**Formation Angular - Ateliers**

**Ateliers 1 : Préparation d'environnement de travail**

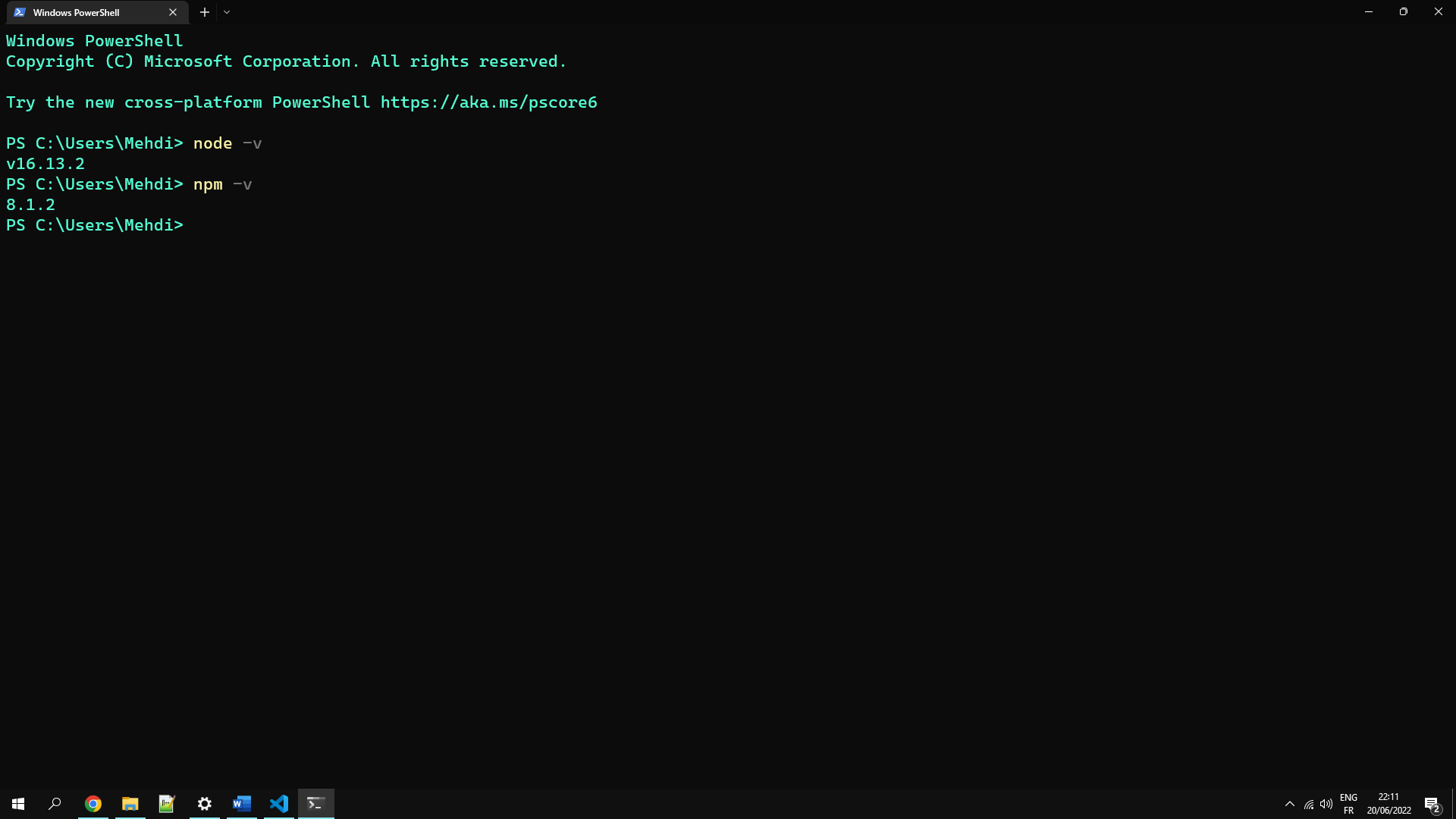
**Objectifs à atteindre :**

1. Installer NodeJs
2. Installer Angular CLI
3. Installer VS Code
4. Créer votre 1er projet « Hello world »

Etapes :

- Télécharger et installer du nodejs via le lien suivant : https://nodejs.org/en/download/

- Vérification d’installation via les deux commandes suivantes : node -v et npm -v



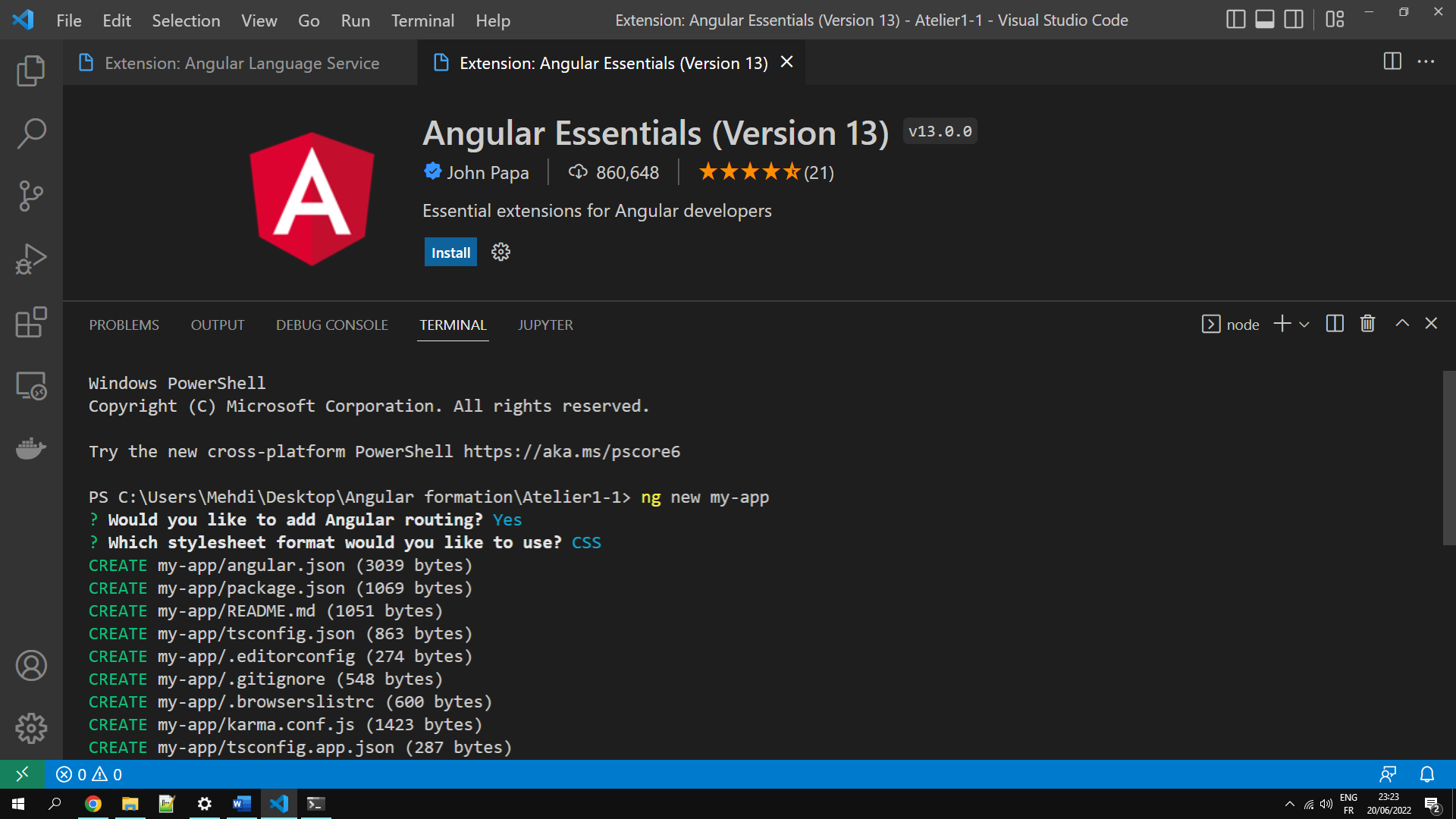
- Installer Angular CLI via la commande suivante : npm install -g @angular/cli

- Télécharger et installer du VS Code via le lien suivant : https://code.visualstudio.com/download

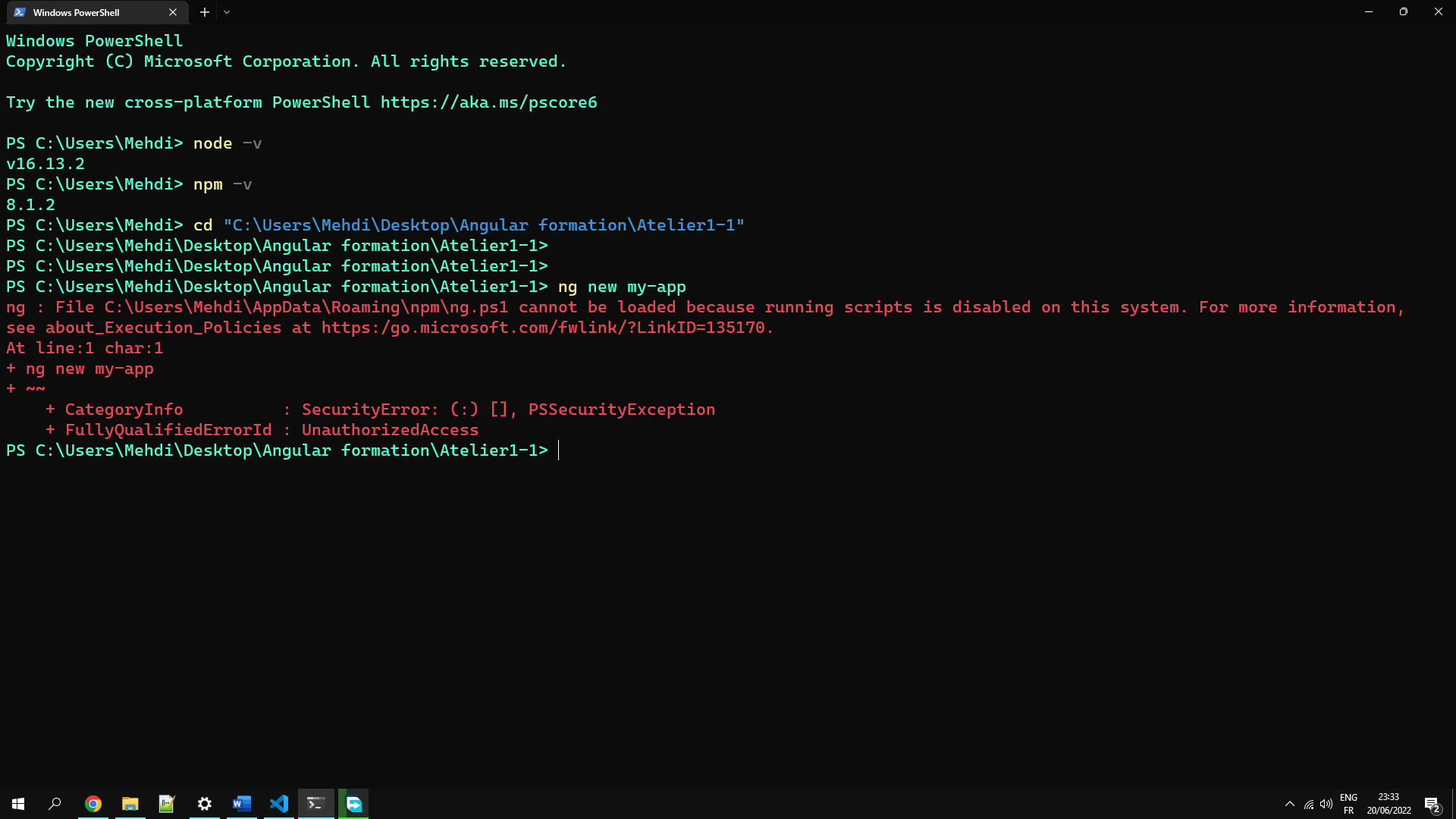
- Dans le VS Code installer l’extension : « Angular Language Service »



- Créer un nouveau projet via la commande suivante : ng new my-app



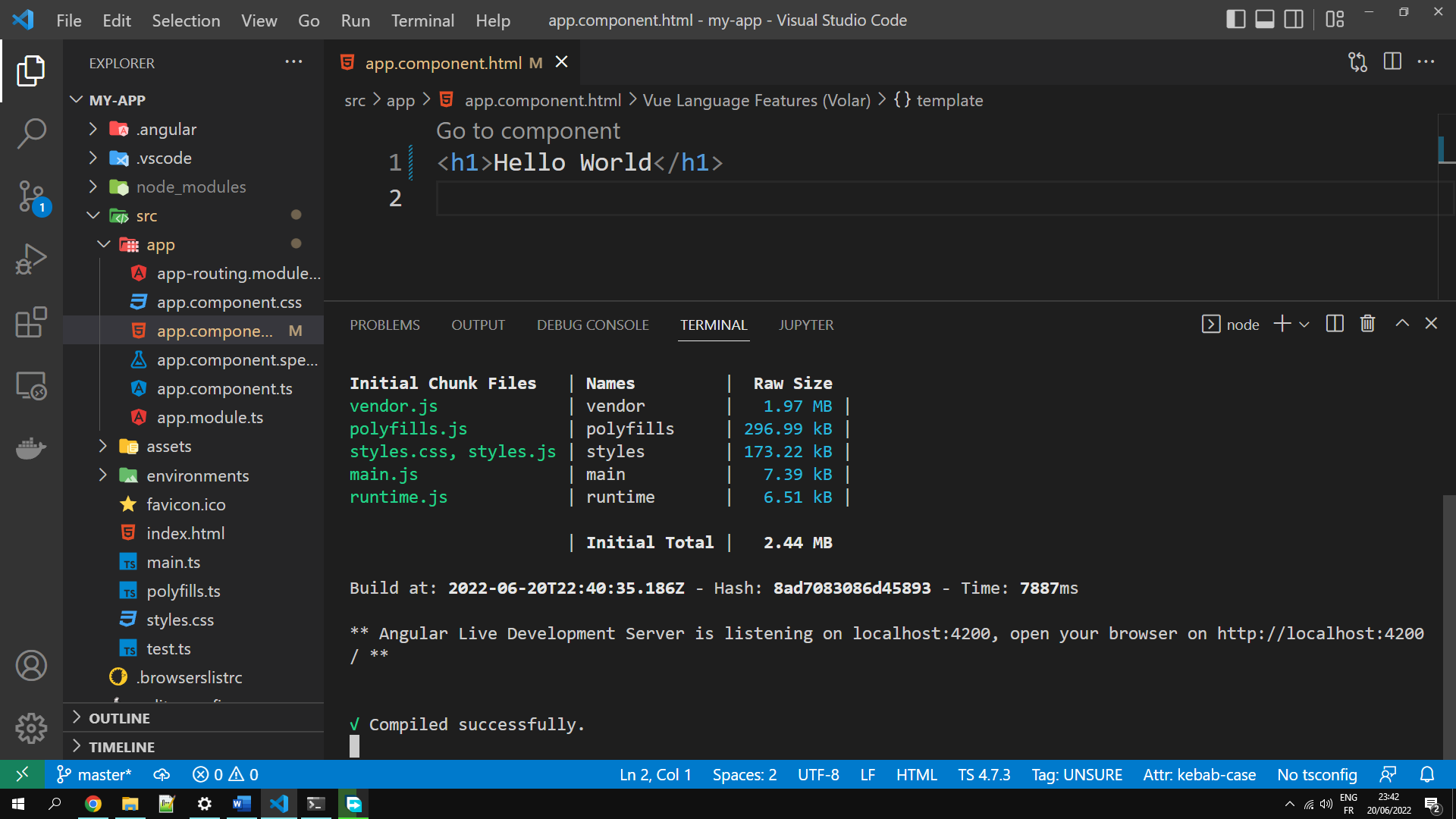
- Si vous obtenez ce message d'erreur, il faut d'exécuter cette commande et redémarre le terminal :Set-ExecutionPolicy RemoteSigned



