

Introduction à Angular

Michel Buffa, Septembre 2022

Partie 1- Introduction

Introduction

- Qu'est ce qu'Angular ?
- Créer sa première application en mode CLI
- Styler les composants avec Angular Material

Qu'est-ce qu'Angular ?

Qu'est-ce que Angular ?

- Angular est un **framework** front-end pour créer des single page WebApps basés sur des WebComponents
 - VueJS et React sont des “gestionnaires de vues...”
- Contrairement à React et VueJS il propose déjà tous les modules “additionnels” classiques de React et Vue (router, gestionnaire d’états centralisés, tests, gestionnaire de modules etc.)
- Angular n'est pas AngularJS !
- Comme Vue et React il est cross platform et permet de développer des Progressive Web Apps (PWAs)
- Mais le standard pour la cross compilation native est React Native...



Points forts

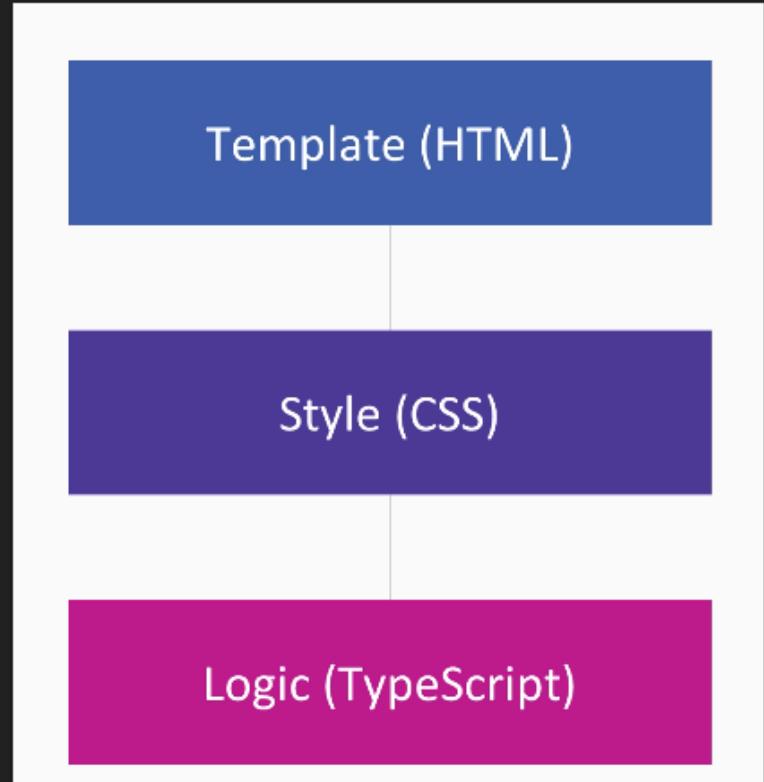
- Utilise les WebComponents
- Basé sur TypeScript (typé) (mais les autres peuvent aussi utiliser TS)
- Excellent support PWA
- CLI propose des commandes pour générer des templates de code (composant, service, module)
 - MÉFIANCE ! Quand on a besoin de cela c'est souvent que le framework est compliqué, syndrome JavaEE !
- Développement rapide, templates puissants
- Modularisation naturelle
- Populaire sur Sophia-Antipolis, voir indeed.com par exemple!
- Adapté aux développeurs aimant les langages typés, Java en particulier (annotations de code, plein de code redondant et inutile etc.)

Points faibles

- Lourd (temps de build, nb de lignes de codes générées), code redondant...
- Loin derrière React en termes de popularité, notamment aux USA
- Dur à debugguer (ex: oubli d'importation d'un module génère des erreurs très difficiles à interpréter)
- TypeScript (ne plaît pas aux purs développeurs JavaScript aimant la programmation fonctionnelle)
- Ne s'est pas imposé comme solution native ou même pour mobiles
 - Aucune killer app connue, contrairement à React...

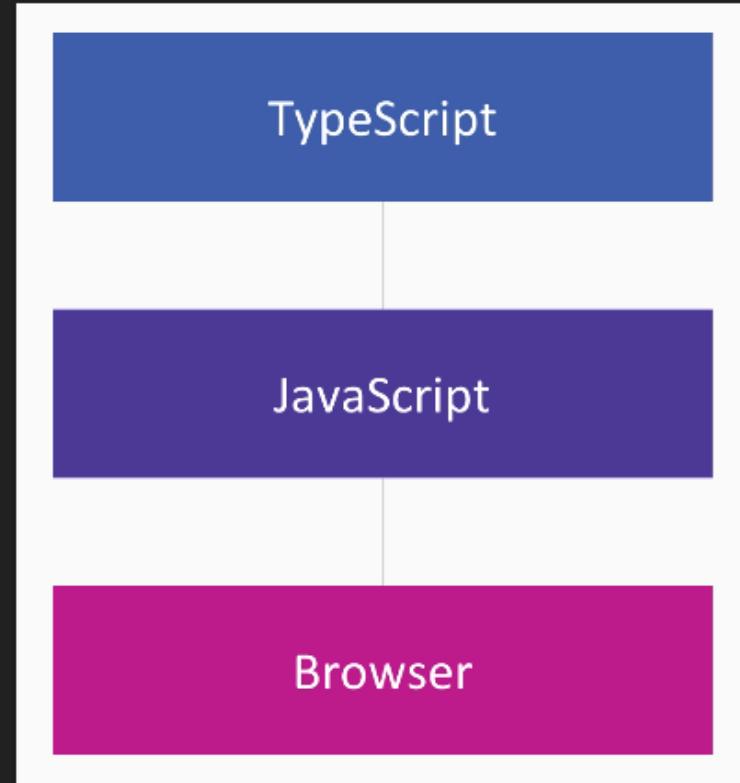
Basé sur le concept de composants

- Une application Angular est une architecture de composants
- Chaque composant est une boîte noire qui propose des fonctionnalités
- Exemples :
 - un composant “Liste de tâches à effectuer”,
 - un composant “Tâche” pour décrire une Tâche, qui a pour composant parent le précédent,
 - un composant “Détail d’une tâche” qui donne une vue détaillée d’une tâche,
 - etc.
- On a une hiérarchie de composants :
 - Un composant “Root”, souvent appelé <App>
 - Des composants parents (i.e ListeDeTaches), et des composants enfants (i.e Tache)
- Un composant : trois parties distinctes (quatre si on ajoute les tests)



TypeScript

- Créé par Microsoft (Office 365 est en TypeScript/React)
 - Remarque amusante : Angular c'est aussi une core team employée par MS
- Ne peut s'exécuter dans le navigateur
- Compilé en JavaScript par Angular pour fonctionner dans le navigateur
- Pose des problèmes de debug...



Aujourd'hui : Angular 14

On a à peu près une nouvelle version par an... parfois les ajouts sont invisibles (mise à jour du compilateur, etc.), parfois importantes. Exemples...

Angular 10 :

- Nouveau composant choix de date dans Angular Material

Angular 7 :

- Meilleure modularisation
- Scrolling virtuel infini (nouveau module)
- Drag'n'drop
- Réduction de la taille du code produit

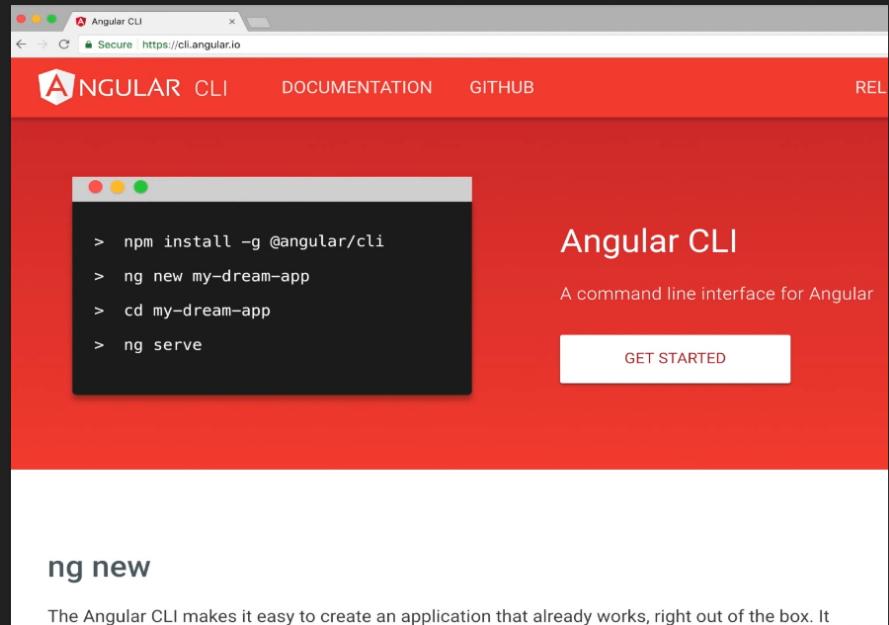
Les “modules”...

- Partie importante du découpage d'une application... on verra plus tard comment créer nous-mêmes des librairies et modules chargeables dynamiquement...
- Nombreux modules standards (pour les formulaires, pour Ajax, pour le scrolling, pour angular material etc.)
- Il faut connaître les modules principaux
- On déclare les modules utilisés dans un fichier app.module
- Un module peut utiliser d'autres modules...
- On a donc une hiérarchie de composants, mais aussi de modules (!)
 - Ni VueJS ni React n'ont cette approche...

Créer sa première application en mode CLI

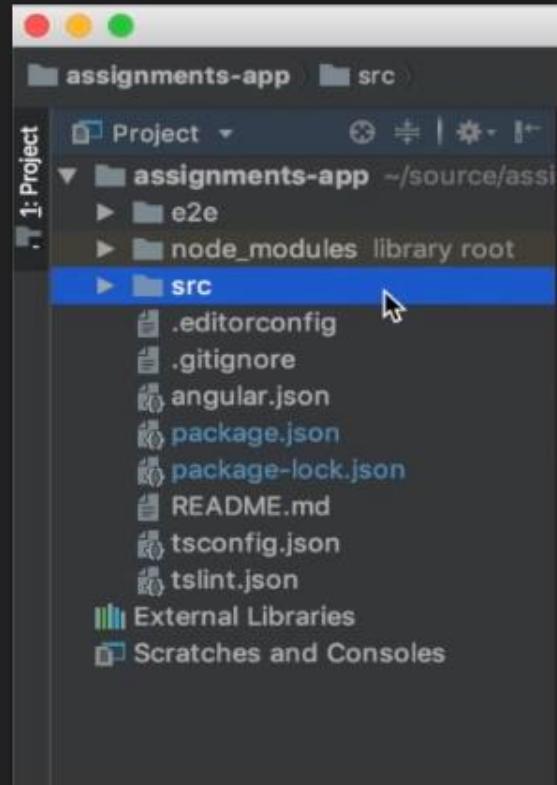
De quoi a-t-on besoin ?

- NodeJS et npm à jour
- Un bon éditeur de code source
(recommandé : Visual Studio Code ou WebStorm ou équivalent)
- SAVOIR QU'ON NE DÉVELOPPE PAS UNE APPLI SANS AVOIR TOUT LE TEMPS LE DEBUGGER DU NAVIGATEUR OUVERT
- ET L'OPTION “disable cache when devtools open” ACTIVÉE !
- Savoir utiliser les devtools (console, network, debugger, inspecter les éléments, etc.)
- De bookmarker le site <https://angular.io/> et <https://cli.angular.io/>



Allez ! Au travail !

1. Installez angular CLI (en super utilisateur):
 - **npm install -g @angular/cli**
 - Note: Être en mode admin ou sudo
 - Répondre aux questions en appuyant sur ENTRÉE (default)
2. Créer un premier projet :
 - **ng new assignment-app**
3. Ouvrir Visual Studio sur le répertoire créé
4. La suite se passe en live coding... on regarde dans un premier temps ce qui a été généré...



Exécuter le projet !

Dans Visual Studio :

- Activer l'auto-save,
- Lancez les commandes dans les Terminaux !

The screenshot shows the Visual Studio Code interface. On the left is the Explorer sidebar with files like .browserslistrc, .editorconfig, .gitignore, angular.json, karma.conf.js, package-lock.json, package.json (which is selected), README.md, tsconfig.app.json, tsconfig.json, tsconfig.spec.json, and tslint.json. The main editor area shows the package.json file with the following content:

```
4  "scripts": {  
5    "ng": "ng",  
6    "start": "ng serve",  
7    "build": "ng build",  
8    "test": "ng test",  
9    "lint": "ng lint",  
10   "e2e": "ng e2e"  
11 },  
12 "private": true,  
13 "dependencies": {  
14   "@angular/animations": "~11.0.0",  
15   "@angular/common": "~11.0.0",  
16   "@angular/compiler": "~11.0.0",  
17   "@angular/core": "~11.0.0",  
18   "@angular/forms": "~11.0.0",  
19   "@angular/platform-browser": "~11.0.0".  
20 }  
21 }
```

To the right of the editor, there are several annotations in red:

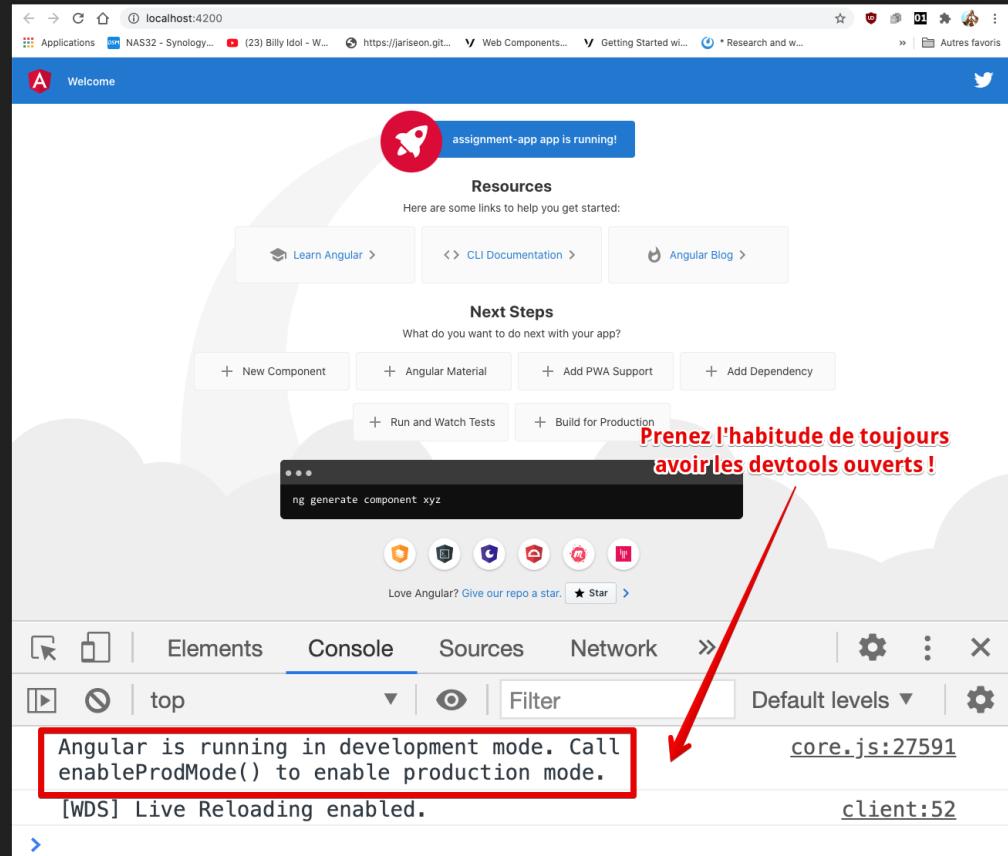
- "ng serve pour lancer." (ng serve to launch.)
- "Laissez le comme ça ce terminal !" (Leave it like this in the terminal!)
- "La commande "surveille" le répertoire et recompile dès qu'un fichier est modifié" (The "serve" command monitors the directory and recompiles whenever a file is modified.)

At the bottom, the terminal window shows the command being run: "(base) mbuffa2@buffy 2-IntroAngular % ng serve".

Ouvrez le projet !

Dans le browser :

- Ouvrez le lien indiqué (<http://localhost:4200/>)
- Cliquez sur les boutons, regardez !
- Modifiez du code par exemple dans `src/app/app.component.ts.html` et vérifiez que la page se recharge



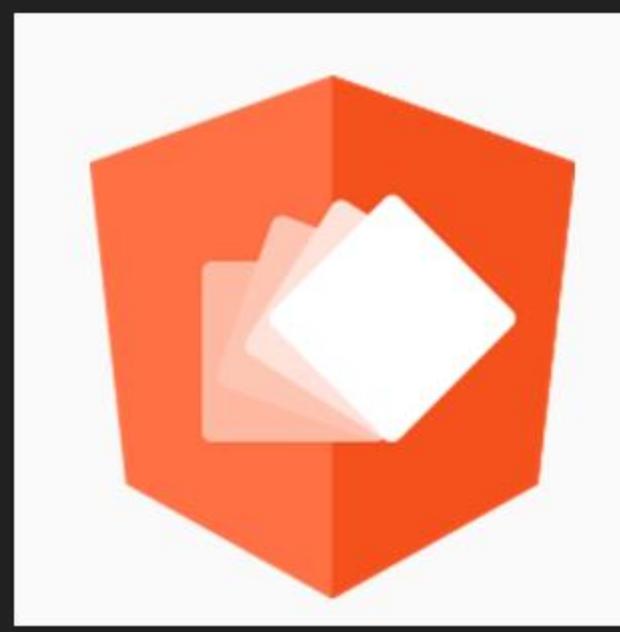
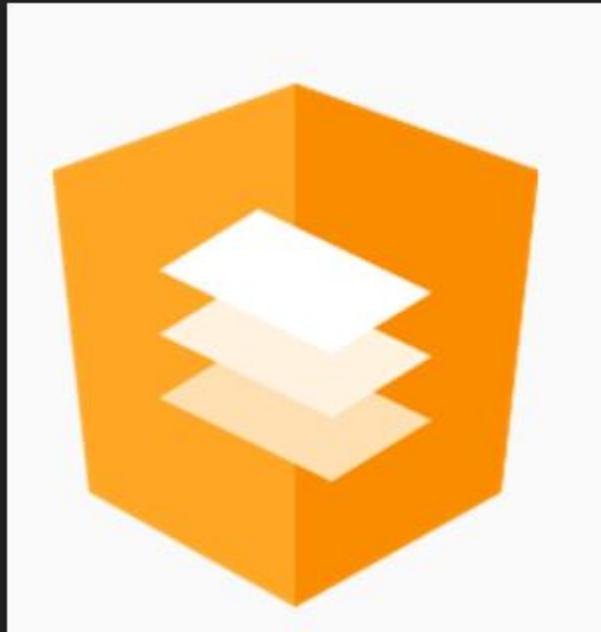
Ajouter le projet sur github

Le plus simple :

1. Allez sur votre compte github et ajoutez un nouveau repository. Dans mon cas : **angularFrontM1Miage2022_2023**
2. Laissez toutes les options par défaut
3. Github vous affiche plusieurs options, choisissez celle qui consiste à importer un dossier existant (copiez les trois lignes de code, ouvrez un second terminal dans VSCode et faites cd dans le dossier du projet. Collez et exécutez les lignes copiées)
4. Faites `git add .` puis `git commit -m "first commit"` puis `git push` pour pousser votre projet sur github.

Styler son application avec Angular Material

Angular Material et Animations, de même que les thèmes, vous permettront de styler vos applications



Cliquer “Get started” et suivre les instructions d’installation

The screenshot shows the Angular Material website (material.angular.io) in a web browser. The page has a dark blue header with navigation links for "Components", "CDK", and "Guides". On the right side of the header, it shows the version "11.0.0" and a dropdown menu icon. Below the header, the main content area features a large white "Angular Material" title and a subtitle "Material Design components for Angular". A prominent "Get started" button is centered at the bottom of the main section. The background of the main content area is a dark blue gradient with abstract white geometric shapes.

material.angular.io

NAS32 - Synology... (23) Billy Idol - W... https://jariseon.git... Web Components... Getting Started wi... * Research and w... michel-wams/mic...

Components CDK Guides 11.0.0

Angular Material

Material Design components for Angular

Get started

Etapes

1. Ouvrir un nouveau terminal dans Visual Studio Code
2. Dans le répertoire du projet, installe Angular Material et Animations:
 - **ng add @angular/material**
 - Répondre aux questions en appuyant sur ENTRÉE (default)
3. Vérifier dans le fichier **package.json** que les dépendances sont bien là.

```
assignment-app > {} package.json > {} scripts
10   "eze": "ng eze"
11 },
12 "private": true,
13 "dependencies": {
14   "@angular/animations": "~11.0.0",
15   "@angular/cdk": "^11.0.0",
16   "@angular/common": "~11.0.0",
17   "@angular/compiler": "~11.0.0",
18   "@angular/core": "~11.0.0",
19   "@angular/forms": "~11.0.0",
20   "@angular/material": "^11.0.0",
21   "@angular/platform-browser": "~11.0.0",
22   "@angular/platform-browser-dynamic": "~11.0.0",
23   "@angular/router": "~11.0.0",
24   "rxjs": "~6.6.0",
25   "tslib": "^2.0.0",
26   "zone.js": "~0.10.2"
27 }
```

Thèmes ?

Parmi les questions posées auxquelles vous avez répondu par ENTREE, il y avait le thème graphique choisi... vous pourrez en tester d'autres par la suite. La documentation est dans la page Get Started :

The screenshot shows a browser window displaying the Angular Material 'Getting Started' guide at material.angular.io/guide/getting-started. The page has a blue header with navigation links for 'Material', 'Components', 'CDK', and 'Guides'. The main content area starts with a heading 'For existing applications, follow the steps below to begin using Angular Material.' Below it is a section titled 'Install Angular Material' with instructions to use the Angular CLI's 'ng add @angular/material' command. A code block shows the command: `ng add @angular/material`. Further down, it explains that the command will install Angular Material, the Component Dev Kit (CDK), Angular Animations, and ask for feature inclusion. A numbered list item 1. Choose a prebuilt theme name, or "custom" for a custom theme is shown, with a callout box highlighting the text: 'You can choose from prebuilt material design themes or set up an extensible custom theme.'

← → ⌛ 🏠 🔒 material.angular.io/guide/getting-started

Applications NAS32 - Synology... (23) Billy Idol - W... https://jariseon.git... Web Components... Getting Started wi... * Research and w... michel-wams/mic...

Material Components CDK Guides

For existing applications, follow the steps below to begin using Angular Material.

Install Angular Material

Use the Angular CLI's install [schematic](#) to set up your Angular Material project by running the following command:

```
ng add @angular/material
```

The `ng add` command will install Angular Material, the [Component Dev Kit \(CDK\)](#), [Angular Animations](#) and ask you the following questions to determine which features to include:

1. Choose a prebuilt theme name, or "custom" for a custom theme:

You can choose from [prebuilt material design themes](#) or set up an extensible [custom theme](#).

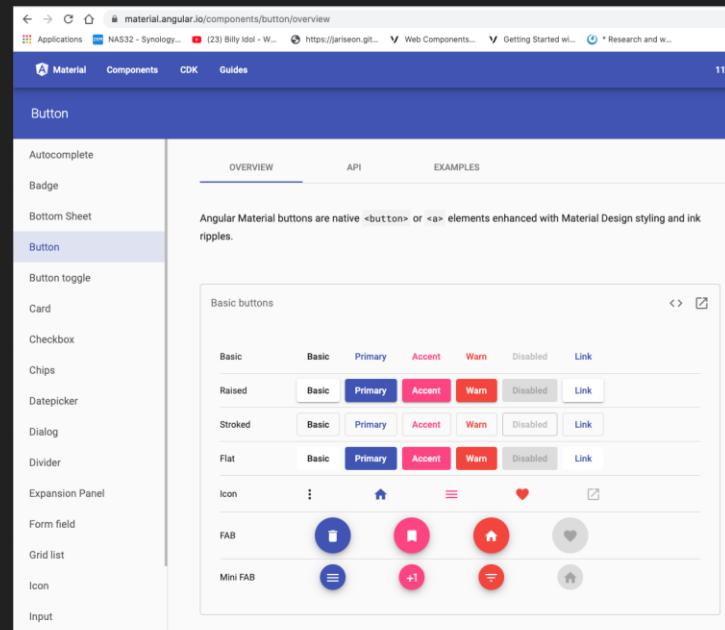
Guide Content

Install Angular Material
Display a component

Angular Material : types de composants

- Formulaires et input fields
- Navigation
- Layout
- Boutons et indicateurs
- Popups et Modals (dialogues bloquants)
- Tables
- Etc.
- Prenez le temps de visiter et tester :

<https://material.angular.io/components/categories>



Création d'un premier composant stylé

Etape 1 : ajouter **MatButtonModule** dans le fichier src/app/app.module.ts



```
TS app.module.ts ×  
assignment-app > src > app > TS 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 { BrowserAnimationsModule } from '@angular/platform-browser';  
6  import { MatButtonModule } from '@angular/material/button';  
7  
8  @NgModule({  
9    declarations: [  
10      AppComponent  
11    ],  
12    imports: [  
13      BrowserModule,  
14      BrowserAnimationsModule,  
15      MatButtonModule  
16    ],  
17    providers: [],  
18    bootstrap: [AppComponent]  
19 })
```

Création d'un premier composant stylé

Etape 2 : effacer le contenu du template HTML du composant “root”
`src/app/app.component.html` et mettre ceci à l’intérieur :

```
<H1>Hello World</H1>
```

```
<button mat-button>Click</button>
```

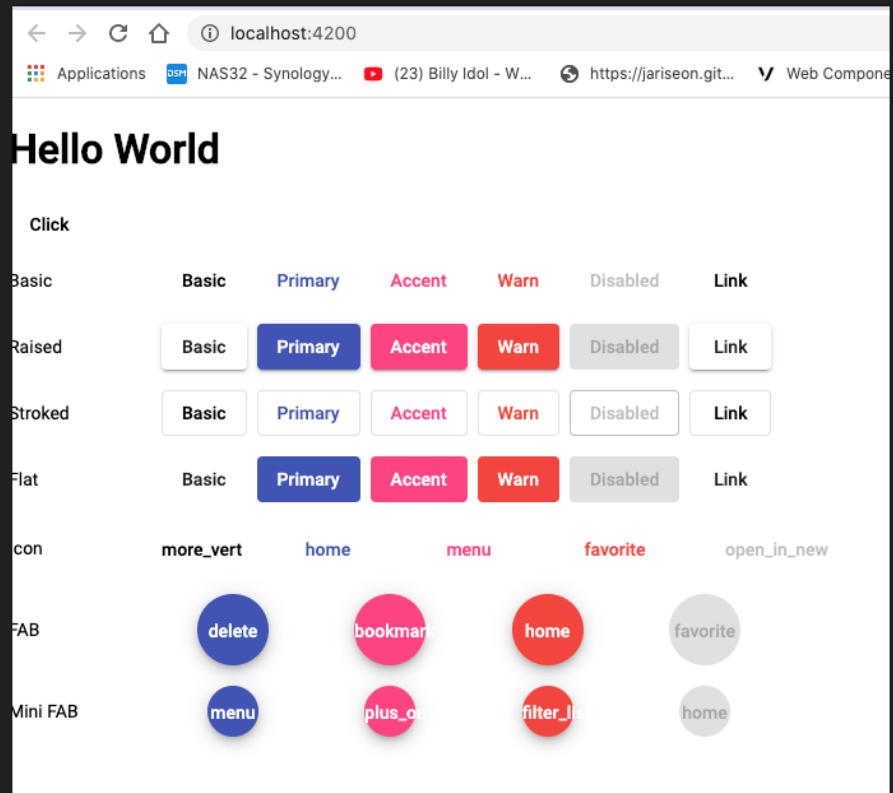
Regardez le résultat, cliquez le bouton, vous devez voir une animation.

Création d'un premier composant stylé

Etape 3 : allez sur la page d'exemple des boutons sur le site material.angular.io et copiez-collez le code HTML de l'exemple dans le template du composant app

Faites de même avec la partie CSS, copiez-là dans app.component.css

Testez !



Mais, ça marche ou bien ça ne marche pas ???

