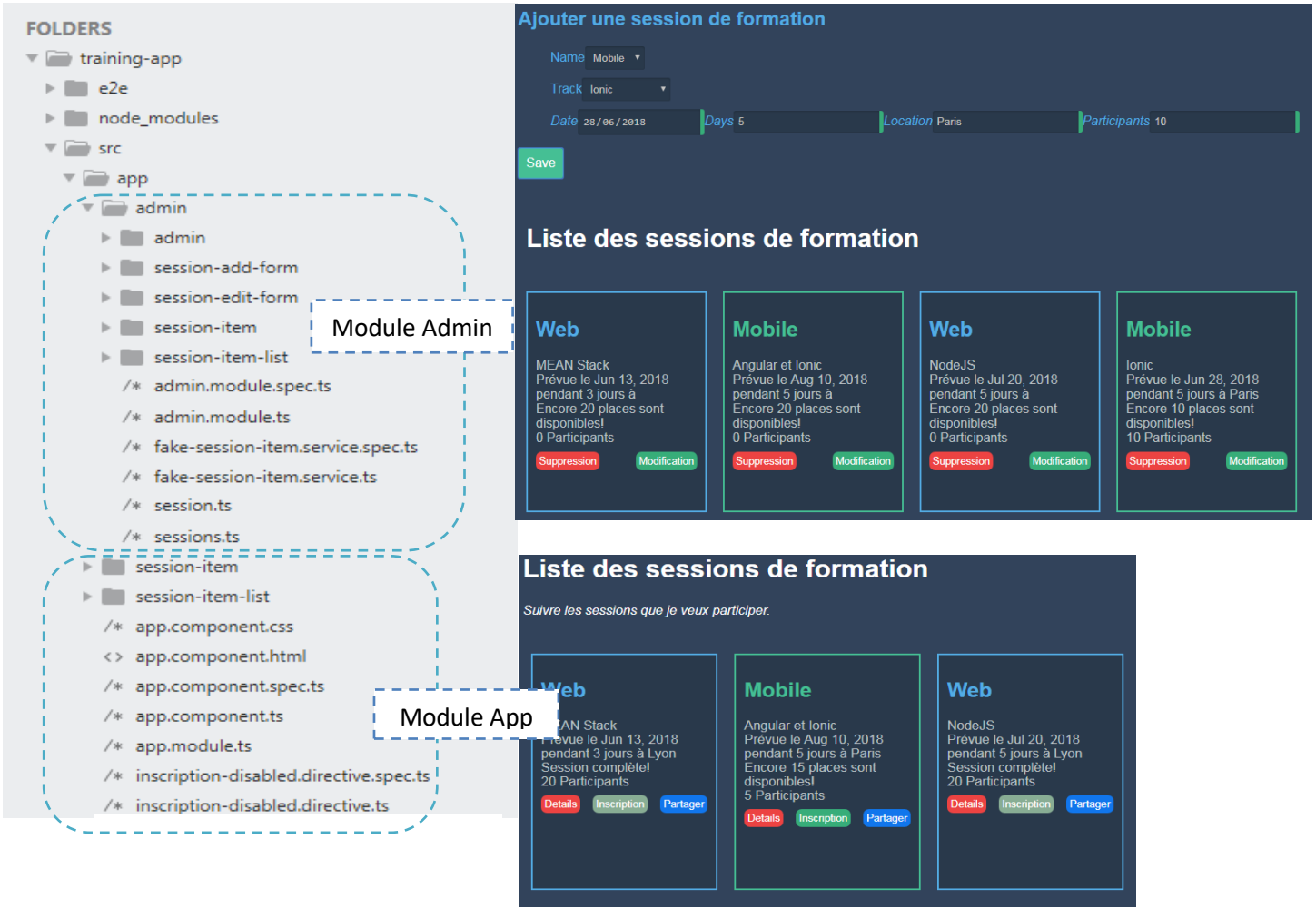
TP 3: Framework Angular (Template-Driven Forms & Dependency Injection)

On s'intéresse, dans un premier lieu, aux formulaires Angular de type *Template-Driven Forms* (piloté par le Template). Ce type de formulaire comporte deux parties : le Template HTML et un component class pour gérer par programme (d'une manière programmatique) les données et les interactions de l'utilisateur.

