

Institut Supérieur des Etudes Technologiques de Mahdia

FRAMEWORK COTÉ CLIENT

Mme KHAYATI Alya

Ingénieur en génie logiciel

Alya.khayati@hotmail.com



OBJECTIFS

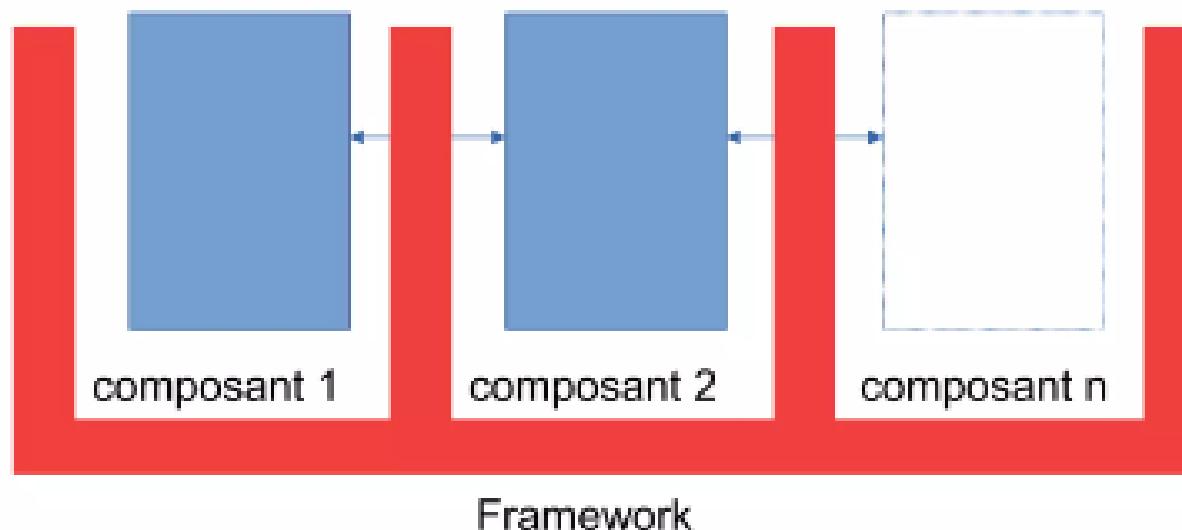
- Comprendre les fondamentaux et les concepts avancés d'Angular.
- créer une application Angular
- Maîtriser le routage, les services, et les modules en Angular.
- Comprendre comment travailler avec des observables et des événements.

PRÉ-REQUIS

- Connaissance de base en programmation.
- de bonnes connaissances en HTML, en CSS et en JavaScript.

NOTION DE FRAMEWORK

- Un Framework désigne un ensemble cohérent de composants logiciels structurels.
- Un framework sert à créer les **fondations** ainsi que **les grandes lignes** de tout ou d'une partie d'un logiciel(architecture)





QU'EST-CE QU'ANGULAR ?

Angular est un **framework de développement** pour créer des **applications web dynamiques**. Il a été développé par **Google** et il est écrit en **TypeScript**.

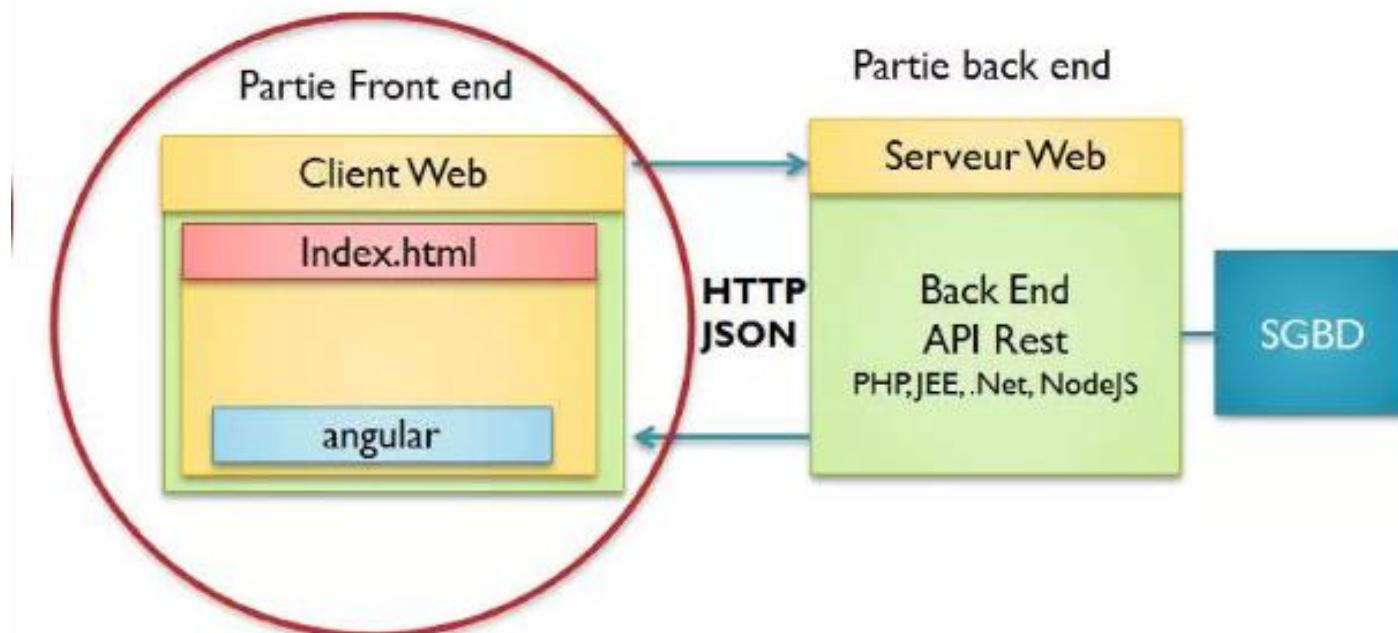
Version Actuelle	Angular 16
Date Lancement	Mai 2023
NodeJs Compatible	^16.14.0 ^18.10.0
TypeScript	>=4.9.3 <5.1.0



Angular

Angular est un Framework front end de google qui aide à créer des applications à page unique (SPA) interactives et dynamiques avec ses fonctionnalités convaincantes, notamment la création de modèles, la liaison bidirectionnelle, la modularisation, la gestion de l'API RESTful, l'injection de dépendances et la gestion AJAX.

- Il permet notamment de créer ce qu'on appelle des **Single Page Applications** (ou **SPA**) : des applications entières qui tournent dans une seule page HTML .
- La seule page contient différents composants web.



POURQUOI ANGULAR ?

Angular offre plusieurs avantages qui en font un choix populaire parmi les développeurs.

Angular est un framework **complet** – on peut créer des applications web complètes sans avoir besoin de librairies tierces supplémentaires. C'est notamment ce qui différencie un *framework* d'une *library*.

Il permet une **modularité**, ce qui signifie que les différentes parties de votre application peuvent être **réutilisées** et **testées séparément**. De plus, sa **communauté active** et les **régulières mises à jour** contribuent à le garder pertinent et à jour.

LES TROIS PILIERS D'ANGULAR

- **Composants:** Ce sont les éléments de base d'une application Angular. Ils contiennent le code **HTML**, **CSS**, et **TypeScript** pour rendre les pages.
- **Modules:** Angular utilise un système modulaire pour organiser votre code. Un module peut encapsuler des **composants**, **directives** et **services** qui sont liés fonctionnellement.
- **Services et Injection de Dépendance:** Les services sont des objets qui sont injectés dans des composants et autres services. Ils permettent de partager des **fonctions et des données** à travers l'application.

QUAND UTILISER ANGULAR ?

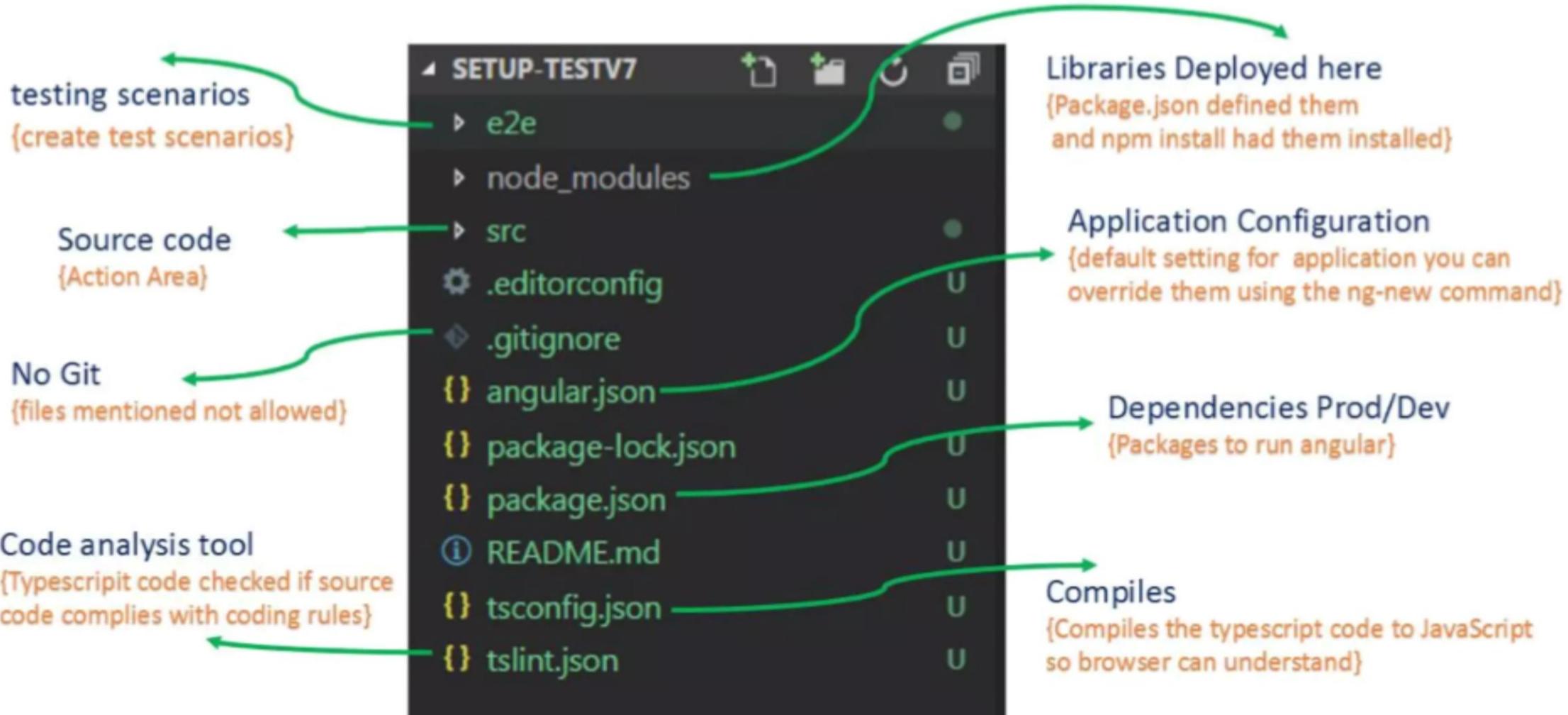
Angular est particulièrement utile lorsque vous développez des **applications web complexes** avec des **fonctionnalités riches**. Il est souvent utilisé pour créer des **applications d'entreprise**, des **dashboards**, et même des **applications mobiles** grâce à des solutions comme **Ionic**.

Angular utilise TypeScript, qui fournit un excellent support pour la vérification de type et d'autres outils externes.

Angular est pris en charge par Google, ce qui signifie qu'il est soutenu par une organisation fiable. Ils travaillent avec une documentation détaillée et une grande communauté, ce qui en fait un cadre d'apprentissage fiable.

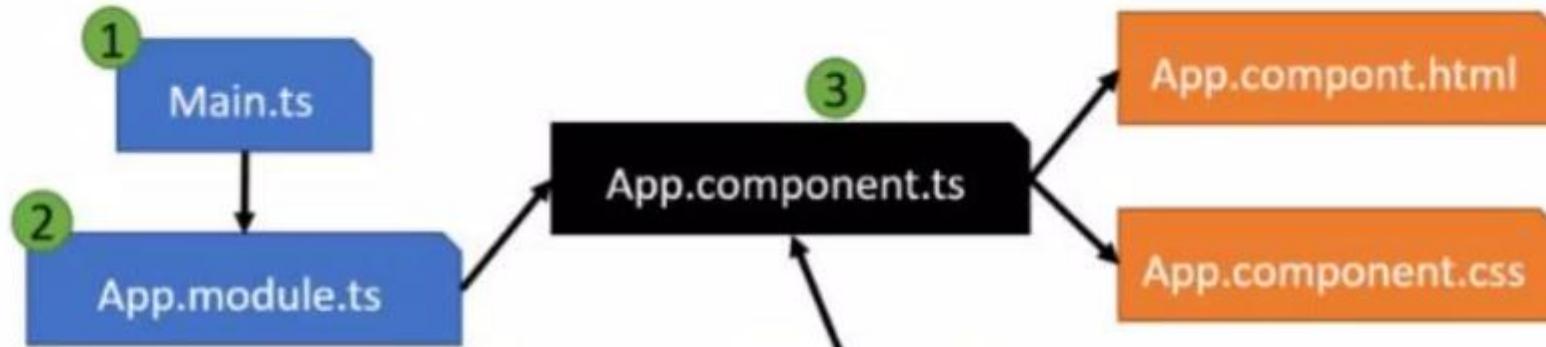
Angular-language-service permet la saisie semi-automatique à l'intérieur des fichiers de modèle HTML externes des composants, vous permettant d'accélérer votre développement.

