PEC 7

Frameworks: Routing en Angular

Universitat Oberta
de Catalunya

Información relevante:

- Fecha límite de entrega: 19 de enero.
- Peso en la nota de FC: 15%.

Contenido

Información docente

Presentación

Esta práctica se centra en conocer la creación de rutas en Angular, para ello se hará uso del módulo `Router` del `core` de Angular. Además, se protegerán rutas en función de si el usuario está logueado o no en el sistema. Finalmente, se mantendrá la sesión del usuario basado en tokens de autenticación.

Objetivos

Los objetivos que se desean lograr con el desarrollo de esta PEC son:

- Comprender el funcionamiento del **router en Angular**.
- Implementar un **sistema de rutas**.
- Construir una aplicación compuesta por **varios componentes y servicios repasando conceptos** de prácticas anteriores.
- **Proteger rutas** utilizando **guardas**.
- Conocer y hacer uso de **Lazy-Loading y estructuración de módulos.**
- **Mantener la sesión** de un usuario basado en **tokens de autenticación**.

Enunciado

Esta PEC **contiene 3 ejercicios evaluables**. Debéis entregar vuestra solución de los 3 ejercicios evaluables (ver el último apartado).

Debido a que las actividades están encadenadas (i.e. para hacer una se debe haber comprendido la anterior), **es altamente recomendable hacer las tareas y ejercicios en el orden en que aparecen en este enunciado**.

Antes de continuar debes:

Haber leído los recursos teóricos disponibles en el apartado “Contenidos y recursos” del aula de esta PEC

Crea una carpeta PEC7 e inicializa git en esa ruta. Al igual que en las anteriores PEC, debes crear un fichero README.md en la raíz de la carpeta que contenga que contendrá al menos esta información

• Login UOC.

• Nombre y apellidos del alumno.

• Breve descripción de lo realizado en esta PEC, dificultades, mejoras realizadas, si hay que tener algo en cuenta a la hora de corregir/ejecutar la practica o cualquier aspecto que queráis destacar.

Ejercicio 1 – Preguntas teóricas (3 puntos)

Crea una carpeta `PEC7_Ej1`, dentro crea un fichero Markdown que tenga como nombre PEC7_Ej1_respuestas_teoria.md y **responde** a cada una de las siguientes preguntas:

a) ¿Qué es y cómo funciona el elemento `<RouterOutlet>`?

b) ¿Para qué se utilizan las directivas `routerLink` y `routerLinkActive`? ¿Existen más directivas relacionadas con el `router`?

c) ¿Qué diferencias hay entre los servicios `Router` y `ActivatedRoute`? ¿Qué funcionalidades tiene cada uno de estos servicios? Describe algunos de los métodos más importantes por los que están compuestos.

d) ¿Qué son las `Route Guards`? ¿Cómo se usan las guardas en Angular? Describe todas las guardas que existen en Angular (consulta para ello la documentación oficial de Angular)

e) ¿Qué es la carga `Lazy` de los módulos de Angular? ¿Cómo se configura en Angular la carga `Lazy`? (https://angular.io/guide/lazy-loading-ngmodules)

f) ¿Qué es/para qué son útiles los `middlewares` en el contexto de `node.js`? ¿Dónde estás usando `middlewares` en nuestra aplicación?

Durante el desarrollo del ejercicio ejecuta los comandos `git` necesarios para añadir esta nueva carpeta, ficheros y los cambios que realices en ellos al repositorio local `git` de la PEC.

Ejercicio 2 – Práctica – Routing (5.5 puntos)

Crea una carpeta PEC7_Ej2 donde partirás del último ejercicio realizado en la PEC anterior (o de la solución publicada) y desarrollarás sobre el proyecto de vinoteca.

