Cours 4 : Frontend  – partie 1

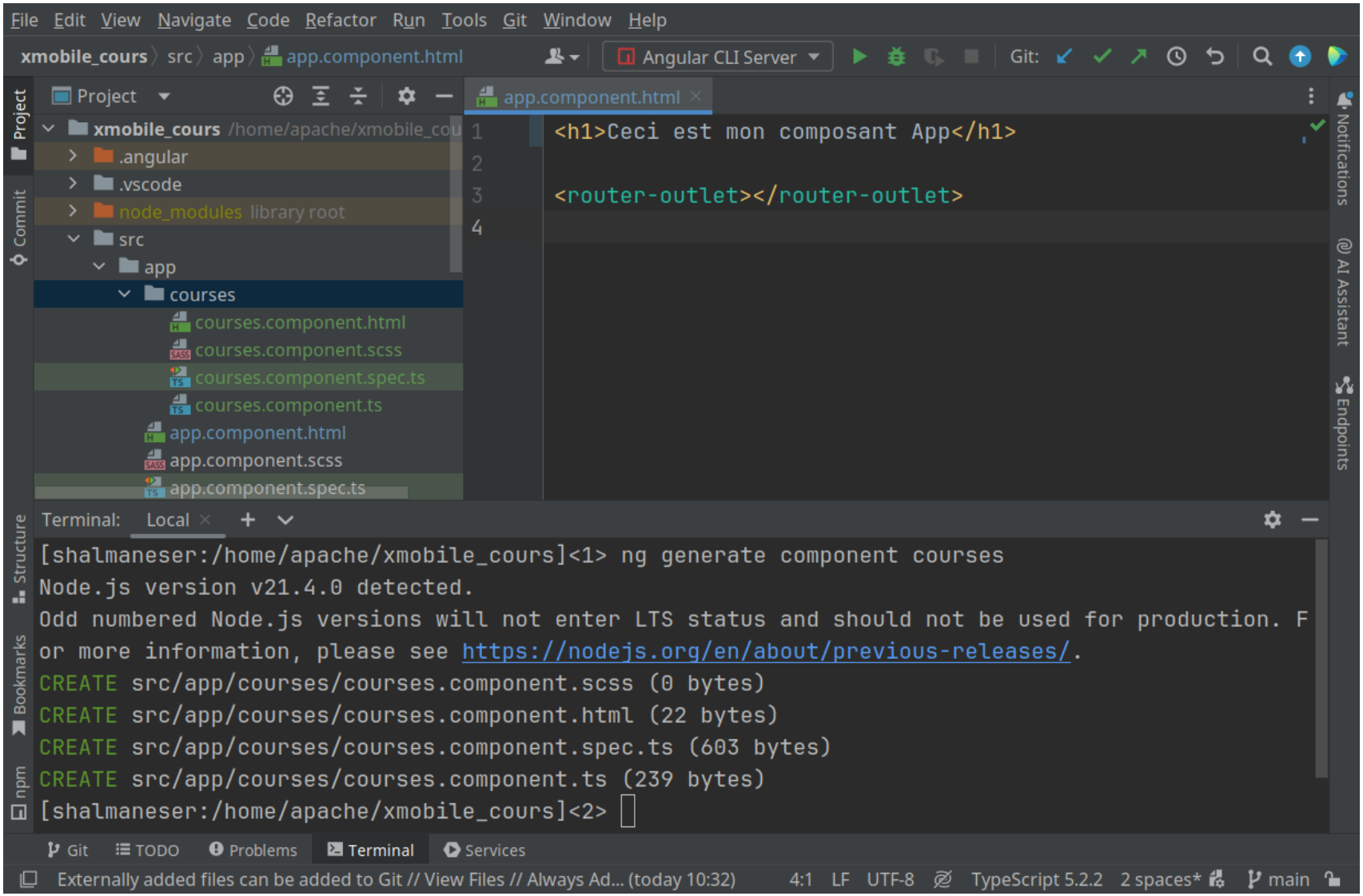
Applications web et mobiles

Christophe Gonzales

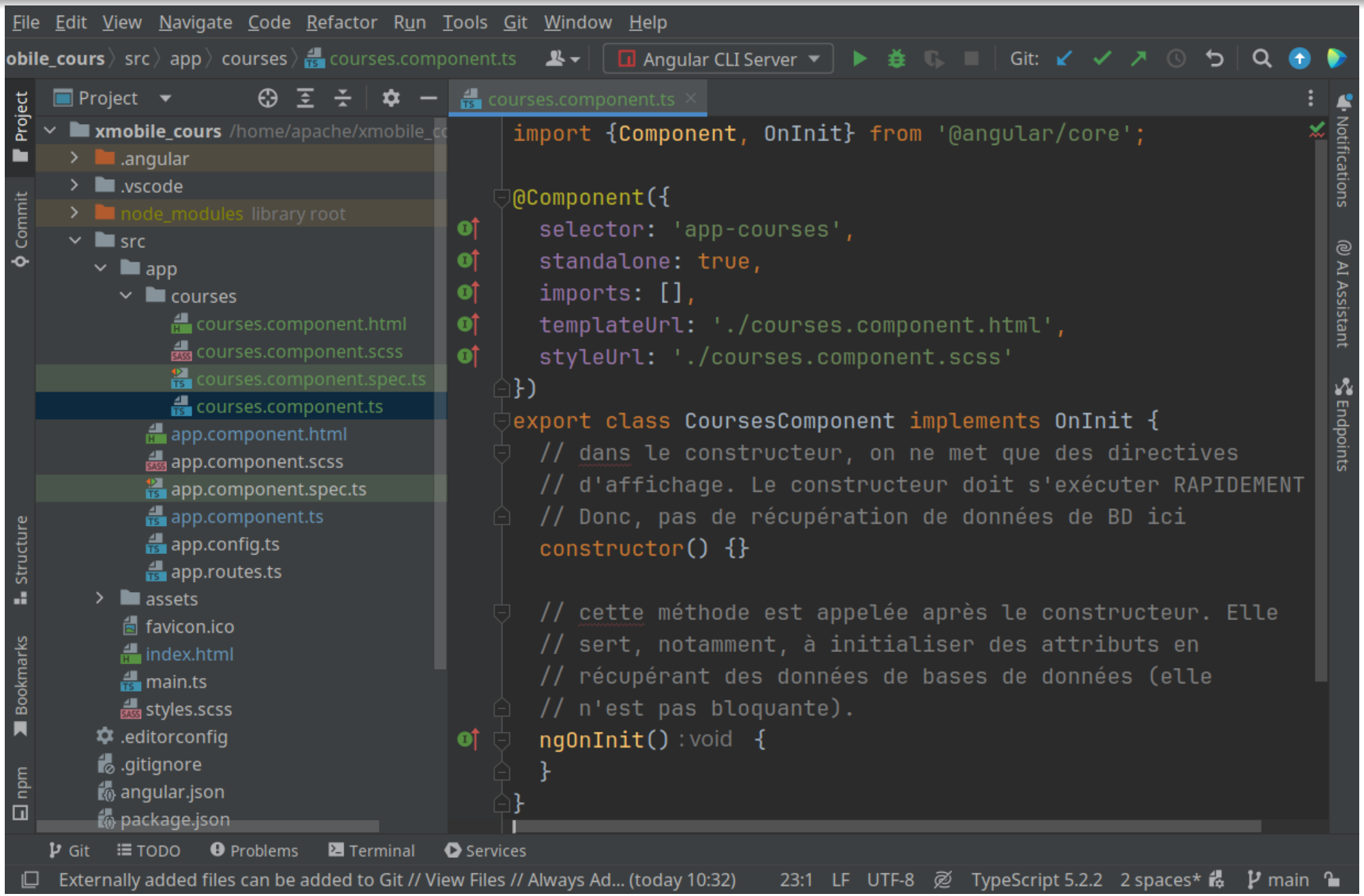
Création d'un nouveau composant

- Utiliser la commande `ng generate component courses`

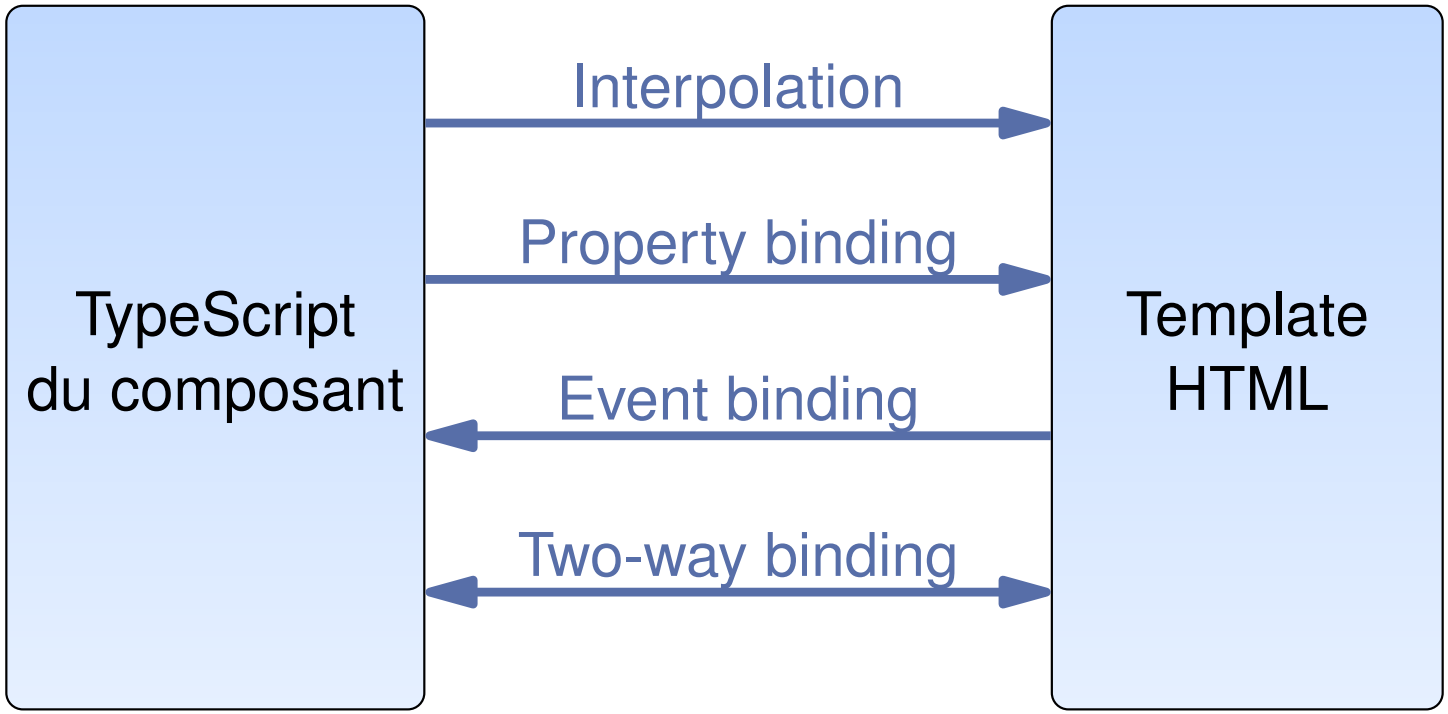
  $\Longrightarrow$ crée un répertoire `courses` et des fichiers spécifiques



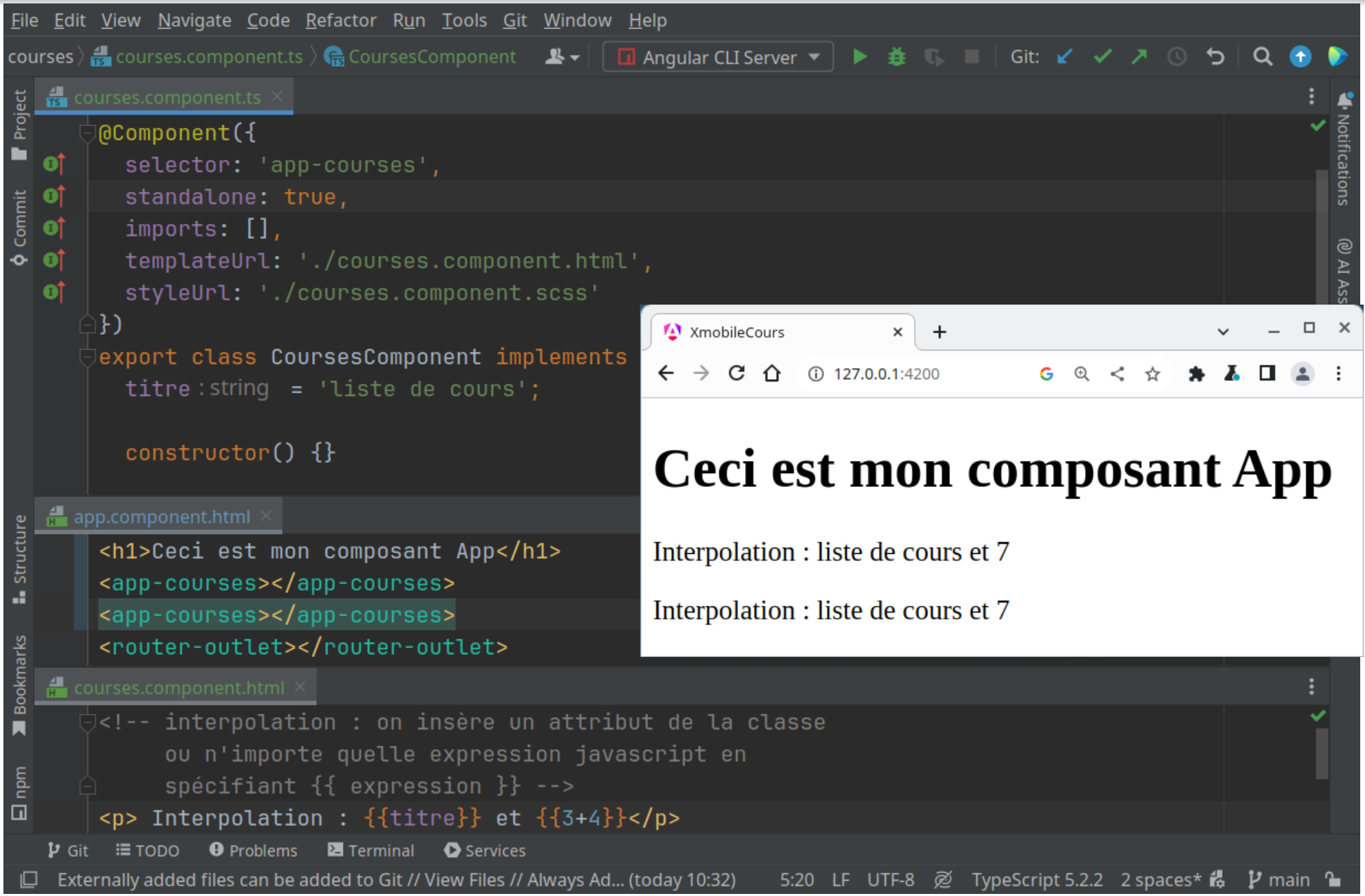
TypeScript du nouveau composant



Interactions entre le TypeScript et le HTML



Interpolation



Interpolation avec des méthodes

File Edit View Navigate Code Refactor Run Tools Git Window Help

courses.component.ts > CoursesComponent > getTitle() | Angular CLI Server | Git:

courses.component.ts

```
  imports: [],
  templateUrl: './courses.component.html',
  styleUrl: './courses.component.scss'
})
export class CoursesComponent implements OnInit {
  titre = 'liste de cours';

  constructor() {}

  ngOnInit() {}

  getTitle() {
    return this.titre;
  }
}
```

XmobileCours — 127.0.0.1:4200

Ceci est mon composant App

