GSB avec Angular2, partie 4

Gestion des rapports, mise à jour et ajout

Introduction

Dans cette partie, vous allez mettre en œuvre tout ce que vous avez appris lors des trois premières parties dans le cadre de la gestion des rapports.

Pour faire ce travail, vous disposez du code déposé ici :

https://github.com/patricegrand/GSBAngular2_V4.0

qui intègre le fichier HTML nécessaire pour cette partie, le *DataService* à jour ainsi que toutes les signatures des méthodes de la classe *app.visites.component*.

Pour ceux qui souhaiteraient continuer à partir des sources produites jusqu'ici :

- remplacer vos fichiers *app.visites.html*, *app.visites.component.ts* et la classe DataService par ceux fournis dans la version 4.0.
- insertion d'un champ dans la classe DataService :

```
visiteur : any;
```

- ainsi que sa valorisation à la connexion dans *app.connexion.component.ts* :

```
.subscribe(
    (data)=>{this.visiteur = data;
     this.dataService.visiteur = data;
      this.router.navigate(['accueil']);}
```

La gestion des rapports se décline en deux parties (voir la description des cas d'utilisation annexée à la partie 1), la mise à jour d'un rapport et l'ajout d'un nouveau rapport.

1) La mise à jour d'un rapport

On envisage les enchaînements suivants :

