Compte-Rendu : Application GSB

Réalisé par Alexandre Bentz

Sommaire

I. Objectifs de l'application

Dans le cadre de leur travail, les visiteurs médicaux réalisent des visites ponctuelles à des professionnels de la santé. Ils réalisent des rapports qu'ils complètent avec diverses informations (praticien, date, motif, médicaments prescrits, échantillons offerts). Les produits qui sont proposés par le laboratoire sont tous identifiés par des numéros et caractérisés par de nombreuses informations (composition, interaction, posologie, famille). Les médecins ciblés par les visites sont répertoriés dans des fichiers achetés par l'entreprise auprès de différents organismes spécialisés. L'entreprise souhaite ainsi mettre à disposition toutes ces informations dans une application afin de permettre à ses démarcheurs de gérer leurs visites avec leurs mobiles et leurs tablettes.

Angular a été la solution retenue afin de réaliser une solution responsive et disponible sur toutes les plateformes : web, mobile et tablette.
Cette application se composera de différents onglets permettant : la gestion des rapports des visiteurs (ajout et modification) et la gestion des médecins (derniers rapports de visite et mise à jour des informations le concernant).

II. Le service REST

Un service REST utilise des requêtes HTTP pour créer (POST), afficher (GET), mettre à jour (OUT) et/ou supprimer des ressources. Il est donc généralement mis en relation avec une base de données afin de fournir des microservices au développeur, il sert d'interface entre la base de données et l'interface de l'application. On remarque ainsi une démarcation entre le « front-end », partie en relation directe avec l'utilisateur, qui s'occupe donc de la mise en page, de l'affichage des données; et le « back-end » qui correspond aux différents services nécessaires au bon fonctionnement de la solution.

En ce qui concerne le projet, le service REST nous a été fourni, il est assimilé à un service « Data » qui regroupe les différentes fonctions d'interaction avec l'API et permet ainsi d'exploiter la base de données.

III. Déroulement du projet

A. MISE EN PLACE DU COMPOSANT DE CONNEXION

La page de connexion doit offrir à l'utilisateur une interface lui permettant d'entrer ses identifiants, une erreur est affichée lorsqu'ils sont incorrects. La vue est donc composée d'un formulaire qui appelle le REST lors de sa validation.

Voici le code de la page **connexion.component.html** :

```
<div class='container'>
  <h2>Connexion</h2>
  <div class='alert alert-danger' [hidden]='estCache'>{{lblMessage}}</div>
  <form class="col-lg-4" name="frmLogin" (ngSubmit)="valider()">
    <div class="form-group">
      <label>Login</label>
      <input class="form-control" [(ngModel)]='login' type="text" name='lg'/
>
    </div>
    <div>
      <label>Password</label>
      <input class="form-control" [(ngModel)]='password' type='password'
name='motdepasse'/>
    </div>
    <button class="btn btn-secondary" type="submit">Valider</button>
  </form>
</div>
```

Le contrôleur lui doit instancier les différentes variables du formulaire et définir la méthode de connexion, qui appelle le service Data pour accéder au service de connexion du REST. En cas de réussite la variable du visiteur est valorisée et l'utilisateur est redirigé dans l'application. Sinon, une erreur est affichée.

Voici la fonction correspondante du fichier **data.service.ts** :

```
public connexion(loginIn: string, mdpIn: string) {
    const url: string = this.urlDomaine + '/connexion?login=' +
loginIn + '&mdp=' + mdpIn;
    const req = this.http.get(url);
    return req;
  }
```

Voici le contenu de la page **connexion.component.ts** :

```
login: string;
  password: string;
  estCache = true;
  lblMessage: string;
  visiteur: any;

  constructor(private router: Router, private dataService: DataService) {
}
  ngOnInit() {
  }
  valider(): void {
    this.dataService.connexion(this.login, this.password).subscribe(
      (data) => { this.visiteur = data; this.dataService.visiteur = data;
this.router.navigate(['accueil']); }
      , (error) => { this.estCache = false; this.lblMessage = 'Erreur :
Identifiants incorrects'; }
    );
  }
```

B. MISE EN PLACE DU ROUTAGE

Afin de mettre en place le routage, qui permet de changer de page sans la recharger, on définit un tableau dans le fichier prévu à cet effet. Chaque chemin qui sera utilisé par l'application y est défini et est associé avec le composant que l'on souhaite afficher. Une balise spéciale est prévue pour exploiter les routes et une ligne doit être ajoutée dans l'index du projet.