Instala y ejecuta el nuevo servidor que se te ha proporcionado (server-vinoteca) ejecutando lo siguiente:

- npm i
- npm start

Esto hará que arranque un servidor local de node.js, el cual estará trabajando con vinos (similar al que has visto en los apuntes sobre stocks). Este servidor expone las siguientes APIs, en el puerto 3000 (http://localhost:3000/api/wine):

- `GET` a `/api/wine` para obtener una lista de vinos. Ésta puede tener un parámetro opcional de consulta q, el cual es el nombre del vino a buscar.
- `POST` a `/api/wine` con la información de un vino en el cuerpo para crear un vino en el servidor (en nuestro caso será en memoria, reiniciando el servidor se perderán todos los datos creados).
- `PATCH` a `api/wine/:id` con el ID del vino en la URL y un campo `changeInQuantity` en el cuerpo cambiará la cantidad en el carrito de vinos por la cantidad pasada como parámetro.

Además tendrás dos nuevos ENDPOINT que corresponden a la gestión de usuarios (falsa) para poder realizar el registro y autenticación de usuarios en nuestro sistema:

- `POST` a `/user/login` recibe el username y password del usuario y comprueba si el usuario está en el sistema y la contraseña es la correcta.

- `POST a /user/register` recibe el username y comprueba que no está repetido, se asigna la contraseña “SECRET” a todos los usuarios.

Puedes comprobar el correcto funcionamiento de la API haciendo uso de la herramienta Postman o Insomnia.

> **NOTA**: Si necesitas ayuda sobre el funcionamiento de POSTMAN puedes consultar el siguiente recurso:
>
> https://learning.oreilly.com/library/view/mastering-spring-5/9781789615692/f2700159-ee0e-44e0-ac12-e1ff130d3f4a.xhtml

Una vez instalado y comprobado el funcionamiento del servidor local de node.js, realiza los siguientes apartados:

a) **Repasando todo lo aprendido (1.75 puntos)**

Sobre el proyecto angular debes realizar las siguientes tareas utilizando los conocimientos adquiridos a lo largo de la asignatura:

1. Crear los siguientes componentes:

- ***LoginComponent*** (Formulario reactivo). Este formulario está compuesto por dos inputs (username y password) y realizará la autenticación de los usuarios en el sistema.

- ***RegisterComponent*** (Formulario reactivo). *Este formulario está compuesto por dos inputs (username y password) y realiza el registro de los usuarios en el sistema.*

- ***WineDetailComponent***. Este componente mostrará los detalles de cada vino (añadir un nuevo campo descripción) y mostrar también con qué comida maridan (foodPairing), se accederá a este detalle haciendo click sobre la imagen de cada vino en el componente WineItemComponent. En la ruta se accederá a través de la variable “id” de cada vino (valor que debe ser único)

2. Crear el servicio UserService el cual hace llamadas HTTP para realizar la autenticación (login) y registro (register) de los usuarios

3. Crear el servicio UserStoreService el cual almacena si el usuario está autenticado en la aplicación o no, haciendo uso de un token de autenticación (será falso y no hay que hacer nada en el backend).

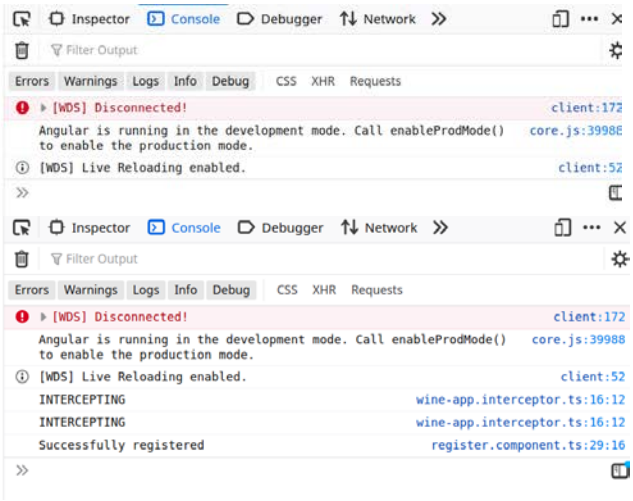
4. Crear un interceptor WineAppInterceptor que enviará el token de autenticación si este existe en cada una de las peticiones HTTP.