Interpolation bis : liste de cours

Interpolation bis : liste de cours

courses.component.html

```
<p> Interpolation bis : {{getTitle()}}</p>
```

Git TODO Problems Terminal Services

Externally added files can be added to Git // View Files // Always Ad... (today 10:32) 17:3 LF UTF-8 TypeScript 5.2.2 2 spaces* main

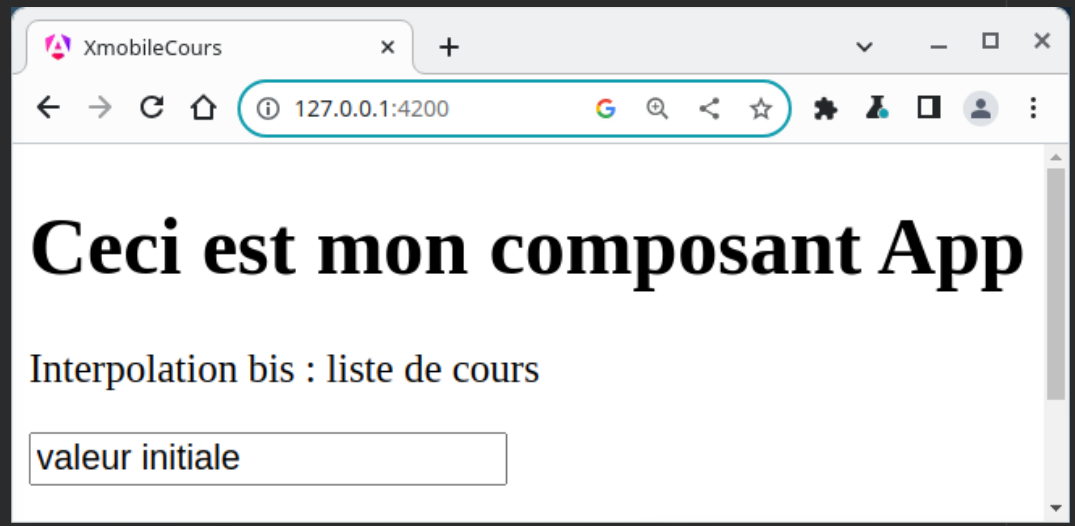
Property binding

File Edit View Navigate Code Refactor Run Tools Git Window Help

obile_cours 〉 src 〉 app 〉 courses 〉 courses.component.ts | Angular CLI Server | Git:

courses.component.ts

```ts
export class CoursesComponent implements OnInit {
  titre = 'liste de cours';
  ma_valeur = 'valeur initiale';

  constructor() {}

  ngOnInit() {}

  getTitle() {
    return this.titre;
  }
}
```



courses.component.html

```html
<p> Interpolation bis : {{getTitle()}}</p>

<!-- Property binding : on insère une valeur dans une propriété d'un élément du DOM.
     L'insertion se fait via :
     [nom_propriété]="valeur provenant du TypeScript" -->
<input type="text" [value]="ma_valeur"/>
```