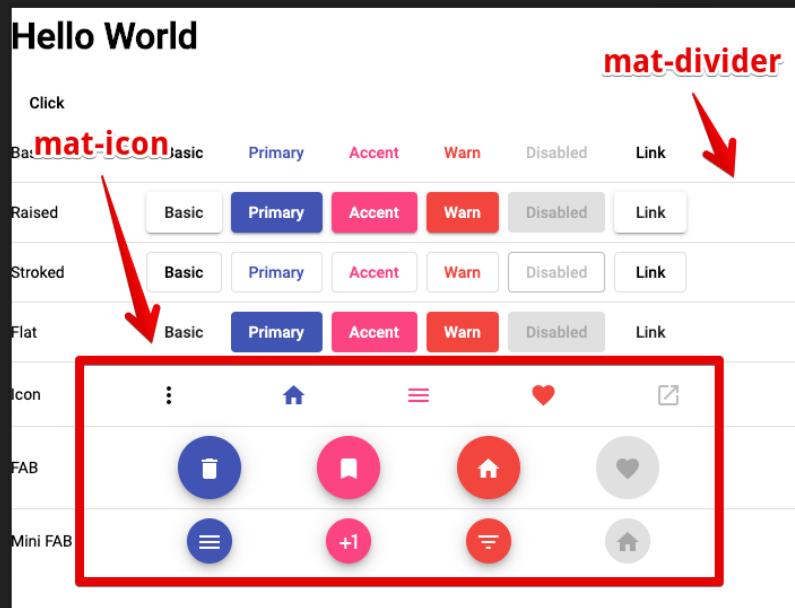
Vous avez bien regardé les erreurs dans le terminal de compilation ?

Dans la console du navigateur ?

Que s'est-il passé ?

Exercice : faire marcher l'exemple pour qu'il ressemble à ce qui est sur la droite et qu'il n'y ait plus de messages d'erreurs.

Angular est TRÈS CAPRICIEUX. Parfois il faut relancer "ng serve" pour que ça compile. Oublier des modules casse souvent la recompilation à chaud.



Et oui il faut importer et déclarer les modules dans app.module.ts

The screenshot shows a GitHub commit interface with three tabs at the top: app.component.css, styles.css, and app.module.ts. The app.module.ts tab is active, showing the following code:

```
# app.component.css M # styles.css M TS app.module.ts M X ↗ ↘ ⌂ ⌂ ⌂
src > app > TS app.module.ts > AppModule
6 import { MatButtonModule } from '@angular/material/button';
7 import { MatIconModule } from '@angular/material/icon';
8 import { MatDividerModule } from '@angular/material/divider';
9
You, 2 minutes ago | 1 author (You)
10 @NgModule({
11   declarations: [
12     AppComponent
13   ],
14   imports: [
15     BrowserModule,
16     BrowserAnimationsModule,
17     MatButtonModule, MatIconModule, MatDividerModule
18   ],
19   providers: [],
20   bootstrap: [AppComponent]
21 })
```

A red box highlights the imports section, specifically the lines: `import { MatIconModule } from '@angular/material/icon';` and `import { MatDividerModule } from '@angular/material/divider';'`. Another red box highlights the module names `MatIconModule` and `MatDividerModule` within the imports array.

Partie 2 - Cr ation de composants

Ce que l'on va voir dans cette section

- Création de composants avec CLI
- Directives
- Styling
- Data Binding
- Transmission de données d'un composant à un autre

Première partie : live coding

Création d'un composant

Afficher le composant dans un composant parent

Utiliser “l'interpolation” pour afficher les données de variables TypeScript dans un template : MVC -> MVVM (Model-View-View-Model c'est-à-dire le MVC quand on a pas besoin de le faire! C'est le “binding” automatique entre variables et vue)

Création d'un composant avec CLI

Dans un terminal (typiquement dans VS Code),

Dans le répertoire racine du projet :

- `ng generate component <nom-du-composant>`, peut-être abrégé par
`ng g c <nom-du-composant>`
- Si on ne veut pas générer le fichier servant aux tests unitaires, la commande recommandée est donc :

`ng g c --skip-tests <nom-du-composant>`

- **EXECUTEZ DONC :**

`ng g c --skip-tests assignments`

Création d'un composant avec CLI

The screenshot shows the Visual Studio Code interface during the creation of an Angular component. On the left, the Explorer sidebar displays the project structure under '2-INTROANGULAR'. A red box highlights the 'assignments' folder, which contains three files: 'assignments.component.css', 'assignments.component.html', and 'assignments.component.ts'. The 'assignments.component.html' file is currently selected and highlighted with a blue background. On the right, the code editor shows the generated component code:

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

Création d'un composant avec CLI: app.module.ts

The screenshot shows a code editor with four tabs at the top: 'TS app.module.ts X', '<> app.component.html', '# styles.css', and 'TS assignments.component.ts'. Below the tabs, the file structure is shown: 'assignment-app > src > app > TS app.module.ts > AppModule'. The code in 'app.module.ts' is as follows:

```
8 import { MatDividerModule } from '@angular/material/divider';
9 import { AssignmentsComponent } from './assignments/assignments.component'
10
11 @NgModule({
12   declarations: [
13     AppComponent,
14     AssignmentsComponent
15   ],
16   imports: [
17     BrowserModule,
18     BrowserAnimationsModule,
19     MatButtonModule, MatIconModule, MatDividerModule
20   ],
21   providers: [],
22   bootstrap: [AppComponent]
23 })
24 export class AppModule { }
```

Annotations highlight specific parts of the code:

- A red box surrounds the import statement for 'AssignmentsComponent':

```
import { AssignmentsComponent } from './assignments/assignments.component'
```
- A red box surrounds the declaration of 'AssignmentsComponent' in the NgModule's declarations array:

```
declarations: [
  AppComponent,
  AssignmentsComponent
```
- A red arrow points from the text 'Ce composant pourra être utilisé par n'importe quel autre composant du module !' to the 'AssignmentsComponent' declaration in the NgModule's declarations array.
- The text 'Ce composant pourra être utilisé par n'importe quel autre composant du module !' is displayed in red.

Afficher le composant dans le composant App

App est le composant “root” par défaut, c'est lui qui est affiché (voir app.component.ts et index.html ci-dessous)

TS app.module.ts TS app.component.ts X <> app.compon

assignment-app > src > app > TS 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 = 'assignment-app';
10 }
11
```

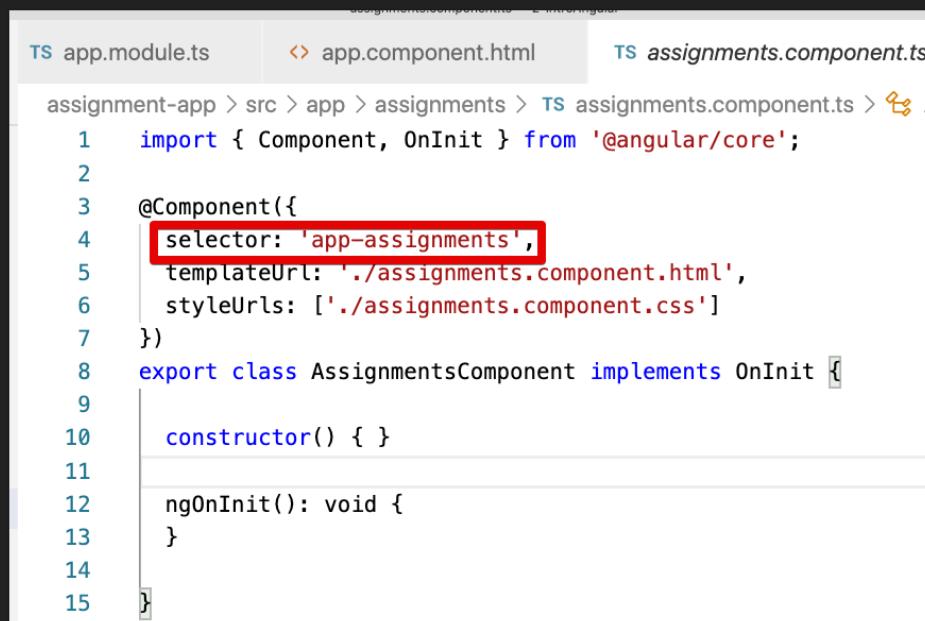
TS app.module.ts TS app.component

assignment-app > src > <> index.html >

```
5 <title>AssignmentApp</title>
6 <base href="/">
7 <meta name="viewport" content="width=device-width, initial-scale=1">
8 <link rel="icon" type="image/x-icon" href="https://fontawesome.com/v4.7.0/icon/regular/icon?text=AssignmentApp" />
9 <link href="https://fonts.googleapis.com/css?family=Material+Icons" />
10 <link href="https://fonts.googleapis.com/icon?family=Material+Icons" />
11
12 </head>
13 <body>
14   <app-root></app-root>
15 </body>
16 </html>
17
```

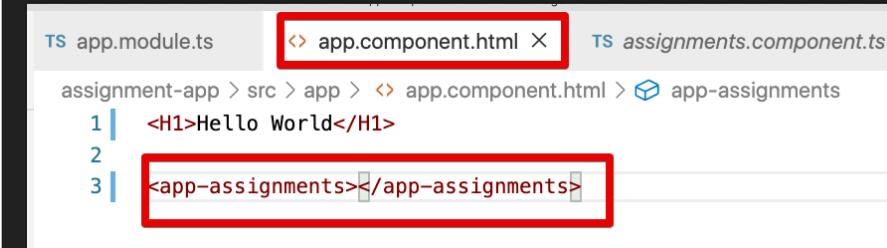
On va donc modifier le template de App

... et ajouter le sélecteur de Assignment (i.e un custom HTML Element, c'est un WebComponent)



```
TS app.module.ts    <> app.component.html    TS assignments.component.ts

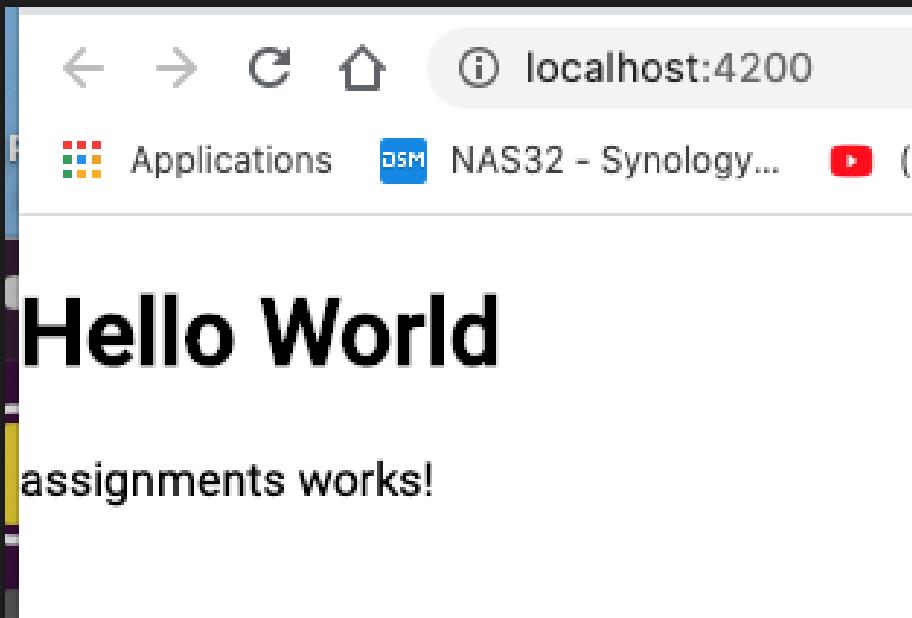
assignment-app > src > app > assignments > TS assignments.component.ts > A
1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-assignments',
5   templateUrl: './assignments.component.html',
6   styleUrls: ['./assignments.component.css']
7 })
8 export class AssignmentsComponent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



```
TS app.module.ts    <> app.component.html X    TS assignments.component.ts

assignment-app > src > app > <> app.component.html > app-assignments
1 <H1>Hello World</H1>
2
3 <app-assignments></app-assignments>
```

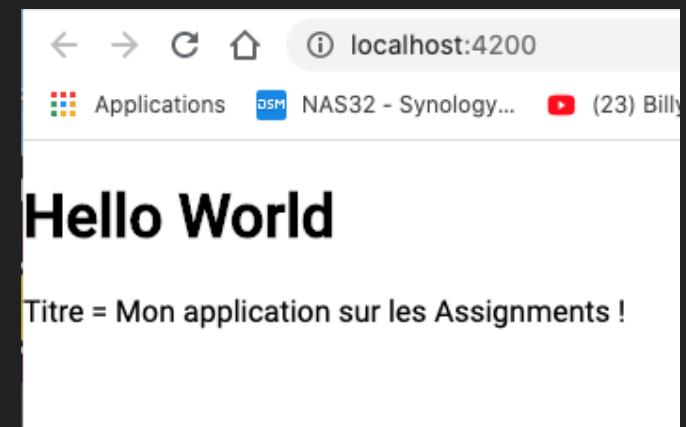
Et voir ce que cela donne....



MVC / Interpolation ajout d'une propriété et affichage dans le template

Ajouter dans assignments.component.ts :

```
export class AssignmentsComponent implements OnInit {  
  titre = "Mon application sur les Assignments !"  
  //...
```



Et dans le template Assignments.components.html :

```
<p>Titre = {{titre}}</p>
```

Allons un peu plus loin...

- Afficher des listes / Collections avec ***ngFor**
 - Utiliser ***ngIf** pour de l'affichage conditionnel de données
 - Compléter l'utilisation de ***ngIf** avec l'utilisation d'un “else”
-
- Ce sont des... “directives” que l'on va utiliser dans les fichiers de template HTML des composants.

Que sont les directives ?

Il y en a de trois sortes :

1. **Structurelles** : elles manipulent le DOM en ajoutant ou en déplaçant des éléments (i.e ***ngFor** ou ***ngIf**)
2. **Attributs** : elles changent l'apparence ou le comportement des composants (**ngClass**, **ngStyle**)
3. **Composant** : ce sont des directives avec un template

Hello World

Devoir intitulé TP1 sur WebComponents, un lecteur audio amélioré, à rendre le 2020-11-17, soumis par l'élève : true.

Devoir intitulé TP2 sur Angular, un joli gestionnaire de devoirs (Assignments), à rendre le 2020-12-15, soumis par l'élève : false.

Devoir intitulé TP3 sur Angular, utilisation du router et de Web Services, à rendre le 2021-01-04, soumis par l'élève : false.

*ngIf

***ngIf** est utilisé pour afficher des données en fonction d'une condition,
else peut être utilisé pour afficher un **ng-template**

Version sans le **else** :

```
<div *ngFor="let assignment of assignments">
    <p *ngIf="assignment.rendu">
        Devoir intitulé {{assignment.nom}}, rendu le
        {{assignment.dateDeRendu}} .
    </p>
</div>
```

*ngIf avec else et <ng-template>

else peut être utilisé pour afficher un **ng-template**

Version avec le **else** :

```
<div *ngFor="let assignment of assignments">
  <p *ngIf="assignment.rendu; else nonsoumis">
    Devoir intitulé {{assignment.nom}}, rendu le
    {{assignment.dateDeRendu}} .
  </p>
  <ng-template #nonsoumis>
    Le devoir {{assignment.nom}} n'a pas été rendu.
  </ng-template>
</div>
```

Partie 3 - styling CSS conditionnel

Ce que l'on va voir

Styler des éléments avec la directive **ngStyle**

Styler des éléments avec la directive **ngClass**

Styler des éléments avec nos propres directives custom

ngStyle

Utilisées avec [...], par exemple :

```
<div *ngFor="let assignment of assignments"
      [ngStyle]="{'color': 'green'}">
```

>

Attention à l'utilisation des simples et double quotes !

Live coding ! On veut les assignments soumis en vert, les autres en rouge...

ngStyle avec une expression

```
<div *ngFor="let assignment of assignments"
      [ngStyle]="{'color': assignment.rendu ? 'green' : 'red'}">
```

Live coding ! On veut les assignments soumis en vert, les autres en rouge...

Parfois il faut arrêter ng serve et relancer (Grrrrrrr !)

ngClass

Permet d'ajouter dynamiquement une classe CSS à un élément...

On va créer dans le fichier **assignments.component.css** deux classes CSS “rendu” et “nonRendu” :

```
# assignments.component.css X
assignment-app > src > app > assignments.component.css
1   .rendu {
2     color: green;
3   }
4
5   .nonRendu {
6     color: red;
7 }
```

ngClass

Et on modifie le template du composant :

```
<div *ngFor="let assignment of assignments"
      [ngClass]="{'rendu':assignment.rendu, 'nonRendu': !assignment.rendu}"
>
```

Attention, j'ai du redémarrer plusieurs fois ng serve pour que ça passe après des erreurs avec les quotes !!!

Exercice : VÉRIFIER AVEC DEVTOOLS QUE LES CLASSES CSS SONT BIEN ASSIGNÉES !

ngStyle ou ngClass ?

Réponse classique : **ngClass** si on doit modifier plusieurs propriétés CSS d'un coup :-)

Directives custom

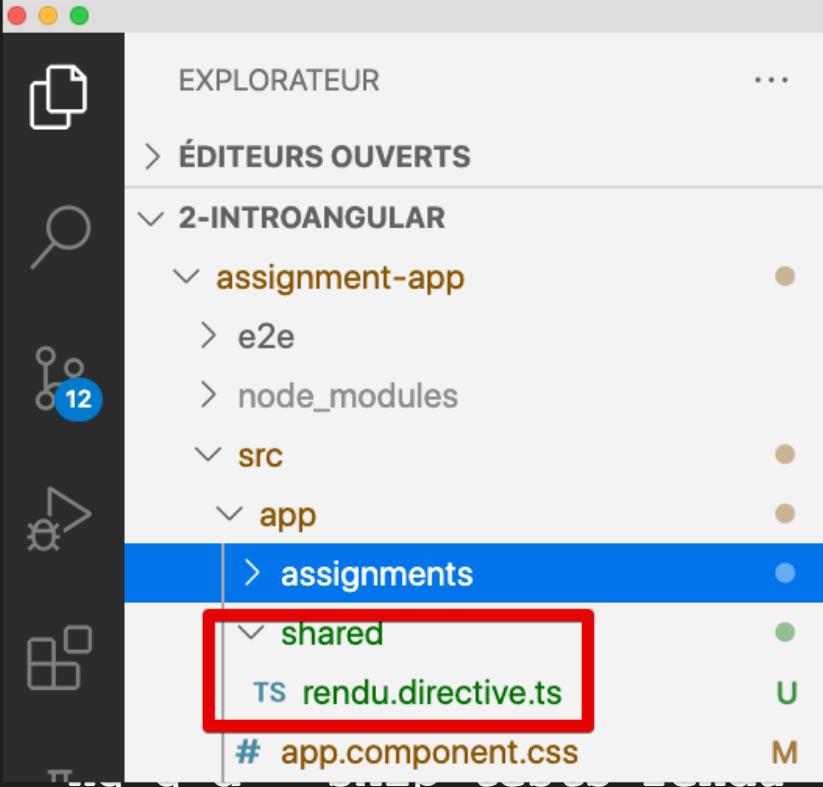
Il est possible de créer ses propres directives et simplement les ajouter au code HTML du template.

Pourquoi ne pas utiliser uniquement les directives standards ?

Très pratique quand on veut styler le même type d'éléments dans plusieurs composants différents.

Rend le code plus modulaire et maintenable.

Si on met à jour juste le code de la directive, tout le projet est impacté.



TS rendu.directive.ts X

<> assignments.component.html

assignment-app > src > app > shared > TS rendu.directive.ts > ...

```
1 import { Directive } from '@angular/core';
2
3 @Directive({
4   selector: '[appRendu]'
5 })
6 export class RenduDirective {
7
8   constructor() { }
9
10 }
```

On écrit un peu de code, notez le sélecteur !

The screenshot shows a code editor with three tabs at the top: 'rendu.directive.ts' (selected), 'assignments.component.html', and 'assignments.co'. Below the tabs, the file path is shown: 'assignment-app > src > app > shared > rendu.directive.ts'. A yellow icon next to the file name indicates it's a component.

```
1 import { Directive, ElementRef } from '@angular/core';
2
3 @Directive({
4   selector: '[appRendu]'
5 })
6 export class RenduDirective {
7   constructor(el: ElementRef) {
8     el.nativeElement.style.color = 'green';
9   }
10
11 }
12
```

The code is annotated with red boxes highlighting specific parts:

- A red box surrounds the import statement for `ElementRef`.
- A red box surrounds the entire constructor block, including the constructor parameter and the assignment to `el.nativeElement.style.color`.

Nouvelle version du template

```
<div *ngFor="let assignment of assignments">
  <p appRendu *ngIf="assignment.rendu; else nonsoumis">
    | Devoir intitulé {{assignment.nom}}, rendu le {{assignment.dateDeRendu}}.
  </p>
  <ng-template #nonsoumis>
    | Le devoir {{assignment.nom}} n'a pas été rendu.
  </ng-template>
</div>
```

Partie 4 - data binding et Formulaires

On va étudier...

Création de formulaire avec Angular Material

Binding de propriétés (MVC/MVVM)

Binding d'événements (relié)

La puissance du binding bi-directionnel

(puissant mais moins performant que l'approche React où il n'y a PAS de binding bi-directionnel)

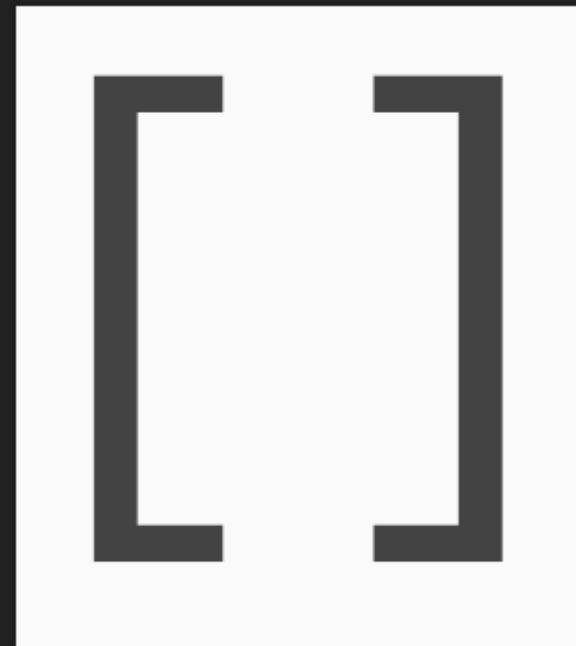
Binding d'attributs

On fera cela à l'aide des crochets []

Par exemple **[ngClass]**

Autre exemple avec un attribut HTML : **[disabled]**

Binding = évaluation...

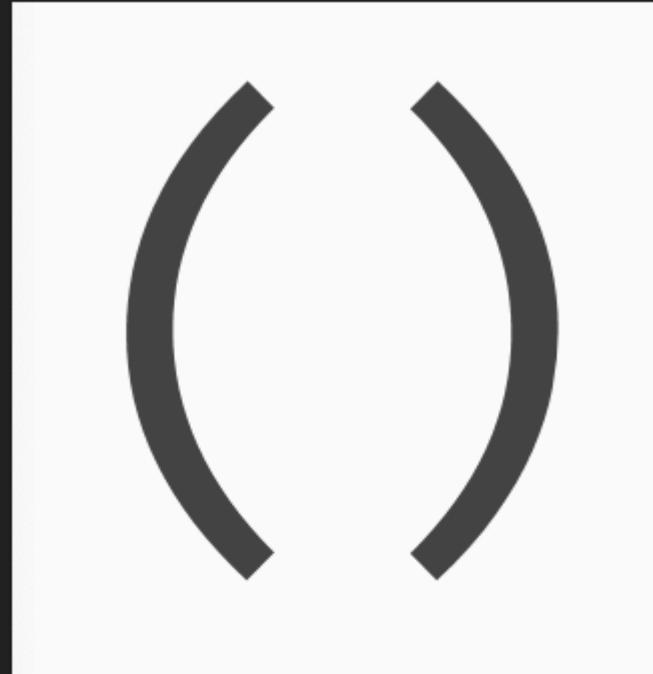


Binding d'évenements

On fera cela à l'aide des crochets parenthèses ()

Par exemple

- **(click)**
- **(keyup)**
- etc.



Binding bi-directionnel

On fera cela à l'aide des crochets et des parenthèses [()]

- Ex : variable affichée dans un input field,
- Si elle est affichée ailleurs, et bien si on modifie sa valeur en tapant dans le **champ** de saisie, on modifie la variable “bindée” mais aussi les autres vues...
- Typiquement : [**ngModel**]=**“assignment.nom”** ;



PIEGE PREMIER RAPPEL : T'AS PAS IMPORTÉ LE MODULE FORM !

Oui, oui, quand on utilise des formulaires, il faut importer le **FormsModule**

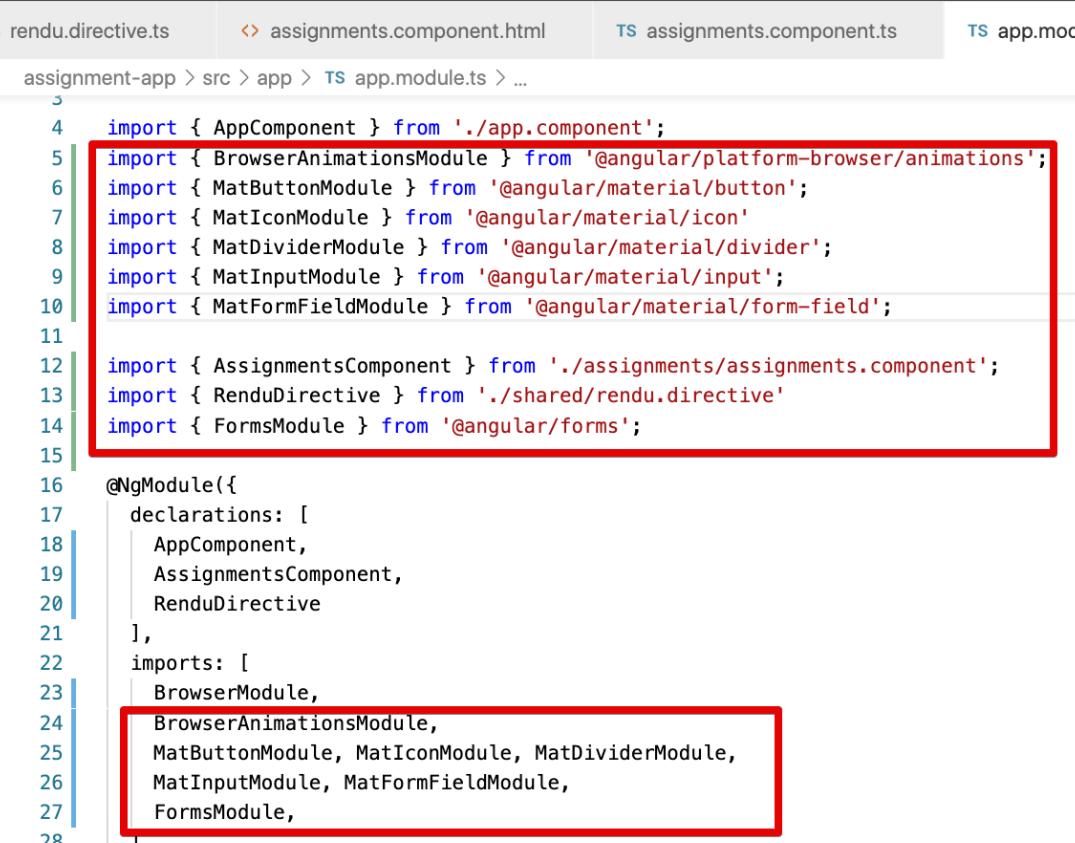
Sinon on a des erreurs de l'espace ! Déjà que....

Et si on fait un formulaire avec Angular Material, penser à importer les modules pour les éléments qu'on va utiliser !

Sinon on a des erreurs de l'espace ! Déjà que....