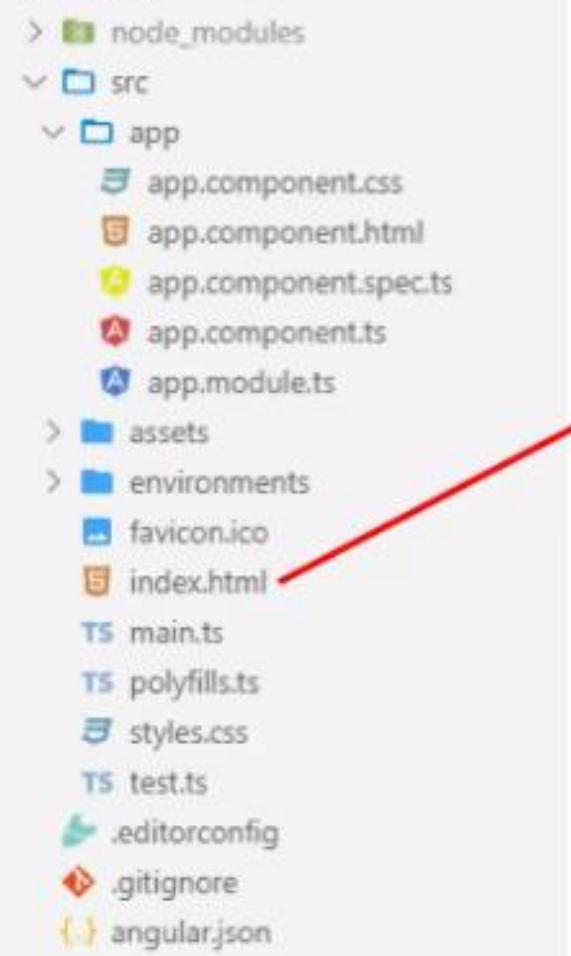
STRUCTURE PROJET ANGULAR



STRUCTURE PROJET ANGULAR DETAILLÉE

```
src
  app
    # app.component.css
    < app.component.html
    TS app.component.spec.ts
    TS app.component.ts
    TS app.module.ts
    assets
    environments
    ★ favicon.ico
    < index.html
    TS main.ts
    TS polyfills.ts
    # styles.css
    TS test.ts
    {} tsconfig.app.json
    {} tsconfig.spec.json
    TS typings.d.ts
```



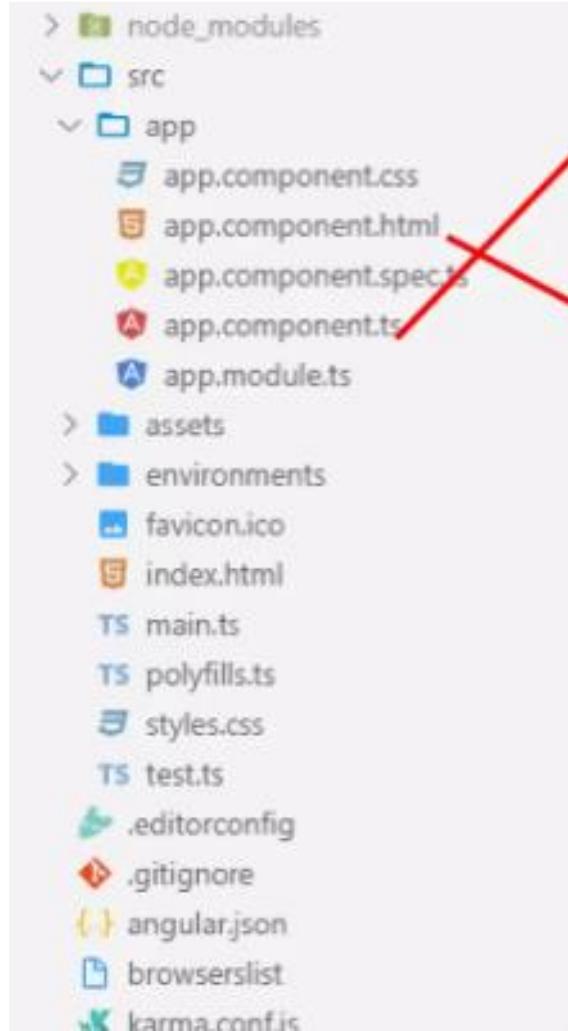


index.html

src > index.html > ...

```
1  <!doctype html>
2  <html lang="en">
3  <head>
4  <meta charset="utf-8">
5  <title>FirstApp</title>
6  <base href="/">
7  <meta name="viewport" content="width=device-width, initial-scale=1">
8  <link rel="icon" type="image/x-icon" href="favicon.ico">
9  </head>
10 <body>
11  <app-root></app-root>
12  </body>
13  </html>
```

Sélecteur du root component

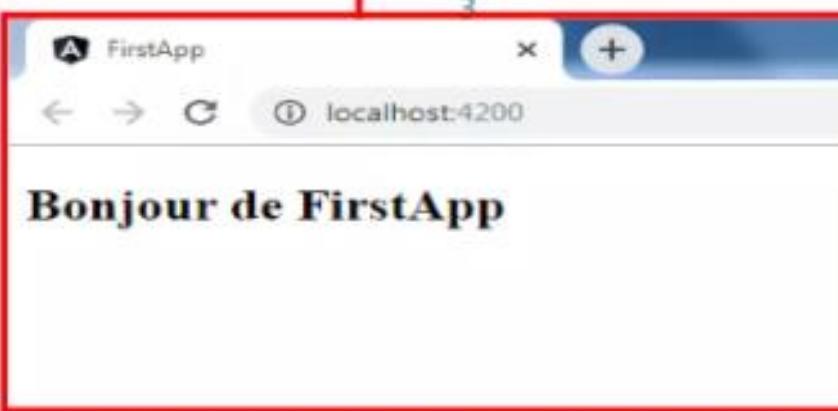


app.component.ts

```
src > app > app.component.ts > ...
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   title = 'FirstApp';
10 }
```

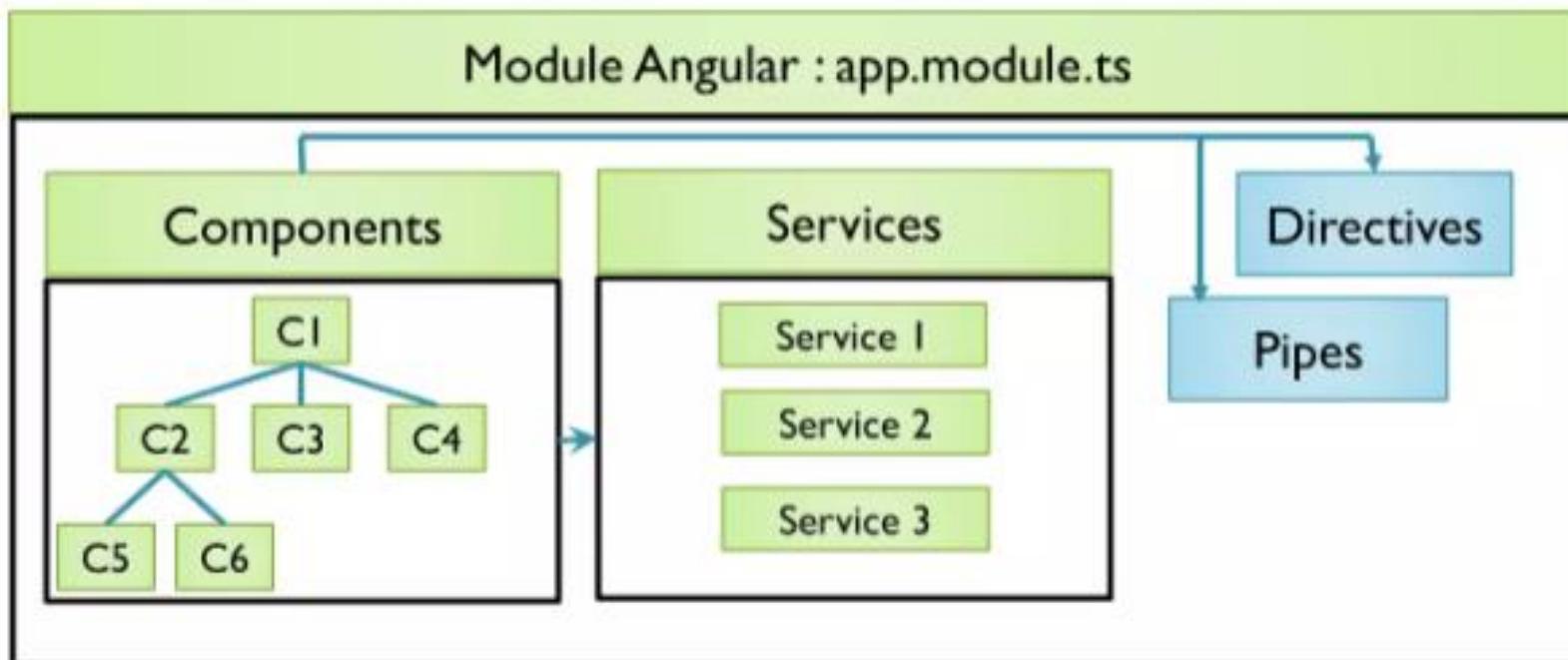
app.component.html

```
src > app > app.component.html > ...
1
2 <h2>Bonjour de {{title}}</h2>
3
```

