DÉVELOPPEMENT DES APPLICATIONS MOBILES HYBRIDES MULTIPLATEFORMES

CH 4 - Utilisation des composants Ionic

FORMATEUR : MAHAMANE SALISSOU YAHAYA : ysalissou@gmail.com

(TechnoLAB -ISTA)

4ième année TEL

OCTOBRE 2021

Ionic est constitué de blocs d'éléments de haut niveau appelés composants. Les **composants** vous permettent de créer rapidement de belles interfaces graphiques pour votre application.

Le framework propose pléthore de composants, du bouton d'action au toast, en passant par des listes d'éléments, soit suffisamment d'UI pour développer à peu près tout type d'application.

Chaque composant à sa propre API, ce qui permet de l'exploiter au maximum. Étudions quelques-uns d'entre eux qui nous seront bien utiles pour la suite de notre projet.

Pour information, la liste de tous les composants actuellement disponibles se trouve à l'adresse https://ionicframework.com/docs/components/. N'hésitez pas à la consulter et faire un tour complet de la documentation.

Composant Bouton

OCTOBRE 2020

Pour ajouter un composant de type bouton à votre application mobile, il suffit simplement de faire...attention, très grand moment ... :

Voilà. C'est tout.

```
8
9     <ion-button>Mon bouton</ion-button>
```

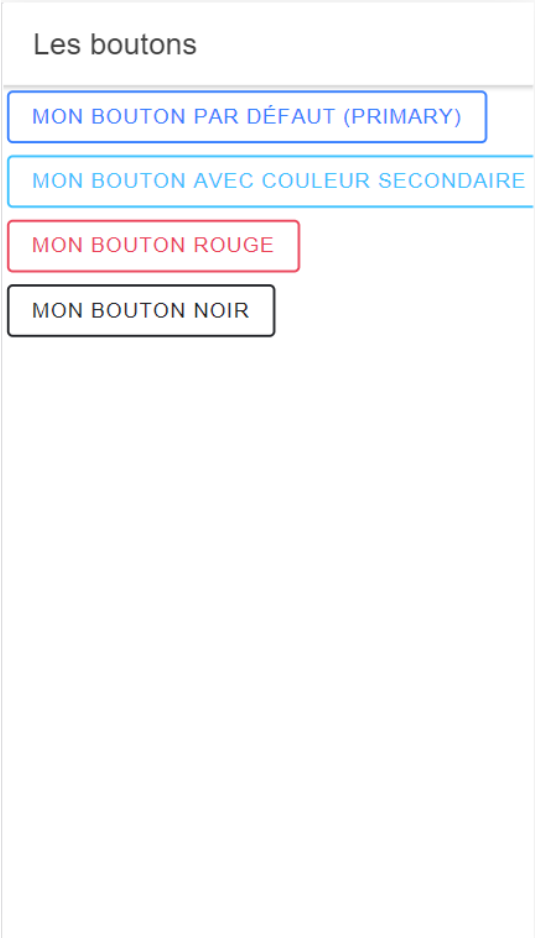
Il est possible de customiser un bouton grâce à de nombreuses **directives**, concept abordé au chapitre 6 :

```
<ion-content>
  <!-- Composant Bouton -->
  <h2 class="ion-text-center">Composant Bouton</h2>
  <ion-button>Mon bouton</ion-button>
  <!-- Un bouton avec un lien -->
  <ion-button href="/monlien">Un Lien fort</ion-button>
  <!-- Un bouton avec la couleur primary -->
  <ion-button color="primary">Couleur primary</ion-button>
  <!-- Un bouton qui occupe toute la largeur -->
  <ion-button expand="full">Bouton sur toute la largeur</ion-button>
  <!-- Un bouton arrondi -->
  <ion-button shape="round">Bouton rond</ion-button>
    <!-- Un bouton avec une icône à gauche -->
  <ion-button><ion-icon slot="start" name="star"></ion-icon>Une icone de gauche</ion-button>
  <!-- Un bouton avec une icône à droite -->
  <ion-button>   Une icone de droite<ion-icon slot="end" name="star"></ion-icon></ion-button>
  <!-- Un bouton avec une icône uniquement -->
  <ion-button><ion-icon slot="icon-only" name="star"></ion-icon></ion-button>
  <!-- Fill -->
  <ion-button expand="full" fill="outline">Sans couleur de fond + Sur toute la largeur</ion-button>
  <ion-button expand="block" fill="outline">Sans couleur de fond + Bloc en Pleine largeur</ion-button>
  <ion-button shape="round" fill="outline">Sans couleur de fond + Rond</ion-button>
  <!-- différentes tailles de boutons -->
  <ion-button size="large">Taille large</ion-button>
  <ion-button>Taille par défaut</ion-button>
  <ion-button size="small">Petite taille</ion-button>
</ion-content>
```