Voici le tableau situé dans le fichier **app-routing.module.ts** :

```
const routes: Routes = [
  { path: '', component: ConnexionComponent },
  { path: 'medecins', component: MedecinsComponent},
  { path: 'visites', component: VisitesComponent},
  { path: 'accueil', component: NavbarComponent}
];
```

Voici la ligne ajoutée dans le reader du fichier **index.html** : `<base href="/">`

La page **app.component.html** ne se résume donc plus qu'à deux lignes :

```
<h1>Gestion des rapports de visite</h1>
<router-outlet></router-outlet>
```

La barre de navigation exploite ces différents chemins et permet à l'utilisateur de naviguer dans les deux onglets de l'application.

Voici le code de la page **navbar.component.html** :

```
<nav class="navbar">
  <ul class="navbar-nav" style="display:flex; flex-direction: row;
justify-content: center;">
    <li><a routerLink="/visites">Gestion des rapports</a></li>
    <li><a routerLink="/medecins">Gestion des médecins</a></li>
  </ul>
</nav>
```

C. MISE EN PLACE DE LA RECHERCHE DES MÉDECINS

Afin de permettre à l'utilisateur de rechercher un médecin, la vue contient une zone de texte ou il peut entrer son nom. Lors de chaque presse d'une frappe, le service Data appelle le REST afin d'obtenir la liste des utilisateurs et ainsi valoriser la variable correspondante. La vue s'occupe ensuite d'afficher une liste à l'écran sous la barre de recherche, l'utilisateur peut donc choisir son médecin.

Voici le code correspondant à la page **medecins.component.html** :

```
<div class="col-lg-6">
  <div class="form-group">
      <input type="search" placeholder="Nom du
médecin..." [(ngModel)]="nomMedecin" (keyup)="charger()" />
  </div>
</div>
<div class="list-group" [hidden]="estCacheListe">
  <ul>
    <li class="list-group-item" *ngFor="let medecin of
lesMedecins" (click)="selectionner(medecin)">
      {{ medecin.nom }} {{ medecin.prenom }} - Département :
{{ medecin.departement }}
    </li>
  </ul>
</div>
```

Le contrôleur n'a donc qu'à effectuer les différents changements d'interface grâce à la propriété « Hidden », contacter le REST pour obtenir la liste des médecins et permettre à l'utilisateur de choisir son médecin.

Voici le code correspondant situé dans le fichier **medecins.component.ts** :

```
charger(): void {
    this.dataService.chargerMedecins(this.nomMedecin)
      .subscribe(
        (data) => { this.lesMedecins = data; },
        (error) => { }
      );
  }

  selectionner(med): void {
    this.medecin = med;
    this.estCacheMenu = false;
    this.nomMedecin = this.medecin.nom + ' ' + this.medecin.prenom + '
dep:' + this.medecin.departement;
    this.lesMedecins = null;

  }
```

Et voici la fonction correspondante dans le service **data.service.ts** :

```
public chargerMedecins(nomMedecin: string) {
      const url: string = this.urlDomaine + '/medecins?nom=' + nomMedecin;
      const req = this.http.get(url);
      return req;
  }
```

D. AFFICHAGE DES RAPPORTS DU MÉDECIN

Une barre de navigation est définie au début du composant afin d'accéder aux deux sections possibles de celui-ci. Lorsque l'utilisateur souhaite consulter les rapports, le contrôleur va tout simplement contacter le service Data, pour récupérer les informations à partir du REST en fonction du médecin sélectionné.