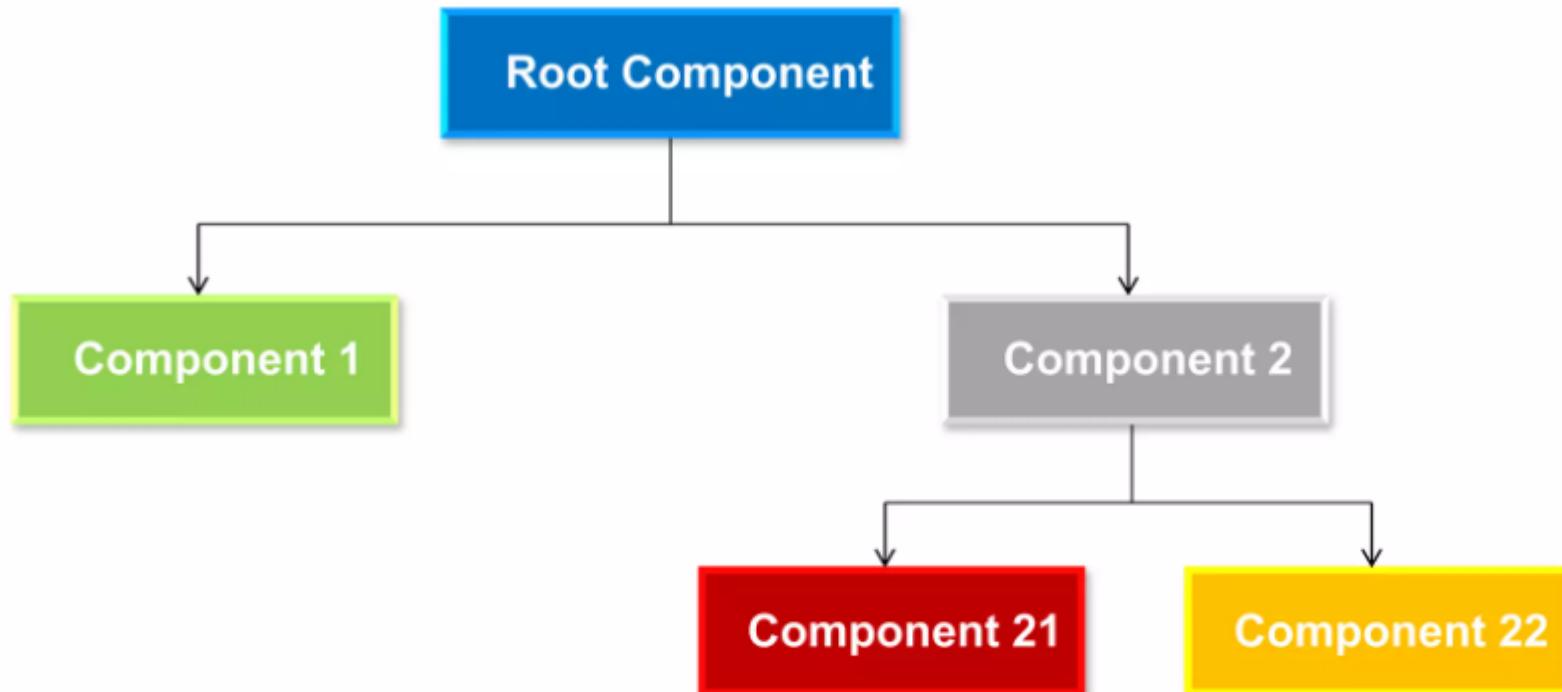


ARCHITECTURE DU FRAMEWORK ANGULAR

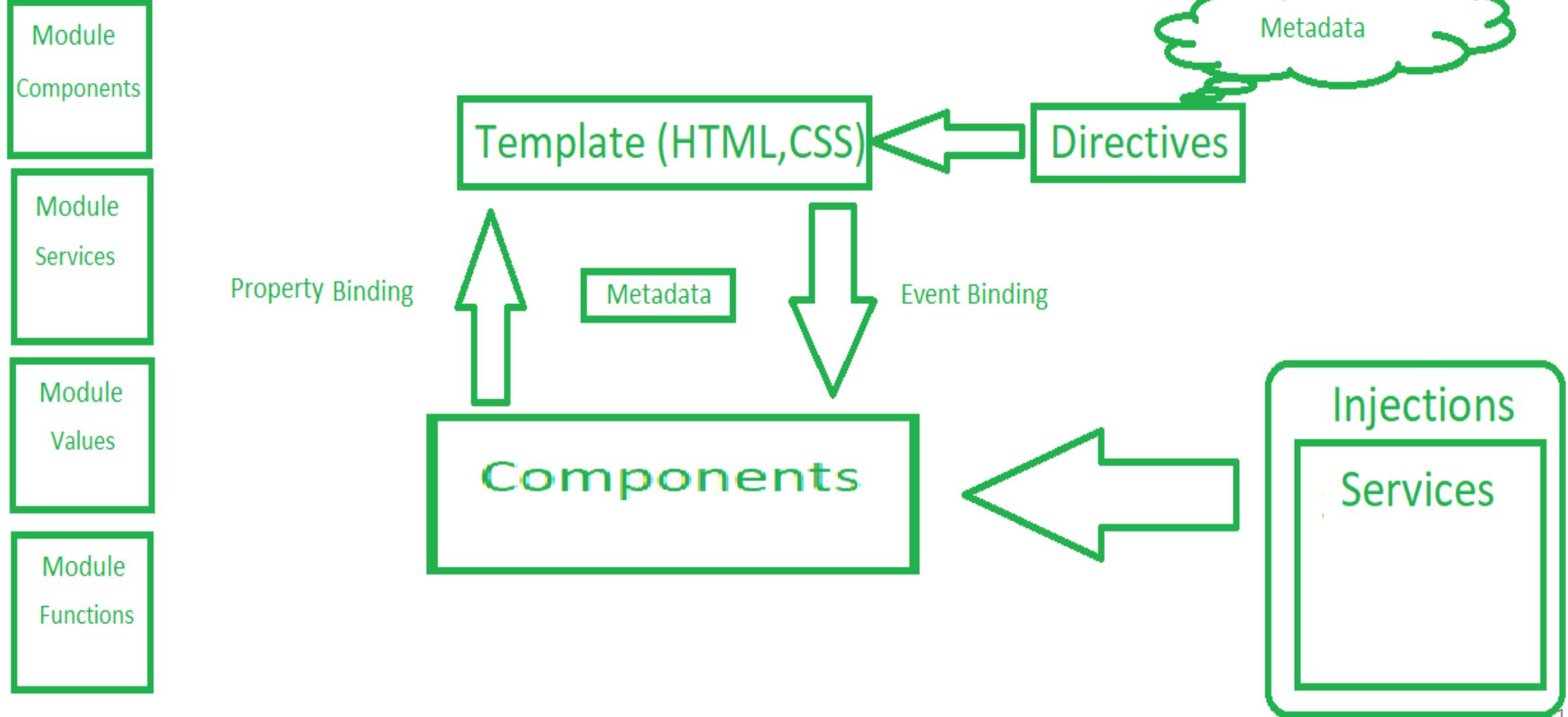
- Une application Angular se compose de :
 - Un à plusieurs modules dont un est principal.
 - Chaque module peut inclure :
 - Des composant web : La partie visible de l'application Web (IHM)
 - Des services pour la logique applicative. Les composants peuvent utiliser les services via le principe de l'injection des dépendances.
 - Les directives : un composant peut utiliser des directives
 - Les pipes : utilisés pour formater l'affichage des données dans les composants.



ARCHITECTURE DU FRAMEWORK ANGULAR



ARCHITECTURE DE BASE DE ANGULAR



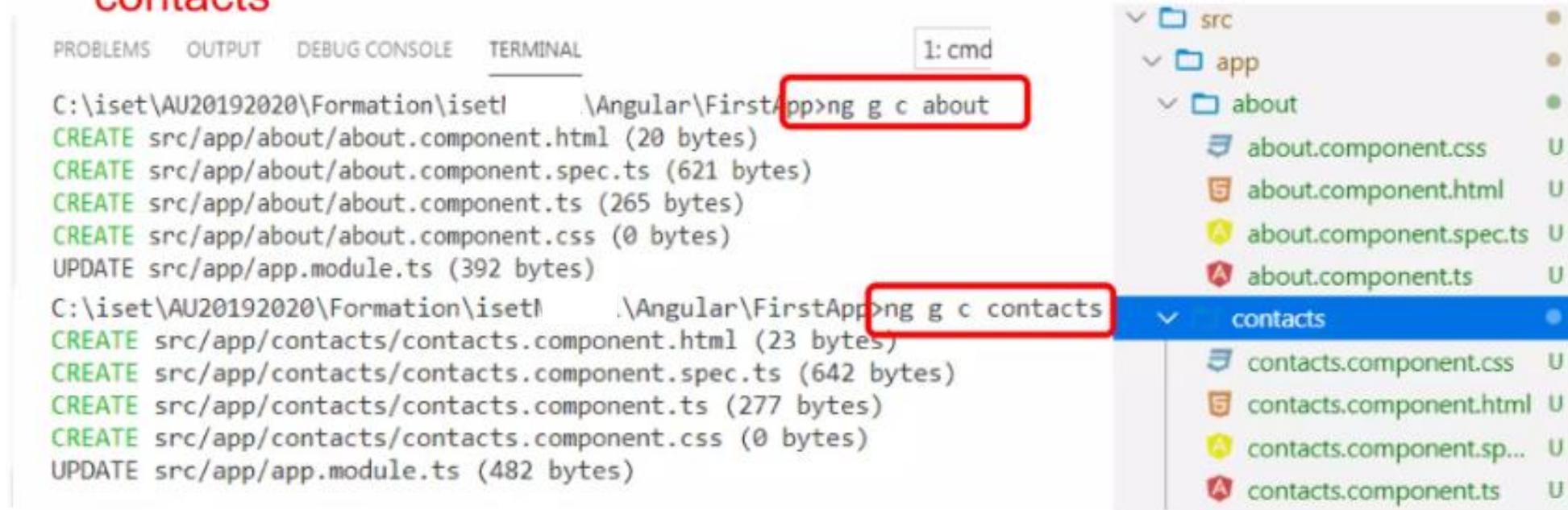
LES COMPOSANTS

Chaque composant se compose de:

- Un classe représentant sa logique métier
- Une template HTML: représentant sa vue
- Une feuille de style css
- Un fichier de test unitaire

CRÉATION DE NOUVEAUX COMPOSANTS

- Pour créer facilement des composants Angular, on peut utiliser la commande ng comme suit :
- ng generate component NomComposant**
- Dans notre exemple, nous allons créer deux composants : **about** et **contacts**



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: cmd  
C:\iset\AU20192020\Formation\isetl\Angular\FirstApp>ng g c about  
CREATE src/app/about/about.component.html (20 bytes)  
CREATE src/app/about/about.component.spec.ts (621 bytes)  
CREATE src/app/about/about.component.ts (265 bytes)  
CREATE src/app/about/about.component.css (0 bytes)  
UPDATE src/app/app.module.ts (392 bytes)  
  
C:\iset\AU20192020\Formation\isetl\Angular\FirstApp>ng g c contacts  
CREATE src/app/contacts/contacts.component.html (23 bytes)  
CREATE src/app/contacts/contacts.component.spec.ts (642 bytes)  
CREATE src/app/contacts/contacts.component.ts (277 bytes)  
CREATE src/app/contacts/contacts.component.css (0 bytes)  
UPDATE src/app/app.module.ts (482 bytes)
```

The file explorer on the right shows the following structure:

- src
 - app
 - about
 - about.component.css
 - about.component.html
 - about.component.spec.ts
 - about.component.ts
 - contacts
 - contacts.component.css
 - contacts.component.html
 - contacts.component.spec.ts
 - contacts.component.ts

A about.component.ts ×

src > app > about > A about.component.ts > ...

```
1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-about',
5   templateUrl: './about.component.html',
6   styleUrls: ['./about.component.css']
7 })
8 export class AboutComponent implements OnInit {
9
10   constructor() { }
11
12   ngOnInit() {
13   }
14
15 }
16
```

Décorateur: indique à Angular que cette classe joue le rôle d'un composant avec:

- Un sélecteur '**app-about**'
- Un template HTML '**./about.component.html**'
- Un seul fichier de style css '**./about.component.css**'

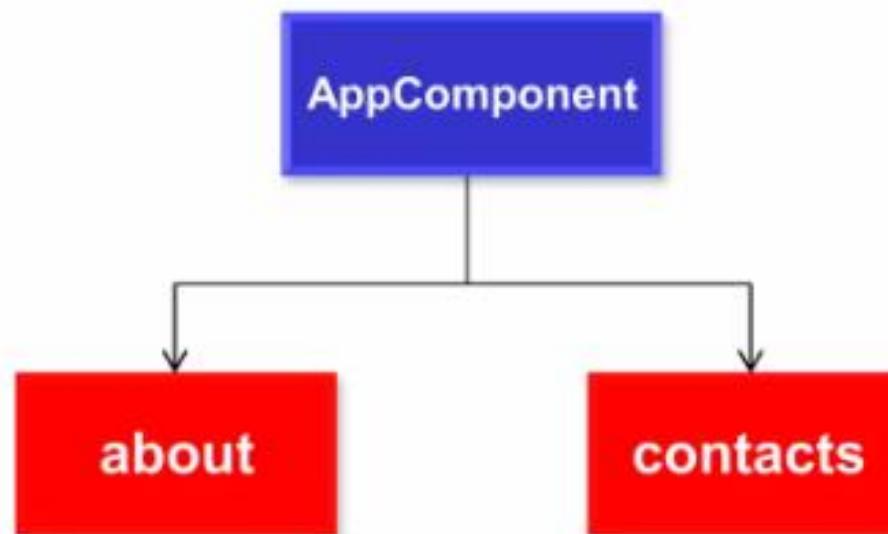
Un composant doit être déclaré dans un module

```
app.module.ts ×  
src > app > app.module.ts > ...  
1 import { BrowserModule } from '@angular/platform-browser';  
2 import { NgModule } from '@angular/core';  
3  
4 import { AppComponent } from './app.component';  
5 import { AboutComponent } from './about/about.component';  
6 import { ContactsComponent } from './contacts/contacts.component';  
7  
8 @NgModule({  
9   declarations: [  
10     AppComponent,  
11     AboutComponent,  
12     ContactsComponent  
13   ],  
14   imports: [  
15     BrowserModule  
16   ],  
17   providers: [],  
18   bootstrap: [AppComponent]  
19 })  
20 export class AppModule { }  
21
```

- Un composant peut être inséré dans n'importe quel partie HTML de l'application en utilisant son **sélecteur**
- Dans cet exemple, les deux composants **about** et **contacts** sont insérés à l'intérieur du composant racine **AppComponent**

app.component.html X

