# Angular Workshop – Hotels System

Download the web server from the course, install all its dependencies and run the server. It is listening on port 5000.

## Create a Angular application

Create a Angular application and prepare the initial project structure. Install Flux and prepare its components.

## Add authentication

Add authentication and make sure the register, login and logout functionalities work correctly. To register a user, you need to send a POST request to the server on ‘/auth/signup’ with ‘name’, ‘email’ and ‘password’ data (sent as JSON). To login a user, you need to send a POST request to the server on ‘/auth/login’ with ‘name’ and ‘password’ data (sent as JSON). You need to save the user token in your application state. Make sure you validate everything on the client application.

## Add creating of hotels

Add a form to create hotels in the system. Each hotel has ‘name’ as string, ‘location’ as string, ‘desription’ as string, ‘numberOfRooms’ as number, ‘image’ as string URL and optional ‘parkingSlots’ as number. Make sure you validate everything on the client application. The data must be sent as POST request to the server on ‘/hotels/create/’. This route is only for authenticated users so you need to send a header with `Authorization` name and value `bearer {*token*}` in order to pass the authentication checks.

## Add listing of hotels

Add a page where all hotels are listed. Since the data from the server comes in pages of 10 hotels, your page must have buttons to navigate between the pages. Try to validate the buttons when the page is no longer valid. You need to make a GET request to ‘/hotels/all’ to receive an array of hotels data. Optionally, you can pass a query string parameter ‘page’ to request more data, for example ‘/hotels/all?page=2’. Link each hotel to its details page. Don’t show every piece of information about the hotel on this page. Leave something for the details page. You may add this functionality on the home page.

## Add hotel details

Add a page where all hotel details are shown. You need to make a GET request to ‘/hotels/details/{*id*}’ to retrieve information about the hotel with the provided id. This route is only for authenticated users so you need to send a header with `Authorization` name and value `bearer {*token*}` in order to pass the authentication checks. Make sure your Angular application redirects to the login page, if the user tries to open the hotel details page and she’s not logged in.

## Add option to add reviews

On the hotel details page add an option for the user to put a rating from 1 to 5 and optionally comment for the current hotel. Each review has ‘rating’ as number between 1 and 5 and ‘comment’ as string data (sent as JSON). You need to make a POST request to ‘/hotels/details/{*id*}/reviews/create’ in order to create a review. The id is the hotel id the user is reviewing. Make sure you validate everything on the client application. This route is only for authenticated users so you need to send a header with `Authorization` name and value `bearer {*token*}` in order to pass the authentication checks.

## Add listing of all reviews

On the hotel details page, you need to list all reviews for the current hotel. You need to make a GET request to ‘/hotels/details/{*id*}/reviews’ to receive an array of review data. This route is only for authenticated users so you need to send a header with `Authorization` name and value `bearer {*token*}` in order to pass the authentication checks.