Donc, on va importer les modules nécessaires

Ca se passe dans app.module.ts



```
rendu.directive.ts assignments.component.html TS assignments.component.ts TS app.module.ts
assignment-app > src > app > TS app.module.ts > ...
4   import { AppComponent } from './app.component';
5   import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
6   import { MatButtonModule } from '@angular/material/button';
7   import { MatIconModule } from '@angular/material/icon';
8   import { MatDividerModule } from '@angular/material/divider';
9   import { MatInputModule } from '@angular/material/input';
10  import { MatFormFieldModule } from '@angular/material/form-field';
11
12 import { AssignmentsComponent } from './assignments/assignments.component';
13 import { RenduDirective } from './shared/rendu.directive'
14 import { FormsModule } from '@angular/forms';
15
16 @NgModule({
17   declarations: [
18     AppComponent,
19     AssignmentsComponent,
20     RenduDirective
21   ],
22   imports: [
23     BrowserModule,
24     BrowserAnimationsModule,
25     MatButtonModule, MatIconModule, MatDividerModule,
26     MatInputModule, MatFormFieldModule,
27     FormsModule,
28   ],
29 })
```

Comment connaître les modules nécessaires ?

Frequent-used modules dans la doc Angular

Pour angular Material, regarder n'importe quel exemple en ligne sur le site Angular Material, et regarder le fichier material-module.ts

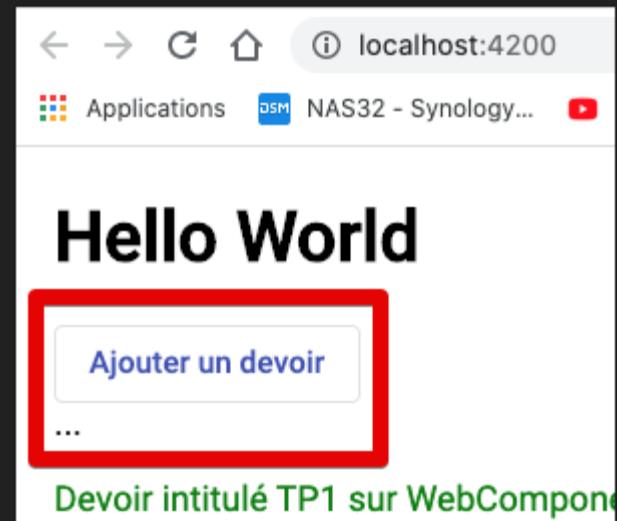
Par exemple, regardez celui-ci.

Bon, alors, ce formulaire ?

On va ajouter un formulaire d'ajout d'Assignment dans le template de `assignments.component.html`, il contiendra un seul bouton pour le moment :

```
<form ngForm #assignmentForm>
  <button mat-stroked-button
    color="primary">
    Ajouter un devoir
  </button>
</form>
<div *ngFor="let assignment of assignments">
  ...

```



Rendons le bouton “disabled” à la demande (binding)

On va l'afficher “disabled” et au bout de deux secondes il sera “enabled”.

On va :

1. Créer une variable **ajoutActive** qui sera false au début
2. On va par exemple ajouter un **setTimeOut** qui va l'activer au bout de deux secondes, par exemple dans **ngInit(...)**
3. On binde l'attribut HTML **disabled** sur la variable **ajoutActive**

Live coding !

Allez, on y va (et on teste !)

Dans assignments.component.ts

```
export class AssignmentsComponent implements OnInit {
    titre = "Mon application sur les Assignments !";
    ajoutActive = false;

    ngOnInit(): void {
        setTimeout(() => {
            this.ajoutActive = true;
        }, 2000);
    }
}
```

On gère l'événement (submit) sur le form

```
<form ngForm (submit)="onSubmit($event)" #assignmentForm>
```

```
<button  
    mat-stroked-button  
    color="primary"  
    [disabled]="!ajoutActive"  
>
```

Et dans assignments.component.ts :

```
onSubmit(event:any) {  
    console.log(event);  
    event.preventDefault();  
}
```

Ajout d'un input field pour le nom du devoir à ajouter

```
<form ngForm #assignmentForm class="form">
  <mat-form-field>
    <input matInput>
  </mat-form-field>
```

Mais comme ce n'est pas super beau (testez !)

...on va rajouter une classe CSS pour le formulaire `<form class="form" . . .>`

```
.form {
  display:flex;
  flex-direction: column;
  margin:5px;
}
```

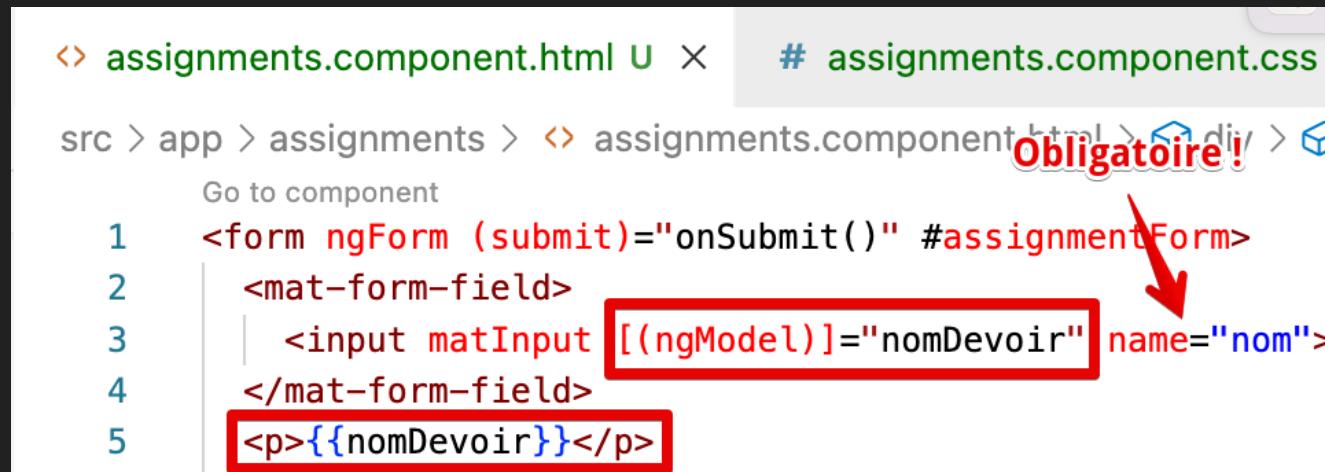
PAS BON ! On pourrait passer le nom lors de la soumission (template / logique)

```
<form ngForm #assignmentForm class="form">
    <mat-form-field>
        <input matInput #nomAssignment>
    </mat-form-field>

    <button
        mat-stroked-button
        color="primary">
        >
```

```
onSubmit(nom:string) {
    console.log(nom);
}
```

Autre solution (recommandée): utiliser le binding bi-directionnel



```
<form ngForm (submit)="onSubmit()" #assignmentForm>
  <mat-form-field>
    <input matInput [(ngModel)]="nomDevoir" name="nom">
  </mat-form-field>
<p>{{nomDevoir}}</p>
```

Ne pas oublier de déclarer la propriété nomDevoir dans le fichier TypeScript :

```
export class AssignmentsComponent implements OnInit {
  titre = "Mon application sur les Assignments !";
  ajoutActive = false;
  nomDevoir:string = "";
```

```
39  onSubmit() {
40  |   console.log(this.nomDevoir);
41 }
```

Bon, on le fait cet ajout ?

Bonne pratique : créer un “modèle” pour l’objet à ajouter...

A FAIRE : créer un fichier **src/app/assignments/assignment.model.ts**

The screenshot shows a code editor interface with the following details:

- EXPLORATEUR**: Shows the project structure:
 - > ÉDITEURS OUVERTS
 - ✓ ASSIGNMENT-APP
 - > .angular
 - > node_modules
 - ✓ src
 - ✓ app
 - ✓ assignments
 - TS assignment.model.ts U
 - # assignments.component.c... U
 - (assignments.component.h... U)
 - TS assignments.component.ts U
 - > shared- t.html U**
- TS assignment.model.ts U X**
- # assignments.component.ts U**

The current file, **assignment.model.ts**, contains the following TypeScript code:

```
1 export class Assignment {  
2   nom!:string;  
3   dateDeRendu!:Date;  
4   rendu!:boolean;  
5 }  
6 |
```

Pour le choix de la date, on va s'inspirer du code source du premier exemple du DatePicker de Angular Material

Exemple à regarder : <https://material.angular.io/components/datepicker/overview>

The screenshot shows the Angular Material Components documentation page. The left sidebar has a 'Datepicker' item selected. The main content area is titled 'OVERVIEW' and contains a brief description of the Datepicker component. Below the description is a code example for a 'Basic datepicker' showing both the HTML and TS code. At the bottom, there is a preview of the datepicker component with a label 'Choose a date' and a calendar icon.

```
<mat-form-field appearance="fill">
  <mat-label>Choose a date</mat-label>
  <input matInput [matDatepicker]="picker">
  <mat-hint>MM/DD/YYYY</mat-hint>
  <mat-datepicker-toggle matSuffix [for]="picker"></mat-datepicker-toggle>
  <mat-datepicker #picker></mat-datepicker>
</mat-form-field>
```

Copier coller cet exemple dans le formulaire

Ajouter les modules manquants dans app.module.ts (cf transparent suivant)

Plus

(assignments.component.html)

(assignments.component.css)

(assignments.component.ts)

src > app > assignments

Go to component

```
1  <form nglForm>
2    <mat-form-field>
3      <input matInput type="text" placeholder="Assignment name" required>
4    </mat-form-field>
5    <mat-form-field>
6      <input matInput type="date" placeholder="Due date" required>
7    </mat-form-field>
8
9    <button mat-raised-button type="submit" (click)="onSubmit()> Ajouter
```

assignment-app > src > app > app.module.ts > AppModule

```
8   import { MatDividerModule } from '@angular/material/divider';
9   import { MatInputModule } from '@angular/material/input';
10  import { MatFormFieldModule } from '@angular/material/form-field';
11  import { MatDatepickerModule } from '@angular/material/datepicker';
12  import { MatNativeDateModule } from '@angular/material/core';
13
14  import { AssignmentsComponent } from './assignments/assignments.component';
```

```
9
10 >
11 <mat-form-field>
12   <input matInput type="text" placeholder="Assignment name" required>
13 </mat-form-field>
14 <mat-form-field>
15   <input matInput type="date" placeholder="Due date" required>
16 </mat-form-field>
17 <button mat-raised-button type="submit" (click)="onSubmit()> Ajouter
18 >
19   <button mat-raised-button type="button" (click)="cancel()> Annuler
20 </button>
21 </form>
```

```
assignments:Assignment[] = [
{
  nom: "Devoir Angular à rendre",
  dateDeRendu: new Date('2022-10-10'),
  rendu: false
},
{
  nom: "Devoir JAVA à rendre",
  dateDeRendu: new Date('2022-09-10'),
  rendu: false
}];
```

Astuce pour remettre à zéro le formulaire

Dans le template :

```
<> assignments.component.html U X # assignments.component.css U  
src > app > assignments > <> assignments.component.html > form > mat  
Go to component  
1 <form ngForm (submit)="onSubmit(); assignmentForm.reset()"  
2 | | | #assignmentForm  
3 >
```

Partie 5 - Passage de données entre composants

On va voir maintenant...

Comment créer un composant fils

Passer des données entre le composant père et le composant fils avec `Input()`

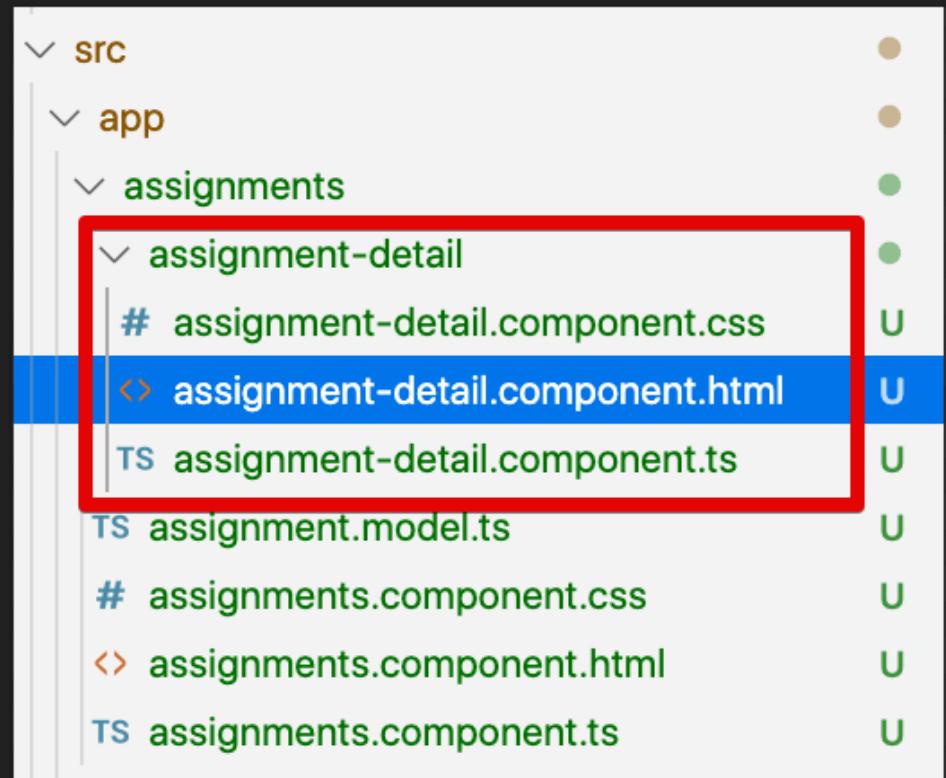
Passer des données du composant fils vers le composant parent avec `Output()`

Création d'un composant assignment-detail

C'est un composant fils du composant src/app/assignments

Pour le créer, dans le terminal, aller dans le dossier ci-dessus, puis exécuter la commande suivante :

```
ng g c --skip-tests  
assignment-detail
```



Hello World

Nom du devoir

Date de rendu

Ajouter un devoir

Devoir intitulé TP1 sur WebComponents, un lecteur audio amélioré, rendu le Fri Jan 17 2020 01:00:00 GMT+0100 (heure normale d'Europe centrale).

Le devoir TP2 sur Angular, un joli gestionnaire de devoirs (Assignments) n'a pas été rendu.

Le devoir TP3 sur Angular, utilisation du router et de Web Services n'a pas été rendu.

assignment-detail works!

DEVOIR WORLD

<m

Nom du devoir

Date de rendu

Pas très lisible...

Ajouter un devoir



Fri Jan 17 2020 01:00:00 GMT+0100 (heure normale d'Europe centrale)

<mat-divider>



Devoir intitulé TP1 sur WebComponents, un lecteur audio amélioré a été rendu.

Tue Dec 15 2020 01:00:00 GMT+0100 (heure normale d'Europe centrale)

</

Le devoir TP2 sur Angular, un joli gestionnaire de devoirs (Assignments) n'a pas été rendu.



Mon Jan 04 2021 01:00:00 GMT+0100 (heure normale d'Europe centrale)

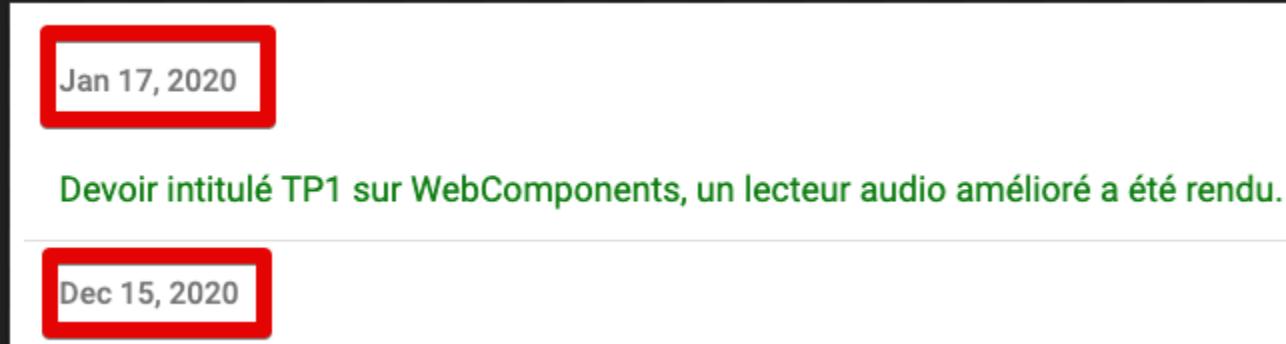
<mat-list-item>

Le devoir TP3 sur Angular, utilisation du router et de Web Services n'a pas été rendu.

Astuce pour formater la date

Utilisation du “pipe” dans le template, très pratique !

```
<h3 mat-subheader>{{assignment.dateDeRendu | date}}</h3>
```



Pour en savoir plus, voir [ce tutorial sur le format des dates et options...](#)

Rendre les assignments clickables

On va rendre le <mat-list-item> clickable et appeler une méthode qui prendra en paramètre l'assignment cliqué :

```
<mat-list-item (click)="assignmentClique(assignment)">
```

et dans le TypeScript du composant assignments :

- Une nouvelle propriété:

```
    assignmentSelectionne:Assignment;
```

- La méthode appelée quand on clique :

```
assignmentClique(assignment:Assignment) {  
    this.assignmentSelectionne = assignment;  
}
```

Mais comment passer l'info à assignment-detail

On a donc la propriété **assignmentSelectionne** qui vaut l'assignment cliqué

Et bien on va passer cette information au composant **assignment-detail** via le template (et via le binding avec [])

Mais avant on va déclarer une propriété récupérée en tant qu'attribut, dans le composant details (équivalent des props de VueJS et React)

Ajout d'une “prop” dans assignment-detail

The screenshot shows a code editor with three tabs at the top: "assignment-detail.component.ts", "assignments.component.ts", and "ass...". The "assignment-detail.component.ts" tab is active. The code is written in TypeScript and defines a component named AssignmentDetailComponent. The component has a selector of 'app-assignment-detail', a template URL of './assignment-detail.component.html', and a style URL of './assignment-detail.component.css'. It implements the OnInit interface and has an @Input() property named assignementTransmis of type Assignment. The code is numbered from 1 to 17. The @Input() line is highlighted with a red rectangle.

```
TS assignment-detail.component.ts × TS assignments.component.ts TS ass...
assignment-app > src > app > assignments > assignment-detail > TS assignment-detail.co
1   import { Component, Input, OnInit } from '@angular/core';
2   import { Assignment } from '../assignment.model';
3
4   @Component({
5     selector: 'app-assignment-detail',
6     templateUrl: './assignment-detail.component.html',
7     styleUrls: ['./assignment-detail.component.css']
8   })
9   export class AssignmentDetailComponent implements OnInit {
10     @Input() assignementTransmis:Assignment;
11
12     constructor() { }
13
14     ngOnInit(): void {
15     }
16   }
17
```

Passage de l'assignment depuis le template du composant père

The screenshot shows a code editor with three tabs: assignments.component.html, assignment-detail.component.ts, and assignments.component.ts. The assignment-detail.component.ts tab is active, displaying the following code:

```
12  </button>
13  </form>
14  <p>{{nomDevoir}}</p>
15  <p>{{dateRendu}}</p>
16
17  <mat-list *ngFor="let assignment of assignments">
18      <h3 mat-subheader>{{assignment.dateDeRendu | date}}</h3>
19      <mat-list-item (click)="assignmentClique(assignment)">
20          <p appRendu *ngIf="assignment.rendu; else nonsoumis">
21              Devoir intitulé {{assignment.nom}} a été rendu.
22          </p>
23          <ng-template #nonsoumis>
24              Le devoir {{assignment.nom}} n'a pas été rendu.
25          </ng-template>    @Input dans le fils
26          <mat-divider></mat-divider>
27      </mat-list-item>
28  </mat-list>
29  <app-assignment-detail [assignmentTransmis]="assignmentSelectionne">
30  </app-assignment-detail>
31
```

Annotations in red highlight specific parts of the code:

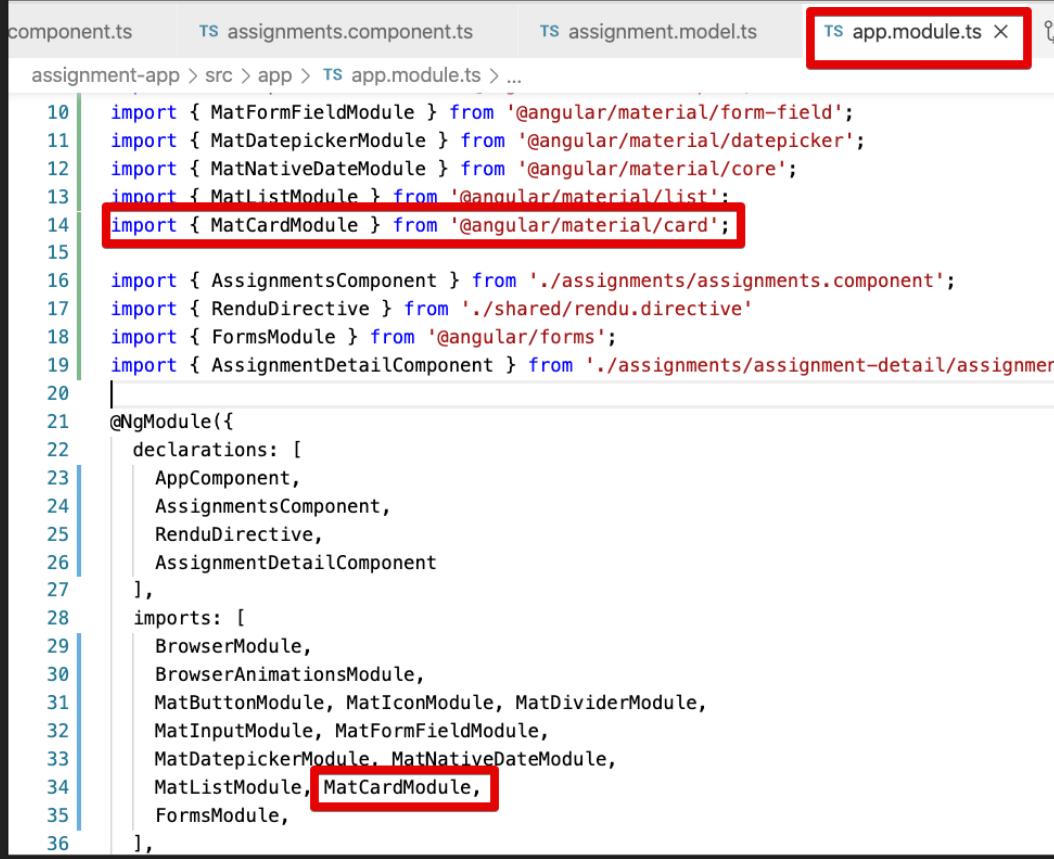
- A callout bubble points to the `(click)="assignmentClique(assignment)"` event handler with the text: "l'assignment cliqué (propriété dans composant père)".
- A red box surrounds the `[assignmentTransmis]="assignmentSelectionne"` binding at the bottom of the component.
- Two red arrows point from the text "l'assignment cliqué (propriété dans composant père)" to the `assignmentClique` method call in line 19 and to the `assignmentTransmis` binding in line 29.

Il ne reste plus qu'à afficher l'assignment transmis dans le template du composant detail

```
<p *ngIf="assignmentTransmis">{{ assignmentTransmis.nom }}</p>
```

Le *ngIf est là car sans lui lors du chargement initial de la page, assignmentTransmis est undefined et une erreur s'affichait dans la console.

Utilisation de MaterialCard pour un plus bel affichage



```
component.ts      TS assignments.component.ts    TS assignment.model.ts    TS app.module.ts × ⚡
assignment-app > src > app > TS app.module.ts > ...
10   import { MatFormFieldModule } from '@angular/material/form-field';
11   import { MatDatepickerModule } from '@angular/material/datepicker';
12   import { MatNativeDateModule } from '@angular/material/core';
13   import { MatListModule } from '@angular/material/list';
14   import { MatCardModule } from '@angular/material/card';
15
16   import { AssignmentsComponent } from './assignments/assignments.component';
17   import { RenduDirective } from './shared/rendu.directive'
18   import { FormsModule } from '@angular/forms';
19   import { AssignmentDetailComponent } from './assignments/assignment-detail/assignment
20
21 @NgModule({
22   declarations: [
23     AppComponent,
24     AssignmentsComponent,
25     RenduDirective,
26     AssignmentDetailComponent
27   ],
28   imports: [
29     BrowserModule,
30     BrowserAnimationsModule,
31     MatButtonModule, MatIconModule, MatDividerModule,
32     MatInputModule, MatFormFieldModule,
33     MatDatepickerModule, MatNativeDateModule,
34     MatListModule, MatCardModule,
35     FormsModule,
36   ],
37 }
```

Utilisation de <mat-card> dans le composant detail

assignment-detail.component.html X

TS assignment-detail.component.ts

TS assignments.component.ts

assignment-app > src > app > assignments > assignment-detail > assignment-detail.component.html > div.cc

```
1  <div class="container">
2    <mat-card *ngIf="assignmentTransmis">
3      <mat-card-title>{{assignmentTransmis.nom}}</mat-card-title>
4      <mat-card-subtitle>{{assignmentTransmis.dateDeRendu | date}}</mat-card-subtitle>
5    </mat-card>
6  </div>
```

TP1 sur WebComponents, un lecteur audio amélioré

Jan 17, 2020

Dec 15, 2020

Le devoir TP2 sur Angular, un joli gestionnaire de devoirs (Assignments) n'a pas été rendu.

Jan 4, 2021

Le devoir TP3 sur Angular, utilisation du router et de Web Services n'a pas été rendu.

TP2 sur Angular, un joli gestionnaire de devoirs (Assignments)

Dec 15, 2020

Devoir rendu

Implémentation de la logique de la checkbox

Le devoir TP2 sur Angular, un joli gestionnaire de devoirs (Assignments) n'a pas été rendu.

Jan 4, 2021

Le devoir TP3 sur Angular, utilisation du router et de Web Services n'a pas été rendu.

TP2 sur Angular, un joli gestionnaire de devoirs (Assignments)

Dec 15, 2020

Devoir rendu

**Si on clique la checkbox
disparait et le devoir apparaît
rendu en vert...**

Création d'un composant fils addAssignment

On va déplacer toute la logique d'ajout d'un assignment (template, etc.) dans un composant fils addAssignment

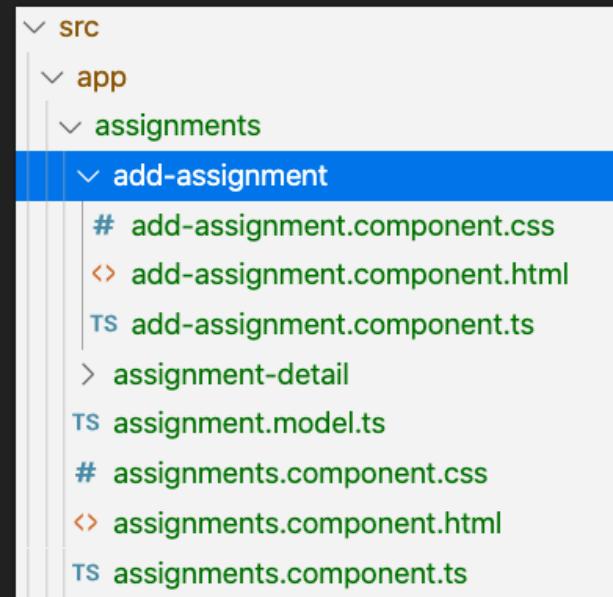
Mais le tableau des assignments et l'affichage de la liste des assignments restera à la charge du composant assignments

Comment faire ? Quels problèmes vont se poser ?

Première étape : créer le composant

Vous commencez à connaître la méthode. On se rend dans src/app/assignments et on utilise la commande :

```
ng g c --skip-tests add-assignment
```



Deuxième étape : déplacer le <form> du composant père dans le template du nouveau composant

add-assignment.component.html X

assignments.component.html

assignment-detail.component.html

assignment-app > src > app > assignments > add-assignment > add-assignment.component.html > ...

```
1  <form ngForm #assignmentForm class="form">
2    <mat-form-field>
3      <input matInput [(ngModel)]="nomDevoir" name="nom" placeholder="Nom du devoir">
4    </mat-form-field>
5    <mat-form-field>
6      <input matInput [matDatepicker]="picker" [(ngModel)]="dateRendu" name="date" placeholder="Date de rendu">
7      <mat-datepicker-toggle matSuffix [for]="picker"></mat-datepicker-toggle>
8      <mat-datepicker #picker></mat-datepicker>
9    </mat-form-field>
10   <button mat-stroked-button color="primary" [disabled]="!ajoutActive" (click)="onSubmit();">
11     Ajouter un devoir
12   </button>
13 </form>
```

Trois

un n

Hello World

... et au

Jan 17, 2020

..