on peut ainsi retirer le background pour n'afficher que la bordure dans la couleur définie grâce à la **directive fill="outline"** :

```
<ion-content>
  <ion-button fill="outline">Mon bouton par défaut (primary)</ion-button>
  <ion-button color="secondary" fill="outline">Mon bouton avec couleur secondaire</ion-button>
  <ion-button color="danger" fill="outline">Mon bouton rouge</ion-button>
  <ion-button color="dark" fill="outline">Mon bouton noir</ion-button>

</ion-content>
```



Ou encore retirer les bordures du boutons avec la directive **clear** :

```
6   <ion-content padding>
7
8     <ion-button clear>Mon bouton par défaut (primary)</ion-button>
9     <ion-button color="secondary" clear>Mon bouton avec couleur secondaire</ion-button>
0     <ion-button color="danger" clear>Mon bouton rouge</ion-button>
1     <ion-button color="dark" clear>Mon bouton noir</ion-button>
2
3   </ion-content>
```

Pour plus d'information, vous pouvez consulter la documentation Ionic à l'adresse :
https://ionicframework.com/docs/api/button

Composant Liste

OCTOBRE 2020

Comme son nom peut le suggérer, ce composant va nous permettre d'ajouter une liste d'éléments à notre application.

C'est ce composant qui est utilisé dans l'écran Rappels de l'application Trombinoscope, et celui qui servira à afficher nos listes d'étudiants.

```
<ion-content>
  <ion-list>
    <ion-item *ngFor="let item of items">
      <ion-icon [name]="item.photo" slot="start"></ion-icon>
      {{item.id}}
      <div class="item-etud" slot="end">
        {{item.nom}} {{item.prenom}}
      </div>
    </ion-item>
  </ion-list>
</ion-content>
```

```
private photo = [
  'flask',
  'wifi',
  'beer',
  'football',
  'basketball',
  'paper-plane',
  'american-football',
  'boat',
  'bluetooth',
  'build'
];
public items: Array<{ id: string; nom: string; prenom: string; photo: string }> = [];
constructor() {
  for (let i = 1; i < 11; i++) {
    this.items.push({
      id: 'Matricule ' + i,
      nom: 'Moussa ' + i,
      prenom: 'DIARRA ' + i,
      photo: this.photo[Math.floor(Math.random() * this.photo.length)]
    });
  }
}
```

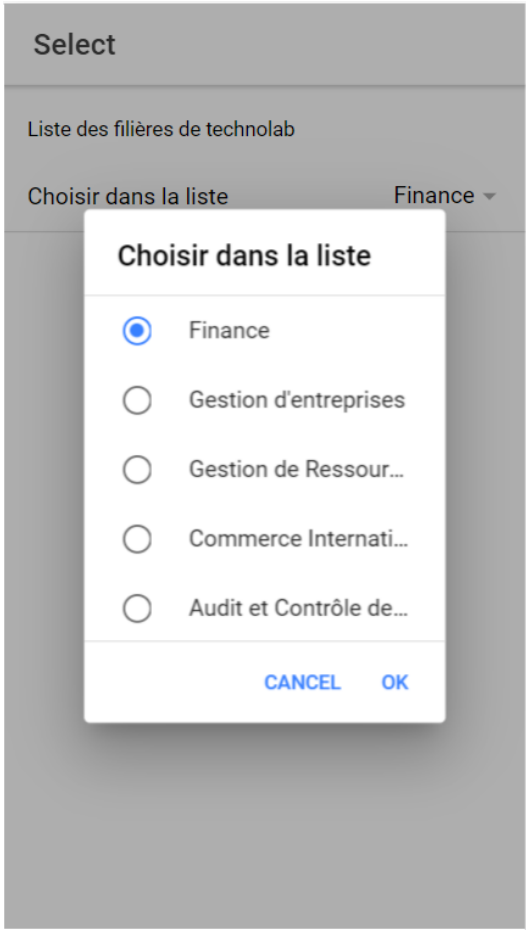
Documentation : https://ionicframework.com/docs/components/#lists

