# Application Mobiles

Diginamic & Christophe Germain

# Bref historique des mobiles

| | |
|---|---|
| ~1990 | Norme GSM 2G |
| 1992 | 1er smartphone : IBM Simon |
| 1999 | Nokia 3210, Blackberry OS (C++) |
| 2002 | BlackBerry 5810 |
| 2006 | Norme GSM 3G |
| 2007 | iPhone 1, lancement d'iOS (XNU) |
| 2007 | Lancement d'Android (Linux) |
| 2008 | App store et Android Market |
| 2009 | Smartphone Samsung Galaxy i7500 |
| 2010 | Norme GSM 4G, Windows Phone (C++/C#) |
| 2012 | Google Play : Android Market + Google Movies + Google Music |


# Applications mobiles (2/3)

## Applications hybrides :

- Mix entre applications web et applications natives :
  Code « web » + librairie accédant aux fonctionnalités de l'OS
  $\Longrightarrow$ appli native
  - **Apache Cordova** (2009) : code en JavaScript, HTML, CSS
  - **Xamarin** (2011) : code en C#
  - **React Native** (2015) : code en Javascript
    + code natif Java, Objective-C ou Swift
  - **Ionic Capacitor** (2018) : code en JavaScript, HTML, CSS
- **Avantages :** Cross-plateforme, fonctionnalités de l'OS
- **Inconvénients :** 1 production par OS, installations

# Applications mobiles (3/3)

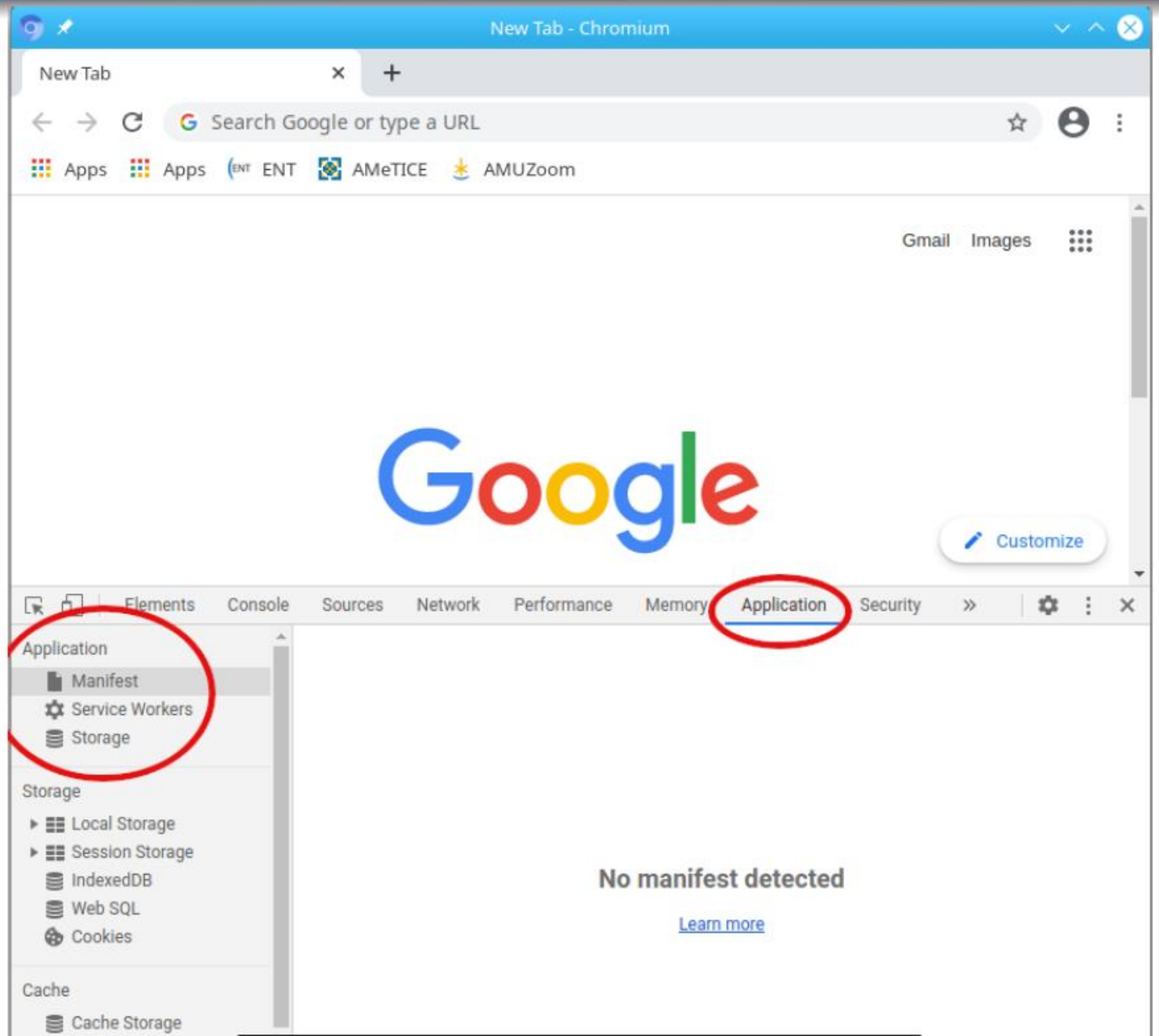
## Progressive Web Applications PWA :