5. Registrar todos los elementos anteriormente citados en el módulo AppModule.

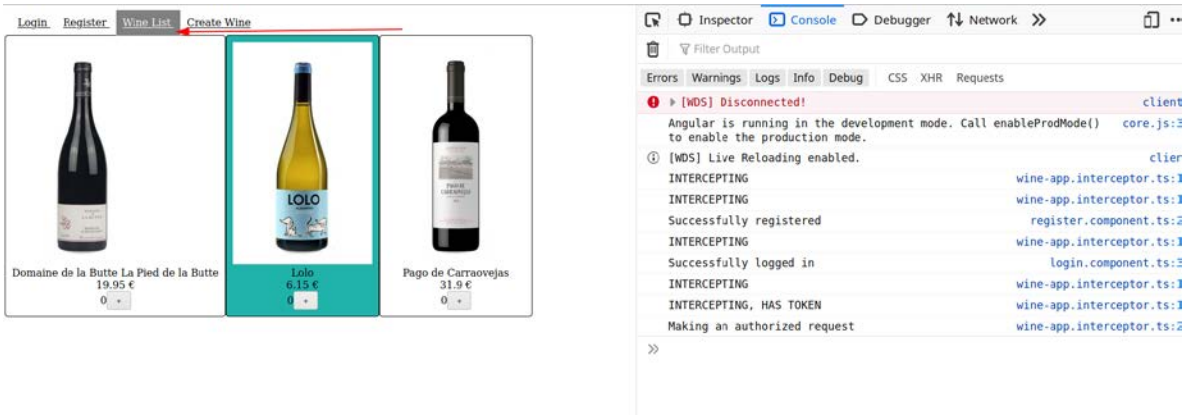
b) Configurar Routing y esquema de rutas en la aplicación (1 punto)

Crea un menú superior, y configura el módulo de routing de manera adecuada en nuestra aplicación para que las diferentes opciones de menú vayan a las siguientes rutas:

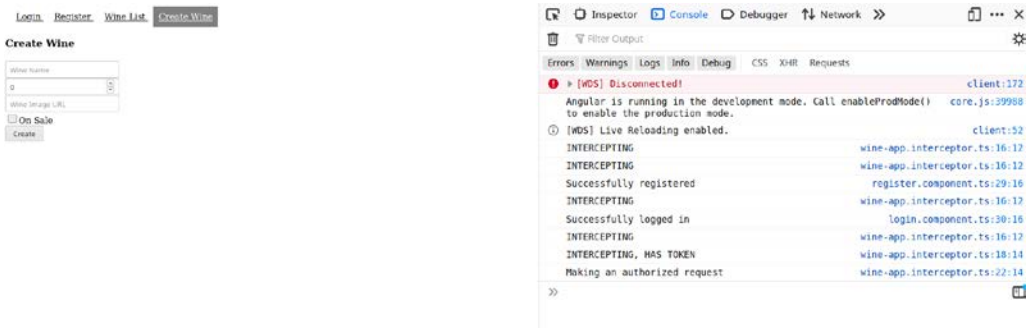
- La ruta por defecto será 'login'.
- path: 'login', component: LoginComponent
- path: 'register', component: RegisterComponent



- path: ‘wine/list’, component: WineListComponent



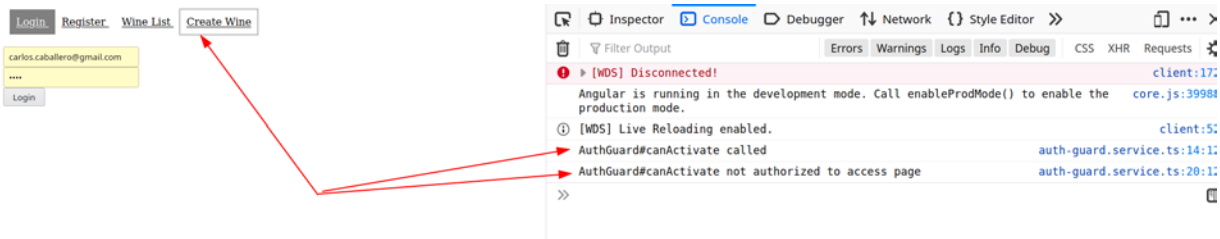
- path: ‘wine/create’, component: WineNewComponent



- path: ‘wine/:id’, component: WineDetailComponent

c) Protegiendo rutas (1.5 puntos)

- La ruta para crear un vino deberá estar protegida y accesible después de autenticarse (login).