Composant Select

OCTOBRE 2020

Ce composant permet un rendu du tag html **<select></select>** et va donc nous permettre d'afficher une liste de choix. Affichons par exemple ici la liste des filières de Technolab:

```
<ion-header>
  <ion-toolbar>
    <ion-title>Select</ion-title>
  </ion-toolbar>
</ion-header>
<ion-content padding>
  <ion-list>
    <ion-list-header text-center>
      Liste des filières de technolab
    </ion-list-header>
    <ion-item>
      <ion-label>Choisir dans la liste</ion-label>
      <ion-select value="fn">
        <ion-select-option value="fn">Finance</ion-select-option>
        <ion-select-option value="ge">Gestion d'entreprises</ion-select-option>
        <ion-select-option value="rh">Gestion de Ressources Humaines</ion-select-option>
        <ion-select-option value="ci">Commerce International</ion-select-option>
        <ion-select-option value="acg">Audit et Contrôle de Gestion</ion-select-option>
      </ion-select>
    </ion-item>
  </ion-list>
</ion-content>
```



Documentation : https://ionicframework.com/docs/api/select

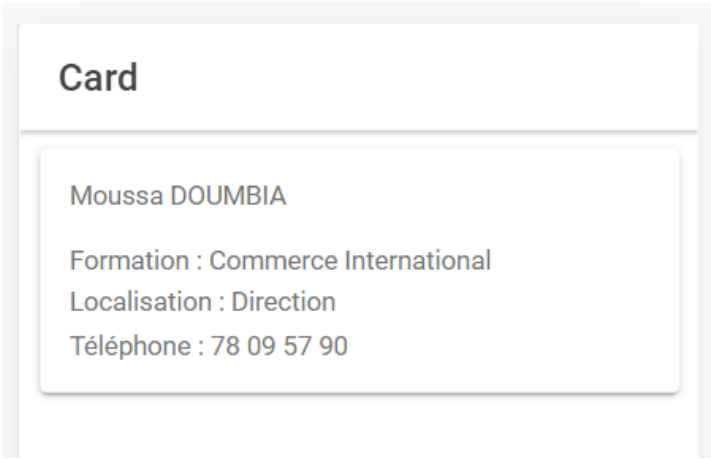
Composant Cards

OCTOBRE 2020

Les cartes sont un bon moyen d'afficher des informations importantes à destination des utilisateurs. Ce pattern s'inspire des cartes de visite personnelles ou professionnelles que nous utilisons dans la vie courante :