```
src > app > app.component.html > ...
1
2  <h2>Bonjour de {{title}}</h2>
3  <div>
4    <app-about></app-about>
5  </div>
6  <div>
7    <app-contacts></app-contacts>
8  </div>
9 ,
```



LIAISON DE DONNÉES

DOM

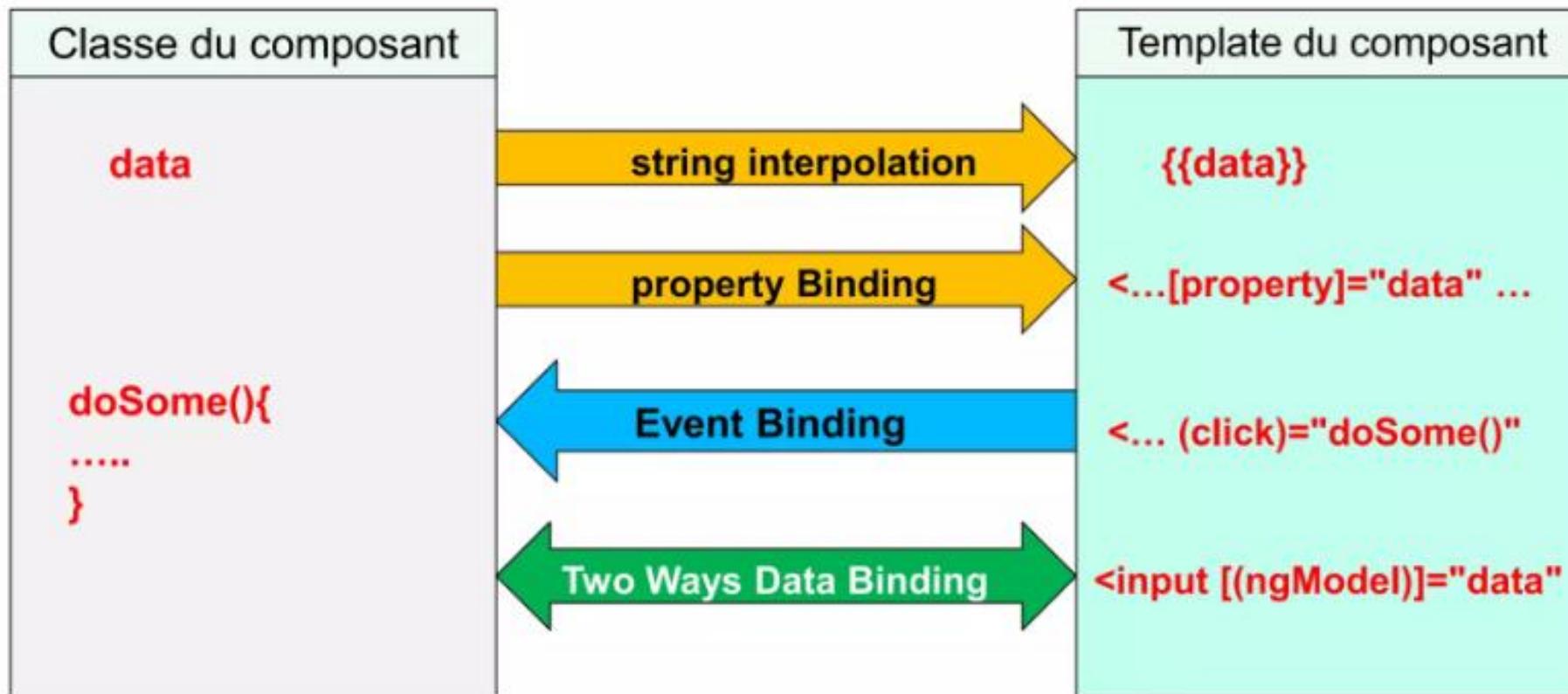
DOM

La structure logique des documents et des documents accessibles et manipulés est définie à l'aide d'éléments DOM. Il définit les événements, les méthodes et les propriétés de tous les éléments HTML en tant qu'objets. DOM dans Angular agit comme une API (interface de programmation) pour javascript.

Chaque fois qu'une page Web est chargée, le navigateur crée un objet de modèle de document (DOM) de cette page.

DATA BINDING

- Permet de faire un lien entre les données de la classe du composant et son Template associé



DATA BINDING : EXEMPLE

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-firsdtcomp',
5   templateUrl: './firsdtcomp.component.html',
6   styleUrls: ['./firsdtcomp.component.css'],
7 })
8 export class FirsdtcompComponent {
9   info = { nom: 'Alya',
10         | email: 'alya.khayati@hotmail.com'
11         | age: 30 };
12   bgColor = 'red';
13
14   Comments: any[] = [];
15
16   comment = { id: 0, message: '' };
17
18   newComment = false;
19   addComment() {
20     if (this.comment.message != '') {
21       this.comment.id = this.Comments.length + 1;
22       this.Comments.push({
23         id: this.comment.id,
24         message: this.comment.message,
25       });
26       this.comment.message = '';
27     }
28   }
29 }
30
```

Bonjour

1. Alya
2. alya.khayati@hotmail.com
3. 30

AjouterCommentaire

Liste des messages:

- 1 : Bonjour



File Edit Selection View Go Run ...

← →

Search tp1



firsdtcomp.component.html X

TS firsdtcomp.component.ts



src > app > firsdtcomp > firsdtcomp.component.html > div > ul > li



```
5      <ol><li>{{c.message}}</li>
6    </ol>
7    <div>
8      <input type="text" [(ngModel)]="comment.message">
9      <button (click)="addComment()" [disabled]="newComment">Ajouter</button>
10     </div>
11   <div *ngIf="Comments.length>0; else noComments">
12     <h3>Liste des messages:</h3>
13     <ul>
14       <li *ngFor="let c of Comments">
15         {{c.id}} : {{c.message}}
16       </li>
17     </ul>
18   </div>
19   <ng-template #noComments>
20     <p [style.backgroundColor]="#0000ff">Liste est vide</p>
21   </ng-template>
```



The screenshot shows the Visual Studio Code interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** tp1
- Explorer Bar:** Shows the project structure under TP1:
 - node_modules
 - src
 - app
 - div-comp
 - firsdtcomp
 - # firsdtcomp.component.css
 - firsdtcomp.component.html
 - firsdtcomp.component.spec.ts
 - firsdtcomp.component.ts
 - app-routing.module.ts
 - # app.component.css
 - app.component.html
 - app.component.spec.ts
 - app.component.ts
 - assets
 - Editor:** The file `app.module.ts` is open, showing the following code:

```
src > app > TS app.module.ts > AppModule
1 import { NgModule } from '@angular/core';
2 import { FormsModule } from '@angular/forms';
3 import { BrowserModule } from '@angular/platform-browser';
4
5 import { AppRoutingModule } from './app-routing.module';
6 import { AppComponent } from './app.component';
7 import { FirsdtcompComponent } from './firsdtcomp/firsdtcomp.component';
8 import { DivCompComponent } from './div-comp/div-comp.component';
9
10 @NgModule({
11   declarations: [AppComponent, FirsdtcompComponent, DivCompComponent],
12   imports: [BrowserModule,
13             AppRoutingModule,
14             FormsModule],
15   providers: [],
16   bootstrap: [AppComponent],
17 })
18 export class AppModule {}
```
 - Status Bar:** Shows the status bar with icons for file, edit, selection, view, go, run, and other settings.

LES DIRECTIVES

LES DIRECTIVES

Les directives sont des classes permettant d'enrichir et modifier la vue par simple ajout d'attributs Html sur le template

Il existe deux types de directives:

- **Les directives structurelles:** Elles ont pour but de modifier le DOM en ajoutant, enlevant ou replaçant un élément du DOM.
- **Les attribute directives:** Elles ont pour but de modifier l'apparence ou le comportement d'un élément.

LES DIRECTIVES STRUCTURELLES

ngIf

ngFor

NgSwitch

- Permet de supprimer ou de recréer l'élément courant suivant l'expression passée en paramètre
- Exemple:

```
<div *ngIf="1 > 0"> Afficher la div</div>
```

```
<div *ngIf="1 < 0"> N'affiche pas la div</div>
```

```
<div *ngIf="afficherNom; else elseBlock">  
    Alya Khayati  
</div>  
  
<ng-template #elseBlock>  
    *****  
</ng-template>
```

```
<div *ngIf="afficherNom; then thenBlock; else elseBlock"></div>

<ng-template #thenBlock>
    Alya Khayati
</ng-template>

<ng-template #elseBlock>
    *****
</ng-template>
```

NGFOR

```
export class TestComponent implements OnInit {  
  public couleurs = ['rouge', 'vert', 'bleu'];  
  
  constructor() { }  
  
  ngOnInit() {  
  }  
}
```

Liste des couleurs:
<div *ngFor="let c of couleurs">
 {{c}}
</div>

Liste des couleurs:
rouge
vert
bleu

NGSWITCH

```
<div [ngSwitch]="couleur">  
  <div *ngSwitchCase="'rouge'">couleur rouge</div>  
  <div *ngSwitchCase="'bleu'">couleur bleu</div>  
  <div *ngSwitchCase="'vert'">couleur vert</div>  
  <div *ngSwitchDefault>choisir une couleur</div>  
</div>
```

LES ATTRIBUT DIRECTIVES

ngStyle

ngClass

NGSTYLE

- permet de modifier le style d'un élément HTML
- s'utilise conjointement avec le **property binding** pour récupérer des valeurs définies dans la classe

```
public styleCorrect={ 'background-color':'green' };
```

```
<div [ngStyle]="styleCorrect">Réponse</div>
```

contacts works!

Réponse

NGCLASS

- permet d'attribuer de nouvelles classes d'un élément HTML
- s'utilise conjointement avec le **property binding** pour récupérer des valeurs définies dans la classe ou dans la feuille de style

```
<div [ngClass]="{'text-success':question.isCorrect}">Réponse</div>
```

QUIZZ

1) Dans Angular CLI, que signifie CLI ?

- a) Control Link Interface
- b) Control Layer Interface
- c) Command Layer Interface
- d) Command Line Interface

2) Dans le fichier index.html, à quoi correspond la balise <app-root> ?

- a) Au component AppComponent
- b) Au module AppModule
- c) Au contenu de main.ts
- d) Au component de test TestComponent

3)Qu'est-ce que la string interpolation ?

- a) La string interpolation permet de concaténer plusieurs chaînes de caractères ensemble.
- b) La string interpolation remplace un chiffre par une chaîne de caractères.
- c) La string interpolation permet d'afficher la valeur d'une variable dans le DOM.
- d) La string interpolation permet de récupérer une chaîne de caractères du DOM pour la traiter dans le fichier TypeScript.

4)Quels sont les fichiers principaux d'un component Angular ?

- a) Un fichier HTML, un fichier SCSS et un fichier TS.
- b) Un fichier HTML, un fichier SCSS et un fichier JS.
- c) Un fichier XML, un fichier CSS et un fichier TS.
- d) Un fichier HTML, un fichier TS et un fichier JS.

5)Qu'est-ce que la liaison par événement, ou event binding ?

- a) Elle permet de lier la valeur d'une variable à un attribut d'un élément du DOM.
- b) Elle permet de lier une méthode TypeScript à un événement du DOM.
- c) Elle permet de remplacer une valeur par une autre dans le DOM.
- d) Elle permet de réagir à des changements de valeur d'un élément dans le DOM.

6)Mon component CoffeeCupComponent a une propriété injectable size . Quelle syntaxe est correcte pour lui passer une variable appelée "largeSize" ?

- a) <app-coffee-cup size="[largeSize]"></app-coffee-cup>
- b) <app-coffee-cup (size)="largeSize"></app-coffee-cup>
- c) <app-coffee-cup [size]="largeSize"></app-coffee-cup>
- d) <app-coffee-cup [largeSize]="size"></app-coffee-cup>

ROUTAGE ET NAVIGATION

ROUTAGE ET NAVIGATION

- Une application Angular contient plusieurs composants,
- Chaque composant possède une vue (template)
- Il faut pouvoir naviguer entre les vues quand l'utilisateur réalise une action.

ROUTAGE ET NAVIGATION

localhost:4200

Routing navigation

Départements

Employés

localhost:4200/departements

localhost:4200/employes

ROUTAGE ET NAVIGATION

- Le routeur angulaire permet la navigation d'une vue à l'autre lorsque les utilisateurs exécutent des tâches d'application.
- Le routeur angulaire est un service facultatif qui présente une vue de composant particulière pour une URL donnée.
- Il ne fait pas partie du noyau angulaire.
- C'est dans son propre paquet de bibliothèque, **@
angulaire/router**.
- Importez ce dont vous avez besoin comme vous le feriez à partir de tout autre paquet Angular.

src/app /app.module.ts (import)

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

AJOUT DU ROUTING À UN PROJET EXISTANT

The screenshot shows a code editor interface with a sidebar containing a file tree. The file tree shows a project structure with 'src' and 'app' folders. Inside 'app', there are files: 'app-routing.module.ts' (which is selected and highlighted in blue), 'app.component.css', 'app.component.html', 'app.component.spec.ts', 'app.component.ts', and 'app.module.ts'. The main editor area displays the content of 'app-routing.module.ts'. The code is as follows:

```
src > app > A app-routing.module.ts > ...
1 import { NgModule } from '@angular/core';
2 import { Routes, RouterModule } from '@angular/router';
3
4
5 const routes: Routes = [];
6
7 @NgModule({
8   imports: [RouterModule.forRoot(routes)],
9   exports: [RouterModule]
10 })
11 export class AppRoutingModule { }
12
```

The line 'const routes: Routes = [];' is highlighted with a red rectangular box.

IMPORTER LE MODULE ROUTING

The screenshot shows a code editor interface with an 'EXPLORER' sidebar on the left and a main editor area on the right.

