Angular: les formulaires (1)

Traiter le formulaire en le référençant avec ses champs et en gérant sa soumission

- **NgForm** : lie automatiquement un **objet au formulaire** pour sa **validation** et son **traitement**
- **NgModel :** enregistre, dans Angular, **un champ dont elle est un attribut.** L'attribut **name** **doit être obligatoirement présent**
- **.value :**
  - propriété de l'objet lié au formulaire
  - objet ayant pour propriétés les noms des champs enregistrés
  - chaque nom de champ stocke la valeur du champ
- **NgSubmit : détecte la soumission du formulaire**
- **NB : FormsModule** est à importer depuis **@angular/forms** et doit être ajouté dans la section imports de **app.module.ts**

Angular: les formulaires (2)

exemple : ajouter un étudiant

- Générer le composant addStudent
- Dans le template de AddStudent, coder le formulaire

```
<form (ngSubmit)="handleAdd(myForm)" #myForm="ngForm" >
  <div> Prenom : <input name="prenom" ngModel ></div>
  <div> Nom : <input name="nom" ngModel > </div>
  <div> <button type="submit" >Ajouter</button> </div>
</form>
```

NB : #myForm="ngForm" assignation d'un objet NgForm à la variable locale myForm associée au formulaire

- Dans la classe de AddStudent, ajouter

```
@Output() studentToAdd:EventEmitter<any> = new EventEmitter();
handleAdd(myForm:NgForm){
  this.studentToAdd.emit(myForm.value);
  myForm.reset();
}
```

NB : importer NgForm à partir de @angular/forms

Angular: les formulaires (3)

Exemple : On veut créer une FORM qui contient le champ nom dans le composant Bonjour : « /bonjour », pour cela on ajoute au fichier « Bonjour.component.html » le code suivant :

bonjour.component.html U

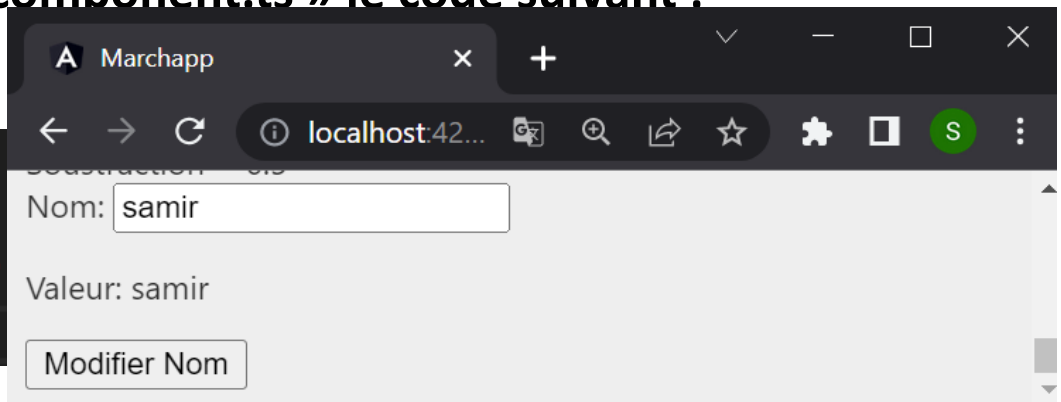
src > app > bonjour > bonjour.component.html > div > label

```
5    <div>
6    <label for="name">Nom: </label>
7    <input id="name" type="text" [formControl]="name">
8    <p>Valeur: {{ name.value }}</p>
9    <button type="button" (click)="updateName()">Modifier Nom</button>
10   </div>
```

On import : `import { FormControl } from '@angular/forms';`

Et on ajoute au fichier Bonjour.component.ts » le code suivant :

```
51   name = new FormControl('');
52     updateName() {
53     this.name.setValue('samir'
54   }
```



Angular :validation des formulaires (1)

Permet de vérifier la cohérence les données d'un formulaire avant de les sauvegarder dans la base de données et cela en définissant un ensemble de règle à respecter.