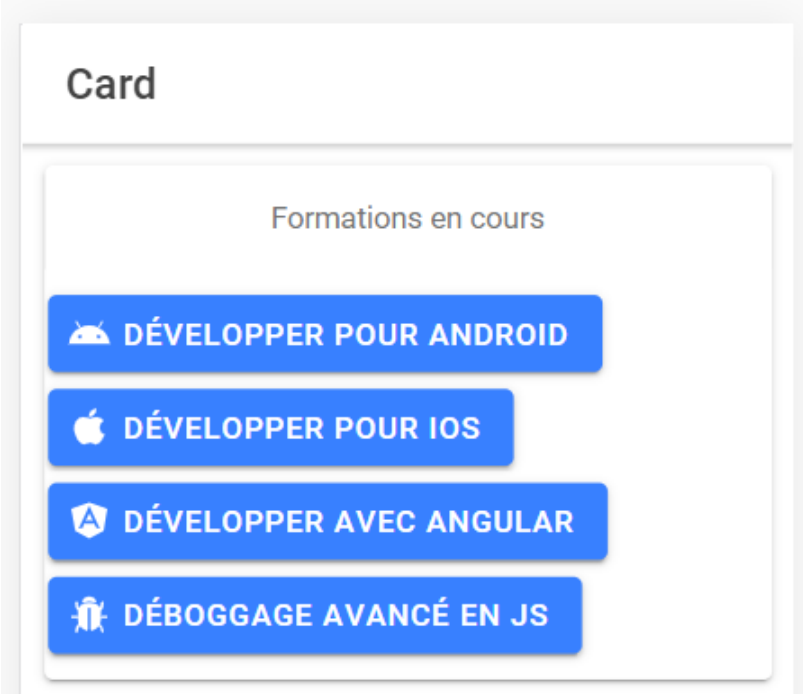
```
<ion-header>
  <ion-toolbar>
    <ion-title>Card</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-card>
    <ion-card-header>
      Moussa DOUMBIA
    </ion-card-header>
    <ion-card-content>
      <p>Formation : Commerce International</p>
      <p>Localisation : Direction</p>
      <p>Téléphone : 78 09 57 90</p>
    </ion-card-content>
  </ion-card>
</ion-content>
```



Il est également possible de combiner carte et liste d'éléments comme ceci :

```
<ion-content>
  <ion-card>
    <ion-card-header class="ion-text-center">
      Formations en cours
    </ion-card-header>
    <ion-list>
      <ion-button>
        <ion-icon name="logo-android" slot="start"></ion-icon>
        Développer pour Android
      </ion-button>
      <ion-button>
        <ion-icon name="logo-apple" slot="start"></ion-icon>
        Développer pour iOS
      </ion-button>
      <ion-button>
        <ion-icon name="logo-angular" slot="start"></ion-icon>
        Développer avec Angular
      </ion-button>
      <ion-button>
        <ion-icon name="bug" slot="start"></ion-icon>
        Déboggage avancé en JS
      </ion-button>
    </ion-list>
  </ion-card>
</ion-content>
```



Ou tout simplement reproduire un design assez proche de réseaux sociaux comme Instagram :

```
<ion-card>
  <img src="assets/imgs/moussa.png" />
  <ion-card-content>
    <ion-card-title>
      Moussa SISSOKO
    </ion-card-title>
    <p>
      Moussa Sissoko, né le 16 août 1989 au Blanc-Mesnil, en Seine-Saint-Denis, est un footballeur international
      français qui évolue au poste de milieu de terrain à Tottenham Hotspur. Avec la sélection, il participe à la
      Coupe du monde 2014 et à l'Euro 2016 dont il est finaliste.
    </p>
  </ion-card-content>
  <ion-row>
    <ion-col>
      <ion-button fill="clear" size="small" color="duckcoin">
        J'aime <ion-icon name="heart" slot="start"></ion-icon>
      </ion-button>
    </ion-col>
    <ion-col text-right>
      <ion-button fill="clear" size="small" color="duckcoin">
        Partager
        <ion-icon name='share-alt' slot="start"></ion-icon>
      </ion-button>
    </ion-col>
  </ion-row>
```

Card

Moussa SISSOKO

Moussa Sissoko, né le 16 août 1989 au Blanc-Mesnil, en Seine-Saint-Denis, est un footballeur international français qui évolue au poste de milieu de terrain à Tottenham Hotspur. Avec la sélection, il participe à la Coupe du monde 2014 et à l'Euro 2016 dont il est finaliste.