Voici le code de la partie concernée dans le fichier **medecins.component.html** :

```
<nav class="navbar-inverse" [hidden]="estCacheMenu">
  <ul class="navbar-nav" style="display:flex; flex-direction: row;
justify-content: center; width: 100%;">
    <li><a (click)="derniersRapports()" style="margin-left:
5%;">Derniers rapports</a></li>
    <li><a (click)="majMedecin()">Mise à jour</a></li>
  </ul>
</nav>
<table class="table table-bordered" *ngIf="afficherRapports">
  <tr>
    <th>Date</th>
    <th>Motif</th>
    <th>Bilan</th>
    <th>Nom du visiteur</th>
  </tr>
  <tr *ngFor="let rapport of lesRapports">
    <td>{{ rapport.date }}</td>
    <td>{{ rapport.motif }}</td>
    <td>{{ rapport.bilan }}</td>
    <td>{{ rapport.nom }}</td>
  </tr>
</table>
```

Voici le code concerné dans le fichier **medecins.component.ts** :

```
derniersRapports() {
    this.afficherRapports = true;
    this.afficherMedecin = false;
    this.dataService.chargerRapports(this.medecin.id).subscribe((data)
=> {
      this.lesRapports = data;
    }, (error) => { }
    );
  }
```

Voici la méthode correspondante dans le service **data.service.ts :**

```
public chargerRapports(idMedecin: number ) {
    const url: string = this.urlDomaine + '/rapports/' + idMedecin;
    const req = this.http.get(url);
    return req;
  }
```

E. MISE EN PLACE DE LA MISE À JOUR DES MÉDECINS

L'interface de mise à jour des médecins s'affiche lorsqu'elle est sélectionnée dans le menu supérieur par l'utilisateur, après le choix du médecin. Différents champs présentent les différentes informations récupérées sur le médecin, et permettent de les modifier (adresse, téléphone, spécialité). Lorsque les modifications sont validées, les nouvelles données sont transmises au service Data et envoyées au REST pour mettre à jour la base de données.

Voici le code correspondant dans le fichier **medecin.component.html** :

```
<form class="col-lg-6" name="frmMedecin"
*ngIf="afficherMedecin" (ngSubmit)="valider()">
  <div class="form-group">
    <label for="adresse">Adresse </label>
    <textarea rows="3" cols="30"
required="true" [(ngModel)]="medecin.adresse" name="adr"> </textarea>
  </div>
  <div class="form-group">
    <label for="tel">Téléphone</label>
    <input type="text" required="true" [(ngModel)]="medecin.tel"
name="tel" />
  </div>
  <div class="form-group">
    <label for="spec">Spécialité</label>
    <input type="text" [(ngModel)]="medecin.specialitecomplementaire"
name="spec" />
  </div>
  <button type="submit">Valider</button>
  <div class="alert alert-danger"*ngIf="afficherMessage">{{lblMessage}}</
div>
</form>
```

Voici le code correspondant dans le fichier **medecin.component.ts** :

```
majMedecin(): void {
    this.afficherRapports = false;
    this.afficherMedecin = true;
    this.afficherMessage = false;

  }
  valider(): void {
    this.afficherMessage = true;
    this.afficherRapports = false;
    this.afficherMedecin = true;
    this.dataService.majMedecin(this.medecin.id, this.medecin.adresse,
this.medecin.tel, this.medecin.specialitecomplementaire)
      .subscribe(
        () => {
          this.lblMessage = 'Mise à jour effectuée';
        }, () => { this.lblMessage = 'Mise à jour effectuée'; }
      );
  }
```

Voici la méthode correspondante dans le service **data.service.ts** :

```
public majMedecin(id: string , adresse: string, tel: string, spe: string) {
    let url: string =  this.urlDomaine + '/majmedecin?idmedecin=' + id +
'&adresse=';
    url += adresse + '&tel=' + tel + '&specialite=' + spe;
    const req = this.http
                .get(url);
    return req;
}
```

F. MISE EN PLACE DE LA MISE À JOUR DES RAPPORTS

Une barre de navigation est définie au début du composant afin d'accéder aux deux sections possibles de celui-ci. Pour mettre à jour un rapport, la vue permet à l'utilisateur d'entrer une date et ainsi d'obtenir la liste des rapports datés de ce jour. Comme pour la recherche des médecins, le contrôleur appelle le service Data à chaque frappe de touche afin d'actualiser la variable contenant le résultat fourni par le REST. Après avoir sélectionné le rapport, les informations (motif, bilan) sont affichées et modifiables par l'utilisateur. Il peut enfin valider ses modifications et ainsi envoyer les données modifiées au service Data, puis au REST qui modifiera la base de données en conséquence.