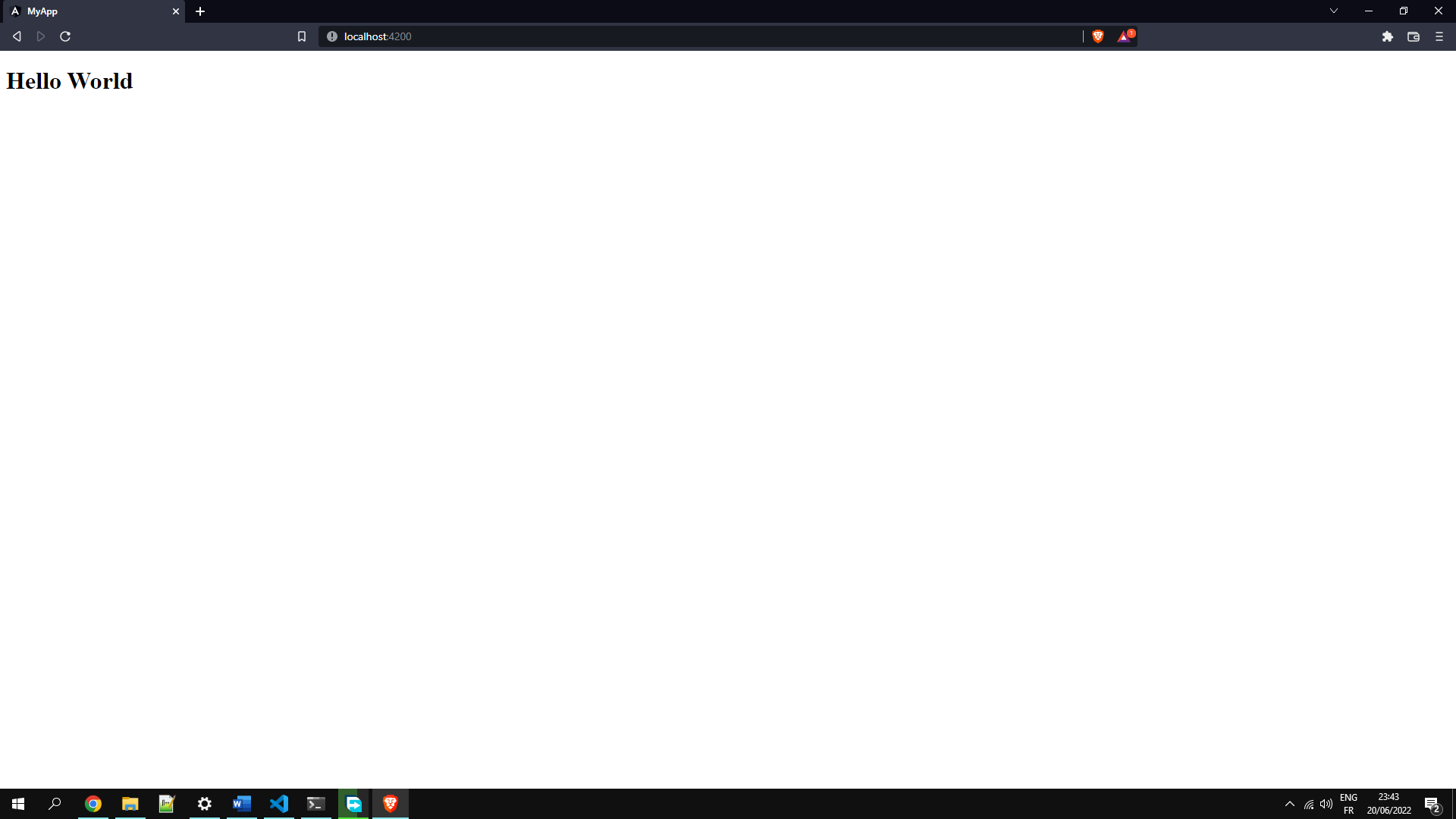
- Remplacer le contenu du fichier src/app/app.component.html projet par :



- Démarrer le projet via la commande suivante : ng serve



- Consulter la page suivante : <http://localhost:4200>



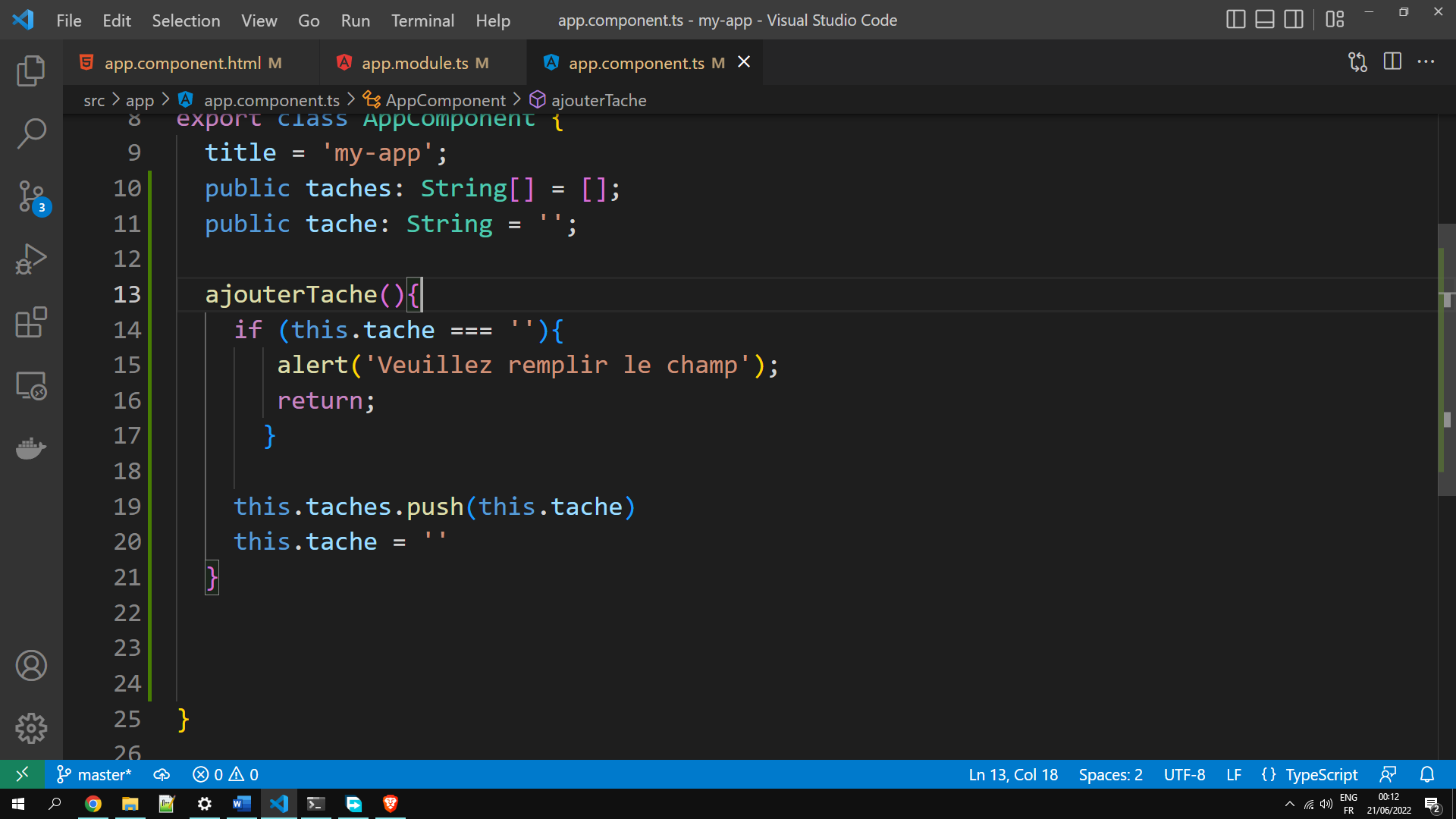
**Ateliers 2 : Les basiques**

**Objectifs à atteindre :**

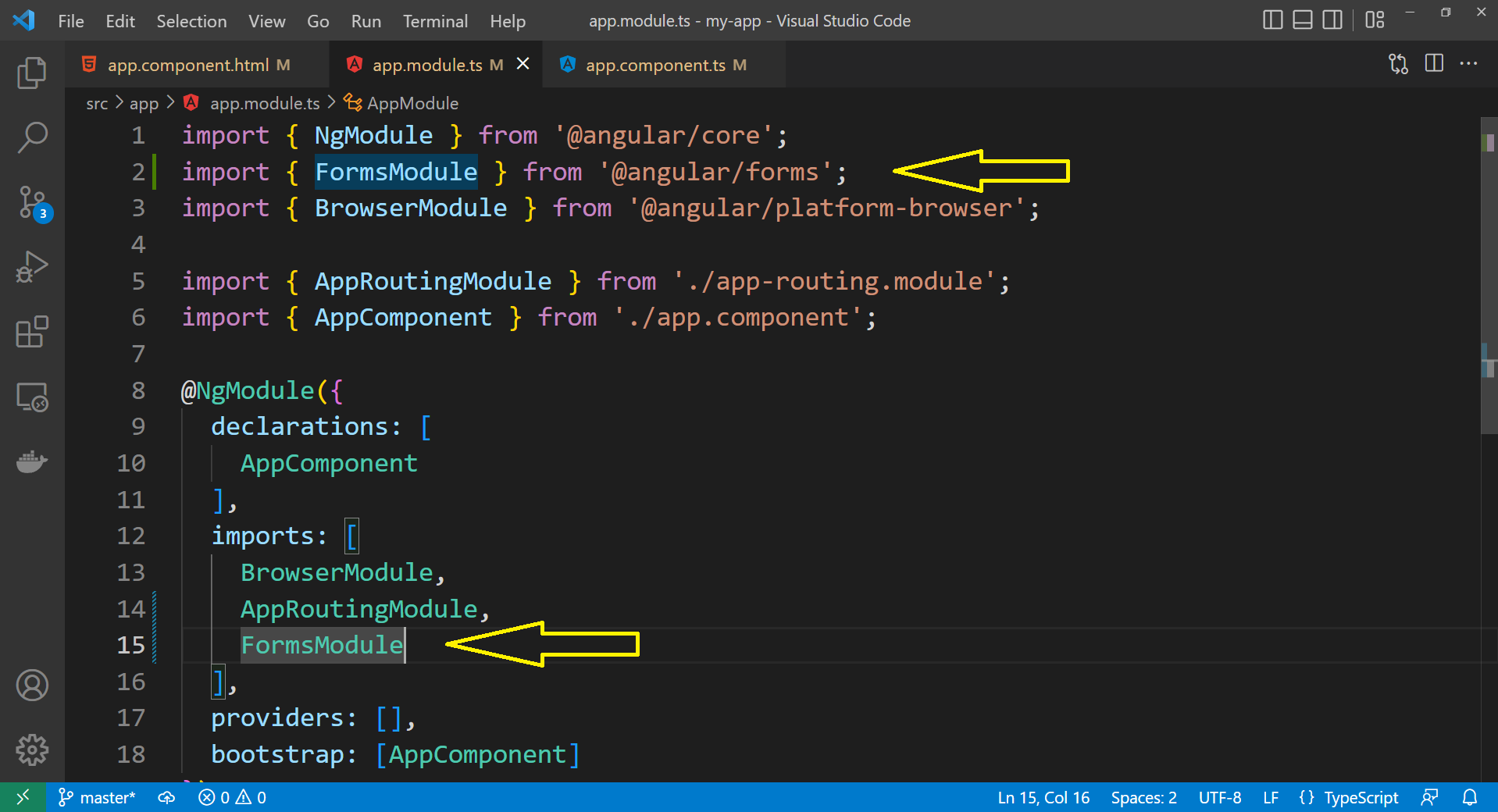
1. Avoir comment faire la Liaison de données et les évènements.
2. L'utilisation des directives : ngIf, ngFor, ngClass …
3. Pratiquer le principe des Pipes

On va développer un Todo List simple pour gérer des taches

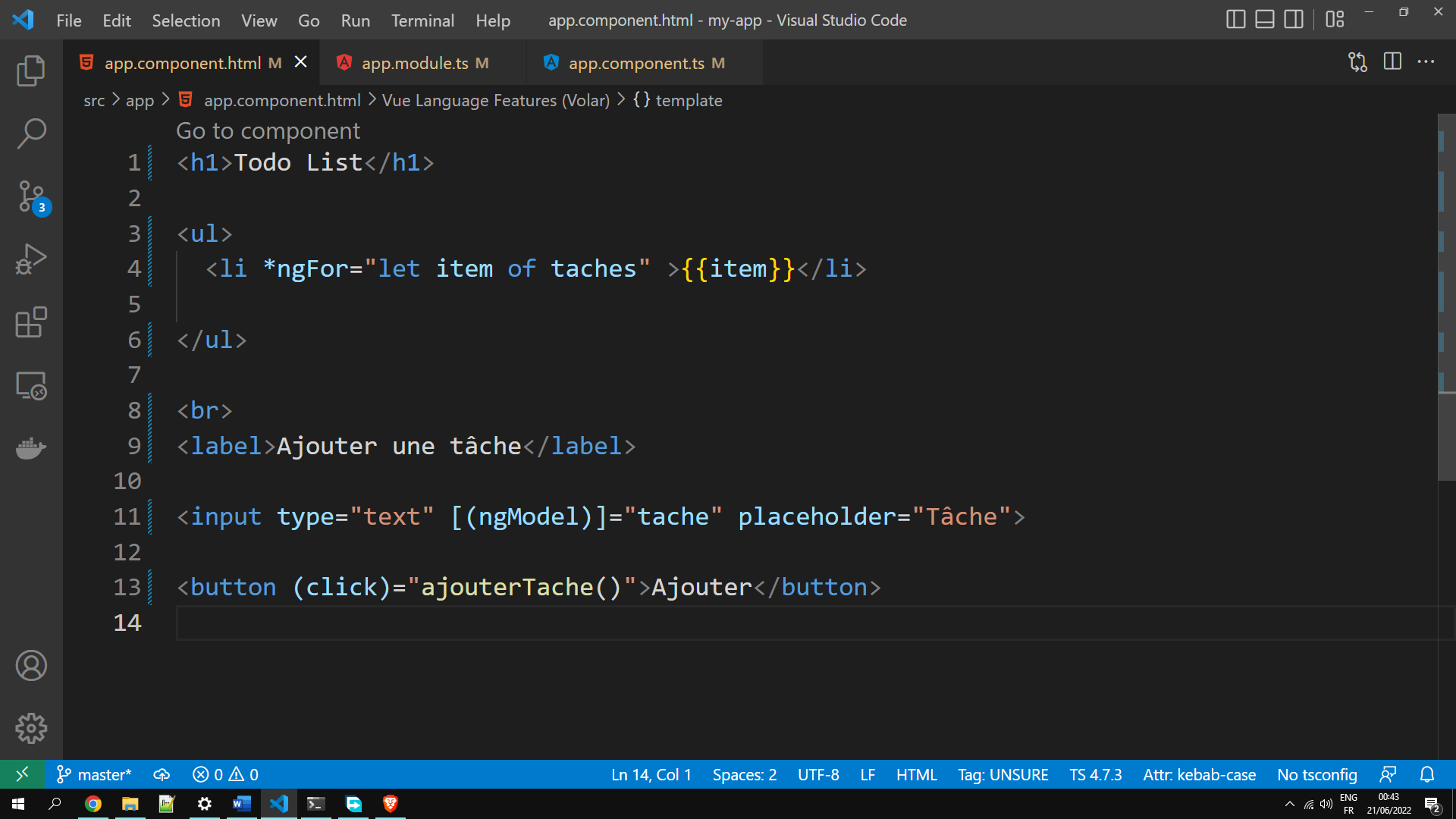
1- Dans un premier temps on va utiliser une liste de type String pour stocker les tâches, alors tout d’abord, dans « src/app/app.component.ts » déclarer une liste « tâches » publique et un string « tache » pour recevoir la tâche et implémenter une méthode « ajouterTache() » pour ajouter une tâche



