

PEC 6

Frameworks: Servicios en Angular

UOC

Universitat Oberta
de Catalunya

Información relevante:

- Fecha límite de entrega: 12 de enero.
- Peso en la nota de FC: 15%.



Contenido

Información docente	3
Presentación	3
Objetivos	3
Enunciado	4
Ejercicio 1 – Preguntas teóricas sobre servicios Angular (1.5 puntos)	4
Ejercicio 2 – Práctica – Servicios (3.5 puntos)	6
Ejercicio 3 – Preguntas teóricas sobre interceptores (1 punto)	8
Ejercicio 4 – Práctica – HttpClient (2.5 puntos)	9
Ejercicio 5 – Práctica – Pipes (1.5 puntos)	11
Formato y fecha de entrega	12

Información docente

Presentación

Esta práctica se centra en conocer la creación de servicios en Angular, se presentará la biblioteca RxJS que se encuentra en el corazón de Angular para la comunicación entre componentes. Finalmente, se hará uso del servicio HttpClient que permite conectar el frontend con el backend.

Objetivos

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

- Comprender el funcionamiento de los **servicios en Angular**.
- Implementar nuestro **primer servicio** con lógica desde los **componentes**.
- Implementar un servicio **que realiza peticiones a una API REST** haciendo uso del **HttpClient**.
- Conocer y hacer uso de **interceptores**.

Enunciado

Esta PEC **contiene 5 ejercicios evaluables**. Debéis entregar vuestra solución de los 5 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 el recurso teórico T01.PEC6_Teoria_2020.pdf disponible en el apartado “Contenidos y recursos” del aula de esta PEC.

Ejercicio 1 – Preguntas teóricas sobre servicios Angular (1.5 puntos)

Crea un documento de texto `PEC6_Ej1_respuestas_teoria` y responde a cada una de las siguientes preguntas:

- ¿Cuál es la función de los **componentes y servicios**? (i.e. cuándo se debe utilizar cada uno de ellos)
- ¿Qué es la **<<inyección de dependencias>>**? ¿Para qué sirve el decorador `@Injectable`?
- Explica los siguientes conceptos de la **programación reactiva** que se usan en **RxJS**:
 - `Observable`.
 - `Subscription`.
 - `Operators`.
 - `Subject`.
 - `Schedulers`.

d) ¿Cuál es la diferencia entre promesas y observables?

e) ¿Cuál es la función de la tubería (pipe) `async`?

Ejercicio 2 – Práctica – Servicios (3.5 puntos)

Partiendo **del último ejercicio realizado en la PEC anterior**, crea una carpeta `PEC6_Ej2` y realiza las siguientes tareas:

1. Crea un **nuevo servicio** llamado `WineService` que actuará como servicio común de los componentes `WineListComponent` y `WineNewComponent`.
2. Haz los componentes muy simples, dejando **sólo la responsabilidad relativa a la vista** y mueve cualquier **lógica al servicio**. El servicio debe tener al menos los **siguientes métodos**:

```
getWines(): Observable<Wine[]> { ...
}

changeQuantity(wineID: number, changeInQuantity: number): Observable<Wine> { ...
}

create(wine: Wine): Observable<any> { ...
}
```

3. **Registra el servicio correctamente al nivel de módulo**, crea un documento de texto `PEC6_Ej2_respuestas_teoria` y responde:
 - a. Cómo lo registrarías **manualmente**.
 - b. Cómo lo harías usando **AngularCLI**.
4. Usa **observables** y construye todos los componentes de modo que utilicen **APIs asíncronas** siempre que sea posible
5. **Utiliza la tubería (pipe) `async`** donde sea posible en lugar de hacer subscripciones manuales a los resultados.

El resultado debe ser de esta forma:

TPV Vinoteca!



Create Wine

☒ On Sale

Ejercicio 3 – Preguntas teóricas sobre interceptores (1 punto)

Crea un documento de texto PEC6_Ej3_respuestas_teoria y responde a cada uno de los siguientes puntos:

- a) ¿Qué son los **interceptores**?
- b) Analiza la siguiente **cadena de operadores de RxJS**, explica cada uno de los pasos que se están desarrollando y explica en qué caso usarías este código:

```
this.wines$ = this.searchSubject
    .startWith(this.searchTerm)
    .debounceTime(300)
    .distinctUntilChanged()
    .merge(this.reloadProductsList)
    .switchMap((query) =>
this.wineService.getWine(this.searchTerm));
```


Ejercicio 4 – Práctica – HttpClient (2.5 puntos)

Continuamos con nuestro ejercicio práctico de Vinoteca. Instala y ejecuta el servidor que se te ha proporcionado (`server-wines`) 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.

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 comprobado el correcto funcionamiento del servidor, crea una carpeta `REC6_Ej4` y realiza las siguientes tareas:

- Cambiar el servicio `WineService` para hacer peticiones HTTP en lugar de responder con datos incrustados en el servicio.
- Agrega un buscador de vinos que permita buscar vinos por su nombre (tecleando en el input) se debe refrescar la lista de vinos que se muestra.

Ejercicio 5 – Práctica – Pipes (1.5 puntos)

Continuamos con nuestro ejercicio práctico de Vinoteca, ahora utilizaremos Pipes para formatear de forma sencilla algunos aspectos. Crea una carpeta `PEC6_Ej5` y realiza las siguientes tareas:

- Utiliza pipes para que el precio de los vinos este formateado a dos decimales y se muestre el símbolo de la moneda €.
- Crea un pipe custom para que si el "imageUrl" del servicio viene sin informar, se muestre una imagen por defecto

Formato y fecha de entrega

Tienes que entregar un fichero *.zip, cuyo nombre tiene que seguir este patrón: loginUOC_PEC6.zip. Por ejemplo: dgarciaso_PEC6.zip. Este fichero comprimido tiene que incluir los siguientes elementos:

- Un fichero de texto PEC6_Ej1_respuestas_teoría para las respuestas del ejercicio 1.
- Una carpeta PEC6_Ej2 con el documento de texto PEC6_Ej2_respuestas_teoría y los ficheros resultado de haber realizado las tareas del ejercicio 2 (**Eliminar la carpeta node_modules**)
- Un fichero de texto PEC5_Ej3_respuestas_teoría para las respuestas del ejercicio 3.
- Una carpeta PEC5_Ej4 con los ficheros resultado de haber realizado las tareas del ejercicio 4 (**Eliminar la carpeta node_modules**)
- Una carpeta PEC5_Ej5 con los ficheros resultado de haber realizado las tareas del ejercicio 5 (**Eliminar la carpeta node_modules**)

El último día para entregar esta PEC es el **12 de enero 2021** hasta las **23:59**. Cualquier PEC entregada más tarde será considerada como no presentada.