UNIT 2

Angular

Exercise week 10

Client-side Web Development
2nd course – DAW
IES San Vicente 2024/2025
Author: Arturo Bernal Mayordomo

Index

Exercise

Update the exercise from last week (week 9) with the following changes:

- We are going to implement a "event detail" page, so create the corresponding component: **event-detail**.
- Also, create a login component for the "**login**" page. But, in this exercise, this page **will not have any functionality** (for now). Just redirect to the "**events**" 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!: events/add must go **before** events/:id, because :id is a variable and if it goes first, It will load the detail page and use 'add' as the id.

- **login** → LoginPageComponent (Create this component)
- **events** → EventsPageComponent
- **events/add** → EventFormComponent
- **events/:id** → EventDetailComponent (Create this component)
- **Default and other** → Redirect to '/login'

Create the **top-menu** component and put the main **navbar** 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 and modify it), with the following links:

- **Home** → '/events'
- **New event** → '/events/add'

Pages

- '**login**' → This page will contain the login form, but **it won't do anything for now**. Just put a link on the button or use the ngSubmit event to go to the 'events' page without calling any service.
- '**events**' → Get and show all events from the server. Event's image and title will be links to the details page.
  - Remove the **event-form** element from this page's template
  - Change the Order by date and price links to **buttons**, or they will redirect to the root page (a.nav-link → **button.nav-link**).
  - Add some CSS to remove the underline from the title

```
.card-title a {
  text-decoration: none;
}
```

- ‘**events/add**’ → Contains only the new event form. When you add a new event successfully, redirect to ‘/events’ instead of resetting the form. Don’t forget to change the boolean that indicates you have created a event, so the CanDeactivate guard lets you leave the page without asking.
- **'events/:id'** → This page will only show the card (**event-card**) of the event (reuse the component), and a button to go back to the events page. Also create a button to go back to the events page navigating by code:

```
<div class="mt-4">
   … (event card here) ...
</div>
<div class="mt-4 mb-4">
  <button class="btn btn-success" (click)="goBack()">Go back</button>
</div>
```

Deleting the event in this page will redirect to the /**events** page instead of removing the card from the DOM.

Page Title

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

Guards

Create 2 guard functions for the routes:

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

Resolvers

Create a resolver function called **eventResolver** (ResolveFn<Event>) and apply it to the ‘**events/:id**’ route. It will get the event data before loading the page. If there’s an error, redirect to ‘/**events**’ page.

Lazy loading

Create the **auth**, **events** 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 events or auth inside the corresponding folders. In the **shared** directory only put global code, not strictly related to events or auth (like interceptors, for example).

- **auth/auth.routes.ts** → 'login'
- **events/event.routes.ts** → '', 'add', ':id'
- **app.routes.ts** → routes with prefix 'auth' should lazy load **auth/routes.ts**, and

routes with prefix 'events' should lazy load **events/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!