Devoir intitulé TP1 sur WebComponents, un lecteur audio amélioré a été rendu.

Dec 15, 2020

Le devoir TP2 sur Angular, un joli gestionnaire de devoirs (Assignments) n'a pas été rendu.

Jan 4, 2021

Le devoir TP3 sur Angular, utilisation du router et de Web Services n'a pas été rendu.

Nom du devoir

Date de rendu



Ajouter un devoir

localhost:4200/?

Applications NAS32 - Synology... (23) Billy Idol - W... https://jariseon.git... Web Components... Getting Started wi... Research and w... michel-wams/mic... Music Technology... Autres favori

Hello World

Jan 17, 2020

Devoir intitulé TP1 sur WebComponents, un lecteur audio amélioré a été rendu.

Dec 15, 2020

Le devoir TP2 sur Angular, un joli gestionnaire de devoirs (Assignments) n'a pas été rendu.

Jan 4, 2021

Le devoir TP3 sur Angular, utilisation du router et de Web Services n'a pas été rendu.

Nom du devoir

Date de rendu

Ajouter un devoir

Elements Console Sources Network Performance »

✖ 1 | ⚙ : X

[WDS] Live Reloading enabled. client:52

✖ ► ERROR TypeError: Cannot read property 'push' of undefined core.js:4610
at AddAssignmentComponent.onSubmit (add-assignment.component.ts:23)
at AddAssignmentComponent_Template_button_click_9_listener (add-assignment.component.html:10)
at executeListenerWithErrorHandling (core.js:15610)
at wrapListenerIn_markDirtyAndPreventDefault (core.js:15645)

On a bien un problème...

En fait on ne peut accéder dans le composant fils au tableau des assignments qui est dans le composant père. Comment faire ?

Avant de faire ça on va un peu arranger l'interface graphique en ajoutant un bouton “Ajouter Devoir” en haut de la fenêtre:

- Par défaut on voit la liste des assignments et le détail de l'un d'eux
- Si on clique le nouveau bouton on verra à la place le formulaire d'ajout, et plus la liste...
- Faisons cela avant de revenir sur le problème de l'accès au tableau des assignments...

Dans le template du père...

```
<button mat-flat-button color="accent"  
       (click)="onAddAssignmentBtnClick()">  
  > Ajouter Assignment  
</button>  
  
<mat-list *ngFor="let assignment of assignments">  
  ...
```

Dans le fichier TypeScript du père

Une nouvelle propriété :

```
formVisible = false;
```

Un nouvelle méthode :

```
onAddAssignmentBtnClick() {
    this.formVisible = true;
}
```

Du code caché

On wrappe dans un `<div>`, et pour régler

```
assignment-app > src > app > assignments > assignments.component.html > ...
1  <main *ngIf="!formVisible">
2    <button mat-flat-button color="accent" (click)="onAddAssignmentBtnClick()">
3      Ajouter Assignment
4    </button>
5    <mat-list *ngFor="let assignment of assignments">
6      <h3 mat-subheader>{{assignment.dateDeRendu | date}}</h3>
7      <mat-list-item (click)="assignmentClique(assignment)">
8        <p appRendu *ngIf="assignment.rendu; else nonsoumis">
9          Devoir intitulé {{assignment.nom}} a été rendu.
10         </p>
11         <ng-template #nonsoumis>
12           Le devoir {{assignment.nom}} n'a pas été rendu.
13         </ng-template>
14         <mat-divider></mat-divider>
15       </mat-list-item>
16     </mat-list>
17     <app-assignment-detail [assignmentTransmis]="assignmentSelectionnée">
18     </app-assignment-detail>
19   </main>
20
21  <app-add-assignment *ngIf="formVisible"></app-add-assignment>
22
```

Pour communiquer avec le parent, utiliser Output()

Principe : le fils (fille) envoie un nouvel Assignment

L'événement sera traité

Le père écoute et ajoute à la liste,

Puis on modifiera la liste des assignments

```
component.ts          TS assignments.component.ts      TS add-assignment.component.ts ×
app > assignments > add-assignment > TS add-assignment.component.ts > AddAssignmentComponent
1 import { Component, EventEmiter, OnInit, Output } from '@angular/core';
2 import {Assignment} from '../assignment.model';
3
4 @Component({
5   selector: 'app-add-assignment',
6   templateUrl: './add-assignment.component.html',
7   styleUrls: ['./add-assignment.component.css']
8 })
9 export class AddAssignmentComponent implements OnInit {
10   @Output() nouvelAssignment = new EventEmiter<Assignment>();
11
12   nomDevoir:string = "";
13   dateRendu:Date;
14
15   constructor() {}
16   ngOnInit(): void {
17   }
18
19   onSubmit() {
20     const newAssignment = new Assignment();
21     newAssignment.nom = this.nomDevoir;
22     newAssignment.dateDeRendu = this.dateRendu;
23     newAssignment.rendu = false;
24
25     //this.assignments.push(newAssignment);
26     this.nouvelAssignment.emit(newAssignment);
27   }
28 }
```

à l'application de

'ai ajouté un

et le rajoutera

ouveau montrer

Dans le composant père on écoute cet événement

Dans le template on va écouter ce nouvel événement custom que peut émettre le composant add-assignment :

The diagram illustrates the communication flow between a child component and its parent template. It shows three main components: a child component (app-add-assignment) that emits a custom event, the custom event itself, and a parent method that listens for and handles the event.

Composant fils émetteur de l'événement "nouvelAssignment"

Événement custom émis par le fils

Méthode du père qui récupère les données émises

L'assignment ajouté, transmis par le fils

```
<app-add-assignment (nouvelAssignment)="onNouvelAssignment($event)" *ngIf="formVisible"></app-add-assignment>
```

Puis on va écrire la méthode qui va faire l'ajout. En général les écouteurs, en Angular, commencent par “on” suivi du nom de l'événement :

```
onNouvelAssignment(event:Assignment) {  
  this.assignments.push(event);  
  this.formVisible = false;  
}
```

On a presque fini, on fait un peu le ménage...

Virer ce “Hello World” : dans `app.component.html`

Afficher le bouton d'ajout sur la droite (et pas en bas) (ici [tuto CSS Flex](#) -à

The screenshot shows a web browser window with the URL `localhost:4200` in the address bar. The page displays a list of tasks:

- Date de rendu Nov 17, 2020
- TP WebComponents INTENSE rendu.** (highlighted with a green dashed border)
- Date de rendu Dec 3, 2020

A red arrow points from the text "Bouton sur la droite" to a pink button labeled "Ajouter un devoir" located on the right side of the page.

Conclusion partielle

Qu'a-t-on appris ?

- Comment créer des composants et lier des données à des vues
- Utiliser et créer des directives
- Styler des éléments sous condition
- Travailler avec un formulaire (il reste encore à apprendre)
- Passer les données entre composants (idem)
- Debugguer Angular c'est dur :
 - Pas d'extension de navigateur
 - Le compilateur à la volée casse souvent
 - Savoir quel module il faut importer et d'où ce n'est pas évident
- On peut néanmoins faire des choses puissantes :
 - MVVM les pipes {{assignment.dateDeRendu | date}}, Angular très complet, etc.

Exercice à faire : dans le composant détail, ajouter une bouton **DELETE** pour supprimer un assignment

En utilisant le transfert d'information fils -> père que nous venons de voir !

A vous de jouer !!!

Partie 6 - utilisation de “services”

Nous allons étudier...

- Création d'un premier service
- Déplacer les données qu'on manipule dans ce service et les consommer dans des composants
- Comprendre le mécanisme des "Observables"
- Créer des méthodes pour travailler avec des Observables
- Injecter des services dans d'autres services

Apprendre comment marchent les services, création d'un premier service

Services ?

Pourquoi a-t-on besoin de cette notion de “service” (qui n'existe pas en React et VueJS) ?

Comment fonctionnent les services.

Création d'un premier service.

Pourquoi des “services”

L'idée proposée par les créateurs d'Angular est qu'on ne doit pas gérer les données dans les composants.

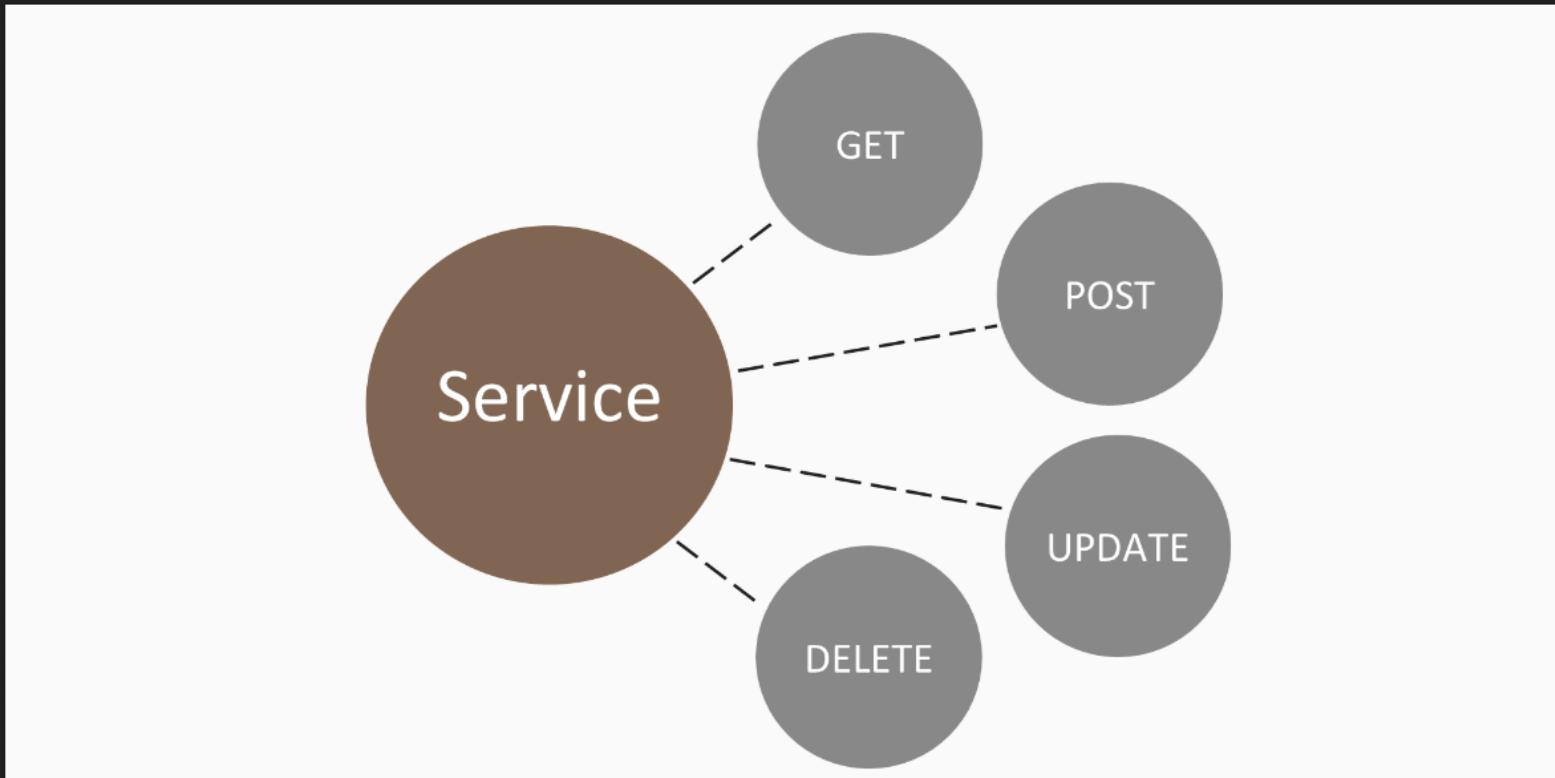
Les composants doivent avant tout “présenter” les données.

Les services sont là pour fournir les données aux composants de l'application.

On peut avoir plusieurs services spécialisés dans tel ou tel type de données (sources différentes, types différents etc.)

On peut même avoir des services qui utilisent d'autres services.

Services inspirés de REST...



Création d'un service

Comme pour les composants, on va utiliser CLI

Naviguer dans le dossier de l'application et dans un terminal, taper :

```
ng g s <nom-du-service>
```

Et si on ne veut pas de fichier de test unitaires :

```
ng g s --skip-tests=true <nom-du-service>
```

Ajout d'un service “assignments” dans le folder “shared” de notre application

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

- EXPLORATEUR**: Shows the project structure:
 - Assignment-App
 - e2e
 - node_modules
 - src
 - app
 - assignments
 - shared
 - assignments.service.ts
 - rendu.directive.ts
 - app.component.css
 - app.component.html
- ÉDITEURS OUVERTS**: The file `assignments.service.ts` is open in the editor.
- TERMINAL**: The terminal shows the command being run to generate the service file.

```
(base) mbuffa2@buffy assignment-app % cd src/app/shared  
(base) mbuffa2@buffy shared % ng g s --skipTests=true assignments  
CREATE src/app/shared/assignments.service.ts (140 bytes)  
(base) mbuffa2@buffy shared %
```

Pas la peine d'ajouter le service créé dans la section “providers” du fichier app.module.ts (vieilles versions)

Car le service est déclaré avec
`providedIn : 'root'`

Voir doc sur les services.

```
@Injectable({
  // permet d'éviter de l'ajouter dans les modules....
  providedIn: 'root'
})
```

Déplacer les données dans le service et
l'injecter dans le composant qui va
l'utiliser

Rappel sur “Dependency Injection” (CDI en Java)

Dependency Injection



assignments-service.ts

```
@Injectable({  
    providedIn: 'root'  
})  
export class AssignmentsService {
```

assignments.component.ts

```
constructor(private assignmentsService: AssignmentsService) {}  
  
ngOnInit() {  
    // WRONG  
    const assignmentSvc = new AssignmentsService();  
  
    // CORRECT  
    this.assignmentsService.getAssignments();  
}
```

TS assignments.service.ts

TS app.module.ts

TS assignments.component.ts

src > app > shared > TS assignments.service.ts > AssignmentsService

```
1 import { Injectable } from '@angular/core';
2 import {Assignment} from '../assignments/assignment.model';
3
4 @Injectable({
5   providedIn: 'root'
6 })
7 export class AssignmentsService {
8   assignments:Assignment[] = [
9     {
10       nom:"TP1 sur WebComponents, un lecteur audio amélioré",
11       dateDeRendu: new Date('2020-01-17'),
12       rendu : true
13     },
14     {
15       nom:"TP2 sur Angular, un joli gestionnaire de devoirs",
16       dateDeRendu: new Date('2020-12-15'),
17       rendu : false
18     },
19     {
20       nom:"TP3 sur Angular, utilisation du router et de Web Services"
21       dateDeRendu: new Date('2021-01-04'),
22       rendu : false
23     }
24   ];
25 }
```

d'un getter dans

TS assignments.service.ts

TS app.module.ts

TS

src > app > shared > TS assignments.service.ts > AssignmentsService

```
25
26   constructor() {}
27
28   getAssignments():Assignment[] {
29     return this.assignments;
30   }
31
32 }
```

Injection du service dans le le composant assignments

```
TS assignments.component.ts X  TS app.module.ts  TS rendu.directive.ts ?  
app > assignments > TS assignments.component.ts > AssignmentsComponent > ass  
1 import { Component, OnInit } from '@angular/core';  
2 import { AssignmentsService } from '../shared/assignments.service';  
3 import {Assignment} from './assignment.model';  
4  
5 @Component({  
6   selector: 'app-assignments',  
7   template: '

Mon application sur les Assignments !

',  
8   styleUrls: ['./assignments.component.css']  
9 })  
10 export class AssignmentsComponent implements OnInit {  
11   titre = "Mon application sur les Assignments !";  
12   formVisible = false;  
13   assignmentSelectionne:Assignment;  
14   assignments: Assignment[];  
15  
16   constructor(private assignmentService:AssignmentsService) { }  
17  
18   ngOnInit(): void {  
19     this.assignments = this.assignmentService.getAssignments();  
20   }  
21 }
```

1 - Injection du service

2 - Utilisation du service

A red arrow points from the 'constructor' line in the component code down to the 'getAssignments()' call in the 'ngOnInit()' method. Another red arrow points from the 'private assignmentService' parameter in the constructor to the 'this.assignmentService' reference in the 'getAssignments()' call.

Puis vérifier que l'application fonctionne comme avant !

Utilisation d'Observables

Ca ne vous rappelle rien ? Observer/Observable en Java, le MVC ?

On va faire un bref rappel de ce que sont des objets “Observables”,

Comment les déclarer,

Comment les manipuler...

Observable en Angular ?

Un Observable va permettre de manipuler des données asynchrones.

Un Observable va permettre de s'abonner (subscribe) à des données asynchrones.

- Il propose un flux de données qui seront “publiées” et on pourra s'y “abonner”. Oui, c'est la pattern publish/subscribe !
- Plusieurs données pourront être émises au cours du temps
- Un Observable a des opérateurs comme `.map()`, `.filter()` et `forEach()`

Les Observables font partie d'un module standard Angular appelé RxJS

Mais... on a déjà les promesses en JavaScript !

- Observables :
 - Un flux de données qui reste ouvert,
 - Un Observable peut être “annulé” (cancelled),
 - Fournit des opérateurs (map, forEach...)
 - Mécanisme assez lourd (à la JavaEE, patterns, encore des patterns)
 - Au coeur de modules comme HttpClient qui sert aux requêtes REST.
- Promesses :
 - Se résolvent une fois que les données sont reçues,
 - Ne peuvent être annulées,
 - Pas d'opérateur
 - Mécanisme léger, à la JS (simples à utiliser avec async/await/Promise.All etc.)

Ajout d'un Observable dans le service assignments

On va importer la classe **Observable** de rxjs

Au lieu de retourner le tableau des assignments, on va retourner un **Observable**.

Cela se fait au travers du package **of** (operator function) proposé par le module

```
TS assignments.service.ts ● TS assignments.component.ts TS app.module.ts
src > app > shared > TS assignments.service.ts > 🚀 AssignmentsService
1 import { Injectable } from '@angular/core';
2 import { Observable, of } from 'rxjs';
3 import {Assignment} from '../assignments/assignment.model';
4

getAssignments():Observable<Assignment[]> {
| return of(this.assignments);
}
    ↘ transforme le tableau en Observable
```

On ne va plus manipuler les assignments de la même manière dans le composant assignments

```
export class AssignmentsComponent implements OnInit {
  titre = "Mon application sur les Assignments !";
  formVisible = false;
  assignmentSelectionne: Assignment;
  assignments: Assignment[];

  constructor(private assignmentService: AssignmentsService) { }

  ngOnInit(): void {
    //this.assignments = this.assignmentService.getAssignments();
    this.getAssignments(); Renvoie un Observable
  }

  getAssignments() {
    this.assignmentService.getAssignments()
      .subscribe(assignments => this.assignments = assignments);
  }
}
```



Puis vérifier que l'application fonctionne comme avant !

Ajout de méthodes ADD et DELETE
dans le service

Ajout d'une méthode addAssignment

Rappel : on va renvoyer un Observable (pour cette fois, juste une chaîne de caractère qui indiquera que l'ajout a été effectué)

```
getAssignments():Observable<Assignment[]> {
    return of(this.assignments);
}

addAssignment(assignment: Assignment): Observable<string> {
    this.assignments.push(assignment);
    return of('Assignment ajouté');
}
```

Et modification du composant assignments

```
onNouvelAssignment(event:Assignment) {  
    // this.assignments.push(event);  
    this.assignmentService.addAssignment(event)  
        .subscribe(message => console.log(message));  
  
    this.formVisible = false;  
}
```

Puis vérifier que
l'application fonctionne
comme avant !

Ajoute d'une méthode UPDATE

```
updateAssignment(assignment:Assignment):Observable<string> {
    // ici envoyer requête PUT à une base de données...
    // En fait on a besoin de rien faire de spécial si on travaille avec
    // un tableau car le paramètre passé EST UN ELEMENT DU TABLEAU
    // et si on l'a modifié dans le composant details, par exemple
    // en mettant sa propriété rendu à true, et bien, cela
    // l'a aussi modifié dans le tableau
    return of("Assignment service: assignment modifié !")
}
```

Exercice

Modifier le composant “detail” :

1. Pour que la checkbox de mise à jour mette à jour l’assignment en utilisant la méthode du slide précédent,
2. Pour que le bouton “delete” que vous aviez ajouté en tant qu’exercice dans une leçon précédente, permette de supprimer l’assignment.

Correction pour l'update...voici le composant “détail” modifié

```
src > app > assignments > assignment-detail > TS assignment-detail.component.ts > ...
1   import { Component, Input, OnInit } from '@angular/core';
2   import { Assignment } from '../assignment.model';
3   import { AssignmentsService} from '../../../../../shared/assignments.service';
4
5   @Component({
6     selector: 'app-assignment-detail',
7     templateUrl: './assignment-detail.component.html',
8     styleUrls: ['./assignment-detail.component.css']
9   })
10  export class AssignmentDetailComponent implements OnInit {
11    @Input() assignementTransmis:Assignment;
12
13    constructor(private assignmentsService: AssignmentsService) { }
14
15    ngOnInit(): void {
16    }
17
18    onAssignmentRendu() {
19      this.assignementTransmis.rendu = true;
20
21      this.assignmentsService.updateAssignment(this.assignementTransmis
22        .subscribe(message => console.log(message));
23    }
24 }
```

Correction pour le DELETE

Partie 1 ajout d'un bouton DELETE + CSS dans le composant detail

```
<div id="bottom">
  <mat-checkbox *ngIf="!assignmentTransmis.rendu" (click)="onAss
    Devoir rendu
  </mat-checkbox>
```

TP2 sur Angular, un joli gestionnaire de devoirs (Assignments)

Dec 15, 2020

Devoir rendu

DELETE

TP1 sur WebComponents, un lecteur audio amélioré

Jan 17, 2020

DELETE

Correction pour le DELETE

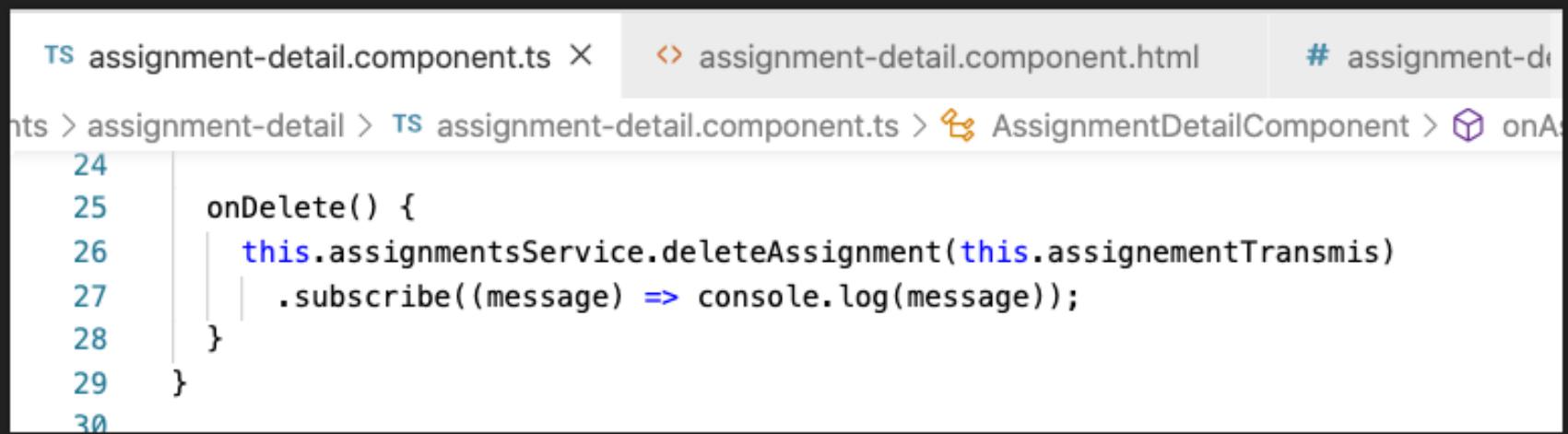
Dans le service :

```
deleteAssignment(assignment:Assignment):Observable<string> {
    let pos = this.assignments.indexOf(assignment);
    this.assignments.splice(pos, 1);

    return of("Assignment service: assignment supprimé !")
}
```

Correction pour le DELETE

Dans le composant détail :



The screenshot shows a code editor with three tabs: 'assignment-detail.component.ts', 'assignment-detail.component.html', and '# assignment-di...'. The 'assignment-detail.component.ts' tab is active, displaying the following TypeScript code:

```
24
25     onDelete() {
26         this.assignmentsService.deleteAssignment(this.assignmentTransmis)
27             .subscribe((message) => console.log(message));
28     }
29 }
30 }
```

The code implements an 'onDelete' method that calls the 'deleteAssignment' method from the 'assignmentsService'. The service's response is subscribed to and logged to the console.

Soucis : la carte avec le détail reste affichée...

Il faut mettre l'assignmentTransmis à null ou undefined pour que la carte n'affiche plus le détail.

```
TS assignment-detail.component.ts X assignment-detail.component.html # assignment-  
src > app > assignments > assignment-detail > TS assignment-detail.component.ts > ...  
24  
25     onDelete() {  
26         this.assignmentsService.deleteAssignment(this.assignmentTransmis)  
27             .subscribe((message) => console.log(message))  
28     }  
29     this.assignmentTransmis = null;  
30 }
```

vous avez compris pourquoi?

NOTE : si jamais vous êtes en mode “strict”, deux possibilités :

1. ajouter : `if (!this.assignmentTransmis) return;` à la première ligne dans la fonction!
2. Ou bien désactiver le mode strict: éditer `tsconfig.json` et changer `strict:true`, en `strict:false`

Injecter un service dans un autre service

Exemple avec un “logging service”

```
cd src/app/shared
```

```
ng g s --skip-tests=true logging
```

The screenshot shows the Angular CLI interface for generating a new service. The command entered is `ng g s --skip-tests=true logging`. The generated file, `logging.service.ts`, is displayed in the editor. The code defines a `LoggingService` class with a constructor.

```
EXPLORATEUR ... TS assignment-detail.component.ts TS logging.service.ts X  
ÉDITEURS OUVERTS 1 NON ENREGISTRÉ(S)  
ASSIGNMENT-APP  
assignments.component.ts U  
shared  
assignments.service.ts U  
logging.service.ts U  
rendu.directive.ts U  
app.component.css M  
app.component.html M  
src > app > shared > TS logging.service.ts > ...  
1 import { Injectable } from '@angular/core';  
2  
3 @Injectable({  
4   providedIn: 'root'  
5 })  
6 export class LoggingService {  
7   constructor() { }  
8 }  
9  
10
```

Ajout d'une méthode log(nomAssignment, action)

On ajoute une méthode qui va par exemple afficher dans la console :

assignment **TP1 angular ajouté**, assignment **TP2 React supprimé** etc.

```
log(assignmentName, action) {  
  console.log("Assignment " + assignmentName + " " + action);  
}
```

Ce service va être utilisé par les autres services dans src/app/shared, il va être un “sous-service” local, pas besoin de le déclarer dans les app.module.ts, il suffira de l’importer...

Injection de ce service dans le service assignments

```
TS assignments.service.ts X TS assignment-detail.component.ts TS logging.service.ts
src > app > shared > TS assignments.service.ts > 🛡 AssignmentsService > ⚙ deleteAssignment
-->
29   constructor private loggingService:LoggingService) }
30
31   getAssignments():Observable<Assignment[]> {
32     return of(this.assignments);
33   }
34
35   addAssignment(assignment: Assignment): Observable<string> {
36     this.assignments.push(assignment);
37     this.loggingService.log(assignment.nom, "ajouté");
38
39     return of('Assignment ajouté');
40   }
41
```

Qu'a-t-on appris ?

A utiliser la notion de service pour exposer des données à des composants de manière centralisée

A les injecter dans les composants (à propos, combien d'instances sont créés ?)