- Amélioration des applications hybrides :
  - **Fiabilité :** peut fonctionner sans réseau (cache)
  - **Sécurité :** utilisation exclusive de réseau sécurisé (https)
  - **Installable :** comme une appli native mais via le web
  - **Progressif :** Updates automatiques via des notifications

---

- 3 caractéristiques pour être une PWA :
  - Utilisation de https
  - Avoir un manifeste (installations)
  - Utiliser un Service Worker (updates, cache, *etc.*)

# PWA et developer tool

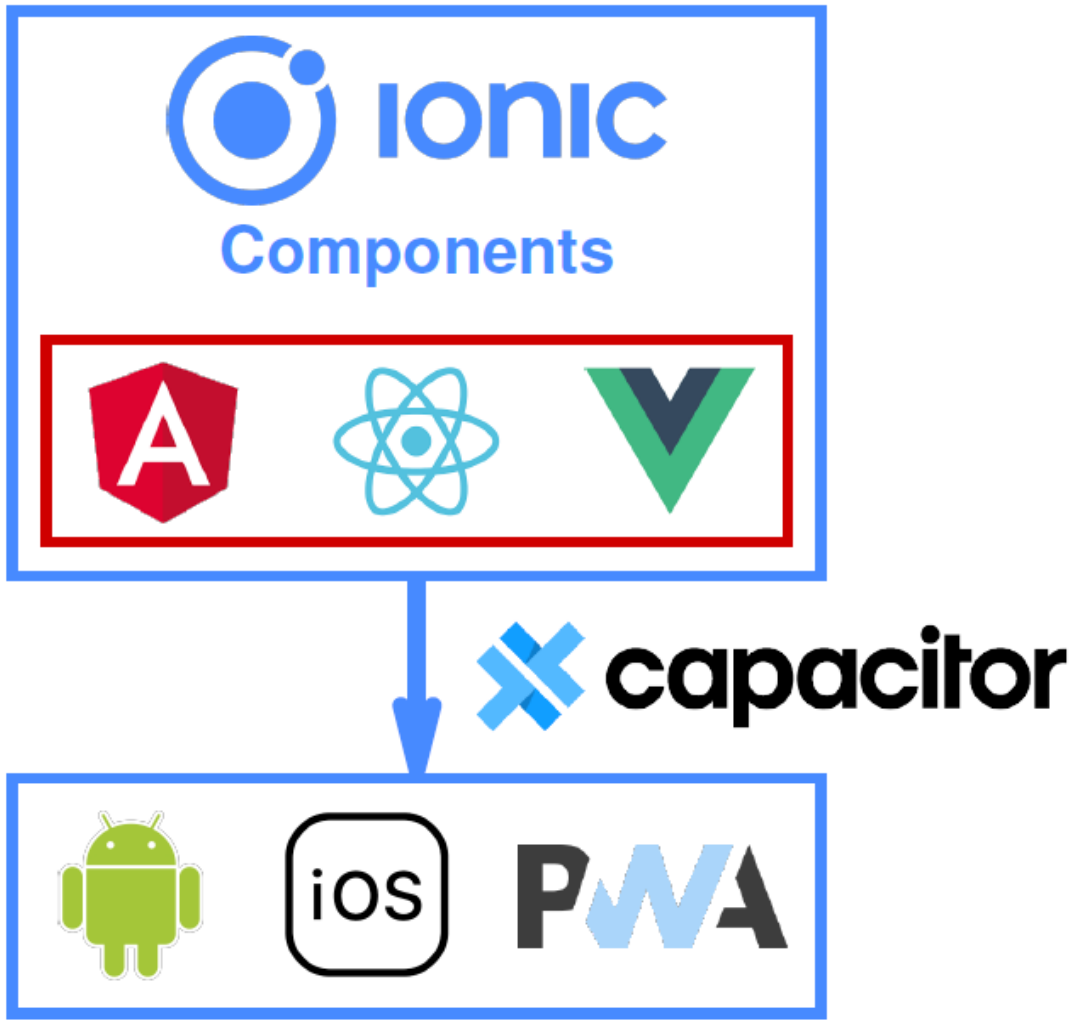


# Objectif de la partie « Applications mobiles »

**Objectif :** développement d'une application hybride

1. Utilisation de Ionic
2. Utilisation de Ionic Capacitor