- Quelques **règles** de validation : **required**, **number**, **email**, **maxlength**, **minlength**, **pattern**
- Qlq **propriétés** de validation d'un **formulaire** ou d'un **champ**
  - **valid** : vraie si toutes les règles sont respectées
  - **invalid** : vraie si au moins une des règles est violée
  - **touched** : vraie si l'élément a été visité
  - **pristine** : vraie si l'élément n'a pas été modifié
  - **dirty** : vraie si l'élément a été modifié
  - **errors.<regle>** : vraie si la règle de validation est violée
- Syntaxes d'accès: **nomFormulaire.propriété** et **nomChamp.propriété** avec **#nomFormulaire="ngForm"** et **#nomChamp**="**ngModel**"
- Afficher tout message d'erreur avec la directive **NgIf**

Angular : validation des formulaires (2)

```
<form (ngSubmit)="handleAdd(myForm)" #myForm="ngForm" >
    <div>
        Prenom : <input name="prenom" ngModel required minlength="3" #prenom="ngModel" >
        <span style="color:red" *ngIf="prenom.invalid && (prenom.dirty || prenom.touched)
        ">
          <span *ngIf="prenom.errors.required">Le prenom est obligatoire</span>
          <span *ngIf="prenom.errors.minlength">Le prenom doit avoir au moins 3 lettres</
          span>
        </span>
    </div>
    <div>
        Nom : <input name="nom" ngModel required #nom="ngModel" >
        <span style="color:red" *ngIf="nom.invalid && (nom.dirty || nom.touched)">
          <span *ngIf="nom.errors.required">Le nom est obligatoire</span>
        </span>
    </div>
    <div>
      <button type="submit" [disabled]="myForm.invalid" >Ajouter</button>
    </div>
</form>
```

Angular : Services (1)

- **Pbl** : comment isoler certaines données et fonctionnalités réutilisables ?

Exemple : communiquer avec le backend

- **Sol** : enveloppées ces données et fonctionnalités réutilisables dans des classes détachées de l'interface
- **Service** : classe TypeScript décorée par **@Injectable** dont l'objectif est d'organiser et de **partager** le **code métier**
- Exemple de service natif : **HttpClient** pour exécuter des requêtes AJAX
- Commande de création : ***ng generate service*** *nomService*
- Il est fourni (provide), par défaut, à la « root injector » pour que sa réutilisation dans l'application soit globale
- **singleton** : instancié une seule fois et cette instance est ensuite utilisée partout

Angular : Services (2)

Communiquer avec un serveur: présentation de HttpClient

- **HttpClient** : service qui facilite la communication avec un serveur **HTTP** distant via l'objet **XMLHttpRequest**
- **HttpClient** dispose des méthodes **get()**, **post()**, **put()**, **patch()** , **delete()** pour effectuer des requêtes HTTP
- Chacune des méthodes construit un « Observable » qui attend un abonnement (**.subscribe()**) pour exécuter la requête
- **get()** permet de retourner le type de l'objet attendu avec la syntaxe suivante : .**get<***typeAttendu***>()**
- **Prérequis** d'utilisation de HttpClient
  - **Importer** HttpClientModule dans **AppModule**
  - **Injecter** HttpClient dans un service de l'application
  - HttpClientModule et HttpClient sont dans **@angular/common/http**

Etude de cas 1:

Gestion des utilisateurs Frontend

Gestion des utilisateurs Frontend

Création des modules pour notre application :

1- Générer les composants suivants:

- **ng generate component Userr1**
- **ng generate component add-userr**

2- Générer le service suivant:

- **ng generate service services/httpClient**

3- Générer le modèle Userr1:

- **ng g class models/Userr1 --type=model**

Création des modules pour notre application :

```
C:\WINDOWS\system32\cmd.exe

C:\Users\x\Documents\micro-frontend-exemple\marchapp>ng generate component Userr1
CREATE src/app/userr1/userr1.component.html (21 bytes)
CREATE src/app/userr1/userr1.component.spec.ts (626 bytes)
CREATE src/app/userr1/userr1.component.ts (275 bytes)
CREATE src/app/userr1/userr1.component.css (0 bytes)
UPDATE src/app/app.module.ts (640 bytes)

C:\Users\x\Documents\micro-frontend-exemple\marchapp>
C:\Users\x\Documents\micro-frontend-exemple\marchapp>ng generate component add-userr
CREATE src/app/add-userr/add-userr.component.html (24 bytes)
CREATE src/app/add-userr/add-userr.component.spec.ts (641 bytes)
CREATE src/app/add-userr/add-userr.component.ts (286 bytes)
CREATE src/app/add-userr/add-userr.component.css (0 bytes)
UPDATE src/app/app.module.ts (732 bytes)

C:\Users\x\Documents\micro-frontend-exemple\marchapp>
C:\Users\x\Documents\micro-frontend-exemple\marchapp>ng generate service services/httpClient
CREATE src/app/services/http-client.service.spec.ts (378 bytes)
CREATE src/app/services/http-client.service.ts (139 bytes)

C:\Users\x\Documents\micro-frontend-exemple\marchapp>
C:\Users\x\Documents\micro-frontend-exemple\marchapp>ng g class models/Userr1 --type=model
CREATE src/app/models/userr1.model.spec.ts (160 bytes)
CREATE src/app/models/userr1.model.ts (24 bytes)
```