La notion d'Observable pour des données asynchrones (publish/subscribe)

Utiliser des observables

A implémenter l'équivalent d'un service RESTFUL/CRUD

A créer un service local et à l'injecter dans un autre service (logging)

Partie 7 - Utilisation d'un routeur

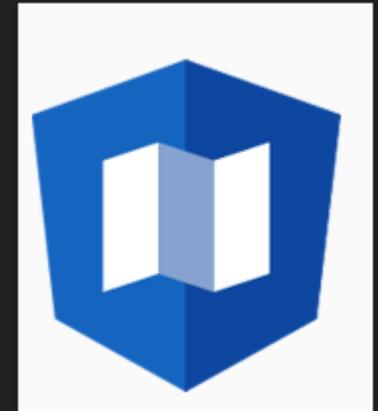
Ce que l'on va voir

- La notion de “routes” en Angular
- Initialisation des routes
- Naviguer dynamiquement dans l’application
- “Queries” et routes
- Protéger des routes

Les bases des “routes”

Routeur Angular

- Angular vient avec un module “routeur” en standard (contrairement à React et Vue)
- Le routeur Angular reprend le modèle de navigation du navigateur Web
 - On entre un URL pour naviguer vers une “page”.
 - Ou bien on clique sur un lien dans une page pour naviguer vers une autre page.
 - On peut utiliser l'historique de navigation et les boutons en forme de flèche du navigateur pour revenir à la page précédente ou aller à la page qu'on a déjà consultée.
- Rappel on a pas de vraies “pages” car les applications Angular sont des Single-Page WebApps
 - Mais on imaginera une page comme un “état affiché” à un instant donné, et associé à un URL.



Pourquoi a-t-on besoin des routes ?

Car les URLs sont utiles !

Ils peuvent être partagés, via mail, chat etc. et permettent d'aller directement à une page donnée (ex: lien vers un produit Amazon, une vidéo YouTube etc.)

Sans routeur et sans routes notre application sur les Assignments a un seul URL :
<http://localhost:4200> !

- On voudrait peut-être avoir un URL unique quand un assignment est affiché en détails ?

Ceci n'est pas pratique. Dans de vraies applications, 99% du temps on aura besoin d'un routeur.

On va modifier notre application !!!

On va donner un titre à l'application et on va lui donner un URL nommé !

On modifie le composant “root” app.component.ts et son template

The image shows a code editor with two tabs open:

- app.component.ts**:
A TypeScript file containing the definition of the root component. It imports the Component decorator from '@angular/core'. The component is defined with a selector 'app-root', a template URL pointing to 'app.component.html', and a style URL pointing to 'app.component.css'. The title is set to 'Application de g...'.

```
src > app > TS 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 = 'Application de g...
10 }
```
- app.component.html**:
An HTML template file. It contains an h1 element with a nested nav element. The nav element has an a tag with a href attribute set to an empty string (""). The routerLink attribute is set to "/home".

```
src > app > <> app.component.html > ...
1 <!-- NE PAS UTILISER A HREF !
2 <h1><nav><a href="">{{title}}</a></nav></h1>
3 -->
4
5 <h1>
6   <nav><a routerLink="/home">{{title}}</a></nav>
7 </h1>
8
9 <app-assignments></app-assignments>
10 
```

On teste l'application

Sans rien faire d'autre l'application fonctionne mais l'attribut routerLink ne fonctionne pas.

C'est normal, par défaut les navigateurs ignorent simplement les tags html inconnus et les attributs inconnus.



Initialisation des routes : ça app.module.ts !

```
TS app.module.ts X # assignment-detail.component.css
src > app > TS app.module.ts > AppModule
.
.
.
22 import {AssignmentsService} from './shared/as
23 import { Routes } from '@angular/router';
24
25 const routes : Routes = [
26   // home page, ce qui sera affiché avec http
27   // ou http://localhost:4200/
28   {path: '', component:AssignmentsComponent},
29   // ou http://localhost:4200/home
30   {path: 'home', component:AssignmentsComponent}
31
32 ];
33
```

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

TS app.module.ts X # assignment-detail.component.css

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

```
52 ;
53
54 @NgModule({
55   declarations: [...,
56   ],
57   imports: [
58     BrowserModule,
59     BrowserAnimationsModule,
60     MatButtonModule, MatIconModule, MatDividerModule,
61     MatInputModule, MatFormFieldModule,
62     MatDatepickerModule, MatNativeDateModule,
63     MatListModule, MatCardModule, MatCheckboxModule,
64     FormsModule,
65     RouterModule.forRoot(routes)
66   ],
67   providers: [AssignmentsService]
68 })
```

TS as

Le routeur est ok, les routes aussi,
Il reste à déclarer <router-outlet> dans le
template du composant root

localhost:4200/home

Applications Repl.it - Entirevor... VerbalExpressions... Badassebikes Box... Linked Open Voca... The Linked Open... Search Datasets ... JSPatcher Autres favoris

Application de gestion des devoirs à rendre (Assignments)

Ajouter Assignment

1 Jan 17, 2020 Cliquez ici ! /home doit apparaître à la fin de l'URL

2 Devoir intitulé TP1 sur WebComponents, un lecteur audio amélioré a été rendu.

3 Dec 15, 2020

4 Le devoir TP2 sur Angular, un joli gestionnaire de devoirs (Assignments) n'a pas été rendu.

5 Jan 4, 2021

6 Le devoir TP3 sur Angular, utilisation du router et de Web Services n'a pas été rendu.

Ajout d'une route /add pour l'ajout d'un assignment

Ajout de la route dans app.module.ts

```
const routes : Routes = [
    // home page, ce qui sera affiché avec http://localhost:4200
    // ou http://localhost:4200/
    {path: '', component:AssignmentsComponent},
    // ou http://localhost:4200/home
    {path:'home', component:AssignmentsComponent},
    {path:'add', component:AddAssignmentComponent}
];
```

Modification du template de app.assignments.html

On n'a plus besoin des
***ngIf="formVisible"** puisque
c'est le **<router-outlet>** qui
affichera tel ou tel composant. Plus
besoin de cacher des parties de la
page...

On a va ajouter un attribut
routerLink au bouton d'ajout

Et on teste que cela fonctionne
encore au niveau de la
navigation...

```
ss  <--<main class="container" *ngIf="!formVisible">-->
src > app > assignments > assignments.component.html > main.container
  1  <!--<main class="container" *ngIf="!formVisible">-->
  2  <main class="container" >
  3  <!--
  4    <button class="ajouterBtn"
  5      mat-flat-button color="accent"
  6      (click)="onAddAssignmentBtnClick()>
  7  -->
  8  <div class="ajouterBtn">
  9    <a routerLink="/add">
 10      <button class="ajouterBtn"
 11        mat-flat-button color="accent"
 12        (click)="onAddAssignmentBtnClick()>
 13        Ajouter Assignment
 14      </button>
 15    </a>
 16  </div>
 17  <mat-list *ngFor="let assignment of assignments">
```

inutile

pour préserver le CSS
Quand on cliquera le
bouton on naviguera

Est-ce encore
utile?

On va localiser l'ajout dans `add-assignment.component`

On supprime l'utilisation du service dans `app.assignments`

The screenshot shows a code editor with two tabs open:

- assignments.component.ts**: This tab contains the component's logic. It includes methods for handling button clicks and adding new assignments. The assignment service is used to add assignments.
- assignments.component.html**: This tab contains the component's template. It uses an `<app-add-assignment>` element to add new assignments.

In the `assignments.component.html` template, the `[assignmentTransmis]` attribute is highlighted in red, indicating it is being used.

```
src > app > assignments > TS assignments.component.ts <-- assignments.component.html
29     this.assignmentSelectionne = assignmenss
30   }
31
32   onAddAssignmentBtnClick() {
33     //this.formVisible = true;
34   }
35   /*
36   onNouvelAssignment(event:Assignment) {
37     // this.assignments.push(event);
38     this.assignmentService.addAssignment(ev
39       .subscribe(message => console.log(mes
40
41     this.formVisible = false;
42   }
43   */
44 }
```

```
src > app > assignments > <-- assignments.component.html > ...
26   |   <mat-divider></mat-divider>
27   |   </mat-list-item>
28   |   </mat-list>
29   |   <app-assignment-detail [assignmentTransmis]="assignmentSelectionne">
30   |   </app-assignment-detail>
31   |   </main>
32
33   |   <!--
34   |   <app-add-assignment
35   |   (nouvelAssignment)="onNouvelAssignment($event)"
36   |   *ngIf="formVisible"
37   |   >
38   |   </app-add-assignment>
39   |   -->
40
```

src > app > assignments > add-assignment > TS add-assignment.component.ts > ...

```
1 < import { Component, OnInit } from '@angular/core';
2 import { AssignmentsService } from 'src/app/shared/assignments.service';
3 import { Assignment } from '../assignment.model';
4
5 < @Component({
6   selector: 'app-add-assignment',
7   templateUrl: './add-assignment.component.html',
8   styleUrls: ['./add-assignment.component.css']
9 })
10 < export class AddAssignmentComponent implements OnInit {
11   nomDevoir: string = "";
12   dateRendu: Date;
13
14   constructor(private assignmentsService: AssignmentsService) { }
15
16   ngOnInit(): void {}
17
18 < onSubmit() {
19   const newAssignment = new Assignment();
20   newAssignment.nom = this.nomDevoir;
21   newAssignment.dateDeRendu = this.dateRendu;
22   newAssignment.rendu = false;
23
24   //this.nouvelAssignment.emit(newAssignment);
25   this.assignmentsService.addAssignment(newAssignment)
26     .subscribe(message => console.log(message));
27 }
```

Assignment.component

t dans le composant d'ajout



Puis vérifier que
l'application
fonctionne
comme avant !

Navigation dynamique

Associer un URL unique à chaque assignment dans la vue “détail”

Il va falloir générer des “URLs dynamiques” à la volée en fonction de l’assignment cliqué.

Un peu comme avec le module “express” de NodeJS, on va pouvoir indiquer dans les routes les parties “variables”. Ca se passe dans `app.module.ts` :

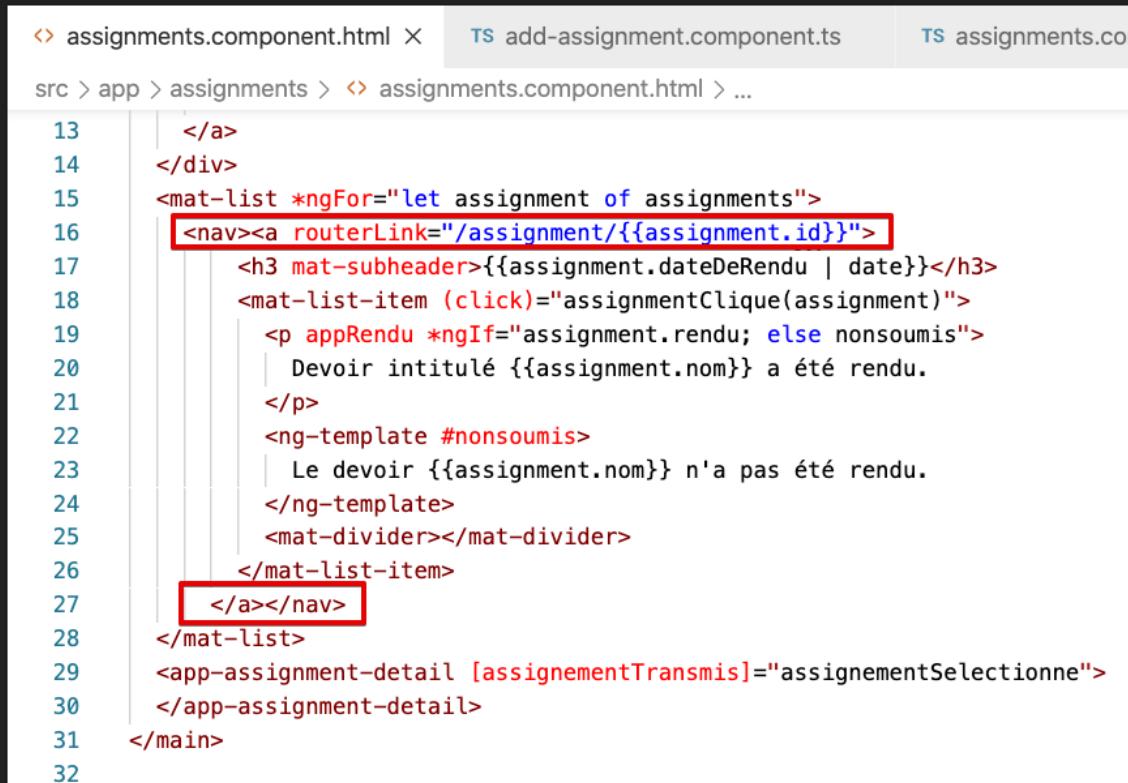
```
const routes : Routes = [
  // home Le :id indique la partie variable. localhost:4200
  // On peut utiliser n'importe quel nom ici,
  {path: 'home', component:AssignmentsComponent},
  {path: 'add', component:AssignmentComponent},
  {path: 'assignment/:id', component:AssignmentDetailComponent}
];
```

`:id` va agir comme une “propriété dynamique”, que l’on pourra voir dans le code TypeScript du composant détail.

L'objet **snapshot** du router Angular

- L'objet **snapshot** du router Angular permet de :
 - Obtenir un paramètre à partir d'un attribut routerLink,
 - Obtenir l'ID assigné par le **routerLink** pour obtenir un "snapshot" d'une des propriétés de l'objet **snapshot**,
- Dans notre exemple, on a appelé cette propriété **id** mais cela aurait pu être n'importe quoi ("nom", "âge" etc.)
- On y accèdera dans le composant avec :
this.route.snapshot.params.id
- **params** ici indique qu'il s'agit d'un élément présent dans l'URL
- Exemple **http://localhost:4200/assignment/1**

Ajout d'un `routerLink` dans le template du composant `assignments.component.html`



```
src > app > assignments > assignments.component.html > ...
13   </a>
14 </div>
15 <mat-list *ngFor="let assignment of assignments">
16   <nav><a routerLink="/assignment/{{assignment.id}}">
17     <h3 mat-subheader>{{assignment.dateDeRendu | date}}</h3>
18     <mat-list-item (click)="assignmentClique(assignment)">
19       <p appRendu *ngIf="assignment.rendu; else nonsoumis">
20         Devoir intitulé {{assignment.nom}} a été rendu.
21       </p>
22       <ng-template #nonsoumis>
23         Le devoir {{assignment.nom}} n'a pas été rendu.
24       </ng-template>
25       <mat-divider></mat-divider>
26     </mat-list-item>
27   </a></nav>
28 </mat-list>
29 <app-assignment-detail [assignmentTransmis]="assignmentSelectionne">
30 </app-assignment-detail>
31 </main>
32
```

Et il va falloir ajouter une propriété id dans le modèle des assignments, et modifier le service d'ajout pour générer un id...

Ajout d'une propriété id aux assignments

TS assignments.component.ts M

TS add-assignment.component.ts M

TS assignment.model.ts M X

src > app > assignments > TS assignment.model.ts > ...

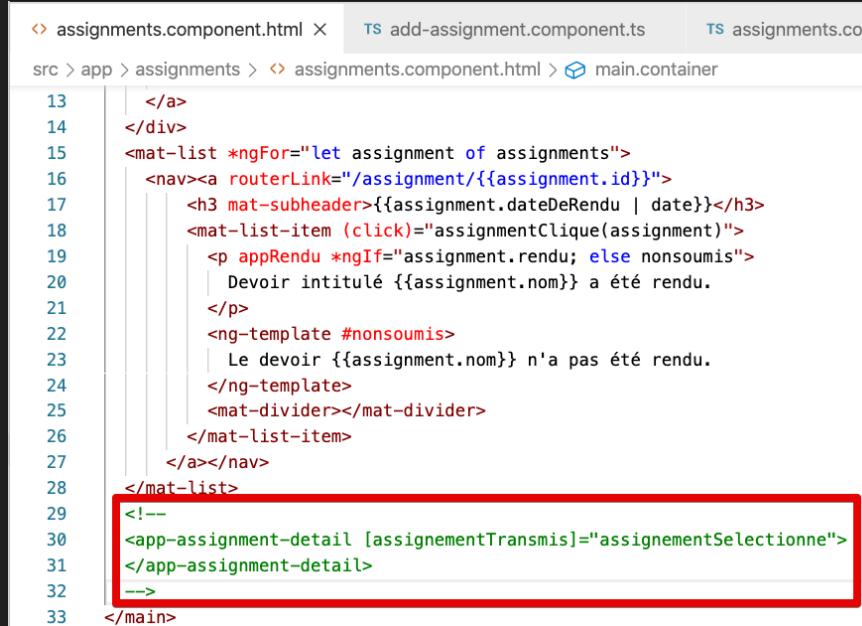
You, il y a 1 seconde | 1 author (You)

```
1 export class Assignment
2 {
3   id!: number;
4   nom!:string;
5   dateDeRendu!:Date;
6   rendu!:boolean;
7 }
```

```
23   },
24   {
25     id:3,
26     nom:"TP3 sur Angular, utilisation du router et de Web Services",
27     dateDeRendu: new Date('2021-01-04'),
28     rendu : false
29   }
30 ];
```

Mais toujours pas de “vue des détails”

On va modifier `assignments.components.html` pour ne plus afficher directement les détails, mais laisser faire le routeur :



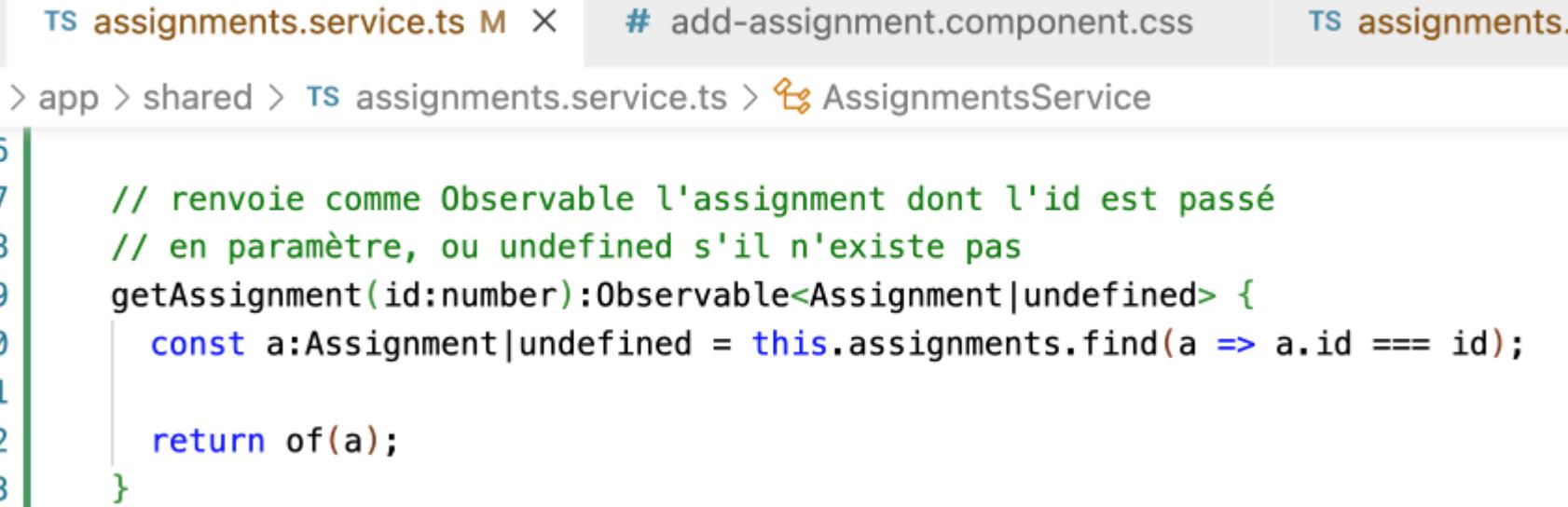
The screenshot shows a code editor with two tabs: `assignments.component.html` and `add-assignment.component.ts`. The `assignments.component.html` tab is active, displaying the following code:

```
13 |     </a>
14 |   </div>
15 |   <mat-list *ngFor="let assignment of assignments">
16 |     <nav><a routerLink="/assignment/{{assignment.id}}">
17 |       <h3 mat-subheader>{{assignment.dateDeRendu | date}}</h3>
18 |       <mat-list-item (click)="assignmentClique(assignment)">
19 |         <p appRendu *ngIf="assignment.rendu; else nonsoumis">
20 |           Devoir intitulé {{assignment.nom}} a été rendu.
21 |         </p>
22 |         <ng-template #nonsoumis>
23 |           Le devoir {{assignment.nom}} n'a pas été rendu.
24 |         </ng-template>
25 |         <mat-divider></mat-divider>
26 |       </mat-list-item>
27 |     </a></nav>
28 |   </mat-list>
29 |   <!--
30 |   <app-assignment-detail [assignmentTransmis]="assignmentSelectionne">
31 |   </app-assignment-detail>
32 |   -->
33 | </main>
```

A red rectangular box highlights the line `<!--` at line 29, indicating that the code between this line and the closing `-->` at line 32 is commented out or has been removed.

Mais toujours pas de “vue des détails”

On va modifier `assignments.service.ts` en ajoutant une fonction qui renvoie un service à partir d'un id:



The screenshot shows a code editor with three tabs at the top: "TS assignments.service.ts M X", "# add-assignment.component.css", and "TS assignments.". Below the tabs, the file structure is shown as > app > shared > TS assignments.service.ts > AssignmentsService. The code itself is as follows:

```
5
6
7 // renvoie comme Observable l'assignment dont l'id est passé
8 // en paramètre, ou undefined s'il n'existe pas
9 getAssignment(id:number):Observable<Assignment|undefined> {
10   const a:Assignment|undefined = this.assignments.find(a => a.id === id);
11
12   return of(a);
13 }
```

Mais toujours pas de “vue des détails”

On va modifier `assignments.detail.component.ts` :

```
ts assignment-detail.component.ts ✘ TS logging.service.ts ⌂ assignment-detail.componen...  
↳ assignments > assignment-detail > TS assignment-detail.component.ts > AssignmentDetailCompon...  
1 import { Component, /*Input,*/ OnInit } from '@angular/core';  
2 import { Assignment } from '../assignment.model';  
3 import { AssignmentsService } from '../../../../../shared/assignments.service';  
4 import { ActivatedRoute } from '@angular/router';  
5  
6 @Component({  
7   selector: 'app-assignment-detail',  
8   templateUrl: './assignment-detail.component.html',  
9   styleUrls: ['./assignment-detail.component.css']  
10 })  
11 export class AssignmentDetailComponent implements OnInit {  
12   /*@Input()*/ assignmentTransmis:Assignment;  
13   1 - supprimer le @Input, plus de transfert inter composants  
14   constructor(private assignmentsService: AssignmentsService,  
15   | | | | private route: ActivatedRoute) { }  
16  
17   ngOnInit(): void {  
18     this.getAssignment();  
19   }  
20  
21   getAssignment() {  
22     // on récupère l'id dans le snapshot passé par le routeur  
23     // le "+" force la conversion de l'id de type string en "number"  
24     const id = +this.route.snapshot.params.id;  
25     this.assignmentsService.getAssignment(id)  
26       .subscribe(assignment => this.assignmentTransmis = assignment);  
27   }  
28 }
```

Et on teste
l'application !

A screenshot of a web browser window displaying the application's homepage. The title bar shows the URL `localhost:4200/assignments`. The main content area has a purple header: Application de gestion des devoirs à rendre (Assignments). Below it, a card displays the text: **TP1 sur WebComponents, un lecteur audio amélioré**, dated Jan 17, 2020. A red button labeled **DELETE** is visible.

A screenshot of a web browser window displaying a different page of the application. The title bar shows the URL `localhost:4200/assignments`. The main content area has a purple header: Application de gestion des devoirs à rendre (Assignments). Below it, a card displays the text: **TP2 sur Angular, un joli gestionnaire de devoirs (Assignments)**, dated Dec 15, 2020. A checkbox labeled **Devoir rendu** is visible.

Et si on clique sur DELETE ou sur la checkbox “rendu” dans la vue détails ?

On veut retourner directement à la page d'accueil....

On va utiliser le routeur programmatiquement, avec sa méthode
navigate([...]);

```
TS assignment-detail.component.ts X  TS logging.service.ts      <> assignment-detail.com
: > app > assignments > assignment-detail > TS assignment-detail.component.ts > Assignmen
1   import { Component, /*Input,*/ OnInit } from '@angular/core';
2   import { Assignment } from '../assignment.model';
3   import { AssignmentsService} from '../../../../../shared/assignments.service';
4   import { ActivatedRoute, Router } from '@angular/router';
5
6   @Component({
7     selector: 'app-assignment-detail',
8     templateUrl: './assignment-detail.component.html',
9     styleUrls: ['./assignment-detail.component.css']
10  })
11  export class AssignmentDetailComponent implements OnInit {
12    /*@Input()*/ assignementTransmis:Assignment;
13
14    constructor(private assignmentsService: AssignmentsService,
15                private route: ActivatedRoute,
16                private router:Router) { }
```

```
onAssignmentRendu() {
  this.assignmentTransmis.rendu = true;

  this.assignmentsService.updateAssignment(this.assignmentTransmis)
    .subscribe(message => console.log(message));

  this.router.navigate(["/home"]);
}

onDelete() {
  this.assignmentsService.deleteAssignment(this.assignmentTransmis)
    .subscribe((message) => console.log(message));

  /*this.assignmentTransmis = null;*/
  this.router.navigate(["/home"]);
}
```

Envoie de “queries” ou “fragments”
à des routes

Mais de quoi s'agit-il ?

De paramètres dynamiques que l'on va pouvoir passer à des composants lors de la navigation.

S'utilisent à travers le router programmatiquement :

```
onClickEdit() {
  this.router.navigate(commands: ['/assignment', this.assignment.id, 'edit'],
    extras: {queryParams: {name: this.assignment.name}, fragment: 'editing'}
  );
}
```

Ou en passant des informations par l'URL :

ⓘ localhost:4200/assignment/1/edit?name=One#editing

Mais de quoi s'agit-il ?

On peut utiliser les “fragments” pour naviguer à un endroit précis d'une page :

ⓘ localhost:4200/assignment/1/edit?name=One#editing

Et on peut aussi utiliser l'objet snapshot :

```
this.route.snapshot.queryParams;  
this.route.snapshot.fragment;
```

Mise en pratique : ajout d'un composant edit-assignment

Comme d'habitude, dans le dossier “assignments” :

```
ng g c --skip-tests=true edit-assignment
```

The screenshot shows the Angular CLI workspace structure for the 'edit-assignment' component. On the left, a tree view of the project structure is displayed:

- ASSIGNMENT-APP
 - e2e
 - node_modules
 - src
 - app
 - assignments
 - add-assignment
 - assignment-detail
 - edit-assignment
 - # edit-assignment.component.css
 - edit-assignment.component.html
 - edit-assignment.component.ts
 - assignment.model.ts

An 'Add' button is visible next to the 'src' folder. On the right, the code editor shows the 'edit-assignment.component.html' file:

```
1 <p>edit-assignment works!</p>
2
```

The 'edit-assignment.component.html' file contains the text: <p>edit-assignment works!</p>

Nouveau code pour son template (vous pouvez sélectionner le texte, similaire à add-assignment...)

```
<div class="container" *ngIf="assignment">
  <h1>Edition de l'assignment {{assignment.nom}}</h1>
  <form ngForm class="form" #formupdate
    (submit)="onSaveAssignment(); formupdate.reset(); ">
    <mat-form-field>
      <input matInput placeholder="Edition du nom"
        [(ngModel)]="nomAssignment" name="assignment-name">
    </mat-form-field>
    <mat-form-field>
      <input matInput [matDatepicker]="picker"
        placeholder="Edition de la date"
        [(ngModel)]="dateDeRendu" name="date">
      <mat-datepicker-toggle matSuffix [for]="picker">
      </mat-datepicker-toggle>
      <mat-datepicker #picker></mat-datepicker>
    </mat-form-field>
```

```
  <button mat-raised-button color="primary"
    [disabled]="{{ (!nomAssignment) || (!dateDeRendu) }}>
    Sauver
  </button>
</form>
</div>
```

Nouveau code pour la partie métier (très similaire au composant details...)

