Desenvolupament Web amb

Angular – (MEAN stack 2)

Serveis http

Raúl García Lepe raulgl@expert.netmind.es

Serveis HTTP

- S'utilitzen per a conectar amb les APIs que ens proveiran de les dades que volem mostrar a l'aplicació i també les farem servir per a persistir les dades noves o modificades.

- Basats en els principis de la programació reactiva amb *streams* de dades asíncrons.

Comunicació síncrona

- La comunicació síncrona implica un client que espera que el servidor respongui a un missatge.
- Els missatges poden fluir en ambdos sentits.
- El remitent envia un missatge al receptor i el receptor rep aquest missatge i dóna resposta al remitent.
- El remitent no enviarà un altre missatge fins que rebi una resposta del receptor.

- Per exemple:
  - Quan es carrega una pàgina
  - En fer clic en un enllaç
  - Quan s'emplena un formulari de **Login**, s'envien les dades i cal esperar al fet que el servidor enviï la resposta (que serà la mateixa pàgina si hi ha error o una altra si és readreçada)

Comunicació asíncrona

- Connexió entre el client i el servidor que permet la transferència de dades no sincrònica: el client pot realitzar diverses peticions al servidor sense necessitat d'esperar per la resposta de la primera.
- Aquest mecanisme permet recarregar en segon pla una part de la pàgina web, deixant desbloquejat la resta.
- El client no es bloqueja esperant la resposta del servidor. Això ajuda al fet que les aplicacions web tinguin una interactivitat similar a les aplicacions d'escriptori i és en part el que fa alguns anys es denomina Web 2.0.

- Exemples d'ús:
  - Valorar una notícia amb estels o una notícia del mur de Facebook amb el “m'agrada”
  - Esborrar un correu en *gmail
  - Comentar una notícia)

Fluxos de dades: *streams*

- Sèries de dades encadenades que poden ser emeses en el temps
- En general es pot pensar una aplicació com una sèries de dades en comptes de valors aïllats
- Ex.:
  - El registre dels moviments del ratolí
  - Enviar i rebre dades de la base de dades per a oferir una interfàs personalitzada a l'usuari
  - Els valors que pot prendre una variable al llarg del temps

Programació Reactiva: Rx

- Consisteix en programar els *streams* de dades de manera asíncrona, reaccionant al eventos sense alterar el fil principal de execució

Ejemplo programación tradicional:

```
let a = 1;
let b = 3;
let resultado = a + b; // resultado vale 4
// Más tarde en las instrucciones...
a = 7; // Asignamos otro valor a la variable a
// Aunque se cambie el valor de "a", resultado sigue valiendo 4,
```

Ejemplo programación reactiva:

```
const suma = new Observable( subscriber => {
  let a = 5;
  let b = 3;
  subscriber.next(a + b);
   a = 7;
  subscriber.next(a + b);
    setTimeout(() => {
     a = 10
    subscriber.next(a+b);
    subscriber.complete();
  }, 1000);
});
```

RxJs: Observables

- Els fluxos de dades es programen mitjançant els **Observables**
- Es basen en el patró OBSERVER i el patró ITERATOR
- L'Observable agrupa tots els eventos en un de sol. Això serveix per a poder detectar uns canvis determinats i comunicar-los a on existeixi algú interessat.
- Ex.: La comunicación a la bbdd. Es transforma a **Observable**, després es crea un **Observer** que reaccionarà als canvis que es vulguin vigilar i, per últim, es suscriu (**suscribe**) l'observador a l'observable
- La resposta és asíncrona. No s'ha d'esperar a que finalitzi la comunicación.
- Observables vs Promises. Diferència les promeses no mès tornen una resposta i no es poden cancel·lar. L'observable molt més flexibles
- Observables vs CallBacks encadenats. Els callbacks son molt més complexos de manegar i raonar.

RxJs: Observables

- Analitzem el comportament de l'aplicació:

  https://github.com/midesweb/ej_facturacion_angular/tree/c240f0de34ab21341c0c85db2b85fabb8c0dab95

Serveis HTTP: Configuració

- Preparem les rutes de conexión a les API's. Farem servir els **environments**:
  - src/environments/environment.ts per a l'enrutament de desenvolupament

```
export const environment = {
  production: false,
  API_URL: 'http://localhost/apis/mi_crud/'
};
```

  - src/environments/environment.prod.ts per a l'enrutament de producció

```
export const environment = {
  production: true,
  API_URL: 'http://mis.apis.com/'
};
```

- Importem el mòdul d'Angular per a establir comunicacions http:

```
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  declarations: [AppComponent],
  imports: [
    RouterModule,
    CommonsModule,
    MembersModule,
    BrowserModule,
    AppRoutingModule,
    HttpClientModule
```