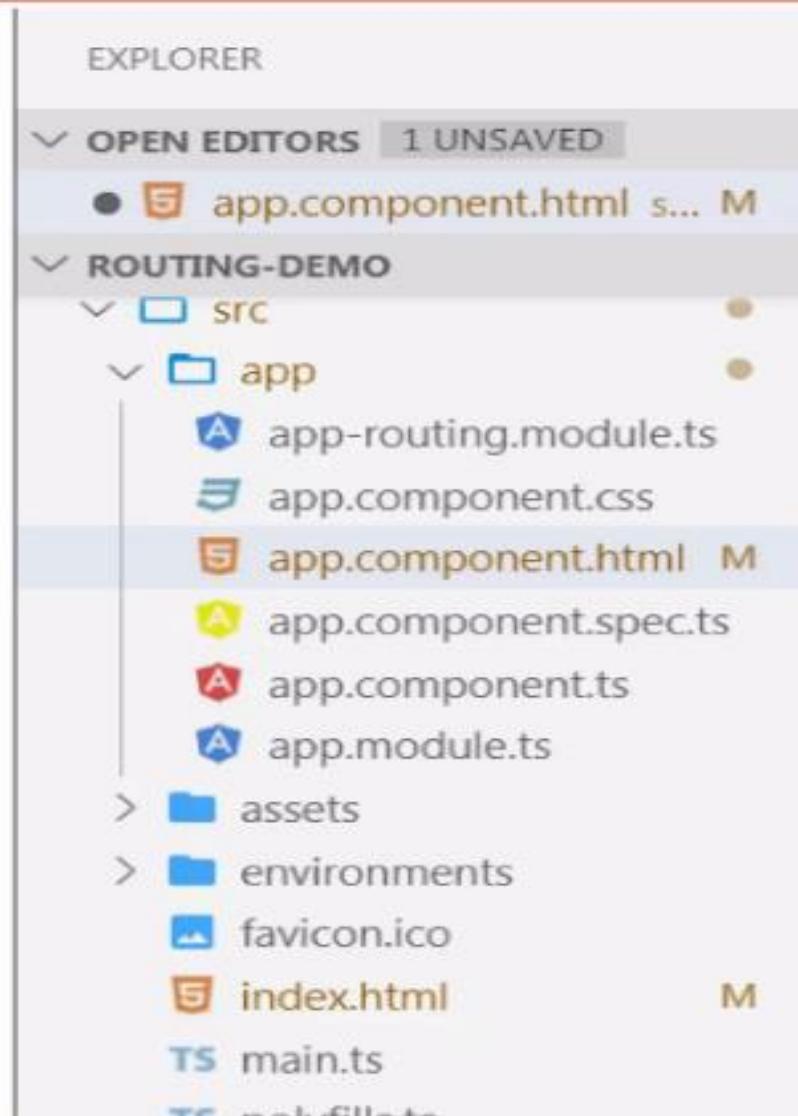
EXPLORER:

- OPEN EDITORS: app.module.ts (src\app)
- ROUTING-DEMO
 - src
 - app-routing.module.ts
 - app.component.css
 - app.component.html
 - app.component.spec.ts
 - app.component.ts
 - app.module.ts (selected)
 - assets
 - environments
 - favicon.ico
 - index.html
 - main.ts
 - polyfills.ts

app.module.ts:

```
src > app > app.module.ts > ...
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3
4 import { AppRoutingModule } from './app-routing.module'; // Line 4 highlighted with a red box
5 import { AppComponent } from './app.component';
6
7 @NgModule({
8   declarations: [
9     AppComponent
10   ],
11   imports: [
12     BrowserModule,
13     AppRoutingModule // Line 13 highlighted with a red box
14   ],
15   providers: [],
16   bootstrap: [AppComponent]
17 })
18 export class AppModule { }
19
```

AJOUTER LA DIRECTIVE <ROUTER-OUTLET>



app.component.html

```
src > app > app.component.html > ...
```

1 <h2>Routing et Navigation</h2>
2
3
4
5 <router-outlet></router-outlet>
6
7 <!--les composants seront injectés ici-->
8
9
10
11
12
13
14
15

• Etant donnée cette configuration,

- Quand l'utilisateur tape :
<http://localhost:4200/about>,
- le routeur cherche et charge le composant AboutComponent et l'affiche dans un élément **<router-outlet>** **</router-outlet>**.
- Cet élément est sensé se trouver dans la vue du composant racine.

GÉNÉRER LES COMPOSANTS

ng generate component departementList

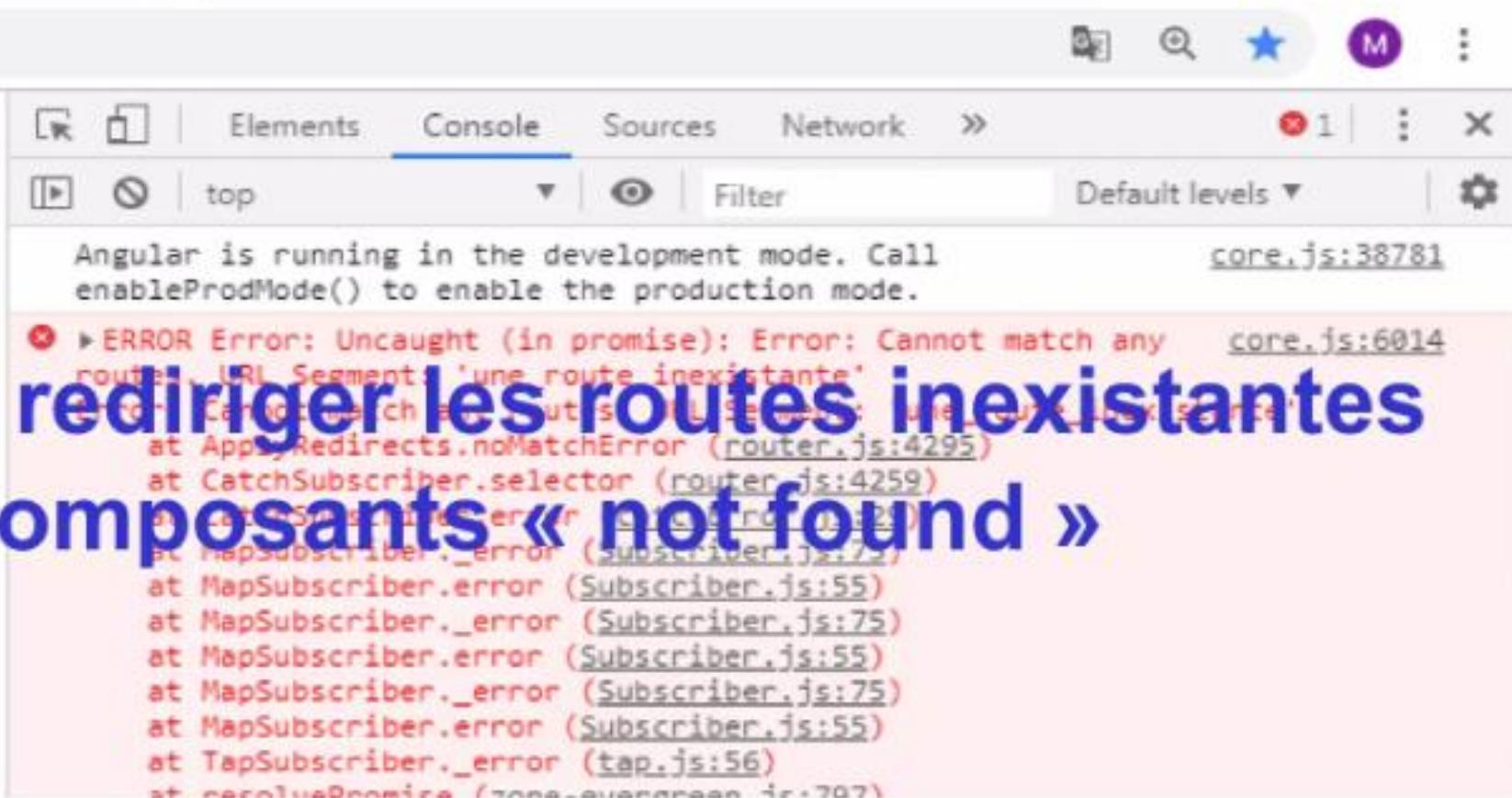
ng generate component employestList

NAVIGATION

```
12
13     <div class="collapse navbar-collapse" id="navbarSupportedContent">
14         <ul class="navbar-nav mr-auto">          You, 2 minutes ago • Uncommitted changes
15             <li class="nav-item active">
16                 |   <a class="nav-link" [routerLink]=["/departements"]> <span class="sr-only">Departements</span></a>
17             </li>
18             <li class="nav-item">
19                 |   <a class="nav-link" [routerLink]=["/employees"]>Employees</a>
20             </li>
21             <li class="nav-item">
22                 |   <a class="nav-link" href="">students</a>
23             </li>
24
25         </ul>
26         <form class="form-inline my-2 my-lg-0">
```

ROUTE DE REDIRECTION

- Très souvent, un utilisateur essaye de naviguer vers une route qui n'est pas configurée.
- Cela génère une exception.



The screenshot shows a browser's developer tools console tab. The console output is as follows:

```
Angular is running in the development mode. Call enableProdMode() to enable the production mode. core.js:38781
✖ > ERROR Error: Uncaught (in promise): Error: Cannot match any routes. URL Segment: 'une route inexistante' core.js:6014
    at CatchSubscriber.selector (router.js:4259)
    at CatchSubscriber._error (subscriber.js:72)
    at MapSubscriber.error (Subscriber.js:55)
    at MapSubscriber._error (Subscriber.js:75)
    at MapSubscriber.error (Subscriber.js:55)
    at MapSubscriber._error (Subscriber.js:75)
    at MapSubscriber.error (Subscriber.js:55)
    at TapSubscriber._error (tap.js:56)
    at resolvePromise (zone-evergreen.js:707)
```

Solution: rediriger les routes inexistantes vers un composants « not found »

EXPLORER

OPEN EDITORS

- page-not-fo... U

ROUTING-DEMO

- src
- app
 - departe...
 - employes...
 - page-not...
 - page-no... U
 - page-no... U
 - page-no... U
 - app-routi... M
 - app.componen...
 - app.com M

page-not-found.component.html ×

src > app > page-not-found > page-not-found.component.html >

```
1 <p>Page not found</p>
2 
```

src > app > A app-routing.module.ts > [listComponents]

```
1 import { NgModule } from '@angular/core';
2 import { Routes, RouterModule } from '@angular/router';
3 import { DepartementListComponent } from './departement-list/departement-list';
4 import { EmployesListComponent } from './employes-list/employes-list.component';
5 import { PageNotFoundComponent } from './page-not-found/page-not-found.component';

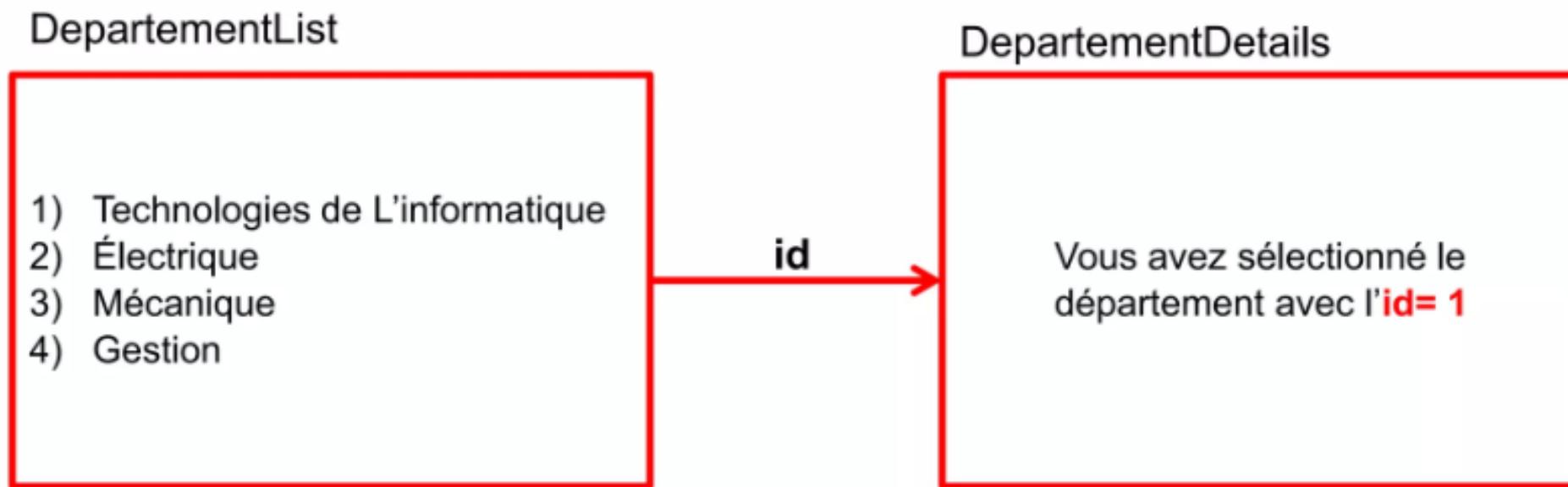
6
7 const routes: Routes = [
8   {path: 'departements', component: DepartementListComponent},
9   {path: 'employes', component: EmployesListComponent},
10  {path: '**', component: PageNotFoundComponent}
11];
12 @NgModule({
13   imports: [RouterModule.forRoot(routes)],
14   exports: [RouterModule]
15 })
16 export class AppRoutingModule { }
```

ROUTE PAR DÉFAUT

app-routing.module.ts ×

```
src > app > app-routing.module.ts > ...
1 import { NgModule } from '@angular/core';
2 import { Routes, RouterModule } from '@angular/router';
3 import { DepartementListComponent } from './departement-list/departement-list.component';
4 import { EmployesListComponent } from './employes-list/employes-list.component';
5 import { PageNotFoundComponent } from './page-not-found/page-not-found.component';
6
7 const routes: Routes = [
8   {path: '', redirectTo: '/departements', pathMatch: 'full'},
9   {path: 'departements', component: DepartementListComponent},
10  {path: 'employes', component: EmployesListComponent},
11  {path: '**', component: PageNotFoundComponent}
12];
13 @NgModule({
14   imports: [RouterModule.forRoot(routes)],
15   exports: [RouterModule]
16 })
17 export class AppRoutingModule { }
18
```

ROUTE AVEC PARAMÈTRE



Générer un composant **DepartementDetails**

Configurer le routage avec paramètres vers DepartementDetails

Dans le composant DepartementList, programmer la navigation vers DepartementDetails

Dans le composant DepartementDetails, extraire l'**id** de la route active.

