

Introduction à Angular, NodeJS, Express, MongoDB, Cloud

Michel Buffa, Septembre 2025

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)
 - Amélioration avec Angular 20 ?
Disponibilité récente d'une extension de navigateur pour debugger!
- TypeScript (ne plait 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...

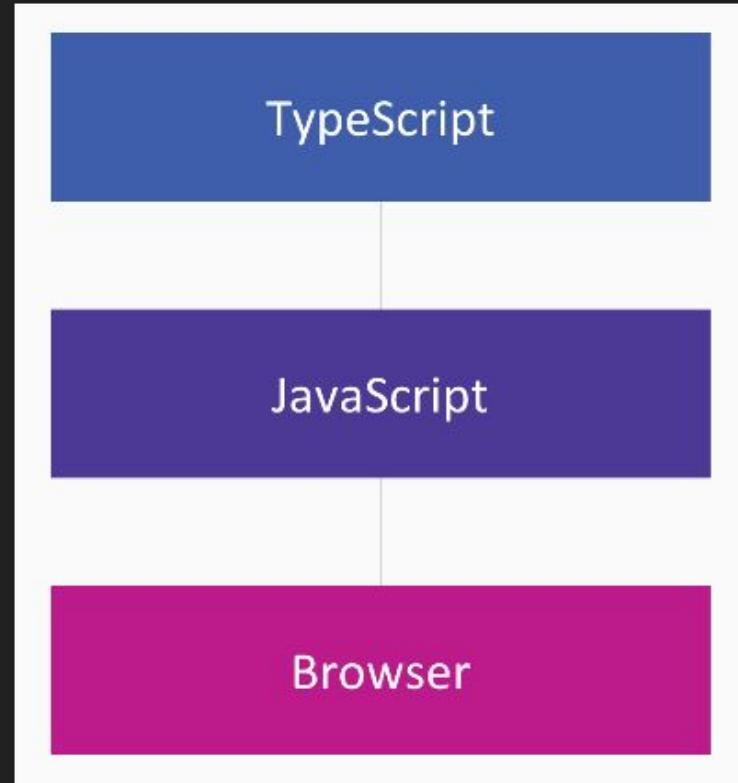
Basé sur le concept de composants

- 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 20 (Mai 2025)

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 20 :

- Changement des règles de nommage des fichiers et de l'architecture en dossier/sous-dossiers conseillées par les bonnes pratiques
- Apparition d'un système de "signaux" pour faciliter les communications entre composants
- Apparition d'une extension de navigateur pour debugger

LE COURS N'EST PAS ENCORE MIS À JOUR, IL LE SERA EN COURS DE
ROUTE....

Nouveautés Angular 19 et 20

- Angular 19: modernisation de certains points:
 - Nouveau logo,
 - Nouvelle plateforme de documentation : <https://angular.dev/> ,
 - *Playground* pour tester son code en ligne : <https://angular.dev/playground> ,
 - Nouvelle façon d'inclure les modules,
 - Nouvelle façon de définir les routes,
 - Nouvelles façons de faire des if et des boucles dans les templates HTML des composants.
- Angular 20 : signaux, etc.
- [Détails des nouveautés d'Angular 18](#)
- [Détails des nouveautés d'Angular 19](#)
- [Détails des nouveautés d'Angular 20](#)

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
- Chaque composant déclare les modules utilisés (composants *standalone*)
- 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 (installez à partir du site officiel)
- Un bon éditeur de code source (recommandé : Visual Studio Code (avec copilot) ou WebStorm ou équivalent), installez les extensions proposées pour Angular,
- SAVOIR QU'ON NE DÉVELOPPE PAS UNE APPLI SANS AVOIR TOUT LE TEMPS LE DEBUGGER DU NAVIGATEUR OUVERT
 - **ctrl-shift-i** sous windows ou F12
 - **command-option-i** sous Mac
- ET L'OPTION “disable cache when devtools open” ACTIVÉE !
- Savoir utiliser les devtools (console, network, debugger, inspecter les éléments, etc.)
- Bookmarker les sites <https://angular.dev/>, <https://cli.angular.dev/> et le nouveau <https://angular.dev>

Attention : ne pas prendre la toute dernière version de NodeJS

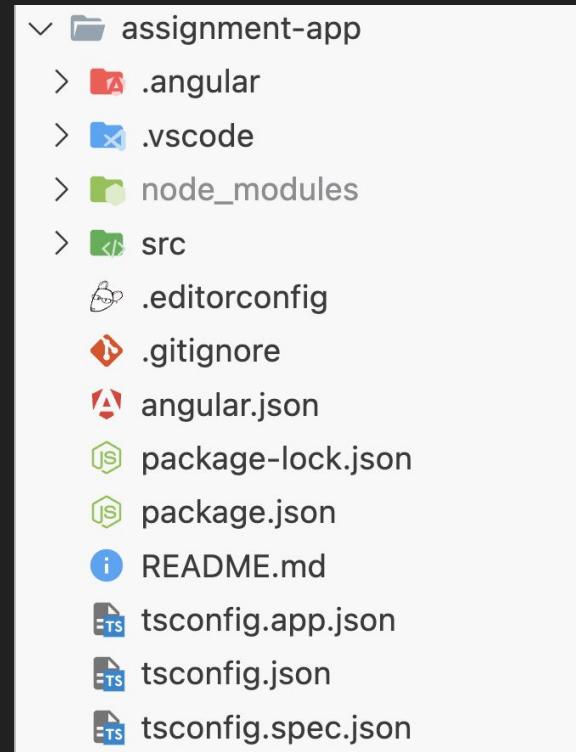
- Je conseille une version stable : prendre celle suggérée sur le site <https://nodejs.org/en/download> (le 12/09/2025 : 22.19.0)
- Si vous êtes sur Mac avec Brew, le gestionnaire de version de node appelé nvm est super pratique !

```
brew install nvm && mkdir -p ~/.nvm
echo 'export NVM_DIR="$HOME/.nvm"' >> ~/.zshrc
echo '[ -s "/opt/homebrew/opt/nvm/nvm.sh" ] && .
"/opt/homebrew/opt/nvm/nvm.sh"' >> ~/.zshrc
source ~/.zshrc
```

```
# installer et utiliser Node 22 (ex.)
nvm install 22
nvm use 22
node -v
```

Allez ! Au travail !

1. Installez angular CLI (en super utilisateur):
 - `npm install -g @angular/cli`
 - i. Ajouter `@20` à la fin pour forcer la version si besoin
 - Note: Être en mode admin ou sudo
 - Répondre aux questions en appuyant sur ENTRÉE (default)
 - Vérifiez la version avec `ng version`
(vous devriez être en 20)
2. Créer un premier projet (**pas** en su !):
 - `ng new assignment-app (pas en admin) SI ERREUR VOIR SLIDE SUIVANT`
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...



Et les assistants IA ?

- C'est quoi cette question lors de la création du projet ?
-

```
~/Documents/Cours/2025-2026/MIAGE/M1/Angular
```

```
ng new assignment-app
```

```
✓ Which stylesheet format would you like to use? CSS [ https://developer.mozilla.org/docs/Web/CSS ]  
✓ Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? No  
✓ Do you want to create a 'zoneless' application without zone.js? No  
? Which AI tools do you want to configure with Angular best practices? https://angular.dev/ai/develop-with-ai  
 None  
 Claude [ https://docs.anthropic.com/en/docs/claude-code/memory ]  
 Cursor [ https://docs.cursor.com/en/context/rules ]  
 Gemini [ https://ai.google.dev/gemini-api/docs ]  
➤  GitHub Copilot [ https://code.visualstudio.com/docs/copilot/copilot-customization ]  
 JetBrains AI [ https://www.jetbrains.com/help/junie/customize-guidelines.html ]  
 Windsurf [ https://docs.windsurf.com/windsurf/cascade/memories#rules ]
```

Et les assistants IA ?

- **ANGULAR TRES BIEN SUPPORTE :** <https://angular.dev/ai/develop-with-ai>
 - Instructions qu'on retrouve dans le projet dans
`.github/copilot-instructions.md`
- Bolt.new, Copilot, Copilot Agent, Claude Code, Windsurf, Cursor, etc. connaissent bien Angular !
- Ils pourront éventuellement vous aider à générer un projet “bootstrap” ou à créer à votre place certaines parties d'un projet.
- Attention aux “trous noirs” de l'IA si vous espérez générer des projets entiers sans écrire du code !
- Je vous conseille dans un premier temps, de suivre ce qui vous est proposé dans ce cours, pour avoir une idée de “comment Angular fonctionne”, ensuite, une fois que vous aurez compris, vous pourrez vous “doper à l'IA”
- Notez que créer un projet entier à partir de prompts est mortellement ennuyeux et demande beaucoup de patience + de l'expertise que vous n'avez pas si vous débutez en Angular.

Si erreurs lors de la création

1. Une erreur classique est que **vous devez supprimer le contenu du dossier \$HOME/.npm**
2. En sudo ou en admin : supprimez le contenu de \$HOME/.npm
3. Recommencez la création du projet
4. Sinon : copiez/collez l'erreur dans chatGPT !

Exécuter le projet !

Dans Visual Studio :

- Activer l'auto-save,
- Lancez les commandes dans les Terminaux !
- Pour lancer : **npm i puis ng serve**
- **N'utilisez pas powershell mais git bash par exemple**
- Si **ng serve** sort une erreur essayer **ng serve --host 0.0.0.0**

The screenshot shows the Visual Studio interface with the package.json file open in the code editor. The file contains the following JSON code:

```
package.json — 2-IntroAngular
{
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "~11.0.0",
    "@angular/common": "~11.0.0",
    "@angular/compiler": "~11.0.0",
    "@angular/core": "~11.0.0",
    "@angular/forms": "~11.0.0",
    "@angular/platform-browser": "~11.0.0"
  }
}
```

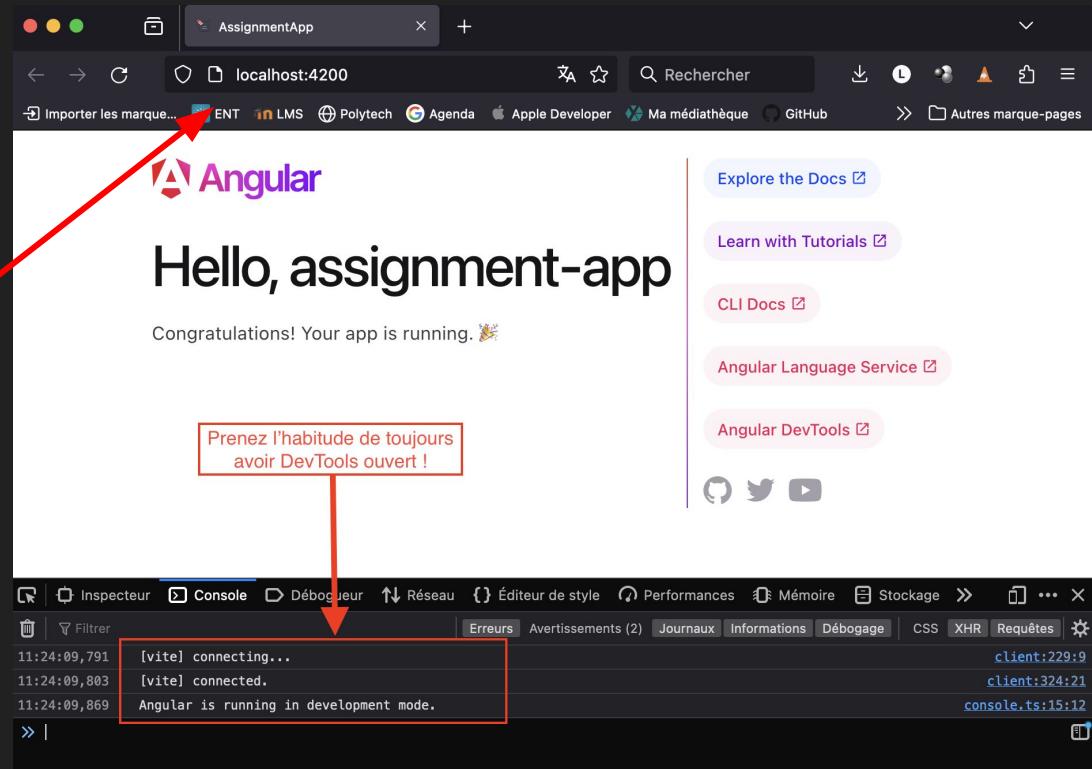
Annotations in red highlight specific parts of the code and terminal:

- "ng serve pour lancer. Laissez le comme ça ce terminal !"
- "La commande "surveille" le répertoire et recompile dès qu'un fichier est modifié"
- A red arrow points from the annotation "La commande "surveille" le répertoire et recompile dès qu'un fichier est modifié" to the "scripts" section of the package.json file.
- A red arrow points from the annotation "ng serve pour lancer. Laissez le comme ça ce terminal !" to the terminal window at the bottom.
- The terminal window shows the command **(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.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 :
https://github.com/micbufffa/AngularFrontEndM1Info2025_2026
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.

Lien GitHub vers le projet fait en classe

M1 MIAGE 2025_2026 :

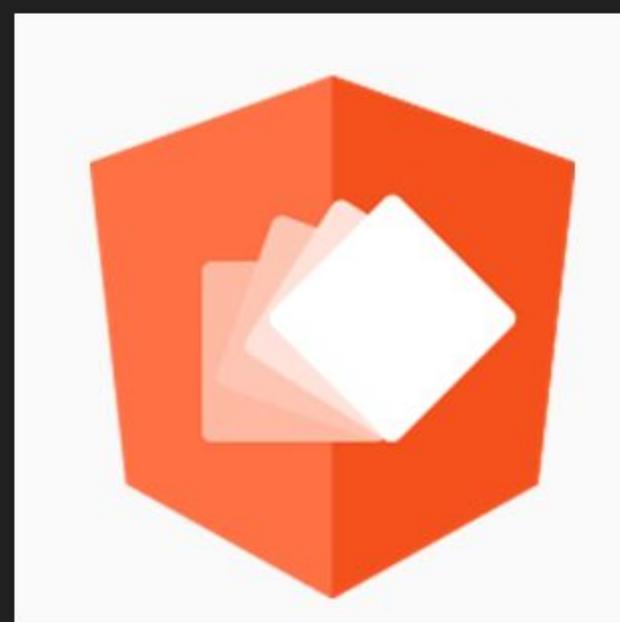
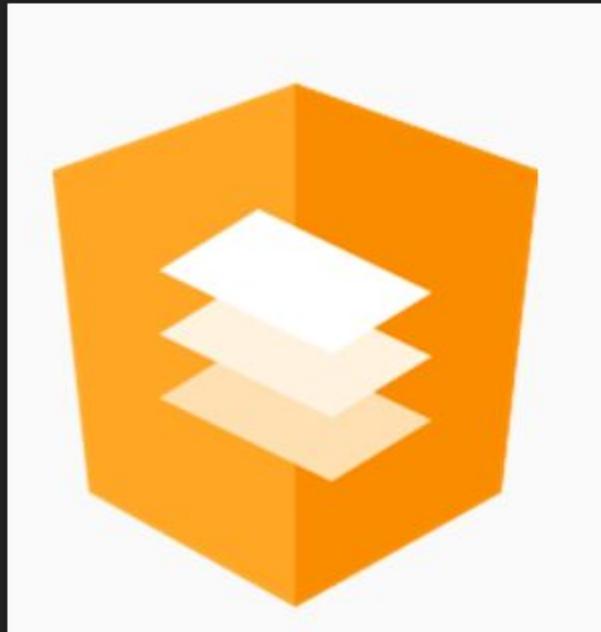
https://github.com/micbuffa/AngularFrontEndM1Info2025_2026

Faire **git clone** de ce repo, puis **npm i**, puis **ng serve**

Plus tard, juste **git pull** et **ng serve**

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 homepage at https://material.angular.io. The page has a purple header with the Angular Material logo and navigation links for Components, CDK, and Guides. A GitHub icon and link are also present. The main title "Angular Material" is displayed prominently in large white font, followed by the subtitle "Material Design components for Angular". A central "Get started" button is visible. Below the purple header, there's a white section with three descriptive cards: "High quality", "Versatile", and "Frictionless", each with a brief subtext.

https://material.angular.io

Importer les marque... ENT LMS Polytech Agenda Apple Developer Ma médiathèque GitHub Autres marque-pages

Material Components CDK Guides 17.0.1 GitHub

Angular Material

Material Design components for Angular

Get started

High quality

Internationalized and

Versatile

Provide tools that help

Frictionless

Built by the Angular team to

Etapes

1. Ouvrir un nouveau terminal dans Visual Studio Code
2. Dans le répertoire du projet, installer 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.

```
"private": true,  
"dependencies": {  
  "@angular/animations": "^17.0.0",  
  "@angular/cdk": "^17.0.1",  
  "@angular/common": "^17.0.0",  
  "@angular/compiler": "^17.0.0",  
  "@angular/core": "^17.0.0",  
  "@angular/forms": "^17.0.0",  
  "@angular/material": "^17.0.1",  
  "@angular/platform-browser": "^17.0.0",  
  "@angular/platform-browser-dynamic": "^17.0.0",  
  "@angular/router": "^17.0.0",  
  "rxjs": "~7.8.0",  
  "tslib": "^2.3.0",  
  "zone.js": "~0.14.2"  
},
```

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 web browser window with the URL material.angular.io/guide/getting-started. The browser's address bar and tab bar are visible at the top. The main content area displays the 'Getting Started' guide for Angular Material. The guide starts with instructions for existing applications, followed by a section titled 'Install Angular Material'. It provides a command-line instruction: `ng add @angular/material`. Below this, 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.' On the right side of the guide, there is a sidebar titled 'Guide Content' with links to 'Install Angular Material' and 'Display a component'.

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:

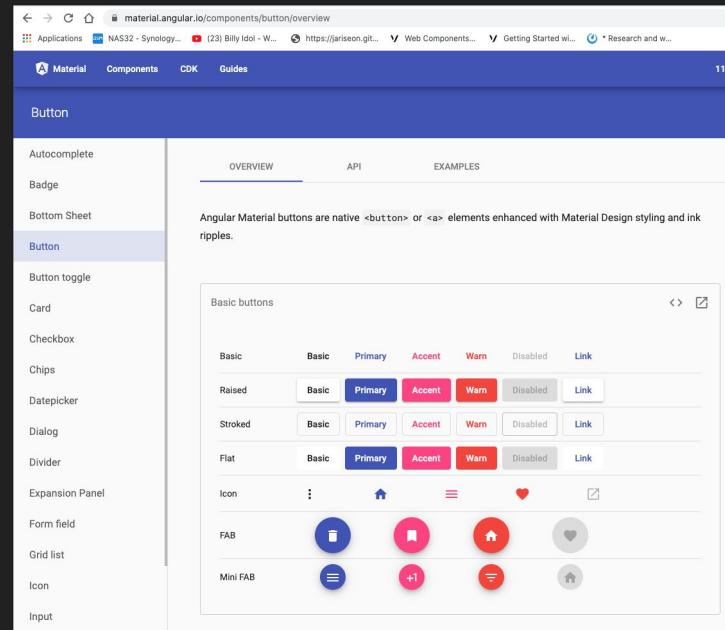
1. Choose a prebuilt theme name, or "custom" for a custom theme:

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

Angular Material : types de composants

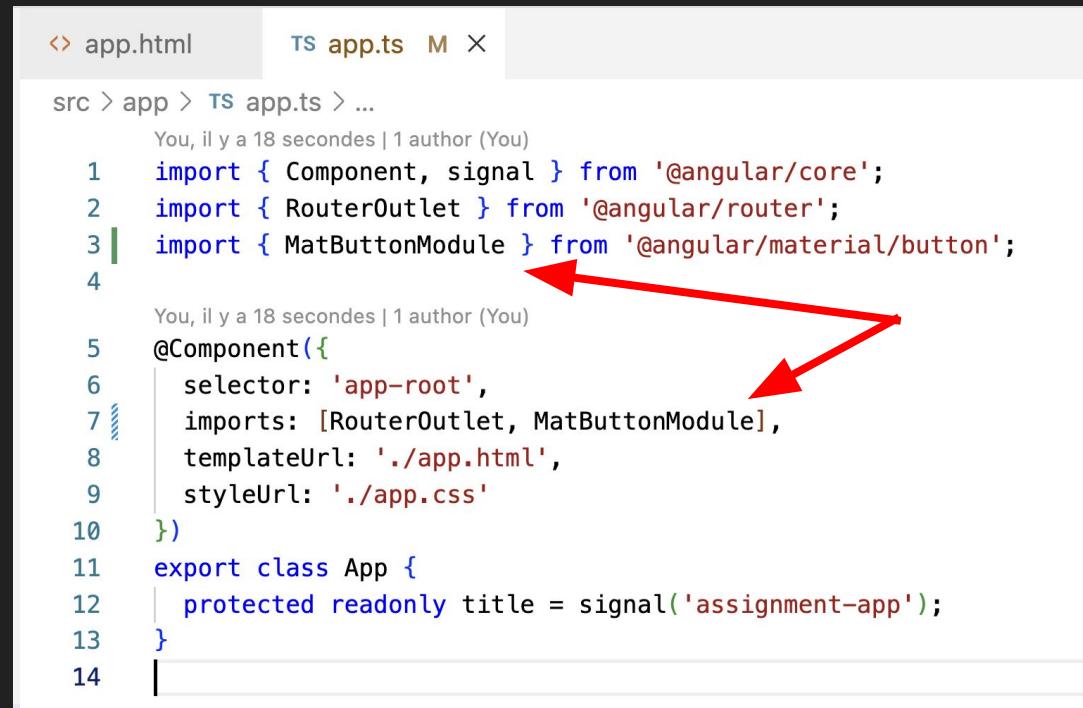
- Formulaires et input fields
- Navigation
- Layout
- Boutons et indicateurs
- Popups et Modals (dialogues bloquants)
- Tables
- Etc.
- Prenez le temps de visiter et tester :

[https://material.angular.dev/
components/categories](https://material.angular.dev/components/categories)



Création d'un premier composant stylé

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



```
app.html      TS app.ts M X
src > app > TS app.ts > ...
You, il y a 18 secondes | 1 author (You)
1 import { Component, signal } from '@angular/core';
2 import { RouterOutlet } from '@angular/router';
3 import { MatButtonModule } from '@angular/material/button'; Two red arrows point to this line.
4 You, il y a 18 secondes | 1 author (You)
5 @Component({
6   selector: 'app-root',
7   imports: [RouterOutlet, MatButtonModule], A red arrow points to this line.
8   templateUrl: './app.html',
9   styleUrls: ['./app.css']
10 })
11 export class App {
12   protected readonly title = signal('assignment-app');
13 }
14 |
```

Création d'un premier composant stylé

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

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

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

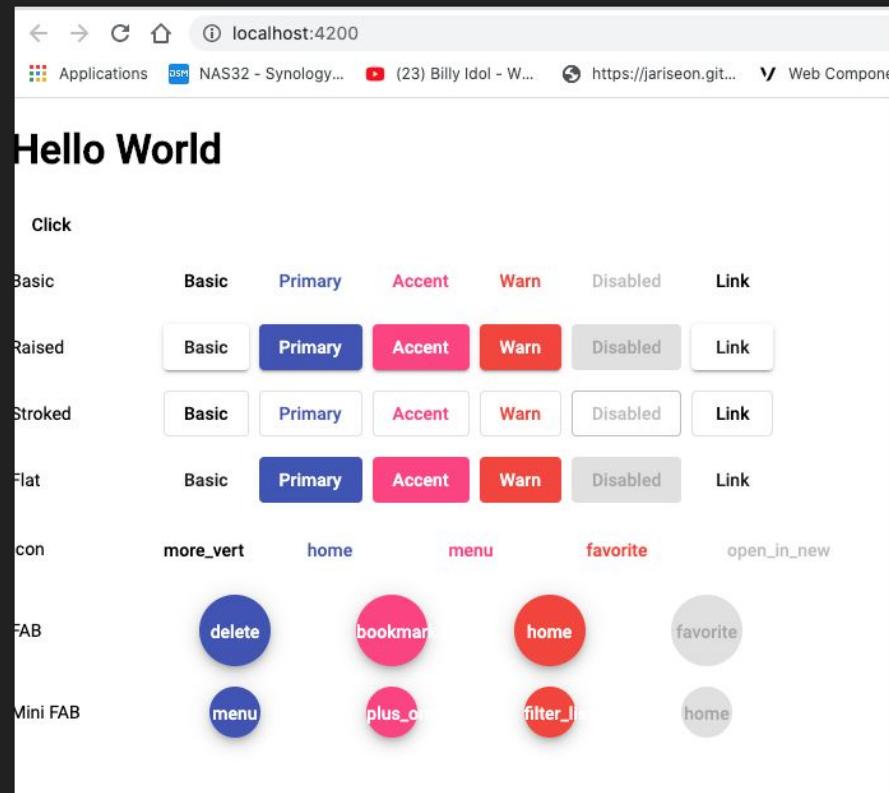
Regardez le résultat, cliquez le bouton, vous devez voir une animation. Si ce n'est pas le cas, relancez `ng serve` ou `ng serve --host 0.0.0.0`

Création d'un premier composant style

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

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

Testez !



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

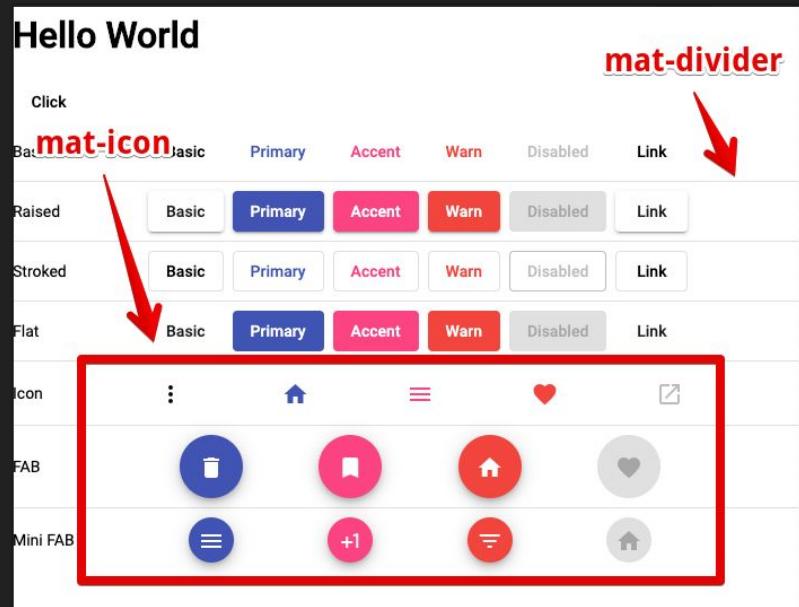
Vous avez bien regardé les erreurs dans le terminal de compilation ?

Dans la console du navigateur ?

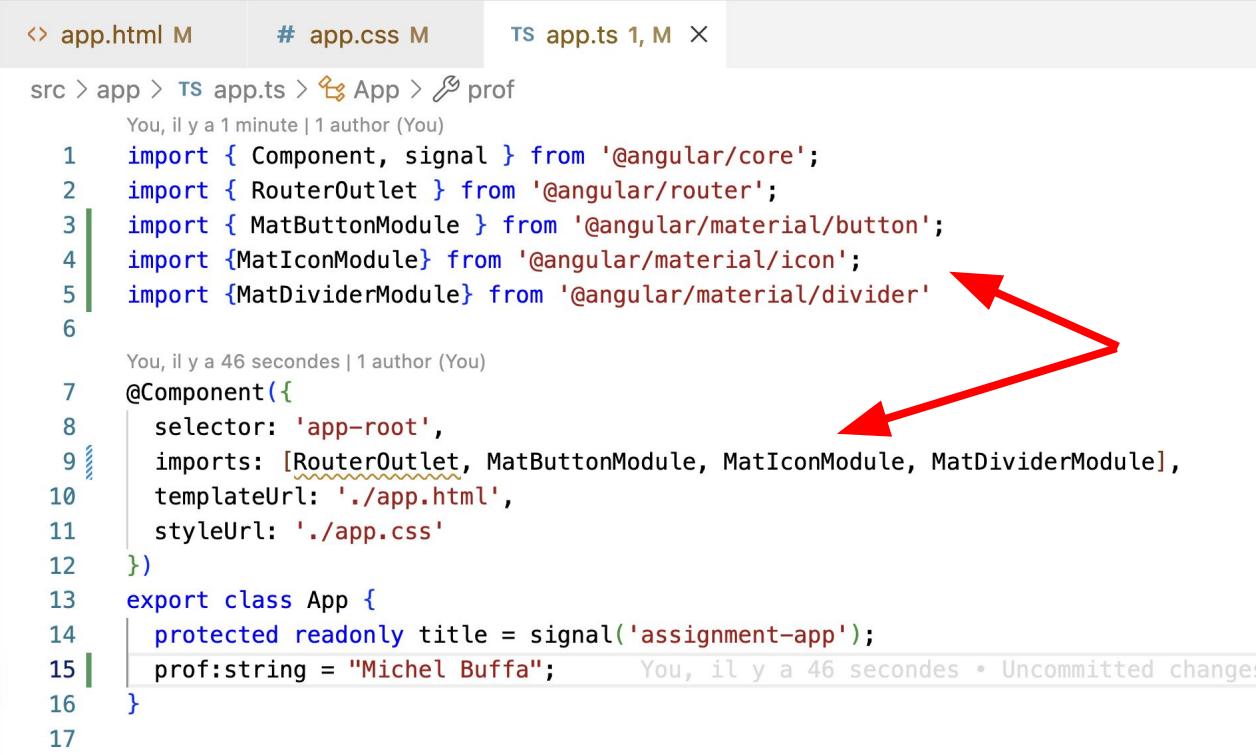
Que s'est-il passé ?