3. TPs : portage du forum sur  et/ou iOS

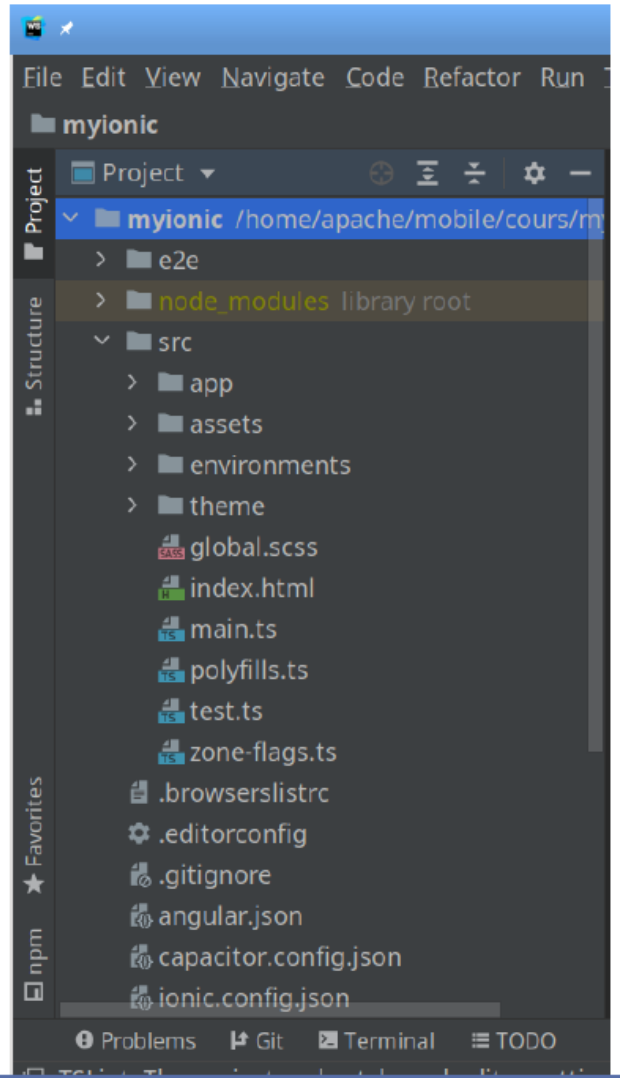
# Ionic



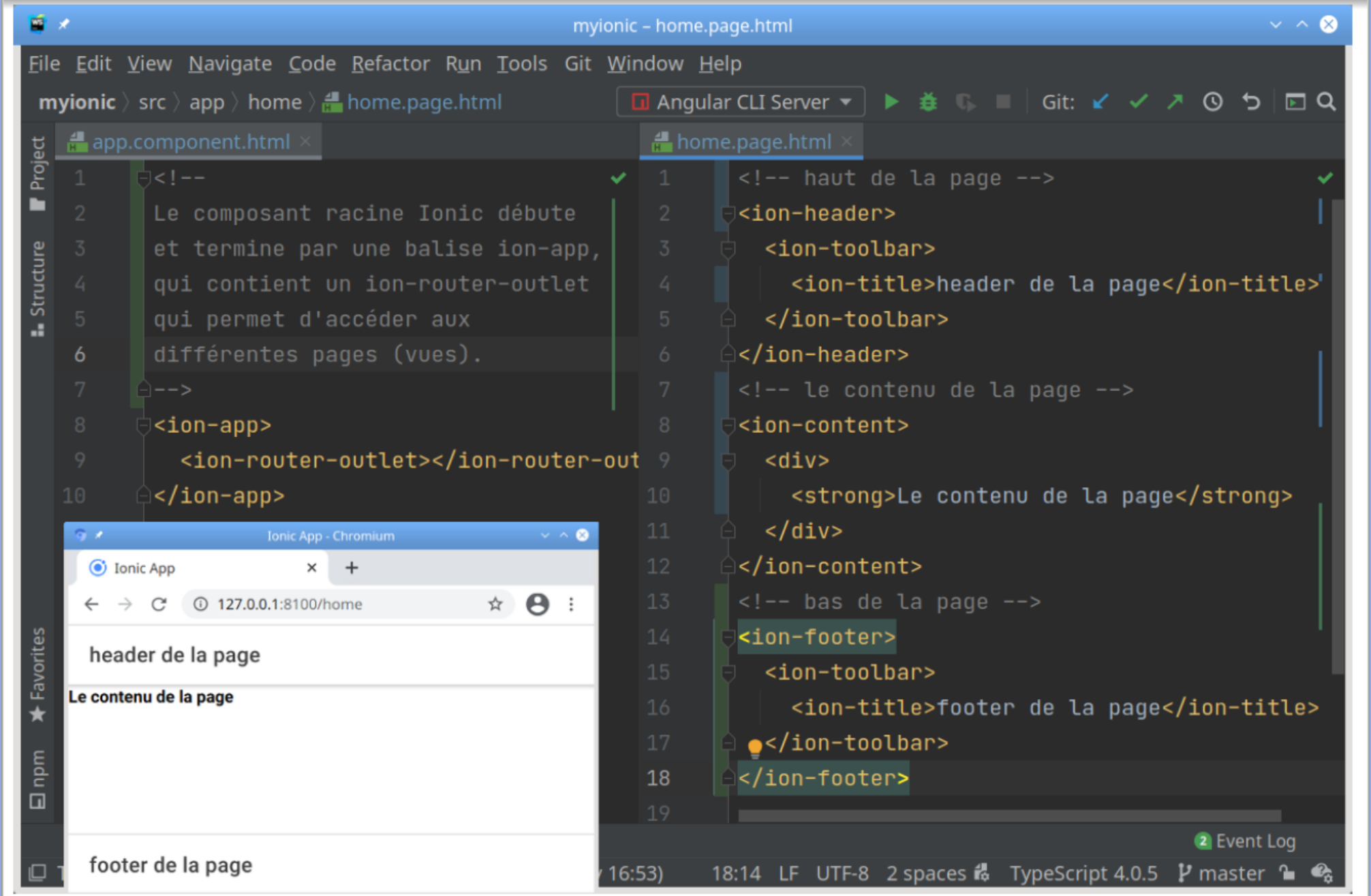
# Premiers pas en Ionic

- Installation : `npm install -g @ionic/cli`
- Création projet : `ionic start monprojet`
  - Framework : 
  - Template : blank
  - Compte ionic : au choix
- Test : `ionic serve`