Voici le code correspondant dans le fichier **visites.component.html** :

```
<nav class="navbar navbar-inverse">
  <ul class="nav navbar-nav" style="display:flex; flex-direction:row; ">
    <li><a (click)="modifierRapport()">Modifier un rapport</a></li>
    <li><a (click)="ajouterRapport()">Ajouter un rapport</a></li>
  </ul>
</nav>

<div *ngIf="gestionMajRapport">
  <div class="col-lg-6">
    <div class="form-group">
      <label for="date">Date de la visite </label>
      <input id="date"
type="date" [(ngModel)]="dateVisite" (change)="chargerVisites()" />
    </div>
  </div>
  <br>
  {{titre}}
  <br>
  <div class="list-group">
    <ul>
      <li class="list-group-item" *ngFor="let rapport of
lesRapports" (click)="selectionner(rapport)">
        {{rapport.nomMedecin}} {{rapport.prenomMedecin}}
      </li>
    </ul>
  </div>
</div>

<form class="col-lg-6" name="frmRapport"
*ngIf="afficherRapport" (ngSubmit)="valider()">
  <div class="form-group">
    <label for="motif">Motif</label>
    <textarea rows="3" cols="30" class="form-control"
required="true" [(ngModel)]="rapport.motif" name="motif"></textarea>
  </div>
  <div>
    <label for="bilan">Bilan</label>
    <textarea rows="3" cols="30" class="form-control"
required="true" [(ngModel)]="rapport.bilan" name="bilan"></textarea>
  </div>
  <button class="btn btn-primary btn-lg" type="submit">Valider</button>
  <div class={{typeMessage}}>{{ messageMAJ }}</div>
</form>
```

Voici le code correspondant dans le fichier **visites.component.ts** :

```
chargerVisites(): void {
    this.titre = 'Medecins visité(s) à ce jour';
    console.log(this.dataService.visiteur.id);
    console.log(this.dateVisite);

this.dataService.chargerRapportsAuneDate(this.dataService.visiteur.id,
this.dateVisite).subscribe(
      (data) => { this.lesRapports = data; }
      , (error) => { }
    );
  }
  selectionner(rapport) {
    this.rapport = rapport;
    this.afficherRapport = true;
    console.log(this.rapport);
  }
  valider() {
    this.dataService.majRapport(this.rapport.idRapport,
this.rapport.motif, this.rapport.bilan).subscribe(
      () => { this.messageMAJ = 'Enregistrement effectué'; },
      () => { this.messageMAJ = 'Enregistrementt effectué'; });
  }
```

Voici les méthodes correspondantes du service **data.service.ts** :

```
public chargerRapportsAuneDate(idVisiteur: string, date: Date ) {
    const url: string =  this.urlDomaine + '/rapports_a_date?
idVisiteur=' + idVisiteur + '&date=' + date;
    console.log(url);
    const req = this.http.get(url);
    return req;
}
  public majRapport(idRapport: string, motif: string, bilan: string) {
    let url: string =  this.urlDomaine + '/majrapport?idRapport=' +
idRapport + '&motif=';
    url += motif + '&bilan=' + bilan;
    const req = this.http.get(url);
    return req;
}
```

G. MISE EN PLACE DE L'AJOUT D'UN RAPPORT

Lorsque l'utilisateur choisit de créer un nouveau rapport, de nombreuses informations lui sont demandées : il doit choisir un médecin, la méthode de recherche des médecins est de nouveau appliquée, et le médecin sélectionné est stocké dans une variable. Un motif, le bilan et la date sont ensuite à remplir et stocker aussi dans des variables. Concernant les médicaments, la méthode de recherche est de nouveau mise en place afin que l'utilisateur sélectionne le médicament offert, puis la quantité. Une fois les bons paramètres sélectionnés, il peut ajouter le médicament à la liste, et en cas d'erreur il a la possibilité de retirer le dernier médicament ajouté. Lorsque tout est rempli, l'utilisateur valide le formulaire et transmet les informations au service Data puis au REST pour ajouter les nouvelles informations dans la base de données.