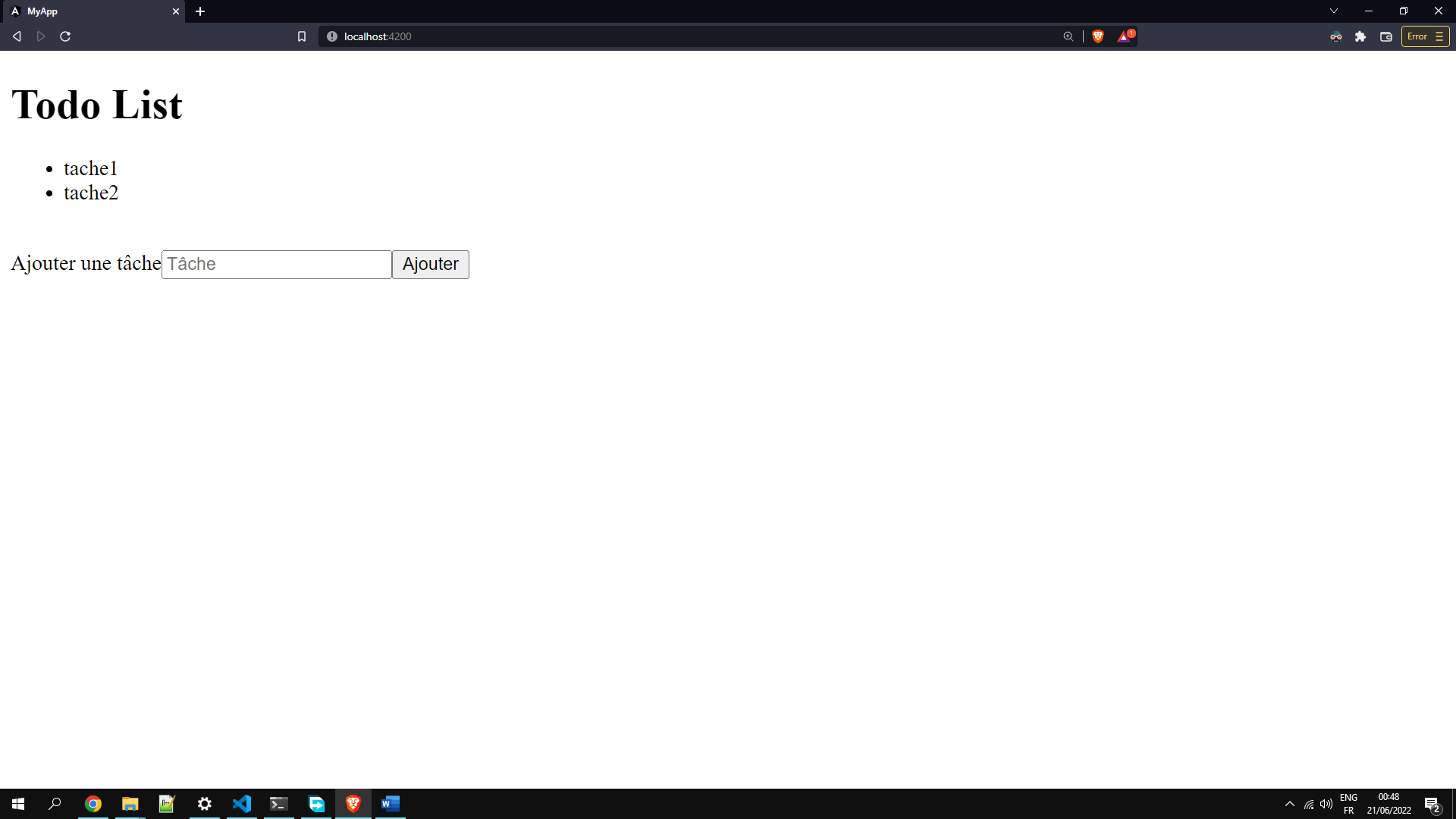
Pour lier la variable tache avec un champ input dans le html via le « ngModel », il faut importer « FormsModule » dans le « app.module »



En utilisant la directive « ngFor » dans le fichier « src/app/app.component.ts » afficher la liste « taches » dans une liste html et lier la variable « tache » avec un champ input de type « text », après ajouter un Bouton pour déclencher la méthode « ajouterTache() » via l’événement « (click) »



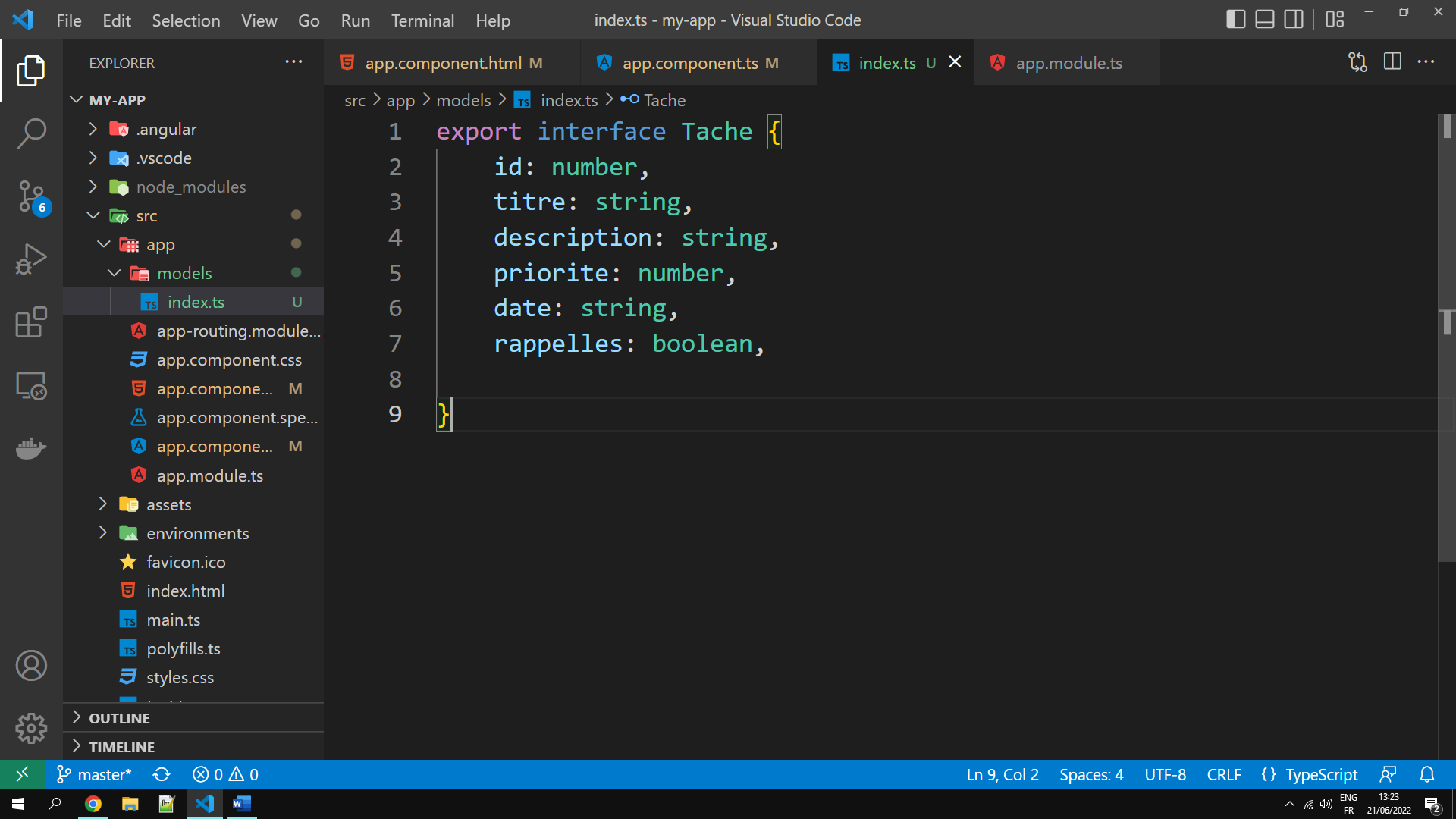
Testez maintenant l'application



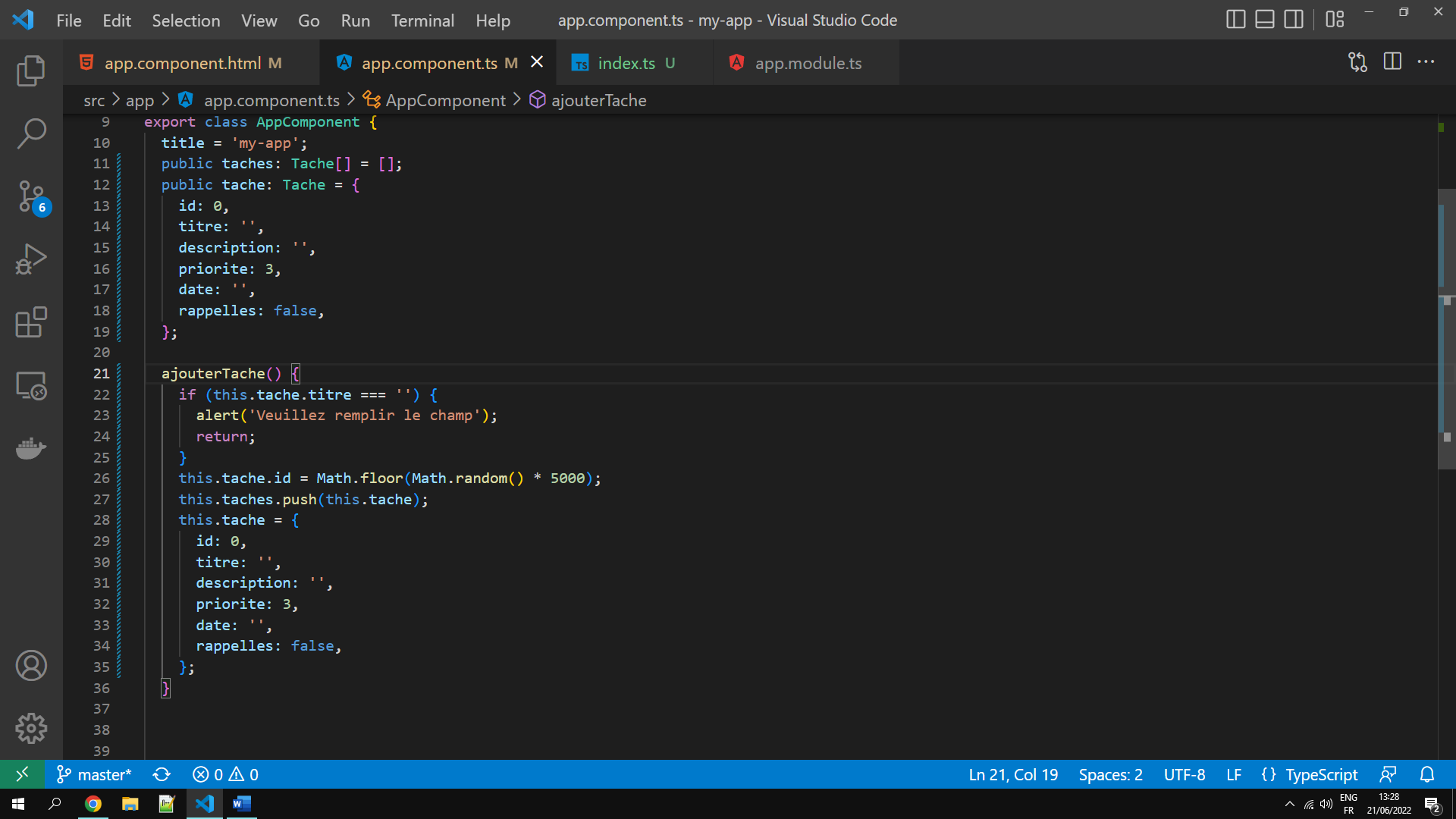
2- Pour bien pratiquer la liaison des données on va utiliser cette fois autres type d'input de HTML (Date, checkbox , textarea et select).

Créer un dossier « models » dans « src/app », et dans le dossier « models » créer un ficheri « index.ts ».

Dans le fichier « index.ts » définir une interface « Tache » pour présenter les tâches. Chaque tâche est définie par un Id, Titre, Description, Date, Priorité et Rappelle.