Etude de cas 1: Paramétrage app.module.ts

TS app.module.ts M

src > app > TS app.module.ts > AppModule

```ts
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { BonjourComponent } from './bonjour/bonjour.component';
import { ReactiveFormsModule } from '@angular/forms';
import { Userr1Component } from './userr1/userr1.component';
import { AddUserrComponent } from './add-userr/add-userr.component';
import { NgbModule } from '@ng-bootstrap/ng-bootstrap';

@NgModule({
  declarations: [
    AppComponent,
    BonjourComponent,
    Userr1Component,
    AddUserrComponent
  ],
  imports: [
    BrowserModule,
    HttpClientModule,
    AppRoutingModule,
    ReactiveFormsModule,
    FormsModule,
    NgbModule
```

Etude de cas 1: Création du modèle Userr1

TS userr1.model.ts U

src > app > models > TS userr1.model.ts > ...

```
1
2   export class Userr1 {
3       constructor(
4           public  userName:string,
5           public  nom:string,
6           public  prnom:string,
7           public  dateDeNaissance: Date,
8           public  lieuDeNaissance:string,
9           public  motPasse:string,
10          public  role:number,
11          public  numService:number,
12          public  numUser:number ) {}
13      }
```

Etude de cas 1: Création du contrôleur Userr1

TS userr1.component.ts ●

src > app > userr1 > TS userr1.component.ts > Userr1Component > handleSuccessfulResponse

```
11  @Injectable()
12  export class Userr1Component implements OnInit {
13    @Input() userrs:Userr1[];
14    RoleList:any[];
15     x: boolean= false;
16     user: Userr1 = new Userr1(null,null, null, null, null, null, null, null, null);
17    constructor( private httpClientService:HttpClientService, private router: Router) { }
18
19    ngOnInit() {
20      this.httpClientService.getUserrs().subscribe( response =>this.handleSuccessfulResponse(response));
21      this.httpClientService.getRoles().subscribe(courrierras => this.RoleList = courrierras);
22    }
23    deleteUserr(employee: Userr1): void {
24      this.httpClientService.deleteUserr(employee)
25        .subscribe( data => {
26          this.userrs = this.userrs.filter(u => u !== employee);
27        })
28    };
29    updateUserr(): void {
30
31      this.httpClientService.updateUserr(this.user.numUser, this.user )
32      .subscribe( data => {
33        alert("User updated successfully."); });
34    };
35
36    editUserr(employee: Userr1){
37      this.x=!(this.x);
38      this.user=employee;    }
39
40  handleSuccessfulResponse(response)
41  {
42      this.userrs=response;}
```

Etude de cas 1: Création de la vue Userr1 (1)

-Afficher la liste des utilisateur

userr1.component.html

src > app > userr1 > userr1.component.html > div.col-md-6

```
<div class="col-md-6">
  <h2 class="text-center">Liste des Utilisateurs</h2>
<table class="table table-striped">
    <thead>
    <tr>
        <th>Numero</th>
        <th>Nom Utilisateur</th>
      <th>Direction</th>
      <th>role</th>
    </tr>
    </thead>
    <tbody>
      <tr *ngFor="let userr1 of userrs">
        <td>{{userr1.numUser}}</td>
        <td>{{userr1.userName}}</td>
        <td>{{getEntiteLib(userr1.numService)}}</td>
        <td>{{userr1.role}}</td>
        <td><button class="btn btn-danger"  (click)="deleteUserr(userr1)">Delete </button></td>
                <td><button class="btn btn-success" (click)="editUserr(userr1)">Update</button></td>

              </tr>
    </tbody>
      </table>
</div>
```

Etude de cas 1: Création de la vue Userr1 (2)

