Building up experience

…

Start by preparing workspace

1. Clone https://github.com/minijus/itacademyblog.git (same repo as day-5)
2. Checkout branch **day-6**
3. Run ***npm ci***
4. Run in two separate terminals:
   a. *npm run start:server*
   b. *npm run start:frontend*

New Resources

Once you have server running there 4 new types of resources available:

- /api/users
- /api/authors
- /api/categories
- /api/posts (old)
- /api/comments

Overview: http://localhost:3000/

Agenda

- Task #1: Use categories for menu
- Task #2: Implement category page
- Task #3: Pages for author
- Task #4: Update recent posts component
- Task #5: Implement most viewed posts component
- Extra tasks

Task 1

Task 1 - Navbar component

1. Generate Category interface inside shared directory
   ng g i shared/category
2. Define Category interface properties (see mocks/data.json for data objects)
3. Generate CategoriesService inside services directory
   ng g s services/categories
4. Implement getCategories method to receive all categories from /api/categories
5. Generate navbar component
   ng g c navbar

Task 1 - Navbar component (continued)

6. Move html nav-bar container from AppComponent template to NavbarComponent template
7. Use NavbarComponent inside AppComponent template
8. Add two @Input to NavbarComponent:
   a. menuItems
   b. categories
9. In AppComponent define two properties (menuItems and categories$):
   a. menuItems - array of objects (i.e. {path: ‘/home’, title: ‘Home’})
   b. categories$ - use a getCategories to assign Observable
10. Bind menuItems and categories$ to NavbarComponent

Task 1 - Navbar component (continued)

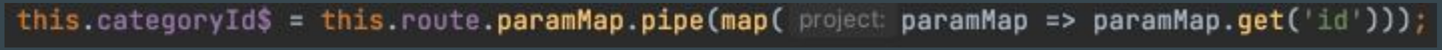
11. Update Navbar component to:
    a. Use menuItems
    b. Use categories
12. Bonus: add missing bootstrap hamburger menu, so when browser window is narrow navigation would be accessible through it

Solution: https://github.com/minijus/itacademyblog/compare/day-6...day-6-task-1

Task 2

Task 2 - CategoryPage component

1. Make sure that after finishing Task 1 your application navigates to category pages from Navbar. Right now category pages are 404.
2. Generate CategoryPage component
   ng g c categoryPage
3. Define a route with newly generated CategoryPage component
4. Create a getCategory method that accepts id parameter and return Observable of single category (httpClient request to /api/categories/:id)
5. Use ActivatedRoute inside CategoryPage and pipe paramMap to listen for id changed:



```
this.categoryId$ = this.route.paramMap.pipe(map( project: paramMap => paramMap.get('id')));
```

Task 2 - CategoryPage component (continued)

6. When id changes make a new getCategory(id) request to get detail of new category
7. In CategoryPage template add:
   a. Category title
   b. Category image
   c. Category description
   d. List of category tags
8. Define new variable in CategoryPage component posts$
9. Implement new method getCategoryPosts(id) in CategoriesService. The method makes Http request to /api/categories/:id/posts
10. Render list of post titles in CategoryPageComponent template. Use:
    a. *ngIf="posts$ | async as posts"
    b. Later *ngFor

Task 2 - CategoryPage component (continued)

11. Make sure every post title (inside CategoryPage) navigates to that post when clicked
12. Generate PostList component
    ng g c postList
    a. Add @Input() posts
    b. Move the post lists part from CategoryPage template to PostList template
    c. Use PostList component with posts binding inside CategoryPage component

Solution: https://github.com/minijus/itacademyblog/compare/day-6-task-1...day-6-task-2

Task 3

Task 3 - Pages for Authors

1. Generate components for AuthorDetails, AuthorPage and AuthorsPage
2. Generate interface inside shared directory for Author
3. Generate AuthorsService inside services directory
4. Implement tree methods in AuthorsService:
   a. getAuthors() calls /api/authors
   b. getAuthor(id) calls /api/authors/:id
   c. getAuthorPosts(id) /api/authors/:id/posts
5. Define two routes:
   a. /authors to AuthorsPage component
   b. /authors/:id to AuthorPage

Task 3 - Pages for Authors (continued)

6. Add menu item Author to link /authors
7. Add @Input() author to AuthorDetails
8. In AuthorDetails component render:
    a. Image
    b. firstName and lastName with routerLink to /author/:id
    c. About text
    d. Company
    e. Email
9. In AuthorsPage use AuthorsService to get all authors and render the list using AuthorDetails component
10. In AuthorPage use ActivatedRoute to get 'id' and get Author details and posts from AuthorsService. Use AuthorDetails and PostList component in template.

Solution: https://github.com/minijus/itacademyblog/compare/day-6-task-2...day-6-task-3

Task 4

Task 4 - Fix RecentPostsComponent

1. Notice Post now contains new property 'created'
2. Create getRecentPosts(limit) in PostsService to return most recent posts
    a. Check documentation for json-server
    b. User route query parameters _sort, _order, _start and _limit
3. Use getRecentPosts inside RecentPostsComponent
4. Use PostListComponent inside RecentPostsComponent template

Solution: https://github.com/minijus/itacademyblog/compare/day-6-task-3...day-6-task-4

Task 5

Task 5 - Implement MostViewedPosts component

1. Notice Post now contains new property views
2. Create getMostViewedPosts(limit) in PostsService to return most viewed posts
    a. Check documentation for json-server
    b. User route query parameters _sort, _order, _start and _limit
3. Use getMostViewedPosts inside newly generated MostViewedPosts component
4. Use PostListComponent inside MostViewedPosts template
5. Use MostViewedPosts component inside AppComponet template

Solution: https://github.com/minijus/itacademyblog/compare/day-6-task-4...day-6-task-5

Extra

Extra tasks

1. List all comments under the post
2. Add a form to submit new comment under the post
3. Add UsersPage, UserPage and UserDetails component
    a. List comments made by User under UserPage
    b. Could this be abstracted with Author?
4. Add a form to create new User (registration)
5. Add links (buttons) that send ‘delete’ Http request (i.e. httpClient.delete(‘/api/comments/:id’)) to delete comments, posts, users or authors.