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { AssignmentsService } from
'src/app/shared/assignments.service';
import { Assignment } from '../assignment.model';

@Component({
  selector: 'app-edit-assignment',
  templateUrl: './edit-assignment.component.html',
  styleUrls: ['./edit-assignment.component.css'],
})
export class EditAssignmentComponent implements OnInit {
  assignment!: Assignment | undefined;
  nomAssignment!: string;
  dateDeRendu!: Date;

  constructor(
    private assignmentsService: AssignmentsService,
    private route: ActivatedRoute,
    private router: Router
  ) {}

  ngOnInit(): void {
    this.getAssignment();
  }
}
```

```
getAssignment() {
  // on récupère l'id dans le snapshot passé par le routeur
  // le "+" force l'id de type string en "number"
  const id = +this.route.snapshot.params['id'];

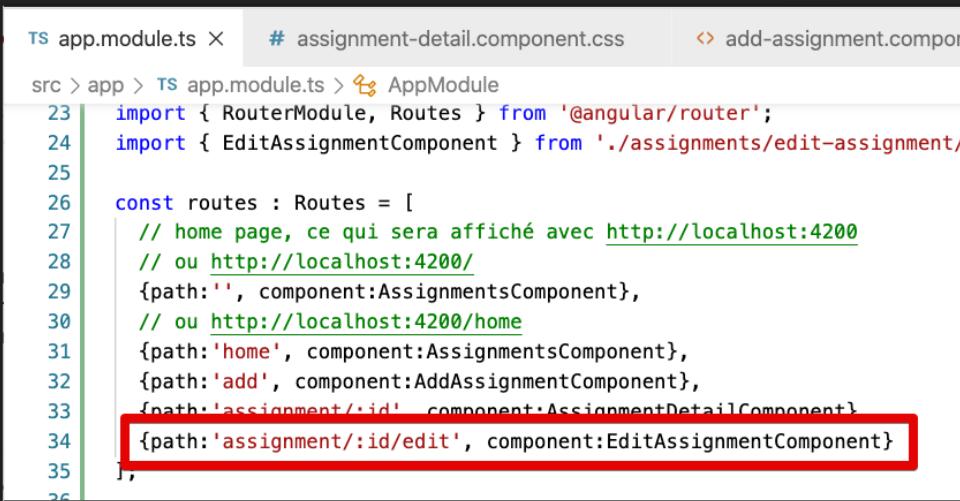
  this.assignmentsService.getAssignment(id).subscribe((assignment) =>
{
  if (!assignment) return;
  this.assignment = assignment;
  // Pour pré-remplir le formulaire
  this.nomAssignment = assignment.nom;
  this.dateDeRendu = assignment.dateDeRendu;
});

onSaveAssignment() {
  if (!this.assignment) return;

  // on récupère les valeurs dans le formulaire
  this.assignment.nom = this.nomAssignment;
  this.assignment.dateDeRendu = this.dateDeRendu;
  this.assignmentsService
    .updateAssignment(this.assignment)
    .subscribe((message) => {
      console.log(message);

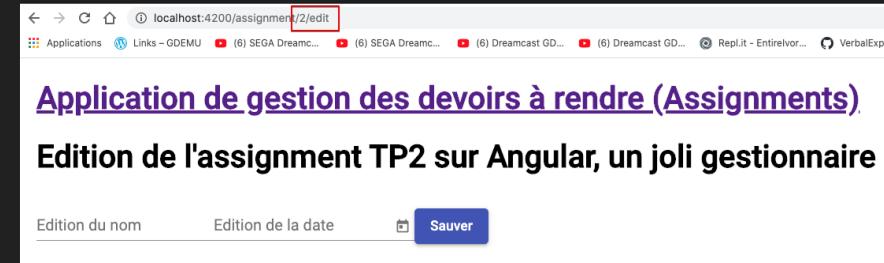
      // navigation vers la home page
      this.router.navigate(['/home']);
    });
}
}
```

Ajout de la route pour l'édition dans le module



```
src > app > TS app.module.ts > # assignment-detail.component.css < add-assignment.component.css
src > app > TS app.module.ts > AppModule
23 import { RouterModule, Routes } from '@angular/router';
24 import { EditAssignmentComponent } from './assignments/edit-assignment/e
25
26 const routes : Routes = [
27   // home page, ce qui sera affiché avec http://localhost:4200
28   // ou http://localhost:4200/
29   {path: '', component:AssignmentsComponent},
30   // ou http://localhost:4200/home
31   {path: 'home', component:AssignmentsComponent},
32   {path: 'add', component:AddAssignmentComponent},
33   {path: 'assignment/:id', component:AssignmentDetailComponent}
34   {path: 'assignment/:id/edit', component:EditAssignmentComponent}
35 ],
36
```

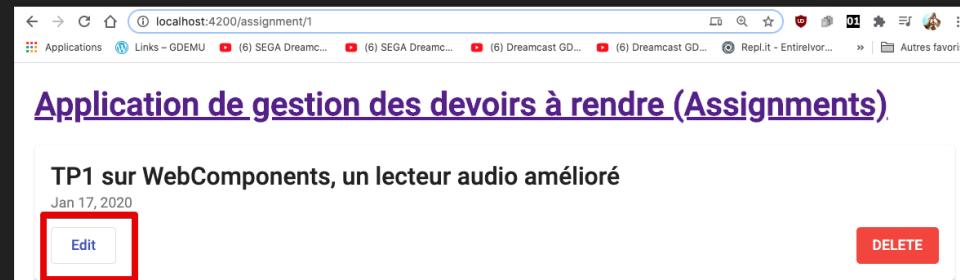
Et on teste dans l'application, en ajoutant /edit à la fin des URLs détails



Ajout d'un bouton EDIT dans le composant detail

Premier essai avec une navigation “statique”, on modifie le template du composant **assignment-detail.component.html**

```
assignment-detail.component.html < app.component.html TS app
src > app > assignments > assignment-detail > assignment-detail.component.html
1  <div class="container">
2    <mat-card>
3      <mat-card-title>{{assignmentTransmis.nom}}</mat-card-title>
4      <mat-card-subtitle>{{assignmentTransmis.dateDeRendu | date:'short'}}</mat-card-subtitle>
5      <div id="bottom">
6        <mat-checkbox *ngIf="!assignmentTransmis.rendu" (click)="rendu(assignmentTransmis)">
7          Devoir rendu
8        </mat-checkbox>
9
10       <nav><a [routerLink]="['/assignment', 1, 'edit']">
11         <button mat-stroked-button color="primary">
12           Edit
13         </button>
14       </a></nav>
15
16       <button mat-flat-button color="warn"
17           (click)="onDelete()">
18         DELETE</button>
19
20     </div>
21   </mat-card>
22 </div>
```



Soucis : c'est toujours l'assignment 1 qui est édité

On va supprimer la navigation statique et à la place appeler une méthode `onClickEdit()` et faire la navigation programmatiquement

The screenshot shows two tabs open in a code editor: `assignment-detail.component.html` and `assignment-detail.component.ts`. The `assignment-detail.component.html` file contains the following HTML:

```
<div class="container">
  <mat-card>
    <mat-card-title>{{assignmentTitre}}</mat-card-title>
    <mat-card-subtitle>{{assignmentSubTitre}}</mat-card-subtitle>
    <div id="bottom">
      <mat-checkbox *ngIf="!assignmentDevoirRendu">
        Devoir rendu
      </mat-checkbox>
    </div>
    <button mat-stroked-button>Edit</button>
  </mat-card>
</div>
```

The `assignment-detail.component.ts` file contains the following TypeScript code:

```
src > app > assignments > assignment-detail > TS assignment-detail.component.ts > ...
43   /*this.assignmentTransmis = null;*/
44
45 }
46
47 onClickEdit() {
48   this.router.navigate(['/assignment', this.assignmentTransmis.id, 'edit']);
49 }
```

A red annotation box highlights the line `this.router.navigate(['/assignment', this.assignmentTransmis.id, 'edit']);`. A red arrow points from this box to the text "Correspond à /assignment/2/edit par ex..." which is displayed above the code editor window.

Et on teste dans l'application, en vérifiant que l'id des assignments est bon !

Passage de fragments dans l'URL (paramètres HTTP)

```
onClickEdit() {  
  this.router.navigate(["/assignment", this.assignmentTransmis.id, 'edit'],  
  {queryParams:{nom:this.assignmentTransmis.nom}, fragment: 'edition'});  
}
```



Utile pour naviguer à un endroit précis de la page (un fragment)

Comment récupérer les queryParams et fragments dans le code d'un composant?

On va utiliser à nouveau **route: ActivatedRoute** pour cela...

The screenshot shows a code editor and a browser window side-by-side.

Code Editor (left):

```
edit-assignment.component.ts
src > app > assignments > edit-assignment
19
20     ngOnInit(): void {
21         this.getAssignment();
22
23         // affichage des queryParams
24         console.log("Query Params : " + this.route.snapshot.queryParams);
25         console.log(this.route.snapshot.fragment);
26         console.log("Fragment : " + this.route.snapshot.fragment);
27         console.log(this.route.snapshot.fragment);
28     }
29 }
```

Browser Window (right):

- Address Bar:** localhost:4200/assignment/1/edit?nom=TP1%20sur%20WebComponents,%20un%20lecteur%20audio%20amélioré#édition
- Title:** Application de gestion des devoirs à rendre (Assignment)
- Section:** Edition de l'assignment TP1 sur WebComponents, un lecteur audio amélioré
- Form Fields:** Edition du nom, Edition de la date, Sauver button
- Console Tab:** Shows the output of the console.log statements from the code:
 - Query Params : {nom: "TP1 sur WebComponents, un lecteur audio amélioré"}
 - Fragment : édition

Gestion des accès / mode admin etc.

- Création d'un service de gestion des autorisations : on va pouvoir filtrer quelles routes seront accessibles si on n'est pas identifié ou si on l'est par exemple...
 - Basé sur une authentification, c'est le cas le plus courant...
- On va voir ce qu'on appelle la “gestion des restrictions d'accès aux routes”.
 - Par exemple, autoriser la navigation après édition que si on a sauvegardé les modifications.
- Par exemple, on va voir comment créer une gestion de login pour donner des privilège au login “admin”.
 - Seul l'admin pourra éditer un Assignment par exemple...

Création d'un service d'authentification

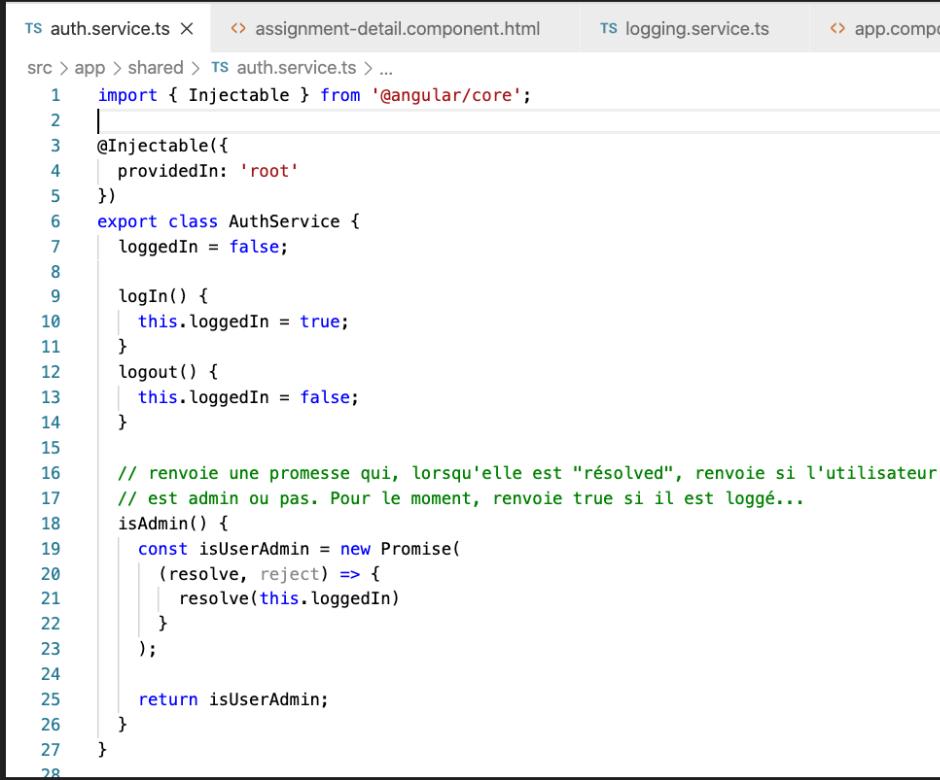
Dans le dossier shared, comme d'habitude pour les services,

```
ng g s --skip-tests=true auth
```

The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays the project structure under 'ASSIGNMENT-APP'. It includes a 'shared' folder containing 'assignments.service.ts', 'auth.service.ts' (which is currently selected), 'logging.service.ts', and 'rendu.directive.ts'. Below these are 'app.component.css' and 'app.component.html'. The status bar at the bottom indicates 'M' for modified. On the right, the main editor window shows the code for 'auth.service.ts'. The code defines an Injectable 'AuthService' with a constructor. The file path 'src > app > shared > auth.service.ts' is visible above the code area.

```
TS auth.service.ts × assignment-detail.component.html T  
src > app > shared > TS auth.service.ts > ...  
1 import { Injectable } from '@angular/core';  
2  
3 @Injectable({  
4   providedIn: 'root'  
5 })  
6 export class AuthService {  
7   constructor() { }  
8 }  
9  
10
```

On ajoute dans le service une propriété loggedIn et des méthodes pour dire qu'on s'est loggué ou délogué



The screenshot shows a code editor with two tabs open: 'auth.service.ts' and 'logging.service.ts'. The 'auth.service.ts' tab is active, displaying the following TypeScript code:

```
TS auth.service.ts <-- assignment-detail.component.html TS logging.service.ts <-- app.compc  
src > app > shared > TS auth.service.ts > ...  
1 import { Injectable } from '@angular/core';  
2  
3 @Injectable({  
4   providedIn: 'root'  
5 })  
6 export class AuthService {  
7   loggedIn = false;  
8  
9   logIn() {  
10    this.loggedIn = true;  
11  }  
12  
13   logout() {  
14    this.loggedIn = false;  
15  }  
16  
17   // renvoie une promesse qui, lorsqu'elle est "résolved", renvoie si l'utilisateur  
18   // est admin ou pas. Pour le moment, renvoie true si il est loggé...  
19   isAdmin() {  
20     const isUserAdmin = new Promise(  
21       (resolve, reject) => {  
22         resolve(this.loggedIn)  
23       }  
24     );  
25  
26     return isUserAdmin;  
27   }  
28 }
```

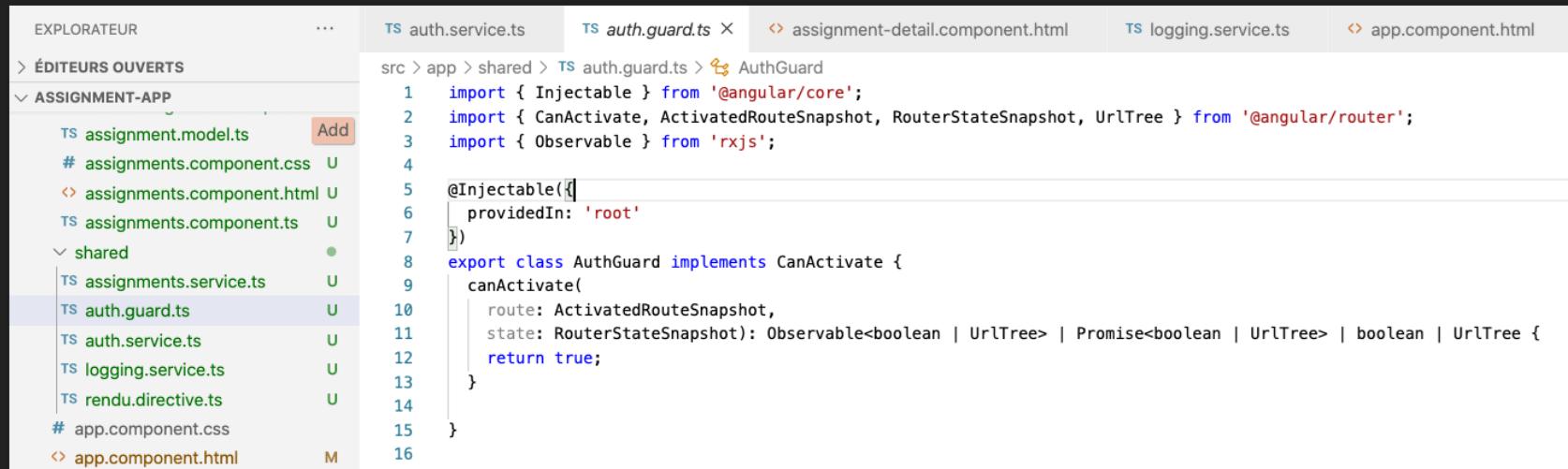
The 'logging.service.ts' tab is visible in the background.

Création d'un “guard” Angular (“auth guard”)

Avec CLI, toujours dans le dossier “shared” :

```
ng g guard --skip-tests=true auth
```

Choisir l'option par défaut (CanActivate)



The screenshot shows a code editor with several tabs open in the background: auth.service.ts, auth.guard.ts (which is currently selected), assignment-detail.component.html, logging.service.ts, and app.component.html. The auth.guard.ts tab contains the following TypeScript code:

```
src > app > shared > TS auth.guard.ts > AuthGuard
1 import { Injectable } from '@angular/core';
2 import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, UrlTree } from '@angular/router';
3 import { Observable } from 'rxjs';
4
5 @Injectable({
6   providedIn: 'root'
7 })
8 export class AuthGuard implements CanActivate {
9   canActivate(
10     route: ActivatedRouteSnapshot,
11     state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
12     return true;
13   }
14 }
15
16 }
```

On va dans ce “auth guard” utiliser le service d’authentification pour donner le droit de naviguer ou pas...

```
export class AuthGuard implements CanActivate {
  constructor(private authService:AuthService) { }

  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> |
    //return true;

    return this.authService.isAdmin()
      .then(authentifie => {
        if (authentifie) {
          return true;
        } else {
          return false;
        }
      })
  }
}
```

Et on ajoute “canActivate” à la route composant d'édition

The screenshot shows the VS Code interface with the following details:

- EXPLORATEUR**: Shows the project structure with files like assignment.model.ts, assignments.component.css, assignments.component.html, assignments.component.ts, auth.guard.ts, auth.service.ts, logging.service.ts, rendu.directive.ts, app.component.css, app.component.html, app.component.spec.ts, app.component.ts, and app.module.ts.
- ÉDITEURS OUVERTS**: Shows the open files: component.html, app.module.ts, and logging.service.ts.
- app.module.ts** (active tab):

```
src > app > TS app.module.ts > routes
22 import {AssignmentsService} from './shared/assignments.service';
23 import {AuthGuard} from './shared/auth.guard';
24
25 import {RouterModule, Routes} from '@angular/router';
26 import {EditAssignmentComponent} from './assignments/edit-assignment.component';
27
28 const routes: Routes = [
29   // home page, ce qui sera affiché avec http://
30   // ou http://localhost:4200/
31   {path: '', component: AssignmentsComponent},
32   // ou http://localhost:4200/home
33   {path: 'home', component: AssignmentsComponent},
34   {path: 'add', component: AddAssignmentComponent},
35   {path: 'assignment/:id', component: AssignmentDetailComponent},
36   {
37     path: 'assignment/:id/edit',
38     component: EditAssignmentComponent,
39     canActivate: [AuthGuard]
40   }
41];
```

The screenshot shows the auth.service.ts file with handwritten annotations in red:

- ligne 7: loggedIn = false;** (boxed) - **Tester en mettant à true,** **là on doit pouvoir éditer tout le temps**
- ligne 10: this.loggedIn = true;** - **à true,** **là on doit pouvoir éditer tout le temps**
- ligne 13: this.loggedIn = false;** - **à false,** **là on doit pas pouvoir éditer tout le temps**

```
TS auth.guard.ts      TS auth.service.ts ×  ◉ assignment-d
src > app > shared > TS auth.service.ts > AuthService >
1 import { Injectable } from '@angular/core';
2
3 @Injectable({
4   providedIn: 'root'
5 })
6 export class AuthService {
7   loggedIn = false; Tester en mettant à true, là on doit pouvoir éditer tout le temps
8
9   logIn() {
10    this.loggedIn = true; à true, là on doit pouvoir éditer tout le temps
11  }
12
13   logout() {
14    this.loggedIn = false; à false, là on doit pas pouvoir éditer tout le temps
15  }
16
17   // renvoie une promesse qui lorsque elle est
18   // est admin ou pas. Pour le moment, elle renvoie
19   isAdmin() {
20     const isUserAdmin = new Promise(
21       (resolve, reject) => {
22         resolve(this.loggedIn);
23       }
24     );
25
26     return isUserAdmin;
27   }
}
```

Ajout d'une GUI simplifiée pour se logguer...

The screenshot shows a code editor with several tabs: 'app.component.html', 'app.module.ts', 'polyfills.ts', and 'material pour dire si on'. The 'app.component.html' tab is active, displaying the following code:

```
src > app > app.component.html > ...
1  <div class="container">
2    <h1>
3      <nav><a rou
4    </h1>
5    <mat-slide-to
6  </div>
7
8  <router-outlet>
```

Below the code editor is a browser window showing the application's home page. The title bar reads 'localhost:4200/home'. The page content is:
Application de gestion des devoirs à rendre (Assignments)
A button labeled 'Login' is highlighted with a red box.

Et on l'ajoute dans le template de **app.component.html**

Ajout de la partie métier

```
TS app.component.ts X TS app.component.spec.ts TS app.module.ts TS polyfills.ts
src > app > TS app.component.ts > ...
1 import { Component } from '@angular/core';
2 import { Router } from '@angular/router';
3 import { AuthService } from './shared/auth.service';
4
5 @Component({
6   selector: 'app-root',
7   templateUrl: './app.component.html',
8   styleUrls: ['./app.component.css']
9 })
10 export class AppComponent {
11   title = 'Application de gestion des devoirs à rendre (Assignments)';
12
13   constructor(private authService:AuthService, private router: Router) {}
14
15   login() {
16     if(!this.authService.loggedIn) {
17       this.authService.logIn();
18     } else {
19       this.authService.logout();
20       // et on renvoie vers la home page
21       this.router.navigate(['/home']);
22     }
23   }
24 }
```

- Et on teste dans l'application,
- Pour cela ajouter un écouteur (click) associé au <mat-slider-toggle> qui appelle la méthode login() !
- On essaie d'éditer sans être loggué
-> ça doit renvoyer vers la page d'accueil.
- Si on est logué on doit pouvoir éditer.

Disabler le bouton EDIT si on n'est pas loggué...

Ça se passe dans le composant detail.....

The screenshot shows two tabs in a code editor: 'assignment-detail.component.html' and 'assignment-detail.component.ts'. The 'assignment-detail.component.html' tab contains the following code:

```
9
10    <button mat-stroked-button color="primary"
11      (click)="onClickEdit()"
12      [disabled]="!isAdmin()"
13    >
14      Edit
15    </button>
```

The line '[disabled]="!isAdmin()" is highlighted with a red box.

N'oubliez pas de l'injecter dans le constructeur !

The screenshot shows two tabs in a code editor: 'assignment-detail.component.ts' and 'auth.guard.ts'. The 'assignment-detail.component.ts' tab contains the following code:

```
53
54  isAdmin():boolean {
55    return this.authService.loggedIn;
56  }
57 }
```

The 'auth.guard.ts' tab contains the following code:

```
53
54  isAdmin():boolean {
55    return this.authService.loggedIn;
56  }
57 }
```

A red arrow points from the 'isAdmin()' call in the component code to the implementation in the guard file.

- Et on teste dans l'application,
- Le bouton doit être grisé/dégrisé selon qu'on est loggué ou pas...

Exercices à faire jq la fin du cours / pour la prochaine fois

- Refaire tout ce que je vous ai montré (un zip avec le projet point de départ est sur la page du cours).
- Modifier le programme pour avoir une identification par login/password
 - Ajouter un tableau de login/password/role (user/admin) dans le service d'authentification
 - Modifiez le code pour avoir isLoggedIn() et isAdmin() au lieu de juste isAdmin()
- Au lieu du slider login, utiliser un bouton “connecter” qui redirige vers un formulaire de connexion (qui peut éventuellement remplacer le bouton, ou plus tard être un dialogue Material)
- Gérer le cas spécial de l'admin. Si on est loggué on peut naviguer partout mais on ne peut éditer/supprimer que si on est admin.
 - Mettre les boutons en grisé si on ne l'est pas

Partie 8 : utilisation d'APIs RESTful

Ce que nous allons voir...

Utilisation d'une BD NoSQL MongoDB et on va définir côté serveur des WebServices (une API pour accéder aux données)

Utilisation du module Angular HttpClient pour requêter cette API

On va voir comment récupérer des données (GET) et travailler avec

On verra comment faire le CRUD sur les données (POST, UPDATE, DELETE)

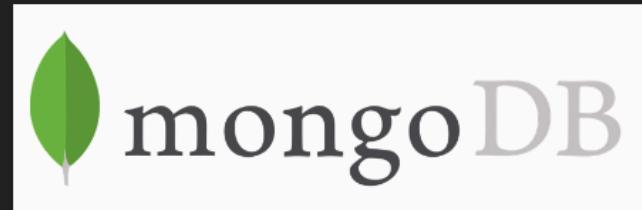
On verra comment gérer les erreurs

BD Mongo (dans le cloud) et API (locale)

MongoDB

Vous la verrez plus tard en détails en Master 1

Ici [un cours plus complet dessus](#) que je donne...

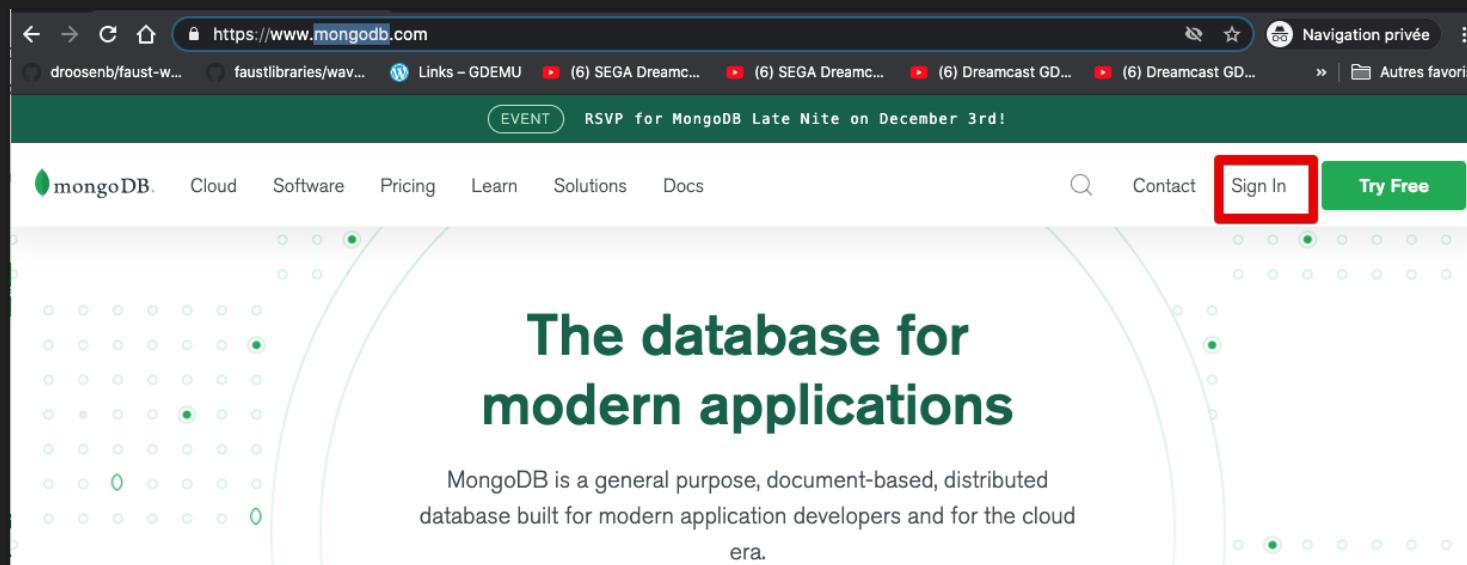


Quand on fait du MongoDB avec du Angular on dit que ça fait partie de la "**MEAN STACK**" (Mongo, Express, Angular, Node)

On va héberger une base Mongo dans le Cloud (Mongo Cloud sur mongodb.com) mais on gardera pour le moment la partie cliente (qui définit l'API) en local, sous NodeJS...

Création d'une base Mongo dans le cloud

Etape 1 : allez sur [mongodb.com](https://www.mongodb.com), créez un compte si besoin et connectez-vous



Création d'une base Mongo dans le cloud

Etape 1 : allez sur [mongodb.com](https://www.mongodb.com), créez un compte si besoin et connectez-vous

Création d'une base Mongo dans le cloud

Etape 0 : allez sur [mongodb.com](https://cloud.mongodb.com), créez une nouvelle organisation et un nouveau projet (si vous n'en avez pas déjà)

The screenshot shows the MongoDB Cloud interface. On the left, there's a sidebar with 'PREFERENCES' (Legacy 2FA, Personalization, Invitations, Organizations, Public API Access) and 'Organizations' (All Organizations, Master 2 MIAGE (IA2,...)). The main area shows the 'Organization' section with 'PROJECTS' selected. A sub-menu on the right lists 'Alerts', 'Activity Feed', 'Settings', 'Access Manager', 'Billing', and 'Support'. The main content area is titled 'Create a Project' under 'MASTER 2 MIAGE (IA2, MBDS, INTENSE, ESTIA ETC.) > PROJECTS'. It has a 'Name Your Project' input field containing 'Tuto Angular Mongo Cloud', an 'Add Members' button, and 'Next' and 'Cancel' buttons.

Création d'une base Mongo dans le cloud

Etape 2 : créer un cluster

The screenshot shows a web browser window for [cloud.mongodb.com](https://cloud.mongodb.com/v2/5fc8a45153448104c1f7d686#clusters). The URL bar shows the path `v2/5fc8a45153448104c1f7d686#clusters`. The top navigation bar includes links for Applications, faosenb/faust-w..., faulibraries/wav..., Links - GDEMU, (6) SEGA Dream..., (6) SEGA Dream..., and (6) Dreamcast GD... The main navigation menu on the left has sections for DATA STORAGE (Clusters, Triggers, Data Lake), SECURITY (Database Access, Network Access, Advanced), and Atlas. The 'Clusters' section is selected and highlighted in green. The central content area is titled 'Clusters' and shows a placeholder message 'Find a cluster...' with a magnifying glass icon. Below this is a large green button with a plus sign and the text 'Create a cluster'. To the right of the button, the text 'Choose your cloud provider, region, and specs.' is displayed. A red box highlights the 'Build a Cluster' button. At the bottom, a note states: 'Once your cluster is up and running, live migrate an existing MongoDB database into Atlas with our [Live Migration Service](#)'.

Création d'une base Mongo dans le cloud

Etape 2 : créer un cluster (suite)

The screenshot shows the MongoDB Atlas Clusters page for the "M2 MIAGE (INTENSE, MBDS, IA2) > TUTOS ANGULAR REACT ETC." project. The page title is "Clusters". A green button "Create a New Cluster" is visible in the top right. On the left, there's a sidebar for "Dedicated MongoDB Clusters" with a red box around the "CONNECT" button for Cluster0. Below it, there are sections for "CLUSTER TIER", "REGION", "TYPE", and "LINKED REALM APP". The "CLUSTER TIER" section shows "M0 Sandbox (General)". The "REGION" section shows "AWS / Frankfurt (eu-central-1)". The "TYPE" section shows "Replica Set - 3 nodes". The "LINKED REALM APP" section shows "None Linked". The main content area displays three cards for Cluster0: "Operations R: 0 W: 0" (100.0/s), "Logical Size 0.0 B" (512.0 MB max), and "Connections 0" (500 max). A red box highlights the "CONNECT" button for Cluster0. A callout box "Enhance Your Experience" encourages upgrading the cluster. The bottom left shows a starting cost of \$0.13 per month.

M2 MIAGE (INTENSE, MBDS, IA2) > TUTOS ANGULAR REACT ETC.

Clusters

Create a New Cluster

Find a cluster...

SANDBOX

Cluster0
Version 4.2.10

CONNECT METRICS COLLECTIONS ...

CLUSTER TIER
M0 Sandbox (General)

REGION
AWS / Frankfurt (eu-central-1)

TYPE
Replica Set - 3 nodes

LINKED REALM APP
None Linked

Operations R: 0 W: 0 100.0/s

Logical Size 0.0 B 512.0 MB max

Connections 0 500 max

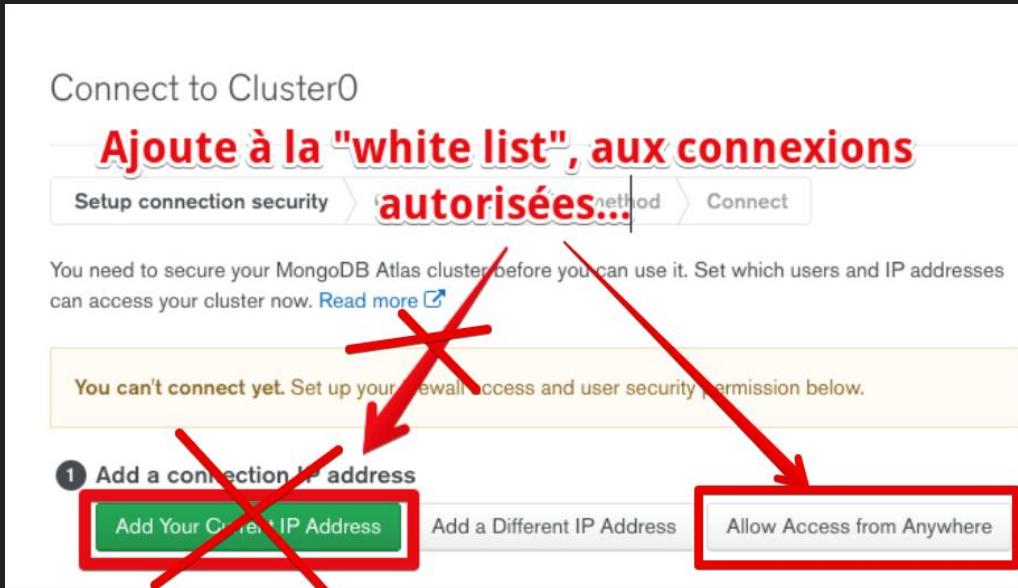
Last 6 Hours Last 6 Hours

Enhance Your Experience
For dedicated throughput, richer metrics and enterprise security options, upgrade your cluster now!

Upgrade

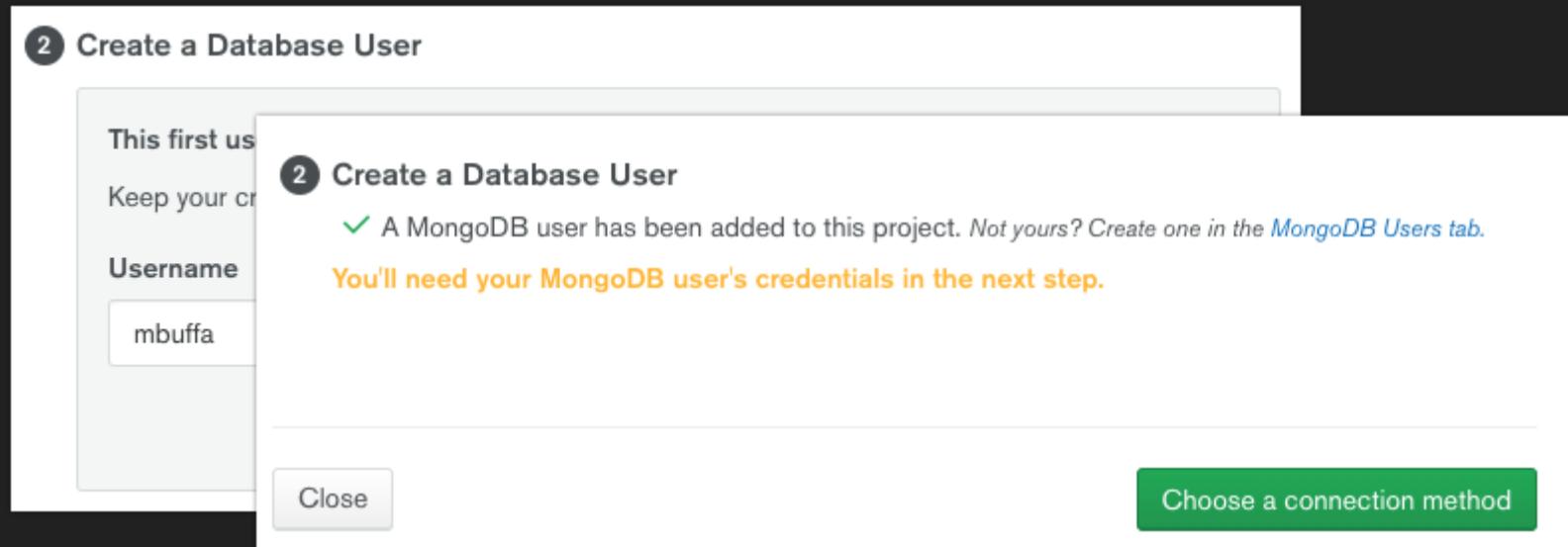
Starting at \$0.13/mo
*estimated cost \$0.13/mo

MongoDB configuration du cluster: network access



Si vous avez oublié de faire ça, vous pouvez cliquer sur “network access” et ajouter 0.0.0.0/0 pour donner accès de n’importe quelle adresse !

MongoDB configuration du cluster: mongoDB user



MongoDB configuration du cluster: Connexion

The screenshot shows the MongoDB Atlas Cluster Configuration interface. On the left, there's a sidebar with 'Connect to Cluster0' and a green checkmark for 'Setup connection security'. Below it are sections for 'Choose a connection profile', 'SECURITY' (with 'Database Access' highlighted), 'Network Access', and 'Advanced' (which has a red box around its icon). At the bottom of the sidebar are 'Go Back' and 'Next Step' buttons.

In the main area, there's a dropdown menu with 'M2 Miage (INTENSE...)'. Below it are tabs for 'DATA STORAGE' and 'SECURITY'. A modal window titled 'Add New Database User' is open, with the text 'Create a database user to grant an application or user, access to databases and collections' and a note 'You can reseigner la méthode d'authentification et le nom de rôle'. The 'Database Users' tab is selected, showing two users:

User Name	Authentication Method	MongoDB Roles
mb	SCRAM	readWriteAnyDatabase@admin
mbuffa	SCRAM	atlasAdmin@admin

At the bottom of the modal, there are buttons for 'Autogenerate Secure Password' and 'Copy'. Below the modal, there's a section for 'Database User Privileges' with a note 'Select a built-in role or privileges for this user.' and a dropdown menu set to 'Read and write to any database'. To the right, there's a link to 'MongoDB documentation'.

Ajout d'une collection “assignments”

The screenshot shows the MongoDB Atlas interface for a cluster named "M2 Miage (INTENSE,...)". The "Collections" tab is selected, and the "assignments" collection is currently active. The "Insert to Collection" dialog is open, displaying a JSON document with the following structure:

```
1  _id : ObjectId("61d6a16ae0096cbde528e41d")
2  nom : "Exemple de devoir dans le cloud"
3  dateDeRendu : 2021-12-30T23:00:00.000+00:00
4  rendu : false
```

A red box highlights the first four lines of this JSON document. To the right of the document, there are four input fields with their corresponding data types: ObjectId, String, Date, and Boolean. A note at the bottom left suggests adding an id field of type int32. The "Insert" button is highlighted with a red box.

M2 MIAGE (INTENSE, MBDS, IA2) > TUTOS ANGULAR REACT ETC. > CLUSTERS

Cluster0

VERSION 4.2.10 REGION AWS Frankfurt (eu-central-1)

Overview Real Time Metrics Collections

DATABASES: 1 COLLECTIONS: 1

+ Create Database

VIEW { } ≡

assignments.assig

COLLECTION SIZE: 0B TOTAL

Find Indexes

FILTER {"filter": "exa

1 _id : ObjectId("61d6a16ae0096cbde528e41d")
2 nom : "Exemple de devoir dans le cloud"
3 dateDeRendu : 2021-12-30T23:00:00.000+00:00
4 rendu : false

ObjectId
String
Date
Boolean

Rajouter id:1 en int32 pour que ça marche par la suite...

Cancel Insert

Création d'un API avec NodeJS + Mongoose

- Le code du petit projet créant une API est disponible ici : [api.zip](#)
- Le dézipper à côté de assignment-app, et faire “npm install”
- Editer le code de serveur.js pour mettre votre propre URI de connexion à votre base (qu'on peut retrouver sur le site mongo, avec cluster/connect/connect application (slide 188)
 - Changer dans l'URI : nom / mot de passe (que vous retrouvez dans “database access”). Supprimez les < et > au début et à la fin du mot de passe.
 - Changer le nom de la base pour mettre “assignments” à la place de myFirstDatabase

```
// remplacer toute cette chaîne par l'URI de connexion à votre propre base dans le cloud s
const uri = 'mongodb+srv://mb:P7zM3VePm0caWA1L@cluster0.zqtee.mongodb.net/assignments?retryWrites=true&w=majority';
```



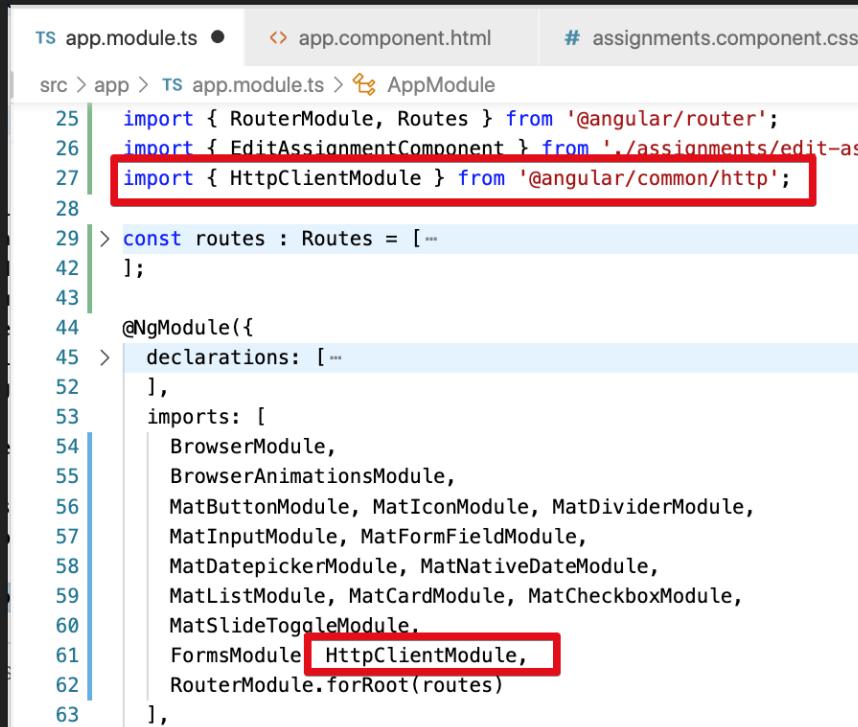
Etudiez le code (serveur.js, routes/assignments.js et model/assignments.js)

- Par rapport à l'an dernier, plusieurs nouvelles choses sont proposées ici :
 - a. Avec [express](#) on utilise [`app.route`](#) et non pas [`app.get`](#), [`app.post`](#) comme on a vu l'an dernier. C'est une feature d'express qui permet de définir les routes de manière plus élégante, notamment en les chaînant sur un même URI (ex: GET, POST, DELETE, UPDATE sur `/assignment/:id`)

```
app.route(prefix + '/assignment/:id')
  .get(assignment.getAssignment)
  .delete(assignment.deleteAssignment);
```
 - b. On utilise [Mongoose](#) qui permet la validation par schéma et la génération automatique de requêtes.

Le module HttpClient

Le module HttpClient va permettre de faire des appels Ajax vers des Web Services...



```
TS app.module.ts ●  app.component.html # assignments.component.css
src > app > TS app.module.ts > AppModule
25 import { RouterModule, Routes } from '@angular/router';
26 import { EditAssignmentComponent } from './assignments/edit-ass
27 import { HttpClientModule } from '@angular/common/http'; [red box]
28
29 > const routes : Routes = [ ...
42 ];
43
44 @NgModule({
45 > declarations: [...],
46   imports: [
47     BrowserModule,
48     BrowserAnimationsModule,
49     MatButtonModule, MatIconModule, MatDividerModule,
50     MatInputModule, MatFormFieldModule,
51     MatDatepickerModule, MatNativeDateModule,
52     MatListModule, MatCardModule, MatCheckboxModule,
53     MatSlideToggleModule,
54     FormsModule, HttpClientModule, [red box]
55     RouterModule.forRoot(routes)
56   ],
57   providers: []
58 })
59 
```

On l'importe dans
app.modules.ts

On l'injecte dans les composants ou services qui en ont besoin

```
constructor(private loggingService:LoggingService,  
           private http:HttpClient) {}
```

On utilise ses méthodes get, post, put delete etc.

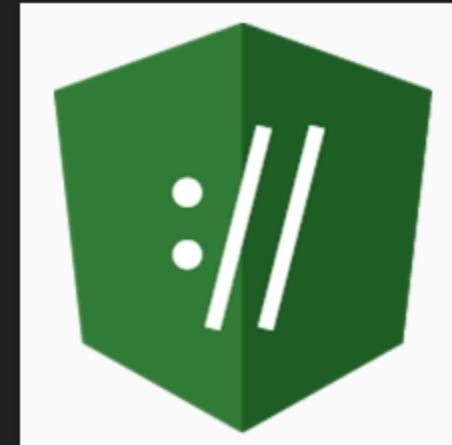
```
getAssignment(...): Observable<Assignment> {  
  this.http.  
    return of( { * get (method) HttpClient.get(url: string, ...  
  }  
    }  
    { * post  
    } * request  
  addAssignment(assignment: Assignment): Observable<string> {
```

Quelques mots encore sur RxJS...

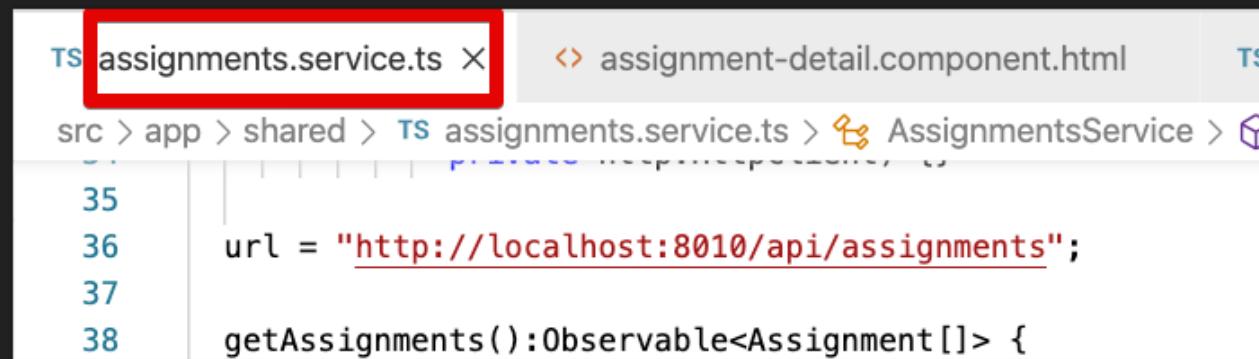
Très utilisé quand on programme en Angular,

On l'a vu : fournir les de quoi manipuler des Observables,

Mais aussi très utilisé avec des requêtes asynchrones,
notamment avec HttpClient

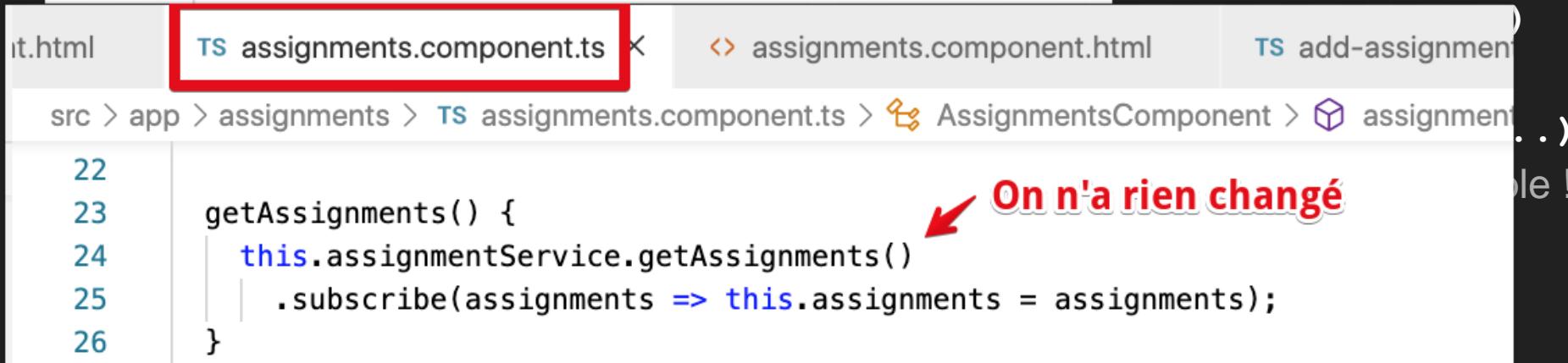


Ajout de HTTP GET dans le service assignments



```
TS assignments.service.ts × < assignment-detail.component.html TS
src > app > shared > TS assignments.service.ts > ⚙ AssignmentsService > 📂
35
36     url = "http://localhost:8010/api/assignments";
37
38     getAssignments(): Observable<Assignment[]> {
```

Et pas besoin de modifier le composant qui utilise



```
it.html TS assignments.component.ts × < assignments.component.html TS add-assignment...
src > app > assignments > TS assignments.component.ts > ⚙ AssignmentsComponent > 📂 assignments...
22
23     getAssignments() {
24         this.assignmentService.getAssignments()
25             .subscribe(assignments => this.assignments = assignments);
26     }

```

On n'a rien changé

Tester l'application

Application de gestion des devoirs à rendre (Assignments)

Le devoir TP Angular avec Mongo Cloud n'a pas été rendu.

Network

XHR

assignments

```
{ "_id": "5fc8b6dfc292ce3b681092e3", "id": 1, "nom": "TP Angular avec Mongo Cloud", "dateDerendu": "2020-01-02", "rendu": false, "_id": "5fc8b6dfc292ce3b681092e3"}
```

Ok, mais si on clique pour avoir de détail ça ne marche pas...

A screenshot of a web browser window. The address bar at the top shows the URL `localhost:4200/assignment/1`. Below the address bar, there is a navigation bar with icons for back, forward, search, and home, followed by a list of recent tabs. The main content area displays a card with the following information:

Application de gestion des devoirs à rendre (Assignments)

TP1 sur WebComponents, un lecteur audio amélioré

Jan 17, 2020

Edit **DELETE**

A red arrow points from the text "Pas le bon titre...." to the title "TP1 sur WebComponents, un lecteur audio amélioré".

Exercice : à vous de faire marcher cela !

Correction:

```
TS assignments.service.ts ×  <> assignment-detail.component.html  TS assi
src > app > shared > TS assignments.service.ts > 🛡 AssignmentsService
35
36   url = "http://localhost:8010/api/assignments";
37
38 >   getAssignments():Observable<Assignment[]> { ...
41   }
42
43   // returns as an Observable the assignment that matches
44   // the id passed as parameter
45   getAssignment(id) : Observable<Assignment> {
46     //return of(this.assignments.find(a => a.id === id));
47     return this.http.get<Assignment>(this.url + "/" + id);
48   }
49
```

Requête

Nous allons
HTTP POST

Et comment

```
JS server.js  X  {} package.json  JS assignments.js  JS
JS server.js > ...
42
43 | // les routes
44 | const prefix = '/api';
45 |
46 | app.route(prefix + '/assignments')
47 |   .get(assignment.getAssignments);
48 |
49 | app.route(prefix + '/assignments/:id')
50 |   .get(assignment.getAssignment)
51 |   .delete(assignment.deleteAssignment);
52 |
53 |
54 | app.route(prefix + '/assignments')
55 |   .post(assignment.postAssignment)
56 |   .put(assignment.updateAssignment);
57 |
```

requête

Ajout d'un assignment (POST)

The screenshot shows a browser window with the title "Application de gestion des devoirs à rendre (Assignments)". Below the title, there is a message: "Le devoir TP Angular avec Mongo Cloud n'a pas été rendu." followed by a list of assignments:

- Le devoir toto n'a pas été rendu.
Dec 16, 2020
- Le devoir fdfdsfdf n'a pas été rendu.
Dec 18, 2020
- Le devoir ffdsf n'a pas été rendu.
Dec 15, 2020
- Le devoir fsdfsdfsd n'a pas été rendu.
Dec 22, 2020
- Le devoir Nouveau TP à rendre sur HttpClient n'a pas été rendu.

On the left, a code editor shows the file "assignments.service.ts". The code contains a function "addAssignment" which logs the assignment to the console and returns the assignment object. A red box highlights the return statement "return this.h" at line 55.

```
TS assignments.service.ts X
src > app > shared > TS assi
49
50     addAssignment(a
51         // this.assignm
52         // this.loggi
53
54         //return of('
55         return this.h
56     }
57
30     .subs
31     }
32     }

Le devoir TP Angular avec Mongo Cloud n'a pas été rendu.

Le devoir toto n'a pas été rendu.
Dec 16, 2020

Le devoir fdfdsfdf n'a pas été rendu.
Dec 18, 2020

Le devoir ffdsf n'a pas été rendu.
Dec 15, 2020

Le devoir fsdfsdfsd n'a pas été rendu.
Dec 22, 2020

Le devoir Nouveau TP à rendre sur HttpClient n'a pas été rendu.