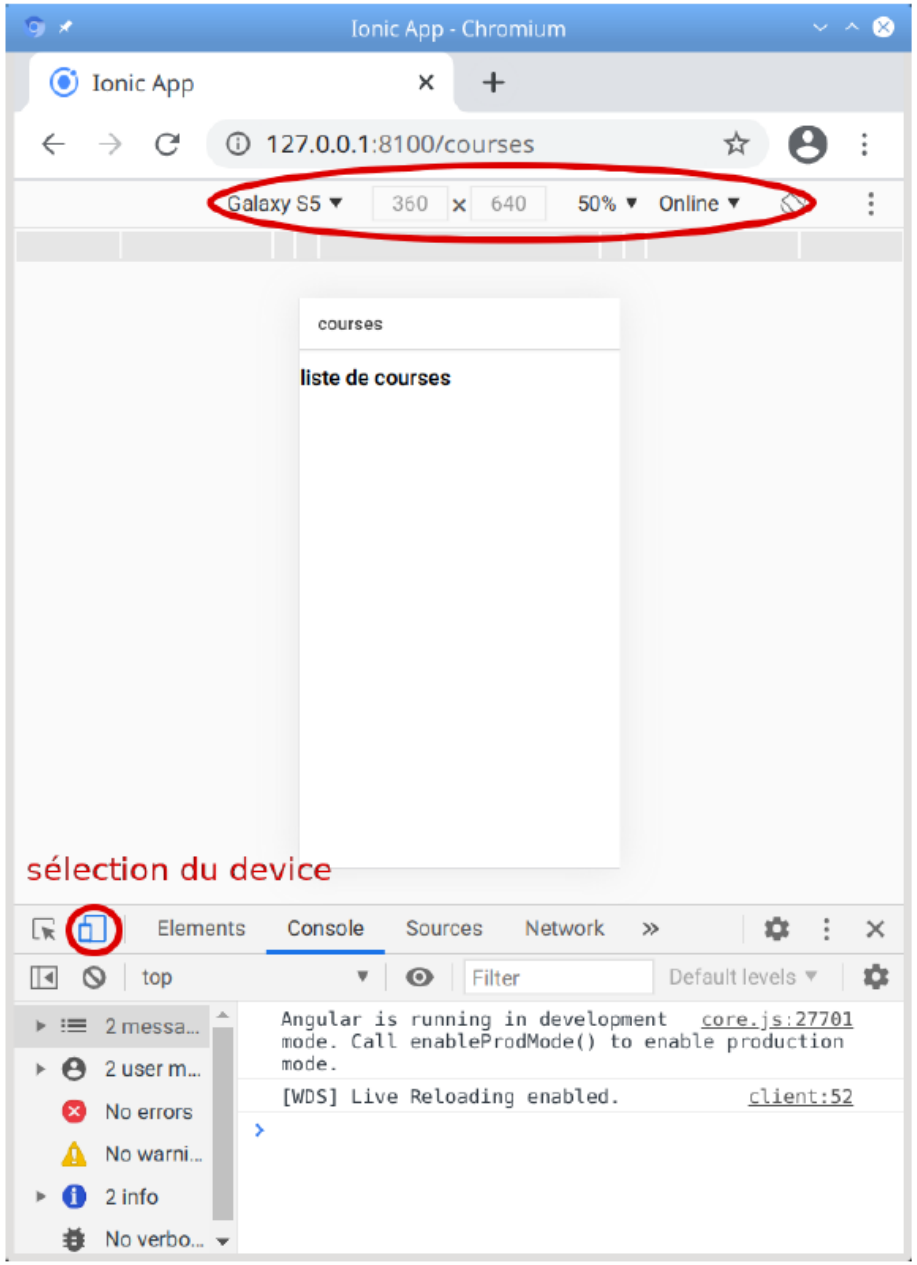
  $\Longrightarrow$ `http://127.0.0.1:8100`
- Capacitor : inclus par défaut