Voici le code correspondant dans le fichier **visites.component.html** :

```
<div *ngIf="gestionAjoutRapport">
  <div class="col-lg-6">
    <div class="form-group">
      <label for="medecin">Selectionner le medecin </label>
      <input type="search" placeholder="Nom du
médecin..." [(ngModel)]="nomMedecin" (keyup)="chargerMedecins()"
        id="medecin" />
    </div>
  </div>
  <div class="list-group">
    <ul>
      <li class="list-group-item" *ngFor="let medecin of
lesMedecins" (click)="selectionnerMedecin(medecin)">
        {{medecin.nom}} {{medecin.prenom}} ;departement :{{medecin.departement}}
</li>
    </ul>
  </div>

  <form class="col-lg-6" (ngSubmit)="enregistrer()">
    <div class="form-group">
      <label for="motif">Motif</label>
      <textarea type="text" [(ngModel)]="motif" class="form-control"
required="true" name="motif"></textarea>
    </div>
    <div class="form-group">
      <label for="bilan">Bilan</label>
      <textarea type="text" [(ngModel)]="bilan" class="form-control"
required="true" name="bilan"></textarea>
    </div>
    <div class="form-group">
      <label for="date">Date</label>
      <input type="date" required="true" [(ngModel)]="dateNouveauRapport"
name="date" />
    </div>
```

```
<h2>Médicaments offerts</h2>
    <div>
      <table>
        <tr>
          <td>
            <div class="col-lg-6">
              <div class="form-group">
                <input
type="search" [(ngModel)]="nomMedicament" (keyup)="chargerMedicaments()"
                   placeholder="Nom du médicament…" name="nomMedicament" />
              </div>
            </div>
            <br><br>
            <div class="list-group">
              <ul>
                <li class="list-group-item" *ngFor="let medicament of lesMedicaments"

(click)="choisirMedicament(medicament)">{{ medicament.nomCommercial }}</li>
              </ul>
            </div>
          </td>
          <td>
            <div class="form-group">
              <label for="qteSelect">Quantité</label>
              <select [(ngModel)]="qteSelect" class="form-control" name="qteSelect">
                <option *ngFor="let qte of qtes" value="{{ qte }}">{{ qte }}</option>
              </select>
            </div>
          </td>
        </tr>
        <tr>
          <td>
            <div class="btn-group">
              <button type="button" class="btn btn-primary btn-
lg" (click)="ajouter()">Ajouter</button>
              <button type="button" class="btn btn-primary btn-
lg" (click)="retirer()">Retirer</button>
            </div>
          </td>
          <td>
            <ul>
              <li *ngFor="let med of medicamentsSelect">{{ med.nom }} : {{ med.qte }}
</li>
            </ul>
          </td>
        </tr>
      </table>
    </div>
    <button type="submit" class="btn btn-primary btn-lg">Enregistrer</button>
    <div class={{typeMessage}}>{{ messageEnregistrement }}</div>
  </form>
</div>
```

Voici le code correspondant dans le fichier **visites.component.ts** :