Git TODO Problems Terminal Services

Externally added files can be added to Git // View Files // Always Ad... (today 10:32) 25:1 LF UTF-8 TypeScript 5.2.2 2 spaces* main

Différence entre interpolation et property binding

- **Interpolation {{}} :**

  Permet de transférer des données du TypeScript n'importe où dans le template HTML

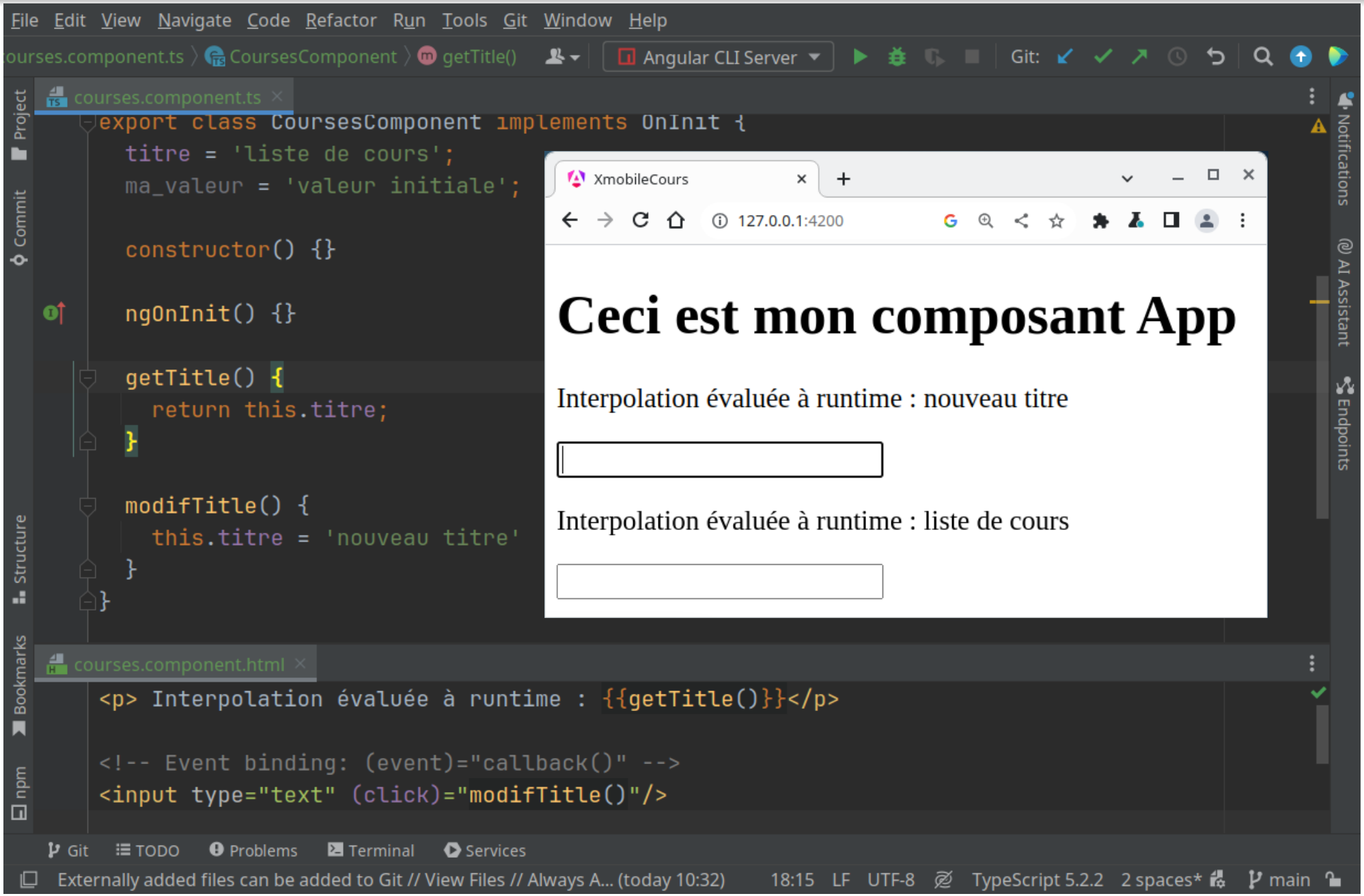
  Évalué à runtime !

- **Property binding [] :**

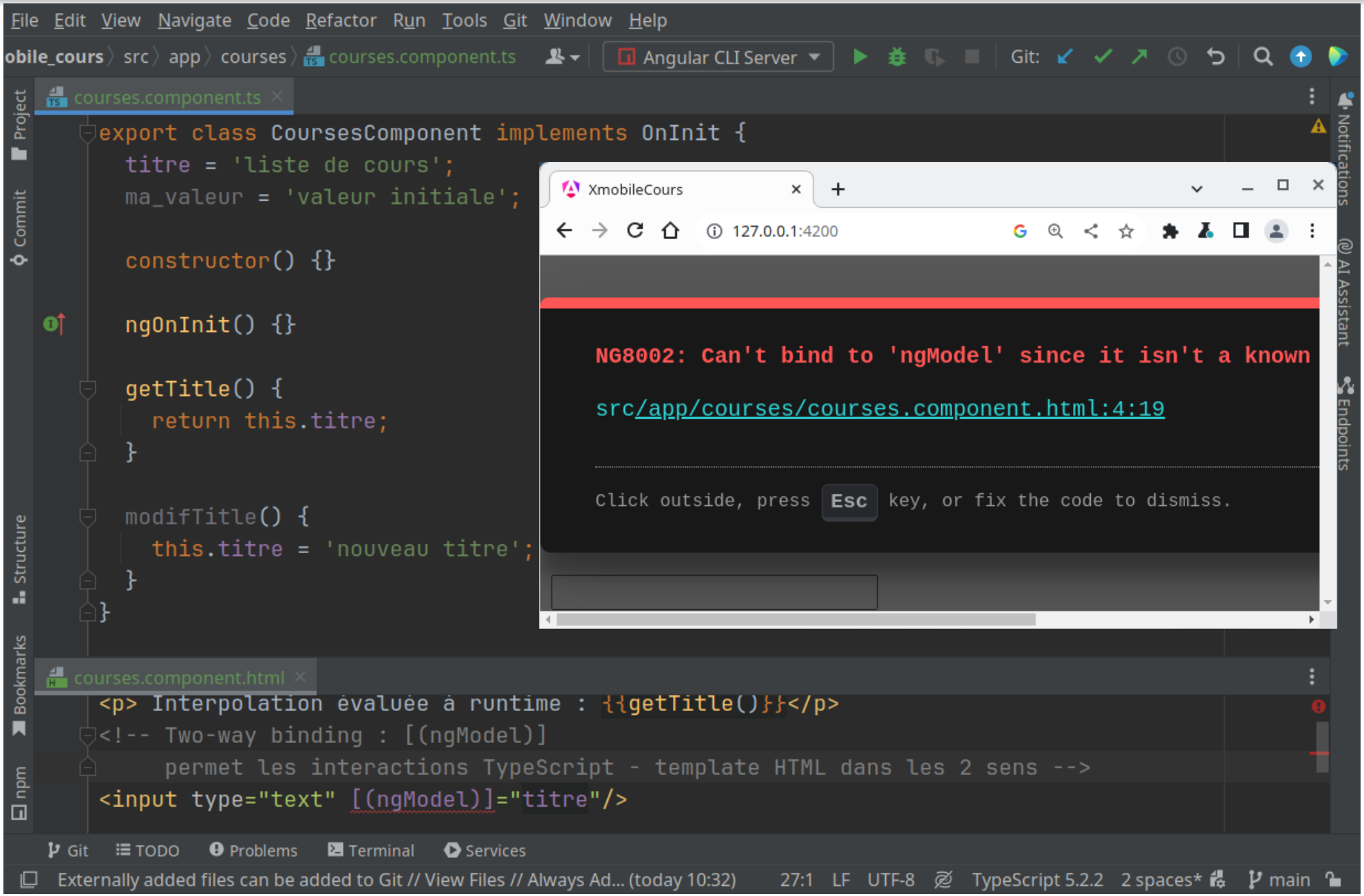
  Permet de mettre des valeurs dans les propriétés des éléments du DOM