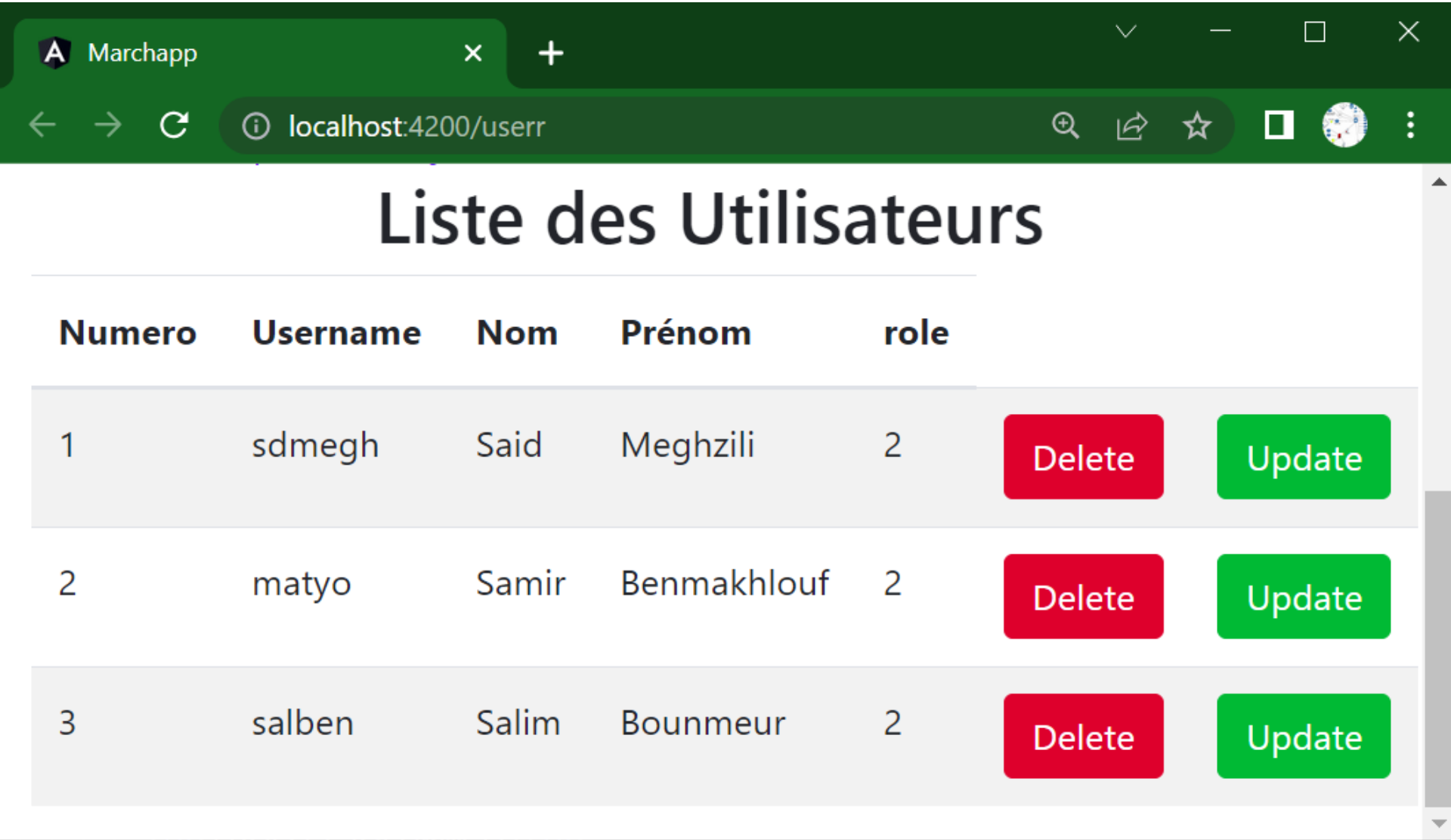
-Modifier un utilisateur

userr1.component.html

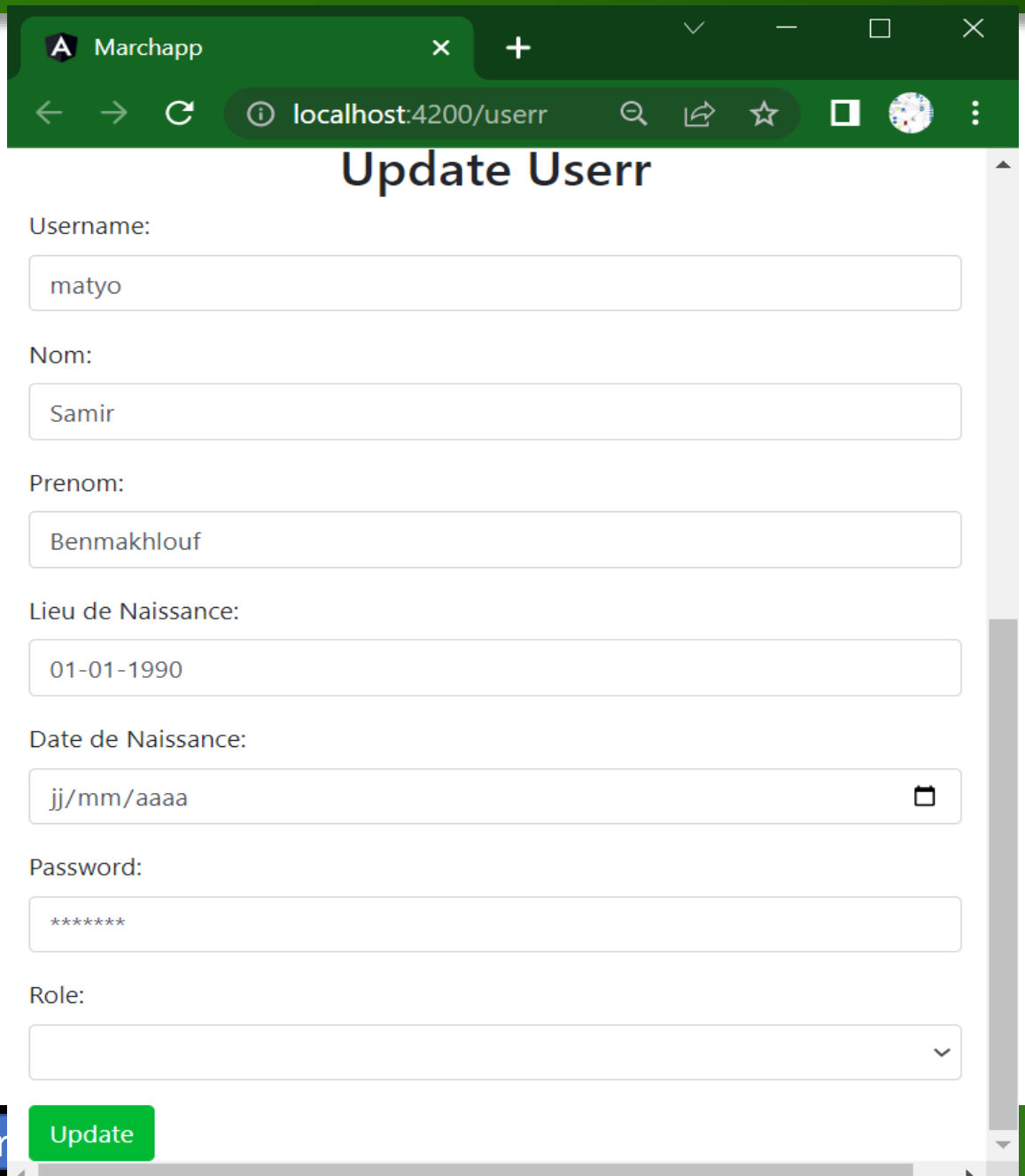
src > app > userr1 > userr1.component.html > div.col-md-6 > form > div.form-group > select#role.form-control

```
26  <div class="col-md-6"  *ngIf="x">  <h2 class="text-center">Update Userr</h2> <form>
27      <div class="form-group">
28        <label for="userName">Username:</label>
29        <input  type="text "[(ngModel)]="user.userName" name="user.userName"   class="form-control">
30      </div>
31      <div class="form-group">
32      <label for="nom">Nom:</label>
33      <input  [(ngModel)]="user.nom"  name="user.nom" class="form-control" id="name">
34    </div>
35    <div class="form-group">
36      <label for="prnom">Prenom:</label>
37      <input  [(ngModel)]="user.prnom"  name="user.prnom" class="form-control" id="prnom">
38    </div>
39    <div class="form-group">
40      <label for="lieuDeNaissance">Lieu de Naissance:</label>
41      <input  [(ngModel)]="user.lieuDeNaissance"  name="user.lieuDeNaissance" class="form-control" >
42    </div>
43    <div class="form-group">
44      <label for="dateDeNaissance">Date de Naissance:</label>
45      <input type="date" [(ngModel)]="user.dateDeNaissance"  name="user.dateDeNaissance" class="form-control">
46    </div>
47    <div class="form-group">
48      <label for="motPasse">Password:</label>
49      <input [(ngModel)]="user.motPasse"  name="user.motPasse" class="form-control" id="motPasse">
50    </div>
51    <div class="form-group">
52      <label for="role">Role:</label>
53      <select  class="form-control"  name="user.role" [(ngModel)]="user.role" id="role">
54        <option disabled>Sélectionner Role</option>
55        <option *ngFor="let r of RoleList"   [value]="r.id">{{r.libelle}}</option>
56      </select>
57    </div> <button class="btn btn-success" (click)="updateUserr()">Update</button> </form> </div>
```