En second lieu, on s'intéresse à l'injection de dépendances qui est un *design pattern* bien connu. Nous allons voir comment un composant de notre application peut avoir besoin de faire appel à des fonctionnalités qui sont définies dans d'autres parties de l'application (un service, par exemple). C'est ce que l'on appelle une dépendance : le composant dépend du service.

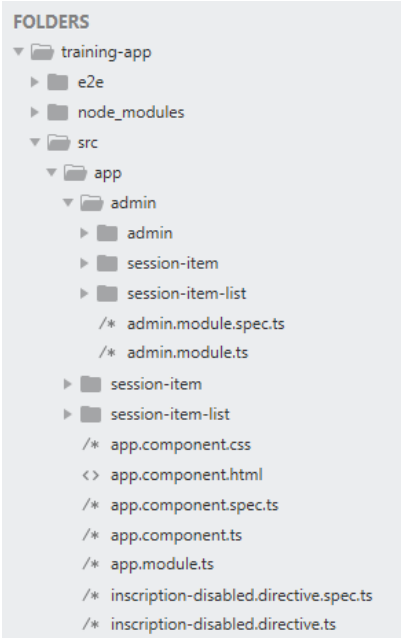
On vous demande dans ce TP de subdiviser l'application en deux modules : *AppModule* et *AdminModule*, et de développer un service qui sera utilisé (injecté) dans l'application en se réfèrent au mécanisme de l'injection de dépendances.



Etapes d'implémentations

Partie 1 : Les formulaires pilotés par le Template

1. Pour rendre notre application plus modulaire, on vous demande d'ajouter un nouveau module *Admin* contenant par défaut les composants suivants : *AdminComponent (root component), SessionItemComponent, SessionItemListComponent.*



2. Faire les modifications nécessaires pour que l'application bootstrape le module *AdminModule* par défaut.
   a. Importer le module *BrowserModule* dans le fichier *admin.module.ts*

```
import { BrowserModule } from '@angular/platform-browser';

@NgModule({

  imports: [BrowserModule],

  declarations: [SessionItemComponent,SessionItemListComponent,
AdminComponent],

  providers: [],

  bootstrap: [AdminComponent]

})
```

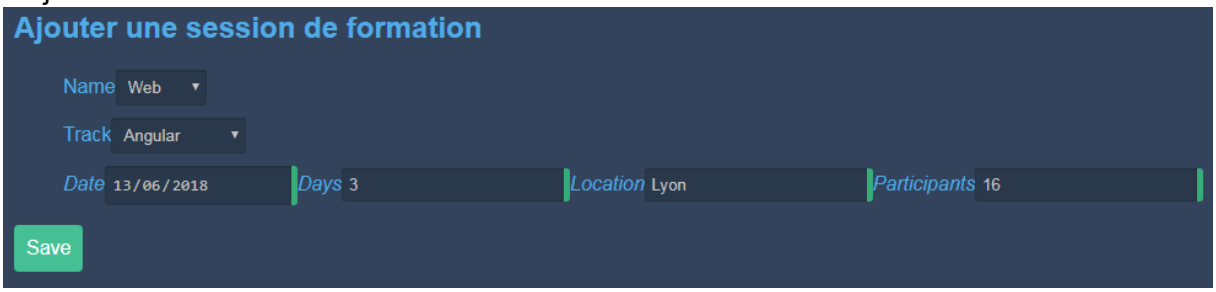
   b. Modifier le fichier *main.ts* pour charger le module AdminModule

```
platformBrowserDynamic().bootstrapModule(AdminModule)
```

c. Modifier la page index.html pour charger par défaut le composant root (*AdminComponent*) du module Admin.

```
<app-admin></app-admin>
```