  Exemple intéressant : `[hidden]="valeur"`

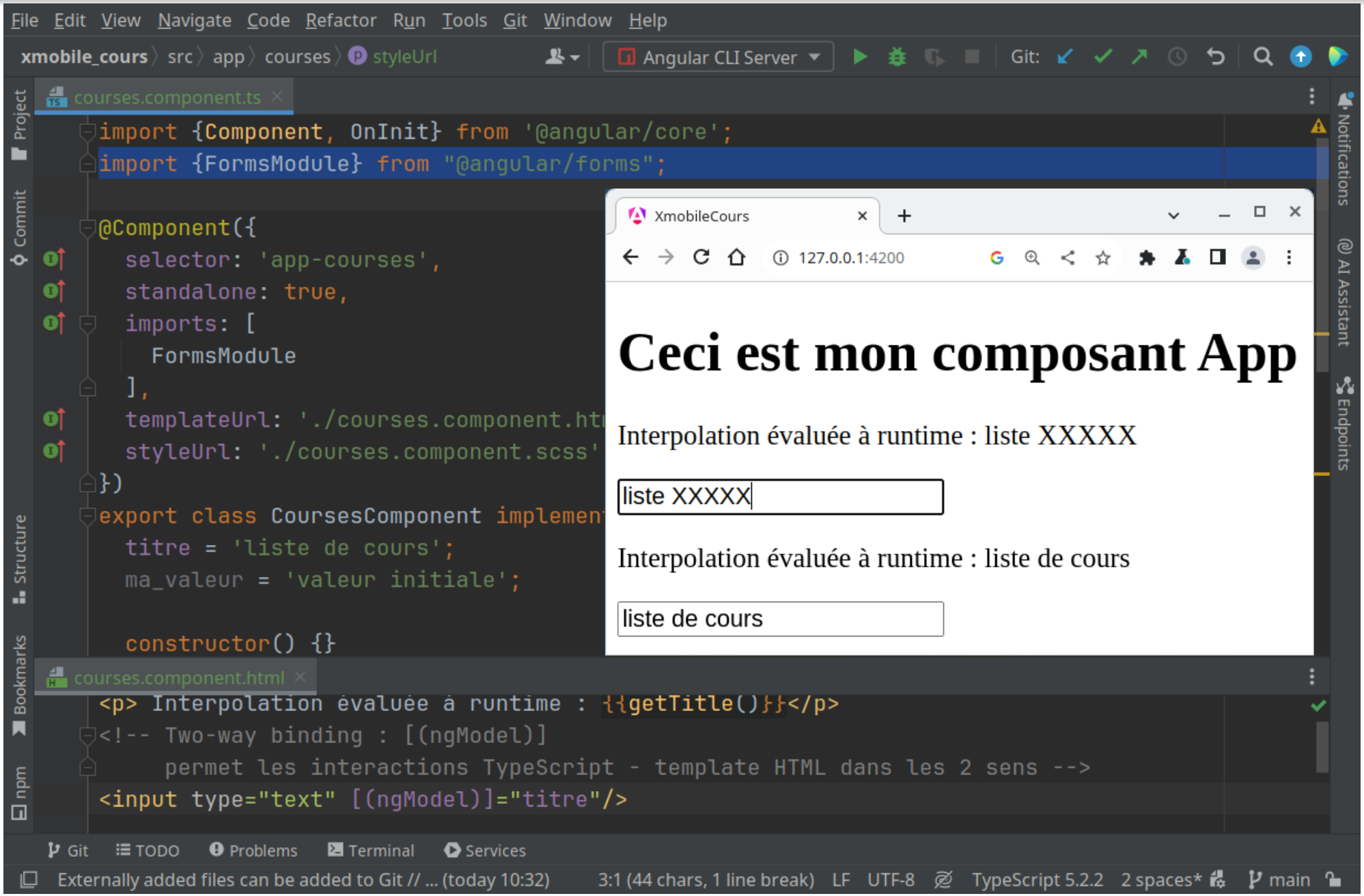
Event binding



Two-way binding (1/2)



Two-way binding (2/2)



Interactions TypeScript-template HTML en résumé

Uniquement de TypeScript vers le template HTML :

- Property Binding : `[propriété]="valeur"`
  affecte des valeurs aux propriétés d'éléments du DOM
- Interpolation : `{{ champ ou méthode }}`
  à utiliser quand on ne peut pas faire de property binding

Uniquement du template HTML vers le TypeScript :

- Event binding : (event)=”méthode()”
  Appelle la méthode quand un événement du DOM arrive

Dans les deux sens :

- Two-way binding : [(ngModel)]=”valeur”

ne pas oublier d'importer FormsModule

Interfaces et données

File Edit View Navigate Code Refactor Run Tools Git Window Help

courses.component.ts > CoursesComponent > UE | Angular CLI Server | Git:

courses.component.ts

```ts
import {Component, OnInit} from '@angular/core';
import {FormsModule} from "@angular/forms";

// interface imposant les informations que doit contenir un cours
export interface Course {
  nom: string;         // on indique le nom des attributs ainsi
  nb_etuds : number;   // que leur type
}
```

courses.component.ts

```ts
})
export class CoursesComponent implements
  titre = 'liste de cours';
  UE : Course[] = [
    {nom: 'c1', nb_etuds: 3},
    {nom: 'c2', nb_etuds: 5}
  ];
```

courses.component.html

```html
<p>{{getTitle()}}</p>
<ul>
  <li>{{UE[0].nom}} : {{UE[0].nb_etuds}} étuds</li>
  <li>{{UE[1].nom}} : {{UE[1].nb_etuds}} étuds</li>
</ul>
```

XmobileCours — 127.0.0.1:4200

Ceci est mon composant App

liste de cours

- c1 : 3 étuds
- c2 : 5 étuds

Git TODO Problems Terminal Services

21:20 LF UTF-8 TypeScript 5.2.2 2 spaces* main

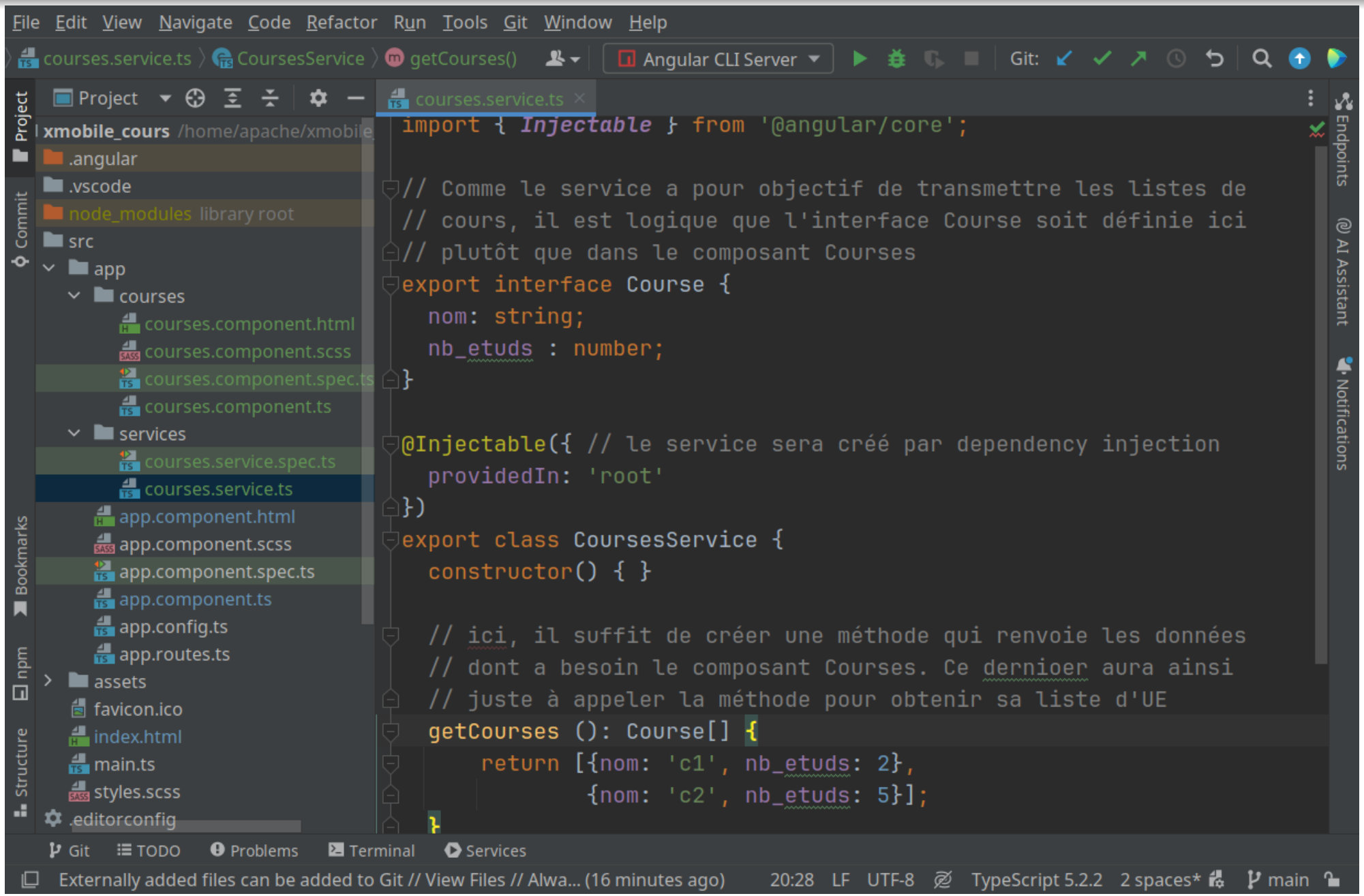
Les services : providers de données

- Actuellement : UE stockées en « dur » dans composant Courses
- Vraie application : UE récupérées d'un serveur