Etude de cas 1: Création de la vue Userr1 (3)



Etude de cas 1: Modification d'un utilisateur



Etude de cas 1: Création du contrôleur add-User

add-userr.component.ts ●

src > app > add-userr > TS add-userr.component.ts > AddUserrComponent

```ts
import { Component, OnInit } from '@angular/core';
import { HttpClientService, Userr1, Entite  } from '../service/http-client.service';
@Component({
  selector: 'app-add-userr',
  templateUrl: './add-userr.component.html',
  styleUrls: ['./add-userr.component.css']
})
export class AddUserrComponent implements OnInit {

  RoleList: any[];
  user: Userr1 =new Userr1(null, null, null, null, null, null, null, null,null);
  constructor(private httpClientService: HttpClientService) { }

  ngOnInit(): void {

    this.httpClientService.getRoles().subscribe(courrierras => this.RoleList = courrierras);

  }

  createUserr(): void {
    this.httpClientService.createUserr(this.user)
        .subscribe( data => {
          alert("User created successfully.");  }); };}
```

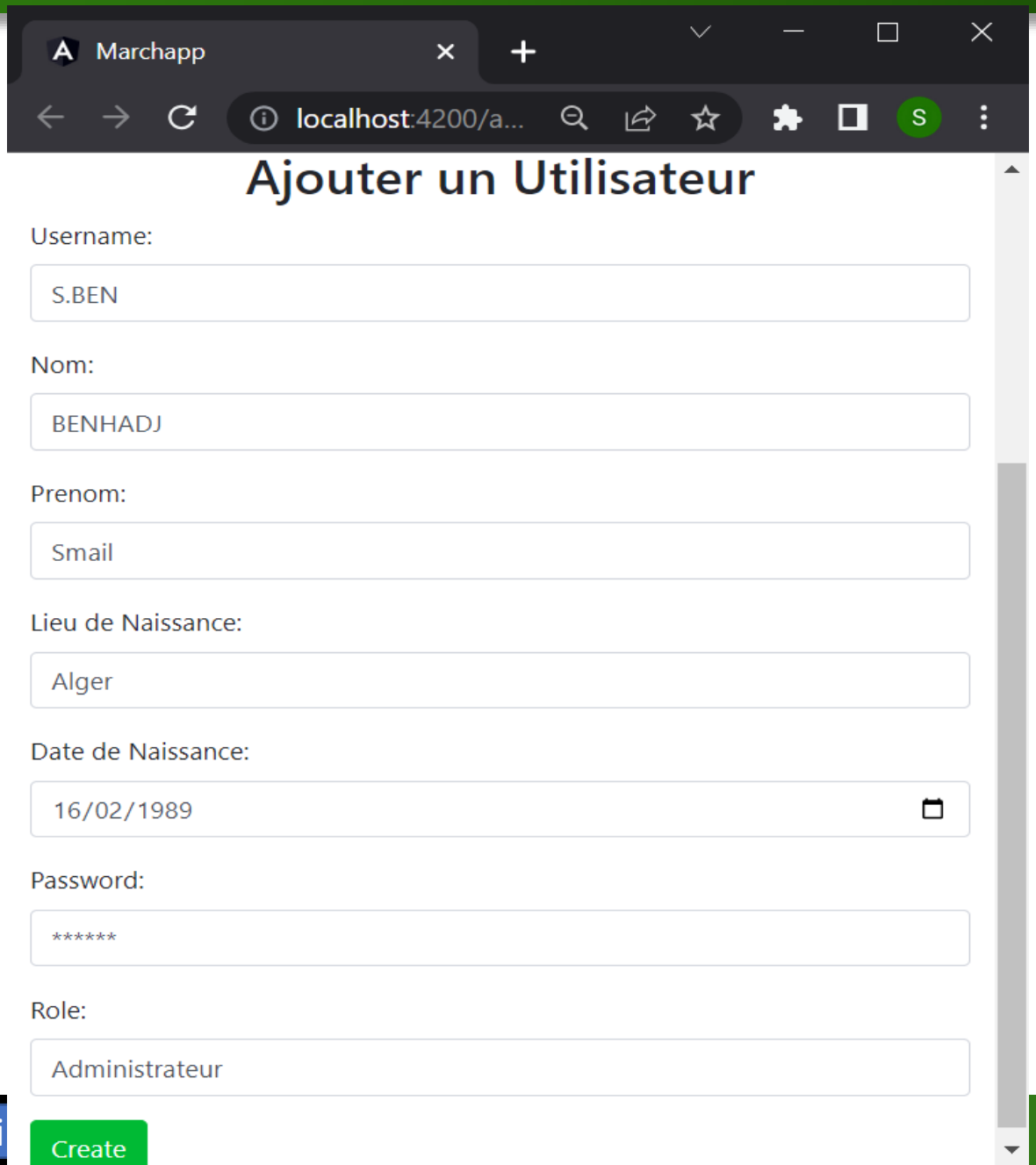
Etude de cas 1: Création de la vue add-User (1)