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.



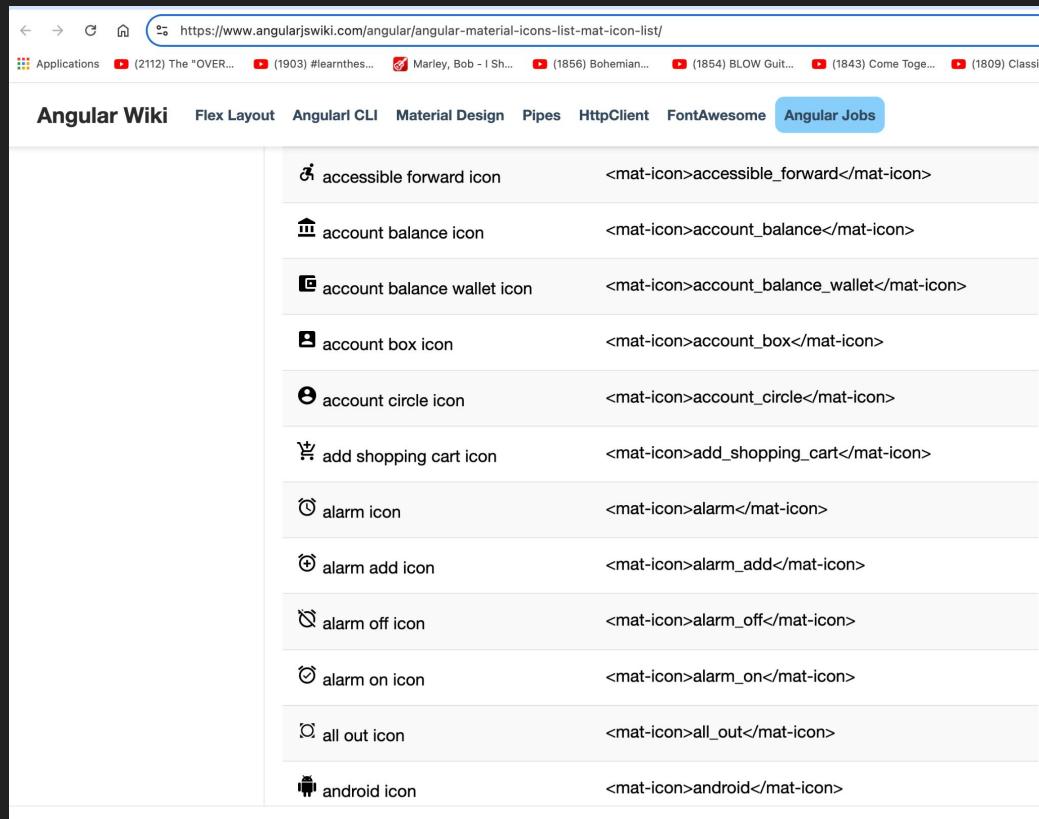
Et oui il faut importer et déclarer les modules dans le composant app.ts



```
app.html M app.css M app.ts 1, M X
src > app > app.ts > App > prof
You, il y a 1 minute | 1 author (You)
1 import { Component, signal } from '@angular/core';
2 import { RouterOutlet } from '@angular/router';
3 import { MatButtonModule } from '@angular/material/button';
4 import {MatIconModule} from '@angular/material/icon';
5 import {MatDividerModule} from '@angular/material/divider'
6
You, il y a 46 secondes | 1 author (You)
7 @Component({
8   selector: 'app-root',
9   imports: [RouterOutlet, MatButtonModule, MatIconModule, MatDividerModule],
10  templateUrl: './app.html',
11  styleUrls: ['./app.css']
12 })
13 export class App {
14   protected readonly title = signal('assignment-app');
15   prof:string = "Michel Buffa"; You, il y a 46 secondes • Uncommitted changes
16 }
17
```

Toujours utile, le Angular Wiki

Ici par exemple, [la page qui donne la liste de toutes les icônes](#) avec le tag HTML à utiliser :



The screenshot shows a web browser displaying the Angular Material Icons List page at <https://www.angularjswiki.com/angular/angular-material-icons-list-mat-icon-list/>. The page has a header with tabs: Angular Wiki, Flex Layout, Angular CLI, Material Design, Pipes, HttpClient, FontAwesome, and Angular Jobs (which is highlighted). Below the tabs is a table listing various icons with their corresponding HTML code. The table rows are as follows:

accessible forward icon	<mat-icon>accessible_forward</mat-icon>
account balance icon	<mat-icon>account_balance</mat-icon>
account balance wallet icon	<mat-icon>account_balance_wallet</mat-icon>
account box icon	<mat-icon>account_box</mat-icon>
account circle icon	<mat-icon>account_circle</mat-icon>
add shopping cart icon	<mat-icon>add_shopping_cart</mat-icon>
alarm icon	<mat-icon>alarm</mat-icon>
alarm add icon	<mat-icon>alarm_add</mat-icon>
alarm off icon	<mat-icon>alarm_off</mat-icon>
alarm on icon	<mat-icon>alarm_on</mat-icon>
all out icon	<mat-icon>all_out</mat-icon>
android icon	<mat-icon>android</mat-icon>

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 VS Code interface with the following details:

- EXPLORATEUR** sidebar: Shows the project structure with files like .angular, .github, .vscode, node_modules, public, and src. The assignments.ts file is currently selected.
- EDITOR**: The assignments.ts file is open, showing the following code:

```
src > app > assignments > TS assignments.ts > ...
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-assignments',
5   imports: [],
6   templateUrl: './assignments.html',
7   styleUrls: ['./assignments.css']
8 })
9 export class Assignments {
10
11 }
12
```

A red arrow points to the selector 'app-assignments' in the component definition.

Import du composant Assignments dans le composant App

The screenshot shows a code editor with four tabs at the top: app.html M, app.css M, app.ts 2, M X, and assignments.ts U. The assignments.ts U tab is active. The code in app.ts is as follows:

```
src > app > TS app.ts > App
4 | import {MatIconModule} from '@angular/material/icon';
5 | import {MatDividerModule} from '@angular/material/divider'
6 | import { Assignments } from './assignments/assignments';
7 |
8 | You, il y a 48 secondes | 1 author (You)
9 | @Component({
10 |   selector: 'app-root',
11 |   imports: [RouterOutlet, MatButtonModule, MatIconModule, MatDividerModule,
12 |             Assignments],
13 |   templateUrl: './app.html',
14 |   styleUrls: ['./app.css'
15 | })
16 | export class App {
17 |   protected readonly title = signal('assignment-app');
18 |   prof:string = "Michel Buffa";
19 | }
```

Two red arrows point to the import statement at line 6 and the import declaration in the imports array at line 11.

Afficher le composant dans le composant App

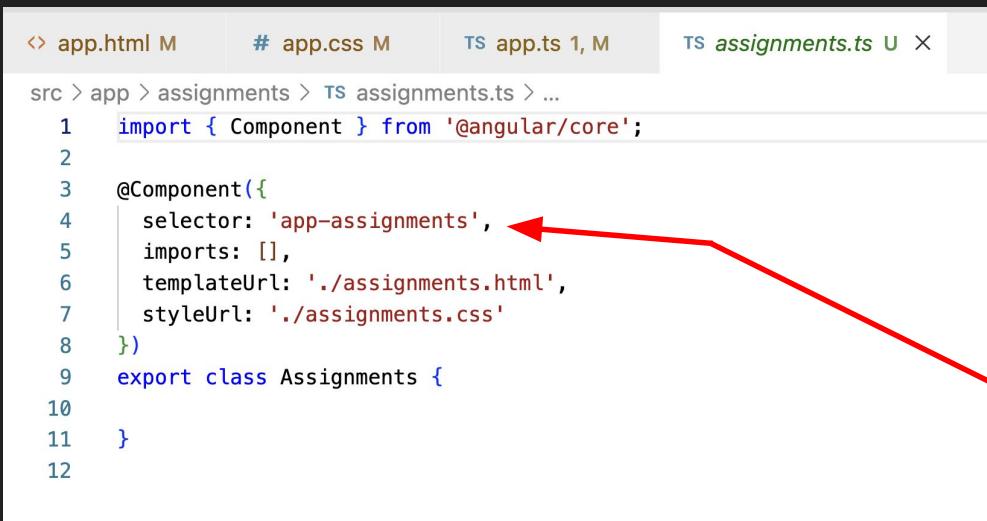
App est le composant “root” par défaut, c'est lui qui est affiché (voir app.ts et index.html ci-dessous)

```
7 You, il y a 48 secondes | 1 author (You)
8 @Component({
9   selector: 'app-root',
10  imports: [RouterOutlet, MatButtonModule, MatIconModule, M
11    ||| Assignments],
12  templateUrl: './app.html',
13  styleUrls: ['./app.css'
14})
15 export class App {
16   protected readonly title = signal('assignment-app');
17   prof:string = "Michel Buffa";
18 }
19 You, il y a 38 minutes • initial commit
```

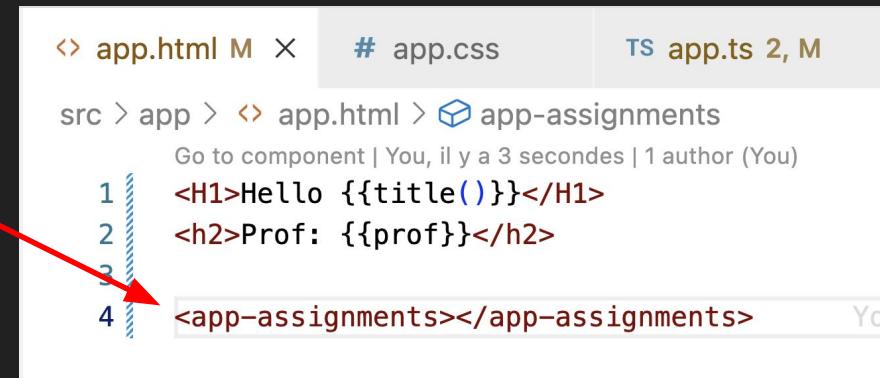
```
TS app.module.ts          TS app.component
assignment-app > src > <> index.html <
1 5 <title>AssignmentApp</title>
2 6 <base href="/">
3 7 <meta name="viewport" co
4 8 <link rel="icon" type="i
5 9 <link href="https://font
6 10 <link href="https://font
7 11 </head>
8 12 <body>
9 13 <app-root></app-root>
10 14 </body>
11 15 </html>
12 16
```

On va donc modifier le template de App

... et ajouter le sélecteur de Assignments (i.e un custom HTML Element, c'est un WebComponent) dans le template de App

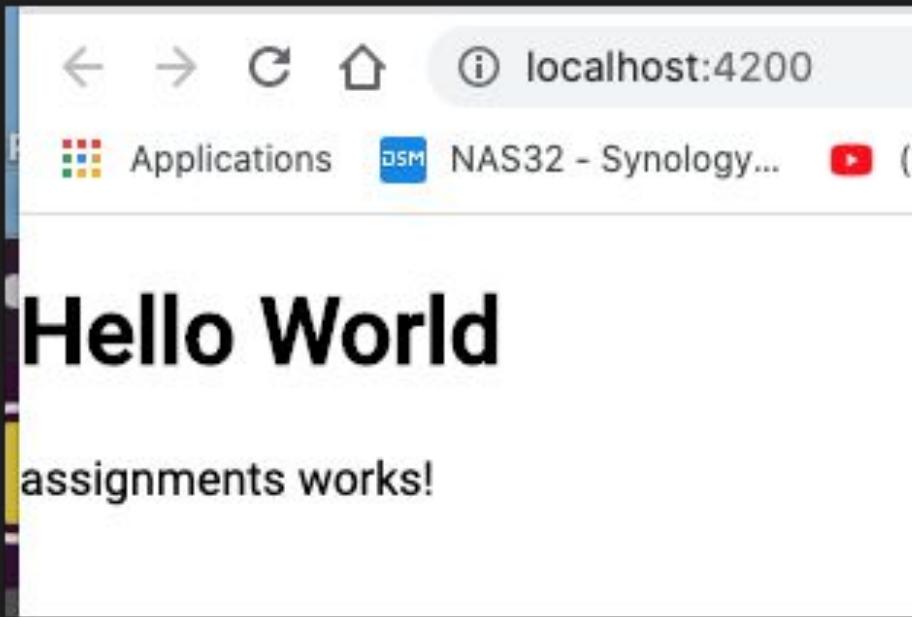


```
src > app > assignments > assignments.ts > ...
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-assignments', ←
5   imports: [],
6   templateUrl: './assignments.html',
7   styleUrls: ['./assignments.css']
8 })
9 export class Assignments {
10
11 }
12
```



```
src > app > app.html > app-assignments
1 <H1>Hello {{title()}}</H1>
2 <h2>Prof: {{prof}}</h2>
3
4 <app-assignments></app-assignments>
```

Et voir ce que cela donne....



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

Ajouter dans assignments.ts :

```
export class Assignments{  
    titre = "Mon application sur les Assignments !"  
}
```

Et dans le template assignments.html :

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



Allons un peu plus loin...

- Utiliser **@for** pour afficher des listes / collections
- Utiliser **@if** pour de l'affichage conditionnel de données
- Compléter l'utilisation de **@if** avec l'utilisation de **@else**
- Ce sont des... “directives” que l'on va utiliser dans les fichiers de template HTML des composants.
- Il existe des outils pour transformer du vieux code angular utilisant les anciennes directives ***ngIf** et ***ngFor** vers la syntaxe utilisée par Angular ≥ 17
 - Voir [cette vidéo](#).

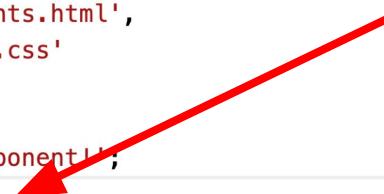
Que sont les directives ?

Il y en a de trois sortes :

1. **Structurelles** : elles manipulent le DOM en ajoutant ou en déplaçant des éléments (i.e `@for` ou `@if`)
2. **Attributs** : elles changent l'apparence ou le comportement des composants
3. **Composant** : ce sont des directives avec un template

@for

On a une propriété du composant qui est iterable (ex: un tableau)



```
s.html U      TS app.spec.ts      # app.css      TS app.ts 1, M      TS assignments.ts U X
src > app > assignments > TS assignments.ts > 📁 Assignments > 📄 assignments
1   import { Component } from '@angular/core';
2   import { DatePipe } from '@angular/common';
3   import { MatDividerModule } from '@angular/material/divider';
4
5   @Component({
6     selector: 'app-assignments',
7     imports: [DatePipe, MatDividerModule],
8     templateUrl: './assignments.html',
9     styleUrls: ['./assignments.css'
10    })
11  export class Assignments {
12    title = 'Assignments Component!';
13    assignments = [
14      {
15        nom: 'Angular Project',
16        dateDeRendu: '2024-12-31',
17        rendu: false
18      },
19      {
20        nom: 'TypeScript Basics',
21        dateDeRendu: '2024-11-15',
22        rendu: true
23      }
24    ];
25  }
```

@for : Le control-flow est intégré dans le template !

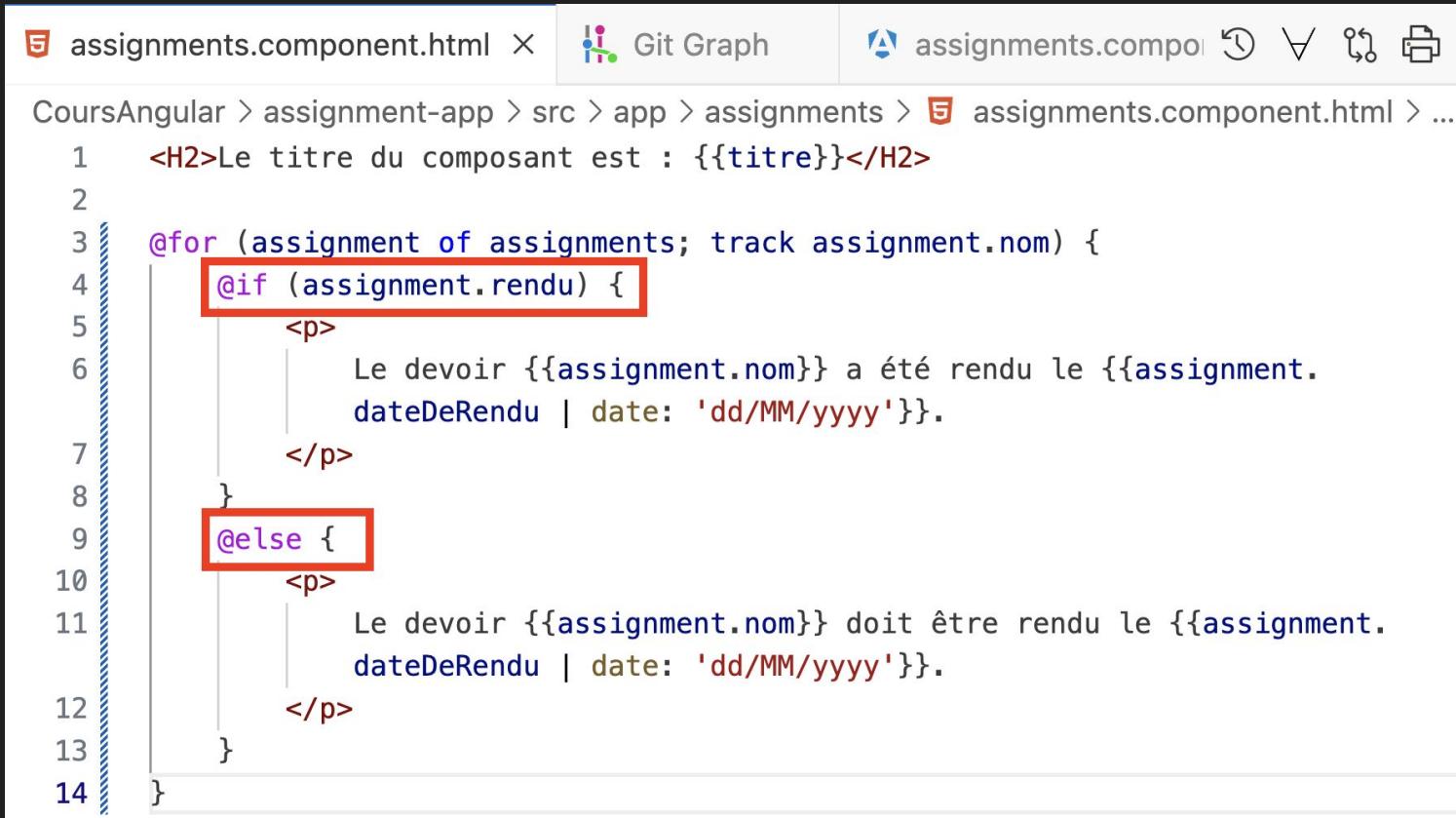
The image shows a code editor interface with two tabs: `assignments.html` and `app.ts`. The `assignments.html` tab contains the following template code:

```
src > app > assignments > assignments.html > ...
Go to component
1 <p>Liste des assignments :</p>
2
3 <mat-divider></mat-divider>
4 @for(assignment of assignments; track assignment.nom) {
5   <div>
6     <h3>{{assignment.nom}}</h3>
7     <p>Due date: {{assignment.dateDeRendu | date}}</p>
8     <p>Rendu: {{assignment.rendu ? 'Oui' : 'Non'}}</p>
9     <mat-divider></mat-divider>
10  </div>
11 }
```

The `app.ts` tab contains the following component definition:

```
src > app > assignments > assignments.ts > Assignments > assignments.ts
1 import { Component } from '@angular/core';
2 import { DatePipe } from '@angular/common';
3 import { MatDividerModule } from '@angular/material/divider';
4
5 @Component({
6   selector: 'app-assignments',
7   imports: [DatePipe, MatDividerModule],
8   templateUrl: './assignments.html',
9   styleUrls: ['./assignments.css']
10 })
11 export class Assignments {
12   title = 'Assignments Component!';
13   assignments = [
14     { nom: 'Angular Project',
```

RECOMMANDÉ: @if et @else (Angular >= 17)



The screenshot shows a code editor with the file `assignments.component.html` open. The code uses Angular's structural directives (@for, @if, @else) to display assignment information. Two specific sections of the code are highlighted with red boxes:

- Line 4:** `@if (assignment.rendu) {`
- Line 9:** `@else {`

```
1  <H2>Le titre du composant est : {{titre}}</H2>
2
3  @for (assignment of assignments; track assignment.nom) {
4      @if (assignment.rendu) {
5          <p>
6              Le devoir {{assignment.nom}} a été rendu le {{assignment.
7                  dateDeRendu | date: 'dd/MM/yyyy'}}.
8          </p>
9      @else {
10          <p>
11              Le devoir {{assignment.nom}} doit être rendu le {{assignment.
12                  dateDeRendu | date: 'dd/MM/yyyy'}}.
13          </p>
14      }
```

Partie 3 - styling CSS conditionnel

Ce que l'on va voir

Styler des éléments avec la directive `[style.css_prop_name]`, par exemple
`[style.color]`

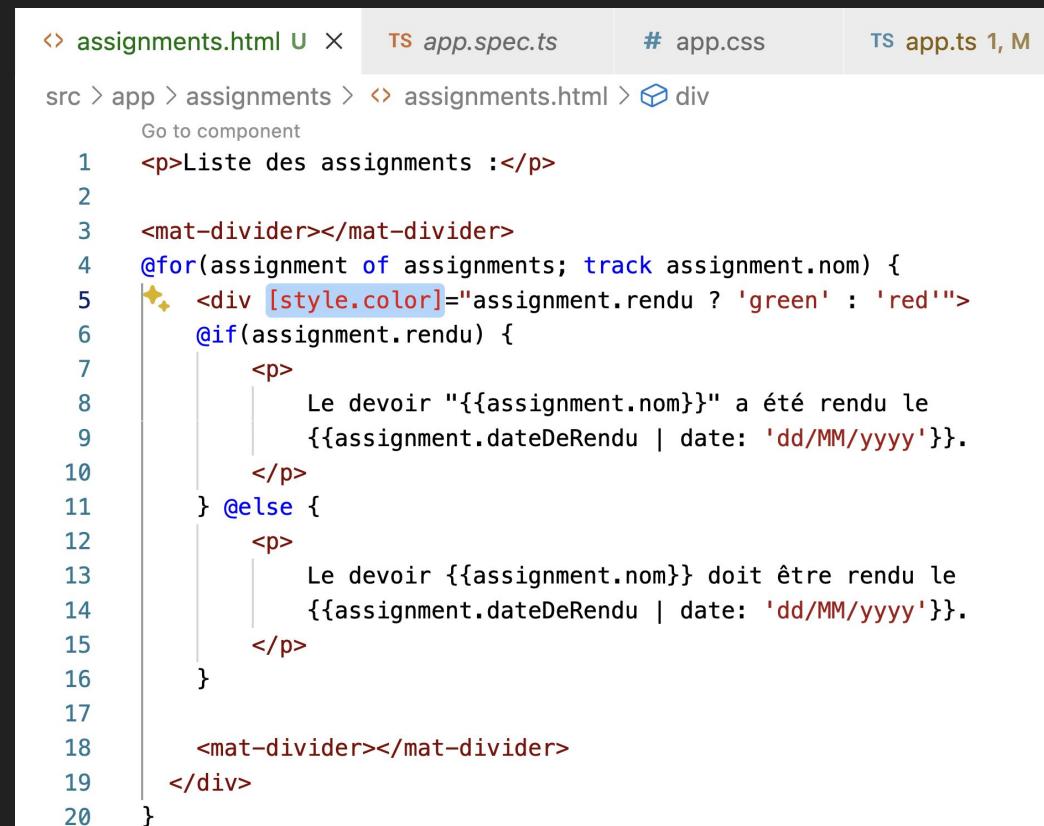
Styler des éléments avec la directive `[class.nom_classe]`

Styler des éléments avec nos propres directives custom

Exemple avec [style.color]

Utilisées avec
Attention à
l'utilisation des
simples et double
quotes !

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



The screenshot shows a code editor with four tabs at the top: assignments.html, app.spec.ts, app.css, and app.ts. The assignments.html tab is active, displaying the following template code:

```
<p>Liste des assignments :</p>
<mat-divider></mat-divider>
@for(assignment of assignments; track assignment.nom) {
  <div [style.color]="assignment.rendu ? 'green' : 'red'">
    @if(assignment.rendu) {
      <p>
        Le devoir "{{assignment.nom}}" a été rendu le
        {{assignment.dateDeRendu | date: 'dd/MM/yyyy'}}.
      </p>
    } @else {
      <p>
        Le devoir {{assignment.nom}} doit être rendu le
        {{assignment.dateDeRendu | date: 'dd/MM/yyyy'}}.
      </p>
    }
  </div>
<mat-divider></mat-divider>
</div>
}
```

A yellow star-shaped cursor is positioned over the line containing the `[style.color]` binding. The code uses template literals and pipes for date formatting.

Styling CSS avec une expression

```
<div [style.color]="assignment.rendu ? 'green' : 'red'">
```

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

On aurait pu écrire aussi :

```
<div [style.color]="getColor(assignment)">
```

et dans le fichier assignments.ts

```
getColor(a: any) {  
  if (a.rendu) return 'green';  
  else return 'red';  
}
```

Classes CSS

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

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

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

Affectation d'une ou plusieurs classes CSS

Et on modifie le template du composant :

```
<div [class.rendu]="assignment.rendu"  
      [class.nonRendu]="!assignment.rendu">
```

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

Style ou Classe ?

Réponse classique : **class** 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.

Exemple typique : créer un directory “shared”

Dans **src/app/shared** c'est là que vous mettrez vos librairies, modules et directives custom !