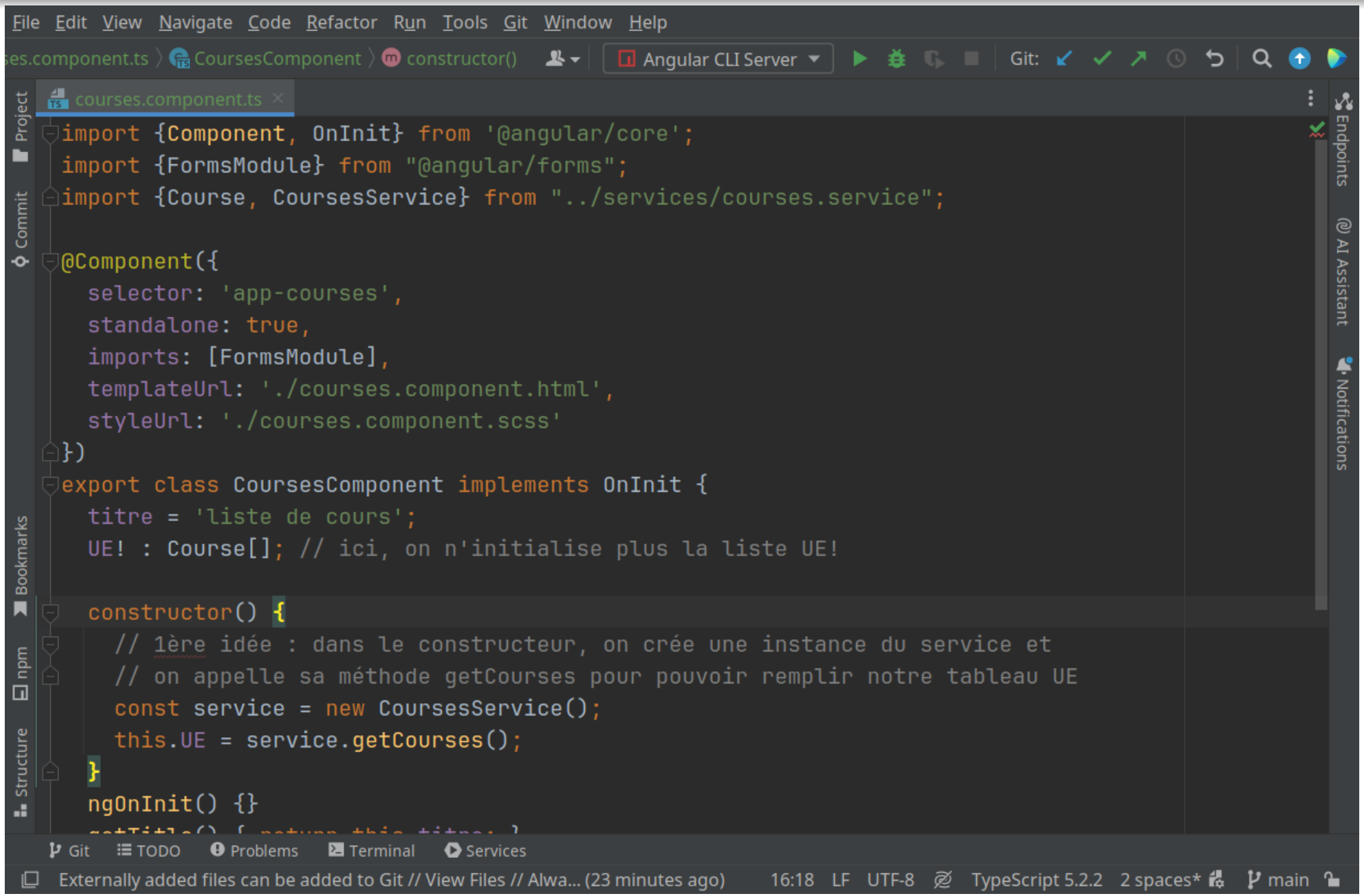
 : les services récupèrent les données

- Créer un service : `ng generate service nom_service`
- Utiliser le service comme n'importe quelle classe

Anatomie d'un service



Exploitation du service dans le composant Courses



Exploitation du service : bonnes et mauvaises idées

Bonne idée :

- Faire en sorte que le constructeur connaisse le service

Mauvaises idées :

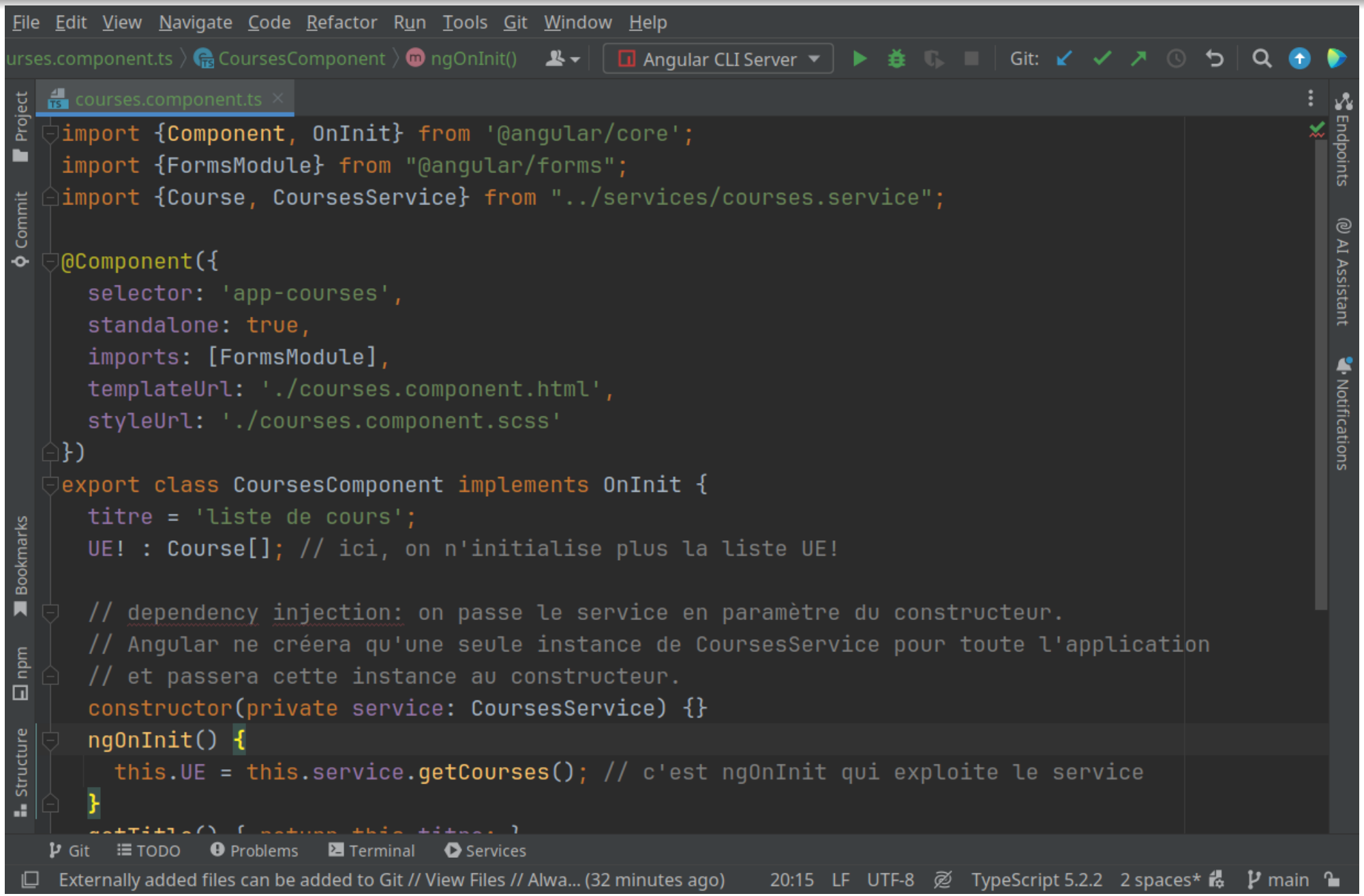
- Demander au constructeur de créer l'instance
  - Plusieurs composants $\Longrightarrow$ plusieurs instances
  - Modif du constructeur du service $\Longrightarrow$ modif du composant
- Utiliser le service dans le constructeur

  $\Longrightarrow$ délais dans les affichages

 Bonne pratique des services :

- Dependency injection
- Utiliser le service dans la méthode ngOnInit

Dependency injection



Asynchronie et observables

- Service actuel : synchrone

  $\Longrightarrow$ ngOnInit doit attendre les données

- Services HTTP : asynchrones

  $\Longrightarrow$ évite de bloquer les affichages

Utilisation de services asynchrones :

1. Le service retourne tout de suite un ***Observable***
2. ngOnInit appelle le service et récupère l'observable
3. ngOnInit souscrit à l'observable en donnant une callback
4. ngOnInit continue son exécution
5. Les données arrivent $\Longrightarrow$ l'observable émet une valeur

