UNIT 2

Angular

Exercise 3

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

Index

Exercise

Update the previous exercise with the following changes:

- We'll use the same web services from part 4 (Unit 1) exercise: (**https://github.com/arturober/inmosanvi-services-lite**).

Directives

Create the **encodeBase64** directive we've seen in class and use it in the **property-form** component (file input). Update it to emit an empty string when there's no file. Delete the code that's not needed anymore in the component.

Interfaces

Create the necessary interfaces for the server responses: Province, Town, ProvincesResponse, TownsResponse, PropertiesResponse, SinglePropertyResponse, …

Also change the Property interface to reflect what the services return. Create a **PropertyInsert** interface with all the attributes you'll need for the new property form (check the Postman collection), and a **Property** interface that inherits from it with the rest of the attributes. Go to the Components sections to see the changes.

```
export interface Property extends Omit<PropertyInsert, 'townId'> { ... }
```

Services

Create these services to make requests to the server:

- **ProvincesService** → This service will manage requests related to provinces and towns. Implement the following attributes/methods:
  - **provincesResource** → Readonly public attribute. **httpResource** to get all provinces.
  - **getTownsResource(**
    provinceId: Signal<number>
    ): HttpResourceRef<TownsResponse> → Returns an **httpResource** that fetches the towns based on the province id. Set a **default value** in the httpResource simulating a response object with an empty towns array.
- **PropertiesService** → This service will manage everything related to properties and have at least the following methods:
  - **propertiesResource** → Readonly public attribute. **httpResource** to get all properties.
  - **addProperty(property: PropertyInsert): Observable<Property>**
  - **deleteProperty(id: number): Observable<void>**

Components

Components like properties-page, property-form and property-card will use the **PropertiesService** to get data from the server, insert new properties and delete properties (at the server). property-form will also use **ProvincesService**. Wait for the server's response (**subscribe**) before changing anything locally.

Cancel subscriptions from any observable using **takeUntilDestroyed**. Also, rembember you won't need the property's id in the property-form component anymore.

Property-form component

The property-form component will now have a **PropertyInsert** object. Keep the province in a separate signal (it won't be in the property object anymore). Use that signal to get the resource from PropertiesService to load the towns when the province changes.

Also, you must add the description field to the HTML (both in this form and in the properties-page component's HTML). See the exercise part 4 from previous unit to copy the necessary HTML.

Set the default option value in the <select> elements to "0" instead of "". Initialize the province signal to 0.

Use an **effect** function to change the newProperty.townId to 0 whenever the province changes. Just read the province signal inside the funcion to create a dependence (don't do anything with it), and set the townId to 0.

Interceptors

Create an interceptor called **baseUrlInterceptor** that adds the server prefix (http://localhost:3000 in our case) to the url of any http request.

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

You must use the modern Angular syntax we've seen in class!. Any outdated syntax will make that component/directive/service not valid for this exercise grading.

HttpResource will store the full server's response object. Create a linked signal derived from the resource, to store the array of properties so it becomes easier to access the array and update it.