J'AIME PARTAGER

Documentation : https://ionicframework.com/docs/components/#cards

Composant FAB

OCTOBRE 2020

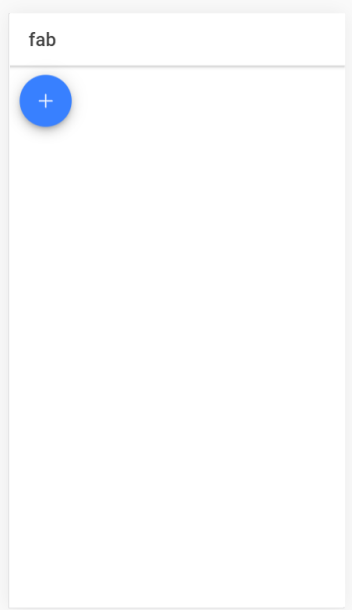
Ch4. Composant Fab (Floating Action Buttons)

Ce composant est issu du Material Design, un ensemble de règles de design proposés par Google et présent notamment dès la version 5.0 du système d'exploitation Android. Il s'agit d'un élément en forme de cercle qui permet, au clic, d'afficher d'autres éléments actionnables eux aussi par clic.

```
<ion-content>
  <ion-fab vertical="bottom" horizontal="end">
    <ion-fab-button>
      <ion-icon name="add"></ion-icon>
    </ion-fab-button>
  </ion-fab>
</ion-content>
```

On peut choisir de placer notre fab à différents endroit sur l'écran. Affichons-le par exemple en haut à gauche

```
<ion-content>
  <ion-fab vertical="top" horizontal="start" slot="fixed">
    <ion-fab-button>
      <ion-icon name="add"></ion-icon>
    </ion-fab-button>
  </ion-fab>
</ion-content>
```



N'hésitez pas au besoin à adapter votre style scss comme ci :

```
ion-fab[vertical="top"] {
    top:60px;
}
```

Documentation : https://ionicframework.com/docs/api/components/fab/FabButton/

Système de routage d'Angular

OCTOBRE 2020

La version 4 de Ionic a vu l'arrivée d'un tout nouveau système de routage inspirée d'Angular. Désormais chaque url est associée à une page de l'application, tout comme tout url dans un navigateur est associé à une page web.

Ce modèle est beaucoup plus intuitif que le précédent : en effet c'est le principe de fonctionnement d'un navigateur web classique. Et vous vous en souvenez surement, mais une application hybride est un simple navigateur en mode plein écran.

On va donc pouvoir avec le routage Ionic :

- Saisir une URL dans la barre d'adresse et être redirigé vers la page correspondante.
- Cliquer sur les liens d'une page et naviguer vers une nouvelle page.
- Cliquer sur les boutons Précédent / Suivant de notre application et en conservant l'historique des pages consultées.

C'est ainsi que fonctionne le routeur Angular. Il peut interpréter une URL comme étant une instruction permettant d'accéder à une ressource de l'application. Il peut transmettre des paramètres facultatifs au composant de la vue prise correspondante à l'url, ce qui l'aide à choisir le contenu spécifique à présenter.

Lorsque vous lancez une application Ionic, c'est le fichier src/index.html qui est appelé en premier.

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8" />
  <title>TrombiSchool</title>

  <base href="/" />

  <meta name="color-scheme" content="light dark" />
  <meta name="viewport" content="viewport-fit=cover, width=device-width, initial-scale=1.0
  <meta name="format-detection" content="telephone=no" />
  <meta name="msapplication-tap-highlight" content="no" />

  <link rel="icon" type="image/png" href="assets/icon/favicon.png" />

  <!-- add to homescreen for ios -->
  <meta name="apple-mobile-web-app-capable" content="yes" />
  <meta name="apple-mobile-web-app-status-bar-style" content="black" />
</head>

<body>
  <app-root></app-root>
</body>

</html>
```

Ce point d'entrée de l'application permet de préciser l'URL de base à utiliser pour recomposer toutes les URL relatives contenues dans l'application.

```
<base href="/" />
```

On déclare ensuite le composant **AppComponent** dans le corps de la page html :

```
<body>
  <app-root></app-root>
</body>
```

Ce composant est lui-même définit dans le fichier **src/app/app.component.ts**.