   $\Longrightarrow$ la callback est appelée

Mise en place des observables dans le service

File Edit View Navigate Code Refactor Run Tools Git Window Help

courses.service.ts 〉 CoursesService 〉 getCourses() | Angular CLI Server | Git:

courses.service.ts

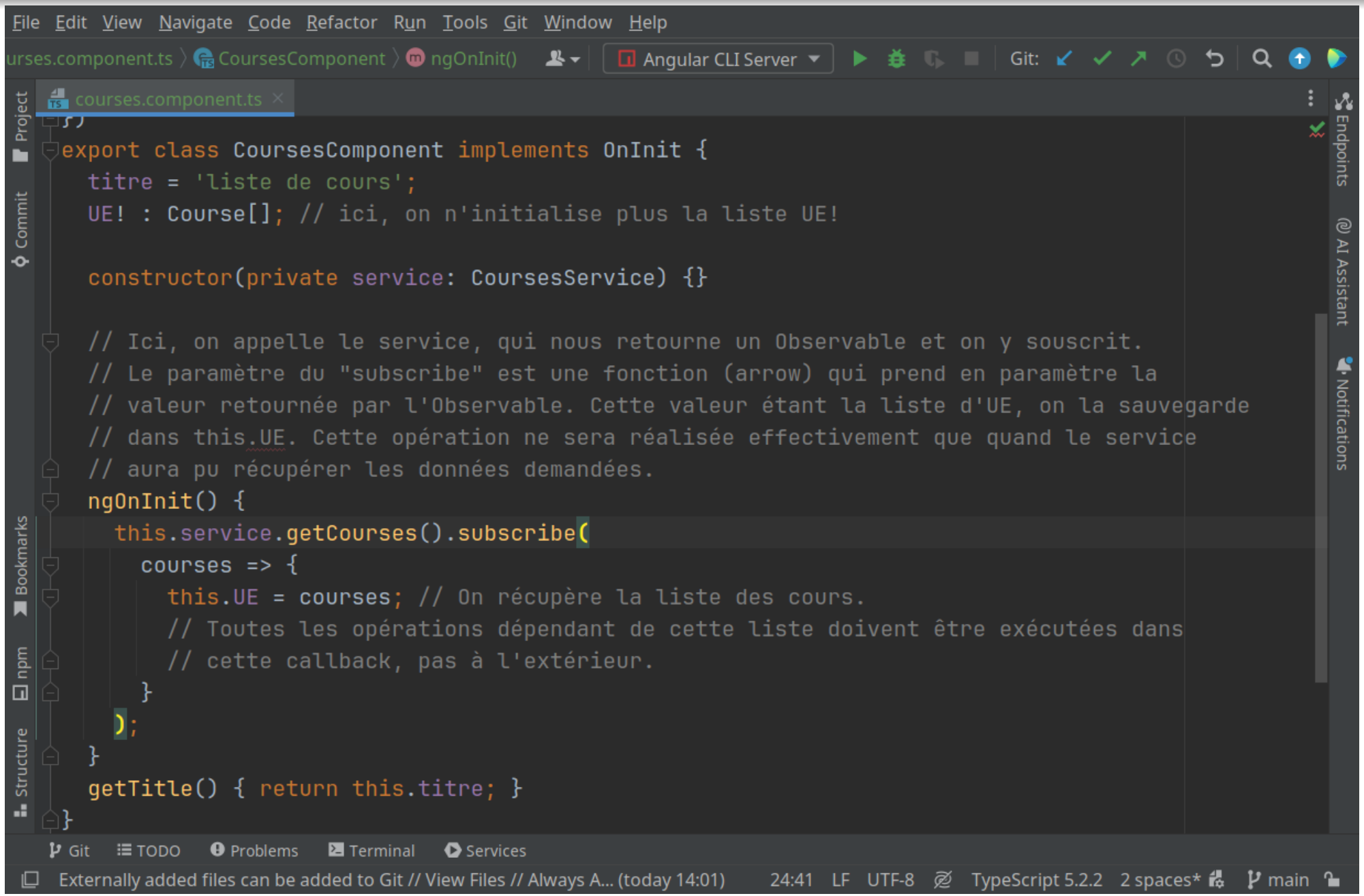
```typescript
import { Injectable } from '@angular/core';
// Au lieu de renvoyer un tableau de Course, on va renvoyer un Observable sur ce
// tableau => il faut importer les déclarations des Observables :
import {Observable, of} from "rxjs";

export interface Course {
  nom: string;
  nb_etuds : number;
}

@Injectable({ // le service sera créé par dependency injection
  providedIn: 'root'
})
export class CoursesService {
  constructor() { }

  // Ici, on retourne un Observable sur un tableau de Course
  getCourses (): Observable<Course[]> {
      return of([{nom: 'c1', nb_etuds: 2},
                 {nom: 'c2', nb_etuds: 5}]);
  }
}
```

Project Commit Bookmarks npm Structure Endpoints AI Assistant Notifications

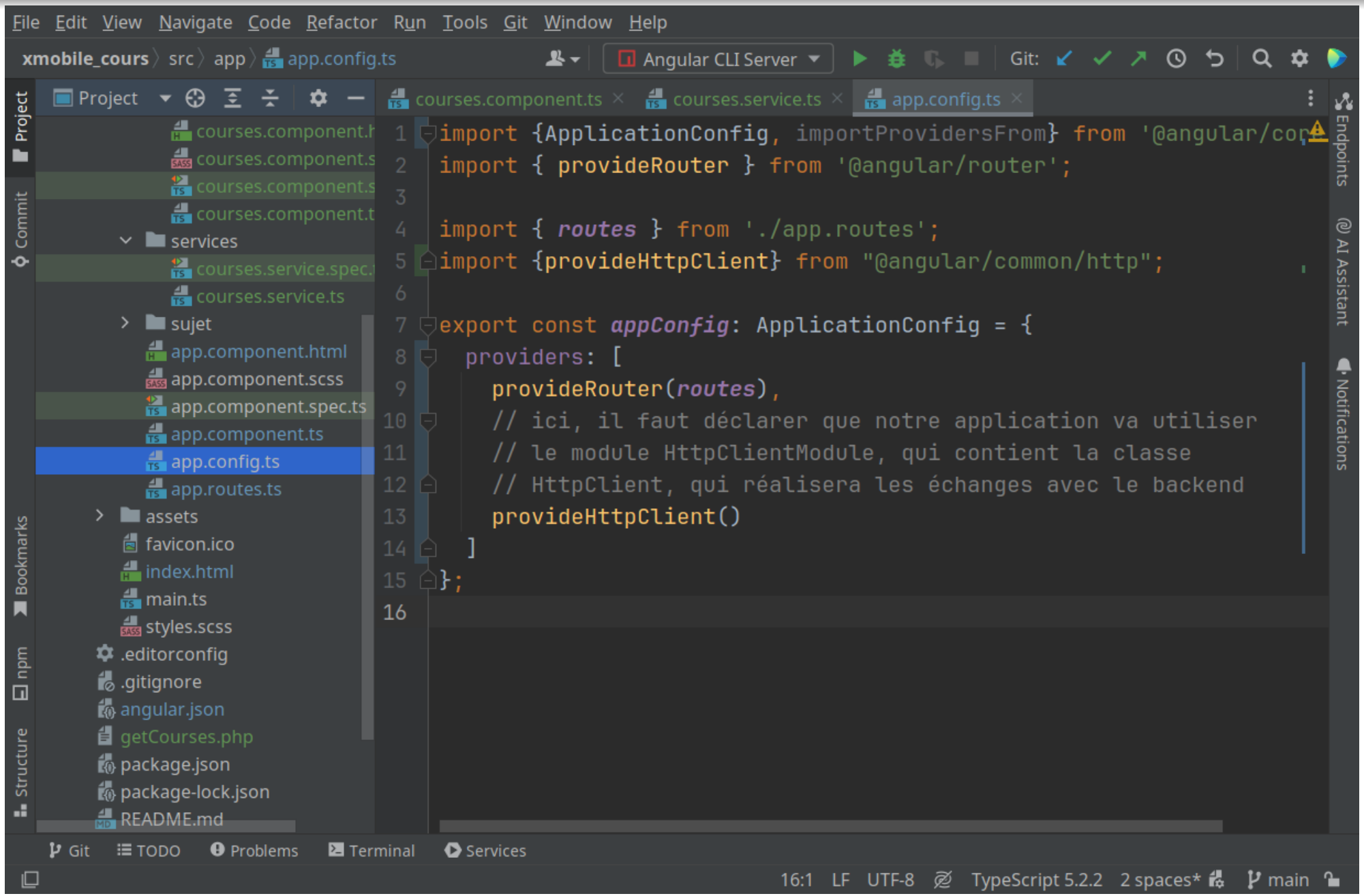
Git TODO Problems Terminal Services

Externally added files can be added to Git // View Files // Always A... (today 14:01) 18:40 LF UTF-8 TypeScript 5.2.2 2 spaces* main

Mise en place des observables dans le composant



Service HTTP : 1 importer le HttpClientModule



Service HTTP : 2 notre service utilise HttpClient

File Edit View Navigate Code Refactor Run Tools Git Window Help

courses.service.ts 〉 CoursesService 〉 getCourses() | Angular CLI Server | Git:

courses.service.ts

```typescript
import { Injectable } from '@angular/core';
// On a toujours besoin des Observables, mais on rajoute HttpClient, qui va
// s'occuper de converser avec le serveur web.
import {Observable, of} from "rxjs";
import {HttpClient} from "@angular/common/http";

export interface Course {
  nom: string;
  nb_etuds : number;
}
@Injectable({
  providedIn: 'root'
})
export class CoursesService {
  // ici, dependency injection : on va utiliser le HttpClient pour interroger le backend
  constructor(private http: HttpClient) { }

  // Ici, on retourne un Observable sur un tableau de Course
  getCourses (): Observable<Course[]> {
    return this.http.get<Course[]>( // on converse avec le backend via une méthode GET
      'http://127.0.0.1/xmobile_cours/getCourses.php'
    );
  }
```

Git TODO Problems Terminal Services

Externally added files can be added to Git // View Files // Always A... (today 14:01) 19:40 LF UTF-8 TypeScript 5.2.2 2 spaces* main