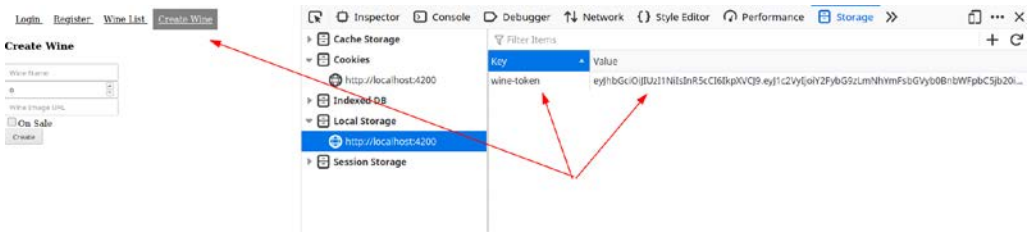


d) Remember login (1.25 punto)

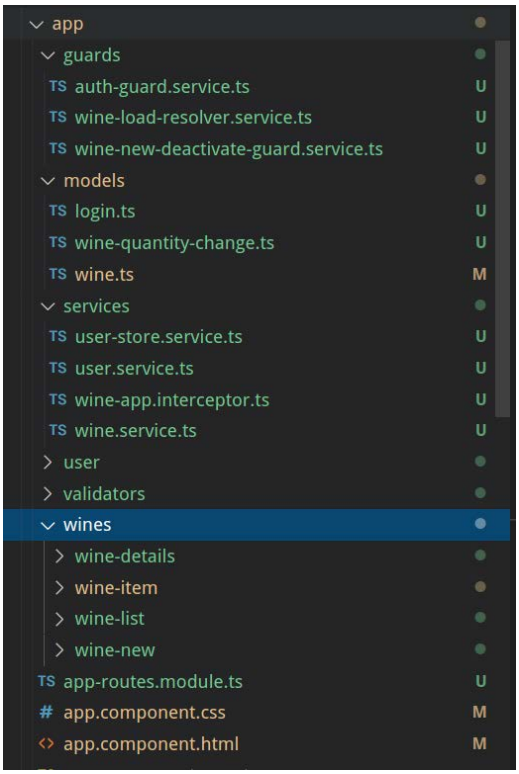
Adapta el código de manera que el flujo de autenticación del usuario recuerde si el usuario está autenticado en el sistema cuando refrescamos la página.

Es decir, si el usuario está autenticado se deberá almacenar el token de autenticación en el sistema de manera que podamos recuperarlo al refrescar la página. En caso de que el usuario esté autenticado no se mostrará la página de login pero en el caso de que no esté autenticado, se mostrará la página de autenticación (login).

Por ejemplo si ya nos hemos logado , al acceder a Create wine no se nos mostrara de nuevo el formulario de login.



Una vez hayamos concluido todos los apartados, la estructura de directorios de nuestro proyecto debería ser algo similar a esto:



Durante el desarrollo del ejercicio ejecuta los comandos `git` necesarios para añadir esta nueva carpeta, ficheros y los cambios que realices en ellos al repositorio local `git` de la PEC.

La carpeta **node_modules** y la carpeta **.angular** , no debe aparecer en el repositorio git local, ni tampoco entregarla en el zip final.