Exemple de création d'une directive “rendu” (une fois qu'on est dans ce répertoire)

```
cd src/app/
```

```
mkdir shared (ou mdir si sous Windows)
```

```
cd shared
```

```
ng g d --skip-tests rendu
```

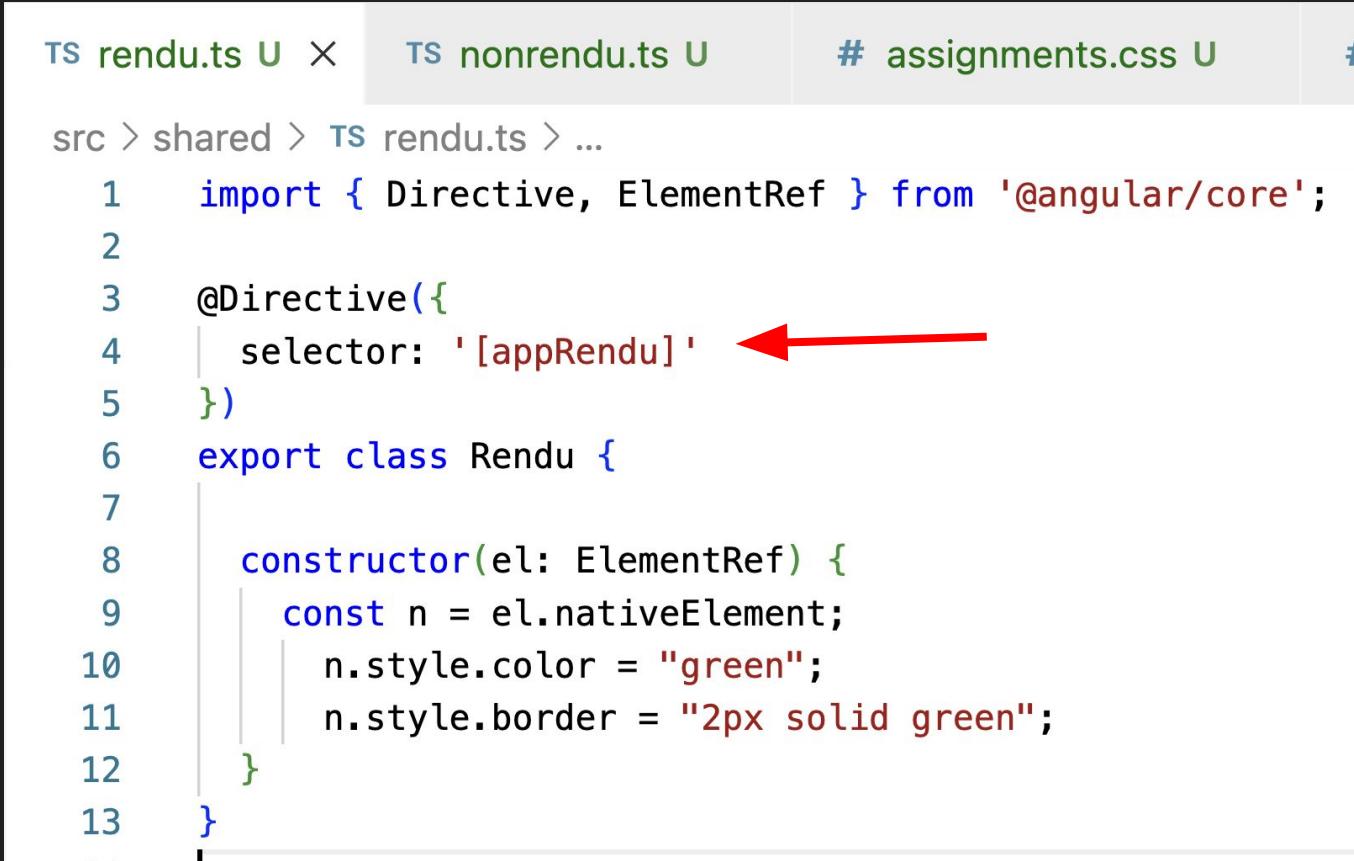
Exemple typique : créer un directory “shared”

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

- EXPLORATEUR**: Shows the project structure under **ASSIGNMENT-APP**.
 - src**: Contains **app**, **assignments** (highlighted), **app.config.ts**, **# app.css**, **app.html** (modified), **app.routes.ts**, **app.spec.ts**, and **app.ts**.
 - shared**: Contains **rendu.ts** (highlighted) and **custom-theme.scss**.
- ...** : Shows the current file tabs.
- assignments.html**: Preview tab.
- TS rendu.ts**: Editor tab, currently active.
- # assignn**: Preview tab.
- Code Editor Content:** The code is a TypeScript file named **rendu.ts**. It defines a class **Rendu** with a constructor. A red arrow points to the selector in line 4: `selector: '[appRendu]'`.

```
1 import { Directive } from '@angular/core';
2
3 @Directive({
4   selector: '[appRendu]' ←
5 })
6 export class Rendu {
7
8   constructor() { }
9
10 }
11
```

On écrit un peu de code, notez le sélecteur !



The screenshot shows a code editor with three tabs at the top: "TS rendu.ts U X", "TS nonrendu.ts U", and "# assignments.css U". The "# assignments.css U" tab is active. Below the tabs, the file path "src > shared > TS rendu.ts > ..." is displayed. The code itself is as follows:

```
1 import { Directive, ElementRef } from '@angular/core';
2
3 @Directive({
4   selector: '[appRendu]' ← Red arrow points here
5 })
6 export class Rendu {
7
8   constructor(el: ElementRef) {
9     const n = el.nativeElement;
10    n.style.color = "green";
11    n.style.border = "2px solid green";
12  }
13}
```

Nouvelle version du template assignments.html

```
@for(assignment of assignments; track assignment.nom) {  
    @if(assignment.rendu) {  
        <p appRendu> ←  
            Le devoir {{assignment.nom}}, à rendre pour le  
            {{assignment.dateDeRendu | date:'d/MM/YYYY' }},  
            a été rendu par l'étudiant.  
        </p>  
    } @else {  
        <p appNonRendu> ←  
            Le devoir {{assignment.nom}}, à rendre pour le  
            {{assignment.dateDeRendu | date:'d/MM/YYYY' }},  
            n'a pas encore été rendu.  
        </p>  
    }  
}
```

Penser à importer la directive dans le composant assignments.ts

The screenshot shows a code editor with tabs for assignments.css, app.css, app.ts, and assignments.ts. The assignments.ts tab is active, displaying the following code:

```
src > app > assignments > assignments.ts > Assignments > assignments
1 import { Component } from '@angular/core';
2 import { DatePipe } from '@angular/common';
3 import { MatDividerModule } from '@angular/material/divider';
4 import { Rendu } from '../../../../../shared/rendu'; // Red arrow here
5 import { NonRendu } from '../../../../../shared/nonrendu'; // Red arrow here
6
7
8 @Component({
9   selector: 'app-assignments',
10  imports: [DatePipe, MatDividerModule, Rendu, NonRendu],
11  templateUrl: './assignments.html',
12  styleUrls: ['./assignments.css']
13})
14 export class Assignments {
```

Three red arrows point to the import statements for the Rendu and NonRendu modules, indicating they should be imported into the component.

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)

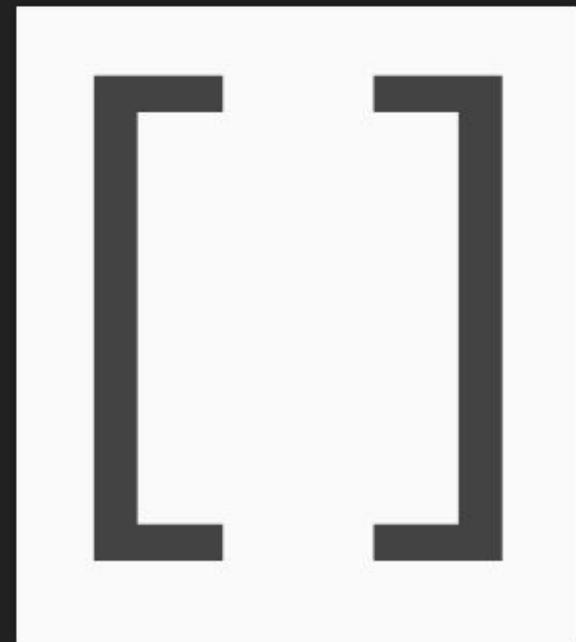
Binding d'attributs

On fera cela à l'aide des crochets []

Par exemple **[style.color]**

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

Binding = évaluation...

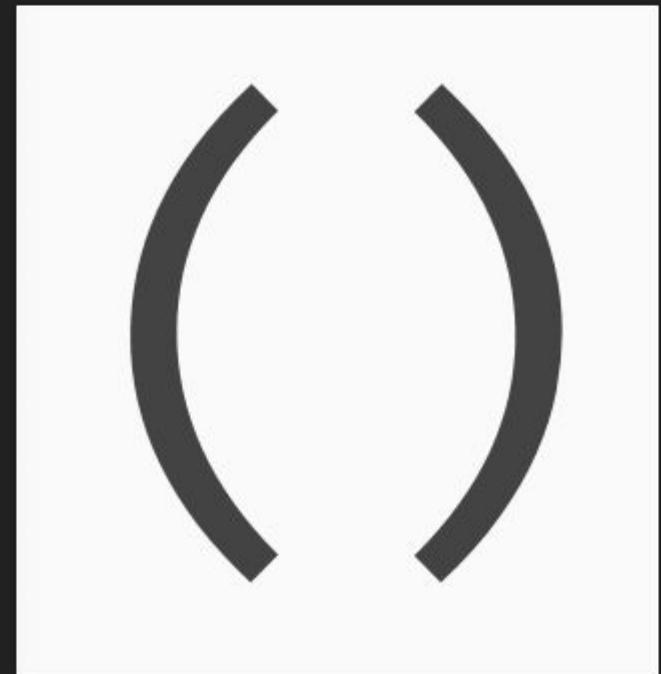


Binding d'évenements et de signaux (on verra plus tard)

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

Par exemple

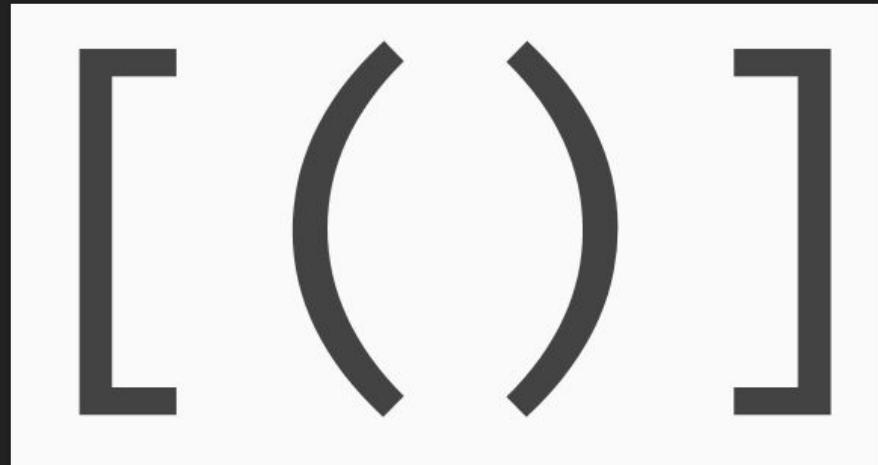
- (**click**)
- (**keyup**)
- etc.



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**

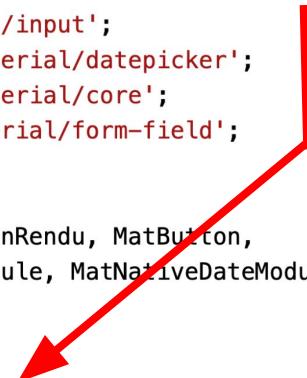
Sinon on a des erreurs de l'espace ! Déjà que....

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

Dans le composant qui en a besoin : ici
assignments.ts



```
src > app > assignments > TS assignments.ts > ↗ Assignments > ↘ ajoutActive
1 import { Component, OnInit } from '@angular/core';
2 import { DatePipe } from '@angular/common';
3 import { MatDividerModule } from '@angular/material/divider';
4 import { Rendu } from '../../shared/rendu';
5 import { NonRendu } from '../../shared/nonrendu';
6 import { MatButton } from '@angular/material/button';
7 import { FormsModule } from '@angular/forms';
8 import { MatInputModule } from '@angular/material/input';
9 import { MatDatepickerModule } from '@angular/material/datepicker';
10 import { MatNativeDateModule } from '@angular/material/core';
11 import { MatFormFieldModule } from '@angular/material/form-field';
12 @Component({
13   selector: 'app-assignments',
14   imports: [DatePipe, MatDividerModule, Rendu, NonRendu, MatButton,
15     FormsModule, MatInputModule, MatDatepickerModule, MatNativeDateModule,
16     MatFormFieldModule],
17   templateUrl: './assignments.html',
18   styleUrls: ['./assignments.css'
19 })
20 export class Assignments implements OnInit {
```

Comment connaître les modules nécessaires ?

Lire la doc Angular ! Regarder le code des exemples !

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.html`, il contiendra un seul bouton pour le moment :



Bon, alors, ce formulaire ?

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

Variable qui fait référence à l'élément HTML form

```
(assignments.html) U X TS rendu.ts U TS nonrendu.ts U  
src > app > assignments > (assignments.html) > ...  
Go to component  
1 <h2>Ajout d'un assignment</h2>  
2 <form ngForm #assignmentForm>  
3   <button mat-stroked-button color="primary">  
4     Ajouter un devoir  
5   </button>  
6 </form>
```



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

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

On va ,dans le composant Assignments

1. Créer une variable **ajoutActive** qui sera false au début
2. Dire que la classe implémente l'interface OnInit (qu'il faut importer) qui impose l'existence d'un *initialiseur* appelé ngInit() appelé à la création.
3. On va par exemple ajouter un **setTimeOut** qui va l'activer au bout de deux secondes, dans **ngInit(...)**
4. On binde l'attribut HTML **disabled** sur la variable **ajoutActive**

Live coding !

Allez, on y va (et on teste !)

Dans assignments.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)

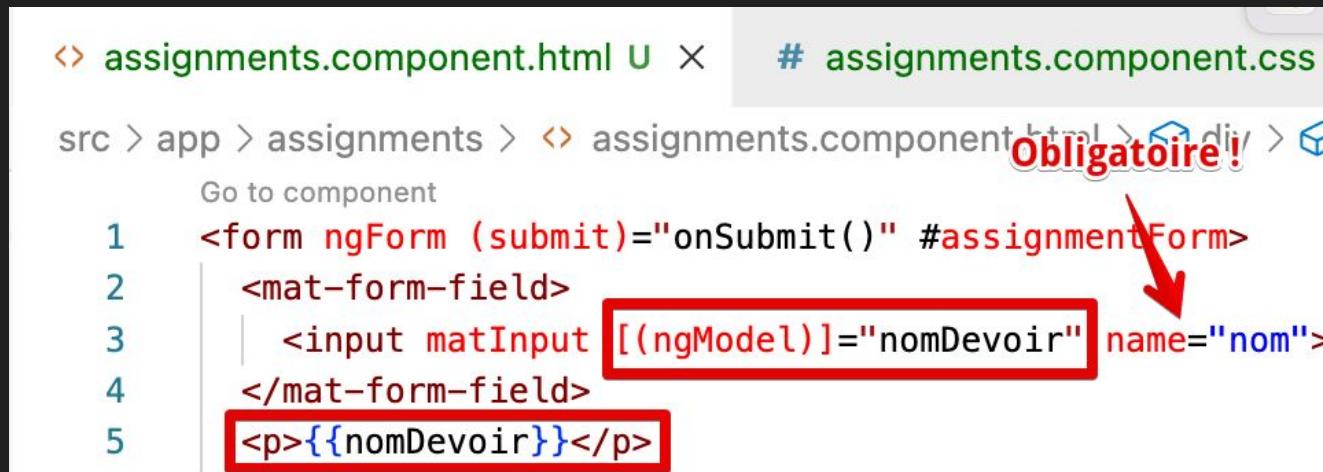
```
<form ngForm #assignmentForm class="form"

(submit)="onSubmit(nomAssignment.value)">
  <mat-form-field>
    <input matInput #nomAssignment>
  </mat-form-field>

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

```
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 the Visual Studio Code interface. On the left is the Explorer sidebar with a tree view of the project structure:

- > EXPLORATEUR
- > 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

The file **assignment.model.ts** is selected in the Explorer and is currently being edited. The editor tab bar shows the file is saved (**U**). The status bar indicates the file path: **src > app > assignments > TS assignment.model.ts**. The code editor 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 overview page. The Datepicker component is highlighted in the sidebar. The main content area displays the 'OVERVIEW' tab, which includes a brief description of thedatepicker component and a code example. The code example shows the HTML and TypeScript code for a basic datepicker. Below the code, there is a preview of the datepicker component with a label 'Choose a date' and a date input field.

```
<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="Nom du devoir" required>
4    </mat-form-field>
5    <mat-form-field>
6      <input matInput type="date" placeholder="Date de rendu" required>
7    </mat-form-field>
8
9    <button mat-raised-button type="submit" (click)="onSubmit()">
10   Ajouter
11 </button>
12 </form>
```

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';
```

onSubmit() {

const newAssignment = new Assignment();

newAssignment.nom = nomInput.value;

newAssignment.dateDeRendu = dateDeRenduInput.value;

newAssignment.rendu = false;

this.assignments.push(newAssignment);

}

```
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()`

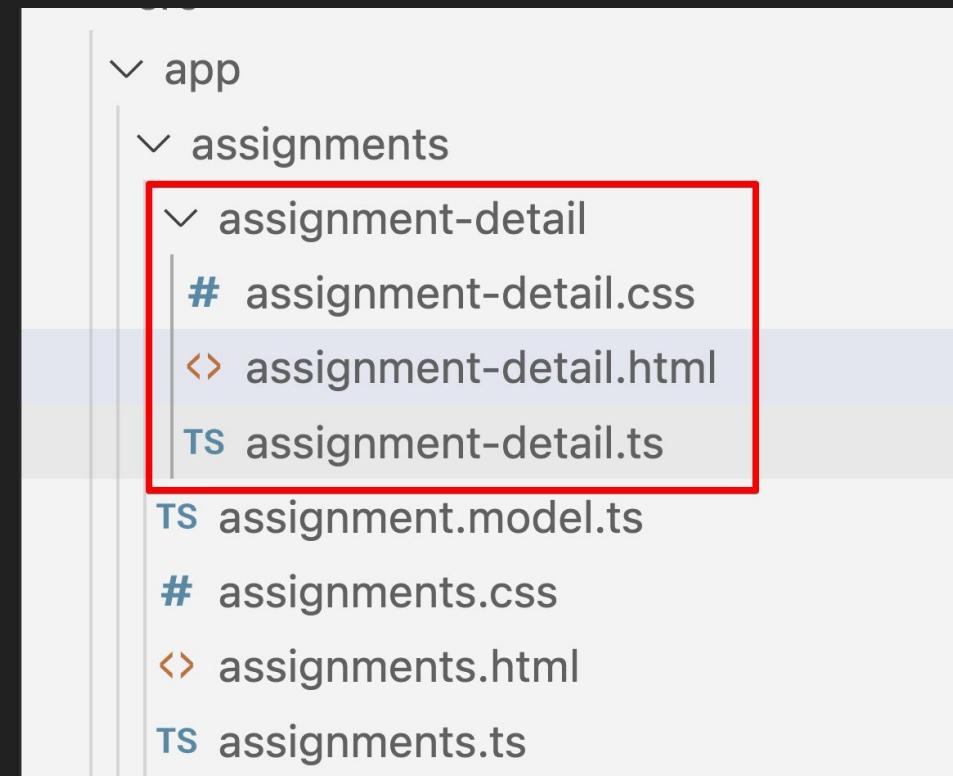
Passer des données du composant fils vers le composant parent avec `Output()`

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 avec **cd src/app/assignments**, puis exécuter la commande suivante :

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



Créer une instance du sous composant dans le

compo

Hello World

1.

Nom du devoir

2.

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

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!

On
clie

Utilis

corre

On v
babie

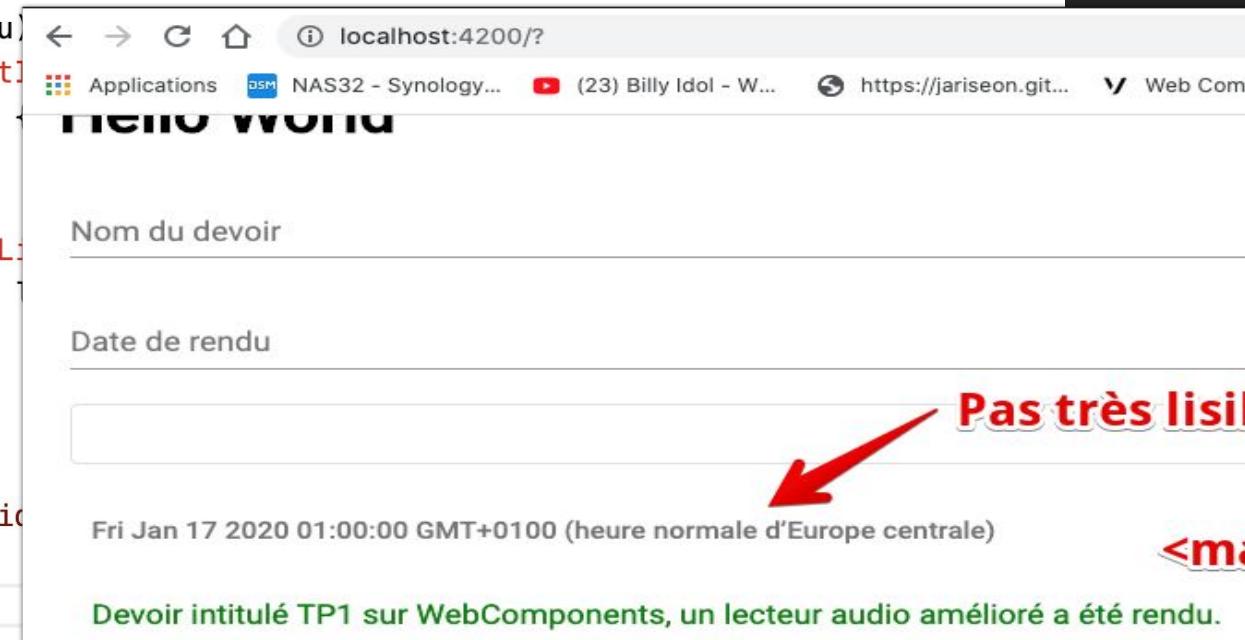
```
<mat-list>
  @for (assignment of assignments; track assignment.nom) {

    <h3 mat-subheader>
      {{ assignment.dateDeRendu | date:'dd/MM/yyyy' }}
    </h3>

    <mat-list-item >
      <div matListTitle>{{ assignment.nom }}</div>

      @if (assignment.rendu)
        <p appRendu matListTitle>
          ... a été rendu le ...
        </p>
      } @else {
        <p appNonRendu matListTitle>
          Non rendu : date ...
        </p>
      }
    </mat-list-item>

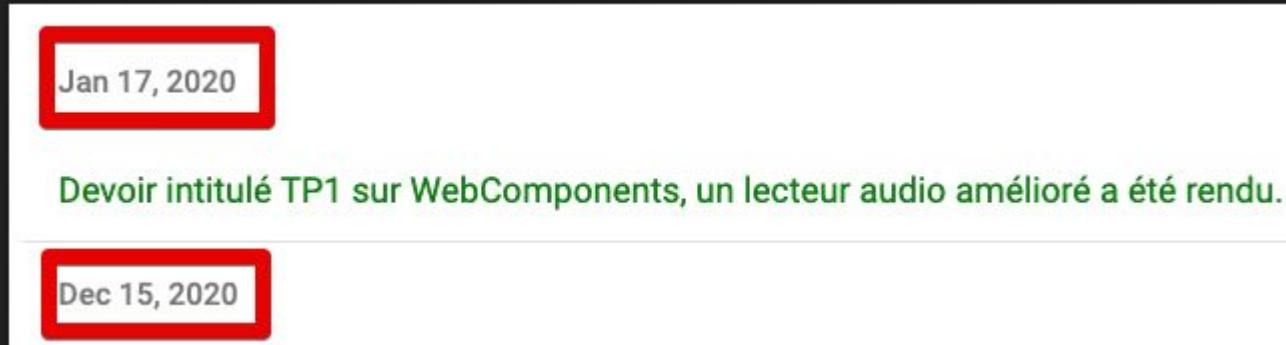
    <mat-divider></mat-divider>
  }
</mat-list>
```



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;  
}
```

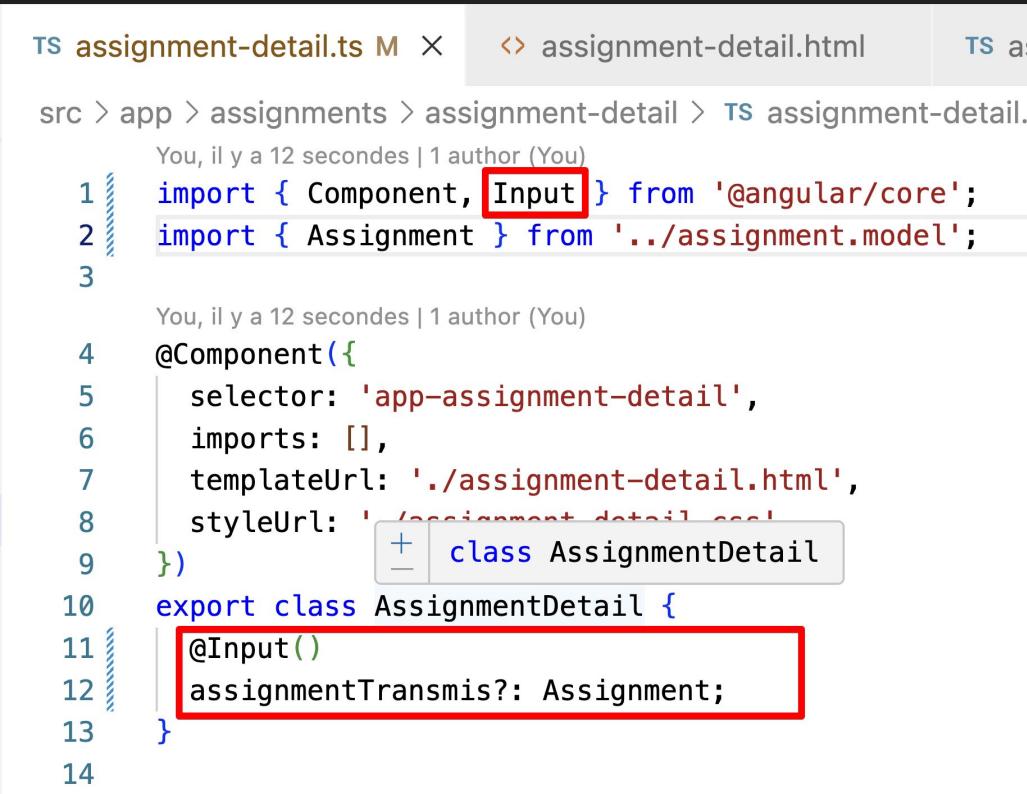
Mais comment passer l'info à assignment-detail

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



```
TS assignment-detail.ts M X assignment-detail.html TS as
src > app > assignments > assignment-detail > TS assignment-detail.t
You, il y a 12 secondes | 1 author (You)
1 import { Component, Input } from '@angular/core';
2 import { Assignment } from '../assignment.model';
3
You, il y a 12 secondes | 1 author (You)
4 @Component({
5   selector: 'app-assignment-detail',
6   imports: [],
7   templateUrl: './assignment-detail.html',
8   styleUrls: ['./assignment-detail.css']
9 })
10 export class AssignmentDetail {
11   @Input()
12   assignmentTransmis?: Assignment;
13 }
14
```

Passage de l'assignment depuis le template du composant père

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

```
src > app > assignments > assignments.html > app-assignment-detail
28  <mat-list>
32      <mat-list-item (click="assignmentClique(assignment)">
33          <p appNonRendu>
34              L'assignment cliqué
35          </p>
36      )
37  </mat-list-item>
38  </mat-list>
39
40  </p>
41  </div>
42  </mat-list-item>
43 </mat-list>
44 </mat-list>
45
46 <h1>Detail d'un assignment</h1>
47 <app-assignment-detail [assignmentTransmis]="assignmentSelectionne">
48     You, il y a 1 seconde • Uncommitted changes
49 </app-assignment-detail>
```

The `assignments.html` file contains the following HTML template:

```
<mat-list>
    <mat-list-item (click="assignmentClique(assignment)">
        <p appNonRendu>
            L'assignment cliqué
        </p>
    )
    <mat-list-item>
        <p>
            @Input dans le fils
        </p>
    </mat-list-item>
</mat-list>
</mat-list>
<h1>Detail d'un assignment</h1>
<app-assignment-detail [assignmentTransmis]="assignmentSelectionne">
    You, il y a 1 seconde • Uncommitted changes
</app-assignment-detail>
```

Annotations in red highlight specific parts of the code:

- `L'assignment cliqué` is annotated with `(propriété dans le composant père)`.
- `@Input dans le fils` is annotated with two red arrows pointing to the `[assignmentTransmis]` attribute in the `app-assignment-detail` component.
- The entire line `<app-assignment-detail [assignmentTransmis]="assignmentSelectionne">` is highlighted with a red box.

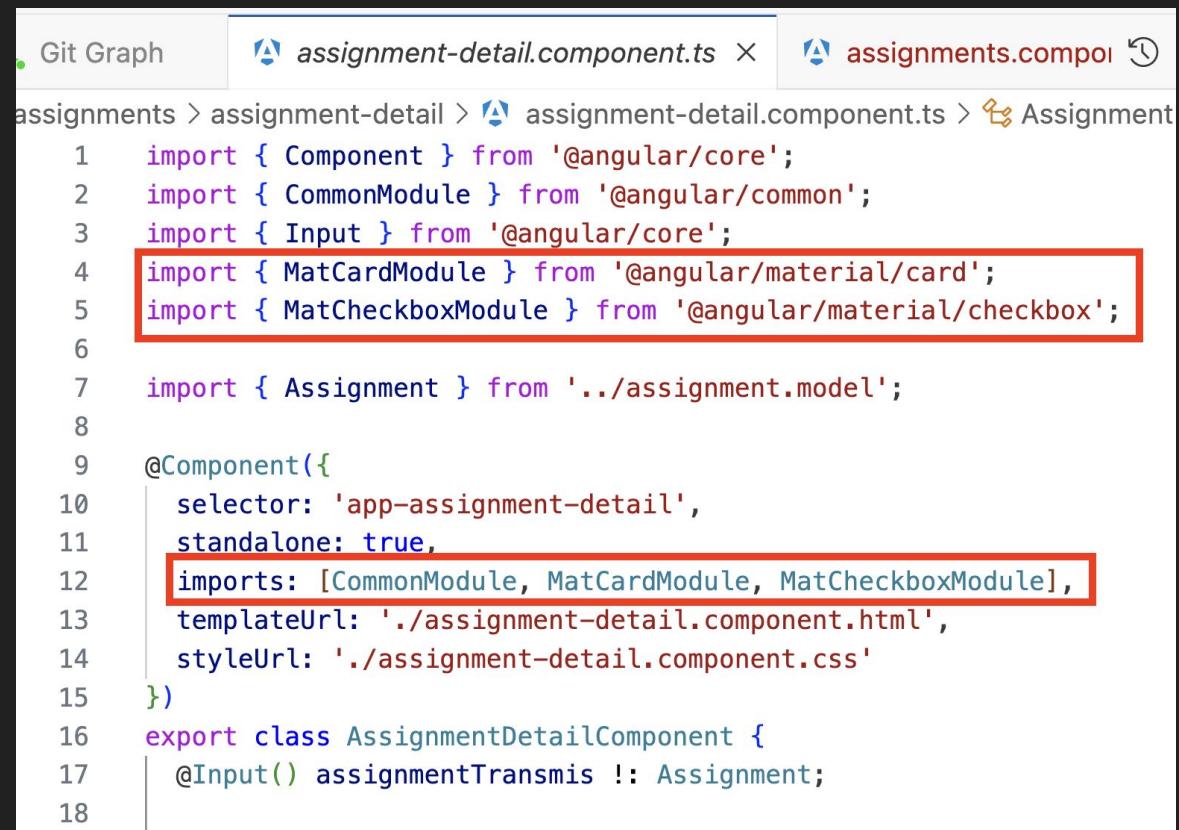
Il ne reste plus qu'à afficher l'assignment transmis dans le template du composant détail

```
@if(assignmentTransmis) {  
    <p>{ {assignmentTransmis.nom} } </p>  
}
```

Le `@if` 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

Importer les modules angular dans le composant AssignmentDetail

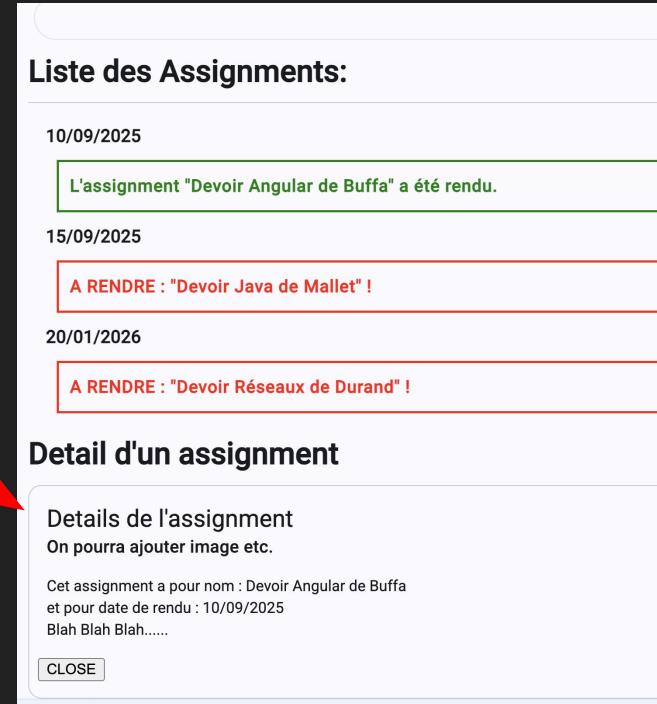


```
Git Graph assignment-detail.component.ts × assignments.compo ⏱
assignments > assignment-detail > assignment-detail.component.ts > Assignment
1 import { Component } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { Input } from '@angular/core';
4 import { MatCardModule } from '@angular/material/card';
5 import { MatCheckboxModule } from '@angular/material/checkbox';
6
7 import { Assignment } from '../assignment.model';
8
9 @Component({
10   selector: 'app-assignment-detail',
11   standalone: true,
12   imports: [CommonModule, MatCardModule, MatCheckboxModule],
13   templateUrl: './assignment-detail.component.html',
14   styleUrls: ['./assignment-detail.component.css']
15 })
16 export class AssignmentDetailComponent {
17   @Input() assignmentTransmis!: Assignment;
18 }
```

Utilisation de <mat-card> dans le composant détail

M TS assignment-detail.ts M ◉ assignment-detail.html M

```
assignments > assignment-detail > ◉ assignment-detail.html > mat-card > mat-card-content
Go to component | You, il y a 1 seconde | 1 author (You)
1 @if(assignmentTransmis) {
2   <mat-card appearance="outlined">
3     <mat-card-header>
4       <mat-card-title>Details de l'assignment</mat-card-title>
5       <mat-card-subtitle>On pourra ajouter image etc.</mat-card-subtitle>
6     </mat-card-header>
7     <mat-card-content>
8       <p>
9         Cet assignment a pour nom : {{assignmentTransmis.nom}} <br>
10        et pour date de rendu :
11        {{assignmentTransmis.dateDeRendu | date:'dd/MM/yyyy'}} <br>
12        Blah Blah Blah.....<br>
13      </p>
14    </mat-card-content>
15    <mat-card-actions>
16      <button matButton>CLOSE</button>
17    </mat-card-actions>
18  </mat-card>
19 }
```



Rajout d'une Material Checkbox pour indiquer si l'assignment a été rendu ou pas

Encore une fois : importer le ou les modules nécessaires !

The screenshot shows a code editor with two tabs: 'assignment-detail.ts' and 'assignment-detail'. The code editor displays the following TypeScript code:

```
src > app > assignments > assignment-detail >
  You, il y a 1 seconde | 1 author (You)
  1 import { Component, Input } from '@angular/core'
  2 import { Assignment } from './assignment'
  3 import { MatCardModule } from '@angular/material/card'
  4 import { MatCheckboxModule } from '@angular/material/checkbox'
  5 import { DatePipe } from '@angular/common'
  6
  You, il y a 1 seconde | 1 author (You)
  7 @Component({
  8   selector: 'app-assignment-detail',
  9   imports: [ MatCardModule, MatCheckboxModule ],
  10  templateUrl: './assignment-detail.component.html',
  11  styleUrls: ['./assignment-detail.component.css']
  12 })
  13 export class AssignmentDetail {
  14   @Input()
  15   assignmentTransmis?: Assignment
  16 }
```

The 'assignment-detail' tab shows a preview of the component. The title is 'Detail d'un assignment'. The main content area contains the text 'Details de l'assignment' and 'On pourra ajouter image etc.'. Below this, it states 'Cet assignment a pour nom : Devoir Java de Mallet et pour date de rendu : 15/09/2025'. It ends with the text 'Blah Blah Blah.....'. At the bottom right, there is a button labeled 'Rendu' with an empty checkbox icon.

Implémentation de la logique de la checkbox

```
<mat-card-actions>
  @if( !assignmentTransmis.rendu) {
    <mat-checkbox (click)="onAssignmentRendu();">
      Rendu
    </mat-checkbox>
  }
</mat-card-action
```

onAssignmentRendu() {
 if(this.assignmentTransmis) {
 this.assignmentTransmis.rendu =
 !this.assignmentTransmis.rendu; **ait que si
as rendu**
 }
}

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

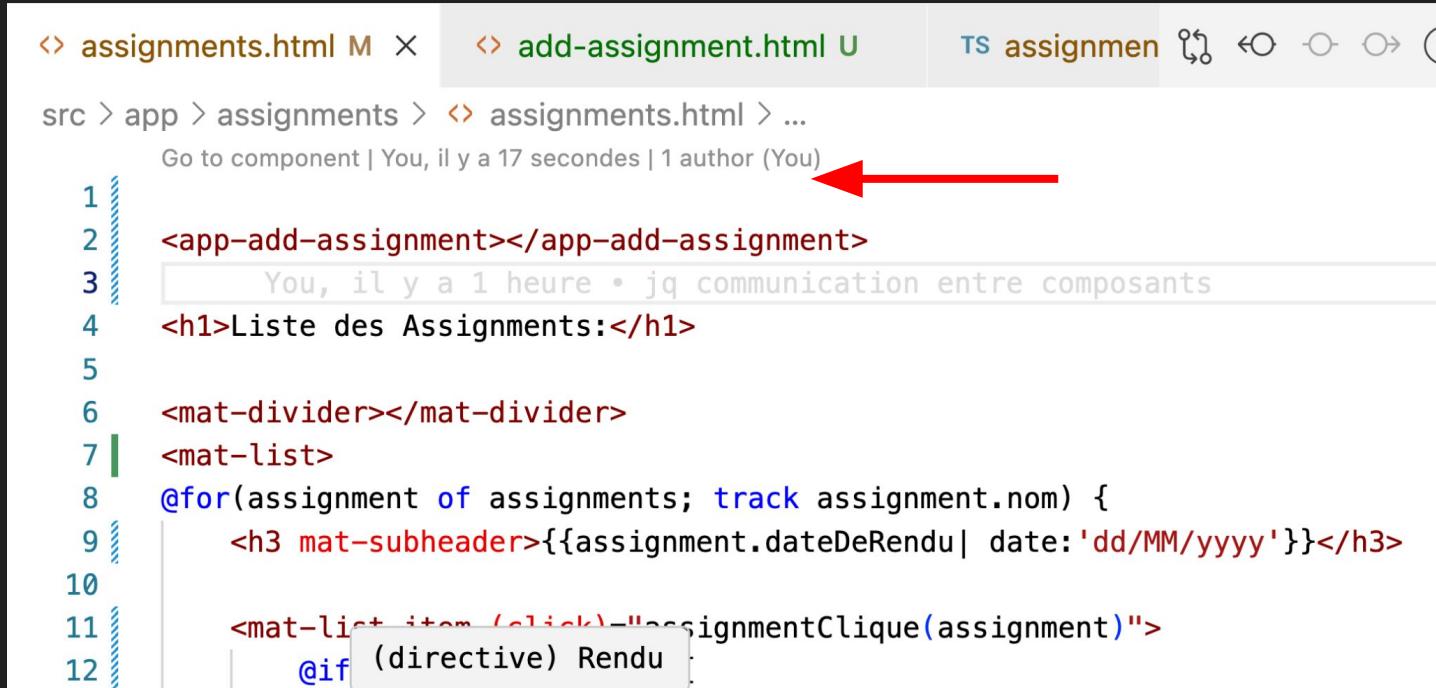
Vous commencez à connaître la méthode. On se rend dans src/app/assignments et on utilise la commande :

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

```
✓ src
  ✓ app
    ✓ assignments
      ✓ add-assignment
        # add-assignment.css
        <> add-assignment.html
        TS add-assignment.ts
        > assignment-detail
        TS assignment.model.ts
        # assignments.css
```

Afficher le fils dans le template du père

Et tester l'application pour vérifier que le template HTML du fils s'affiche bien !



The screenshot shows a code editor with three tabs: 'assignments.html M X', 'add-assignment.html U', and 'TS assignmen'. The 'assignments.html' tab is active. The code in the editor is:

```
src > app > assignments > <> assignments.html > ...
Go to component | You, il y a 17 secondes | 1 author (You)
1 <app-add-assignment></app-add-assignment>
2   You, il y a 1 heure • jq communication entre composants
3 <h1>Liste des Assignments:</h1>
4
5
6 <mat-divider></mat-divider>
7 <mat-list>
8 @for(assignment of assignments; track assignment.nom) {
9   <h3 mat-subheader>{{assignment.dateDeRendu| date:'dd/MM/yyyy'}}</h3>
10
11   <mat-list-item (click)="assignmentClique(assignment)">
12     @if (directive) Rendu
```

A red arrow points from the text 'You, il y a 1 heure • jq communication entre composants' to the line of code containing the child component tag '<app-add-assignment></app-add-assignment>'.

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

Penser à déplacer de assignments.ts vers add-assignment.ts : les imports et les variables associées aux champs input du formulaire, et la méthode onAddAssignment, pour qu'il n'y ait plus d'erreurs.

```
add-assignment.html U X assignment-detail.ts M assignment-detail.html M
src > app > assignments > add-assignment > add-assignment.html > form.form > button

1 <h1>Ajout d'un Assignment</h1>
2
3 <form ngForm #assignmentForm (submit)="onAjouterAssignment($event); assignmentForm.reset()" class="form">
4   <mat-form-field>
5     <input matInput placeholder="Nom de l'assignment" name="nomAssignment" type="text" required="required" [(ngModel)]="nomAssignment" [disabled]="!dateDeRendu" />
6   <mat-form-field>
7     <mat-label>Choose a date</mat-label>
8     <input matInput [matDatepicker]="picker" name="dateDeRendu" type="text" [(ngModel)]="dateDeRendu" required="required" />
9     <mat-hint>MM/DD/YYYY</mat-hint>
10    <mat-datepicker-toggle matIconSuffix [for]="picker"></mat-datepicker-toggle>
11    <mat-datepicker #picker></mat-datepicker>
12  </mat-form-field>
13
14  <button mat-stroked-button color="primary" [disabled]="!nomAssignment || !dateDeRendu">
15    Ajouter
16  </button>
17
18
19
20
21
22
23
24
25
```

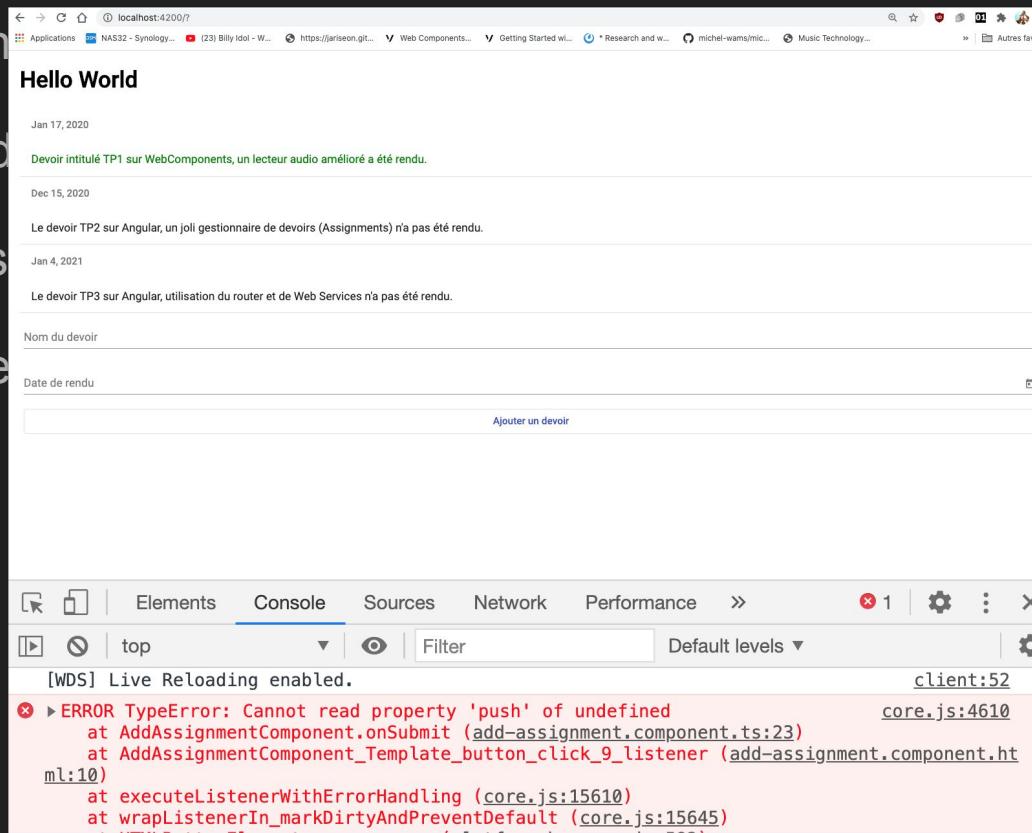
Finalement : on déplace la classe CSS .form

... du CSS du composant

On supprime le [disabled]

On ajoute un import vers

Et on va regarder ce que



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 avec un bouton “Ajouter Devoir” en haut de la fenêtre:

- Par défaut on voit la liste des assignments
- Si on clique le nouveau bouton on verra à plus la liste...
- Faisons cela avant de revenir sur le problème assignments...

```
src > app > assignments > add-assignment > TS add-assignment.ts > AddAssignment
15  }
16  export class AddAssignment {
17    // Pour le formulaire, une variable par champ
18    nomAssignment = "";
19    dateDeRendu!: Date;
20
21    onAjouterAssignment(event:any) {
22      console.log("Ajout NOM = " + this.nomAssignment + " date = " + this.dateDeRendu);
23
24      // On crée un nouvel assignment
25      const nouvelAssignment: Assignment = new Assignment();
26      nouvelAssignment.nom = this.nomAssignment;
27      const nouvelAssignment: Assignment = new Assignment();
28      nouvelAssignment.dateDeRendu;
29      nouvelAssignment.rendu = false; // Pas rendu
30
31      console.log(nouvelAssignment);
32
33      // On l'ajoute à la liste
34      //this.assignments.push(nouvelAssignment);
35    }
36
37  }
```

Ceci ne peut pas fonctionner car le tableau des assignments est dans le composant père !

On va arranger quelques trucs avant...

Avant de faire marcher l'ajout 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 (assignments.html)

```
<button mat-flat-button color="accent"  
       (click)="onAddAssignmentBtnClick()"  
> Ajouter Assignment  
</button>
```

```
@for(assignment of assignments; track assignment.nom) {  
...}
```

Dans le fichier TypeScript du père (Assignments.ts)

Une nouvelle propriété :

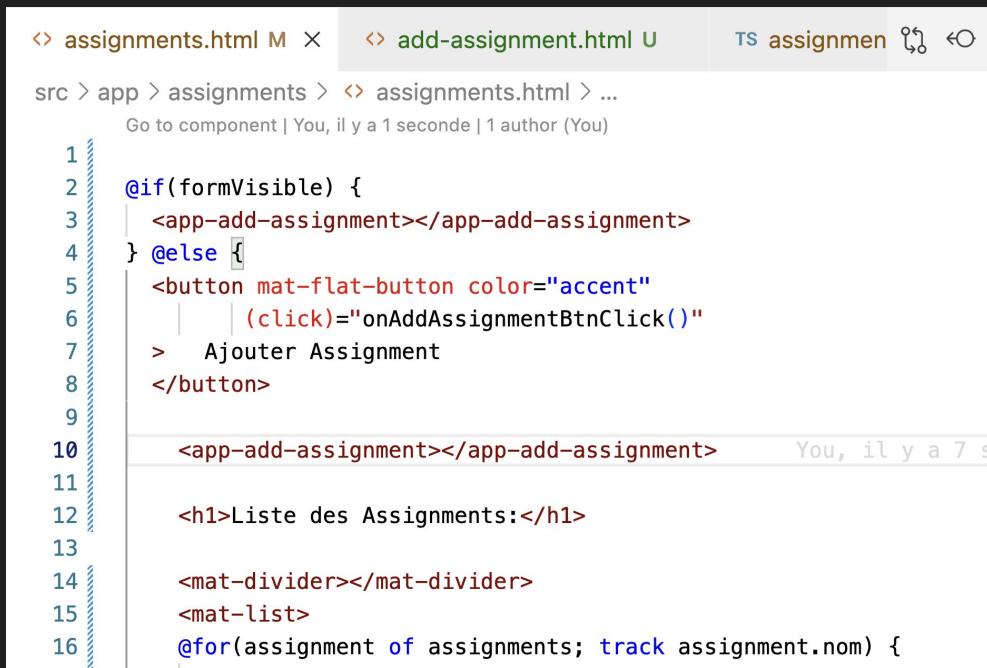
```
formVisible = false;
```

Un nouvelle méthode :

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

Du coup on peut utiliser ce flag pour montrer / cacher des parties du template

On ne va afficher QUE le formulaire (le composant fils add-assignment) que si forVisible vaut true, sinon on affichera le reste (le bouton d'ajout et la liste etc.)



The screenshot shows a code editor with three tabs: 'assignments.html M X', 'add-assignment.html U', and 'ts assignmen'. The 'assignments.html' tab is active. The code editor displays the 'assignments.html' file with the following content:

```
src > app > assignments > <> assignments.html > ...
Go to component | You, il y a 1 seconde | 1 author (You)

1  @if(formVisible) {
2    <app-add-assignment></app-add-assignment>
3  } @else [
4    <button mat-flat-button color="accent"
5      | (click)="onAddAssignmentBtnClick()"
6    > Ajouter Assignment
7    </button>
8
9
10   <app-add-assignment></app-add-assignment> You, il y a 7 s
11
12   <h1>Liste des Assignments:</h1>
13
14   <mat-divider></mat-divider>
15   <mat-list>
16     @for(assignment of assignments; track assignment.nom) {
```

Pour communiquer enfant vers père : utilisation de Output() et d'EventEmitter

Principe : le fils (**add-assignment**) va envoyer un événement (“J’ai ajouté un nouvel Assignment”) vers le père (**assignments**)

L'événement sera porteur de l'assignment ajouté.

Le père écoutera cet événement, récupérera le nouvel assignment et le rajoutera à la liste,

Puis on modifiera la valeur de la propriété **formVisible** pour à nouveau montrer la liste des assignments à jour...

```
du.ts # assignments.css M # add-assignment.css U TS assignments.ts M TS add-assignment.ts U >
src > app > assignments > add-assignment > TS add-assignment.ts > ...
9  @Component({
10    selector: 'app-add-assignment',
11    imports: [MatInputModule, MatFormFieldModule, MatButtonModule,
12      MatDatepickerModule, FormsModule],
13    templateUrl: './add-assignment.html',
14    styleUrls: ['./add-assignment.css'
15  })
16  export class AddAssignment {
17    @Output() nouvelAssignment = new EventEmitter<Assignment>();
18
19    // Pour le formulaire, une variable par champ
20    nomAssignment = "";
21    dateDeRendu!: Date;
22
23    onAjouterAssignment(event:any) {
24      // On crée un nouvel assignment
25      const nouvelAssignment: Assignment = new Assignment();
26      nouvelAssignment.nom = this.nomAssignment;
27      nouvelAssignment.dateDeRendu = this.dateDeRendu;
28      nouvelAssignment.rendu = false; // Par défaut, il n'est pas rendu
29
30      // On l'ajoute à la liste
31      this.nouvelAssignment.emit(nouvelAssignment);
32    }
33  }
34 }
```

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 :

Nom de l'événement/émetteur déclaré dans le fils

Méthode du père qui récupère les données reçues

Assignment passé par le fils

```
<!-- assignments.html -->
<!-- add-assignment.html -->
<!-- assignments.ts -->

1  @if(formVisible) {
2    <app-add-assignment (nouvelAssignment)="onNouvelAssignment($event)">
3      </app-add-assignment>
4    } @else {
5  }
```

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 connaître-)

```
@if(formVisible) {  
  <app-add-assignment (nou  
  </app-add-assignment>  
} @else {  
  <div class="container">  
    <button mat-flat-button  
           (click)="onA  
    >  
    Ajouter Assignment  
  </div>
```

The screenshot shows a development environment with a code editor and a browser preview.

Code Editor:

```
src > app > assignments > # assignments.css M X  
el.ts TS rendu.ts TS nonRendu.ts # assignments.css M X  
Hello les M1 Miage !!!  
Prof : Buffa Michel MODIFIEE  
Ajouter Assignment
```

Browser Preview:

- Header:** Hello les M1 Miage !!!
- Text:** Prof : Buffa Michel MODIFIEE
- Button:** Ajouter Assignment (with a red arrow pointing to it)
- Section:** Liste des Assignments:
 - 10/09/2025 L'assignment "Devoir Angular de Buffa" a été rendu.
 - 15/09/2025 A RENDRE : "Devoir Java de Mallet" !
- Footer:** 20/01/2026

Et si on clique le bouton, le formulaire apparaît

localhost:4200

Making Generative... GridForm AI Video Generato... orlarey/McpUI: FA... Remove Paywalls -... CCNT-DIGICREA 2... (258) LA FIN DU F... ACM SIGMM - the... SampleBy Promotion – CNU... Tous les favoris

Hello les M1 Miage !!!

Prof : Buffa Michel MODIFIEE

Ajout d'un Assignment

Nom de l'assignment

Choose a date*

MM/DD/YYYY

Ajouter

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.

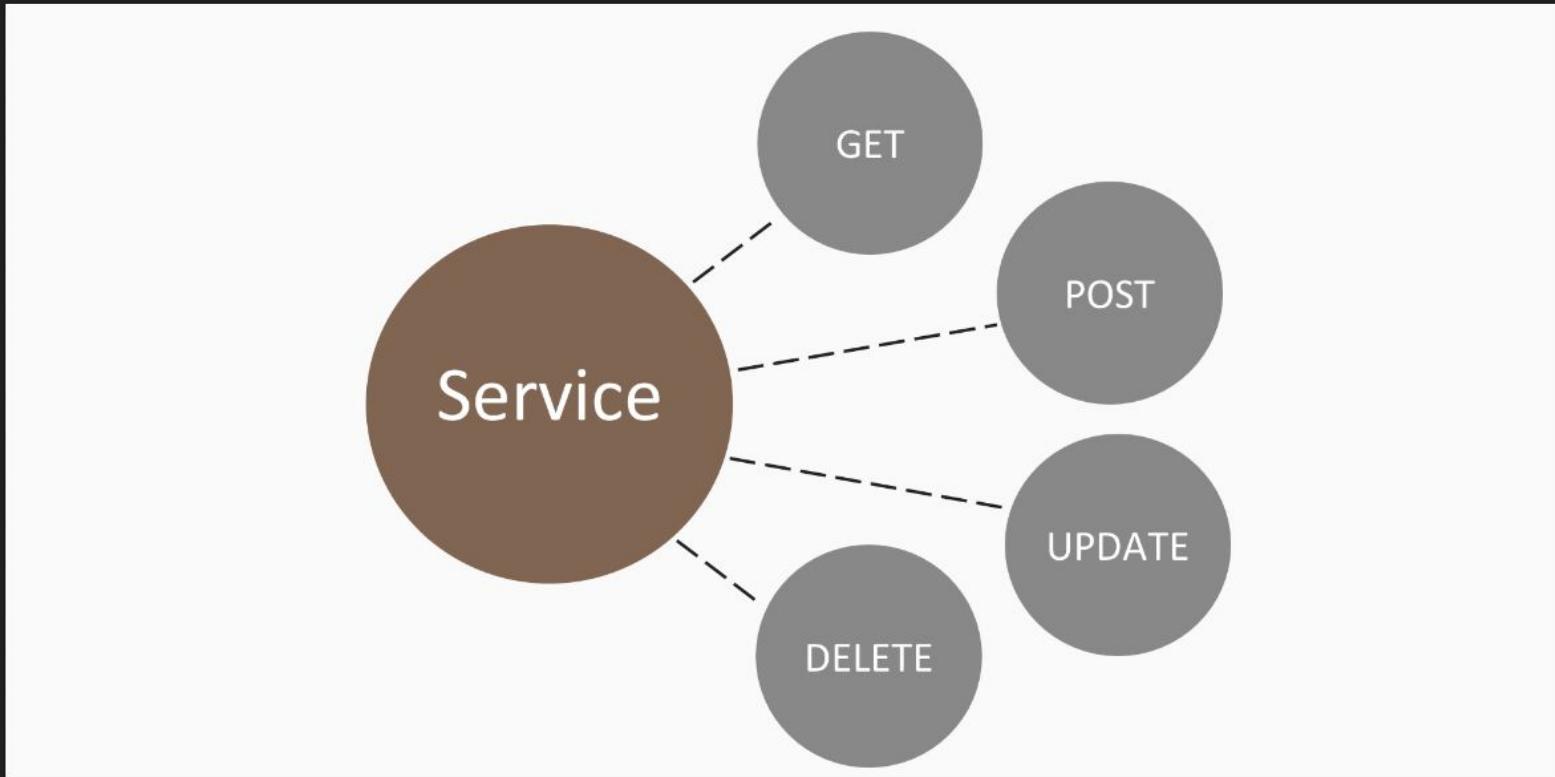
Les composants doivent avant tout “présenter” les données.

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 <nom-du-service>
```

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

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

- Terminal:** Shows the command `cd src/app/shared` followed by the Angular generate command: `(base) mbuffa2@buffy assignment-app % cd src/app/shared % ng g s --skipTests=true assignments`.
- File Explorer:** 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
- Code Editor:** Displays the content of `assignments.service.ts`:

```
TS assignments.service.ts X
src > app > shared > TS assignments.service.ts > ...
1 import { Injectable } from '@angular/core';
2
3 @Injectable({
4   providedIn: 'root'
5 })
6 export class AssignmentsService {
7
8   constructor() { }
9 }
```

Angular < 17 seulement! 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 {
```

TS assignments.ts

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

ts assignments.ts M

⚠ assignments.service.ts U X

ts assignm

src > app > shared > ⚠ assignments.service.ts > AssignmentService

```
1 import { Injectable } from '@angular/core';
2 import { Assignment } from '../assignments/assignment.model';
3
4 @Injectable({
5   providedIn: 'root'
6 })
7 export class AssignmentService {
8
9   assignments : Assignment[] = [
10   {
11     nom : "TP Angular",
12     dateDeRendu : new Date("2025-01-01"),
13     rendu : false
14   },
15   {
16     nom : "Projet Java",
17     dateDeRendu : new Date("2025-01-15"),
18     rendu : true
19   },
20   {
21     nom : "Examen HTML",
22     dateDeRendu : new Date("2025-01-31"),
23     rendu : false
24 }
25];
```

but d'un getter dans

sant assignments.ts vers le

ts assignments.ts M

⚠ assignments.service.ts U X

src > app > shared > ⚠ assignments.service.ts > ...