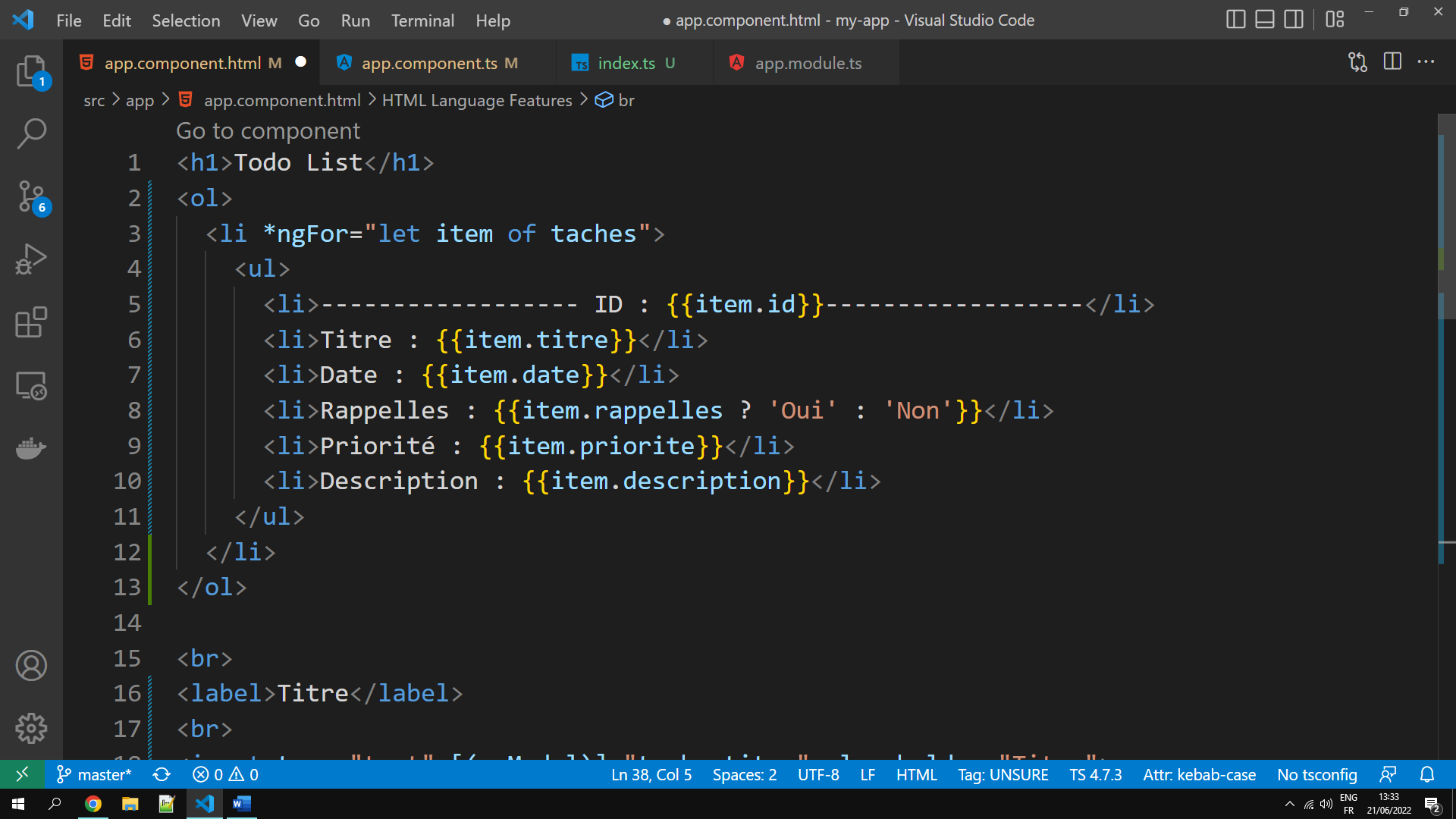


Dans le fichier « src/app/app.component.ts» changer le type de « tache » et la liste « taches » et réimplémenter la méthode « ajoutrTache() »

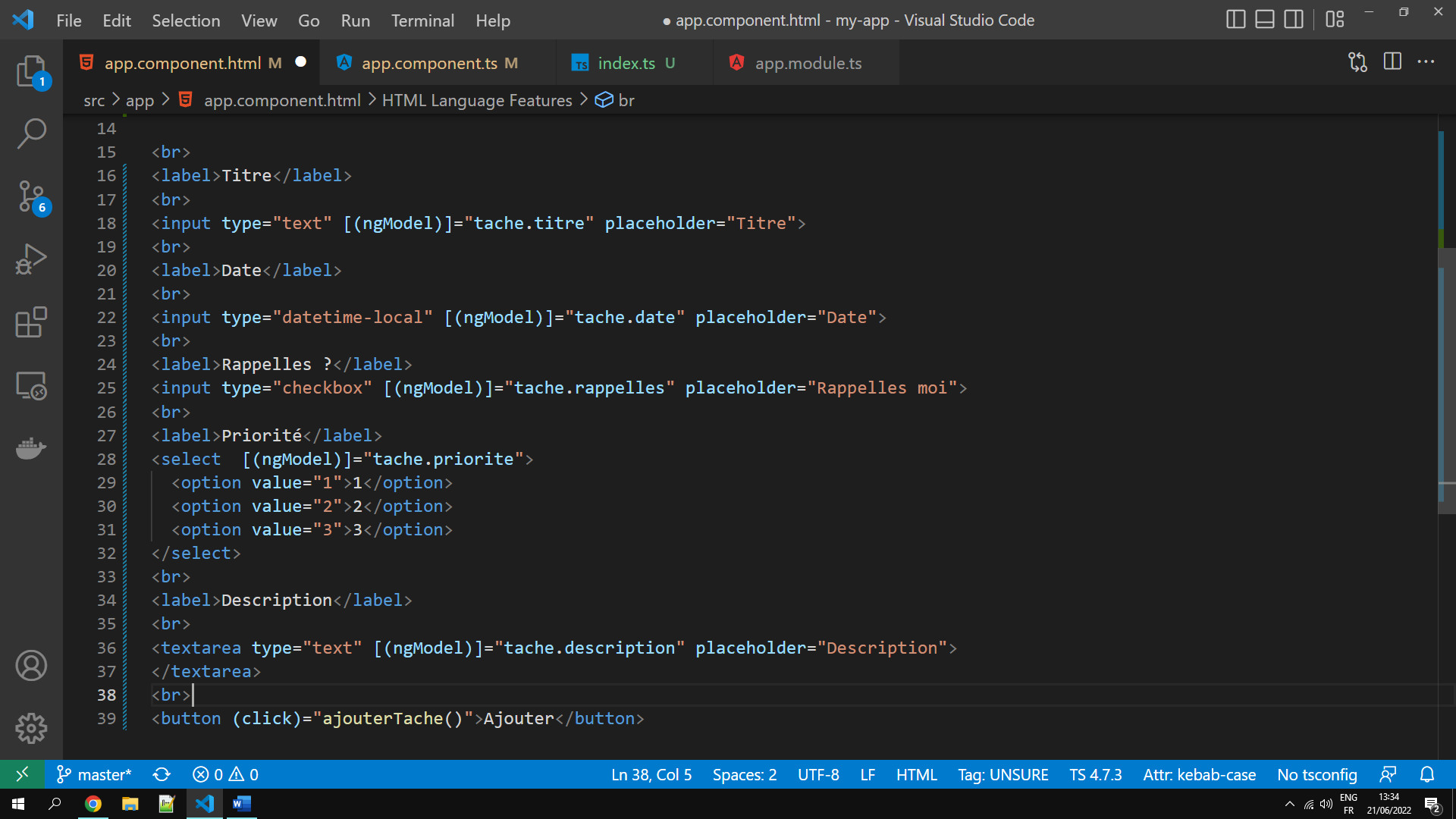


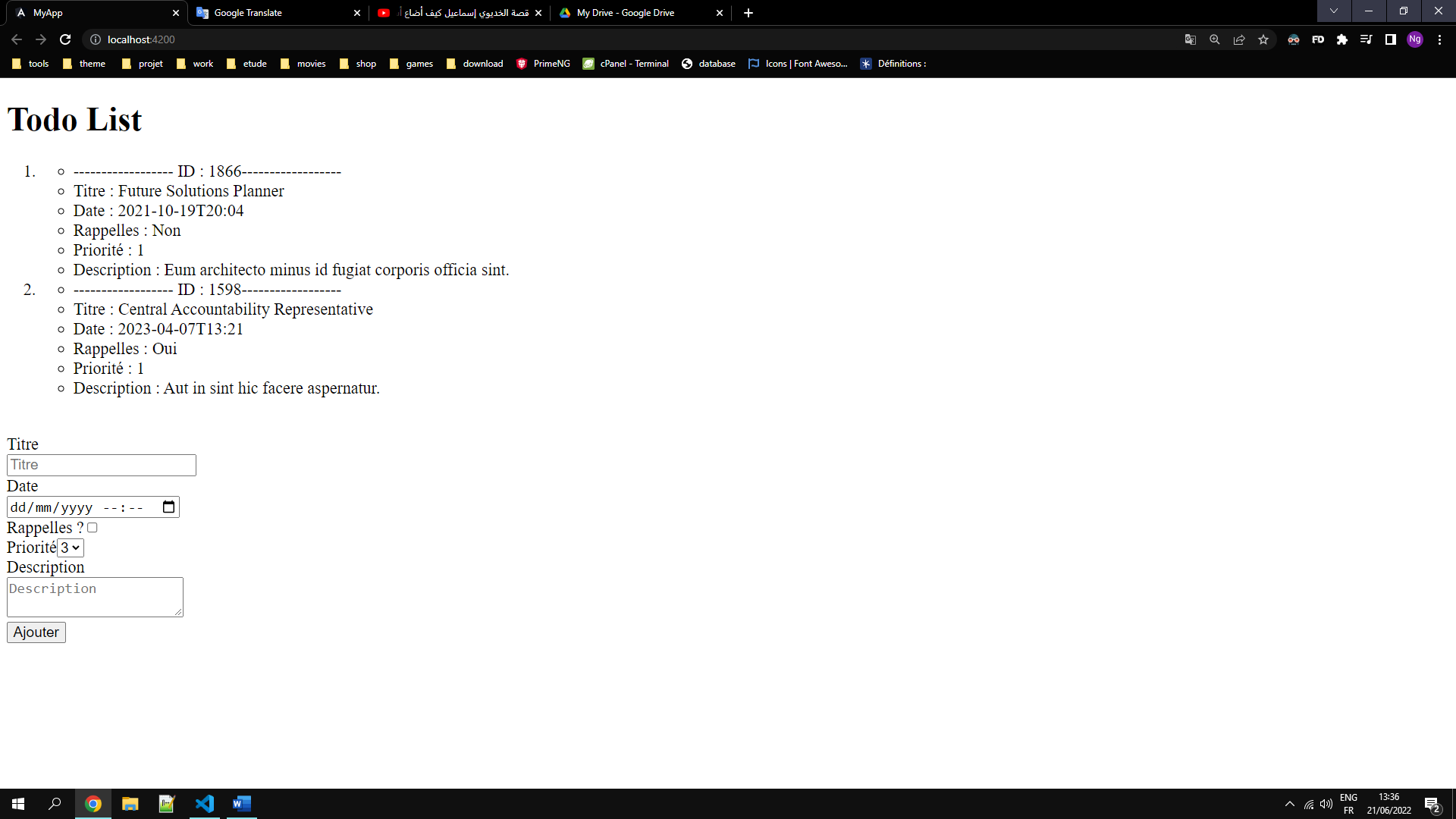
En utilisant le FormsModule « ngModel », lier les propretés de tache avec une champ input dans le html dans le fichier « src/app/app.component.html », et n’oubliez pas de changer l’affichage de la liste des tâches via le « ngFor »

Affichage :



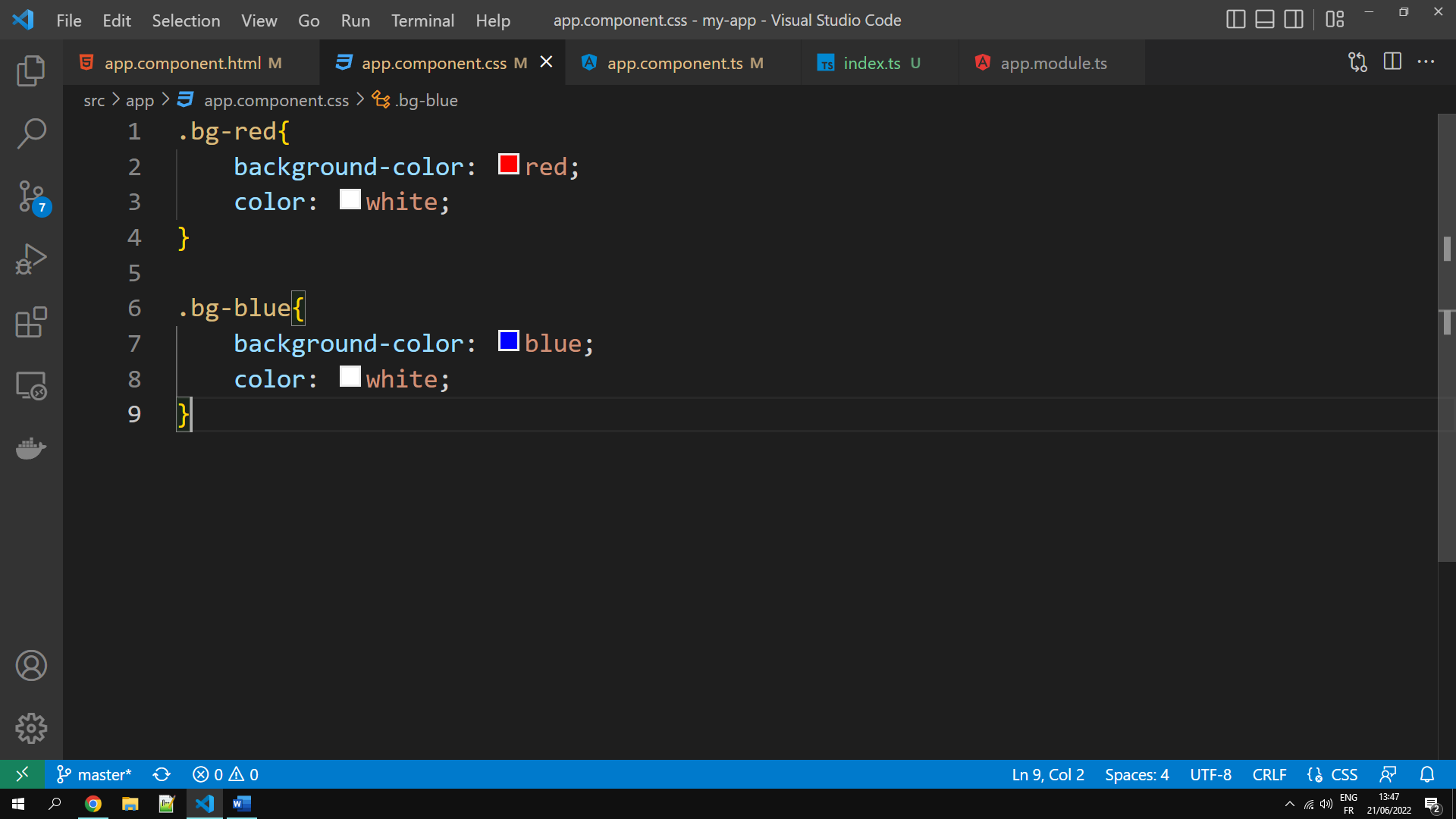
La liaison des données :