3. Dans le module Admin, on vous demande de créer un formulaire permettant d'ajouter une nouvelle session de formation.



a. Dans le module Admin, importer le package FormsModule :

```
import { FormsModule } from '@angular/forms';
@NgModule({
  imports: [
    BrowserModule,
    FormsModule
  ]
```

b. Créer le component class du formulaire.

```
@Component({
  selector: 'app-session-add-form',
  templateUrl: './session-add-form.component.html',
  styleUrls: ['./session-add-form.component.css']
})
export class SessionAddFormComponent {
    constructor(){}
    ngOnInit() {
      }
     addSession(sessionItem) {
            console.log(sessionItem);
            }
    }
```

c. Créer le Template HTML du formulaire.

```
<header>
  <h2>Ajouter une session de formation</h2>
</header>
<form>
  <ul>
    <li>
      <label for="name">Name</label>
```

```
            <select name="name" id="name" required>
              <option value="Web">Web</option>
              <option value="Mobile">Mobile</option>
            </select>
          </li>
          <li>
            <label for="track">Track</label>
            <select name="track" id="track">
              <option value="MEAN">MEAN Stack</option>
<option value="Angular">Angular</option>
<option value="NodeJS">NodeJS</option>
<option value="Symphony">Symphony</option>
<option value="Laravel">Laravel</option>
<option value="Ionic">Ionic</option>
<option value="Android">Android</option>
<option value="Xamarin">Xamarin</option>
<option value="Swift">Swift</option>

            </select>
          </li>
          <i>
            <label for="date">Date</label>
          <input type="date" name="date" id="date"  required>
            <label for="duree">Days</label>
            <input type="number" name="duree" id="duree" required>
          </i>
          <i>
          <label for=" adress ">Location</label>
          <input type="text" name="adress" id=" adress "  required>
          </i>
          <i>
          <label for="participants">Participants</label>
          <input type="number" name="participants" id="participants"
required>
          </i>
        </ul>
        <button type="submit">Enregistrer</button>
</form>
```

d. Créer le style CSS du composant.

```
:host {
  display: block;
  padding: 10px;
}
ul {
  list-style-type: none;
}
ul li {
  margin: 10px 0;
}
header, label {
  color: #53ace4;
}
input, select {
  background-color: #29394b;
  color: #c6c5c3;
  border-radius: 3px;
  border: none;
```

```
  box-shadow: 0 1px 2px rgba(0,0,0,0.2) inset, 0 -
1px 0 rgba(0,0,0,0.05) inset;
  border-color: #53ace4;
  padding: 6px;
}
.ng-invalid:not(.ng-pristine):not(.required-
invalid) {
  border: 1px solid #d93a3e;
}
input[required].ng-invalid {
  border-right: 5px solid #d93a3e;
}
input[required]:not(.required-invalid),
input[required].ng-invalid:not(.required-invalid) {
  border-right: 5px solid #37ad79;
}
.error {
  color: #d93a3e;
}
#year {
  width: 50px;
}
button[type=submit] {
  background-color: #45bf94;
  border: 0;
  padding: 10px;
  font-size: 1em;
  border-radius: 4px;
  color: #ffffff;
  cursor: pointer;
}
button[type=submit]:disabled {
  background-color: #333;
  color: #666;
  cursor: default;
}
```

e. Utiliser la directive ***ngForm*** pour envoyer les données du formulaire vers le composant pour les afficher sur la console Chrome.
Modifier la balise *form* de la manière suivante :

```
<form #sessionItemForm="ngForm"
(ngSubmit)="addSession(sessionItemForm.value)">
```

Commentaires :

La directive ***ngForm*** représente le groupe de contrôles du formulaire, ici on en crée une référence `#sessionItemForm`. A la soumission (avec ***ngsubmit***) on transmet cette référence en choisissant la propriété **value** qui contient tous les contrôles avec les valeurs saisies

sous la forme d'un objet. La fonction *addSession()* permet d'afficher les données dans le console.

f. On va utiliser la directive ***ngModel*** pour envoyer les données du formulaire vers le composant pour les afficher sur la console Chrome. Ajouter la directive ***ngModel*** dans tous les éléments du formulaire. Voilà le cas de l'élément select.