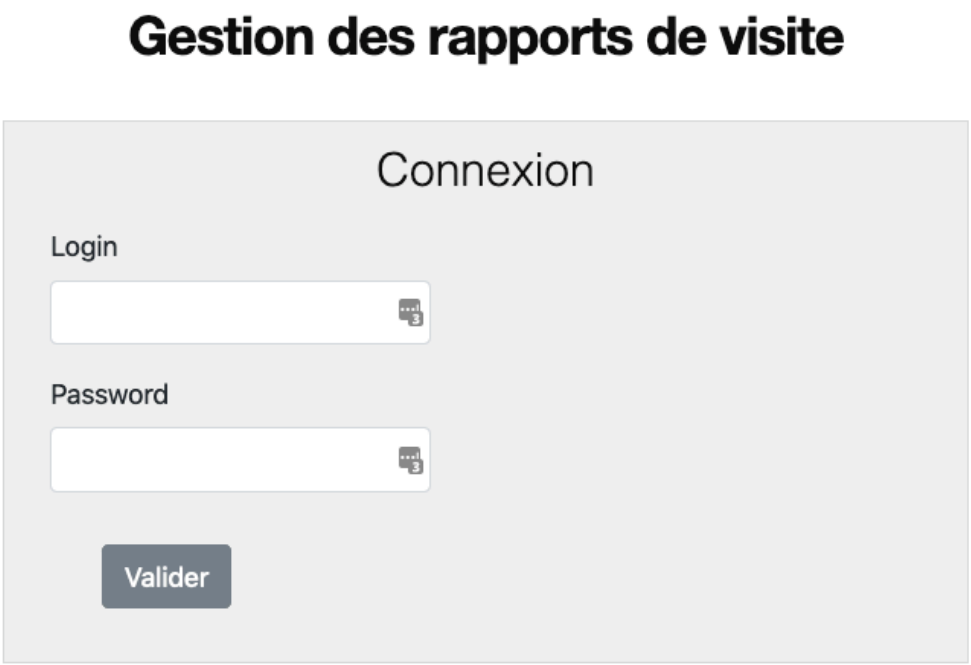
```
chargerMedecins() {
    this.dataService.chargerMedecins(this.nomMedecin)
      .subscribe(
        (data) => { this.lesMedecins = data; },
        (error) => { }
      );
  }
  selectionnerMedecin(medecin) {
    this.medecin = medecin;
    this.nomMedecin = this.medecin.nom + ' ' + this.medecin.prenom + ' dep:' +
this.medecin.departement;
    this.lesMedecins = null;
  }
  enregistrer() {
    this.dataService.enregistrerRapport(this.dataService.visiteur.id,
this.medecin.id, this.motif, this.dateNouveauRapport,
      this.bilan, this.medicamentsSelect).subscribe(
      () => { this.messageEnregistrement = 'Enregistrement effectué'; },
      () => { this.messageEnregistrement = 'Enregistrement effectué'; });
  }
  ajouter() {
    this.medicamentsSelect.push({ nom: this.medicament.nomCommercial, qte:
this.qteSelect });
    this.nomMedicament = '';
  }
  retirer() {
    this.medicamentsSelect.pop();
  }
  chargerMedicaments() {
    this.dataService.chargerMedicaments(this.nomMedicament).subscribe(