```
7   export class AssignmentService {
26
27   getAssignments() : Assignment[]{
28     return this.assignments;
29   }
30
31   constructor() {
32     console.log("Service Assignments créé !");
33   }
34 }
```

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: '  
8     <div>  
9       <h1>{{titre}}</h1>  
10      <table>  
11        <thead>  
12          <tr>  
13            <th>Titre</th>  
14            <th>Description</th>  
15            <th>Deadline</th>  
16          </tr>  
17        </thead>  
18        <tbody>  
19          <tr>  
20            <td>{{assignmentSelectionne.titre}}</td>  
21            <td>{{assignmentSelectionne.description}}</td>  
22            <td>{{assignmentSelectionne.deadline}}</td>  
23          </tr>  
24        </tbody>  
25      </table>  
26    </div>  
27  </div>  
28 </div>  
29 </div>  
30 </div>  
31 </div>  
32 </div>  
33 </div>  
34 </div>  
35 </div>  
36 </div>  
37 </div>  
38 </div>  
39 </div>  
40 </div>  
41 </div>  
42 </div>  
43 </div>  
44 </div>  
45 </div>  
46 </div>  
47 </div>  
48 </div>  
49 </div>  
50 </div>  
51 </div>  
52 </div>  
53 </div>  
54 </div>  
55 </div>  
56 </div>  
57 </div>  
58 </div>  
59 </div>  
60 </div>  
61 </div>  
62 </div>  
63 </div>  
64 </div>  
65 </div>  
66 </div>  
67 </div>  
68 </div>  
69 </div>  
70 </div>  
71 </div>  
72 </div>  
73 </div>  
74 </div>  
75 </div>  
76 </div>  
77 </div>  
78 </div>  
79 </div>  
80 </div>  
81 </div>  
82 </div>  
83 </div>  
84 </div>  
85 </div>  
86 </div>  
87 </div>  
88 </div>  
89 </div>  
90 </div>  
91 </div>  
92 </div>  
93 </div>  
94 </div>  
95 </div>  
96 </div>  
97 </div>  
98 </div>  
99 </div>  
100 </div>  
101 </div>  
102 </div>  
103 </div>  
104 </div>  
105 </div>  
106 </div>  
107 </div>  
108 </div>  
109 </div>  
110 </div>  
111 </div>  
112 </div>  
113 </div>  
114 </div>  
115 </div>  
116 </div>  
117 </div>  
118 </div>  
119 </div>  
120 </div>  
121 </div>
```

1 - Injection du service

2 - Utilisation du service

```
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: '  
8     <div>  
9       <h1>{{titre}}</h1>  
10      <table>  
11        <thead>  
12          <tr>  
13            <th>Titre</th>  
14            <th>Description</th>  
15            <th>Deadline</th>  
16          </tr>  
17        </thead>  
18        <tbody>  
19          <tr>  
20            <td>{{assignmentSelectionne.titre}}</td>  
21            <td>{{assignmentSelectionne.description}}</td>  
22            <td>{{assignmentSelectionne.deadline}}</td>  
23          </tr>  
24        </tbody>  
25      </table>  
26    </div>  
27  </div>  
28 </div>  
29 </div>  
30 </div>  
31 </div>  
32 </div>  
33 </div>  
34 </div>  
35 </div>  
36 </div>  
37 </div>  
38 </div>  
39 </div>  
40 </div>  
41 </div>  
42 </div>  
43 </div>  
44 </div>  
45 </div>  
46 </div>  
47 </div>  
48 </div>  
49 </div>  
50 </div>  
51 </div>  
52 </div>  
53 </div>  
54 </div>  
55 </div>  
56 </div>  
57 </div>  
58 </div>  
59 </div>  
60 </div>  
61 </div>  
62 </div>  
63 </div>  
64 </div>  
65 </div>  
66 </div>  
67 </div>  
68 </div>  
69 </div>  
70 </div>  
71 </div>  
72 </div>  
73 </div>  
74 </div>  
75 </div>  
76 </div>  
77 </div>  
78 </div>  
79 </div>  
80 </div>  
81 </div>  
82 </div>  
83 </div>  
84 </div>  
85 </div>  
86 </div>  
87 </div>  
88 </div>  
89 </div>  
90 </div>  
91 </div>  
92 </div>  
93 </div>  
94 </div>  
95 </div>  
96 </div>  
97 </div>  
98 </div>  
99 </div>  
100 </div>  
101 </div>  
102 </div>  
103 </div>  
104 </div>  
105 </div>  
106 </div>  
107 </div>  
108 </div>  
109 </div>  
110 </div>  
111 </div>  
112 </div>  
113 </div>  
114 </div>  
115 </div>  
116 </div>  
117 </div>  
118 </div>  
119 </div>  
120 </div>  
121 </div>
```

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 service présentée dans le 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

```
<mat-card-actions id="bottom">          You, il y a 1 seconde • Uncommitted changes  
  <mat-checkbox *ngIf="!assignmentTransmis.rendu" (click)="onAssignmentTransmis($event)">  
    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 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(messa  
28  
29         this.assignmentTransmis = null;  
30     }  
vous avez compris  
pourquoi ?
```

NOTE : si jamais vous êtes en mode “strict” (par défaut), 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.service
```

The screenshot shows the Visual Studio Code interface. The Explorer pane on the left lists files in the 'ASSIGNMENT-APP' folder, including 'assignments.component.ts', 'assignments.service.ts', 'logging.service.ts' (which is selected and highlighted in blue), 'rendu.directive.ts', 'app.component.css', and 'app.component.html'. The Editor pane on the right displays the code for 'logging.service.ts'. The Terminal pane at the bottom is empty.

```
TS assignment-detail.component.ts      TS logging.service.ts ×  
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 client front-end 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: "app.component.ts" and "app.html".

app.component.ts:

```
src > app > TS app.ts > ...
6 import { Assignments } from './assignments/assignments';
7
8 You, il y a 2 secondes | 1 author (You)
9 @Component({
10   selector: 'app-root',
11   imports: [RouterOutlet, MatButtonModule, MatIconModule, MatDi
12   templateUrl: './app.html',
13   styleUrls: ['./app.css'
14 })
15 export class App implements OnInit {
16   titre = "Application de gestion des assignments";
17
18   nomDuProf = 'Michel Buffa';
19
20   ngOnInit(): void {
21
22 }
```

app.html:

```
src > app > app.html M X # styles.css # app.css TS app.ts M
src > app > < app.html > h1 > nav > a
1 You, il y a 16 secondes | 1 author (You)
2 <!-- NE PAS UTILISER A HREF=...
3 <h1><a href="....">{{titre}}</a></h1>
4
5 <h1>
6   <nav>
7     <a routerLink="/home">{{titre}}</a>
8   </nav>
9 </h1>
10 <app-assignments></app-assignments>
11
```

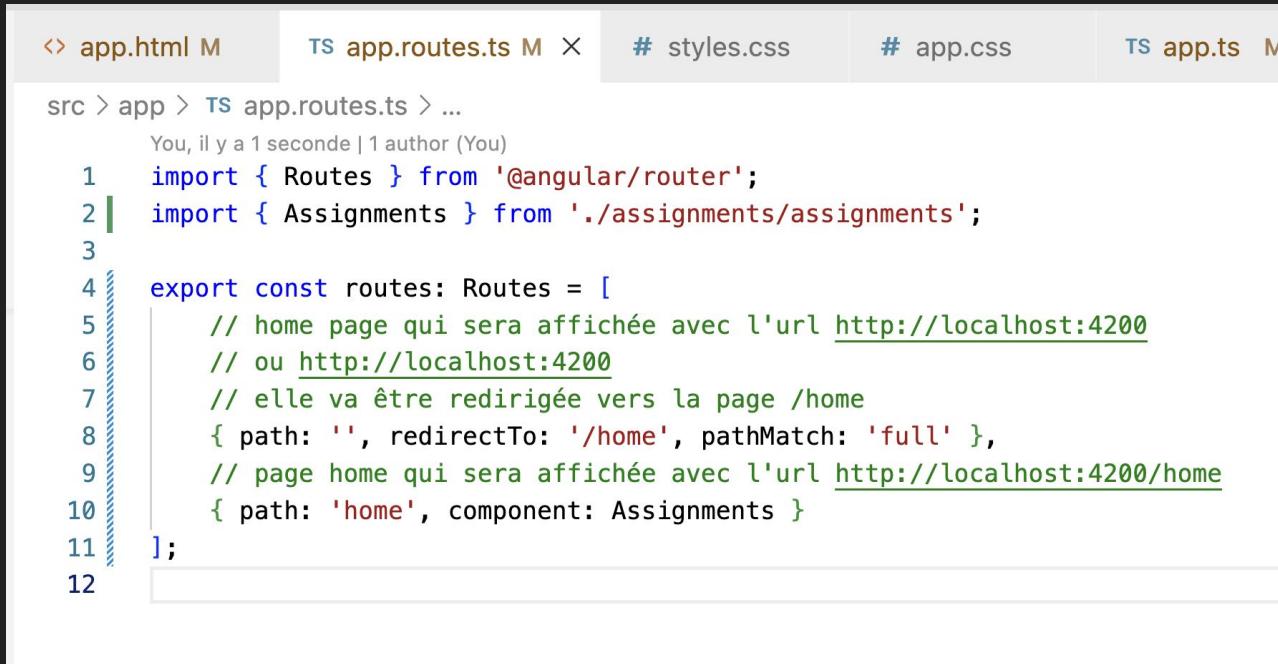
On teste l'application

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 se passe dans app.route.ts ! (nouveauté Angular 17)



The screenshot shows a code editor with several tabs at the top: app.html, app.routes.ts (which is the active tab), styles.css, app.css, and app.ts. The code editor displays the content of the app.routes.ts file. The code defines a routes array with two entries: a redirect to the /home page and a component assignment.

```
src > app > TS app.routes.ts > ...
You, il y a 1 seconde | 1 author (You)
1 import { Routes } from '@angular/router';
2 import { Assignments } from './assignments/assignments';
3
4 export const routes: Routes = [
5   // home page qui sera affichée avec l'url http://localhost:4200
6   // ou http://localhost:4200
7   // elle va être redirigée vers la page /home
8   { path: '', redirectTo: '/home', pathMatch: 'full' },
9   // page home qui sera affichée avec l'url http://localhost:4200/home
10  { path: 'home', component: Assignments }
11];
12
```

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

On teste l'application !

<> app.html M X TS app.routes.ts M # styles.css # app.css TS app.ts M

```
src > app > <--> localhost:4200/home  
Applications Repl.it - Entirelavor... VerbalExpressions... Badassebikes E  
1 You, il  
2 <!--  
3 <h1>  
4 --->  
5 <h1>  
6 |  
7 | Jan 17, 2020  
8 |  
9 |</h1>  
10 |  
11 |<kroute  
12 |  
13 |<!--  
14 |<app-  
15 | --->  
  


# Application de gestion des devoirs



Jan 17, 2020



Cliquez ici ! /home de



Devoir intitulé TP1 sur WebComponents, un lecteur audio



---



Dec 15, 2020



Le devoir TP2 sur Angular, un joli gestionnaire de devoirs



---



Jan 4, 2021



Le devoir TP3 sur Angular, utilisation du router et de WebComponents

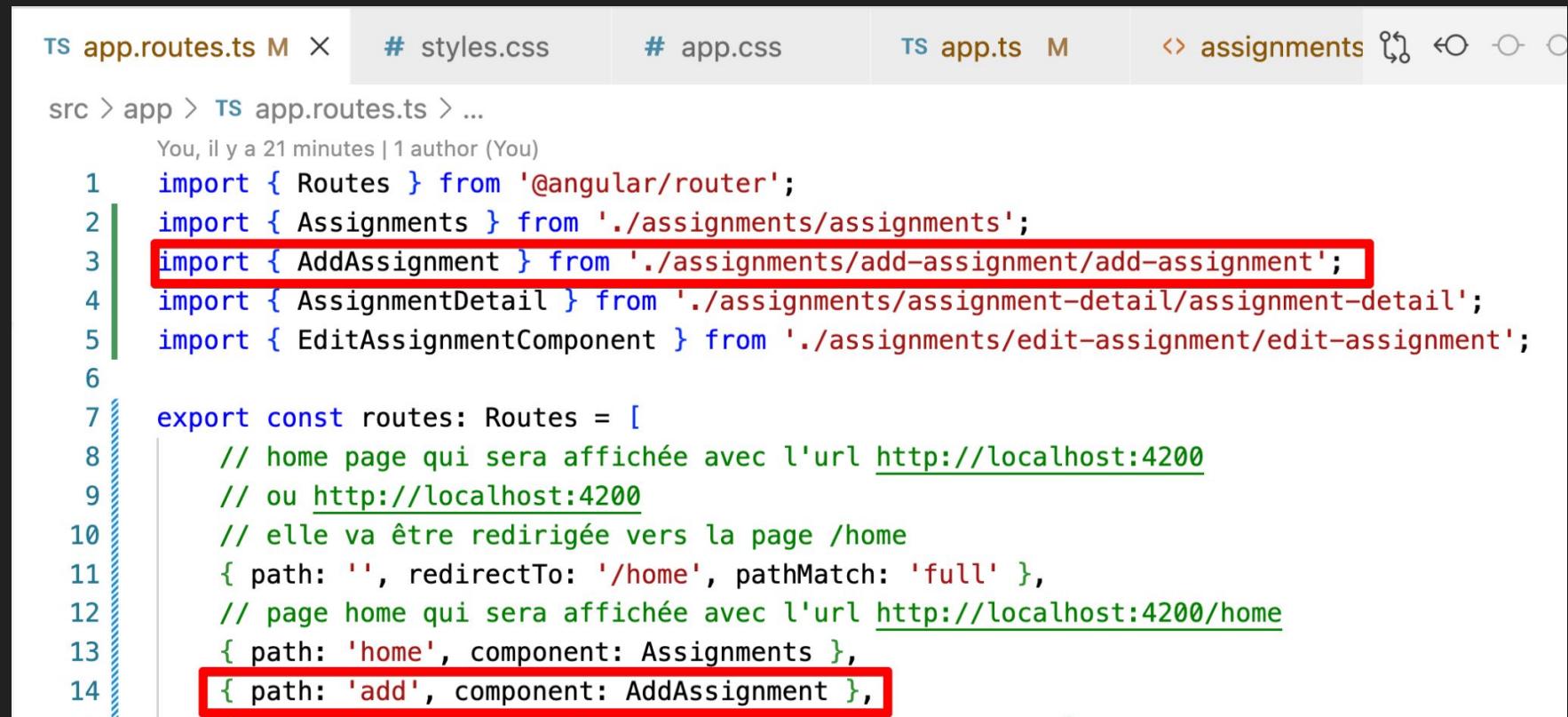

```

The screenshot shows a code editor with several tabs at the top: 'app.html M', 'TS app.routes.ts M', '# styles.css', '# app.css', 'TS app.ts M X'. The main content area displays the 'app.ts' file under the 'src > app > TS app.ts' path. The file contains the following code:

```
You, il y a 11 secondes | 1 author (You)
1 import { Component, OnInit, signal } from '@angular/core';
2 import { RouterOutlet, RouterLink } from '@angular/router';
3 import { MatButtonModule } from '@angular/material/button';
4 import { MatIconModule } from '@angular/material/icon';
5 import { MatDividerModule } from '@angular/material/divider';
6 import { Assignments } from './assignments/assignments';
7
You, il y a 11 secondes | 1 author (You)
8 @Component({
9   selector: 'app-root',
10  imports: [RouterOutlet, RouterLink,
11    MatButtonModule, MatIconModule, MatDividerModule, Assignments],
12  templateUrl: './app.html',
13  styleUrls: ['./app.css']
14 })
15 export class App implements OnInit {
16   titre = "Application de gestion des assignments";
17 }
```

The 'RouterOutlet' and 'RouterLink' imports from line 2 and the 'RouterOutlet' import from line 10 are highlighted with red boxes.

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



The screenshot shows a code editor with multiple tabs at the top: 'app.routes.ts M X', '# styles.css', '# app.css', 'app.ts M', and 'assignments'. The 'app.routes.ts' tab is active. Below the tabs, a message from 'You' is displayed: 'You, il y a 21 minutes | 1 author (You)'. The code editor displays the following TypeScript code:

```
src > app > TS app.routes.ts > ...
You, il y a 21 minutes | 1 author (You)
1 import { Routes } from '@angular/router';
2 import { Assignments } from './assignments/assignments';
3 import { AddAssignment } from './assignments/add-assignment/add-assignment'; // Line 3 is highlighted with a red box
4 import { AssignmentDetail } from './assignments/assignment-detail/assignment-detail';
5 import { EditAssignmentComponent } from './assignments/edit-assignment/edit-assignment';
6
7 export const routes: Routes = [
8     // home page qui sera affichée avec l'url http://localhost:4200
9     // ou http://localhost:4200
10    // elle va être redirigée vers la page /home
11    { path: '', redirectTo: '/home', pathMatch: 'full' },
12    // page home qui sera affichée avec l'url http://localhost:4200/home
13    { path: 'home', component: Assignments },
14    { path: 'add', component: AddAssignment }, // Line 14 is highlighted with a red box
```

Modification du template de assignments.html

On n'a plus besoin du **@if ...
@else** 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...

The screenshot shows a code editor with several files open:

- assignments.html**: The template file containing the following code:

```
You, il y a 2 secondes | 1 author (You)
1 <!-- Le @if ... @else... devient inutile, grace
2 au router, on affichera le composant d'ajout que si
3 l'URL (la route) finit par /add -->
4 <!--if(formVisible) {
5 <app-add-assignment (nouvelAssignment)="onNouvelAssignment($event)">
6 </app-add-assignment>
7 } @else {
8   -->
9   <div
```
- assignments.ts**: The corresponding TypeScript file:

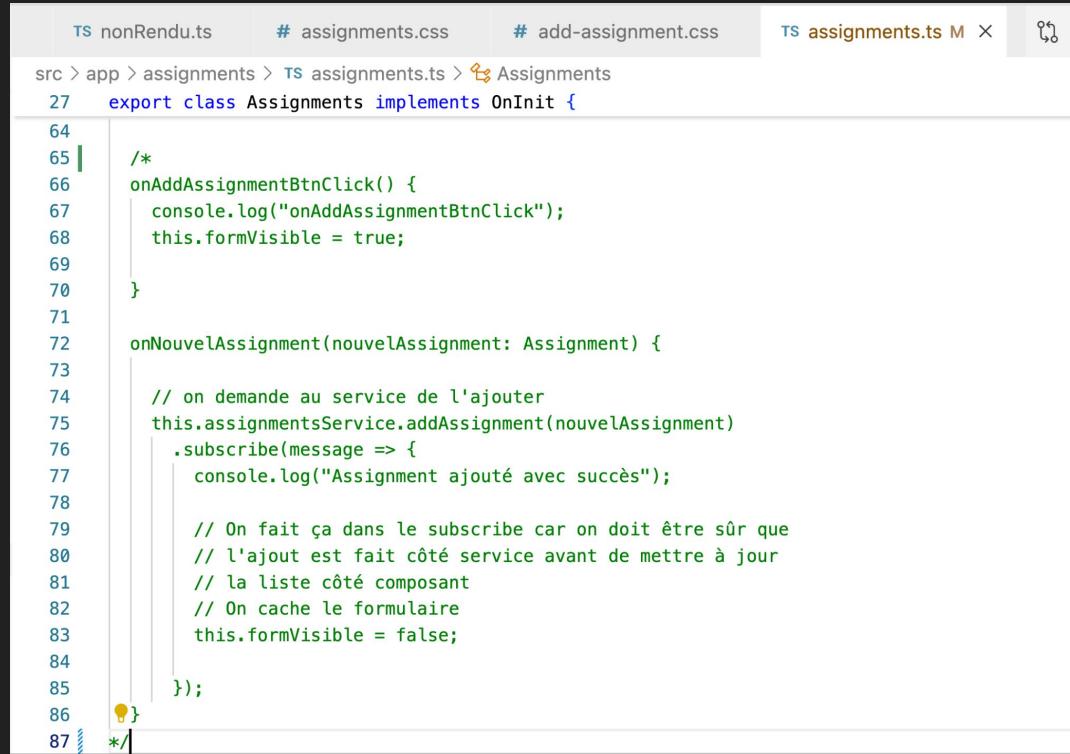
```
You, il y a 1 minute | 1 author (You)
1 <!-- Le @if ... @else... devient inutile, grace
2 au router, on affiche le composant d'ajout que si
3 l'URL (la route) finit par /add -->
4 <!--if(formVisible) {
5   <app-add-assignment
6   </app-add-assignment>
7 } @else {
8   <div
```
- add-assignment.html**: Another template file showing a button with **routerLink**:
- add-assignment.ts**: The corresponding TypeScript file for the **add-assignment.html** template.

Annotations in red highlight specific parts of the code:

- An annotation points to the removed **<!--if(formVisible) {** block in **assignments.html** with the text: **On n'a plus besoin de tester formVisible**.
- An annotation points to the **import { RouterLink } from '@angular/router';** line in **add-assignment.ts** with the text: **Penser à ajouter l'import de RouterLink dans le typescript du composant pour que <a routerLink> fonctionne !**.
- An annotation points to the **RouterLink** attribute in the **<a>** tag in **add-assignment.html** with the text: **RouterLink**.

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

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



```
src > app > assignments > TS assignments.ts > 📄 Assignments
27  export class Assignments implements OnInit {
64
65  /*
66  onAddAssignmentBtnClick() {
67    console.log("onAddAssignmentBtnClick");
68    this.formVisible = true;
69
70  }
71
72  onNouvelAssignment(nouvelAssignment: Assignment) {
73
74    // on demande au service de l'ajouter
75    this.assignmentsService.addAssignment(nouvelAssignment)
76      .subscribe(message => {
77        console.log("Assignment ajouté avec succès");
78
79        // On fait ça dans le subscribe car on doit être sûr que
80        // l'ajout est fait côté service avant de mettre à jour
81        // la liste côté composant
82        // On cache le formulaire
83        this.formVisible = false;
84
85      });
86
87  */

```

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

On va utiliser `assignments.service` directement dans le composant d'ajout

```
# add-assignment.css      TS assignments.ts M      TS app.config.ts      TS add-assignment.ts M ×

src > app > assignments > add-assignment > TS add-assignment.ts > ...
10   ...
11
12   export class AddAssignment {
13     //@Output() nouvelAssignment = new EventEmitter<Assignment>();
14
15     // Pour le formulaire, une variable par champ
16     nomAssignment = "";
17     dateDeRendu!: Date;
18
19
20     // on a injecté le service dans le constructeur
21     constructor(private assignmentsService: AssignmentsService) { }
22
23
24     onAjouterAssignment(event: any) {
25       // On crée un nouvel assignment
26       const nouvelAssignment: Assignment = new Assignment();
27       nouvelAssignment.nom = this.nomAssignment;
28       nouvelAssignment.dateDeRendu = this.dateDeRendu;
29       nouvelAssignment.rendu = false; // Par défaut, il n'est pas rendu
30
31       //this.nouvelAssignment.emit(nouvelAssignment);
32       this.assignmentsService.addAssignment(nouvelAssignment)
33         .subscribe(message => {
34           console.log(message);
35         });
36     }
37
38   }
39
40
41 }
```

Inutile

Inutile

Le composant d'ajout fait directement l'ajout en utilisant le service !

