PAC 6

Frameworks: Serveis a Angular

Universitat Oberta de Catalunya

Informació rellevant:

- Data límit de lliurament: 12 de juny.
- Pes a la nota de FC: 15%.

Contingut

Informació docent

Presentació

Aquesta pràctica se centra a conèixer la creació de serveis a Angular, es presentarà la biblioteca RxJS que es troba al cor d'Angular per a la comunicació entre components. Finalment, es farà ús del servei HttpClient que permet connectar el frontend amb el backend.

Objectius

Els objectius que es volen assolir amb el desenvolupament d'aquesta PAC són:

- Comprendre el funcionament dels **serveis a Angular**.
- Implementar el nostre **primer servei** amb lògica des dels **components**.
- Implementar un servei **que fa peticions a una API REST** fent ús del **`HttpClient`**.
- Conèixer i fer ús d'**interceptors**.

Enunciat

Aquesta PAC **conté 5 exercicis avaluables**. Heu de lliurar la vostra solució dels 5 exercicis avaluables (veure l'últim apartat).

Com que les activitats estan encadenades (i.e. per fer-ne una s'ha d'haver comprès l'anterior), **és altament recomanable fer les tasques i els exercicis en l'ordre en què apareixen en aquest enunciat.**

Abans de continuar has de:

Haver llegit els recursos teòrics disponibles a l'apartat “Continguts i recursos” de l'aula d'aquesta PAC

Crea una carpeta **PAC6** NO ÉS NECESSARI INICIALITZAR git en aquesta ruta, els commits es poden fer únicament al projecte ANGULAR. Igual que a les anteriors PAC, has de crear un fitxer README.md a l'arrel de la carpeta que contingui que contindrà almenys aquesta informació

• Login UOC.

• Nom i cognoms de l'alumne.

• Breu descripció del que s'ha realitzat en aquesta PAC, dificultats, millores realitzades, si cal tenir alguna cosa en compte a l'hora de corregir/executar la pràctica o qualsevol aspecte que vulgueu destacar.

Exercici 1 – Preguntes teòriques sobre serveis Angular (1.5 punts)

Crea una carpeta `PEC6_Ej_Teor`, dins d'aquesta carpeta creeu **un document markdown** que tinga com a nom **`PEC6_Ej1_respuestas_teoria.md`** i **respon** a cadascuna de les preguntes següents:

a) Quina és la funció dels **components i serveis**? (i.e. quan s'ha d'utilitzar cadascun)

b) Què és la <<**injecció de dependències**>>? Per què serveix el decorador **`@Injectable`**?

c) Explica els conceptes següents de la **programació reactiva** que es fan servir a **RxJS**:

- `Observable.`
- `Subscription.`
- `Operators.`
- `Subject.`
- `Schedulers.`

d) Quina és la diferència entre promeses i observables?

e) Quina és la funció de la tuberia (pipe) `async`?

Exercici 2 – Pràctica – Serveis (3.5 punts)

Crea una carpeta **`PEC6_Ej_Prac` partint de l'exercici pràctic ecommerce realitzat a la PAC anterior** i realitza les tasques següents:

1. Crea un **nou servei** anomenat `article-service` que actuarà com a servei comú dels components `article-list` i el component de la PAC5 `article-new-reactive`.

2. Fes els components molt simples, deixant **només la responsabilitat relativa a la vista** i mou qualsevol **lògica al servei**. El servei ha de tenir almenys els **mètodes següents**:

```
getArticles(): Observable<Article[]>{ ...
}

changeQuantity(articleID: number, chagenInQuantity: number): Observable <Article>{...
}

create(article: Article); Observable<any> {…
}
```

3. **Registra el servei correctament usant el paràmetre** `providedIn`, a la carpeta PEC6_Ej_Teor crea un document `PEC6_Ej2_respuestas_teoria.md` i respon:

    a. Quina és la diferència entre definir un servei usant el decorador @Injectable o `@NgModule`? Referència: https://angular.io/guide/providers

   b. Quines altres opcions admeten el decorador `@Injectable` per a la propietat `ProvidedIn`? Per què serveixen les altres configuracions? Referència: https://dev.to/christiankohler/improved-dependeny-injection-with-the-new-providedin-scopes-any-and-platform-30bb

4. Fes servir **observables** i construeix tots els components de manera que utilitzin **APIs asíncrones** sempre que sigui possible

5. **Utilitza la tuberia (pipe) `async`** on sigui possible en lloc de fer subscripcions mensuals als resultats.

Durant el desenvolupament de l'exercici executa les ordres `git` necessàries per afegir aquesta nova carpeta, fitxers i els canvis que hi facis al repositori local `git` de la PAC.

La carpeta **node_modules** i la carpeta **.angular**, no ha d'aparèixer al repositori git local, ni tampoc lliurar-la al zip final.