GÉNÉRER DEPARTEMENTDETAILS

ng generate component departementDetails

CONFIGURER LA ROUTE AVEC PARAMÈTRE ID

```
app-routing.module.ts
src > app > app-routing.module.ts > ...
1 import { NgModule } from '@angular/core';
2 import { Routes, RouterModule } from '@angular/router';
3 import { EmployesListComponent } from './employes-list/employes-list.component';
4 import { PageNotFoundComponent } from './page-not-found/page-not-found.component';
5 import { DepartementListComponent } from './departement-list/departement-list.component';
6 import { DepartementDetailsComponent } from './departement-details/departement-details.component';
7
8 const routes: Routes = [
9   {path: '', redirectTo:'/departements', pathMatch:'full'},
10  {path: 'departements', component: DepartementListComponent},
11  {path: 'departements/:id', component: DepartementDetailsComponent}, // Line 11 highlighted with a red box
12  {path: 'employes', component: EmployesListComponent},
13  {path: '**', component: PageNotFoundComponent}
14];
15 @NgModule({
16   imports: [RouterModule.forRoot(routes)],
17   exports: [RouterModule]
18 })
19 export class AppRoutingModule { }
20
21 export const listComponents = [DepartementListComponent,
22                               EmployesListComponent,
23                               PageNotFoundComponent,
24                               DepartementDetailsComponent] // Line 24 highlighted with a red box
25
```

PROGRAMMER LA NAVIGATION: SERVICE ROUTER

departement-list.component.ts

src > app > departement-list > departement-list.component.ts > DepartementListComponent

```
1 import { Component, OnInit } from '@angular/core';
2 import { Router } from '@angular/router';
3
4 @Component({
5   selector: 'app-departement-list',
6   templateUrl: './departement-list.component.html',
7   styleUrls: ['./departement-list.component.css']
8 })
9 export class DepartementListComponent implements OnInit {
10
11   departements = [
12     {id: 1, nom: 'Technologies de l\'informatique'},
13     {id: 2, nom: 'Electrique'},
14     {id: 3, nom: 'Mecanique'},
15     {id: 4, nom: 'Gestion'}
16   ];
17
18   constructor(private _router:Router) { }
19
20   ngOnInit() {
21   }
22   onSelectDep(departement){
23     this._router.navigate(['/departements', departement.id]);
24 }
```

Injecter le service de routage Router

PROGRAMMER LA NAVIGATION: SERVICE ROUTER

```
5 departement-list.component.html ×  
src > app > departement-list > 5 departement-list.component.html > ...  
1 <div>  
2   liste des départements:  
3 </div>  
4 <div>  
5   <ul class="items">  
6     <li (click)="onSelectDep(departement)" *ngFor="let departement of departements">  
7       <span class="badge">{{departement.id}}</span>{{departement.nom}}  
8     </li>  
9   </ul>  
10 </div>  
11 
```

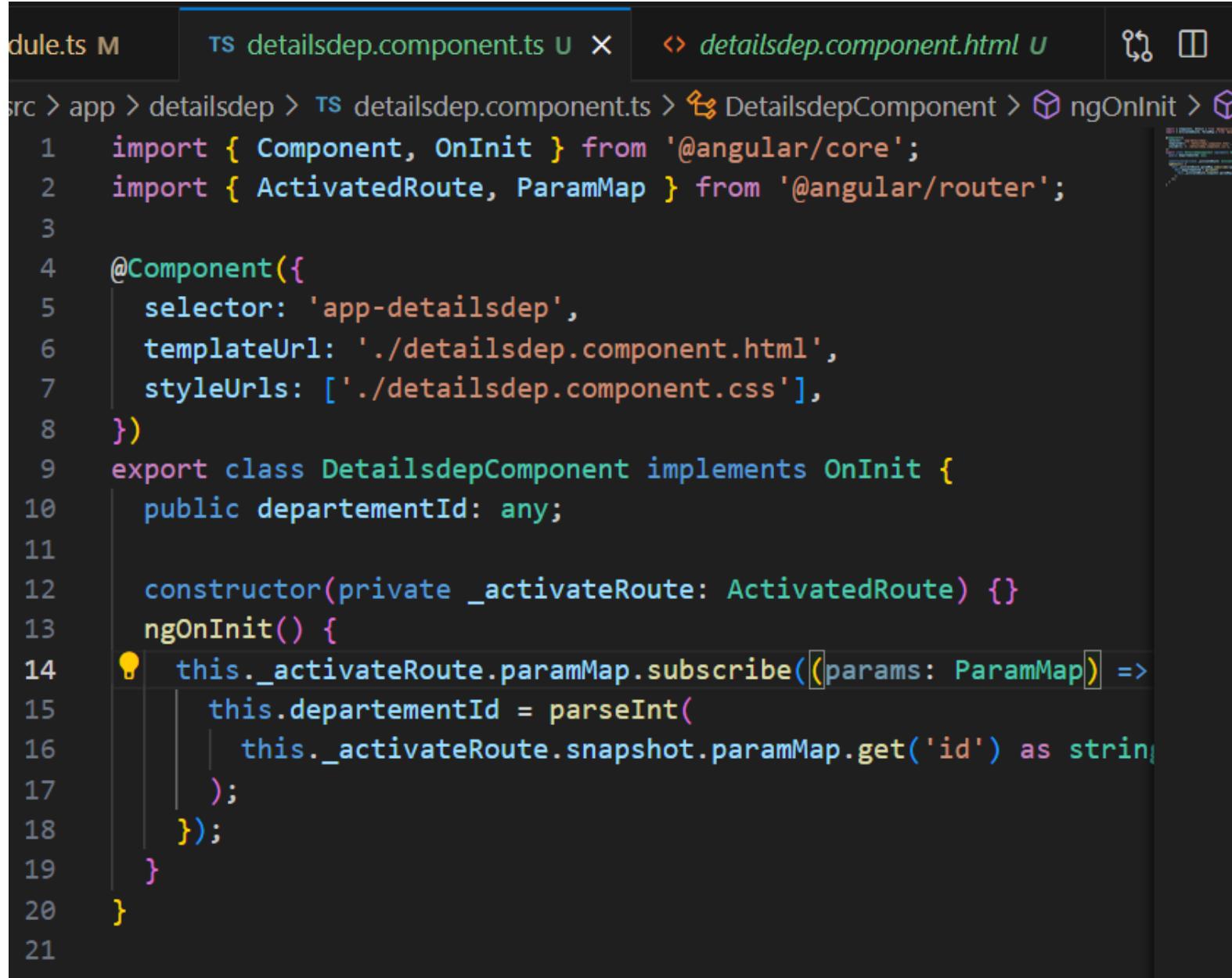
RÉSULTAT

The screenshot displays a web application interface. At the top, there is a navigation bar with links: 'Navbar', 'Departements', 'Employes', 'students', and a search bar containing a 'Search' button. Below the navigation bar, the text 'departement-list works!' is displayed. A list of departments follows, with each item preceded by a bullet point:

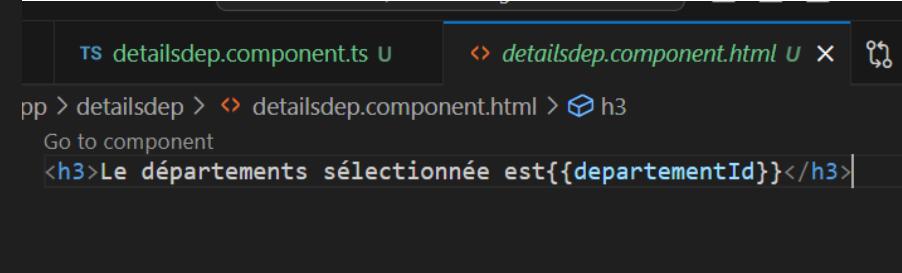
- IT
- Elect
- meca
- gestion

A blue arrow points from the 'meca' item in the list towards the right side of the screen. On the right side, there is a detailed view of a department. The URL in the browser's address bar is 'localhost:4200/departements/3'. The page title is 'departement-list works!'. The content area displays the message 'Le départements sélectionnée est3'.

DEPARTEMENTSDETAILS:EXTRAIRE LE PARAM ID



```
src > app > detailsdep > ts detailsdep.component.ts u x < detailsdep.component.html u ..  
1 import { Component, OnInit } from '@angular/core';  
2 import { ActivatedRoute, ParamMap } from '@angular/router';  
3  
4 @Component({  
5   selector: 'app-detailsdep',  
6   templateUrl: './detailsdep.component.html',  
7   styleUrls: ['./detailsdep.component.css'],  
8 })  
9 export class DetailsdepComponent implements OnInit {  
10   public departementId: any;  
11  
12   constructor(private _activateRoute: ActivatedRoute) {}  
13   ngOnInit() {  
14     this._activateRoute.paramMap.subscribe((params: ParamMap) =>  
15       this.departementId = parseInt(  
16         this._activateRoute.snapshot.paramMap.get('id') as string,  
17         10);  
18     );  
19   }  
20 }  
21
```



```
ts detailsdep.component.ts u < detailsdep.component.html u ..  
pp > detailsdep > detailsdep.component.html > h3  
Go to component  
<h3>Le départements sélectionnée est{{departementId}}</h3>
```

LES SERVICES

INCONVÉNIENTS

Les données sont codés en dur.

Les données sont dupliquées dans les deux composants

Forte possibilité d'incohérence de données

Ajout de responsabilité aux composants (gestion des données)

SOLUTION

- Déporter les données dans un **service**.

Les services sont conçus pour offrir des données et/ou des fonctionnalités partagées entre plusieurs composants ou modules:

- Partage de données
- Partage de fonctions (exemple calcul d'âge)
- Connexion réseau et mise à jour des données persistantes avec un service.

