Client Angular 5 du Cabinet médical	**Développement Web avancé** *M2 – MIASHS* A. *Demeure – année 2017/2018*	

1) Création d'un client Angular 2

Nous allons créer un client Angular 2 à partir de NPM et d'Angular-CLI. La procédure varie un peu suivant que vous installez sur une machine pour laquelle vous avez les droits administrateurs ou non.

Pour une machine où vous avez les droits administrateurs, vous pouvez installez la CLI Angular en mode global (elle sera accessible pour tous vos projets :

```
npm install --save-dev @angular/cli
```

Vous disposez alors de la CLI Angular sous le forme de la commande ng et vous pouvez suivre les étapes suivantes :

1. Placez vous dans le répertoire où vous souhaitez installer votre projet
   (Prenez donc un répertoire vide)
2. Initialiser un nouveau projet avec npm :
   ```
   npm init
   ```
   Répondez aux questions posées, si vous n'êtes pas inspiré appuyez sur ENTRER.
3. Puis passez au point 5 de la liste suivante en utilisant la commande
   ```
   ng
   ```
   À la place de
   ```
   node ./node_modules/@angular/cli/bin/ng
   ```

Si vous êtes sur une machine pour laquelle vous n'avez pas les droits administrateurs, alors suivez les étapes suivantes :

1. Placez vous dans le répertoire où vous souhaitez installer votre projet
   (Prenez donc un répertoire vide)
2. Initialiser un nouveau projet avec npm :
   ```
   npm init
   ```
   Répondez aux questions posées, si vous n'êtes pas inspiré appuyez sur ENTRER.
3. Installez ensuite la Angular-CLI, cela nous servira a :
   ```
   npm install --save-dev @angular/cli
   ```
4. Renommez le fichier package.json en package.json.old.
   Nous faisons cela pour que la CLI Angular ne pense pas qu'un projet existe déjà.
5. Initiez un projet avec Angular-CLI :
   ```
   node ./node_modules/@angular/cli/bin/ng new clientAngular
   ```
   Cela crée un sous répertoire clientAngular qui contient le squelette d'une application.
6. Pour pouvoir développer facilement avec le serveur de développement Angular nous allons configurer un proxy pour que l'application Angular puisse communiquer facilement avec le serveur.

a. Placez vous dans le répertoire clientAngular
b. Créez un fichier **proxy.conf.json**
c. Dans ce fichier, placez le contenu suivant (vous pourrez l'enrichir d'autres indirections si besoin). Ce sont les ressources que vous allez implémenter dans votre serveur pour communiquer avec votre client:

```
{
 "/getCabinetData": {
   "target": "http://localhost:8080",
   "secure": false
 },
 "/addPatient": {
   "target": "http://localhost:8080",
   "secure": false
 },
 "/affectation": {
   "target": "http://localhost:8080",
   "secure": false
 }
}
```

d. Dans le fichier package.json, modifiez le script start pour prendre en compte les indirections définies dans le proxy:
`"start": "ng serve --proxy-config proxy.conf.json"`

7. Créez un service Angular qui servira à communiquer avec le serveur :
`node ./node_modules/@angular/cli/bin/ng generate service cabinet-medical --module app`
8. Créez un premier composant pour la secrétaire :
`node ./node_modules/@angular/cli/bin/ng generate component secretary`

3) Mise en place de la communication avec le serveur

Nous allons définir le modèle de données que nous allons utiliser dans le projet pour représenter les informations du cabinet médical. Créer un répertoire **src/app/dataInterfaces** dans lequel nous allons placer cinq fichiers :

- sexe.ts

```
export enum sexeEnum {M, F}
```

- adress.ts

```
export interface Adresse {
 ville: string;
 codePostal: number;
 rue: string;
 numéro: string;
 étage: string;
 lat: number;
 lng: number;
```

```
}
```

- patient.ts

```
import {sexeEnum} from "./sexe";
import {Adresse} from "./adress";

export interface PatientInterface {
 prénom: string;
 nom: string;
 sexe: sexeEnum;
 numéroSécuritéSociale: string;
 adresse: Adresse;
}
```

- nurse.ts

```
import {PatientInterface} from "./patient";
import {Adresse} from "./adress";

export interface InfirmierInterface {
 id: string;
 prénom: string;
 nom: string;
 photo: string;
 patients: PatientInterface[];
 adresse: Adresse;
}
```

- cabinet.ts

```
import {InfirmierInterface} from "./nurse";
import {PatientInterface} from "./patient";
import {Adresse} from "./adress";

export interface CabinetInterface {
 infirmiers: InfirmierInterface[];
 patientsNonAffectés: PatientInterface  [];
 adresse: Adresse;
}
```

Maintenant, nous allons éditer le service /src/app/cabinet-medical.service.ts et le composant secretary (Typescript et HTML).

Le but du service sera :

- D'obtenir les données stockés au format XML sur le serveur (ressource /data/cabinetInfirmier.xml) et de les traduire au format de données Typescript exposé ci-dessus.
- De communiquer avec le serveur des demandes de création et d'affectation des patients (voir la partie A: Comprendre le serveur)

Question 0) Initialisation

Allez dans le composant secretary (fichier `src/app/secretary/secretary.component.ts`) et ajouter à son constructeur une dépendance au service cabinet-medical (cela revient à ajouter un paramètre de type CabinetMedicalService au constructeur du composant).

Ajouter une instance du composant secretary dans le template du composant "principal" (fichier `src/app/app.component.html`).

Question 1) Obtenir et convertir les données.

Implémentons une méthode **`getData`** qui renvoie une Promesse d'instance de CabinetInterface

```
getData( url: string ): Promise<CabinetInterface>
```

Indications :

- Faites un appel à la ressource **HTTP GET /getCabinetData** en utilisant le service Http fourni par Angular.
  ```
  import {Http} from '@angular/http';
  ```
  Spécifiez ensuite un paramètre de type Http pour le constructeur de votre composant secretary pour pouvoir réaliser des requêtes HTTP.
  Vous pouvez alors faire des requêtes HTTP GET avec la méthode get de l'instance du service Http passé à votre constructeur.
- Dans app.module.ts :
  import {HttpModule} from "@angular/http"
  Et ajouter HttpModule dans le tableau des imports.
- Une fois les données reçu via la requête HTTP GET, convertissez les au format demandé.

Question 2) Composants Infirmier et Patient

Utilisez Angular CLI pour créer deux nouveaux composants : Infirmier et Patient.
Prenez le temps de réfléchir à la façon dont vous allez utiliser ces composants pour concevoir l'interface du secrétaire médical.

Pour cela, vous pouvez utiliser les bibliothèques suivantes :

- Angular Material : un ensemble de composants de haut niveaux.
- alx-dragdrop : Une bibliothèque pour la gestion du drag&drop.

Question 3) Intégration de Google Map

Pour visualiser où se trouvent les patients, les infirmiers et le cabinet médical, nous allons utiliser une carte Google à l'aide du composant AGM.

```
npm install --save @agm/core
```