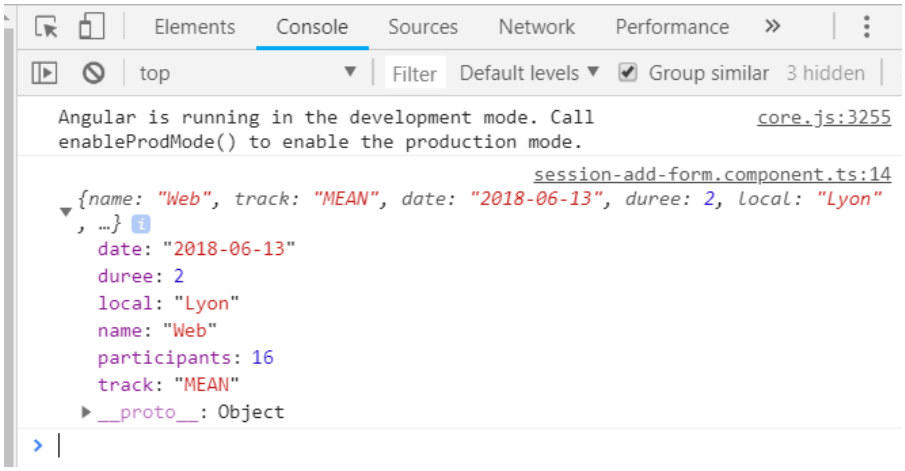
```
<li>
<label for="name">Name</label>
<select name="name" id="name" ngModel required>
  <option value="Web">Web</option>
  <option value="Mobile">Mobile</option>
</select>
</li>
```

g. Valider le remplissage du formulaire avant de l'envoyer (en cliquant sur le bouton Save).

```
<button type="submit"
[disabled]="!sessionItemForm.form.valid">Save</button>
```

h. On veut tester maintenant notre formulaire. Intégrer le composant *SessionAddFormComponent* dans le composant root (*AdminComponent*) en l'affichant dans la première position.

```
1  <section>
2    <app-session-add-form></app-session-add-form>
3    <app-session-item-list></app-session-item-list>
4  </section>
```



4. Dans le module Admin, créer un composant formulaire pour modifier une session de formation. L'objectif est de tester le Tow-Way Data Binding dans Angular.
   a. Créer la classe TS *Session* (fichier *session.ts*) qui présentera notre modèle (structure de données).

```
export class Session {
  constructor(
    public id: number,
    public name: string,
    public track: string,
    public date: Date,
    public duree: number,
    public adress: string,
    public participants: number,
    public isCompleted: boolean) {  }
}
```

   b. Créer le component class du formulaire.

```
import {Session} from '../session';
@Component({
  selector: 'app-session-edit-form',
  templateUrl: './session-edit-form.component.html',
  styleUrls: ['./session-edit-form.component.css']
})
export class SessionEditFormComponent implements OnInit {
  tracks = ['MEAN', 'Angular',
    'NodeJS', 'Android', 'Swift', 'Xamarin'];
 session = new Session(1, 'Web', this.tracks[0], new
Date('2018-06-11'), 10, 'Lyon', 0, false);

constructor() { }
ngOnInit() { }
editSession(sessionItem) {
    console.log(sessionItem);}
}
```