Service HTTP : 3 getCourses.php

File Edit View Navigate Code Refactor Run Tools Git Window Help

xmobile_cours 〉 getCourses.php — Angular CLI Server — Git:

getCourses.php

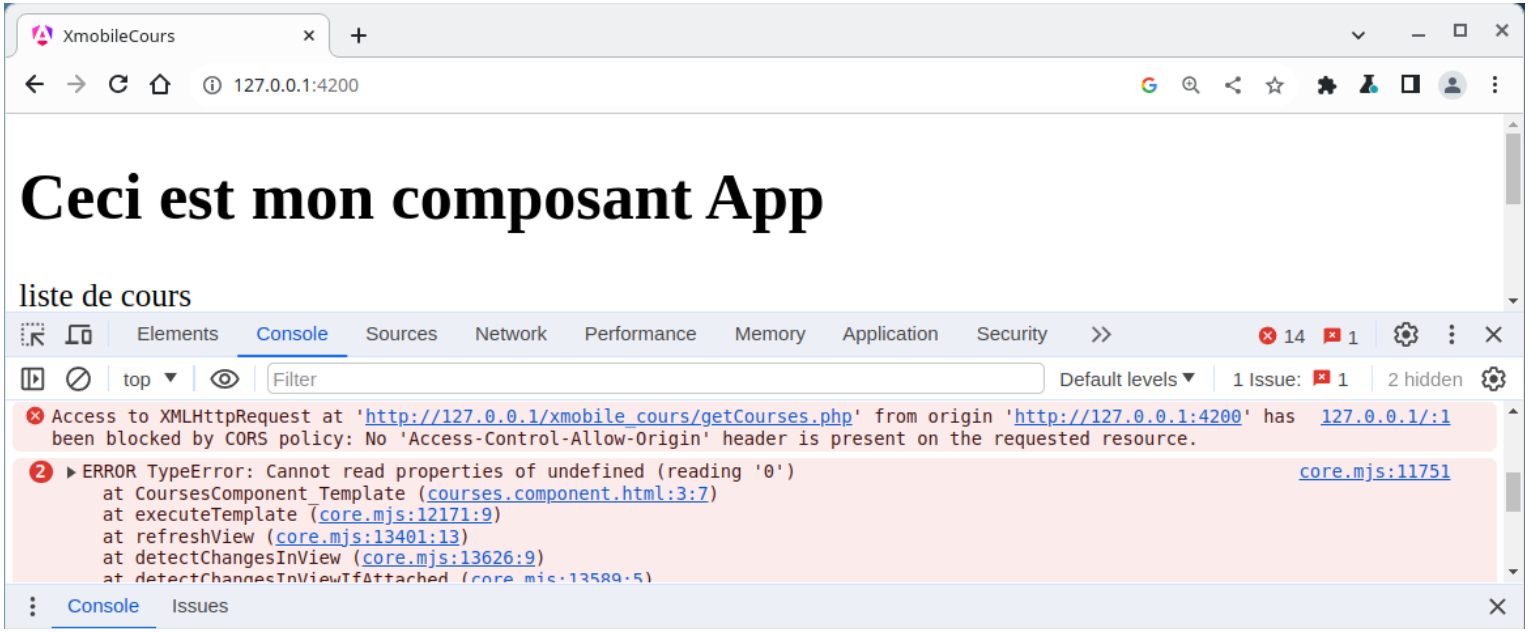
```php
<?php

// on indique ici que le script va transmettre des données JSON
header('Content-type:application/json;charset=utf8');

echo json_encode ([
  [ 'nom' => 'c1', 'nb_etuds' => 2 ],
  [ 'nom' => 'c2', 'nb_etuds' => 5 ],
  [ 'nom' => 'c3', 'nb_etuds' => 6 ]
]);

?>
```

Project Commit Bookmarks npm Structure Endpoints AI Assistant Notifications

Git TODO Problems Terminal Services

Externally added files can be added to Git // View Files // Always Ad... (today 14:01) 13:1 LF UTF-8 TypeScript 5.2.2 2 spaces* main

Service HTTP : Résultat

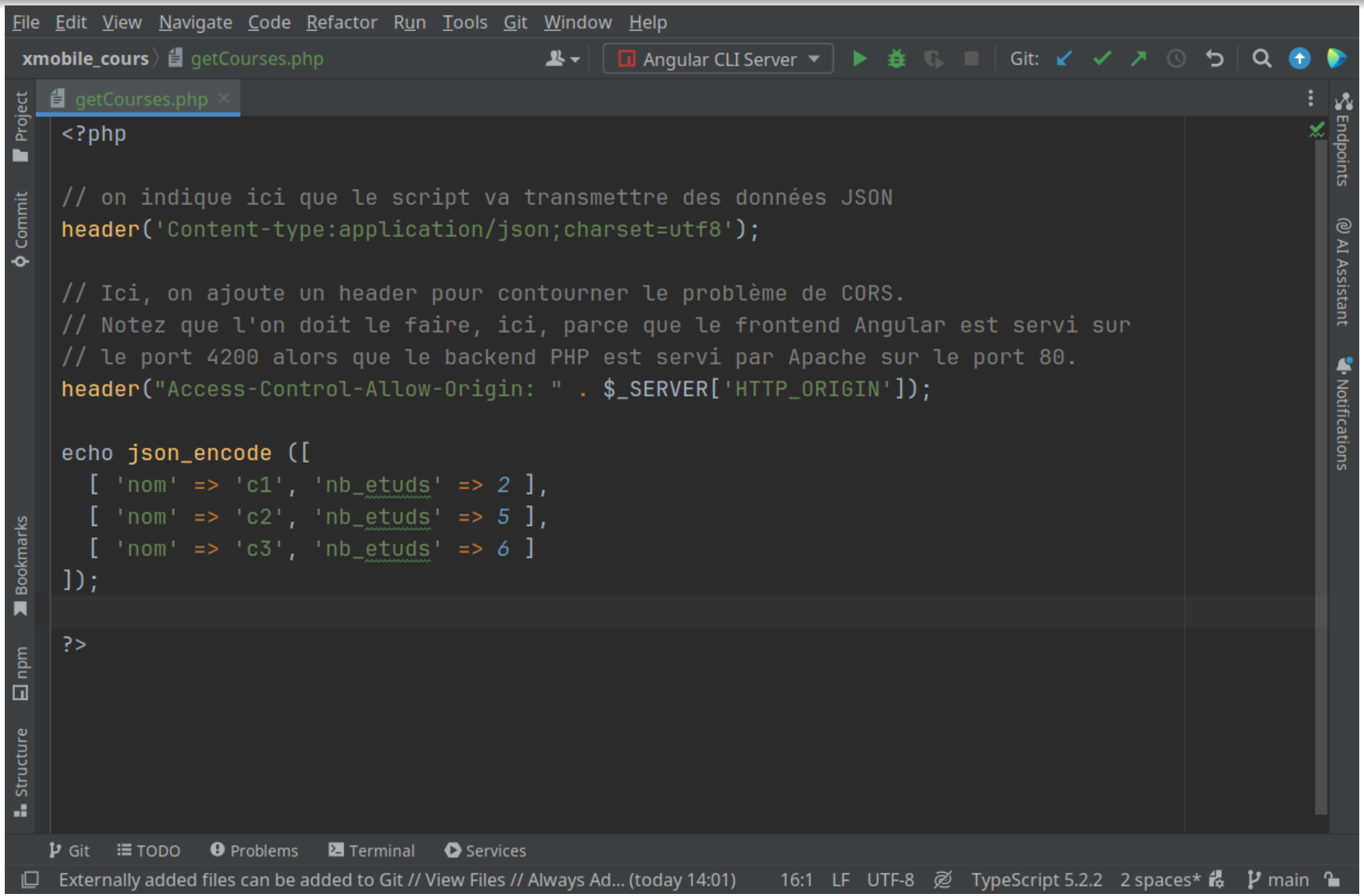


- **Cross-Origin Resource Sharing (CORS) :**
- Requête cross-origine : provient de 127.0.0.1:4200
  accède à 127.0.0.1:80