Exercici 3 – Preguntes teòriques sobre interceptors (1 punt)

A la carpeta PEC5_Ej_Teor, crea un fitxer Markdown que tingui com a nom PEC6_Ej3_respuestas_teoria.md i **respon** a cadascun dels punts següents:

a) Què són els **interceptors**?

b) Analitza la següent **cadena d'operadors de RxJS**, explica cadascun dels passos que s'estan desenvolupant i explica en quin cas faries servir aquest codi:

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

Exercici 4 – Pràctica – HttpClient (2.5 punts)

Continuem amb el nostre exercici pràctic d'ecommerce. Descomprimeix i executa el servidor que se t'ha proporcionat (`server-articles`) amb les següents ordres:

- npm i
- npm start

Això farà que arrenqui un servidor local de `node.js`, el qual estarà treballant amb articles (similar al que has vist als apunts sobre estocs). Aquest servidor exposa les següents APIs, al port 3000 (`http://localhost:3000/api/articles`):

- `GET` a `/api/articles` per obtenir una llista d'articles. Aquesta pot tenir un paràmetre opcional de consulta q, el qual és el nom de l'article a buscar. (per exemple http://localhost:3000/api/articles?q=arti)
- `POST` a `/api/articles` amb la informació d'un article al cos per crear un article al servidor (en el nostre cas serà en memòria, reiniciant el servidor es perdran totes les dades creades).
- `PATCH` a `api/articles/:id` amb l'ID de l'article a la URL i un camp `changeInQuantity` al cos canviarà la quantitat al carret d'articles per la quantitat passada com a paràmetre.

Podeu comprovar el funcionament correcte de l'API fent ús de l'eina Postman Desktop o Insomnia.

NOTA: Si necessites ajuda sobre el funcionament de POSTMAN Desktop pots consultar el següent recurs:

https://learning.oreilly.com/videos/postman-tutorial-getting/9781803243351/9781803243351-video2_3/

Un cop comprovat el correcte funcionament del servidor, dins del projecte de **ecommerce** realitza les tasques següents:

- Canviar el servei `article-service` per fer peticions HTTP en lloc de respondre amb dades incrustades al servei.
- Col·loca unes imatges article1.jpg i article3.jpg al lloc adequat perquè es mostrin correctament. Per a l'article2 que té imageUrl amb cadena buida, veurem com tractar-ho més endavant.
- Al component `article-list` afegeix un cercador d'articles que permeti cercar articles pel seu nom (teclejant a l'input) s'ha de refrescar la llista d'articles que es mostra.

Durant el desenvolupament de l'exercici executa les ordres `git` necessàries per afegir aquesta nova carpeta, fitxers i els canvis que hi facis al repositori local `git` de la PAC.

La carpeta **node_modules** i la carpeta **.angular**, no ha d'aparèixer al repositori git local, ni tampoc lliurar-la al zip final.

Exercici 5 – Pràctica – Pipes (1.5 punts)

Continuem amb el nostre exercici pràctic d'**ecommerce**, ara farem servir Pipes per formatar de forma senzilla alguns aspectes. Realitza les tasques següents sobre l'exercici **ecommerce**:

- Utilitza pipes perquè el preu dels articles estigui formatat a dos decimals i es mostri el símbol de la moneda €.

- Crea un pipe custom perquè si el camp "imgUrl" del servei ve amb cadena buida (per exemple l'article2 del servidor d'exemple), es mostri una imatge per defecte.

Durant el desenvolupament de l'exercici executa les ordres `git` necessàries per afegir aquesta nova carpeta, fitxers i els canvis que hi facis al repositori local `git` de la PAC.

La carpeta **node_modules** i la carpeta **.angular**, no ha d'aparèixer al repositori git local, ni tampoc lliurar-la al zip final.

Format i data de lliurament

Has de lliurar un fitxer *.zip, el nom del qual ha de seguir aquest patró: `loginUOC_PEC6.zip`. Per exemple: `dgarciaso_PEC6.zip`. Aquest fitxer comprimit ha d'incloure els elements següents:

Una **carpeta amb nom PEC6**, i al seu interior:

- Un **fitxer README.md** a l'arrel de la carpeta amb la informació indicada.
- Una carpeta **`PEC6_Ej_Teor`** amb els fitxers markdown sol·licitat per als exercicis amb explicacions teòriques:
  - Fitxer `PEC6_Ej1_respuestas_teoria.md` amb les respostes de l'exercici 1.
  - Fitxer `PEC6_Ej2_respuestas_teoria.md` amb les respostes de l'exercici 2.
  - Fitxer `PEC6_Ej3_respuestas_teoria.md` amb les respostes de l'exercici 3.
- Una carpeta **`PEC6_Ej_Prac`** amb el repositori git i els fitxers amb el resultat d'haver realitzat les tasques dels exercicis 2, 4 i 5 (No oblidis **eliminar les carpetes node_modules i .angular**)

Penalitzacions

- Lliurament en un altre format que no sigui l'especificat (ex. `zip`): **-0.75 punts**
- Comprimir fitxers dins del `zip`: **-1 punts.**
- Per a cada exercici/apartat on no es respecti la nomenclatura exacta de les carpetes o fitxers indicats (símbols, minúscules, majúscules, etc.): **-0.75 punts.**
- La no entrega del repositori local `git`: **-3 punts**
- Per cada carpeta node_modules i .angular lliurada: **-0.75 punts**

El darrer dia per lliurar aquesta PAC és el **12 de juny** fins a les **23:59**. Qualsevol PAC lliurada més tard serà penalitzada.