Fig 1

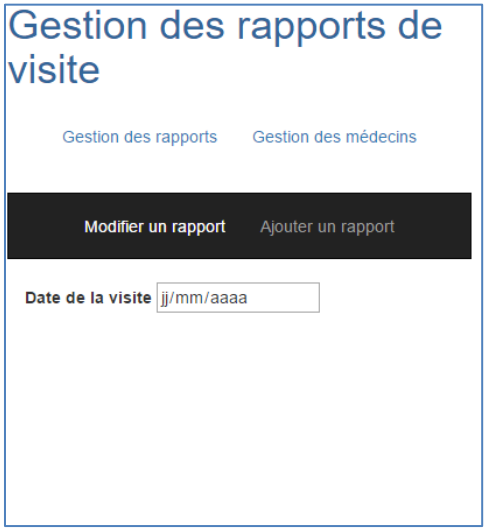

Fig 2

Fig 3

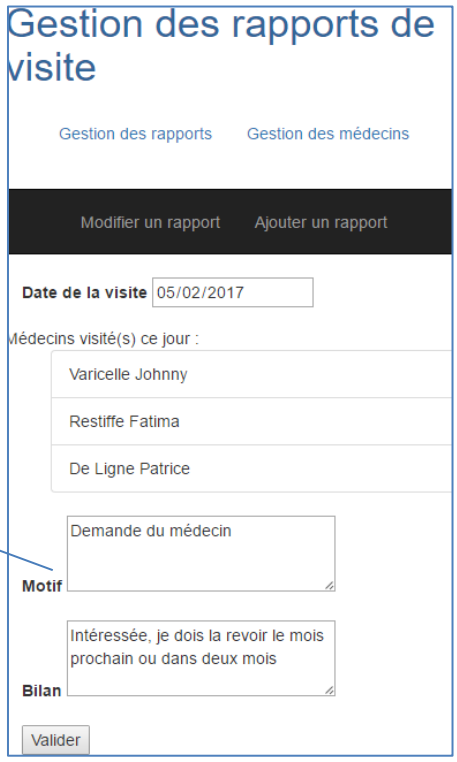

Fig 4

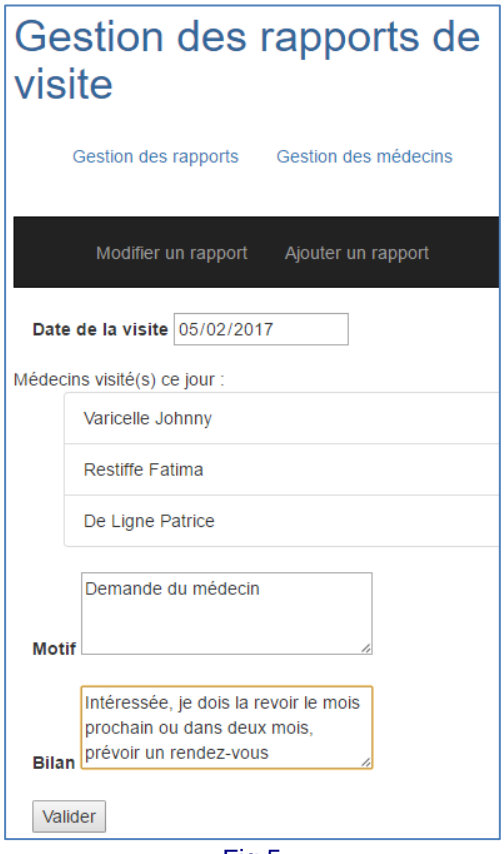

Fig 5

Fig 6

Nous allons commenter certains détails du fichier HTML app.visites.html ; la première partie concerne la recherche du rapport à partir de sa date (figures 2 et 3) :

```
1   <my-navbar></my-navbar>
2
3   <nav class="navbar navbar-inverse">
4       <ul class="nav navbar-nav" style="display:flex; flex-direction:
5                   row; justify-content: center">
6           <li><a (click)="modifierRapport()">Modifier un rapport</a></li>
7           <li><a (click)="ajouterRapport()">Ajouter un rapport</a></li>
8       </ul>
9   </nav>
10
11  <!--GESTION DE LA MISE A JOUR D'UN RAPPORT-->
12
13  <div *ngIf="gestionMajRapport">
14      <!--Recherche du rapport-->
15      <div class="col-lg-6">
16          <div class="form-group">
17              <label for="date"> Date de la visite</label>
18
19              <input id="date" type="date" [(ngModel)]="dateVisite"
20                                  (change)="chargerVisites()" />
21          </div>
22      </div>
23      {{titre}}
24      <div class="list-group">
25          <ul>
26              <li class="list-group-item"
27                  *ngFor="let rapport of lesRapports"
28                  (click)="selectionner(rapport)">
29                          {{rapport.nomMedecin}} {{rapport.prenomMedecin}}
30              </li>
31          </ul>
32      </div>
```

- Aux lignes 3 et 4, des classes Bootstrap permettent de faire apparaître le menu figure 1
- A la ligne 6, l'événement *click* appelle la méthode modifierRapport dont code ne fera que gérer les les propriétés associées aux directives *ngIf.
- Les lignes 15 à 22 permettent d'afficher la zone de recherche par date (figure 2) ; la propriété *dateVisite* doit être déclarée dans la classe Component, de type **Date** ; l'événement *change* appelle la méthode *chargerVisites*. Cette méthode doit :
  - valoriser le champ titre

  - s'abonner à la méthode *chargerRapportsAuneDate* (de la classe DataService) grâce à la méthode *subscribe* (voir son utilisation dans les deux dernières parties) ; il faudra au préalable ajouter un champ (ici *lesRapports*, cf ligne 27) de type *Array<any>* afin de récupérer les data retournées par l'appel

  Remarque : la méthode *chargerRapportsAuneDate* prend comme argument l'id du visiteur ; une propriété du *DataService* permet de le récupérer.
- Les lignes 24 à 32 permettent d'afficher les médecins concernés (figure 3) ; la directive d'itération *ngFor est utilisée, la méthode *selectionner* (associée au click sur un élément de la liste) devra :

  - enregistrer le rapport (son argument) dans un champ de la classe et afficher le formulaire qui suit en générant la directive *ngIf

Ainsi, la méthode *selectionner* permet l'affichage du formulaire (figure 4) de mise à jour du rapport.

La deuxième partie concerne la gestion de ce formulaire :

```
34      <!--formulaire de MAJ du rapport-->
35      <form class="col-lg-6" name="frmRapport"
36                      *ngIf="afficherRapport" (ngSubmit)="valider()">
37          <div class="form-group">
38              <label for="motif">Motif </label>
39              <textarea rows="3" cols="30" required="true"
40                  [(ngModel)]="rapport.motif" name="motif">
41          </textarea>
42
43          </div>
44          <div class="form-group">
45              <label for="bilan">Bilan</label>
46              <textarea rows="3" cols="30" required="true"
47                      [(ngModel)]="rapport.bilan" name="bilan"></textarea>
48          </div>
49          <button type="submit">Valider</button>
50          <div class={{typeMessage}}>{{messageMAJ }}</div>
51
52      </form>
53
```

- à la ligne 36, la directive **ngIf* permet de dévoiler ou non ce formulaire
- les directives ngModel accompagnent les champs de saisie nécessitant une déclaration d'un champ nommé *rapport* dans la classe de type *any*.
- la soumission du formulaire grâce à la méthode *valider*, ligne 36, va appeler la méthode *majRapport* du *DataService*
- la ligne 50 permet d'afficher un message (Fig 6), selon que la méthode *majRapport* s'est bien déroulée ou pas. Les classes (CSS) utilisées sont "*alert alert-success*" ou "*alert alert-danger*".

Travail à faire

Compléter la classe *VisitesComponent* permettant de mettre en œuvre ce qui vient d'être présenté.