```
@Component({
  selector: 'app-root',
  templateUrl: 'app.component.html',
  styleUrls: ['app.component.scss']
})
export class AppComponent {
  constructor(
    private platform: Platform,
    private splashScreen: SplashScreen,
    private statusBar: StatusBar
  ) {
    this.initializeApp();
  }

  initializeApp() {
    this.platform.ready().then(() => {
      this.statusBar.styleDefault();
      this.splashScreen.hide();
    });
  }
}
```

C'est ce composant qui permet d'appeler le module de routage, comme on peut le voir dans le fichier **src/app/app.component.html** :

```
<ion-app>
  <ion-router-outlet></ion-router-outlet>
</ion-app>
```

Module qui est défini dans le fichier **src/app/app-routing.module.ts**. On y retrouve toutes les routes de notre application :

```
import { NgModule } from '@angular/core';
import { PreloadAllModules, RouterModule, Routes } from '@angular/router';

const routes: Routes = [
  {
    path: '',
    loadChildren: () => import('./tabs/tabs.module').then(m => m.TabsPageModule)
  },
  {
    path: 'profile',
    loadChildren: () => import('./profile/profile.module').then(m => m.ProfilePageModule)
  },
  {
    path: 'etudiants',
    loadChildren: () => import('./etudiants/etudiants.module').then(m => m.EtudiantsPageModule)
  },
  {
    path: 'etudiants/:id',
    loadChildren: () => import('./etudiant/etudiant.module').then(m => m.EtudiantPageModule)
  },
  {
    path: 'ui',
    loadChildren: () => import('./ui/ui.module').then( m => m.UiPageModule)
  }
];
@NgModule({
  imports: [
    RouterModule.forRoot(routes, { preloadingStrategy: PreloadAllModules })
  ],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

- La première entrée permet de rediriger toutes les urls "vides" (path:' '), mal formées ou celle par défaut, vers la page d'accueil
- La deuxième entrée permet d'afficher la page profil à partir de l'url /**profile**.

Dans la définition du routage, nous avons une clé **loadChildren** dans laquelle nous fournissons un chemin vers le module de notre page.
Ce fichier de module contient des informations et des importations pour la page. C'est en quelque sorte le moteur de la page, à l'image d'un moteur de voiture.

Grâce au tag **<ion-router-outlet main></ion-router-outlet>,** le routeur Angular remplacera toutes les entrées définies dans la table de routage par les informations du module associé (HomePageModule, ProfilePageModule,...).

Pour plus d'informations sur le routage Angular, n'hésitez pas à consulter la documentation officielle :
https://angular.io/guide/router

Editons à présent l'onglet étudiant, de manière à être redirigé vers la page liste étudiants au clic sur le bouton de recherche :

```
<ion-content>
<h2 class="ion-text-center ion-text-uppercase">Rechercher un étudiant</h2>
<ion-button color="primary" routerLink="/etudiants" routerDirection="root">Rechercher</ion-button>
</ion-content>
```

En cliquant sur le bouton « Rechercher", on est tout de suite redirigé vers la page, presque sans transition. C'est la directive **routerDirection="root"** qui permet cette animation de type "changement de page principale" entre les pages. On aurait pu également utiliser les transition **forward** et **backward** suivantes :

- routerDirection="forward" : fondu en entrée
- routerDirection="back" : fondu en sortie

Pour le moment la page d'accueil reste quand même la page par défaut.

Mais nous allons pouvoir faire en sorte que la page par défaut soit désormais la page de profil. Pour cela, modifions notre table de routage comme ceci :
src/app/app-routing.module.ts

```
const routes: Routes = [
  // La page par défaut devient la page de profil
  {
    path: '',
    redirectTo: 'profile',
    pathMatch: 'full'
  },
```

Protection de pages avec les Guards angular

Il est souvent nécessaire de protéger certains contenus et ne les afficher que selon certaines conditions : utilisateur connecté, abonnement premium, ... Dans ces cas là, Angular propose une solution très pratique appelée **Guards**.

Ceux-ci vont permettre de contrôler l'accès à une "route" particulière ou encore s'assurer que le passage d'une route à une autre se déroule selon les standards : alerte en cas de sortie d'un formulaire non enregistré ou de perte de connection au moment de la validation d'une action,...