   c. Créer le Template HTML du formulaire. Tester la directive ngModel pour mettre à jour les données de la session affichée (voir le fonctionnement du mécanisme Two-way Data Binding).

```
<header>
  <h2>Modifier une session de formation</h2>
</header>
<form #sessionItemForm="ngForm"
(ngSubmit)="editSession(sessionItemForm.value)">
  <ul>
    <li>
      <label for="name">Name</label>
      <select name="name" id="name" [(ngModel)]="session.name"
required>
        <option value="Web">Web</option>
        <option value="Mobile">Mobile</option>
      </select>
    </li>
```

```
      <li>
        <label for="track">Track</label>
        <select id="track"
                required
                [(ngModel)]="session.track" name="track">
          <option *ngFor="let track of tracks"
[value]="track">{{track}}</option>
        </select> {{session.track}}

      </li>
      <i>
        <label for="date">Date</label>
        <input type="date" name="date" id="date"
[(ngModel)]="session.date" required>

        <label for="duree">Days</label>
        <input type="number" name="duree" id="duree"
[(ngModel)]="session.duree" required>
      </i>
      <i>
        <label for="adress">Adress</label>
        <input type="text" name=" adress " id=" adress "
[(ngModel)]="session.adress" required>
      </i>
      <i>
        <label for="participants">Participants</label>
        <input type="number" name="participants"
id="participants"  [(ngModel)]="session.participants" required>
      </i>
    </ul>
    <button type="submit"
[disabled]="!sessionItemForm.form.valid">Modifier</button>
</form>
```

d. Intégrer le composant *SessionEditFormComponent* dans le composant root (*AdminComponent*) en l'affichant dans la troisième position.

```
1  <section>
2    <app-session-add-form></app-session-add-form>
3    <app-session-item-list></app-session-item-list>
4      <app-session-edit-form></app-session-edit-form>
5  </section>
```

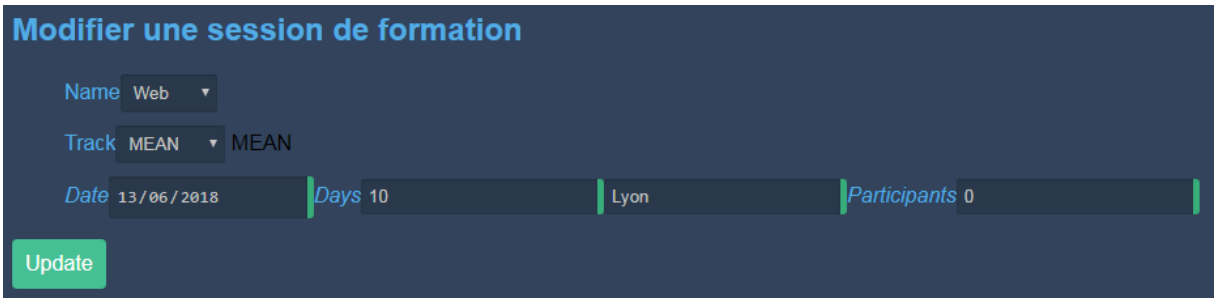
e. Validation du champ Adresse (l'utilisateur doit saisir 4 caractères au mimum).

```
<input type="text" name="adress" id="adress"
[(ngModel)]="session.adress"  #adress="ngModel"
minlength="4" required>
<div *ngIf=" adress.invalid && (adress.dirty ||
adress.touched)">
  <div *ngIf="adress.errors.minlength">
    Minimum 4 caractères!
  </div>
  <div  *ngIf="adress.errors.required">
    Le champ adresse est obligatoire!
```