Serveis HTTP: Configuració

- Dins dels components i/o serveis des d'on anem a fer les comunicacions haurem de fer les següents importacións:

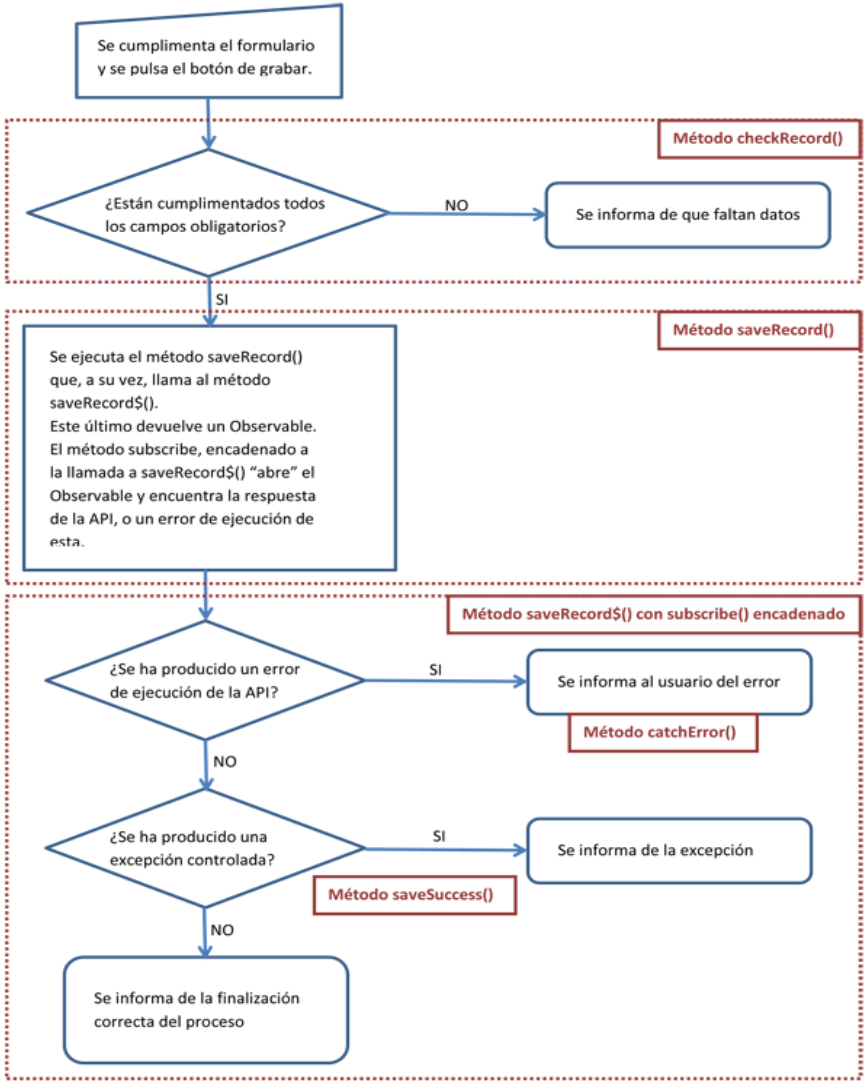
```
import { Observable } from 'rxjs/Observable';
import { HttpClient } from '@angular/common/http';

import { environment } from '../../../environments/environment';
```

Angular5

- Al constructor del component o servei s'haurà de pasar un objecte del tipus HttpClient:

```
constructor(private http: HttpClient) {
```

CRUDs: Create mètode



CRUDs: Create mètode

- Partint de l'aplicació de gestió de socis, afegim la lògica de conexión a l'API dins del servei.
- STEPS:

1. Creem una variable per a manegar la ruta d'environment (*afegida al fitxer d'environment*)

```
private URL = environment.API_URL;
```

2. Creem un Observable a partir de la resposta que ens doni el mètode http.POST enviat a l'API.

```
private saveRecord$(member: Member): Observable<any> {
  return this.http.post(this.URL + 'save_record.php', member);
}
```

ELs mètodes i variables rel·lacionades amb Observables normalment finalitzen amb "$"

CRUDs: Create mètode

- STEPS (cont.):

3. Suscripció a l'Observable:

```
private saveRecord() {
  this.saveRecord$(this.User).subscribe(
    this.saveSuccess.bind(this),
    this.catchError.bind(this)
  );
}
```

4. Implementación de la repuesta:

```
private saveSuccess(result) {
  if (result === '23000') {
    this.repeatedDNI = true;
  } else {
    this.dataUpgradedOk = true;
    this.dni = '';
    this.nombre = '';
  }
}
```

Exercici 1

- Implementa la resta de mètodes CRUD.