Ejercicio 3 – Práctica sobre Lazy-Loading (1.5 puntos)

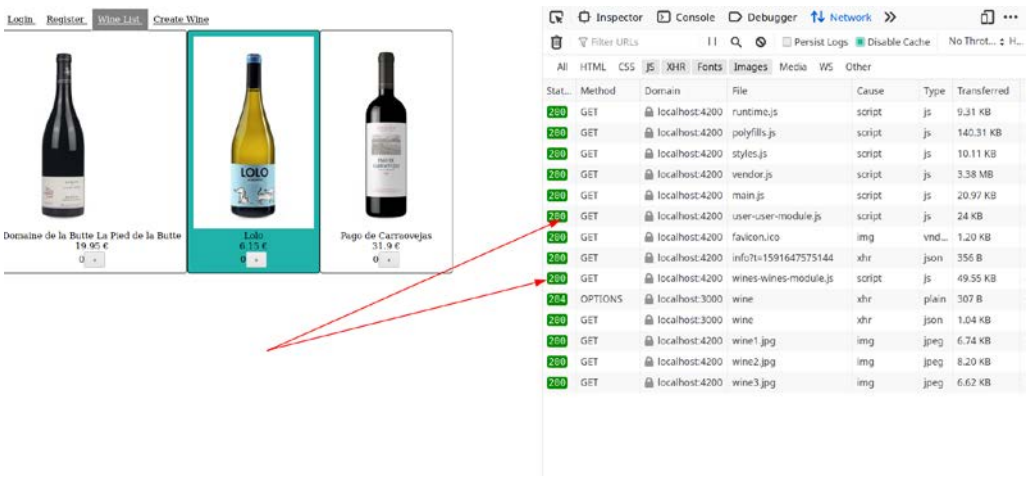
Partiendo del anterior ejercicio crea una nueva carpeta PEC7_Ej3 y realiza las modificaciones necesarias para disponer de dos módulos claramente diferenciados:

- User
- Wine

Los cuales se carguen de manera lazy en función de si se requiere un módulo u otro. La nueva estructura de directorios debe transformarse de manera que ahora tengamos un espacio de modelos, servicios, guardas que se puedan considerar shared y otros que sean específicos por cada uno de los módulos concretos.

Realiza los cambios en el código para disponer de Lazy-Loading.

En la carpeta PEC7_Ej3 incluir una captura de pantalla del inspector de elementos donde se visualice claramente que se están cargando los módulos de manera Lazy.



Durante el desarrollo del ejercicio ejecuta los comandos `git` necesarios para añadir esta nueva carpeta, ficheros y los cambios que realices en ellos al repositorio local `git` de la PEC.

La carpeta **node_modules** y la carpeta **.angular** , no debe aparecer en el repositorio git local, ni tampoco entregarla en el zip final.

Formato y fecha de entrega

Tienes que entregar un fichero *.zip, cuyo nombre tiene que seguir este patrón: `loginUOC_PEC7.zip`. Por ejemplo: `dgarciaso_PEC7.zip`. Este fichero comprimido tiene que incluir los siguientes elementos:

Una **carpeta con nombre PEC7**, y en su interior:

- Un **fichero `README.md`** en la raíz de la carpeta con la información indicada.
- La **carpeta oculta** `.git` con el contenido del repositorio local git.
- Una **carpeta `PEC7_Ej1`** con un fichero `markdown` **`PEC7_Ej1_respuestas_teoria.md`** para las respuestas del ejercicio 1.
- Una **carpeta `PEC7_Ej2`** con los ficheros resultado de haber realizado las tareas del ejercicio 2 **(Eliminar la carpeta node_modules y la carpeta .angular**)
- Una **carpeta `PEC7_Ej3`** con la captura de pantalla del inspector de elementos y los ficheros resultado de haber realizado las tareas del ejercicio 3 (**Eliminar la carpeta node_modules y la carpeta .angular**)

Penalizaciones

- Entrega en otro formato que no sea el especificado (ej. en `zip`): **-0.75 puntos**
- Comprimir archivos dentro del `zip`: -**1 puntos**.
- Para cada ejercicio/apartado donde no se respete la nomenclatura exacta de las carpetas o ficheros indicados (símbolos, minúsculas, mayúsculas, etc.): **-0.75 puntos.**
- La no entrega del repositório local `git` **-3 puntos**
- Por cada carpeta node_modules o .angular entregada **-0.75 puntos**

El último día para entregar esta PEC es el **19 de enero 2022** hasta las **23:59**. Cualquier PEC entregada más tarde será penalizada.