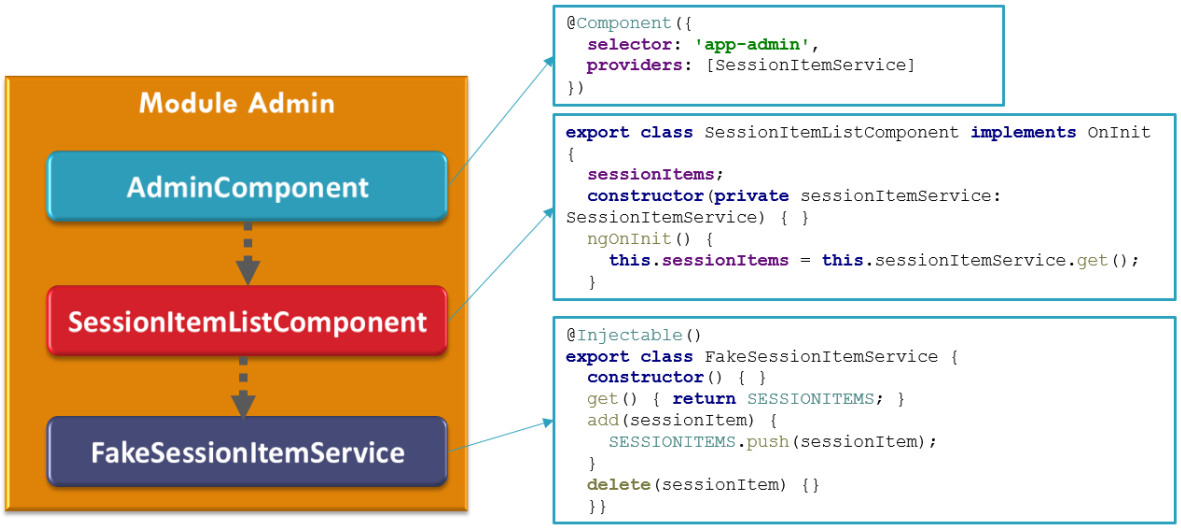
```
    </div>
</div>
```

Commentaires :

On ne veut pas que l'application affiche des erreurs avant que l'utilisateur ait la possibilité de modifier le formulaire. Les contrôles de *dirty* et de *touched* empêchent les erreurs de s'afficher jusqu'à ce que l'utilisateur tape la bonne valeur de l'adresse.



Partie 2 : Dependency Injection (La gestion de services)



1. Dans le module Admin, ajouter le service Fake*SessionitemService* qui permettra de gérer les sessions en termes d'affichage, d'ajout, de modification et de suppression. Pour cela, vous devez apporter des multiples modifications dans les autres composants.
    a. Pour l'instant, la liste de formations est stockée dans le tableau *sessionItems* qui est définit dans le composant *SessionItemListComponent*. On veut maintenant définir la liste de formations dans le fichier suivant (*sessions.ts)*.

```
import { Session } from './session';
export const SESSIONITEMS: Session[] = [
  {
```

```
    id: 1,
    name: 'Web',
    track: 'MEAN',
    date: new Date('2018-06-13'),
    duree: 3,
    adress: 'Lyon',
    participants: 0,
    isCompleted : false
  },
  {
    id: 2,
    name: 'Mobile',
    track: 'Ionic',
    date: new Date('2018-08-10'),
    duree: 5,
    adress: 'Paris',
    participants: 0,
    isCompleted : false
  },
  {
    id: 3,
    name: 'Web',
    track: 'NodeJS',
    date: new Date('2018-07/20'),
    duree: 5,
    adress: 'Lyon',
    participants: 0,
    isCompleted : false
  }
];
```

b. Créer le service *FakeSessionItemService*.

```
import { Injectable } from '@angular/core';
import {SESSIONITEMS} from '../sessions';
@Injectable()
export class FakeSessionItemService {

  constructor() { }
  get() { return SESSIONITEMS; }
  add(sessionItem) {
   sessionItem.id = SESSIONITEMS.length + 1;
   SESSIONITEMS.push(sessionItem);
  }
  delete(sessionItem) {
    let index;
    index = SESSIONITEMS.indexOf(sessionItem);
    if (SESSIONITEMS.indexOf(sessionItem) >= 0) {
      SESSIONITEMS.splice(index, 1);
    }
  }
}
```