NOUVEAU SERVICE

ng generate service employs

SERVICE EMPLOYS

```
ts employes.service.ts U X   
src > app > Services > ts employes.service.ts >  EmployesService >  getEmployes  
1 import { Injectable } from '@angular/core';  
2  
3 @Injectable({  
4   providedIn: 'root',  
5 })  
6 export class EmployesService {  
7   getEmployes() {  
8     return [  
9       { id: 1, nom: 'alya', age: 30 },  
10      { id: 2, nom: 'larine', age: 5 },  
11      { id: 3, nom: 'Bader', age: 5 },  
12    ];  
13  }  
14  constructor() {}  
15 }
```

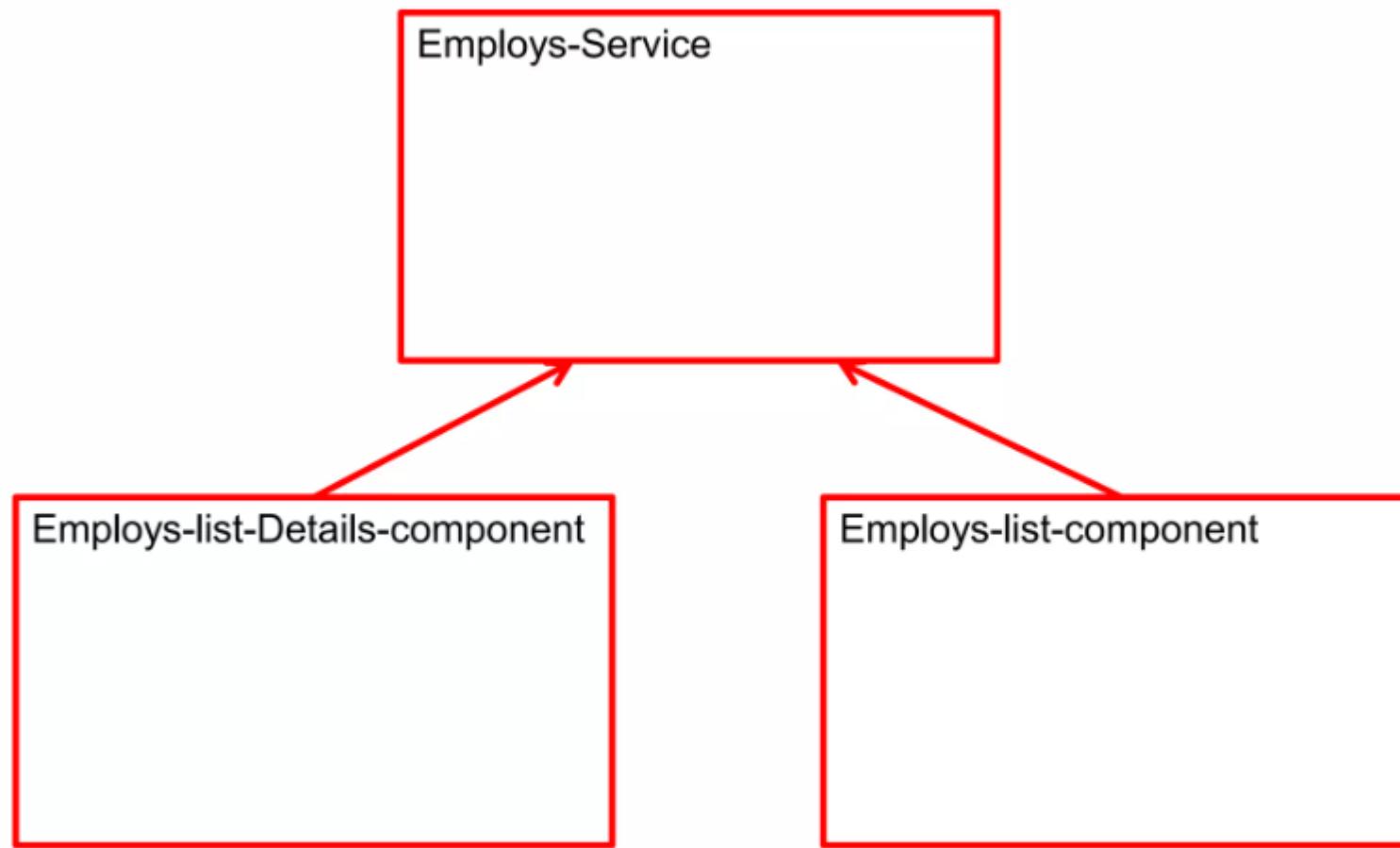
MISE À JOUR DU COMPOSANT EMPLOYS-LIST

```
1 import { EmployService } from './Services/employs.service';
2 import { Component, OnInit } from '@angular/core';
3
4 @Component({
5   selector: 'app-employes-list',
6   templateUrl: './employes-list.component.html',
7   styleUrls: ['./employes-list.component.css'],
8 })
9 export class EmployesListComponent implements OnInit {
10   public employs: any;
11   constructor(private employService: EmployService) {}
12
13   ngOnInit(): void {
14     this.employ = this.employService.getEmploy;
15   }
16 }
```

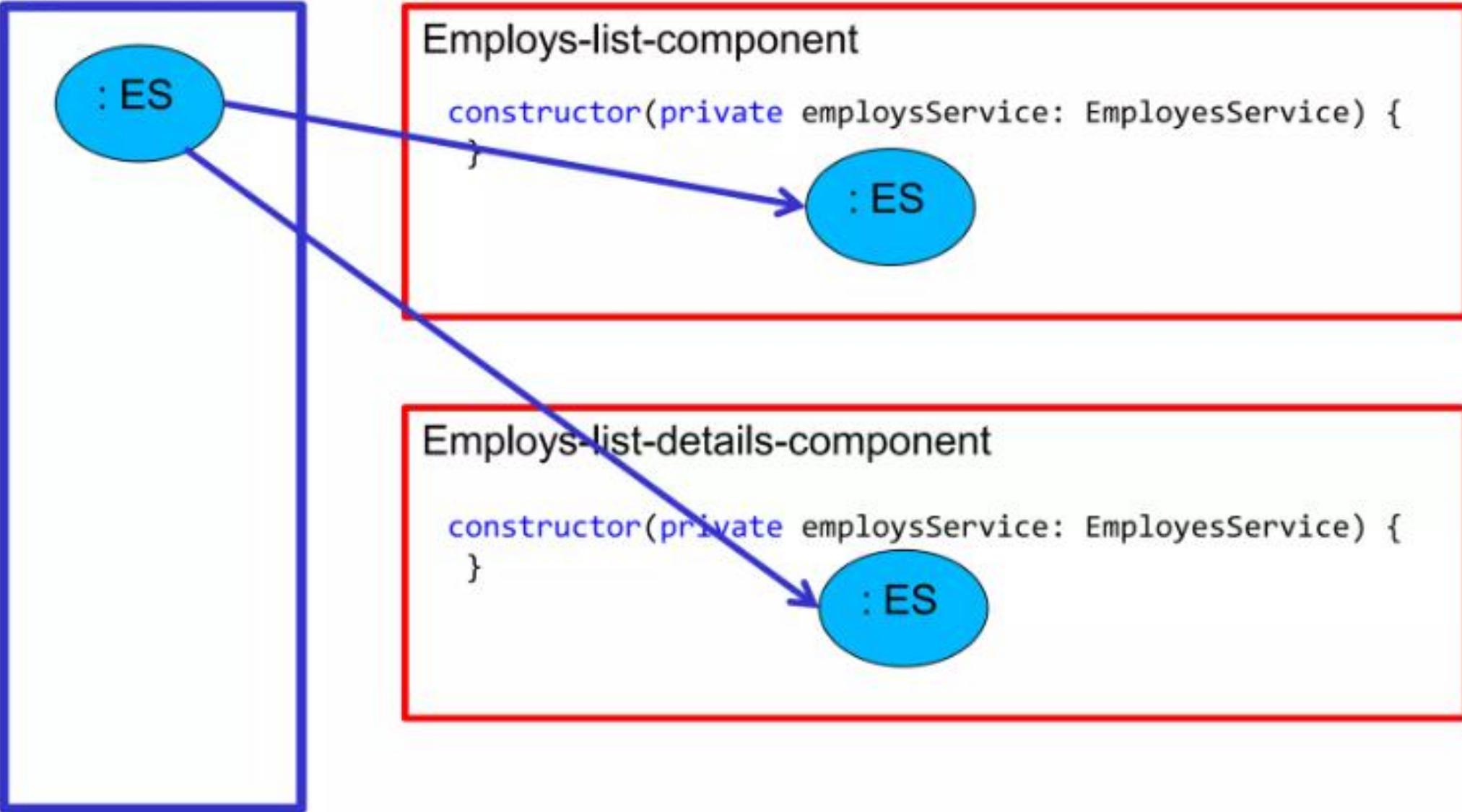
MISE À JOUR DU COMPOSANT EMPLOYS-LIST-DETAILS

```
1 import { Component, OnInit } from '@angular/core';
2 import { EmployService } from '../Services/employs.service';
3
4 @Component({
5   selector: 'app-employs-list-details',
6   templateUrl: './employs-list-details.component.html',
7   styleUrls: ['./employs-list-details.component.css'],
8 })
9 export class EmployListDetailsComponent implements OnInit {
10   public employs: any;
11   constructor(private employService: EmployService) {}
12
13   ngOnInit(): void {
14     this.employs = this.employService.getEmploy;
15   }
16 }
```

NOUVELLE ARCHITECTURE



Injector



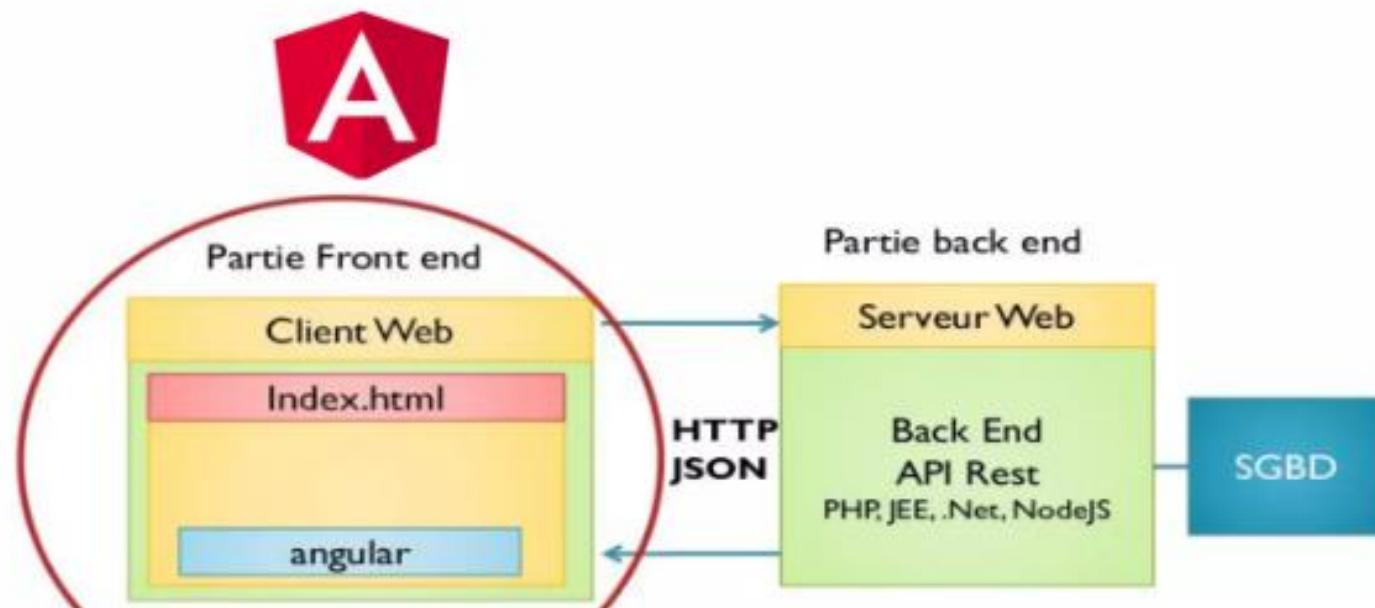
HTTP ET LES OBSERVABLES

HTTP

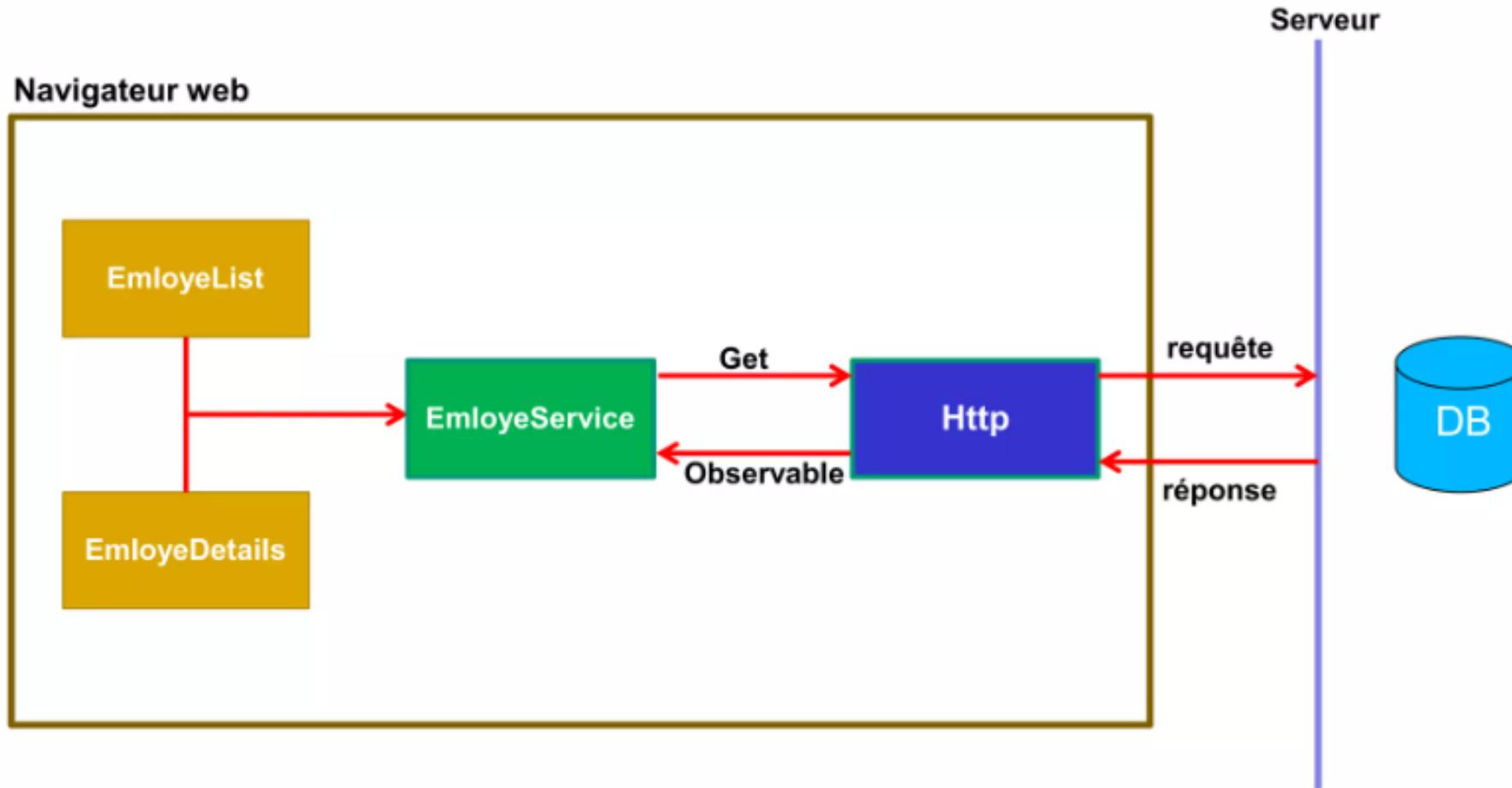
Une application Angular est SPA (Single Page Application)

Une application Angular s'exécute côté client

Elle doit pouvoir se connecter à un serveur pour synchroniser les données métier (demander liste de voitures, ajouter nouveau client, modification, suppression d'une entité métier,...).