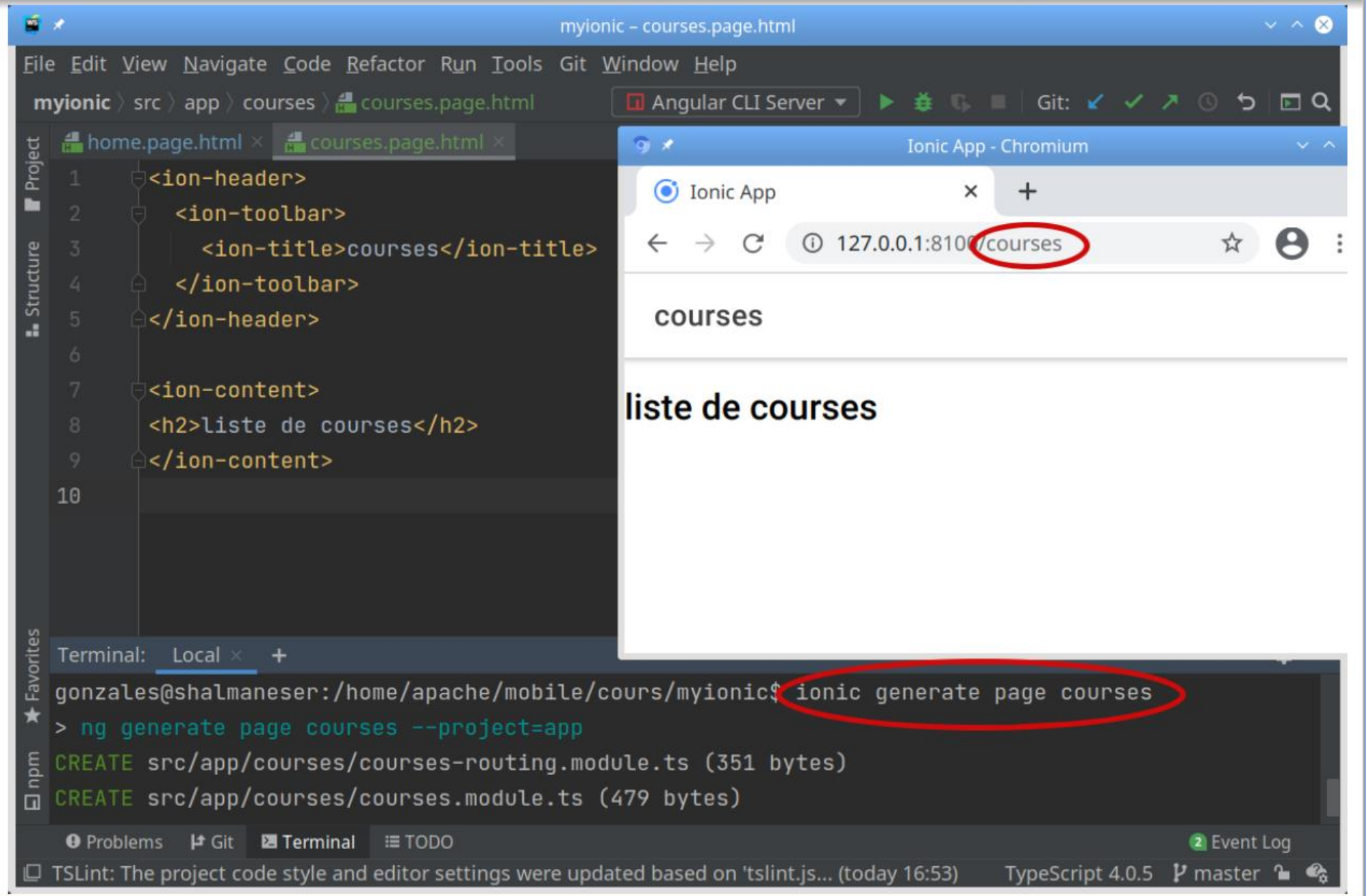
# Anatomie d'une application / page