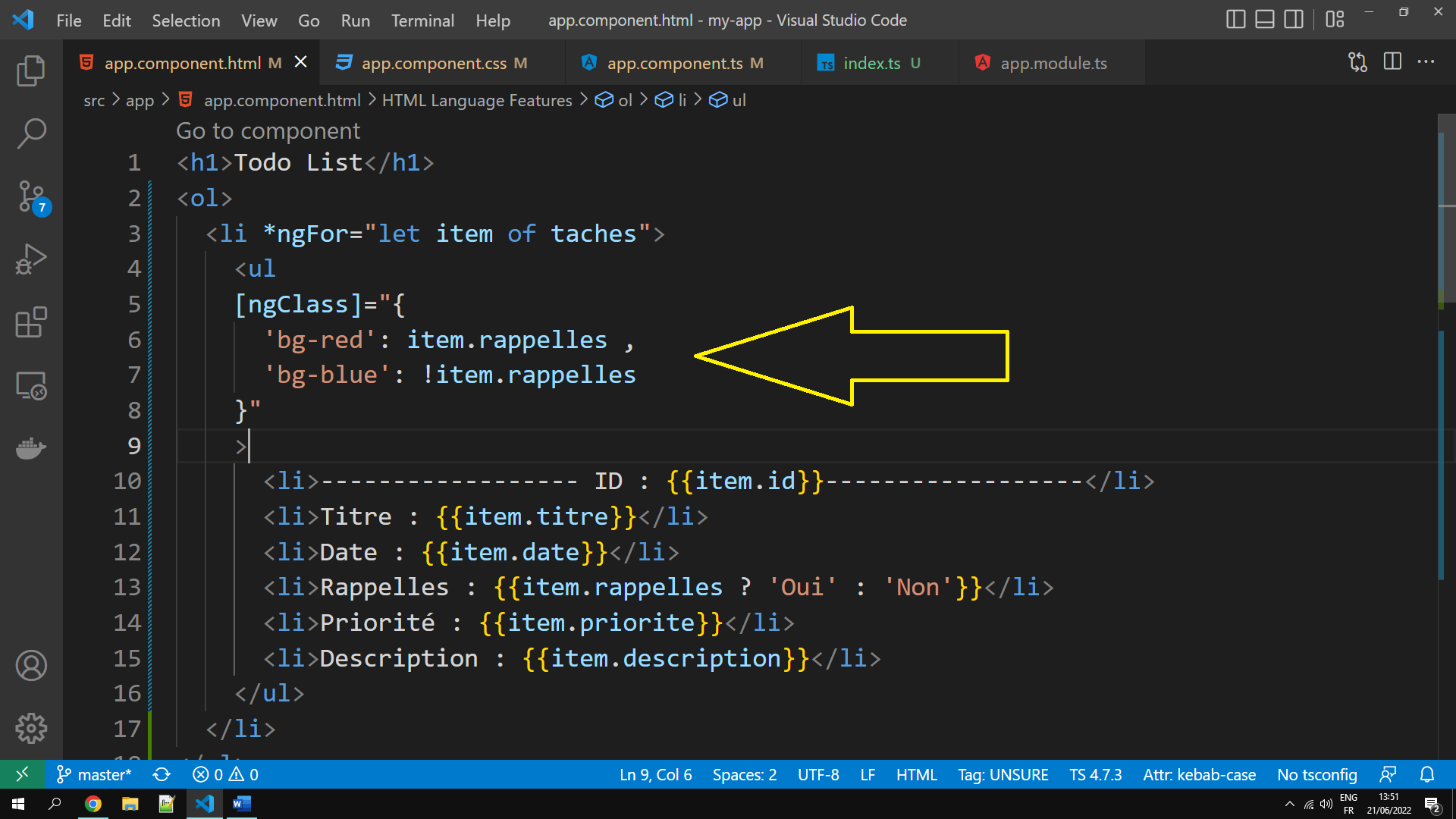
Tester 

Maintenant on va touche les styles CSS en Angular, supposant que on veut changer la couleur d’une tache selon sa rappelle, si elle est rappelée la couleur de la tache sera rouge sino bleu.

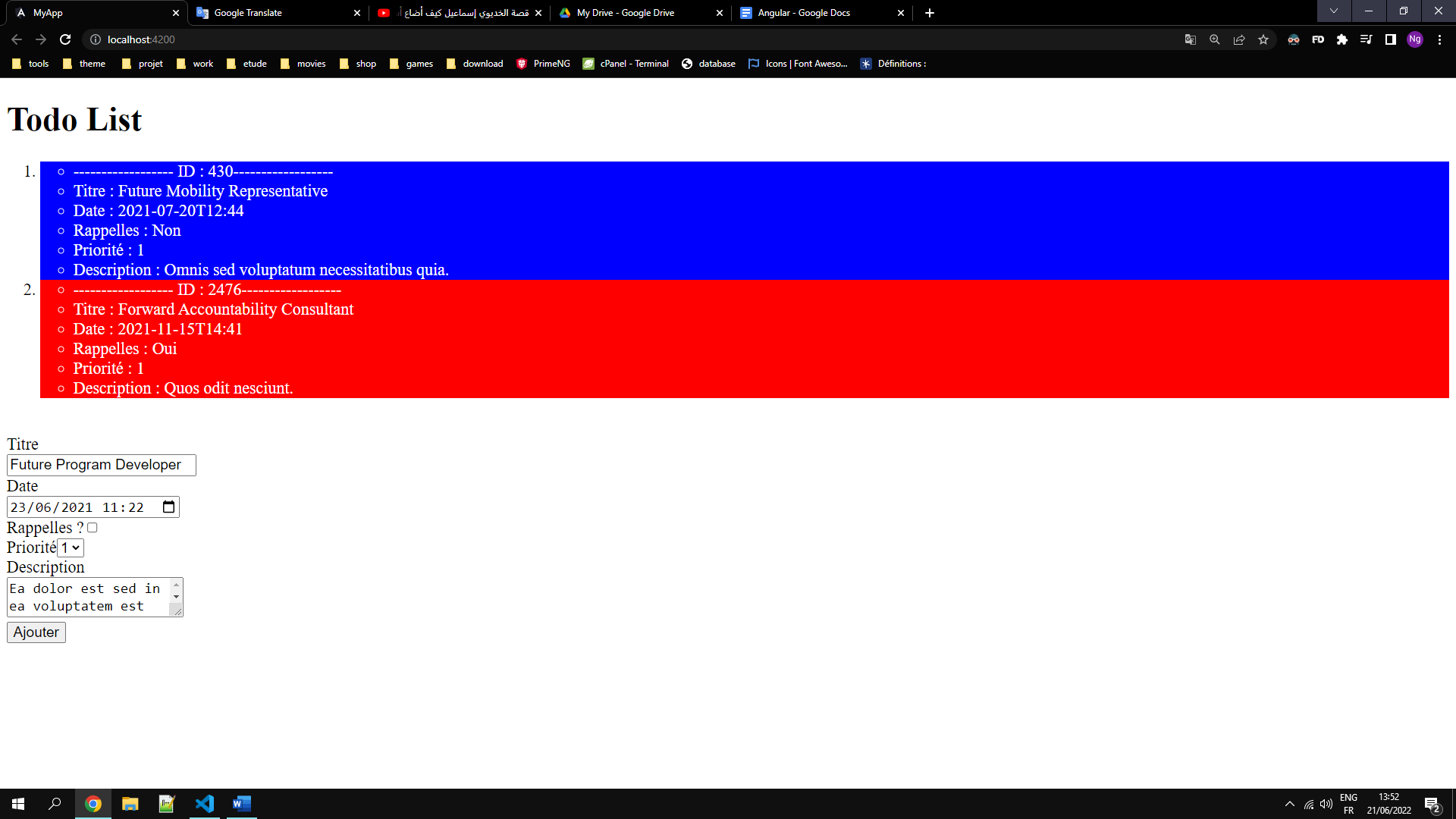
Dans le fichier « src/app/app.component.css » définir les deux classes CSS.



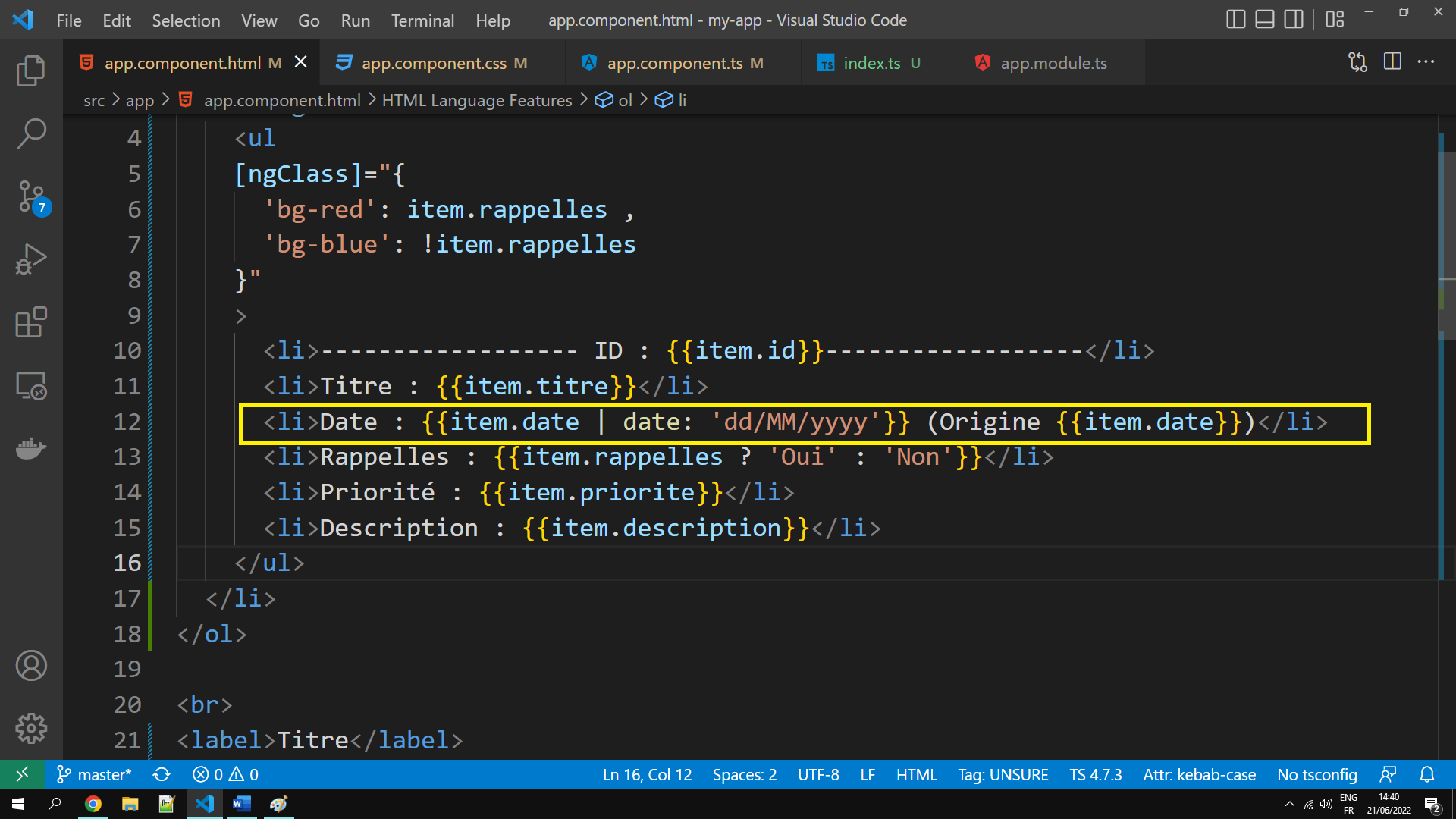
En utilisant la directive « ngCalss », lier la propreté « rappelles » de tâche par les deux classes CSS définis dans « src/app/app.component.css ».



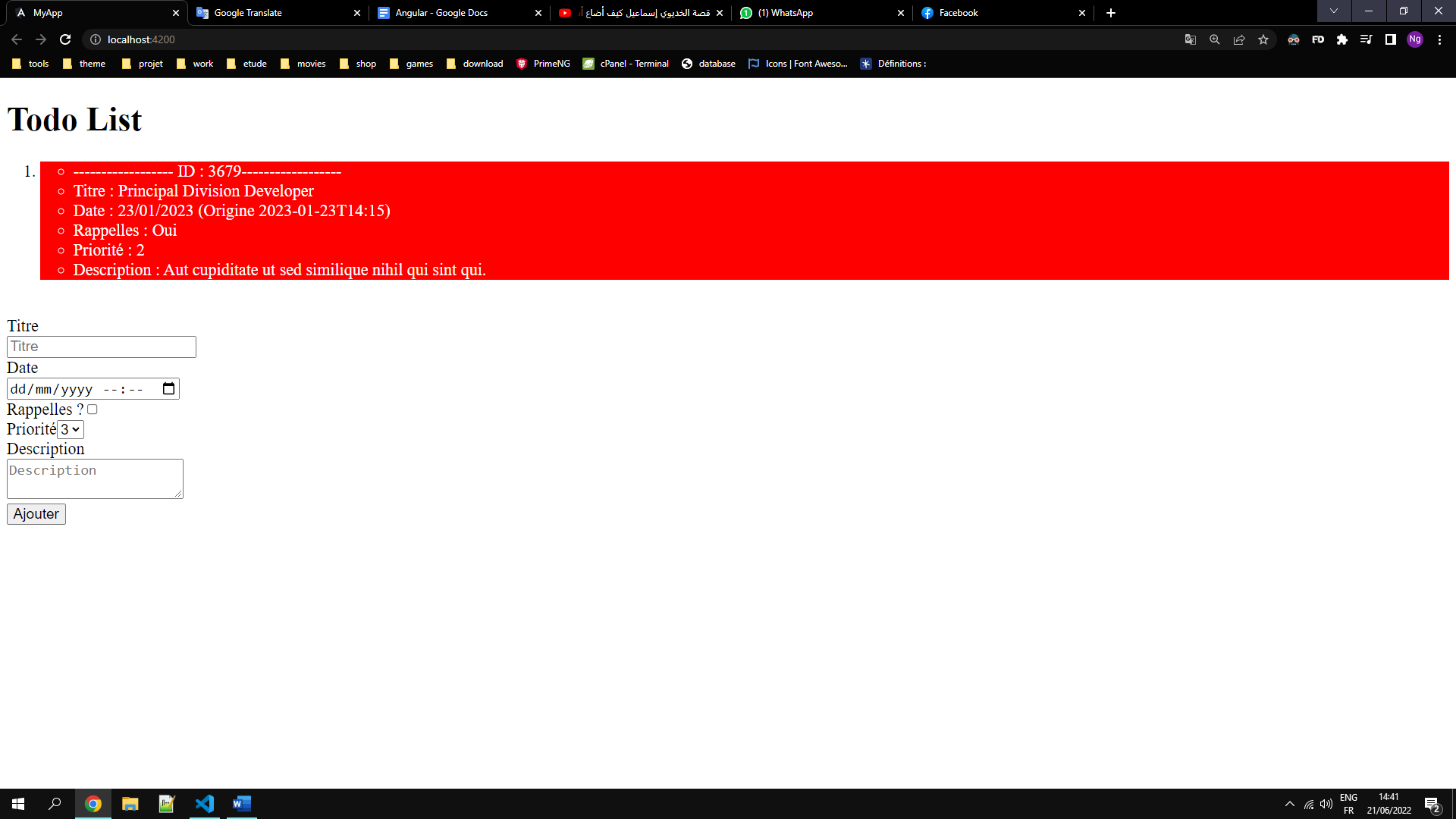
Tester :



En utilisant les pipes pour change le format de la date de tache de « 2023-01-23T14:15 » à « 23/01/2023 »