Angular is running in development mode. Call enableProdMode() to enable production mode.
[WDS] Live Reloading enabled.
▶ {message: "fsdfsdfsd saved!"}
▶ {message: "Nouveau TP à rendre sur HttpClient saved!"}
```

Modification d'un assignment (PUT)

The screenshot shows a code editor with several tabs open. The main tab is `assignment-detail.component.html`, which contains the following code:

```
src > app > assignments > edit-assignment > edit-assignment.component.ts > ...  
59     30     }  
60     31     onAssignmentChange = (assignment: Assignment) => {  
61         32     this.assignment = assignment;  
62         33     this.assignment.nom = assignment.nom;  
63         34     this.assignment.dateDeRendu = assignment.dateDeRendu;  
64         35     this.assignment.id = assignment.id;  
65     Pourquoi on a changé cela?  
66         36     this.assignmentService.updateAssignment(this.assignment);  
67         37     .subscribe(message => {  
68             38     console.log(message);  
69             39     // navigation vers la home page uniquement après que la modification ait été effectuée  
70             40     this.router.navigate(['/home']);  
71             41     // ou pourquoi pas ['/home' + this.assignment.id] ?  
72         42     });  
73     };  
74 };
```

A red arrow points from the text "Pourquoi on a changé cela?" to the line `this.router.navigate(['/home']);`. A red box highlights the same line, along with the line above it (`// navigation vers la home page uniquement après que la modification ait été effectuée`) and the line below it (`// ou pourquoi pas ['/home' + this.assignment.id] ?`).

Suppression d'un assignment (DELETE)

The screenshot shows a browser window with two tabs open. The active tab is titled "Assignment de gestion des devoirs à rendre (Assignments)" and displays a list of assignments with their status and creation date. Below the list, there is a message indicating a successful deletion. The second tab, "assignment.model.ts", shows the corresponding TypeScript code for managing assignments.

assignment.model.ts

```
src > app > ass
1  export
2    // 
3    // 
4    // 
5    -i
6    id
7    no
8    da
9    re
10 }
```

assignment.ts

```
src > app
43
44
45
46
47
48
49
50
51
52
53
54
```

Console Output:

```
Angular is running in development mode. Call enableProdMode() to enable production mode.
[WDS] Live Reloading enabled.
▶ {message: "toto deleted"}
▶ {message: "rhaaaaaaaa deleted"}
▶ {message: "ffdsf deleted"}
▶ {message: "fsdfsdfsd deleted"}
▶ {message: "dfsfd deleted"}
▶ {message: "rr deleted"}
```

Page Content:

Devoir intitulé TP Angular avec Mongo Cloud a été rendu.
Dec 22, 2020
Devoir intitulé Nouveau TP à rendre sur HttpClient a été rendu.

Opérateurs fournis par le module RxJS

- **Map**
 - Semblable à JavaScript. Permet d'appliquer une fonction sur chaque élément d'une collection associée à un Observable.
- **Pipe**
 - Permet de “chaîner plusieurs fonctions” pour traiter un Observable avant de le retourner.
- **catchError**
 - Intercepte les Observables qui ont “échoué”, qui ont provoqué une erreur (mauvaise requête, URI, permissions, etc.).
- **Tap**
 - Utile pour débugger les Observables.
- **Filter, pair,**

Exemple d'utilisation des opérateurs “pipe” et “map”

The screenshot shows a web browser window with the URL `localhost:4200/assignment/1`. The title bar reads "Assignment detail". The page content is titled "Application de gestion des devoirs à rendre (Assignments)". Below this, there is a card for an assignment with the text "TP Angular avec Mongo Cloud transformé avec un pipe...." which is highlighted with a red rectangle. There are two buttons: "Edit" and "DELETE". At the bottom of the page, there is a navigation bar with a red tab labeled "DÉCOUVERTE PRACTIQUE" and a page number "7".

On peut mettre plusieurs fonctions dans pipe. Ici on utilise l'opérateur tap pour débugger...

The screenshot shows a browser window with four tabs at the top: `assignments.service.ts`, `assignment-detail.component.ts`, `assignment-detail.component.html`, and `auth.guard`. The main content area displays a heading "Application de gestion des devoirs à rendre (Assignments)" and a card for an assignment with the title "TP Angular avec Mongo Cloud reçu et transformé avec un pipe....". Below the card are "Edit" and "DELETE" buttons. At the bottom, the developer tools' "Console" tab is selected, showing the following log output:

```
Angular is running in development mode. Call enableProdMode() to enable production mode.
tap: assignment avec id = 1 requête GET envoyée sur MongoDB cloud
assignment.nom = TP Angular avec Mongo Cloud reçu et transformé avec un pipe....
[WDS] Live Reloading enabled.
```

The message "tap: assignment avec id = 1 requête GET envoyée sur MongoDB cloud" is highlighted with a red box.

Gestion des erreurs: utilisation de l'opérateur `catchError` dans le `pipe`

The screenshot shows a web browser window with the URL `localhost:4200/assignment/1`. The page title is Application de gestion des devoirs à rendre (Assignments). On the right, there is a `Login` button. Below the page content, the browser's developer tools are visible, specifically the `Console` tab. The console output includes:

- An Angular development mode message: `Angular is running in development mode. Call enableProdMode() to enable production mode.` (core.js:27591)
- An error message from `assignments.service.ts:61`:

```
HttpErrorResponse {headers: HttpHeaders, status: 0, statusText: "Unknown Error", url: "http://localhost:8010/api/assignments/1", ok: false, ...}
```
- A red-highlighted error message from `assignments.service.ts:62`:

```
getAssignment(id=1) a échoué Http failure response for http://localhost:8010/api/assignments/1: 0 assignments.service.ts:62 Unknown Error
```
- An error message from `zone-evergreen.js:2845`:

```
GET http://localhost:8010/api/assignments/1 net::ERR_CONNECTION_REFUSED zone-evergreen.js:2845
```
- A status message at the bottom: `[WDS] Live Reloading enabled.` (client:52)

Ici code nécessaire (pour le catchError, à copier/coller

La méthode à ajouter dans le service :

```
private handleError<T>(operation: any, result?: T) {  
    return (error: any): Observable<T> => {  
        console.log(error); // pour afficher dans la console  
        console.log(operation + ' a échoué ' + error.message);  
  
        return of(result as T);  
    }  
};
```

Et la ligne **catchError** à ajouter dans le **pipe(....)**

```
catchError(this.handleError<any>('### catchError: getAssignments by id avec id=' + id))
```

Complément sur HttpClient

TS assignments.service.ts X

TS assignment-detail.component.ts

↳ assignment-

src > app > shared > TS assignments.service.ts > 🛡 AssignmentsService

```
1 import { HttpClient, HttpHeaders } from '@angular/common/http';
2 import { Injectable } from '@angular/core';
3 import { Observable, of } from 'rxjs';
4 import { Assignment } from '../assignments/assignment.model';
5 import { LoggingService } from './logging.service';
6 import { catchError, map, tap} from 'rxjs/operators';
```

es GET, PUT, POST,

comme application/json,

TS assignments.service.ts X

TS assignment-detail.component.ts

↳ assignment-detail.componen

src > app > shared > TS assignments.service.ts > 🛡 AssignmentsService > 🕒 updateAssignment

```
52
53     addAssignment(assignment: Assignment): Observable<any> {
54         return this.http.post<Assignment>(this.url, assignment, this.HttpOptions);
55     }
56 }
```

Pagination avec Mongoose

Première étape : peupler la Base avec
un grand nombre d'assignments

Premier travail : peupler la BD avec quelques centaines de données

Travail à faire :

1. Utiliser un des sites de génération de données dans le document qui propose [les meilleurs outils pour le développeur Web](#) (chercher “générer des données de test en JSON” dans le doc)
2. Récupérer les données JSON pour 500 assignments, les mettre dans un fichier data.ts par exemple, dans le répertoire “shared”,
3. Modifier le service de gestion des assignments :
 - a. Importer le fichier data.ts,
 - b. Ajouter une méthode “peuplerBD()” qui va insérer les 500 assignments,
 - c. Ajouter dans le composant principal app.component, un bouton “PeuplerBD” qui appelle la méthode précédente du service.
4. Vérifier sur mongodb.com que ça marche, que la base a bien été peuplée...

Importer les données json

Il est très facile de transformer les données JSON en variable exportée depuis un fichier **data.ts** :

```
const bdInitialAssignments = [  
  { id: 1, nom: 'Tresom', dateDeRendu: '5/5/2020', rendu: true },  
  { id: 2, nom: 'Zathin', dateDeRendu: '4/27/2021', rendu: true }  
]  
...  
  
export { bdInitialAssignments };
```

Et l'import dans le service :

```
import { bdInitialAssignments } from './data';
```

Version 1 du code pour peupler la base

Version simple et naïve (on ne peut pas être prévenu de quand l'opération est terminée) :

```
peuplerBD() {
    bdInitialAssignments.forEach(a => {
        let nouvelAssignment = new Assignment();
        nouvelAssignment.nom = a.nom;
        nouvelAssignment.id = a.id;
        nouvelAssignment.dateDeRendu = new Date(a.dateDeRendu);
        nouvelAssignment.rendu = a.rendu;

        this.addAssignment(nouvelAssignment)
        .subscribe(reponse => {
            console.log(reponse.message);
        })
    })
}
```

Version 2 du code pour peupler la base

Version permettant d'appeler un subscribe une fois que tous les inserts ont été effectués (utilisation de l'opérateur **forkJoin** de rxjs) :

```
peuplerBDWithForkJoin(): Observable<any> {
  const appelsVersAddAssignment: any = [];

  bdInitialAssignments.forEach((a) => {
    const nouvelAssignment: any = new Assignment();

    nouvelAssignment.id = a.id;
    nouvelAssignment.nom = a.nom;
    nouvelAssignment.dateDeRendu = new Date(a.dateDeRendu);
    nouvelAssignment.rendu = a.rendu;

    appelsVersAddAssignment.push(this.addAssignment(nouvelAssignment));
  });

  return forkJoin(appelsVersAddAssignment); // renvoie un seul Observable
  pour dire que c'est fini
}
```

```
// dans le composant principal
peuplerBD() {
  // version naïve et simple
  //this.assignmentsService.peuplerBD();

  // meilleure version :
  this.assignmentsService.peuplerBDWithForkJoin()
    .subscribe(() => {
      console.log("LA BD A ETE PEUPLEE, TOUS LES ASSIGNMENTS
AJOUTES,
ON RE-AFFICHE LA LISTE");
      // replaceUrl = true = force le refresh, même
      si
        // on est déjà sur la page d'accueil
      // Marche plus avec la dernière version d'angular
      this.router.navigate(["/home"], {replaceUrl: true});
    })
}
```

Deuxième étape : modification du back-end pour gérer la pagination

On va modifier le back-end pour utiliser la pagination

On va utiliser le plugin Mongoose [Aggregate Paginate V2](#) (faites `npm install --save mongoose-aggregate-paginate-v2`) dans le dossier `api` pour ajouter le plugin au projet!

Très peu de modifications sont requises.

Dans la déclaration du modèle Mongoose :
deux lignes à ajouter !

```
let mongoose = require("mongoose");
var aggregatePaginate = require("mongoose-aggregate-paginate-v2");
let Schema = mongoose.Schema;

let AssignmentSchema = Schema({
  id: Number,
  dateDeRendu: Date,
  nom: String,
  rendu: Boolean,
});
AssignmentSchema.plugin(aggregatePaginate);
// C'est à travers ce modèle Mongoose qu'on pourra faire le CRUD
module.exports = mongoose.model("Assignment", AssignmentSchema);
```

On va modifier le back-end pour utiliser la pagination

Et on modifie l'implémentation de la route pour obtenir plusieurs assignments :

```
// Récupérer tous les assignments (GET)
function getAssignments(req, res) {
  var aggregateQuery = Assignment.aggregate();
  Assignment.aggregatePaginate(aggregateQuery,
    {
      page: parseInt(req.query.page) || 1,
      limit: parseInt(req.query.limit) || 10,
    },
    (err, assignments) => {
      if (err) {
        res.send(err);
      }
      res.send(assignments);
    }
  );
}
```

On va devoir passer en paramètres dans l'URL, sous forme de “queries”, la **page** demandée et le nombre d'assignments demandés (**limit**)

Une requête envoyée sur
<http://localhost:8010/api/assignments> renvoie
maintenant

```
localhost:8010/api/assignments
{
  + docs: [...],
  totalDocs: 500,
  limit: 10,
  page: 1,
  totalPages: 50,
  pagingCounter: 1,
  hasPrevPage: false,
  hasNextPage: true,
  prevPage: null,
  nextPage: 2
}
```

```
localhost:8010/api/assignments?page=10&limit=3
{
  - docs: [
    - {
      _id: "5ff2f6da43585509586fdfce",
      id: 38890,
      nom: "Mertensia campanulata A. Nelson",
      dateDeRendu: "2020-10-28T17:47:47.000Z",
      rendu: false,
      __v: 0
    },
    - {
      _id: "5ff2f6da43585509586fdfcf",
      id: 90540,
      nom: "Oxytropis lambertii Pursh var. articulata (Greene) Barneby",
      dateDeRendu: "2020-08-24T16:09:15.000Z",
      rendu: false,
      __v: 0
    },
    - {
      _id: "5ff2f6da43585509586fdfd0",
      id: 74113,
      nom: "Calamagrostis avenoides (Hook. f.) Cockayne",
      dateDeRendu: "2020-10-29T22:00:26.000Z",
      rendu: false,
      __v: 0
    }
  ],
  totalDocs: 500,
  limit: 3,
  page: 10,
  totalPages: 167,
  pagingCounter: 28,
  hasPrevPage: true,
  hasNextPage: true,
  prevPage: 9,
  nextPage: 11
}
```

Modification du composant angular qui affiche la liste des assignments

```
export class AssignmentsComponent implements OnInit
{
  page: number=1;
  limit: number=10;
  totalDocs: number;
  totalPages: number;
  hasPrevPage: boolean;
  prevPage: number;
  hasNextPage: boolean;
  nextPage: number;
  ...
  ...
}
```

```
ngOnInit() {
  this.assignmentsService.getAssignmentsPagine(this.page, this.limit)
    .subscribe(data => {
      this.assignments = data.docs;
      this.page = data.page;
      this.limit = data.limit;
      this.totalDocs = data.totalDocs;
      this.totalPages = data.totalPages;
      this.hasPrevPage = data.hasPrevPage;
      this.prevPage = data.prevPage;
      this.hasNextPage = data.hasNextPage;
      this.nextPage = data.nextPage;
      console.log("données reçues");
    });
}
```

Exercice à faire : faire une GUI de pagination

Dans un premier temps, juste avec des boutons page précédente, page suivante, première page, dernière page, etc. afficher le nombre de documents total, de page total etc.

Modifiez le template du composant qui affiche la liste des assignments.

Dans un second temps :

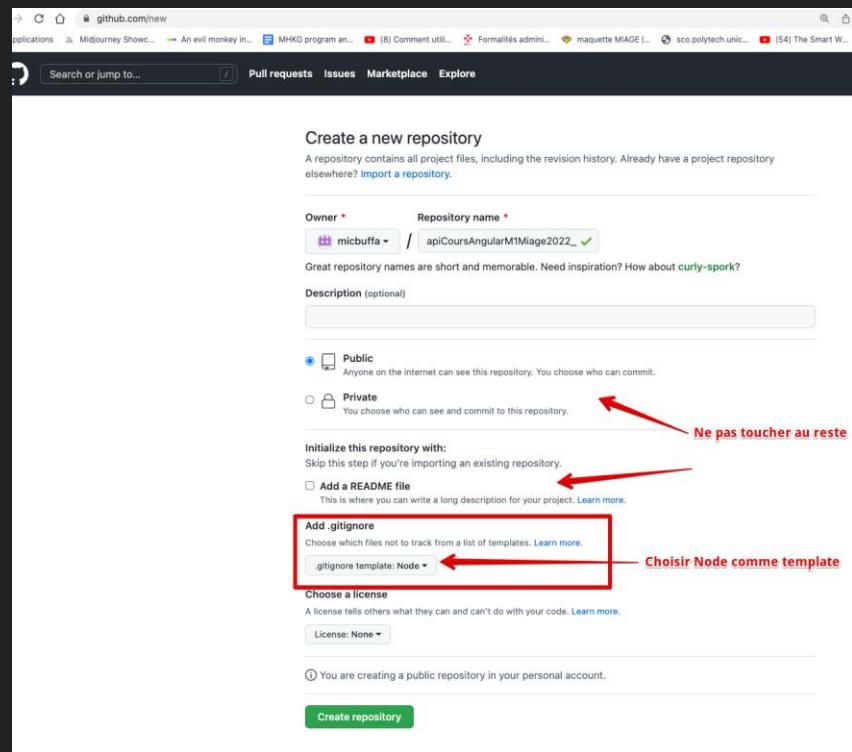
- Mettre en grisé (disabled) les boutons si on est sur la première ou dernière page
- Regarder dans Angular Material les widgets de pagination et essayer de les utiliser

Hébergement sur Heroku

Première étape : hébergement du back end

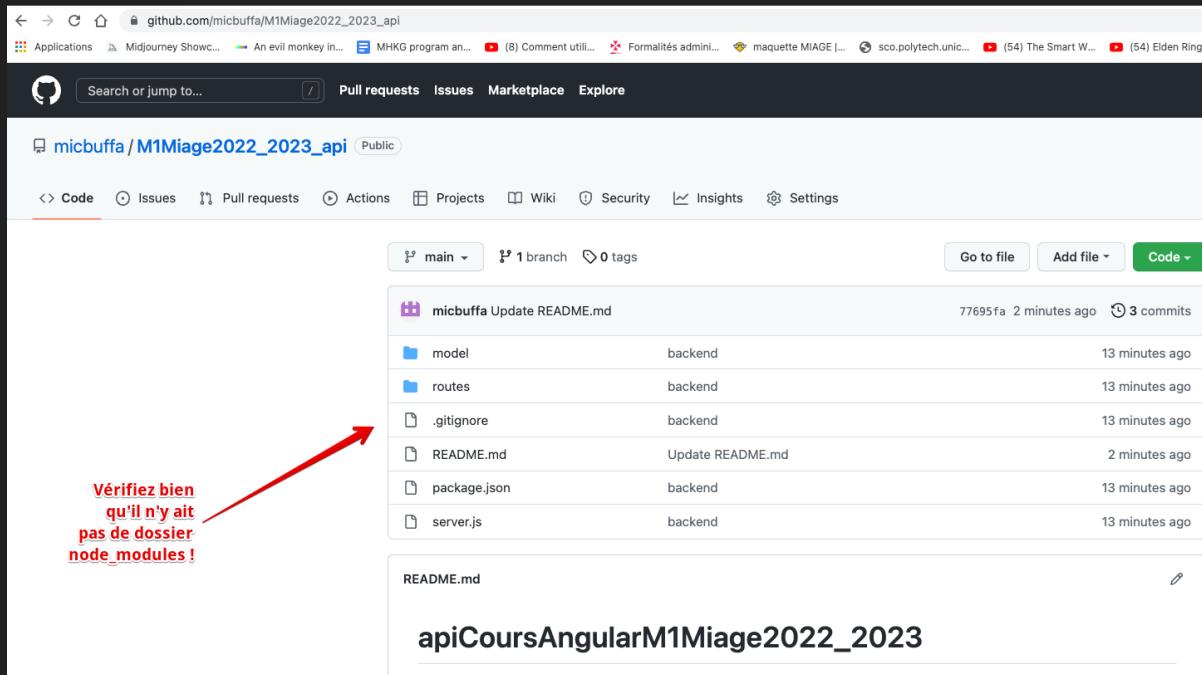
Mettre le projet backend sur gothub : méthode la plus simple..

1. Créer un nouveau repository github,
choisir comme template pour le .gitignore
“Node” : ça évitera de mettre le dossier
node_modules sur github.
2. faire **git clone** du repo dans le dossier
AU DESSUS de assignment_app
3. Copier dedans les sources du projet api
4. Supprimer le dossier api
5. renommer le repo du backend en api



Première étape : hébergement du back end

Vérifiez ensuite que votre projet est bien sur le repository github !



Hébergement du back-end sur Heroku (suite)

Il faut savoir que quand on déploie une application sur Heroku, ce dernier exécute :

1. npm install
2. npm run build
3. npm run start

Vous devrez donc tester ces commandes localement sur votre machine pour vérifier que cela fonctionne. Bien regarder les scripts du fichier package.json et vérifier que tous les packages npm sont dans les dépendances et s'assurer qu'il y a :

```
"scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1",  
    "start": "node server.js"  
}, You, 4 days ago • Initial version
```

Hébergement du back-end sur Heroku (suite)

Pour être bien sûr :

1. Effacer le dossier node_modules
2. Faire
 - a. `npm install`
 - b. `npm run build` (s'il y a une erreur ce n'est pas grave), le backend n'a pas besoin d'être buildé)
 - c. `npm run start`
3. L'application node doit s'exécuter correctement !
4. Une fois cela fait on peut créer une application sur Heroku et la lier au repo github

Ajout d'une application (1)

The screenshot shows the Heroku dashboard at dashboard.heroku.com/apps. The top navigation bar includes links for Applications, COUPONS Bangga..., Ma sélection de jo..., Es-tu capable de r..., Un jour en France..., Discussion | HTM..., Autres favoris, and Liste de lecture. Below the navigation is a search bar and a favorites icon. A sidebar on the left shows 'Personal' apps: angular-emsi-mbds-2021-2022 and api-mbds-2021-2022, each with a star icon. The main area features a 'Jump to Favorites, Apps, Pipelines, Spaces...' button. On the right, there are two buttons: 'New' (with a dropdown arrow) and 'Create new app' (which is highlighted with a red box). Below these are 'Create new pipeline' and another 'New' button.

dashboard.heroku.com/apps

Applications COUPONS Bangga... Ma sélection de jo... Es-tu capable de r... Un jour en France... Discussion | HTM... Autres favoris Liste de lecture

Salesforce Platform

HEROKU Jump to Favorites, Apps, Pipelines, Spaces...

Personal

Filter apps and pipelines

angular-emsi-mbds-2021-2022

api-mbds-2021-2022

New

Create new app

Create new pipeline

Ajout d'une application (2)

Create New App

App name **nom libre mais sans majuscules**

api-cours-angular2022

api-cours-angular2022 is available

Choose a region

Europe

Add to pipeline...

Create app

The screenshot shows a 'Create New App' form. At the top, it says 'Create New App'. Below that, there's a 'App name' field containing 'api-cours-angular2022', which is highlighted with a red box and has a red arrow pointing to the right with the text 'nom libre mais sans majuscules' above it. Below the name, it says 'api-cours-angular2022 is available'. Under 'Choose a region', there's a dropdown menu with 'Europe' selected, also highlighted with a red box. At the bottom, there's a 'Create app' button, which is also highlighted with a red box.

Ajout d'une application (3)

Deployment method

 Heroku Git
Use Heroku CLI

 GitHub
Connect to GitHub

 Container Registry
Use Heroku CLI

Connect to GitHub

Connect this app to GitHub to enable code diffs and deploys.

Search for a repository to connect to

 micbuffa 

Missing a GitHub organization? [Ensure Heroku Dashboard has team access.](#)

 micbuffa/apiPourCoursAngular2022 

Ajout d'une application (4)

Quand vous allez cliquer sur “Deploy Branch, cela va faire un git checkout puis npm install, npm run build, npm run start

Ensuite une fenêtre va montrer la progression du déploiement.

Pour voir les logs installer [la ligne de commande Heroku](#) et [lire la doc, section logs](#).

```
heroku logs --app client-intense-angular2022
```

The screenshot shows the Heroku GitHub integration settings. At the top, it says "Enable automatic deploys from GitHub". Below that, a note states: "Every push to the branch you specify here will deploy a new version of this app. Deploys happen automatically: be sure that this branch is always in a deployable state and any tests have passed before you push." A link "Learn more" is provided. A red box highlights this entire section. Below this, there's a "Choose a branch to deploy" dropdown set to "main", with a "Wait for CI to pass before deploy" checkbox (unchecked) and a note about CI configuration. A large red box highlights the "Enable Automatic Deploys" button. Further down, there's a "Deploy a GitHub branch" section with a note about deploying the current state of the specified branch, followed by another "Choose a branch to deploy" dropdown set to "main" and a red-bordered "Deploy Branch" button.

Enable automatic deploys from GitHub

Every push to the branch you specify here will deploy a new version of this app. Deploys happen automatically: be sure that this branch is always in a deployable state and any tests have passed before you push. [Learn more](#).

Choose a branch to deploy

main

Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

Enable Automatic Deploys

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. [Learn more](#).

Choose a branch to deploy

main

Deploy Branch

Vérifiez que cela fonctionne !

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. [Learn more](#).

Choose a branch to deploy

Deploy Branch

Receive code from GitHub

Build main 77695fa1

Release phase

Deploy to Heroku

Your app was successfully deployed.

[View](#)

Cliquer ici ! 



The screenshot shows a browser window displaying a JSON response from the API endpoint `api-cours-angular-2023.herokuapp.com/api/assignments`. The response is a single assignment object:

```
[{"id": 1, "nom": "Devoir dans le cloud", "rendu": false, "dateDeRendu": "2022-10-10T00:00:00.000Z"}]
```

A red arrow points to the URL in the browser's address bar, and the text **Completez l'URL !** is overlaid next to it.

Hébergement du front-end sur Heroku (suite)

Rappel : lors du déploiement, Heroku exécute :

1. npm install
2. npm run build
3. npm run start

Jusqu'à présent on utilisait la commande ng serve, qui correspond au mode "développement" d'Angular.

En réalité cette commande lance un NodeJS qui sert une page index.html générée à la volée....

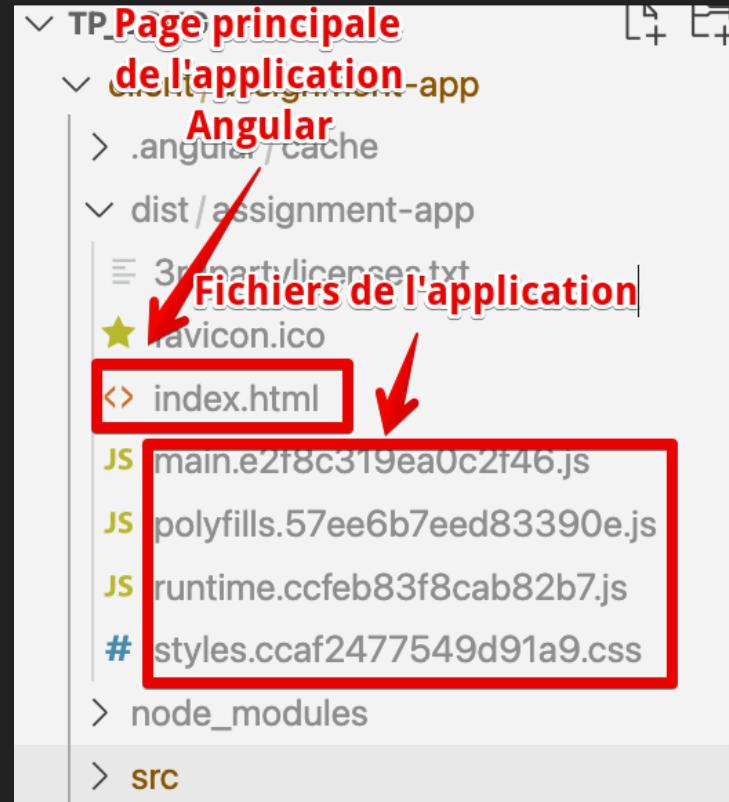
Hébergement du front-end sur Heroku (suite)

Rappel : lors du déploiement, Heroku exécute :

1. `npm install`
2. `npm run build`
3. `npm run start`

Dans notre cas, exécutons manuellement `npm run build` pour voir ce que cela fait....

...Cela crée un dossier dist qui contient une page HTML et des fichiers JS/CSS minifiés. Tout ce dossier correspond à l'application complète en mode “production”.



Ajout d'un mini serveur web sous nodeJS pour servir index.html

Nous allons ajouter un fichier **server.js** à la racine du projet

Il s'agit d'un serveur http minimal codé avec NodeJS et le module **npm express**

Il faut par conséquent, pour que Heroku puisse l'exécuter, modifier package.json :

1. Ajouter le module express dans les dépendances pour que le npm install installe ce module,
2. Personnaliser le script “start” dans le fichier package.json pour qu'il lance “node server.js”

ON VA FAIRE TOUT CELA DANS LE TRANSPARENT SUIVANT !

Ajout d'un mini serveur web sous nodeJS pour servir index.html

1. Faire:

- a. cd à la racine du projet
- b. `npm install --save express`

2. Modifier dans package.json le script start :

```
"scripts": {  
  "ng": "ng",  
  "start": "node server.js", You, 4 days ago • added mi  
  "build": "ng build",  
  "watch": "ng build --watch --configuration development",  
  "test": "ng test"  
},
```

A la racine du projet, créer un fichier `server.js`

Code typique à copier/coller dans ce fichier :

```
//Install express server
const express = require("express");
const path = require("path");

const app = express();

// Serve only the static files form the dist directory
app.use(express.static(__dirname + "/dist/assignment-app/"));

app.get("/*", function (req, res) {
  res.sendFile(path.join(__dirname + "/dist/assignment-app/index.html"));
});

// Start the app by listening on the default Heroku port
app.listen(process.env.PORT || 8081);
```

Puis tester manuellement avant d'essayer de mettre sur Heroku

1. Effacer le dossier `node_modules`
2. Faire `npm install`
3. Faire `npm run build` -> cela doit créer le dossier dist avec l'application dedans
4. Faire `npm run start`
5. Ouvrir <http://localhost:8081> -> cela doit ouvrir la page `dist/index.html` avec l'application dedans... L'appli se lance dans votre navigateur.
6. Penser à changer l'URL dans le service de gestion des assignments pour mettre celui de votre back-end sur Heroku
7. Déployer sur heroku de la même manière qu'on a déployé le back-end

Déploiement du front-end sur Heroku

1. Commitez vos modifications sur le repository github du front-end
2. Vérifier bien que l'URL utilisé dans le service de gestion des assignments est bien celui de votre backend sur heroku.

Par exemple pour moi : <https://api-cours-angular-2023.herokuapp.com/api/assignments>

3. Suivez ensuite les mêmes étapes que pour le déploiement du back end, vous changerez juste le nom du repository github.