2) L'ajout d'un nouveau rapport

Voici l'enchainement proposé :

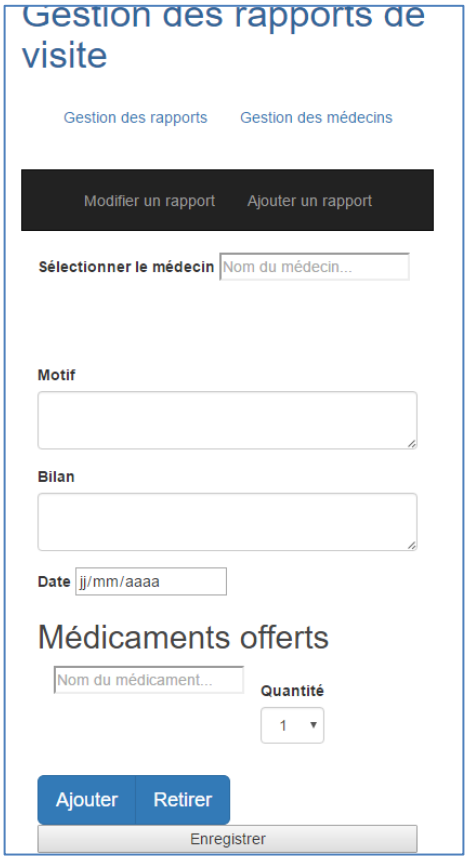

Fig 1

Fig 2

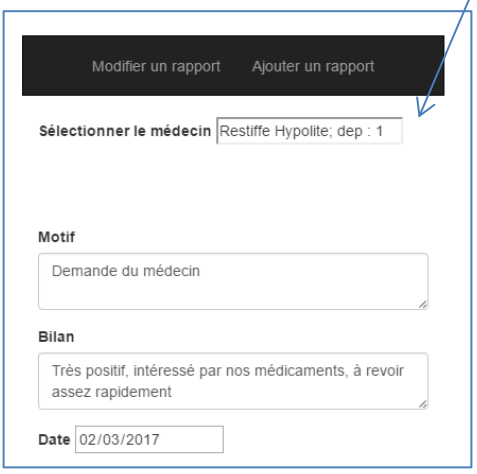

Fig 3

Fig 4

Fig 5

Fig 6

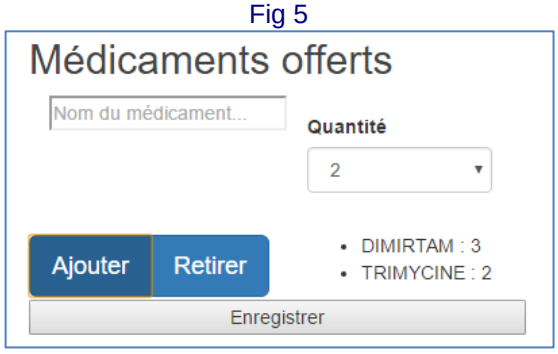

Fig 7

Fig 8

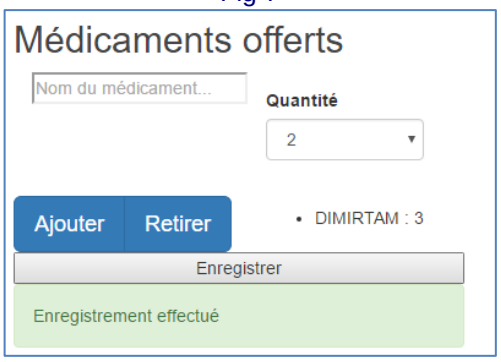

Fig 9

- Les figures 6, 7 et 8 permettent d'ajouter ou retirer des médicaments dans une liste avant d'enregistrer le rapport.

Nous allons, pour cette partie également, détailler certains éléments du fichier HTML :

```
56  <!--GESTION DE L'AJOUT D'UN RAPPORT-->
57  <div *ngIf="gestionAjoutRapport">
58      <!--Recherche du médecin -->
59      <div class="col-lg-6">
60          <div class="form-group">
61              <label for="medecin"> Sélectionner le médecin </label>
62              <input type="search" placeholder="Nom du médecin..."
63                      [(ngModel)]="nomMedecin"
64                      (keyup)="chargerMedecins()" id="medecin" />
65          </div>
66      </div>
67        <div class="list-group">
68          <ul>
69              <li class="list-group-item"
70                      *ngFor="let medecin of lesMedecins"
71                      (click)="selectionnerMedecin(medecin)">
72                      {{medecin.nom}} {{medecin.prenom}} ;departement :
73                      {{medecin.departement}}
74              </li>
75          </ul>
76      </div>
```