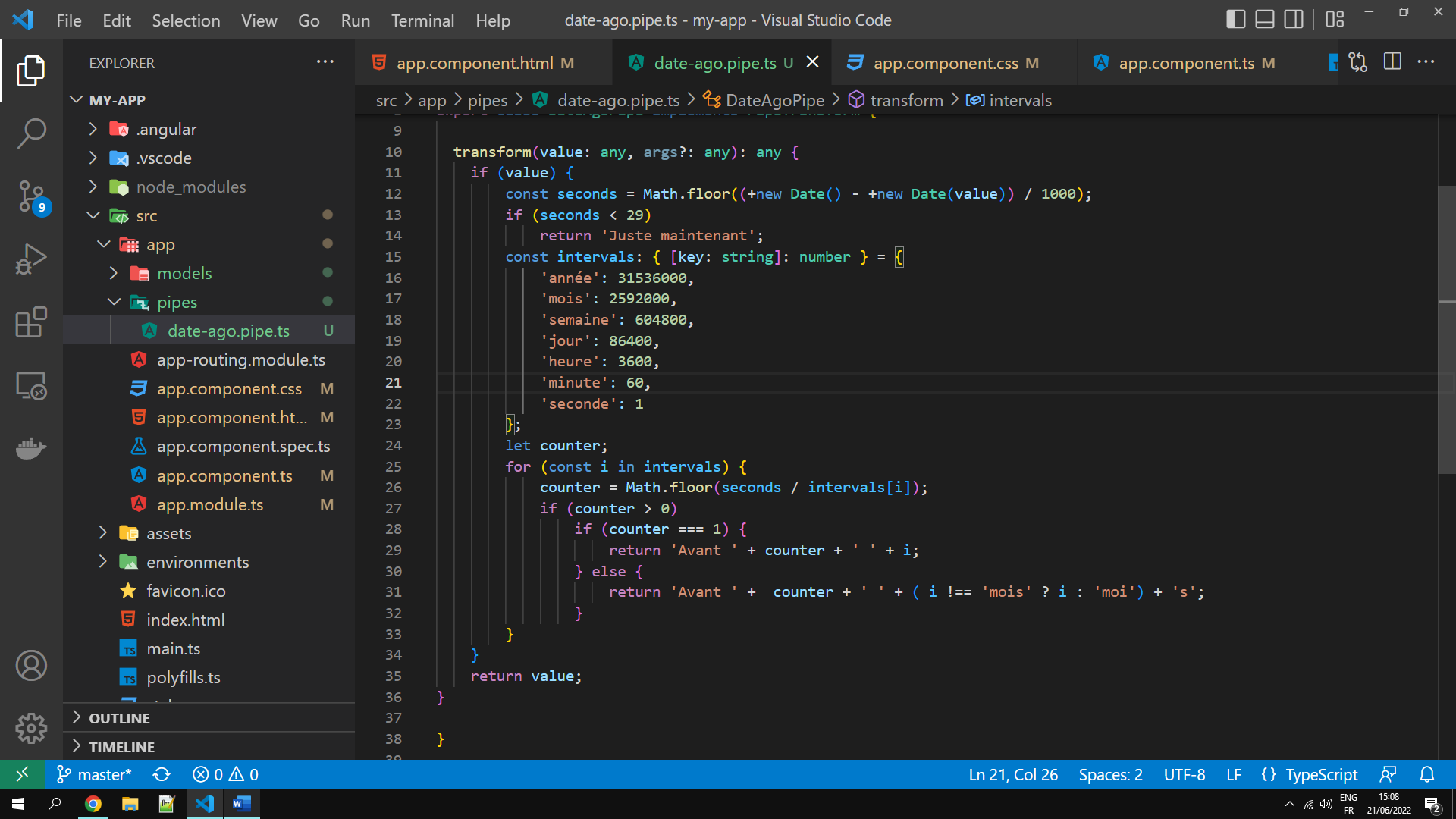
Tester :



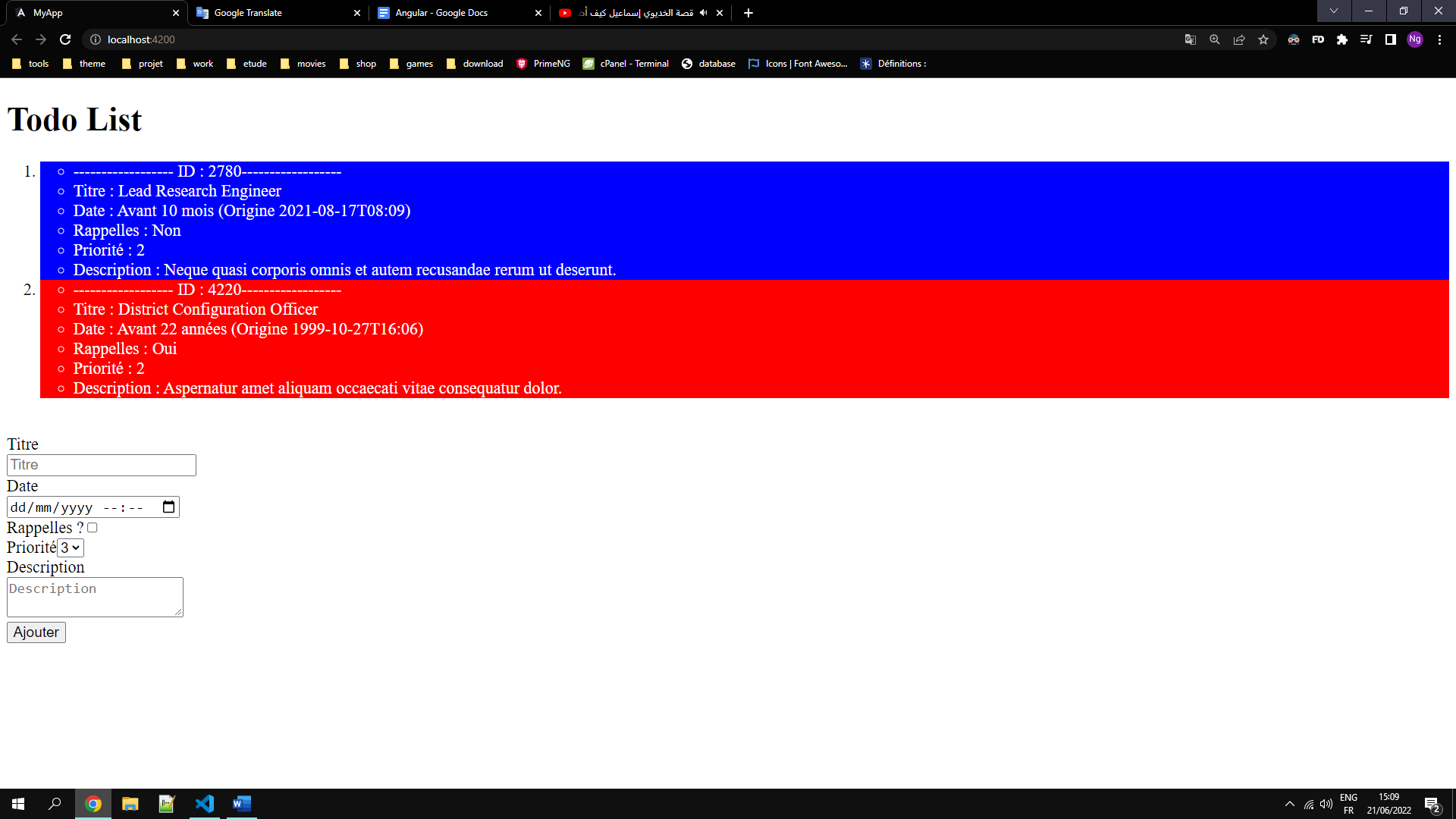
Maintenant on va avancer un peu, au lieu d'afficher la date « 2021-08-17T08:09 » on va le convertir pour afficher le nombre d'années mois jours ... dépassés « Avant 10 mois », et pour implémenter cette modification on doit utiliser les pipes personnalisés.

1- Créer une nouvelle pipe via la commande suivante : ng g p pipes/DateAgo

2- Implémenter la conversion dans la fonction « transform » dans la pipe « DateAgo »



Tester :



**Ateliers 3 : Les Forms et les Validations**

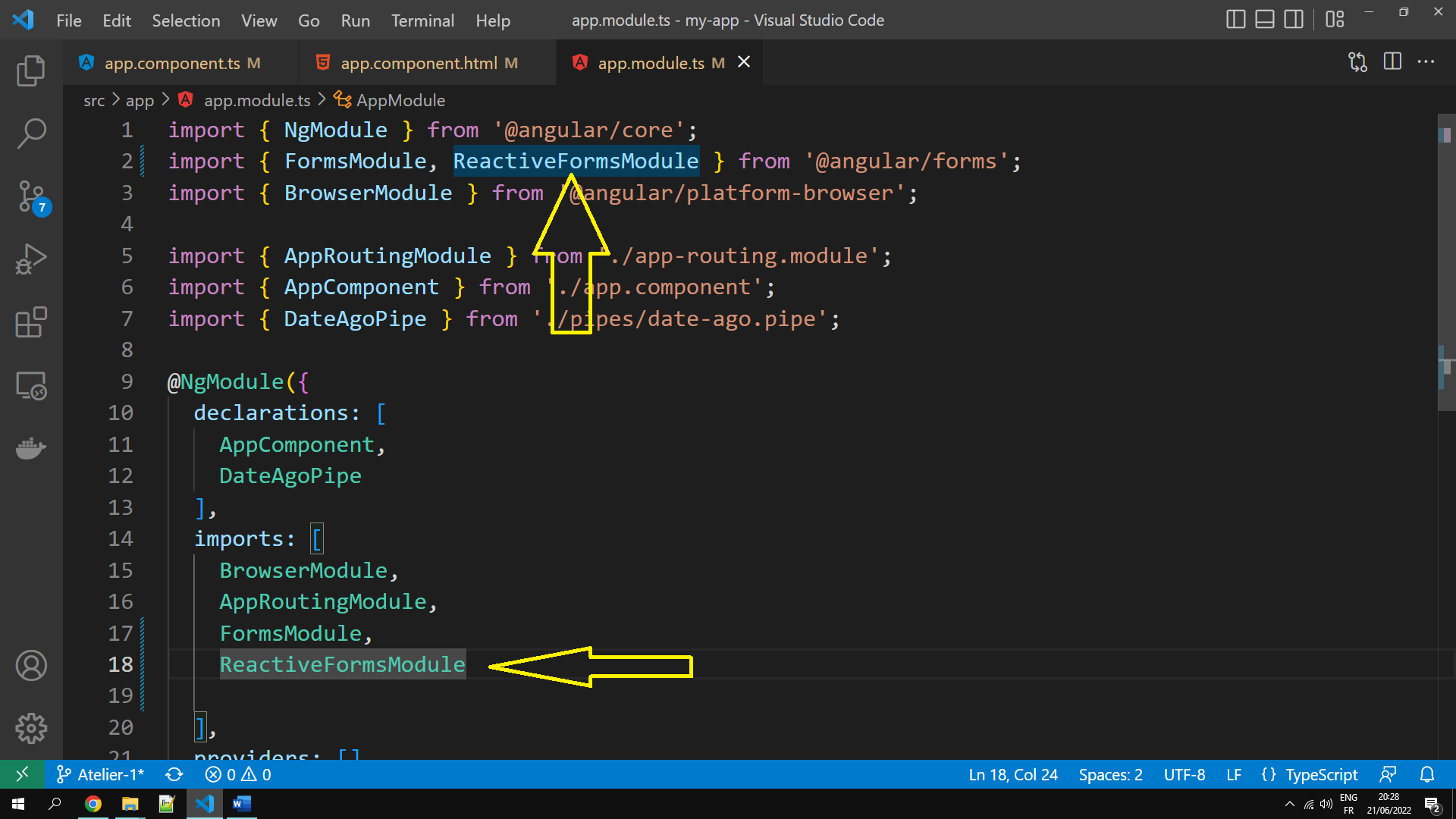
**Objectifs à atteindre :**

1. Bien pratiquer l'utilisation des Reactive Forms.
2. Valider les entrées d'utilisateur

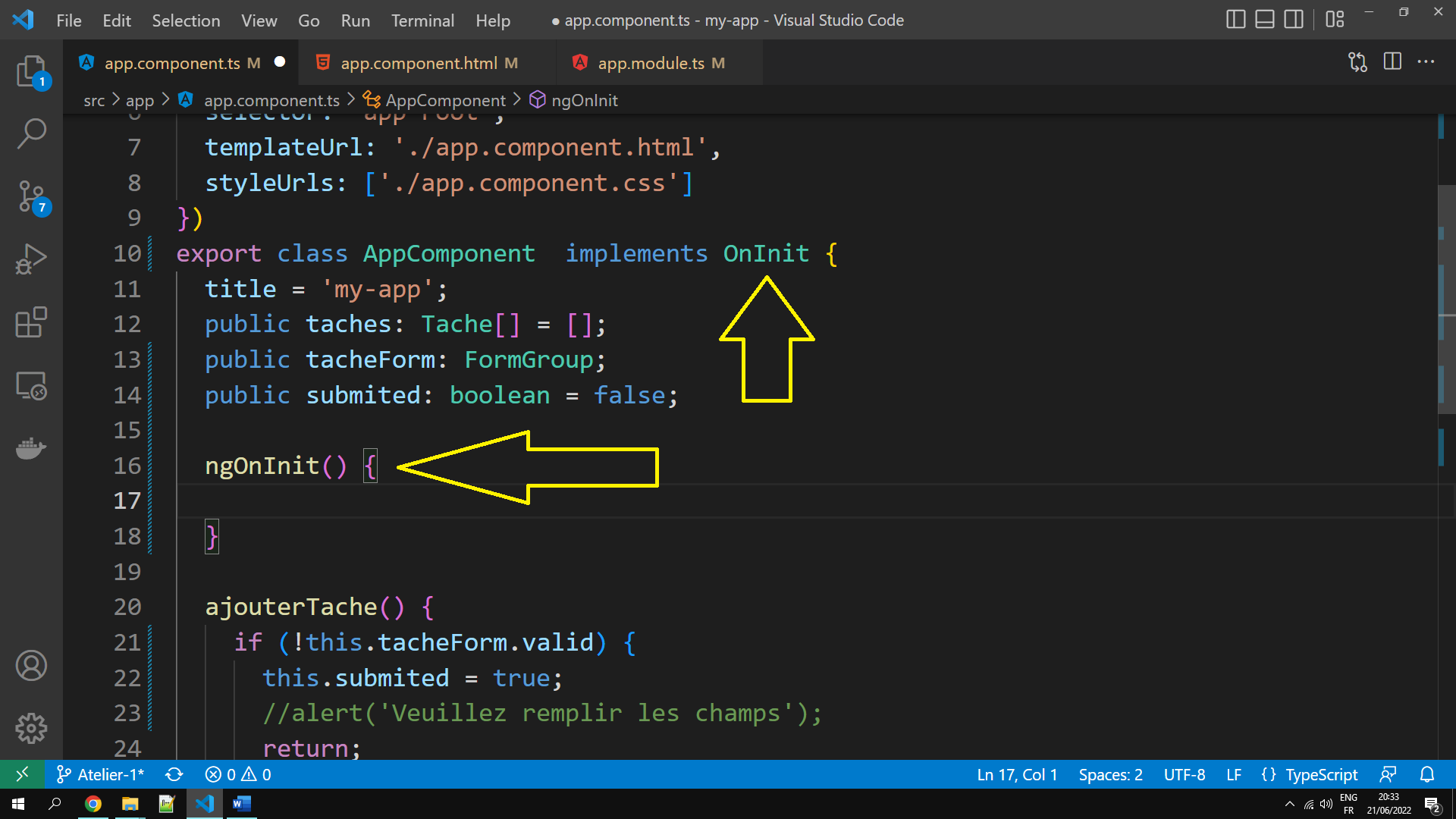
Pour remédier aux différentes limitations des Template-driven Forms ( ngModel ), Angular offre une approche originale et efficace nommée « Reactive Forms » présentant les avantages suivants :

* La logique des formulaires se fait dans le code TypeScript. Le formulaire devient alors plus facile à tester et à générer dynamiquement.
* Les « Reactive Forms » utilisent des Observables pour faciliter et encourager le "Reactive Programming".