Puis vérifier que l'application fonctionne comme avant en cliquant sur le bouton d'ajout d'un assignment

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.routes.ts` :

```
export const routes: Routes = [
  // home page qui sera affichée avec l'url http://localhost:4200
  // ou http://localhost:4200
  // elle va être redirigée vers la page /home
  { path: '', redirectTo: '/home', pathMatch: 'full' }
  // page home qui sera affichée avec l'url http://localhost:4200/home
  { path: 'home', component: Assignments },
  { path: 'add', component: AddAssignment },
  { path: 'assignment/:id', component: AssignmentDetail }
];
```

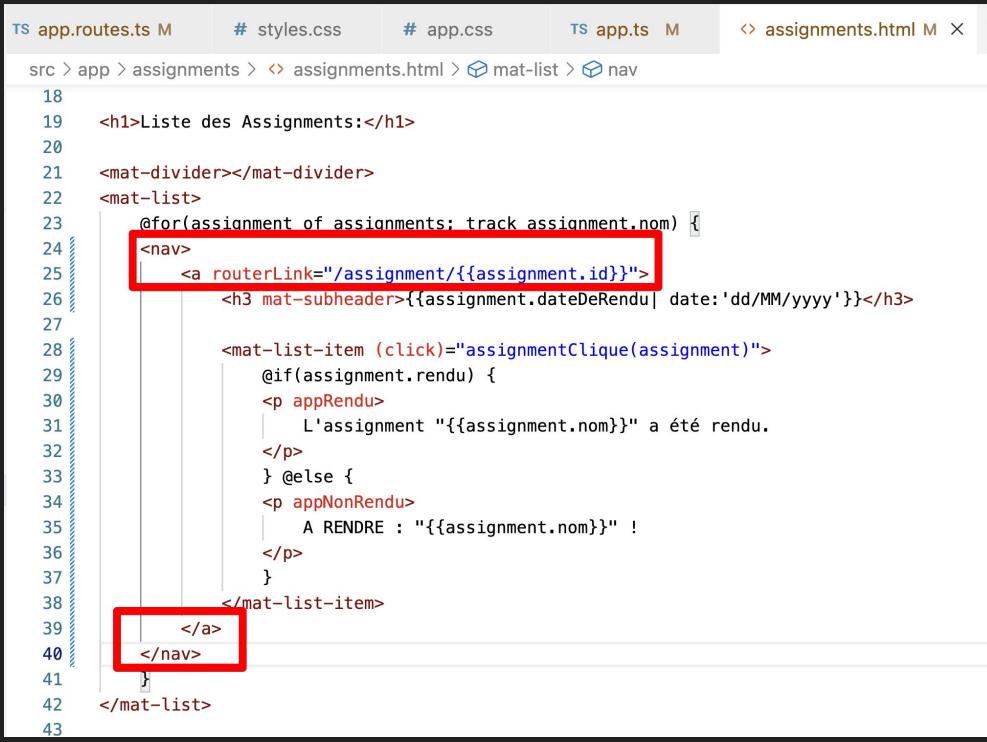
*:id ici indique une partie variable. On aurait pu prendre n'importe quel nom au lieu de 'id'
on manipulera cet élément depuis le fichier typescript du composant*

: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.html`



```
src > app > assignments > assignments.html > mat-list > nav
18
19 <h1>Liste des Assignments:</h1>
20
21 <mat-divider></mat-divider>
22 <mat-list>
23   @for(assignment of assignments; track assignment.nom) {
24     <nav>
25       <a routerLink="/assignment/{{assignment.id}}">
26         <h3 mat-subheader>{{assignment.dateDeRendu | date:'dd/MM/yyyy'}}</h3>
27
28         <mat-list-item (click)="assignmentClique(assignment)">
29           @if(assignment.rendu) {
30             <p appRendu>
31               L'assignment "{{assignment.nom}}" a été rendu.
32             </p>
33           } @else {
34             <p appNonRendu>
35               A RENDRE : "{{assignment.nom}}" !
36             </p>
37           }
38         </mat-list-item>
39       </a>
40     </nav>
41   </mat-list>
42
43
```

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

```
export class Assignment {
  id!: number;
  nom!: string;
  dateDeRendu!: Date;
  rendu!: boolean;
}
```

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)

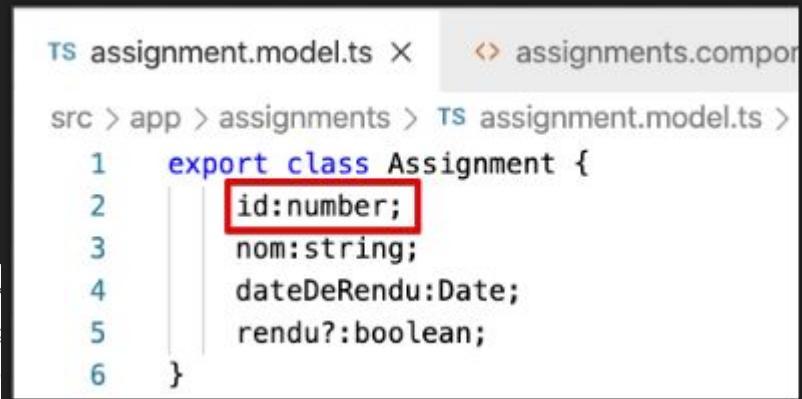
```
1 export class Assignment
2 {
3   id!: number;
4   nom!:string;
5   dateDeRendu!:Date;
6   rendu!:boolean;
7 }
```

```
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 ];
```

Ajout d'une propriété id aux assignments

Il faut mettre à jour aussi le composant d'ajout pour que lors de l'ajout on génère un id aléatoire (qui ait des chances d'être unique):

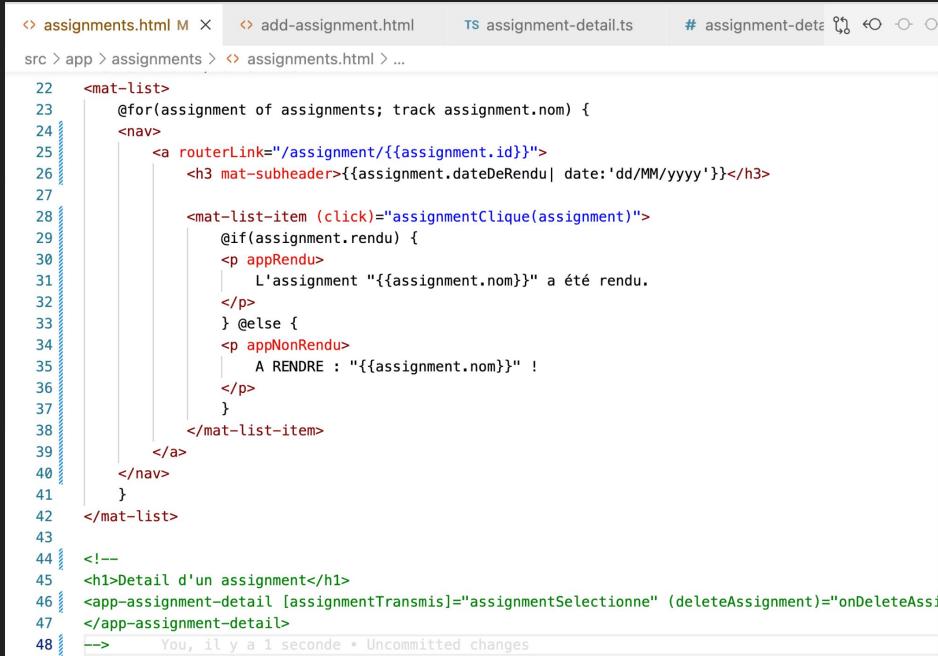
```
# add-assignment.css      TS assignments.ts M      TS app.config.ts      TS add-assignment.ts M >
src > app > assignments > add-assignment > TS add-assignment.ts > AddAssignment > onAjouterAssignment
21  export class AddAssignment {
22
23    onAjouterAssignment(event: any) {
24      // On crée un nouvel assignment
25      const nouvelAssignment: Assignment = new Assignment();
26      nouvelAssignment.id = Math.floor(Math.random() * 100000); // un id au hasard
27      nouvelAssignment.nom = this.nomAssignment;
28      nouvelAssignment.dateDeRendu = this.dateDeRendu;
29      nouvelAssignment.rendu = false; // Par défaut, il n'est pas rendu
30
31      //this.nouvelAssignment.emit(nouvelAssignment);
32      this.assignmentsService.addAssignment(nouvelAssignment)
33        .subscribe(message => [
34          console.log(message),
35        ]);
36
37      You, il y a 1 seconde • Uncommitted changes
38
39    }
40  }
41
42  You, il y a 1 seconde • Uncommitted changes
43
44  You, il y a 1 seconde • Uncommitted changes
45}
```



```
TS assignment.model.ts ×  ↗ assignments.compon
src > app > assignments > TS assignment.model.ts >
1  export class Assignment {
2    id:number;
3    nom:string;
4    dateDeRendu:Date;
5    rendu?:boolean;
6 }
```

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 several tabs at the top: 'assignments.html M X', 'add-assignment.html', 'TS assignment-detail.ts', and '# assignment-deta'. The main content area displays the 'assignments.html' template code:

```
src/app/assignments/assignments.html ...
22  <mat-list>
23    @for(assignment of assignments; track assignment.nom) {
24      <nav>
25        <a routerLink="/assignment/{{assignment.id}}">
26          <h3 mat-subheader>{{assignment.dateDeRendu| date:'dd/MM/yyyy'}}</h3>
27
28          <mat-list-item (click)="assignmentClique(assignment)">
29            @if(assignment.rendu) {
30              <p appRendu>
31                L'assignment "{{assignment.nom}}" a été rendu.
32              </p>
33            } @else {
34              <p appNonRendu>
35                A RENDRE : "{{assignment.nom}}" !
36              </p>
37            }
38          </mat-list-item>
39        </a>
40      </nav>
41    }
42  </mat-list>
43
44  <!--
45  <h1>Detail d'un assignment</h1>
46  <app-assignment-detail [assignmentTransmis]="assignmentSelectionne" (deleteAssignment)="onDeleteAssignment(assignment)">
47  </app-assignment-detail>
48  --> You, il y a 1 seconde * Uncommitted changes
```

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 several tabs at the top: `etail.ts`, `# assignment-detail.css`, `<> assignment-detail.html`, `TS assignments.service.ts M X`. The `assignments.service.ts` tab is active. Below the tabs, there is a breadcrumb navigation: `src > app > shared > TS assignments.service.ts > 📁 AssignmentsService > ⚙️ updateAssignment`. The main content area displays the `assignments.service.ts` file with line numbers 8 through 39. The code defines a class `AssignmentsService` with a method `getAssignment` that returns an observable assignment by its ID.

```
etail.ts      # assignment-detail.css      <> assignment-detail.html      TS assignments.service.ts M X
src > app > shared > TS assignments.service.ts > 📁 AssignmentsService > ⚙️ updateAssignment
8   export class AssignmentsService {
33
34     // Get assignment by id
35     getAssignment(id: number): Observable<Assignment | undefined> {
36       const a = this.assignments.find(a => a.id === id);
37       return of(a);
38     }
39   }
```

Mais toujours pas de “vue des détails”

On va modifier `assignments.detail.ts` :

```
ts assignment-detail.ts M × # assignment-detail.css      < assignment-detail.html      TS assign
src > app > assignments > assignment-detail > TS assignment-detail.ts > AssignmentDetail
8 import { ActivatedRoute } from '@angular/router';
You, il y a 30 secondes | 1 author (You)
9 @Component({
10   selector: 'app-assignment-detail',
11   imports: [MatCardModule, MatCheckboxModule, DatePipe, MatButton],
12   templateUrl: './assignment-detail.html',
13   styleUrls: ['./assignment-detail.css'
14 })
15 export class AssignmentDetail implements OnInit {
16   On l'enlève (plus d'assignment transmis par
17   //@Input() ← le père)
18   assignmentTransmis?: Assignment;
19   @Output() deleteAssignment = new EventEmitter<Assignment>();
20
21   constructor(private assignmentsService: AssignmentsService,
22   private route: ActivatedRoute) {}
23
24   ngOnInit(): void {
25     this.getAssignment();
26   }
27
28   You, il y a 6 jours • Jusqu'à la correction du TP2
29   getAssignment() {
30     // On va utiliser ActivatedRoute pour lire l'id dans l'URL
31     const id = +this.route.snapshot.params['id']; // le + convertit en number
32
33     this.assignmentsService.getAssignment(id)
34       .subscribe(a => {
35         this.assignmentTransmis = a;
36       });
37   }
38 }
```

On l'enlève (plus d'assignment transmis par le père)

Pour récupérer l'id dans l'URL!

Utilisation du service pour récupérer l'assignment par son id

Et on teste l'application !

localhost:4200/assignment/1

Application de gestion des devoirs à rendre (Assignments)

TP1 sur WebComponents, un lecteur audio amélioré

Jan 17, 2020

DELETE

localhost:4200/assignment/2

Application de gestion des devoirs à rendre (Assignments)

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

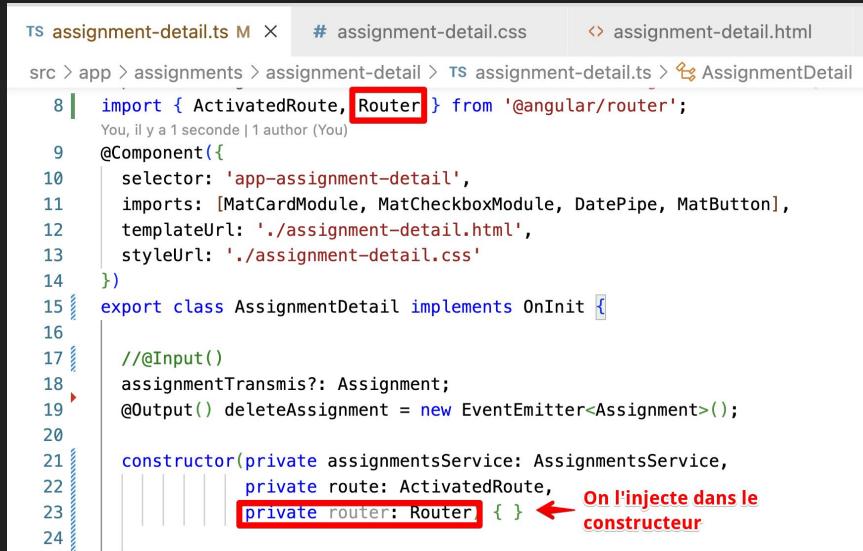
Dec 15, 2020

Devoir rendu

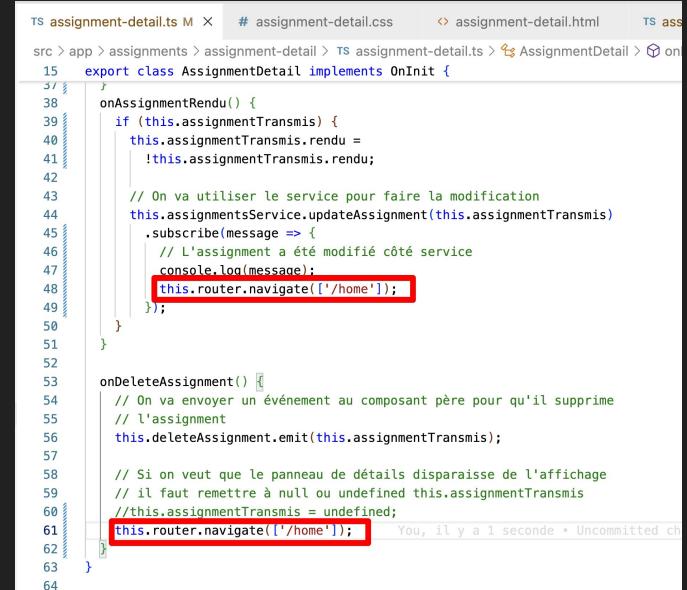
Et si on clique sur DELETE ou sur la checkbox “rendu” dans la vue détails ?

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

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



```
TS assignment-detail.ts M # assignment-detail.css < assignment-detail.html
src > app > assignments > assignment-detail > TS assignment-detail.ts > AssignmentDetail
8 | import { ActivatedRoute, Router } from '@angular/router';
9 | @Component({
10 |   selector: 'app-assignment-detail',
11 |   imports: [MatCardModule, MatCheckboxModule, DatePipe, MatButton],
12 |   templateUrl: './assignment-detail.html',
13 |   styleUrls: ['./assignment-detail.css'
14 | })
15 export class AssignmentDetail implements OnInit {
16
17   // @Input()
18   assignmentTransmis?: Assignment;
19   @Output() deleteAssignment = new EventEmitter<Assignment>();
20
21   constructor(private assignmentsService: AssignmentsService,
22     private route: ActivatedRoute,
23     private router: Router) { } ← On l'injecte dans le constructeur
24 }
```

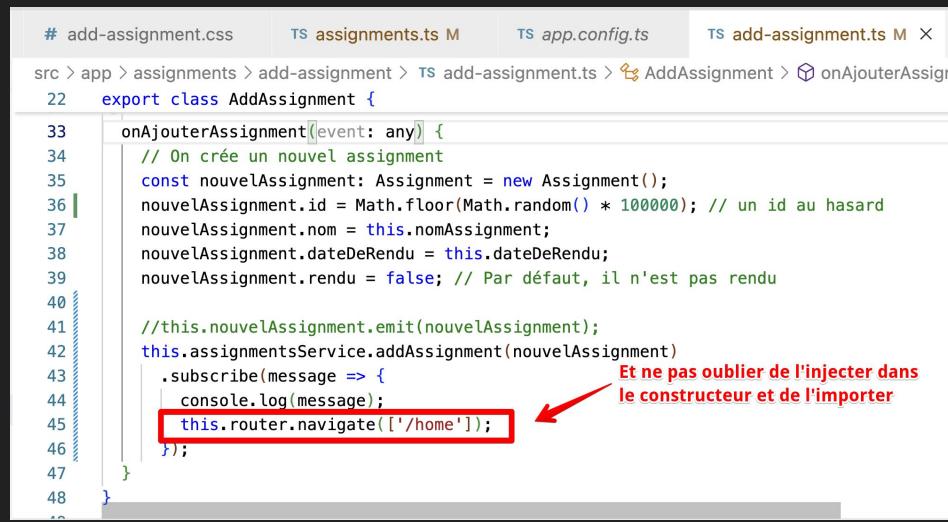


```
TS assignment-detail.ts M # assignment-detail.css < assignment-detail.html
src > app > assignments > assignment-detail > TS assignment-detail.ts > AssignmentDetail > onAssignmentRendu()
15 export class AssignmentDetail implements OnInit {
31
38   onAssignmentRendu() {
39     if (this.assignmentTransmis) {
40       this.assignmentTransmis.rendu =
41         !this.assignmentTransmis.rendu;
42
43       // On va utiliser le service pour faire la modification
44       this.assignmentsService.updateAssignment(this.assignmentTransmis)
45         .subscribe(message => {
46           // L'assignment a été modifié côté service
47           console.log(message);
48           this.router.navigate(['/home']);
49         });
50
51
52
53   onDeleteAssignment() {
54     // On va envoyer un événement au composant père pour qu'il supprime
55     // l'assignment
56     this.deleteAssignment.emit(this.assignmentTransmis);
57
58     // Si on veut que le panneau de détails disparaîsse de l'affichage
59     // il faut remettre à null ou undefined this.assignmentTransmis
60     // this.assignmentTransmis = undefined;
61     this.router.navigate(['/home']); ← On va utiliser le service pour faire la modification
62
63
64 }
```

Penser aussi à rajouter router.navigate après un ajout, dans le composant AddAssignment

Après un ajout, on veut retourner directement à la page d'accueil....

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



```
# add-assignment.css      TS assignments.ts M      TS app.config.ts      TS add-assignment.ts M ×
src > app > assignments > add-assignment > TS add-assignment.ts > AddAssignment > onAjouterAssignment
22 export class AddAssignment {
23   onAjouterAssignment(event: any) {
24     // On crée un nouvel assignment
25     const nouvelAssignment: Assignment = new Assignment();
26     nouvelAssignment.id = Math.floor(Math.random() * 100000); // un id au hasard
27     nouvelAssignment.nom = this.nomAssignment;
28     nouvelAssignment.dateDeRendu = this.dateDeRendu;
29     nouvelAssignment.rendu = false; // Par défaut, il n'est pas rendu
30
31     //this.nouvelAssignment.emit(nouvelAssignment);
32     this.assignmentsService.addAssignment(nouvelAssignment)
33       .subscribe(message => {
34         console.log(message);
35         this.router.navigate(['/home']);
36       });
37   }
38 }
```

Et ne pas oublier de l'injecter dans le constructeur et de l'importer

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;
```

Mais de quoi s'agit-il ?

On utilisera un objet `route: ActivatedRoute` qu'on a déjà vu, au travers de ses propriétés `route.queryParams` ou `route.fragment` si les valeurs peuvent changer d'une navigation sur la page à l'autre....

A CORRIGER : en Angular 9, `this.route.snapshot.queryParams` etc.

```
this.route.queryParams.subscribe(  
    next: params => console.log(params)  
);  
  
this.route.fragment.subscribe(  
    next: fragment => console.log(fragment)  
);
```

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

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

```
cd src/assignments  
ng g c --skip-tests=true edit-assignment
```



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

```
@if(assignment) {  
  
  <div class="container">  
  
    <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>  
  </div>
```

```
    <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 { FormsModule } from '@angular/forms';
import { MatInputModule } from '@angular/material/input';
import { MatButtonModule } from '@angular/material/button';
import { MatFormFieldModule } from '@angular/material/form-field';
import { MatDatepickerModule } from '@angular/material/date-picker';
import { provideNativeDateAdapter } from '@angular/material/core';
import { Assignment } from '../assignment.model';
import { AssignmentsService } from '../../shared/assignments.service';
import { ActivatedRoute, Router } from '@angular/router';

@Component({
  selector: 'app-edit-assignment',
  standalone: true,
  providers: [provideNativeDateAdapter()],
  imports: [
    FormsModule,
    MatInputModule,
    MatFormFieldModule,
    MatDatepickerModule,
    MatButtonModule,
  ],
  templateUrl: './edit-assignment.html',
  styleUrls: ['./edit-assignment.css'],
})
```

```
export class EditAssignmentComponent implements OnInit {
  assignment: Assignment | undefined;
  // Pour les champs de formulaire
  nomAssignment = '';
  dateDeRendu?: Date = undefined;

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

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

  getAssignment() {
    // On va utiliser ActivatedRoute pour lire l'id dans l'URL
    const id = +this.route.snapshot.params['id'];
    this.assignmentsService.getAssignment(id)
      .subscribe(a => {
        this.assignment = a;
        if(a === undefined) return;
        this.nomAssignment = a.nom;
        this.dateDeRendu = a.dateDeRendu;
      });
  }
}
```

Nouveau code pour la partie métier (suite)

```
onSaveAssignment() {
  if (!this.assignment) return;
  if (this.nomAssignment === '' || this.dateDeRendu === undefined) 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

```
export const routes: Routes = [
  // home page qui sera affichée avec l'url http://localhost:4200
  // ou http://localhost:4200
  // elle va être redirigée vers la page /home
  { path: '', redirectTo: '/home', pathMatch: 'full' },
  // page home qui sera affichée avec l'url http://localhost:4200/home
  { path: 'home', component: Assignments },
  { path: 'add', component: AddAssignment },
  { path: 'assignment/:id', component: AssignmentDetail },
  { path: 'assignment/:id/edit', component: EditAssignmentComponent }
];
```

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



Ajout d'un bouton EDIT dans le composant détail

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

```
assignment-detail.html M assignment-detail.ts M # assignment-detail.css
c > app > assignments > assignment-detail > assignment-detail.html > mat-card > mat-card-actions.container
You, il y a 1 seconde | 1 author (You)
1  @if(assignmentTransmis) {
2    <mat-card appearance="outlined">
3      <mat-card-header>
4        <mat-card-title>Details de l'assignment</mat-card-title>
5        <mat-card-subtitle>On pourra ajouter image etc.</mat-card-subtitle>
6      </mat-card-header>
7      <mat-card-content>
8        <p>
9          Cet assignment a pour nom : {{assignmentTransmis.nom}} <br>
10         et pour date de rendu :
11         {{assignmentTransmis.dateDeRendu | date:'dd/MM/yyyy'}} <br> Route relative.
12         Blah Blah Blah.....
13        </p>
14      </mat-card-content>
15      <mat-card-actions class="container">
16        @if(!assignmentTransmis.rendu) {
17          <mat-checkbox (click)="onAssignmentRendu()">Rendu</mat-checkbox>
18        }
19        <nav>
20          <button mat-stroked-button color="primary" routerLink="edit">EDIT</button>
21        </nav>
22        <button mat-flat-button color="warn" (click)="onDeleteAssignment()">DELETE</button>
23      </mat-card-actions>
24    </mat-card>
25 }
```



localhost:4200/assignment/1

Applications Links - GDEMU (6) SEGA Dreamcast GD... (6) SEGA Dreamcast GD... (6) Dreamcast GD... (6) Dreamcast GD... Repl.it - Entirelavoro... Autres favoris

Application de gestion des devoirs à rendre (Assignments)

TP1 sur WebComponents, un lecteur audio amélioré

Jan 17, 2020

Edit DELETE



localhost:4200/assignment/1/edit

Applications Links - GDEMU (6) SEGA Dreamcast GD... (6) SEGA Dreamcast GD... (6) Dreamcast GD... (6) Dreamcast GD... Repl.it - Entirelavoro... VerbalExpressions... Badassebikes Box... Liens

Application de gestion des devoirs à rendre (Assignments)

Edition de l'assignment TP1 sur WebComponents, un lecteur audio amélioré

Edition du nom Edition de la date Sauver

Ajout d'un bouton EDIT dans le composant détail

Autres possibilités avec une route “absolue” et donc de la navigation dynamique

```
<nav>
  <button mat-stroked-button color="primary"
    [routerLink]=["'/assignment', assignmentTransmis.id, 'edit']">EDIT
  </button>
</nav>
```

```
<nav>
  <button mat-stroked-button color="primary"
    routerLink="/assignment/{{assignmentTransmis.id}}/edit">EDIT
  </button>
</nav>
```

Troisième possibilité par router.navigate(...)

On va supprimer la navigation depuis le template HTML et utiliser un événement click sur le bouton EDIT. On appellera une méthode dans le TypeScript **onClickEdit()**

```
<nav>
  <button mat-stroked-button color="primary"
    | | (click)="onClickEdit()">EDIT
  </button>
</nav>
```

Et dans le assignment-detail.ts :

```
onClickEdit() {
  if(!this.assignmentTransmis) return;
  this.router.navigate(['/assignment', this.assignmentTransmis.id, 'edit']);
}
```

```
onClickEdit() {
  if(!this.assignmentTransmis) return;
  this.router.navigate(['~/assignment/${this.assignmentTransmis.id}/edit']);
}
```

Ou encore, autre possibilité :

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 illustrates the process of retrieving query parameters and fragments in a component. On the left, the code editor shows `edit-assignment.component.ts` with the following code:

```
19
20  ngOnInit(): void {
21      this.getAssignment();
22
23      // affichage des queryParams
24      console.log("Query Params :");
25      console.log(this.route.snapshot.queryParams);
26      console.log("Fragment :");
27      console.log(this.route.snapshot.fragment);
28
29 }
```

A red box highlights the code from line 23 to 29. An arrow points from this highlighted code to the browser window on the right.

The browser window displays the application's interface for editing assignments. The title bar shows the URL: `localhost:4200/assignment/1/edit?nom=TP1%20sur%20WebComponents,%20un%20lecteur%20audio%20amélioré#edition`. The main content area shows:

Application de gestion des devoirs à rendre (Assignment Manager)

Edition de l'assignment TP1 sur WebComponents, un lecteur audio amélioré

Edition du nom Edition de la date Sauver

Console tab is selected in the developer tools. The Query Params section shows:

```
Query Params :  
  {nom: "TP1 sur WebComponents, un lecteur audio amélioré"}
```

The Fragment section shows:

```
Fragment :  
  edition
```

Autre méthode avec des Observables (utile si les valeurs peuvent changer dynamiquement)

The screenshot shows a development setup with two main components:

- Code Editor:** On the left, the file `edit-assignment.component.ts` is open. It contains TypeScript code for an Angular component. A red box highlights the section where observables are used to handle query parameters and route fragments.
- Browser Preview:** On the right, a browser window displays the application's interface. The title bar shows the URL `localhost:4200/assignment/1/edit?nom=TP1%20sur%20WebComponents,%20un%20lecteur%20audio%20amélioré#edition`. The page content includes:
 - Section Headers:** "Application de gestion des devoirs à rendre (Assignment Management)" and "Edition de l'assignment TP1 sur WebComponents, un lecteur audio amélioré"
 - Form Fields:** "Edition du nom" and "Edition de la date". A "Sauver" (Save) button is highlighted in blue.
 - Console Tab:** The "Console" tab of the developer tools is selected. It shows the following log entries:
 - Query Params : `{nom: "TP1 sur WebComponents, un lecteur audio amélioré"}`
 - Fragment : `edition`

Gestion des accès / mode admin etc.

- Création d'un service de gestion des authentifications : 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 sauvégardé 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.service
```

The screenshot shows the Angular CLI workspace interface. On the left, the Explorer sidebar lists files under 'ASSIGNMENT-APP/shared': assignments.service.ts, auth.service.ts (selected), logging.service.ts, rendu.directive.ts, app.component.css, and app.component.html. The 'auth.service.ts' file is currently open in the main editor area. The code is as follows:

```
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
8   constructor() { }
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 and contains 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 "résolue", 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 }
```

Voir aussi [mon cours sur les Promesses](#) (pour utilisation de new Promise)

[Mon cours sur async/await](#)

[Cours MDN sur les promesses](#)

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)

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

```
import { CanActivateFn, Router } from '@angular/router';
import { AuthService } from './auth.service';
import { inject } from '@angular/core';

export const authGuard: CanActivateFn = (route, state) => {

  // injection par programme (au lieu de le faire dans
  // le constructeur d'un composant)
  let authService = inject(AuthService);
  let router = inject(Router);

  // si ça renvoie true, alors, on peut activer la route
  return authService.isAdmin()
    .then(authentifie => {
      if(authentifie) {
        console.log("Vous êtes admin, navigation autorisée !");
        return true;
      } else {
        console.log("Vous n'êtes pas admin ! Navigation refusée !");
        // et on retourne vers la page d'accueil
        router.navigate(['/home']);
        return false;
      }
    })
};
```

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

app.routes.ts X assignments.component.ts add-assignment.component.ts

```
assignment-app > src > app > app.routes.ts > ...
1 import { Routes } from '@angular/router';
2 import { AssignmentsComponent } from './assignments/assignments.component';
3 import { AddAssignmentComponent } from './assignments/add-assignment.component';
4 import { AssignmentDetailComponent } from './assignments/assignment-detail.component';
5 import { EditAssignmentComponent } from './assignments/edit-assignment.component';
6 import { authGuard } from './shared/auth.guard';
7
8
9
10 export const routes: Routes = [
11   {path: '', redirectTo: 'home', pathMatch: 'full'},
12   {path: 'home', component: AssignmentsComponent},
13   {path: 'add', component: AddAssignmentComponent},
14   {path: 'assignment/:id', component: AssignmentDetailComponent},
15   {path: 'assignment/:id/edit', component: EditAssignmentComponent}
16 ];
```

TS auth.guard.ts TS auth.service.ts X 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;
8
9   logIn() {
10     this.loggedIn = true;
11   }
12   logout() {
13     this.loggedIn = false;
14   }
15
16 // renvoie une promesse qui sera résolue lorsque l'utilisateur est
17 // administrateur ou pas. Pour le moment, elle renvoie
18 isAdmin() {
19   const isUserAdmin = new Promise(
20     (resolve, reject) => {
21       resolve(this.loggedIn)
22     }
23   );
24
25   return isUserAdmin;
26 }
27 }
```

Tester en mettant à true, là on doit pouvoir éditer tout le temps

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 says 'localhost:4200/home'. The page content is:

Application de gestion des devoirs à rendre (Assignments)

(The 'Login' button is highlighted with a red box.)