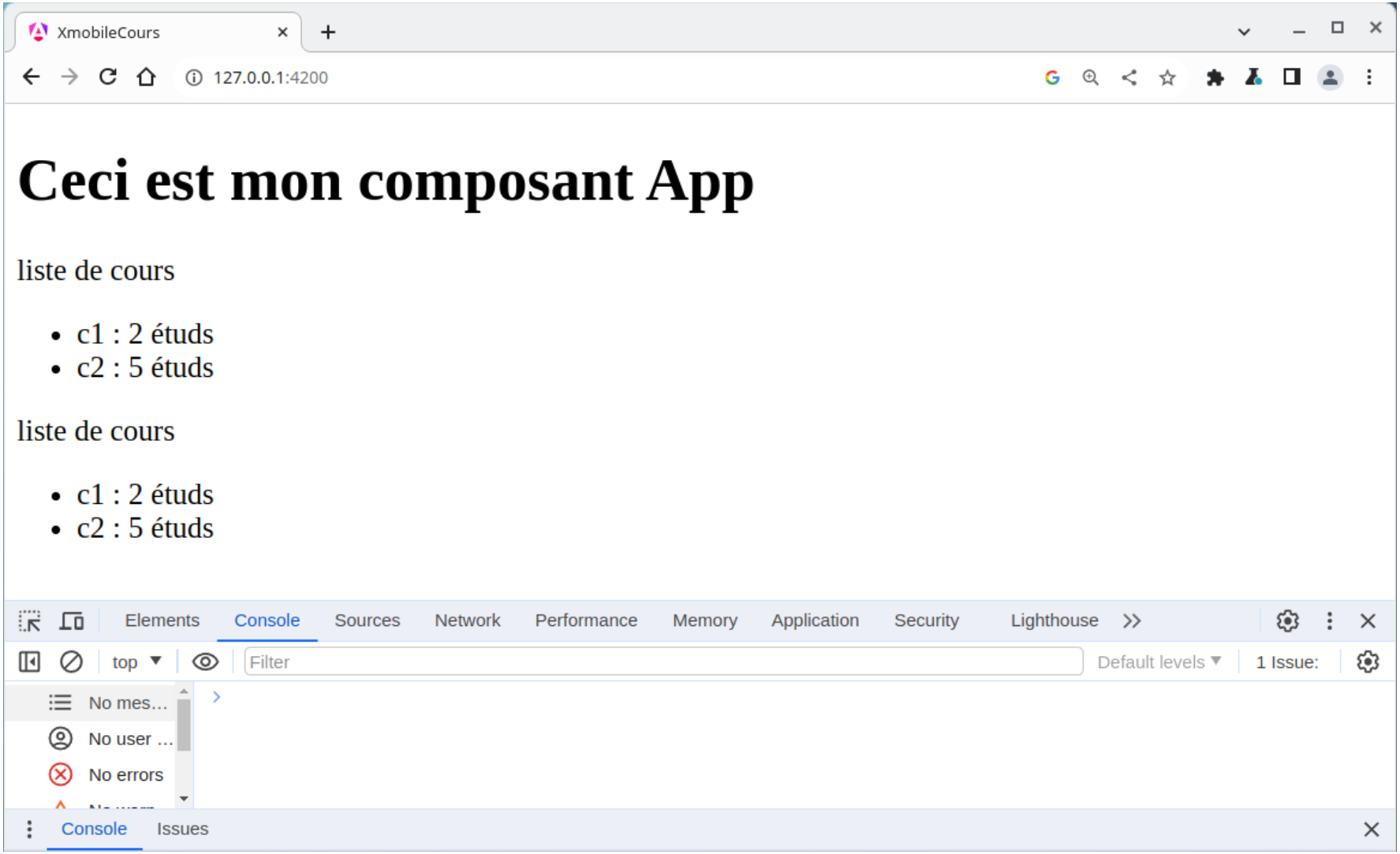
  $\Longrightarrow$ CORS refuse la requête
- **Contre-mesure :** dans getCourses.php, rajouter :

```
header("Access-Control-Allow-Origin: *");
```

Service HTTP : 3 le bon getCourses.php



Service HTTP : le bon résultat



Service HTTP : 2 HttpClient avec méthode POST

File Edit View Navigate Code Refactor Run Tools Git Window Help

courses.service.ts 〉 CoursesService 〉 getCourses() | Angular CLI Server | Git:

courses.service.ts

```typescript
import {Observable, of} from "rxjs";
import {HttpClient} from "@angular/common/http";

export interface Course {
  nom: string;
  nb_etuds : number;
}
@Injectable({
  providedIn: 'root'
})
export class CoursesService {
  // ici, dependency injection : on va utiliser le HttpClient pour interroger le backend
  constructor(private http: HttpClient) { }

  // Ici, on retourne un Observable sur un tableau de Course
  getCourses (): Observable<Course[]> {
    return this.http.post<Course[]>( // on converse avec le backend via une méthode GET
      'http://127.0.0.1/xmobile_cours/getCourses.php', // url du backend
      null                                             // les données qu'on envoie par POST
    );
  }
}
```

Git TODO Problems Terminal Services

Externally added files can be added to Git // View Files // Alwa... (today 14:01) 19:40 LF UTF-8 TypeScript 5.2.2 2 spaces* main

Service HTTP : 3 méthode POST et session

File Edit View Navigate Code Refactor Run Tools Git Window Help

courses.service.ts 〉 CoursesService 〉 getCourses() | Angular CLI Server | Git:

courses.service.ts

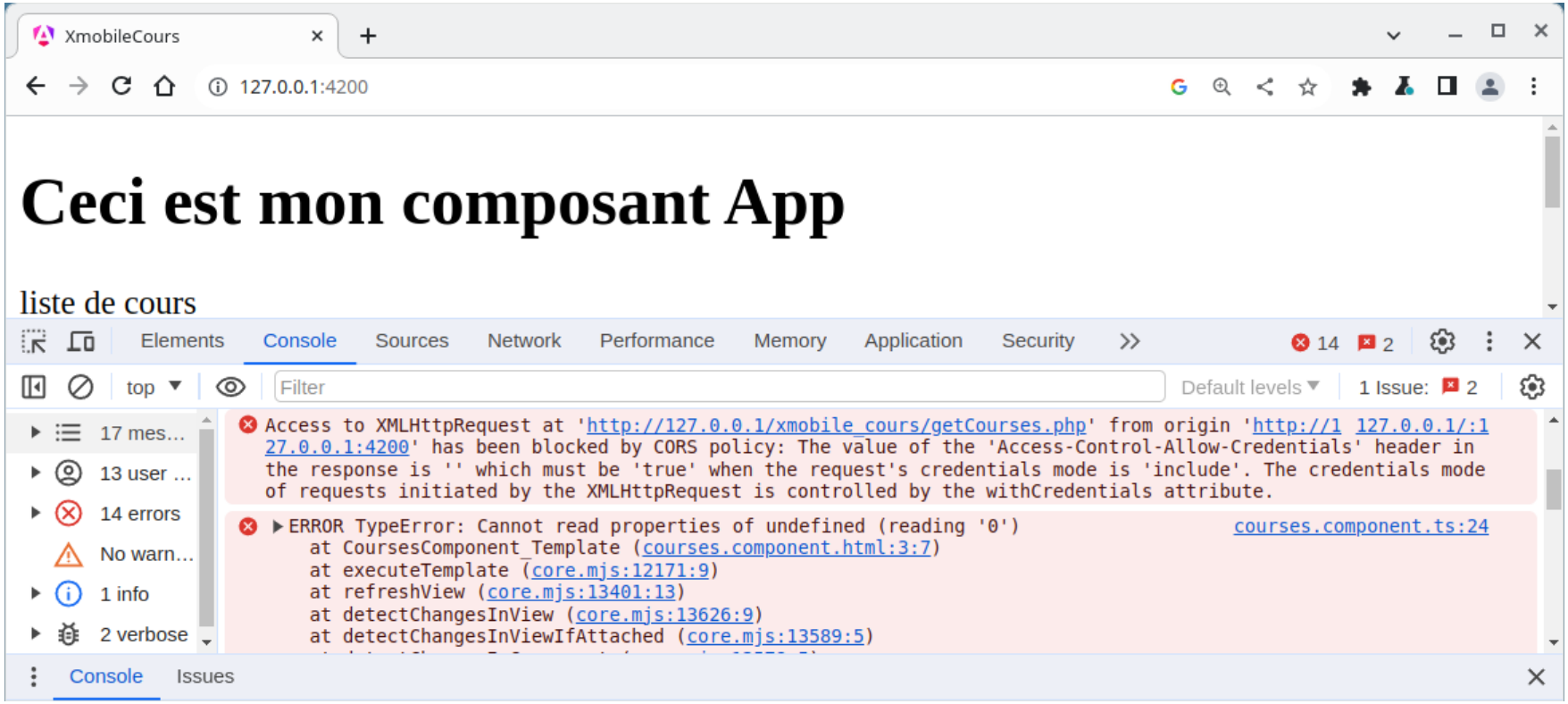
```typescript
import {Observable, of} from "rxjs";
import {HttpClient} from "@angular/common/http";

export interface Course {
  nom: string;
  nb_etuds : number;
}
@Injectable({
  providedIn: 'root'
})
export class CoursesService {
  // ici, dependency injection : on va utiliser le HttpClient pour interroger le backend
  constructor(private http: HttpClient) { }

  // Ici, on retourne un Observable sur un tableau de Course
  getCourses (): Observable<Course[]> {
    return this.http.post<Course[]>( // on converse avec le backend via une méthode GET
      'http://127.0.0.1/xmobile_cours/getCourses.php', // url du backend
      null,                                            // les données qu'on envoie par POST
      {withCredentials: true} // pour capturer les cookies de session
    );
  }
}
```

Project Commit Bookmarks npm Structure Endpoints AI Assistant Notifications

Git TODO Problems Terminal Services

Externally added files can be added to Git // View Files // Alwa... (today 14:01) 23:30 LF UTF-8 TypeScript 5.2.2 2 spaces* main

Service HTTP : 4 problème et solution



- **Contre-mesures :** dans getCourses.php :

```
header("Access-Control-Allow-Origin: " .
        $_SERVER['HTTP_ORIGIN']);
header("Access-Control-Allow-Credentials: true");
```

Service HTTP en résumé

1. Dans `app.config.ts` : importer le HttpClientModule
2. Faire `ng generate service mon_service`
3. `mon_service` importe la classe HttpClient
4. Constructeur de `mon_service` : dependency injection de HttpClient
5. Méthodes qui appellent `get` ou `post` de HttpClient
   $\Longrightarrow$ renvoient des observables
6. Composant « client » importe le service
7. Dans son ngOnInit, on récupère les observables et on y souscrit
   Le code dépendant des données est dans la callback en paramètre de `subscribe`