Pour utiliser le « Reactive Forms ». il faut importer « ReactiveFormsModule» dans le « app.module »



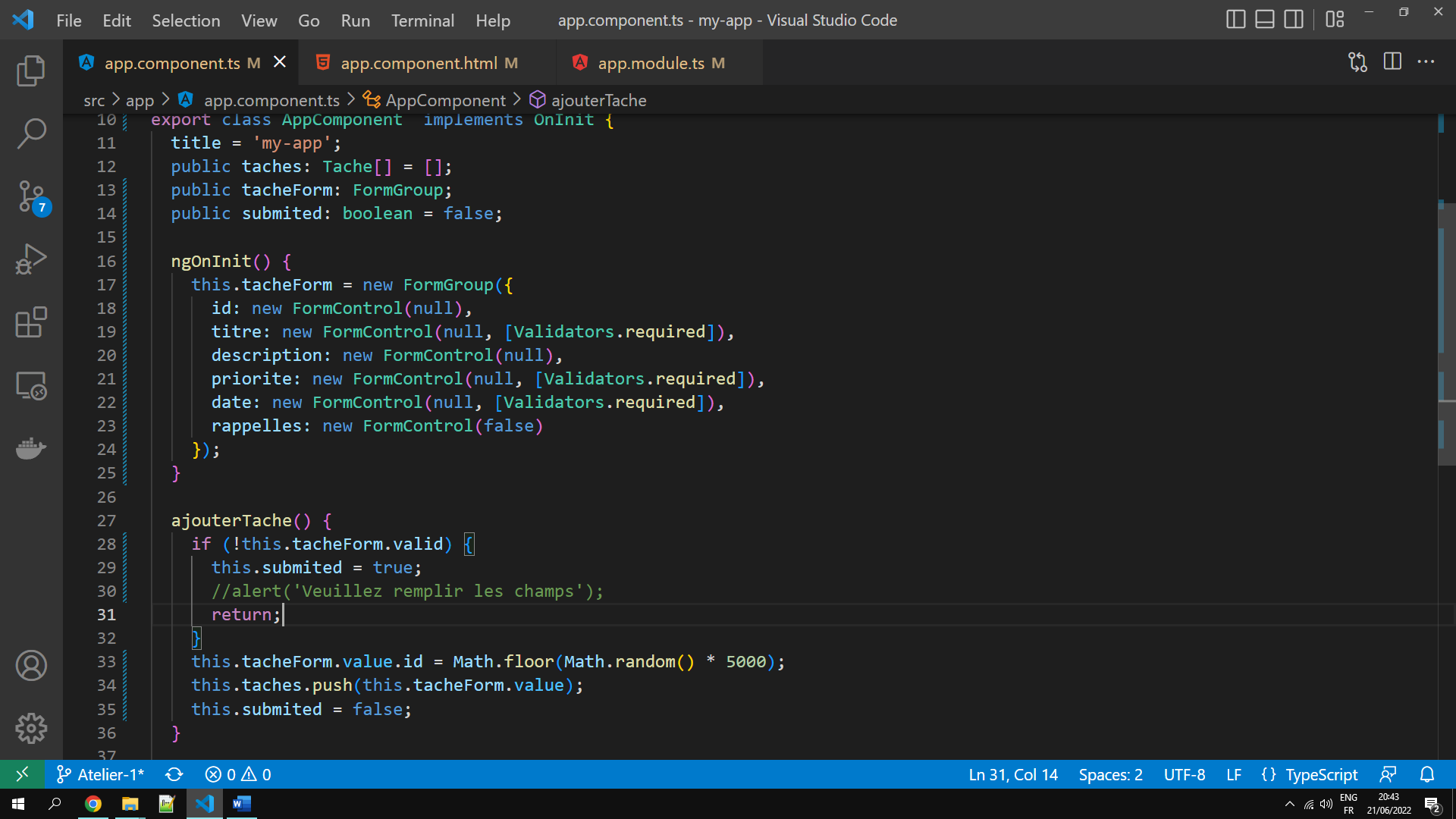
Dans le fichier « src/app/app.component.ts » assurer que la classe « AppComponent » est bien implémentée l’interface « OnInit » et sa méthode « ngOnInit() ».



En utilisant le « FormGroup » et « FormControl » déclarer et instancier une variable « tacheForm », après réimplémenter la méthode « ajouterTache() ».

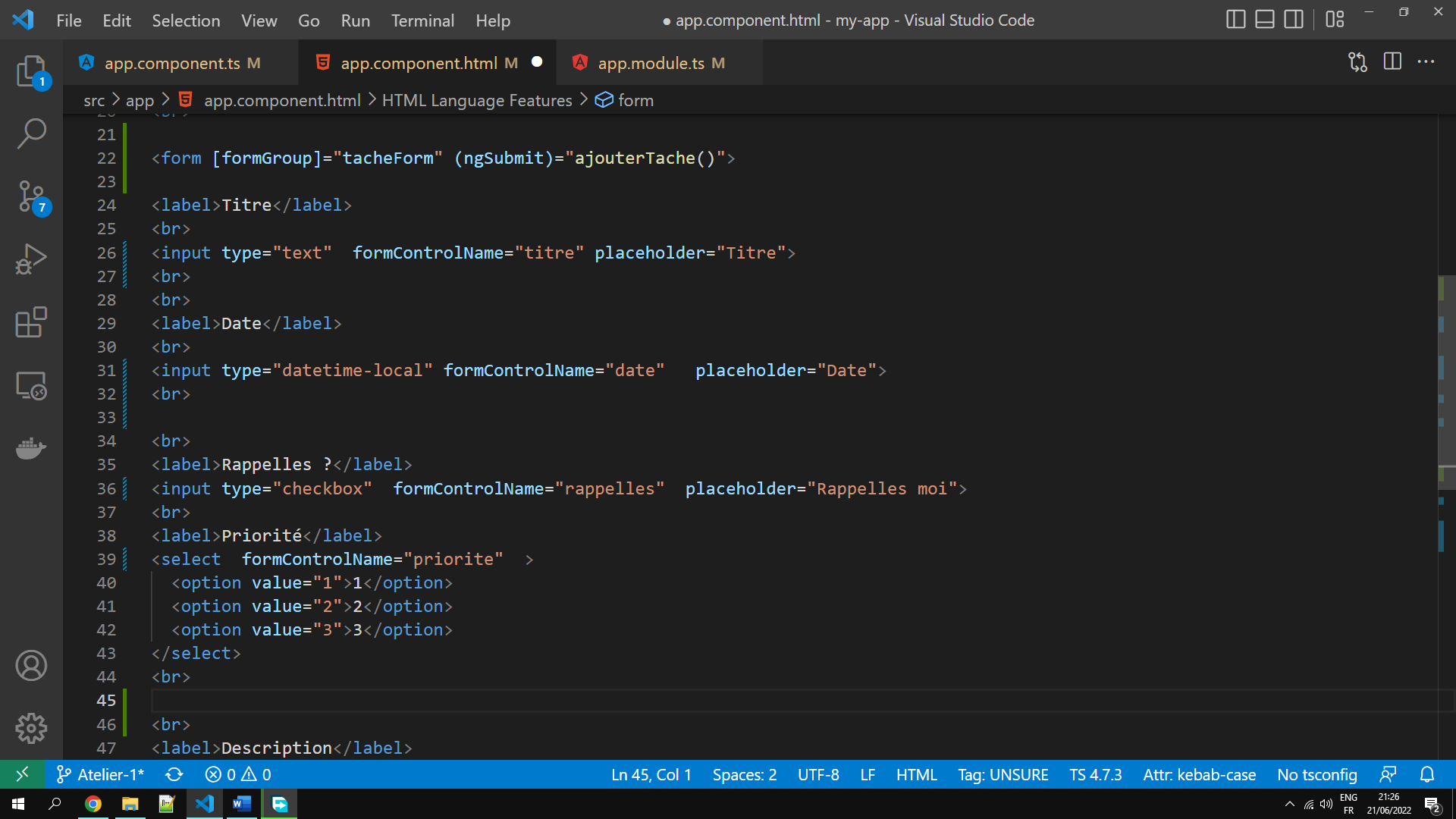
Le module « Validators » est utilisable pour la validation des inputs.

Note que les seules propriétés obligatoires sont le titre, la date et la priorité.

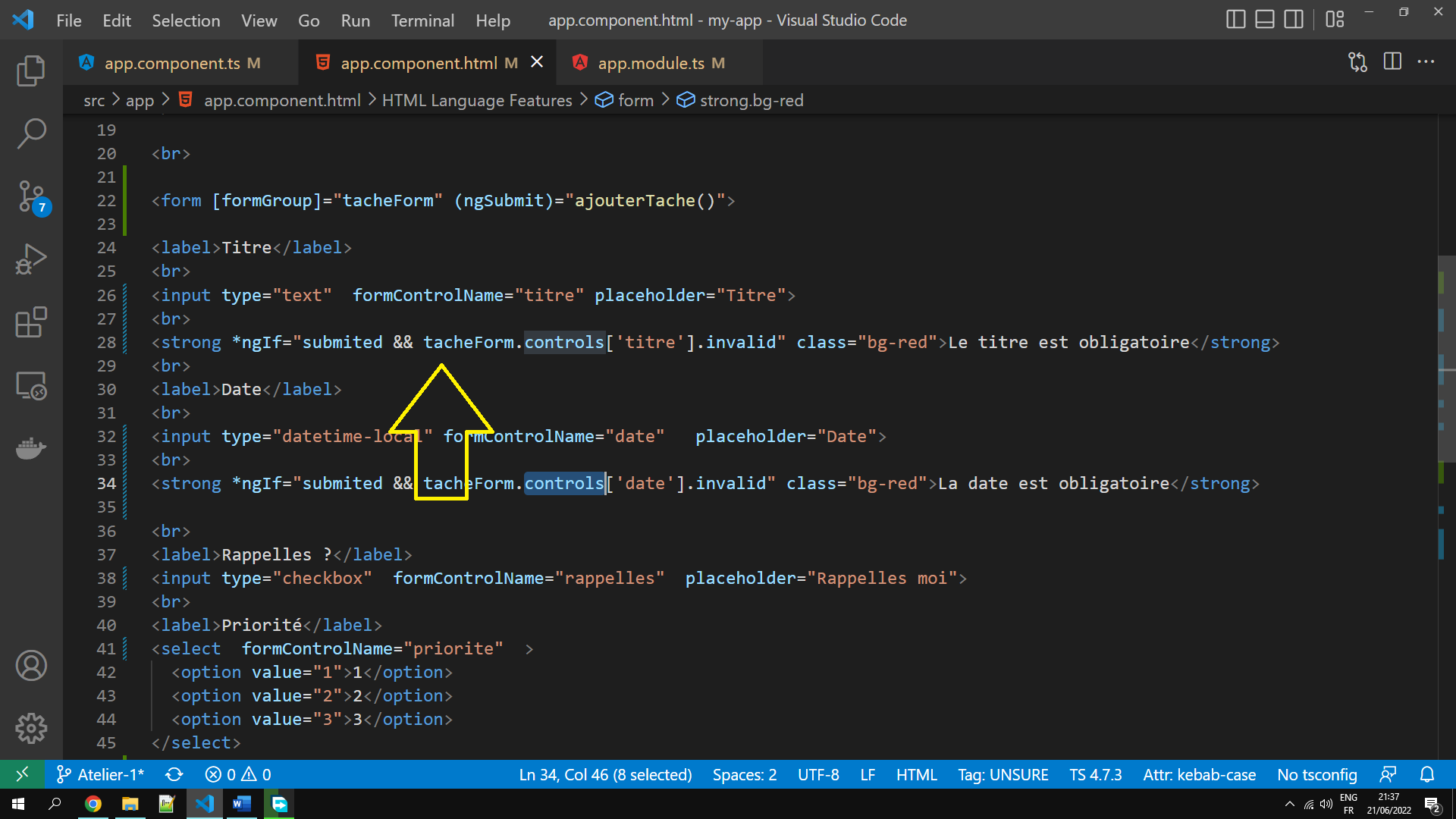


Dans le fichier « src/app/app.component.html » ajouter une forme et lier le via la propriété « formGroup », et propriété « formControlName » pour indiquer la liaison entre les champs HTML et les « FormControl » dans l’objet « tacheForm ».

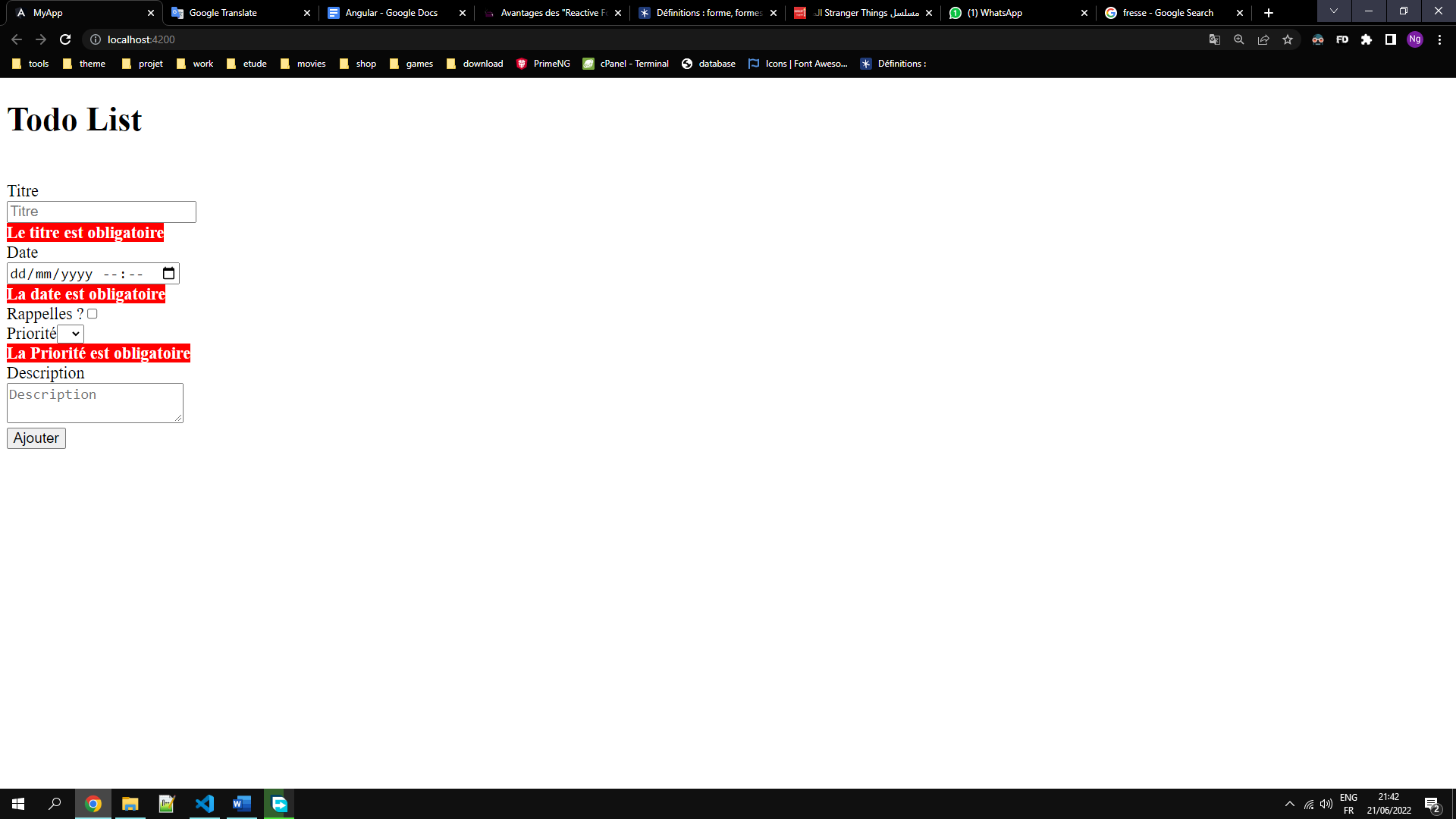
N’oubliez pas d’associer la forme « tacheForm » avec la méthode « ajouterTache() » via l’événement « ngSubmit », et supprimer l’évènement « click » du bouton « Ajouter »



Maintenant on va afficher des messages d’erreurs sous les champs invalidés lorsque la soumission de la forme, pour ça on va utiliser la dictionnaire « controls » d’objet « tacheForm ». et un booléen « submited » pour vérifier la soumission de la forme.