Et on l'ajoute dans le template de `app.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.....

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

```
assignment-detail.component.html
src > app > assignments > assignment-detail > assignment-detail.component.html
9
10      <button mat-stroked-button color="primary"
11          (click)="onClickEdit()"
12          [disabled]="!isAdmin()"
13      >
14          Edit
15      </button>
```

```
assignment-detail.component.ts
src > app > assignments > assignment-detail > assignment-detail.component.ts
53
54      isAdmin():boolean {
55          return this.authService.loggedIn;
56      }
57 }
```

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

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

- Refaire tout ce que je vous ai montré (reprendre le repo Git si besoin).
- 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
- Regarder comment insérer une Toolbar et une Sidebar dans la GUI de l'application. Voir doc en ligne d'Angular Material

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

Ici un cours plus complet dessus que je donnais...

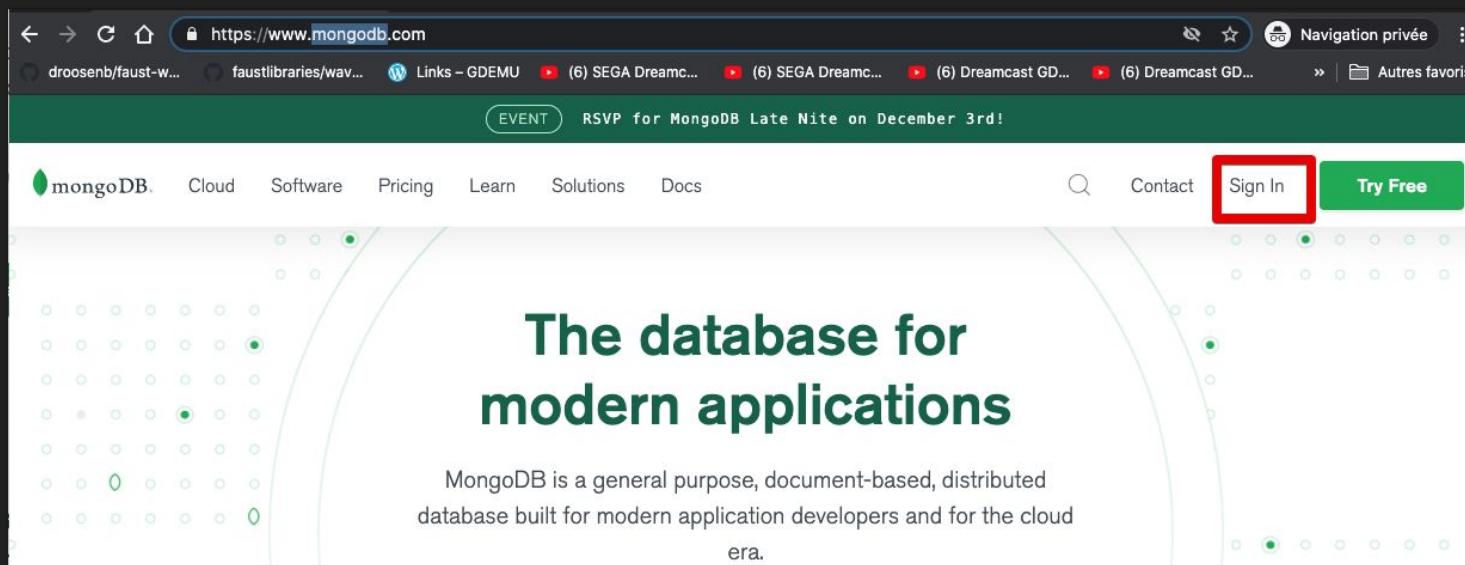


Quand on fait du MongoDB avec du Angular on dit que ça fait partie de la "**MEAN STACK**" (Mongo, Express, Angular, Node)

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/v2#/preferences/organizations), créez une nouvelle organisation et un nouveau projet (si vous n'en avez pas déjà)

The screenshot shows a web browser window with two tabs open, both displaying the MongoDB Cloud interface.

The left tab shows the "All Organizations" page with a sidebar containing "PREFERENCES" (Legacy 2FA, Personalization, Invitations, Organizations, Public API Access) and "ORGANIZATIONS" (Master 2 MIAGE (IA2, MBDS, INTENSE, ESTIA ETC.), Access Manager, Settings, Billing, Support).

The right tab shows the "Create a Project" page for the "Master 2 MIAGE (IA2, MBDS, INTENSE, ESTIA ETC.)" organization. The page has a header with "Access Manager", "Support", and "Billing". It displays a "Create a Project" form with fields for "Name Your Project" (containing "Tuto Angular Mongo Cloud") and "Add Members". A "Next" button is visible at the bottom right of the form.

Création d'une base Mongo dans le cloud

Etape 2 : créer un cluster

The screenshot shows a web browser window for the MongoDB Atlas interface at cloud.mongodb.com/v2/5fc8a45153448104c1f7d686#clusters. The navigation bar includes links for Applications, droosenb/faust-w..., faustlibraries/wav..., Links - GDEMU, (6) SEGA Dreamcast GD..., (6) SEGA Dreamcast GD..., and (6) Dreamcast GD... The main menu has sections for DATA STORAGE (Clusters, Triggers, Data Lake), SECURITY (Database Access, Network Access, Advanced), and Atlas (Access Manager, Support, Billing). The 'Clusters' section is selected. A sub-menu for 'Tutos Angular React etc.' is open. The central area displays the 'Clusters' page with a search bar 'Find a cluster...' and a large green button labeled 'Build a Cluster'. Above the button is the text 'Create a cluster' and 'Choose your cloud provider, region, and specs.' Below the button, a note says 'Once your cluster is up and running, live migrate an existing MongoDB database into Atlas with our [Live Migration Service](#)'. A red box highlights the 'Build a Cluster' button.

Création d'une base Mongo dans le cloud

Etape 2 : créer un cluster (suite)

The screenshot shows the MongoDB Atlas Clusters interface. On the left, a sidebar provides navigation and information about Dedicated MongoDB Clusters, including a summary of included features like Shared and Private Clusters, Replication across multiple regions, Globally distributed and write operations, and Control over data access. A green button at the bottom of the sidebar says "Create a Cluster". The main area displays a list of clusters under the heading "Clusters". One cluster, "Cluster0" (Version 4.2.10), is highlighted with a red box around the "CONNECT" button. Other tabs for "METRICS" and "COLLECTIONS" are visible. Below the cluster details, sections show "CLUSTER TIER" (M0 Sandbox (General)), "REGION" (AWS / Frankfurt (eu-central-1)), "TYPE" (Replica Set - 3 nodes), and "LINKED REALM APP" (None Linked). To the right, three cards provide real-time metrics: "Operations R: 0 W: 0" (100.0/s), "Logical Size 0.0 B" (512.0 MB max), and "Connections 0" (500 max). A call-to-action "Enhance Your Experience" encourages upgrading the cluster, with a green "Upgrade" button.

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

Clusters

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

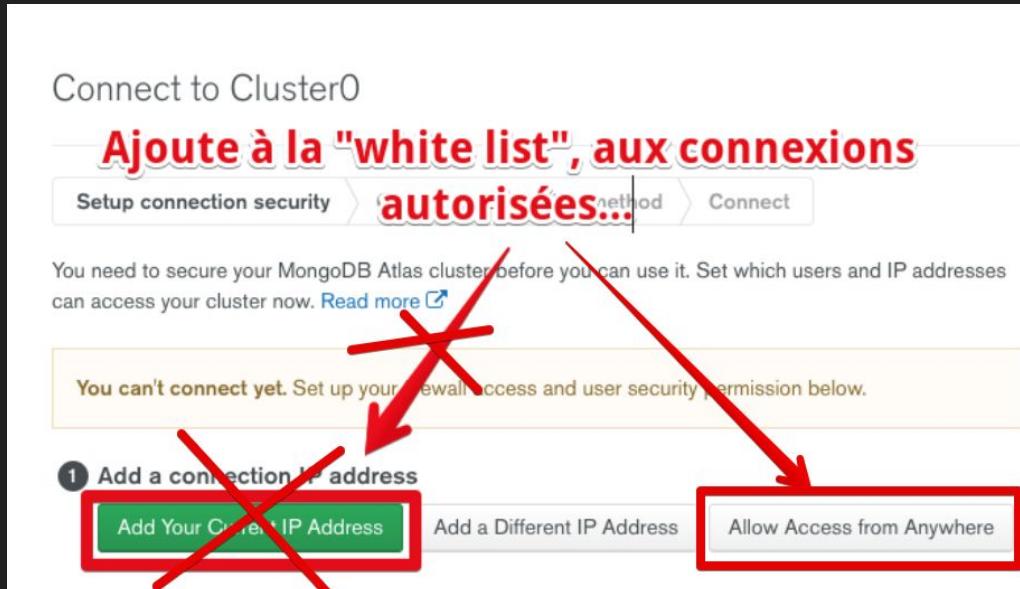
Connections 0 500 max

Enhance Your Experience

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

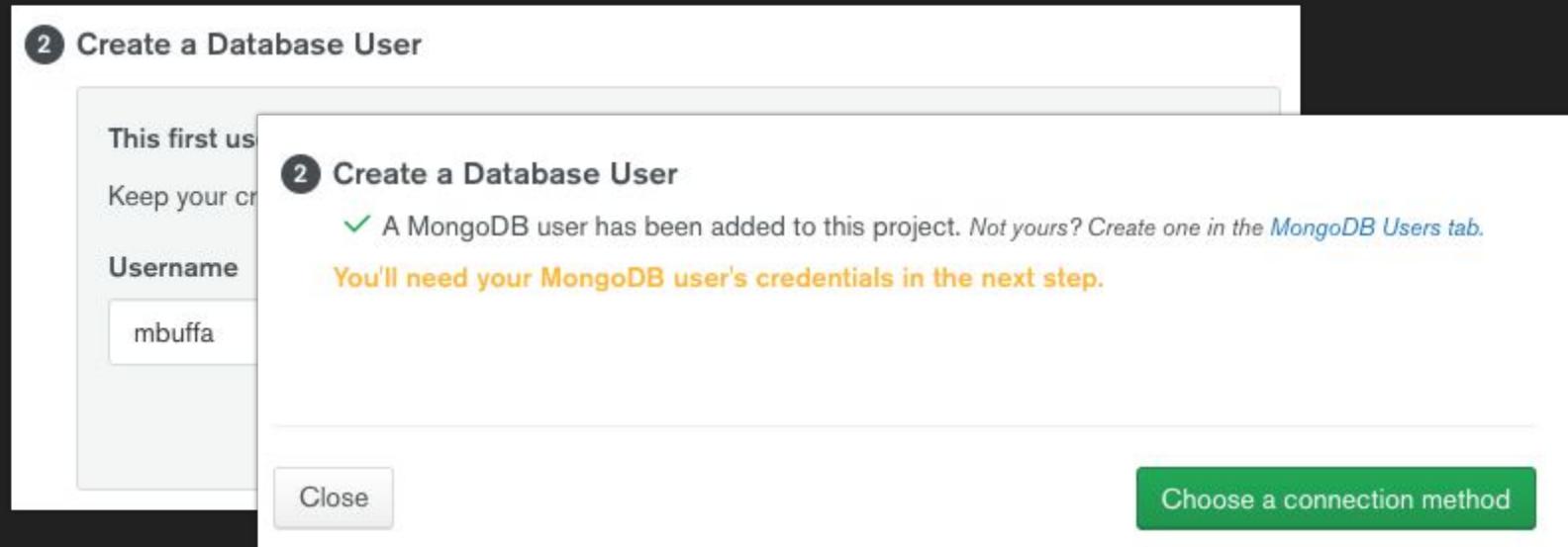
Upgrade

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



Connect to Cluster0



Connect to your application

Drivers
Access your Atlas data using MongoDB's native drivers (e.g. Node.js, Go, etc.)

A card with a green icon of a database containing binary code. A red arrow points from this section to the 'Drivers' section in the right panel.

Access your data through tools

Compass
Explore, modify, and visualize your data with MongoDB's GUI

A card with a green icon of a compass. A red arrow points from this section to the 'Compass' section in the right panel.

Shell
Quickly add & update data using MongoDB's Javascript command-line interface

A card with a green icon of a terminal window showing a '>' symbol. A red arrow points from this section to the 'Shell' section in the right panel.

MongoDB for VS Code
Work with your data in MongoDB directly from your VS Code environment

A card with a green icon of a puzzle piece. A red arrow points from this section to the 'MongoDB for VS Code' section in the right panel.

Atlas SQL
Easily connect SQL tools to Atlas for data analysis and visualization

A card with a green icon of a database with a SQL logo. A red arrow points from this section to the 'Atlas SQL' section in the right panel.

Go Back

Close

Connect to Cluster0



Connecting with MongoDB Driver

1. Select your driver and version

We recommend installing and using the latest driver version.

Driver Version

Node.js 5.5 or later

2. Install your driver

Run the following on the command line

npm install mongodb

[View MongoDB Node.js Driver installation instructions.](#)

3. Add your connection string into your application code

View full code sample

```
mongodb+srv://leodonati:<password>@cluster0.hmqibqn.mongodb.net/?retryWrites=true&w=majority
```

Replace <password> with the password for the leodonati user. Ensure any option params are [URL encoded](#).

RESOURCES

[Get started with the Node.js Driver](#)
[Access your Database Users](#)

[Node.js Starter Sample App](#)
[Troubleshoot Connections](#)



Close

A.

The screenshot shows the MongoDB Atlas Cluster Overview page. The top navigation bar includes 'M2 Miage (INTENSE...)', 'Access Manager', 'Support', and 'Billing'. Below the navigation is the breadcrumb 'ESTIA-MBDS-LD > TUTO ANGULAR > DATABASES'. The main header 'ClusterO' is followed by tabs: 'Overview', 'Real Time', 'Metrics', 'Collections' (which is selected), 'Atlas Search', 'Profiler', 'Performance Advisor', 'Online Archive', and 'Cmd Line Tools'. On the left, a sidebar under 'DATA STORAGE' has 'Clusters' selected, along with 'Triggers', 'Data Lake', 'SECURITY', 'Database Access', and 'Network Access'. The central area displays 'Databases: 0' and 'Collections: 0'. A large 'Explore Your Data' section features a search icon and a brief description of available operations: 'Find', 'Indexes', 'Aggregation', and 'Search'. Buttons for 'Load a Sample Dataset' and 'Add My Own Data' are present, along with a link to 'Learn more in Docs and Tutorials'. A red arrow points from the bottom right towards the 'Add My Own Data' button.

A.

The screenshot shows the MongoDB Atlas Collection Overview page for the 'assignmentsDB.assignments' collection. The top navigation bar and tabs are identical to the previous screen. The central area shows 'DATABASES: 1' and 'COLLECTIONS: 1'. A 'Create Database' button is available. A search bar for 'Search Namespaces' is present. The left sidebar shows the database 'assignmentsDB' and the collection 'assignments'. The main content area for 'assignmentsDB.assignments' displays storage details: 'STORAGE SIZE: 4KB', 'LOGICAL DATA SIZE: 0B', 'TOTAL DOCUMENTS: 0', and 'INDEXES TOTAL SIZE: 4KB'. It includes tabs for 'Find', 'Indexes', 'Schema Anti-Patterns', 'Aggregation', and 'Search Indexes'. A red arrow points from the bottom right towards the 'INSERT DOCUMENT' button. The bottom section contains a 'Filter' input field with the placeholder 'Type a query: { field: 'value' }', and buttons for 'Reset', 'Apply', and 'Options'.

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.
 - Ca doit ressembler à ceci :

```
// remplacer toute cette chaîne par l'URI de connexion à votre propre base dans le cloud s
const uri = 'mongodb+srv://<utilisateur>:<password>@cluster0.euczw.mongodb.net/assignmentsDB?retryWrites=true&w=majority';
```

← → C ⌂ ① localhost:8010/api/assignments

Applications droosenb/faust-w... faustlibraries/wav... Links – GDEMU (6) SEGA Dreamc... (6) SEGA Dreamc... (6) Dreamcast GD... (6) Dreamcast GD... Repl.it - Entirely

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

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...

```
assignment.component.ts settings.json assignment-detail.component.ts assignments.service.ts M X
assignment-app > src > app > shared > assignments.service.ts > AssignmentsService
1 import { Injectable } from '@angular/core';
2 import { Assignment } from '../assignments/assignment.model';
3 import { Observable, of } from 'rxjs';
4 import { LoggingService } from './logging.service';
5 import { HttpClient } from '@angular/common/http';
6
7 @Injectable({
8   providedIn: 'root'
9 })
10 export class AssignmentsService {
11   backendURL = 'http://localhost:8010/api/assignments';
12
13   constructor(private loggingService: LoggingService, private http: HttpClient) {}
14
15   getAssignments(): Observable<any> {
16     return this.http.get<any>(this.backendURL);
17   }
18
19   getAssignment(id: number): Observable<any> {
20     return this.http.get<any>(this.backendURL+ id);
21   }

```

On l'importe dans le service **assignment.service.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(id) : Observable<Assignment> {
  this.http.
    return of( ⚡ * get      (method) HttpClient.get(url: string, ...
  }
  ⚡ * post
  ⚡ * request
addAssignment(assignment: Assignment): Observable<string> {
```

Angular 18 et supérieur: Ajouter ceci pour rendre HttpClient utilisable, dans app.config.ts



The screenshot shows a code editor with multiple tabs at the top: assignments.service.ts, .gitignore, assignments.component.ts, and app.config.ts. The app.config.ts tab is active, showing the following code:

```
assignment-app > src > app > TS app.config.ts > ...
You, il y a 1 seconde | 2 authors (Michel Buffa and one other)
1 import { ApplicationConfig, provideZoneChangeDetection } from '@angular/core';
2 import { provideRouter } from '@angular/router';
3
4 import { routes } from './app.routes';
5 import { provideAnimationsAsync } from '@angular/platform-browser/animations/async';
6
7 import { provideHttpClient, withInterceptorsFromDi } from '@angular/common/http';
8
9 export const appConfig: ApplicationConfig = {
10   providers: [provideZoneChangeDetection({ eventCoalescing: true }),
11     provideRouter(routes),
12     provideAnimationsAsync(),
13     provideHttpClient(withInterceptorsFromDi())],
14 };
15
```

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';
28
29 > const routes : Routes = [ ...
40 ];
41
42 @NgModule({
43   declarations: [...],
44   imports: [
45     BrowserModule,
46     BrowserAnimationsModule,
47     MatButtonModule, MatIconModule, MatDividerModule,
48     MatInputModule, MatFormFieldModule,
49     MatDatepickerModule, MatNativeDateModule,
50     MatListModule, MatCardModule, MatCheckboxModule,
51     MatSlideToggleModule,
52     FormsModule, HttpClientModule,
53     RouterModule.forRoot(routes)
54   ],
55 }
```

On l'importe dans le **service assignment.service.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...

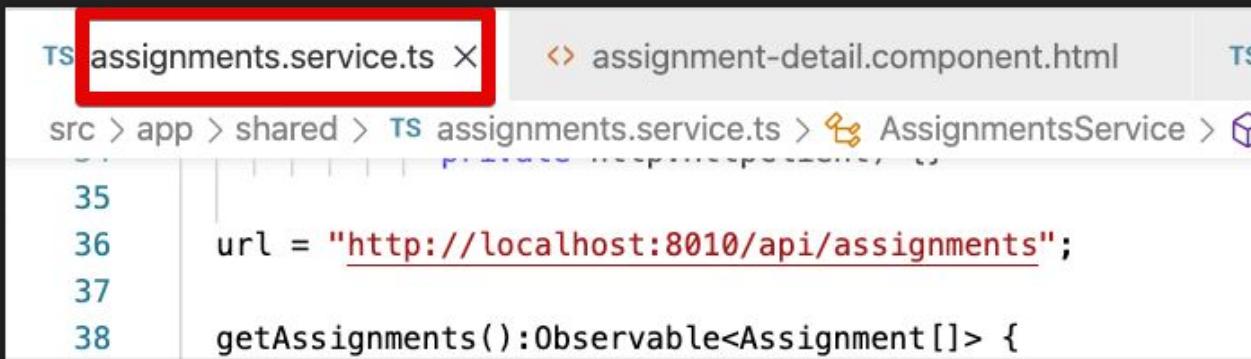
Très utilisé quand on programme en Angular,

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

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

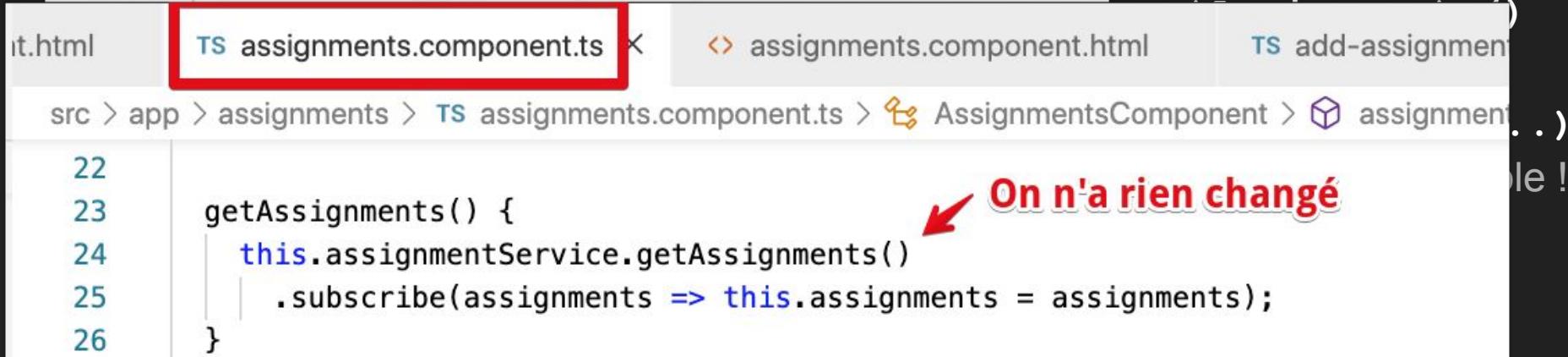


Ajout de HTTP GET dans le service assignments



```
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 > ...
22
23     getAssignments() {
24         this.assignmentService.getAssignments()
25             .subscribe(assignments => this.assignments = assignments);
26     }
```

On n'a rien changé

Tester l'application

The screenshot shows a web browser window with the URL `localhost:4200/home`. The page title is Application de gestion des devoirs à rendre (Assignments). A message at the top states: Le devoir TP Angular avec Mongo Cloud n'a pas été rendu. On the right, there is a Login button and a pink Ajouter Assignment button. The browser's developer tools are open, specifically the Network tab, which is highlighted with a red box. The Network tab shows an XHR request for the URL `/assignments`. The response pane displays the JSON data for the assignment:

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

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

localhost:4200/assignment/1

Applications droosenb/faust-w... faustlibraries/wav... Links – GDEMU (6) SEGA Dreamc... (6) SEGA Dreamc... (6) Dreamc...

Application de gestion des devoirs à rendre (Assignments)

Pas le bon titre....

TP1 sur WebComponents, un lecteur audio amélioré

Jan 17, 2020

Edit DELETE

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 faire une
HTTP POST

Et comment faire

requête

```
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 |
```

Ajout d'un assignment (POST)

The screenshot shows a browser window with a list of assignments on the left and a developer tools console on the right.

Code in assignments.service.ts:

```
src > app > shared > TS assignments.service.ts X

49
50     addAssignment(a
51         // this.assignm
52         // this.loggi
53
54         //return of('
55             return this.h
56 }
57
```

Assignment List:

- 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.

Developer Tools Console:

```
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 > TS assignments.service.ts < ◊ assignment-detail.component.html < TS assignn  
src > TS assignment-detail.c < TS edit-assignment.component.ts < # assignment-detail.component.css < add-assignment.component.html  
59     src > app > assignments > edit-assignment > TS edit-assignment.component.ts > ...  
60     30 }  
61     31     event.preventDefault();  
62     32     if(this.nomAssignment) {  
63     33         this.assignment.nom = this.nomAssignment;  
64     34     };  
65     35     if(this.dateDeRendu) {  
66     36         this.assignment.dateDeRendu = this.dateDeRendu;  
67     37     };  
68     38     this.assignmentsService.updateAssignment(this.assignment)  
69     39         .subscribe(message => {  
70     40             console.log(message);  
71     41             // navigation vers la home page uniquement après que la modification ait été effectuée  
72     42             this.router.navigate(["/home"]);  
73     43             // ou pourquoi pas ["//home" + this.assignment.id] ?  
74     44         });  
75     45         //this.router.navigate(["/home"]);  
76     46     }  
77     47 }
```

A red arrow points from the text "Pourquoi on a changé cela ?" to the line `this.router.navigate(["/home"]);`. A red box highlights the line `this.router.navigate(["/home"]);` and the line below it, `// ou pourquoi pas ["//home" + this.assignment.id] ?`.

Suppression d'un assianment (DELETE)

The screenshot shows a browser window with the URL `localhost:4200/home`. The main content area displays a heading "Application de gestion des devoirs à rendre (Assignments)" and two messages: "Devoir intitulé TP Angular avec Mongo Cloud a été rendu." and "Devoir intitulé Nouveau TP à rendre sur HttpClient a été rendu.". Below the messages, the developer tools' "Console" tab is selected, showing the following log output:

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

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 page title is Application de gestion des devoirs à rendre (Assignments). Below the title, there is a list item with the text "TP Angular avec Mongo Cloud transformé avec un pipe....". To the left of this text is a "Edit" button, and to the right is a red "DELETE" button. At the bottom of the page, there is a navigation bar with a "DÉCOUVRIR" button and a "DÉMARREZ LE PREMIER TP" button. The number "7" is visible in the bottom-left corner.

localhost:4200/assignment/1

Applications Plan du Cours | O... droosenb/faust-w... faustlibraries/wav... Links – GDEMU (6) SEGA Dream... (6) SEGA Dream...

Application de gestion des devoirs à rendre (Assignments)

TP Angular avec Mongo Cloud transformé avec un pipe....

Edit

DELETE

DÉCOUVRIR

DÉMARREZ LE PREMIER TP

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`. Below the tabs, the address bar shows `localhost:4200/assignment/1`. The main content area displays a heading Application de gestion des devoirs à rendre (Assignments). Below it, there is a card with the text "TP Angular avec Mongo Cloud reçu et transformé avec un pipe...." and two buttons: "Edit" and "DELETE". At the bottom of the screen, 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 last three lines of the console output are highlighted with a red box.