        (data) => { this.lesMedicaments = data; },
        (error) => { }
      );
  }
  choisirMedicament(medicament) {
    this.medicament = medicament;
    this.nomMedicament = this.medicament.nomCommercial;
    this.lesMedicaments = null;

  }
```

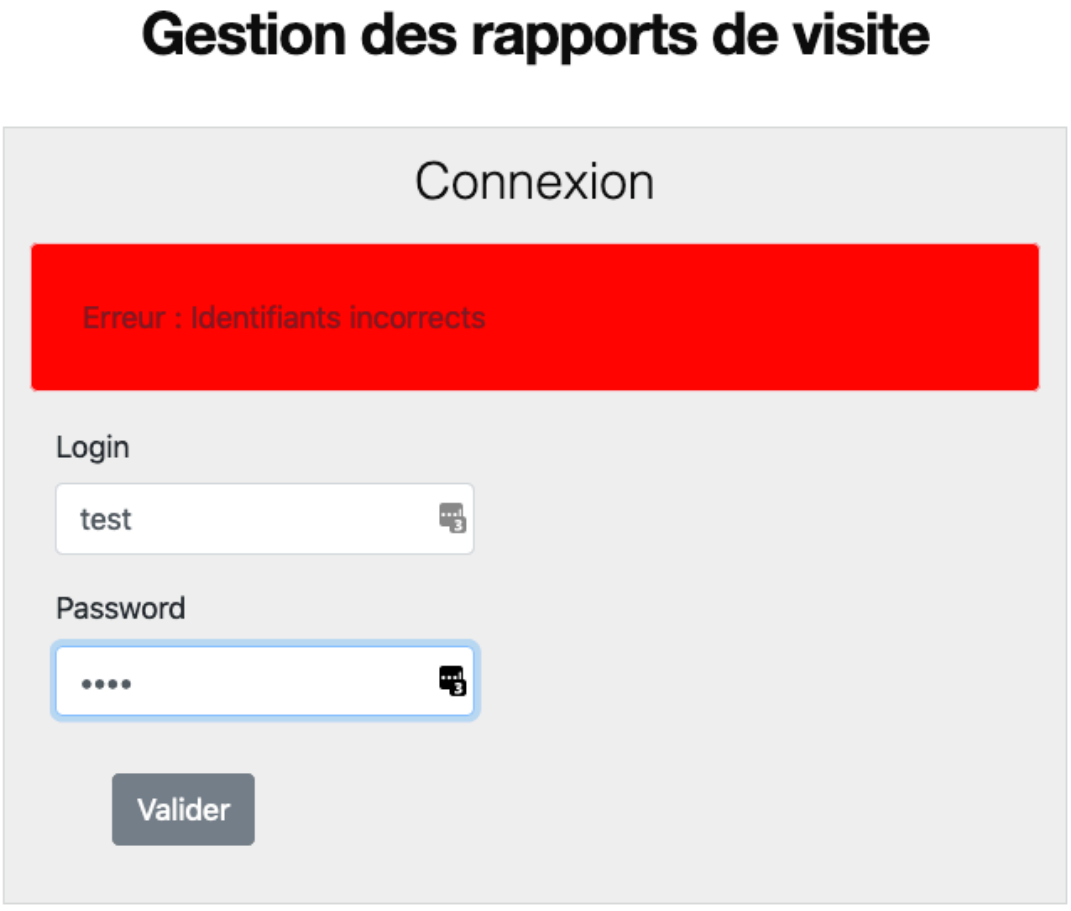
Voici les méthodes correspondantes dans le service **data.service.ts** :

```
public enregistrerRapport(idVisiteur: string, idMedecin: string, motif:
string, date: Date, bilan: string, lesMedicaments: Array<any>  ) {
  let url: string =  this.urlDomaine + '/nouveaurapport?idVisiteur=' +
idVisiteur + '&motif=';
  url += motif + '&bilan=' + bilan + '&idMedecin=' + idMedecin +
'&date=' + date;
  lesMedicaments.forEach((med) => {url += '&medicaments[' + med.id +
']=' + med.qte; } );
  const req = this.http
                  .get(url);
  return req;

}

public chargerMedicaments(nom: string) {
  const url: string =  this.urlDomaine + '/medicaments?nom=' + nom;
  const req = this.http
          .get(url);
  return req;
}
```