Tester :



**Ateliers 4 :** **Components**

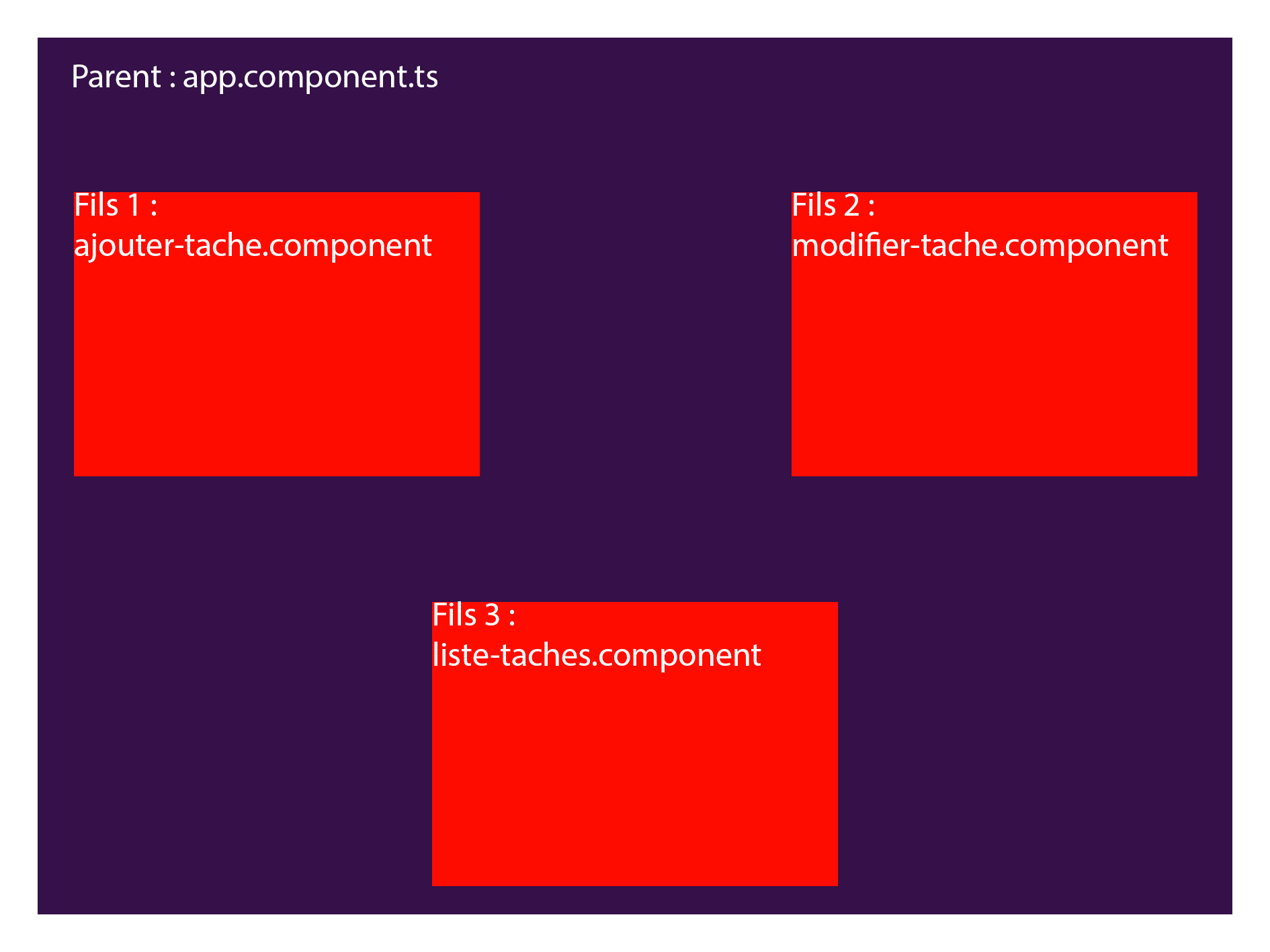
**Objectifs à atteindre :**

1. Pratiquer l’utilisation des Components Angular
2. Pratiquer la communication entre les des Components Angular

Jusqu'à maintenant, nous avons implémenté toutes les fonctionnalités de notre TodoList dans une seule composante, ça peut devenir plus compliqué lorsque le développement d'une grande application, pour éviter la complexité on va décomposer notre application en 3 composantes :

1. AjouterTache, pour ajouter une nouvelle tâche.
2. ModifierTache, pour modifier une tâche.
3. ListeTaches, pour consulter les tâches.

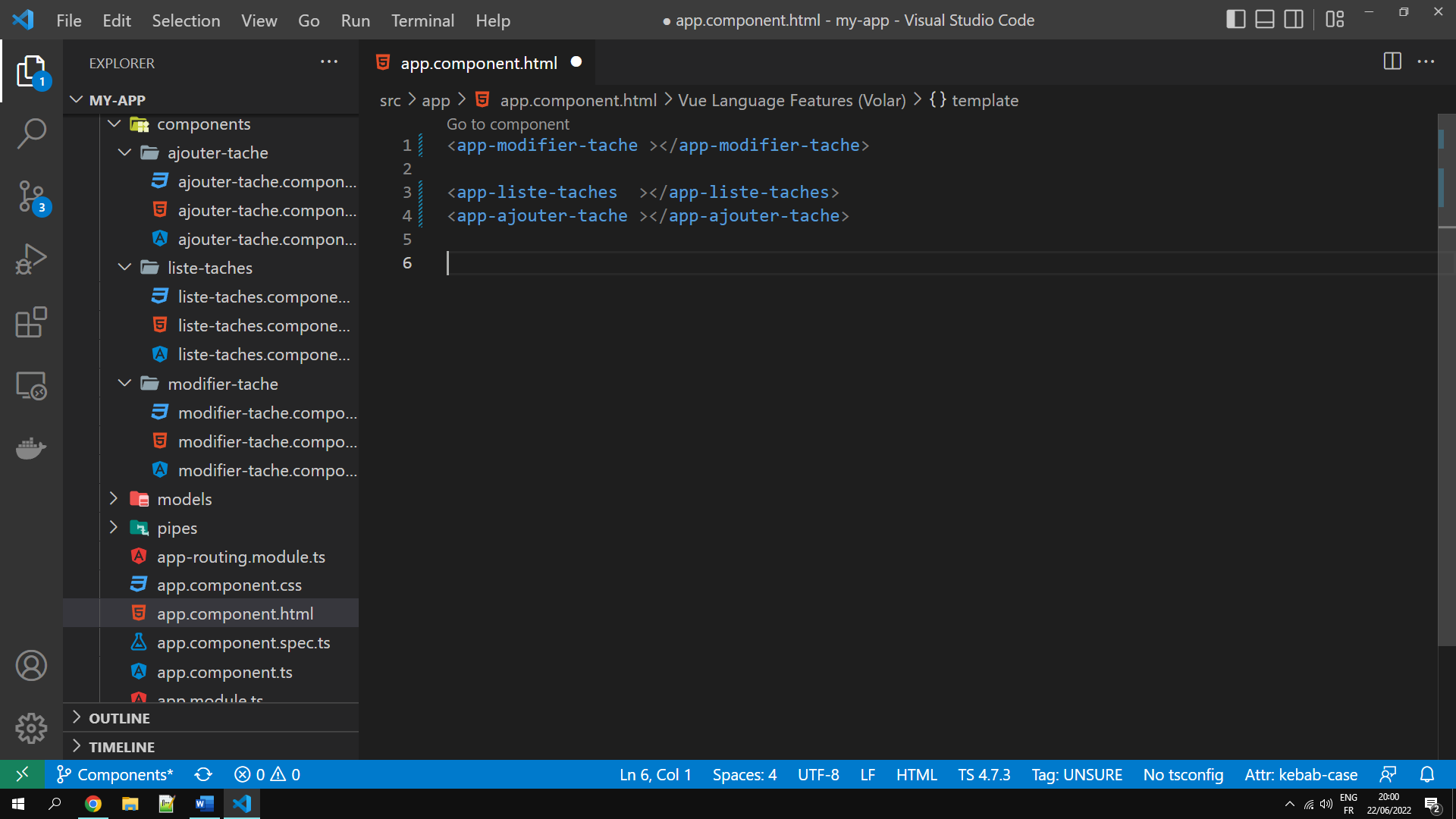
Tous les 3 composantes sont des fils de la composante père App.



- Lancer les 3 commandes pour créer ces composantes :

1. ng g c components/ajouterTache
2. ng g c components/modifierTache
3. ng g c components/listeTaches

- Importer les 3 composantes dans le fichier « app.component.html (on va utiliser la composante App pour gérer la communication entre les trois composantes fils)



- En utilisant les Décorateur @Input() et @Output(), déclarer une liste des taches dans « listeTaches » et une tache dans « ajouterTache » et « modifierTache »

- Déplacer les fonctionnalités de l’ajout et affichage des taches vers les nouvelles composantes « ajouterTache, listeTaches ».

- En utilisant la même forme d’ajout, implémenter la modification d’une tache dans la composante « modifierTache »

- Implémenter la suppression dans la composante « listeTaches »

Fichiers :

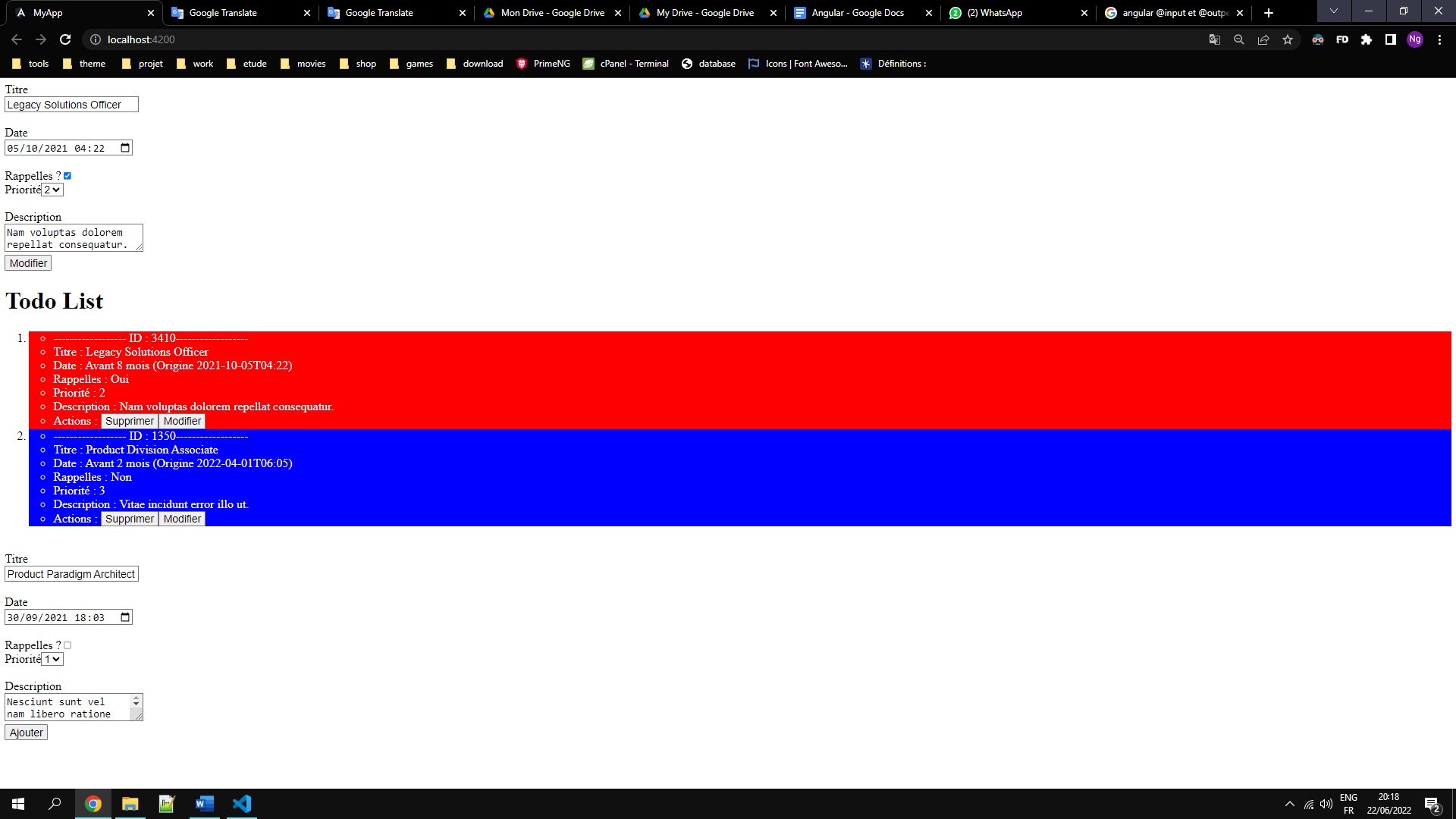








Tester:



**Ateliers 4 : Services et HTTP**

**Objectifs à atteindre :**

1. Pratiquer l’utilisation des service et l’injection de dépendance.
2. Pratiquer le REST API via HttpClient

Jusqu'à maintenant, nous avons implémenté toutes les fonctionnalités de notre TodoList dans une seule composante, ça peut devenir plus compliqué lorsque le développement d'une grande application, pour éviter la complexité on va décomposer notre application en 3 composantes :