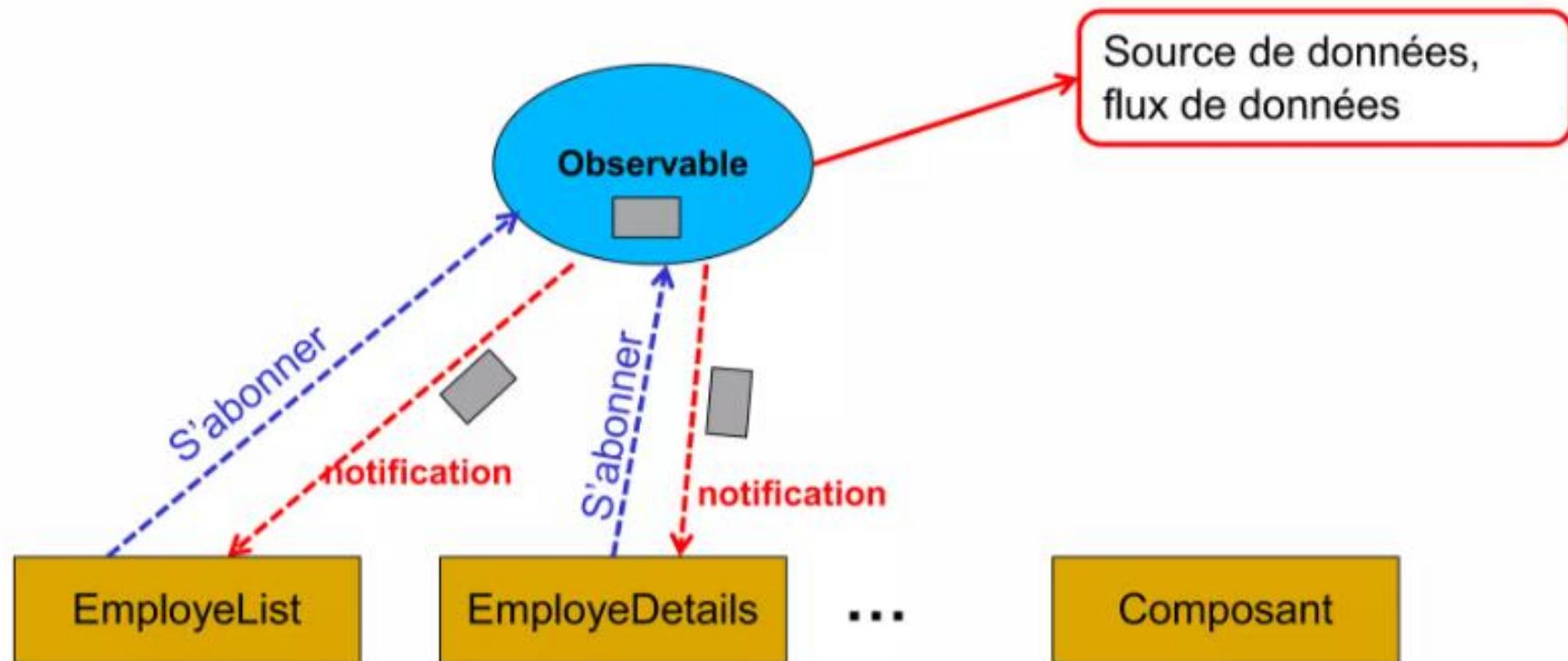


HTTP:MÉCANISME



Les Observables: programmation réactive

- La programme réactive se base sur le concept d'observateur.



IMPORTER HTTPCLIENTMODULE

The screenshot shows a code editor with an Angular application's `app.module.ts` file open. The file defines the `AppModule` which imports `HttpClientModule` from `@angular/common/http`. The module has declarations for `AppComponent`, `EmployesListComponent`, and `EmployesListDetailsComponent`. It also imports `BrowserModule` and `AppRoutingModule`.

```
src > app > TS app.module.ts > ...
...
1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3
4 import { AppRoutingModule } from './app-routing.module';
5 import { AppComponent } from './app.component';
6 import { EmployesListComponent } from './employes-list/employes
7 import { EmployesListDetailsComponent } from './employes-list-det
8 import { HttpClientModule } from '@angular/common/http';
...
9 @NgModule({
10   declarations: [
11     AppComponent,
12     EmployesListComponent,
13     EmployesListDetailsComponent,
14   ],
15   imports: [BrowserModule,
16           AppRoutingModule,
17           HttpClientModule],
18   providers: [],
19   bootstrap: [AppComponent],
20 })
21 export class AppModule {}
```

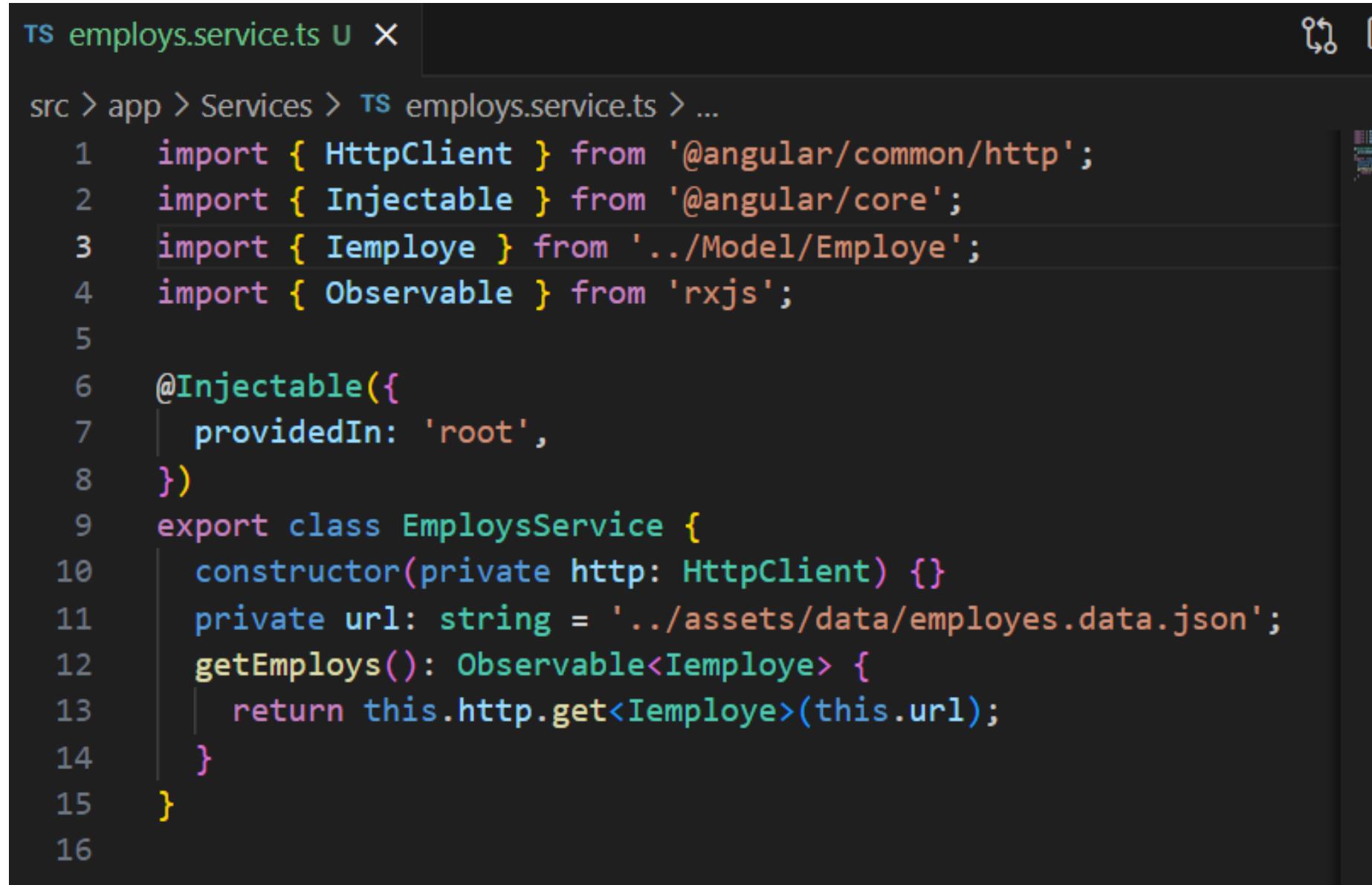
DÉPORTER LES DONNÉES DANS UN FICHIER JSON

```
{} employes.data.json U X
src > app > data > {} employes.data.json > ...
1  [
2      { "id": 1, "nom": "alya", "age": 30 },
3      { "id": 2, "nom": "larine", "age": 5 },
4      { "id": 3, "nom": "Bader", "age": 5 }
5 ]
```

DÉFINIR LE TYPE IEMPLOYE

```
{} employes.data.json U X
src > app > data > {} employes.data.json > ...
1  [
2      { "id": 1, "nom": "alya", "age": 30 },
3      { "id": 2, "nom": "larine", "age": 5 },
4      { "id": 3, "nom": "Bader", "age": 5 }
5 ]
```

METTRE À JOUR EMPLOYEESERVICE



The screenshot shows a code editor window with the file `employes.service.ts` open. The file path is indicated as `src > app > Services > employes.service.ts > ...`. The code itself is a TypeScript class definition for `EmployesService`.

```
ts employes.service.ts U X
src > app > Services > TS employes.service.ts > ...
1 import { HttpClient } from '@angular/common/http';
2 import { Injectable } from '@angular/core';
3 import { Iemploye } from '../Model/Employe';
4 import { Observable } from 'rxjs';
5
6 @Injectable({
7   providedIn: 'root',
8 })
9 export class EmployesService {
10   constructor(private http: HttpClient) {}
11   private url: string = '../assets/data/employes.data.json';
12   getEmployes(): Observable<Iemploye> {
13     return this.http.get<Iemploye>(this.url);
14   }
15 }
16
```

Type de retour:
Observable d'Employé

```
getEmployes(): Observable<IEmployee>{  
    return this.http.get<IEmployee>(this.url);  
}  
}
```

Appel GET au service web en spécifiant
l'url. le type de retour est spécialisé à
Employee

ts employes.service.ts U

ts employes-list.component.ts U X



```
src > app > employes-list > ts employes-list.component.ts > 📁 EmployesListComponent > 📂
1 import { EmployesService } from './Services/employes.service';
2 import { Component, OnInit } from '@angular/core';
3
4 @Component({
5   selector: 'app-employes-list',
6   templateUrl: './employes-list.component.html',
7   styleUrls: ['./employes-list.component.css'],
8 })
● 9 export class EmployesListComponent implements OnInit {
10   public employes: any;
11   constructor(private employesService: EmployesService) {}
12   🌟
13   ngOnInit(): void {
14     //this.employes = this.employesService.getEmployes;
15     this.employesService.getEmployes().subscribe((data) => {
16       this.employes = data;
17     });
18   }
19 }
```

```
ngOnInit() {  
    this.employeesService.getEmployes().subscribe(data => {this.employees=data});  
}
```

The diagram illustrates the execution flow of the code. Three red-bordered boxes contain French annotations:

- Retourne un observable d'employés (Returns an observable of employees) points to the line `this.employeesService.getEmployes()`.
- Abonnement au flux de l'observable (Subscription to the observable stream) points to the line `.subscribe`.
- Traitement du flux par une fonction lambda (Processing the stream via a lambda function) points to the line `data => {this.employees=data})`.

Red arrows connect these annotations to their corresponding code snippets.