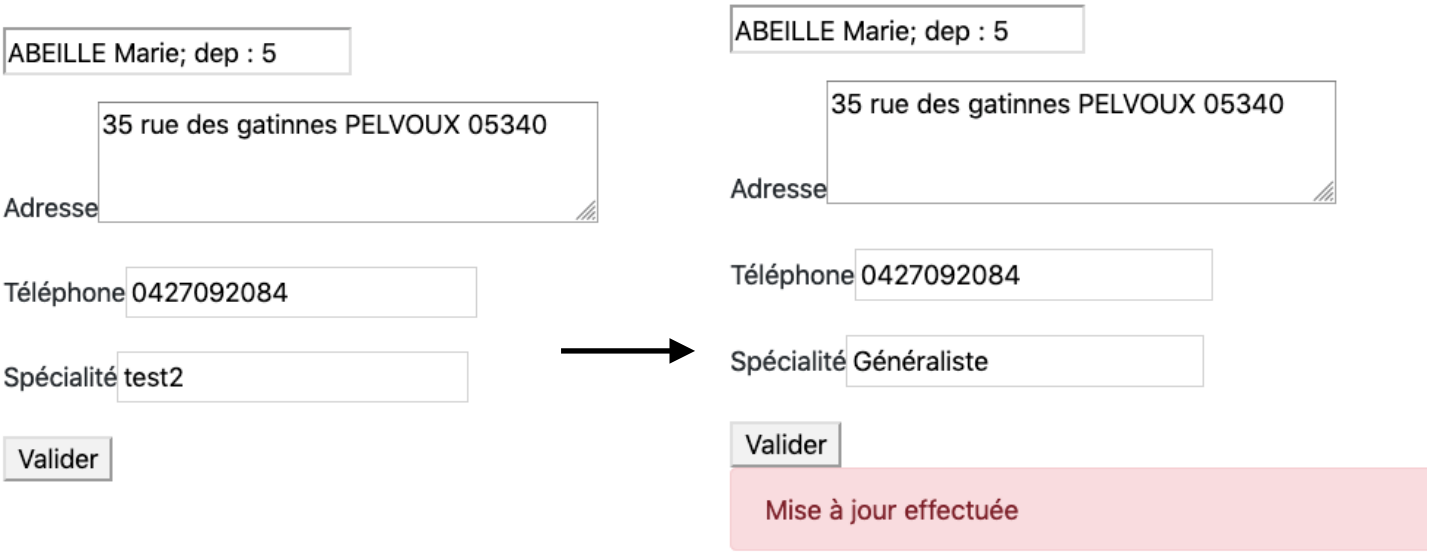
III. Extrait de l'application

A. INTERFACE DE CONNEXION

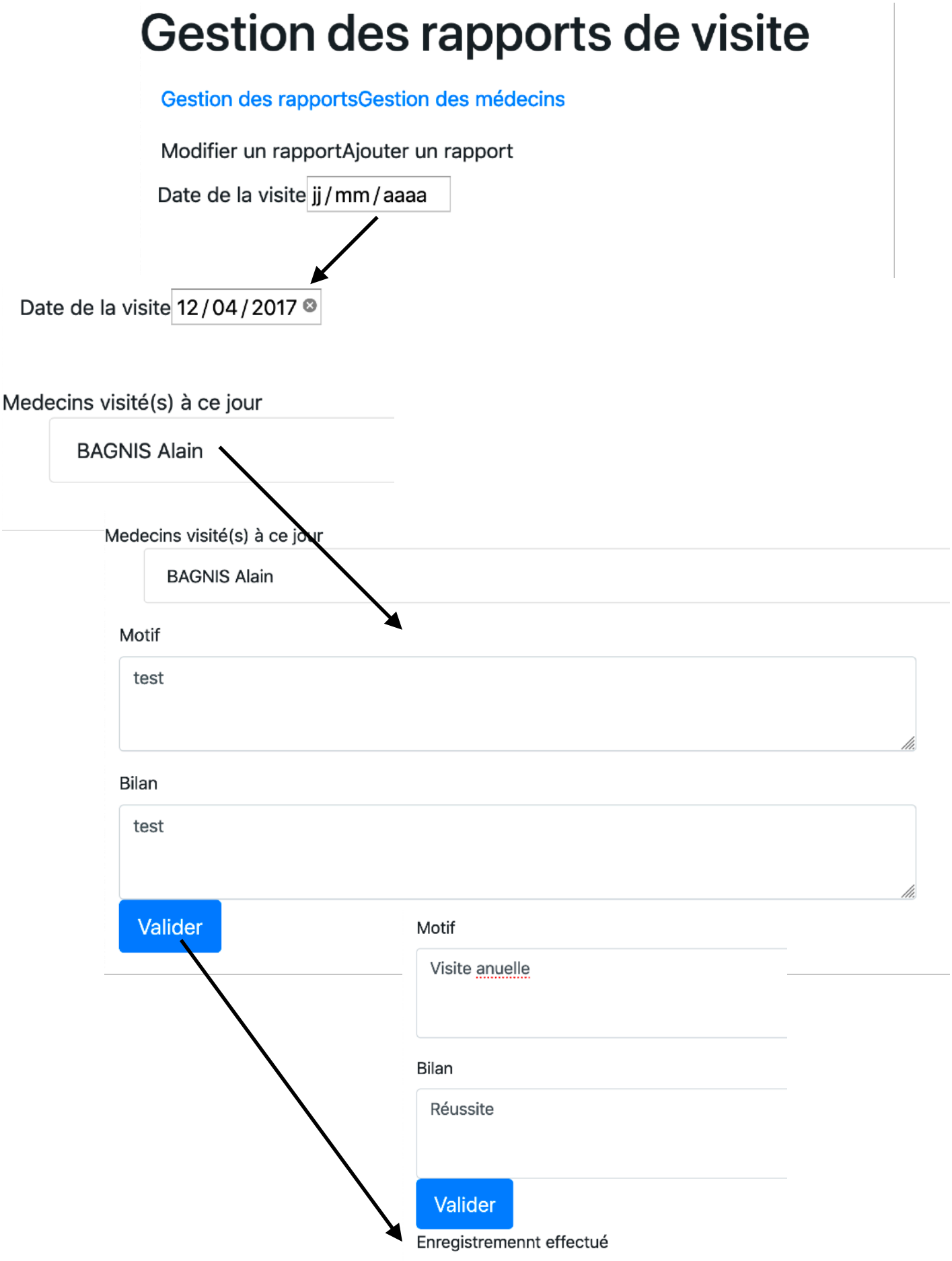




B. INTERFACE DE GESTION DES MÉDECINS



C. INTERFACE DE MODIFICATION DES RAPPORTS



D. INTERFACE D'AJOUT D'UN RAPPORT