Commentaires :

Pour indiquer à Angular 2 que ce service a lui-même des dépendances, on doit lui ajouter un décorateur *@Injectable()*

c. Enregistrer le service *FakeSessionItemService* comme étant un provider dans le composant *AdminComponent*.

```
import {FakeSessionItemService} from './fake-session-
item.service';

@Component({
  selector: 'app-admin',
  providers: [FakeSessionItemService],
  templateUrl: './admin.component.html',
  styleUrls: ['./admin.component.css']
})
```

Commentaires :

Pour qu'on puisse utiliser le service on doit l'enregistrer dans la liste de providers du composant *AdminComponent.* Ce service sera accessible et connu uniquement par ce composant et ses sous-composants *SessionAddFormComponent, SessionEditFormComponent et SessionItemListComponent*. Il suffit d'importer le service dans chaque sous-composant.

d. Maintenant, si on veut rendre notre *FakeSessionItemService* disponible à l'injection dans d'autres composants et services, il nous faut aussi l'enregistrer dans le module AdminModule (dans propriété *providers* du décorateur @NgModule). Modifier le fichier *admin.module.ts* :

```
import {FakeSessionItemService} from './fake-session-
item.service';
@NgModule({
  imports: [
    CommonModule,
    FormsModule
  ],
  declarations: [
    SessionItemComponent,
    SessionItemListComponent,
    SessionAddFormComponent,
    SessionEditFormComponent,
    AdminComponent
  ],
  providers: [FakeSessionItemService],
  bootstrap: [AdminComponent]
})
```

2. Invoquer le service *FakeSessionItemService* dans le composant *SessionItemListComponent* pour afficher la liste de formations.

a. Fichier TS *session-item-list.component.ts* :

```
import { Component, OnInit } from '@angular/core';
import {FakeSessionItemService} from '../fake-session-item.service';
@Component({
  selector: 'app-session-item-list',
  templateUrl: './session-item-list.component.html',
  styleUrls: ['./session-item-list.component.css']
})
export class SessionItemListComponent implements OnInit {
  sessionItems;

  constructor(private sessionItemService: FakeSessionItemService) { }
  ngOnInit() {
    this.sessionItems = this.sessionItemService.get();
  }
}
```

3. Invoquer le service `FakeSessionItemService` dans le composant *SessionAddFormComponent* pour ajouter une nouvelle session de formation.

```
import { Component, OnInit } from '@angular/core';
import {FakeSessionItemService} from '../fake-session-item.service';
@Component({
  selector: 'app-session-add-form',
  templateUrl: './session-add-form.component.html',
  styleUrls: ['./session-add-form.component.css']
})
export class SessionAddFormComponent implements OnInit {

  constructor(private sessionItemService: FakeSessionItemService) { }
  ngOnInit() {
  }
addSession(sessionItem) {
    console.log(sessionItem);
    this.sessionItemService.add(sessionItem);
  }
}
```

4. Invoquer le service `FakeSessionItemService` dans le composant *SessionItemComponent* pour supprimer une session de formation.
   a. Fichier TS *session-item.component.ts* :

```
import {Component, EventEmitter, Input, OnInit, Output} from '@angular/core';
import {FakeSessionItemService} from '../fake-session-item.service';
@Component({
  selector: 'app-session-item',
  templateUrl: './session-item.component.html',
  styleUrls: ['./session-item.component.css']
})export class SessionItemComponent implements OnInit {
  @Input() session;
  constructor(private sessionItemService: FakeSessionItemService) { }
  ngOnInit() {
  }
  onDelete() {
  console.log(this.session);
  this.sessionItemService.delete(this.session);
  }
}
```

b. Fichier Template HTML *session-item.component.html*:

```
<a class="delete" (click)="onDelete()">
  Suppression
</a>
```