add-userr.component.html ●

src > app > add-userr > add-userr.component.html > div.col-md-6

```
<div class="col-md-6">     <h2 class="text-center">Ajouter un Utilisateur</h2>
    <form>
      <div class="form-group">
        <label for="userName">Username:</label>
        <input type="name" [(ngModel)]="user.userName"  name="name" class="form-control" id="username">
      </div>
      <div class="form-group">
      <label for="nom">Nom:</label>
      <input type="name" [(ngModel)]="user.nom"  name="name" class="form-control" id="name">
    </div>
    <div class="form-group">
      <label for="prnom">Prenom:</label>
      <input type="name" [(ngModel)]="user.prnom"  name="name" class="form-control" id="prnom">
    </div>
    <div class="form-group">
      <label for="lieuDeNaissance">Lieu de Naissance:</label>
      <input type="name" [(ngModel)]="user.lieuDeNaissance"  name="name" class="form-control" id="lieuDeNaissance">
    </div>
    <div class="form-group">
      <label for="dateDeNaissance">Date de Naissance:</label>
      <input type="date" [(ngModel)]="user.dateDeNaissance"  name="name" class="form-control" id="dateDeNaissance">
    </div>
    <div class="form-group">
      <label for="motPasse">Password:</label>
      <input type="name" [(ngModel)]="user.motPasse"  name="name" class="form-control" id="motPasse">
    </div>
    <div class="form-group">
      <label for="role">Role:</label>
      <input type="name" [(ngModel)]="user.role"  name="name" class="form-control" id="role">
    </div>
      <button class="btn btn-success" (click)="createUserr()">Create</button> </form> </div>
```

Etude de cas 1: Création de la vue add-User (2)



Etude de cas 1: Création du service HttpclientService

Permet la connexion avec l'application back-end

```
export class HttpClientService {
  private url = 'http://Www…….';

  constructor( private httpClient:HttpClient) { }

getUserrs()
{
  return this.httpClient.get<Userr1[]>(url);
}

public deleteUserr(employee) {
  return this.httpClient.delete<Userr1>(url + "/"+ employee.numUser);
}

public createUserr(employee) {
  return this.httpClient.post<Userr1>(url, employee);

}
public updateUserr(id,employee) {
return this.httpClient.put<Userr1>(url+ "/"+id,employee);
```

Voila comment inclure Bootstrap 5 dans les applications Angular 15 à l'aide de la bibliothèque ng-bootstrap:

npm install bootstrap

npm i @ng-bootstrap/ng-bootstrap --legacy-peer-deps

npm uninstall @ng-bootstrap/ng-bootstrap

ng add @ng-bootstrap/ng-bootstrap