Gestion des erreurs: utiliser catchError dans le service

TS assignments.service.ts TS assignment-detail.component.ts

```

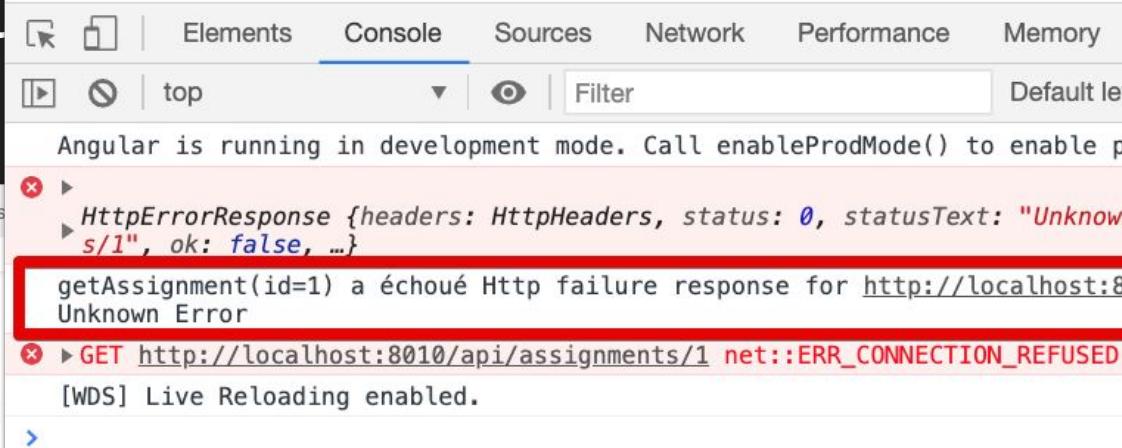
src > app > shared > TS assignments.service.ts > AssignmentsService
44 // the id passed as parameter
45 getAssignment(id) : Observable<Assignment> {
46   return this.http.get<Assignment>(this.url + "/" + id)
47     .pipe(
48       map(a => {
49         a.nom += " reçu et transformé avec un pipe....";
50         return a;
51       }),
52       tap(_ => {
53         console.log("tap: assignment avec id = " + id + " requête GET envoyée sur MongoDB cloud");
54       })
55     catchError(this.handleError<Assignment>(`getAssignment(id=${id})`))
56   );
57 }

59 private handleError<T>(operation, result?: T) {
60   return (error: any) : Observable<T> => {
61     console.error(error); // pour afficher l'erreur dans la console
62     console.log(operation + ' a échoué ' + error.message);

64     return of(result as T);
65   };
66 }

```

Méthode générique bien pratique...

sur le port 8010)

- On demande à voir le détail d'un assignment
- On n'oublie pas de regarder la console de debug

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';
```

les GET, PUT, POST,

comme application/json,

TS assignments.service.ts X

TS assignment-detail.component.ts

↳ assignment-detail.component

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';
```

Astuce : on peut importer du JSON directement dans un fichier typescript (ne pas faire dans TP)

1 - Dans `tsconfig.app.json` à la racine du projet, on va modifier les options du compilateur typescript pour qu'il accepte les imports de fichiers

```
{
  "extends": "./tsconfig.json",
  "compilerOptions": {
    "resolveJsonModule": true,
    "outDir": "./out-tsc/app",
    "types": []
  },
}
```

2 - On suppose qu'on a du JSON dans un fichier `assignments.json`, on peut l'importer comme cela :

```
import * as data from "assignments.json"
```

3 - Exemple de contenu :

```
[{"id":1,"nom":"Devoir 1","dateDeRendu":"2020-09-08 05:06:09","rendu":true}, {"id":2,"nom":"Devoir 2","dateDeRendu":"2020-04-28 18:34:08","rendu":true}]
```

4 - Exemple d'utilisation après l'import

```
assignmentsJson: any = (data as any).default;

peuplerBD() {
  for (let i = 0; i < this.assignmentsJson.length; i++)
    {...}
}
```

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) :

```
peuplerBDavecForkJoin():Observable<any> {
  let appelsVersAddAssignment:Observable<any>[] = [];

  bdInitialAssignments.forEach(a => {
    const nouvelAssignment = new Assignment();
    nouvelAssignment.nom = a.nom;
    nouvelAssignment.dateDeRendu = new Date(a.dateDeRendu);
    nouvelAssignment.rendu = a.rendu;

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

  return forkJoin(appelsVersAddAssignment);
}
```

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

  // meilleure version :
  this.assignmentsService.peuplerBDavecForkJoin()
    .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});
      // ceci marche...
      window.location.reload();
    })
}
```

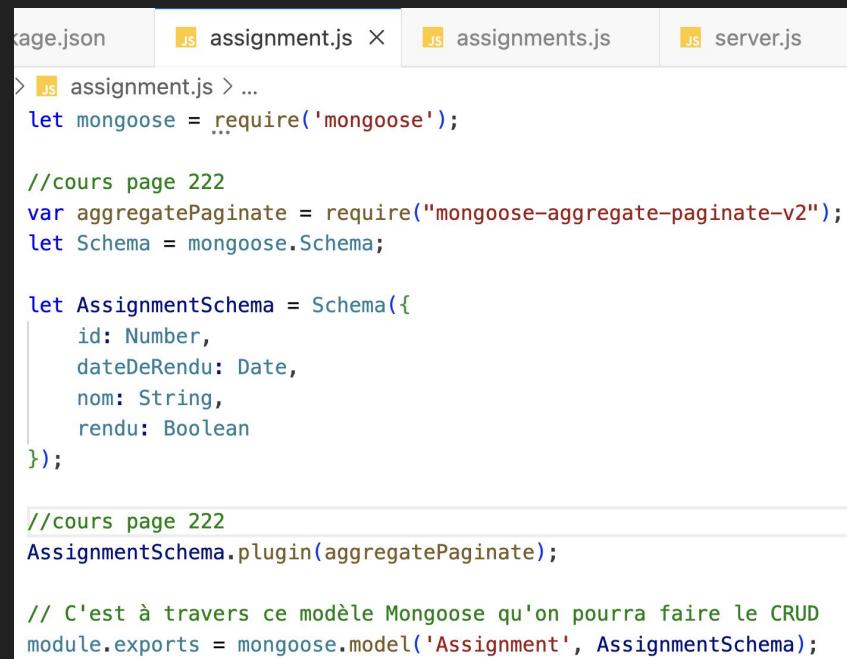
Deuxième étape : modification du
back-end pour gérer la pagination

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 !



```
package.json      js assignment.js ×      js assignments.js      js server.js
> js assignment.js > ...
let mongoose = ...require('mongoose');

// cours page 222
var aggregatePaginate = require("mongoose-aggregate-paginate-v2");
let Schema = mongoose.Schema;

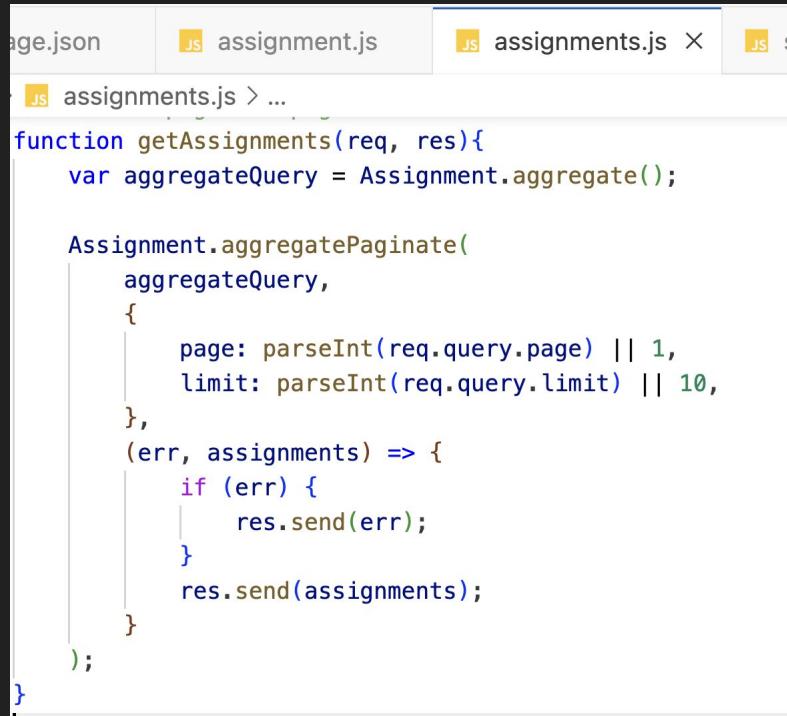
let AssignmentSchema = Schema({
  id: Number,
  dateDeRendu: Date,
  nom: String,
  rendu: Boolean
});

// cours page 222
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 :



```
page.json    JS assignment.js    JS assignments.js ×    JS s
· JS assignments.js > ...
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);
    }
);
```

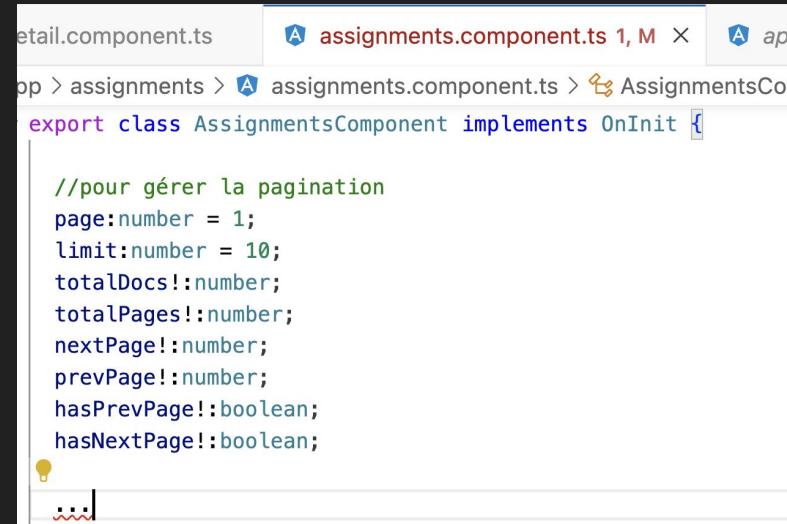
On va devoir passer en paramètres dans l'URL, sous forme de “queries”, la **page** demandée et le nombre d'assignments demandés (**limit**)

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

```
← → C ⌂ ① localhost:8010/api/assignments
Applications BSR Tuning GLK2... TUTO Remplacem...
{
  + docs: [...],
  totalDocs: 500,
  limit: 10,
  page: 1,
  totalPages: 50,
  pagingCounter: 1,
  hasPrevPage: false,
  hasNextPage: true,
  prevPage: null,
  nextPage: 2
}
```

```
← → C ⌂ ① localhost:8010/api/assignments?page=10&limit=3
Applications BSR Tuning GLK2... TUTO Remplacem... 12.5" IPS Screen... Archived-SANSA... LODE - Live OWL...
{
  - 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
}
```

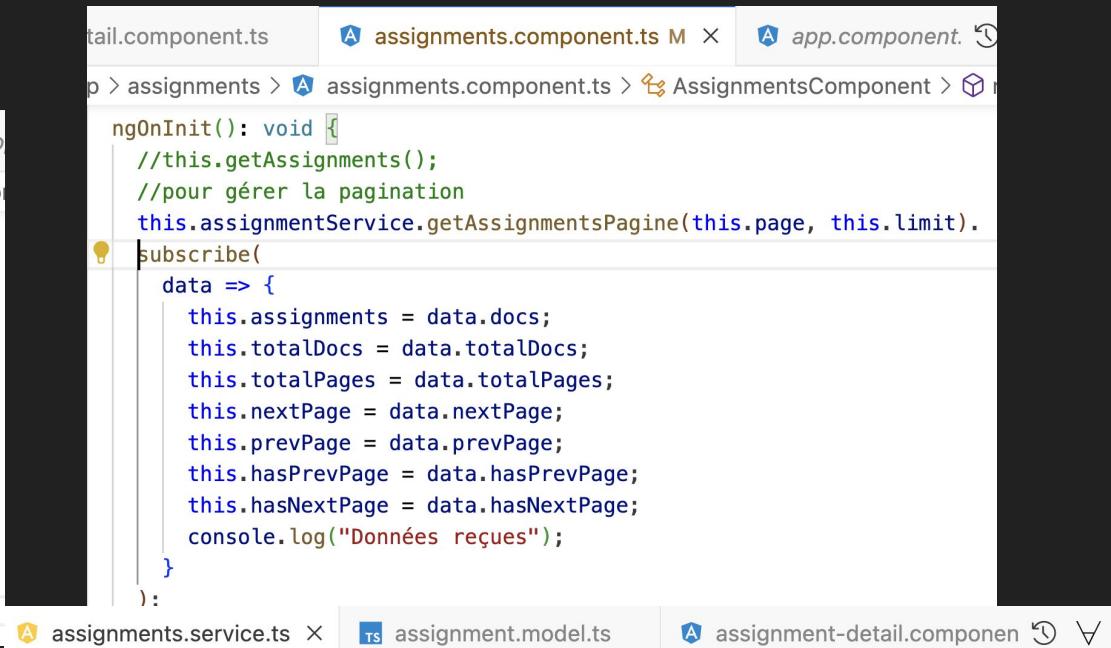
Modification du composant angular qui affiche la liste des assignments



```
detail.component.ts          A assignments.component.ts 1, M X      A app.
op > assignments > A assignments.component.ts > ⚭ AssignmentsComponent > ⓘ
export class AssignmentsComponent implements OnInit {

    //pour gérer la pagination
    page:number = 1;
    limit:number = 10;
    totalDocs!:number;
    totalPages!:number;
    nextPage!:number;
    prevPage!:number;
    hasPrevPage!:boolean;
    hasNextPage!:boolean;

    ...
}
```



```
tail.component.ts           A assignments.component.ts M X      A app.component. ⓘ
o > assignments > A assignments.component.ts > ⚭ AssignmentsComponent > ⓘ
ngOnInit(): void {
    //this.getAssignments();
    //pour gérer la pagination
    this.assignmentService.getAssignmentsPagine(this.page, this.limit).
        subscribe(
            data => {
                this.assignments = data.docs;
                this.totalDocs = data.totalDocs;
                this.totalPages = data.totalPages;
                this.nextPage = data.nextPage;
                this.prevPage = data.prevPage;
                this.hasPrevPage = data.hasPrevPage;
                this.hasNextPage = data.hasNextPage;
                console.log("Données reçues");
            }
        );
}

A assignments.service.ts X      ts assignment.model.ts      A assignment-detail.componen ⓘ
src > app > shared > A assignments.service.ts > ⚭ AssignmentsService
23
24
25
26
27
28
getAssignmentsPagine(page:number, limit:number) : Observable<any> {
    return this.http.get<any>(this.url + '?page=' + page + '&limit=' + limit);
}
```

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 :

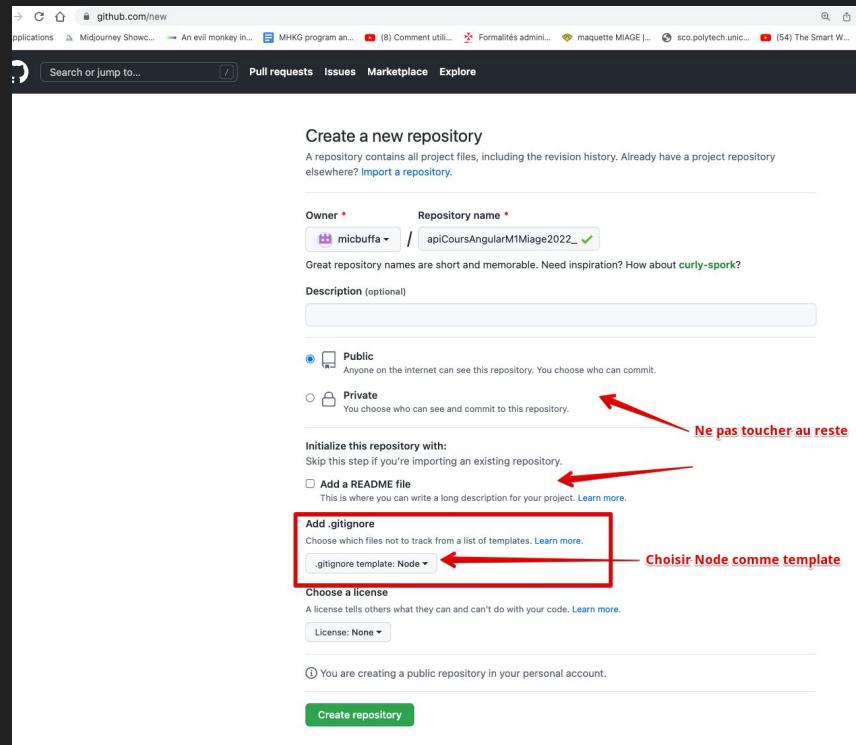
- Mettre en grisé (disabled) les boutons si on est sur la première ou dernière page
- Regarder dans Angular Material les widgets de pagination et essayer de les utiliser

Hébergement sur Render.com

Première étape : hébergement du back end

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

1. Pour un déploiement le projet doit être sur GitHub (public)
2. Il doit être un projet “node” correct (avec un bon .gitignore)
3. Il doit avoir un package.json avec des scripts pour builder (si besoin)



Première étape : hébergement du back end

Vérifiez ensuite que votre projet est bien sur le repository github, et qu'il est correct !

The screenshot shows a GitHub repository page for 'micbuffa / M1Miage2022_2023_api'. The 'Code' tab is selected. The commit history is displayed, showing the following entries:

Commit	Type	Time
micbuffa Update README.md	backend	77695fa 2 minutes ago
model	backend	13 minutes ago
routes	backend	13 minutes ago
.gitignore	backend	13 minutes ago
README.md	Update README.md	2 minutes ago
package.json	backend	13 minutes ago
server.js	backend	13 minutes ago

A red box with the text 'Vérifiez bien qu'il n'y ait pas de dossier node_modules !' is overlaid on the commit list, pointing to the 'node_modules' entry.

At the bottom of the page, the file 'README.md' is shown with the text 'apiCoursAngularM1Miage2022_2023'.

Hébergement du back-end sur render.com (suite)

Il faut savoir que quand on déploie une application sur Render.com, ce dernier demandera les commandes à exécuter pour builder le projet (ex: npm install) et pour l'exécuter (ex: node server.js ou 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",
    "build": "npm install",
    "start": "node server.js"
  },
```

Hébergement du back-end sur Render.com (suite)

Pour être bien sûr :

1. Effacer le dossier node_modules
2. Faire
 - a. `npm install`
 - b. `npm run build`
 - c. `npm run start`
3. L'application node doit s'exécuter correctement !
4. Une fois cela fait on peut créer une application sur Render.com (en tant que Web Service) et la lier au repo github

Ajout d'une application (1)

dashboard.render.com

Cart | Displate MyVoice| TTS | Te... How to train Chat... Significant-Gravit... (291) ChatGPT Co... Disque Portable 1... Nouvelle Machine... Crée un Bot #Ch... Marble and Ball -... (147) Perplexus S...

render Dashboard Blueprints Env Groups Docs Community Help New + micbuffa@gmail.com

Overview

Search services

NAME	STATUS
MBDS Madagascar 2022-2023 front-end	Deploy suc...
MBDS Madagascar 2022-2023 back-end	Deploy suc...

Web Service

Web services are kept up and running at all times, with native SSL and HTTP/2 support. Add a persistent disk or custom domain. Scale up and down with ease.

Learn more.

- Static Site
- Web Service**
- Private Service
- Background Worker
- Cron Job
- PostgreSQL
- Redis
- Blueprint

ON LAST DEPLOYED kfurt an hour ago kfurt 2 hours ago

Ajout d'une application (2)

Votre repository est public, donc entrer son URL dans la zone correspondante...

dashboard.render.com/select-repo?type=web

Cart | Displate MyVoice| TTS | Te... How to train Chat... Significant-Gravit... (291) ChatGPT Co... Disque Portable 1... Nouvelle Mac

render Dashboard Blueprints Env Groups Docs Community Help

Create a new Web Service

Connect your Git repository or use an existing public repository URL.

Connect a repository

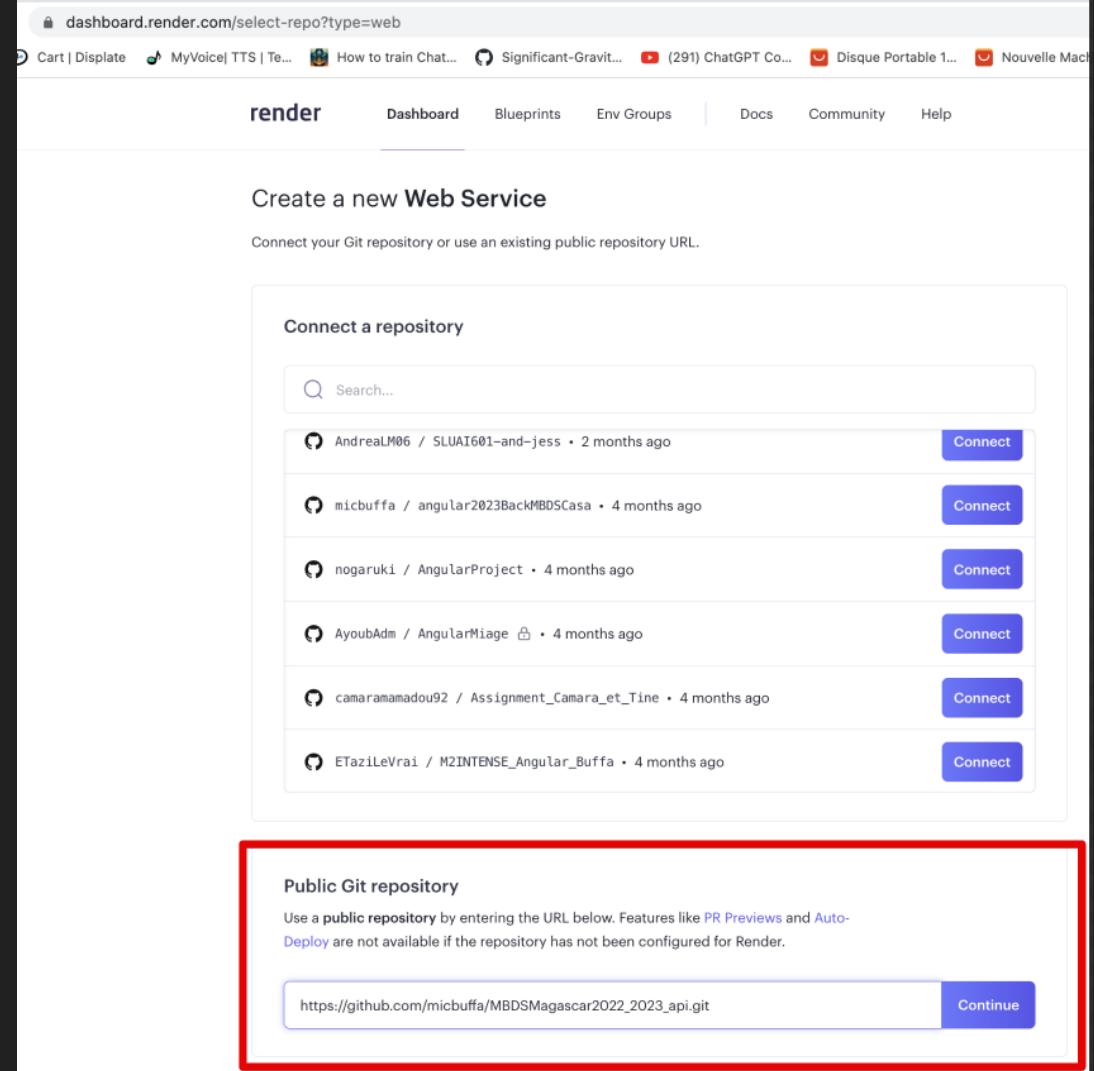
Search...

AndreaLM06 / SLUAI601-and-jess • 2 months ago	Connect
micbuffa / angular2023BackMBDSCas... • 4 months ago	Connect
nogaruki / AngularProject • 4 months ago	Connect
AyoubAdm / AngularMiage • 4 months ago	Connect
camaramamadou92 / Assignment_Camara_et_Tine • 4 months ago	Connect
ETaziLeVrai / M2INTENSE_Angular_Buffa • 4 months ago	Connect

Public Git repository

Use a **public repository** by entering the URL below. Features like PR Previews and Auto-Deploy are not available if the repository has not been configured for Render.

Continue



Ajout d'une application (3)

Un peu plus bas dans la page, choisissez l'option gratuite et faites “create Web Service”

You are deploying a web service for [micbuffa/MBDSMagascar2022_2023_api](#).

You seem to be using Node, so we've autofilled some fields accordingly. Make sure the values look right to you!

Entrez un nom ! 

Name
A unique name for your web service.

Region
The [region](#) where your web service runs. Services must be in the same region to communicate privately and you currently have services running in [Frankfurt](#).

Branch
The repository branch used for your web service.

Root Directory Optional
Defaults to repository root. When you specify a [root directory](#) that is different from your repository root, Render runs all your commands in the [specified directory](#) and ignores changes outside the directory.

Runtime
The runtime for your web service.

Build Command
This command runs in the root directory of your repository when a new version of your code is pushed, or when you deploy manually. It is typically a script that installs libraries, runs migrations, or compiles resources needed by your app.

Start Command
This command runs in the root directory of your app and is responsible for starting it up once it's built in the build step.

Ajout d'une application (4)

Ca va ensuite exécuter les commandes de build et d'exécution que vous avez indiquées...

Les logs s'affichent...

Le lien vers l'appli déployée est en haut à gauche. Une fois déployé sans erreur, testez !

The screenshot shows the Render dashboard for a service named 'test'. The service is running on a Node.js instance and is currently free. It is associated with the repository 'micbuffa/MBDSMagagascar2022_2023_api' and branch 'main'. A link to the deployed application at 'https://test-h35p.onrender.com' is provided. On the left, there are navigation links for Events, Logs, Disks, Environment, Shell, PRs, Jobs, Metrics, Scaling, and Settings. The main area displays a log of the deployment process:

```
May 17 01:01:07 PM > npm install
May 17 01:01:07 PM
May 17 01:01:10 PM added 100 packages from 135 contributors and audited 100 packages in 2.419s
May 17 01:01:10 PM
May 17 01:01:10 PM 2 packages are looking for funding
May 17 01:01:10 PM run `npm fund` for details
May 17 01:01:10 PM
May 17 01:01:10 PM found 7 vulnerabilities (2 moderate, 4 high, 1 critical)
May 17 01:01:10 PM run `npm audit fix` to fix them, or `npm audit` for details
May 17 01:01:11 PM =>> Uploading build...
May 17 01:01:18 PM =>> Build uploaded in 7s
May 17 01:01:18 PM =>> Build successful
May 17 01:01:18 PM =>> Deploying...
```

Hébergement du front-end sur Render.com (suite)

Rappel : lors du déploiement, Render.com exécute la commande de build et la commande d'exécution que vous aurez indiquées :

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....

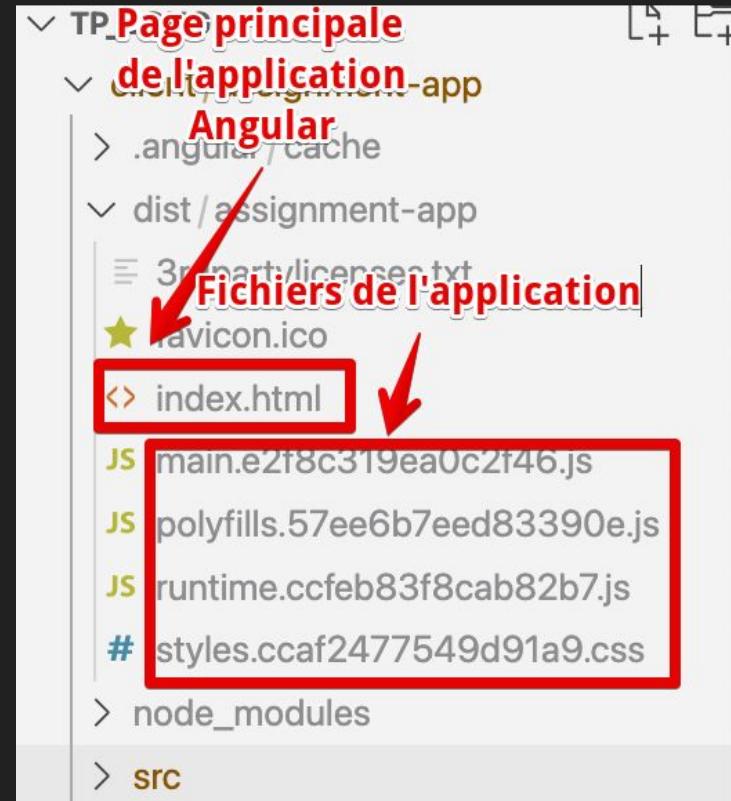
Hébergement du front-end sur Render.com (suite)

Rappel : lors du déploiement, Render exécute :

1. `npm run build`
2. `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 ou Render.com 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/browser"));

app.get("/", function (req, res) {
  res.sendFile(path.join( dirname + "/dist/assignment-app/browser/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 Render

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

ATTENTION!

Avant de déployer sur render.com, il faut ajouter un fichier **.node-version** à la racine du projet

Ce fichier contiendra la version que vous avez testée localement. Par exemple,
une seule ligne avec 16.14.0 dedans!

La commande ng build appelée par npm run build a sorti une erreur quand j'ai déployé : nodeJS trop ancien !!!! Du coup, si le fichier .node-version est présent, render.com installe la version requise.

Bonus : DataTable, Infinite Scrolling

Using an Angular Material Table along with Pagination widget

Check the [Angular Material Table documentation](#), and check the very first example source code. “datasource” will be your assignment array. Adapt the code.

[Video from 2023](#) showing how to use Angular data Tables along with the Paginator widget.

Live coding!

Infinite Scrolling

Infinite scrolling: you scroll the page and when you reach the end, it automatically sends a new request to the server, which retrieves new data, and you can continue scrolling.

See the [Angular scrolling doc here](#), and also [a tutorial](#) showing how you can further improve things by adding new data to the list of assignments, which you fetch from the database as you scroll... LIVE CODING VERSION : check [this video I made in 2023](#)

APPARENTLY, THERE IS A MORE MODERN WAY TO DO THAT (CHECK ANGULAR DOC LINK ABOVE, AND SCROLL DOWN TO "WITH DATASOURCE"...