Cette partie correspond à la recherche du médecin (fig 2 et 3) ; cette partie est identique à la gestion de la mise à jour d'un médecin (partie 3 du support).
La ligne 57 permet de faire apparaitre cette partie grâce à la propriété *gestionAjoutRapport* associée à une directive *ngIf.

```
77      <!--Formulaire de saisie des infos du rapport -->
78      <form class="col-lg-6" (ngSubmit)="enregistrer()">
79          <div class="form-group">
80              <label for="motif"> Motif </label>
81              <textarea type="text" [(ngModel)]="motif"
82                  class="form-control" required="true" name="motif">
83          </textarea>
84          </div>
85          <div class="form-group">
86              <label for="bilan"> Bilan </label>
87              <textarea type="text" [(ngModel)]="bilan"
88                  class="form-control" required="true" name="bilan">
89          </textarea>
90          </div>
91          <div class="form-group">
92              <label for="date">Date</label>
93              <input type="date" required="true"
94                  [(ngModel)]="dateNouveauRapport" name="date" />
95          </div>
96
```

Cette partie permet de saisir les informations d'un rapport : motif, bilan et date (fig 3) :
- chaque directive *ngModel* nécessite de déclarer le champ correspondant dans la classe.

```
97          <!--Gestion des médicaments offerts -->
98          <h2>Médicaments offerts</h2>
99          <div>
100             <table>
101                 <tr>
102                     <td>
103                         <div class="col-lg-6">
104                             <div class="form-group">
105                                 <input type="search" [(ngModel)]="nomMedicament"
106                                         (keyup)="chargerMedicaments()"
107                                         placeholder="Nom du médicament..."
108                                         name="nomMedicament" />
109                             </div>
110                         </div>
111                         <br><br>
112                         <div class="list-group">
113                             <ul>
114                                 <li class="list-group-item"
115                                         *ngFor="let medicament of lesMedicaments"
116                                         (click)="choisirMedicament(medicament)">
117                                                 {{medicament.nomCommercial }}
118                                 </li>
119                             </ul>
120                         </div>
121                     </td>
```

Ce code permet de rechercher un médicament à partir du début du nom commercial (fig 4) :

- à la ligne 106, la méthode *chargerMedicament* (associée à l'événement *keyup*) devra appeler la méthode adéquate de la classe *DataService* et enregistrer les données retournées dans un champ (*lesMedicaments* –ligne 115) qu'il faudra bien sûr déclarer dans la classe *VisitesComponent*,

- la ligne 116 permet de mémoriser dans la classe le médicament sélectionné grâce à la méthode *choisirMedicament* ; il faudra déclarer un champ de type *any* pour cela.

```
103                         <div class="form-group">
104                             <label for="qteSelect"> Quantité</label>
105                             <select [(ngModel)]="qteSelect" class="form-control" name="qteSelect">
106                                 <option   *ngFor="let qte of qtes "
107                                           value="{{qte}}" >{{qte}}</option>
108                             </select>
109                         </div>
110                     </td>
111                 </tr>
112                 <tr>
113                     <td>
114                         <div class="btn-group">
115                             <button type="button" class="btn btn-primary btn-lg" (click)="ajouter()">
116                                     Ajouter</button>
117                             <button type="button" class="btn btn-primary btn-lg" (click)="retirer()">
118                                     Retirer</button>
119                         </div>
120                     </td>
121                     <td>
122                         <ul>
123                             <li *ngFor="let med of medicamentsSelect">
124                                 {{med.nom}} : {{med.qte}}</li>
125                         </ul>
126                     </td>
127                 </tr>
128             </table>
129         </div>
130         <button type="submit" class="btn-group btn-group-justified">Enregistrer
131         </button>
132         <div class={{typeMessage}}>{{messageEnregistrement}}</div>
133     </form>
134 </div>
```

- les lignes 104 à 108 permettent de gérer la liste déroulante des quantités (fig 4) ; le tableau de valeurs, *qtes,* doit être déclaré comme champ de la classe et initialisé :

```
qtes : Array<number> = [1,2,3,4,5];
```

- les lignes 122 à 125 permettent de gérer la mémorisation et l'affichage des médicaments successivement sélectionnés (fig 7). Une structure de donnée doit être déclarée et initialisée au préalable :

```
medicamentsSelect : Array<any> = new Array();
```

- ce tableau contiendra l'id du médicament, son nom commercial ainsi que la quantité sélectionnée.

On vous fournit le code de la méthode *ajouter ()* (ligne 115) qui ajoute un élément :

```
ajouter(): void{
    this.medicamentsSelect.push({id : this.medicamentSelect.id, nom :
      this.medicamentSelect.nomCommercial, qte : this.qteSelect});
    this.nomMedicament ="";
  }
```

La méthode **push** prend ici comme argument un objet constitué de 3 champs nommés *id*, *nom* et *qte*. Pour retirer un élément, on utilise une méthode **pop** (sans argument).

Travail à faire

Compléter la classe *VisitesComponent* afin de permettre l'ajout d'un rapport.

Le corrigé se trouve ici : https://github.com/patricegrand/GSBAngular2_V4.1