinite scroll so that more posts are loaded while user scrolls down.

Solution was developed using Angular 8 framework. All pages that are available to user are divided in to separate components, service was created for required CRUD operations and filtering was implemented by using of pipes.

3. Technical choices

For the purpose of developing of the application Angular 8 was used. This framework supports different options for consuming API services, looping through results and implementation of two-way data binding of components properties with html so it is a good choice for developing frontend applications. Code is clean well divided in to logical parts so it is easy to develop, update and maintain.

API was consumed by application by CRUD service that was created. It uses observables to communicate with backend, respond from the server is processed inside component by putting results inside array and then displayed by html.

For the purpose of displaying of posts, \*ngFor directive is used inside html. It loops through array that was created by component from server respond and in each loop one post is displayed as result.

Posts are filtered by using of pipes. Inside of the pipe logic is created that takes users input and displays only posts that are matching this input. Pipe is used along with \*ngFor directive in order to loop just through the posts that matches users input in search box.

For creating of new posts new component is added with appropriate form. It takes post title and body from users input and send a post request with the id of user, title and body of the post by using CRUD service which is created. As result new post is created with unique id, user id, title and post.

For deleting purposes delete button is added for each post that is displayed. It takes post id and pass it to the observable in CRUD service which is responsible for delete request.

For both create and update functionality after successful creating or deleting of the post list is displayed that is corresponding to action that is triggered. Since API is fake results are displayed inside console and post is not really created or deleted.

Loading spinner and lazy loading infinite scroll were implemented by adding of NgxSpinner and ngx-infinite-scroll plugins. They are declared in app module and implemented inside application components and htmls.

Loading spinner was added before every CRUD service is called and it is hidden when call is finished and it is placed inside html by using of spinner selector. In the case of any delay of respond from the server spinner is displayed.

Lazy load infinite scroll is added inside html for the element that is displaying list of posts and it displays a message in the console when it is triggered, for the testing purposes.

4. Tradeoffs, leftovers and things that can be improved

Since application is using fake API it is not possible to really create or delete posts and because of that results for those actions are displayed in console.

When user creates a new post user id is hardcoded and post is really never created it always gets same unique id with the value of 101. Selection of user that created a new post might be added and different information for a new post might be created rather than displaying it in console.

Search is using pipe that is filtering posts by titles. Search that also filter body of the post might be added.

Application is responsive for the screen sizes bigger than 900px. Responsive design might be added for smaller screen sizes and mobile devices.

5. Link to developer code

All of the code that I have created for the purpose of learning and for different tests can be found in my git repository:

<https://github.com/milosvejnovic83>

6. Link to developer LinkedIn profile

My LinkedIn profile is available on following address:

<https://www.linkedin.com/in/milos-vejnovic83/>