UNIT 2

Angular

Exercise 4

Client-side Web Development
2nd course – DAW
IES San Vicente 2025/2026
Arturo Bernal / Rosa Medina

Index

Exercise

Update the previous exercise (3) with the following changes:

- We are going to implement a "property detail" page, so create the corresponding component: **property-detail**.
- Also, create a **login-page** component for the login page. But, in this exercise, this page **will not have any functionality** (for now). Just redirect to the "**properties**" page when the login button is clicked.

Routes

Create these routes in your app. Don't forget to put the **router-outlet** component in the app component template.

Order is important!: properties/add must go **before** properties/:id, because :id is a variable and if it goes first, It would load the detail page and use 'add' as the id.

- **login** → LoginPage (Create this component)
- **properties** → PropertiesPage
- **properties/add** → PropertyForm
- **properties/:id** → PropertyDetail (Create this component)
- **Default and other** → Redirect to '/login'

Create the **top-menu** component and put the page **header** there (the one that's in the app component). Import and use that component inside app component. Update the navbar (you can reuse the one in the unit 1 project), removing the ids (not necessary anymore) and adding the following links:

- **Home** → '/properties'
- **New Property** → '/properties/add'

Other changes you must apply:

- Copy the **icons** folder from the unit 1 project into the public folder.
- Make the "**New Property**" link visible by default (remove "hidden" class).
- Use the class '**font-semibold**' to highlight the link of the current page ([routerActive] directive).
- Change the <main> element classes inside the app component's template:

```
<main class="container mx-auto px-4 py-8 mt-16 grow flex flex-col">
  <router-outlet></router-outlet>
</main>
```

- When copying the HTML from project 1. In every page (component) created copy what's inside the main element (it's already in the app component)

- To center the login, just copy this into the login-page @Component decorator:

```
host: {
  class: 'grow flex items-center justify-center',
},
```

- You can do the same in the PropertiesCard component. Copy the main div classes into a host property (applied directly to the properties-card element) and you can remove that div (it will also work better).

Services (PropertiesService)

After adding or deleting a property, call the **reload** method on the **propertiesResource** object. Add a **tap** function in the observable pipe method to do this after the response.

Create a method to get a resource for getting a property by id:

- **getPropertyResource(id: Signal<number>)** → Returns a httpResource for getting the property using the signal received as the id.

Pages

- '**login**' → This page will contain the login form (unit 1 project), but **it won't do anything for now**. Just put a link on the button or use the ngSubmit event to go to the 'properties' page without calling any service.
- '**properties**' → Get and show all properties from the server. Property's image and title will be links to the details page.
  - Remove the **property-form** element from this page's template
- '**properties/add**' → Contains only the new property form. When you add a new property successfully do this:
  - **Redirect to the new property detail page** instead of resetting the form.
  - Change the boolean that indicates you have created a property, so the CanDeactivate guard lets you leave the page without asking.
- **'properties/:id'** → For now, just show the card (property-card) in this component (reuse the component). In the future we'll change this and use the same HTML as in the first project.
  - Deleting the property in this page will redirect to the /**properties** page instead of removing the card from the DOM.

Page Title

Change the title of the app depending on the page you are on ("Properties page", "New property", etc.). In the property detail page, use the property title as the page title.

Guards

Create 2 guard functions for the routes:

- **numericIdGuard** (CanActivateFn) → Use it for the property detail's page. Check if the id in the url is numeric, or redirect to the properties page.
- **leavePageGuard** (CanDeactivateFn) → Ask the user if he/she wants to leave the page (use a confirm dialog). Use it for the 'properties/add' route. Also, create a boolean in the component (false) and set it to true when you add a property, and **only** ask the user if the property has not been inserted.

Lazy loading

Create the **auth**, **properties** and **shared** folders in your project (inside src/app). Divide those routes of your application and **lazy load** them. Put also the components, pipes, etc related to properties or auth inside the corresponding folders. In the **shared** directory only put global code, not strictly related to properties or auth (like interceptors, for example).

- **auth/auth.routes.ts** → 'login'
- **properties/properties.routes.ts** → '', 'add', ':id'
- **app.routes.ts** → routes with prefix 'auth' should lazy load **auth/routes.ts**, and routes with prefix 'properties' should lazy load **properties/routes.ts** (use **loadChildren**).

Components that represent each page should also be lazy loaded using **loadComponent**.

Use the **preloading strategy** for lazy loading.

Don't forget to delete the .angular folder alongside the node_modules folder before uploading the exercise!