Pour mettre en place ce mécanisme de protection, nous allons saisir la commande suivante :

```
$ ionic g guard guards/auth
```

Puis modifions le fichier qui a été généré pour qu'il ressemble à ceci:

src/app/guards/auth.guard.ts

```
import { Injectable } from '@angular/core';
import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, UrlTree } from '@angular/router';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate {
  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean> | Promise<boolean> | boolean  {
      let userAuthenticated = false; // Pour le moment nous allons garder cette valeur à false

      if (userAuthenticated) {
        return true;
      } else {
        return false;
      }
  }

}
```

Notre «**gardien**» n'implémente qu'une seule méthode : **canActivate**. _Celle-ci permet, selon un critère donné, d'afficher ou non une route. Ici notre condition est simplement basée sur la valeur de la variable userAuthenticated_, que l'on pourra plus tard récupérer depuis la session ou en base de données.

Pour le moment, ce gardien n'est pas vraiment utile. Pour qu'il le soit, il va nous falloir l'appeler depuis la table de routage :

src/app/app-routing.module.ts.

```
import { NgModule } from '@angular/core';
import { PreloadAllModules, RouterModule, Routes } from '@angular/router';

// On appelle notre gardien ici
import { AuthGuard } from './guards/auth.guard';

const routes: Routes = [
  // La page par défaut devient la page de profil
  {
    path: '',
    redirectTo: 'profile',
    pathMatch: 'full'
  },
  // Pas d'accès la page d'accueil si non connecté
  {
    path: 'tabs',
    loadChildren: () => import('./tabs/tabs.module').then(m => m.TabsPageModule),
    canActivate: [AuthGuard]
  },
```

Dans cette nouvelle configuration, nous interdisons tout accès à la page d'accueil si l'on est pas connecté (userAuthenticated à false).

Si vous tenter d'afficher la page d'acceuil, rien ne se passe. En modifiant la valeur de la variable **userAuthenticated** en la passant à **true**, on peut de nouveau d'afficher la page d'accueil.

Pour le moment, ce gardien n'est pas vraiment utile. Pour qu'il le soit, il va nous falloir l'appeler depuis la table de routage :

src/app/app-routing.module.ts.

```
import { NgModule } from '@angular/core';
import { PreloadAllModules, RouterModule, Routes } from '@angular/router';

// On appelle notre gardien ici
import { AuthGuard } from './guards/auth.guard';

const routes: Routes = [
  // La page par défaut devient la page de profil
  {
    path: '',
    redirectTo: 'profile',
    pathMatch: 'full'
  },
  // Pas d'accès la page d'accueil si non connecté
  {
    path: 'tabs',
    loadChildren: () => import('./tabs/tabs.module').then(m => m.TabsPageModule),
    canActivate: [AuthGuard]
  },
```

Dans cette nouvelle configuration, nous interdisons tout accès à la page d'accueil si l'on est pas connecté (userAuthenticated à false).

Si vous tenter d'afficher la page d'acceuil, rien ne se passe. En modifiant la valeur de la variable **userAuthenticated** en la passant à **true**, on peut de nouveau d'afficher la page d'accueil.

Notre gardien fais plutôt bien son travail. Mais le fait qu'il ne se passe rien en affichant la page d'accueil peut être source d'incompréhension pour un utilisateur qui pourrait penser à un bug de l'application. De plus, si l'on décide d'afficher la page d'accueil en y accédant via l'url http://localhost:8100/home, on a droit cette fois à une belle page blanche.

Modifions donc un peu notre guard, pour rediriger l'utilisateur vers la page d'accueil si l'on est déjà connecté, et vers la page de profif dans le cas contraire.

```
export class AuthGuard implements CanActivate {
  // Dans le constructeur on déclare notre variable de routage
  constructor(private router: Router) {
  }
  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean> | Promise<boolean> | boolean {
    let userAuthenticated = false; // Pour le moment nous allons garder cette valeur à false

    if (userAuthenticated) {
      return true;
    } else {
      this.router.navigate(['/profile']);
      //return false;
    }
  }

}
```

Voilà, on a désormais les bonnes redirections en fonction du statut de connexion. Beaucoup mieux.

Attention cependant à ne pas considérer les gardiens comme des barrières de protection infaillibles, surtout si vous affichez des données ultra sensibles dans votre application. Privilégiez plutôt une authentification côté serveur forte, puis ajoutez les gardiens en complément de celle-ci.