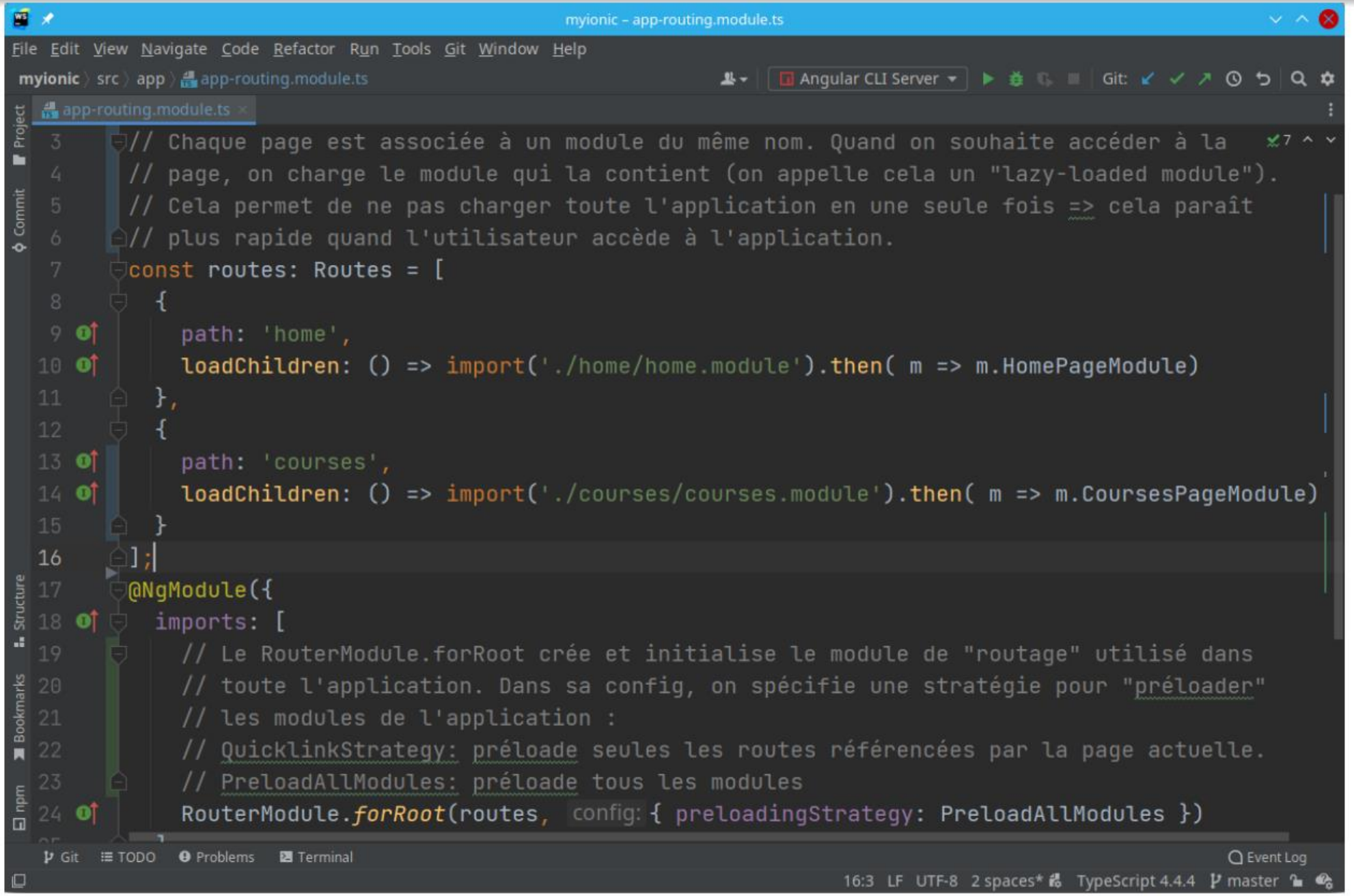
# Visualisation d'une page dans un browser



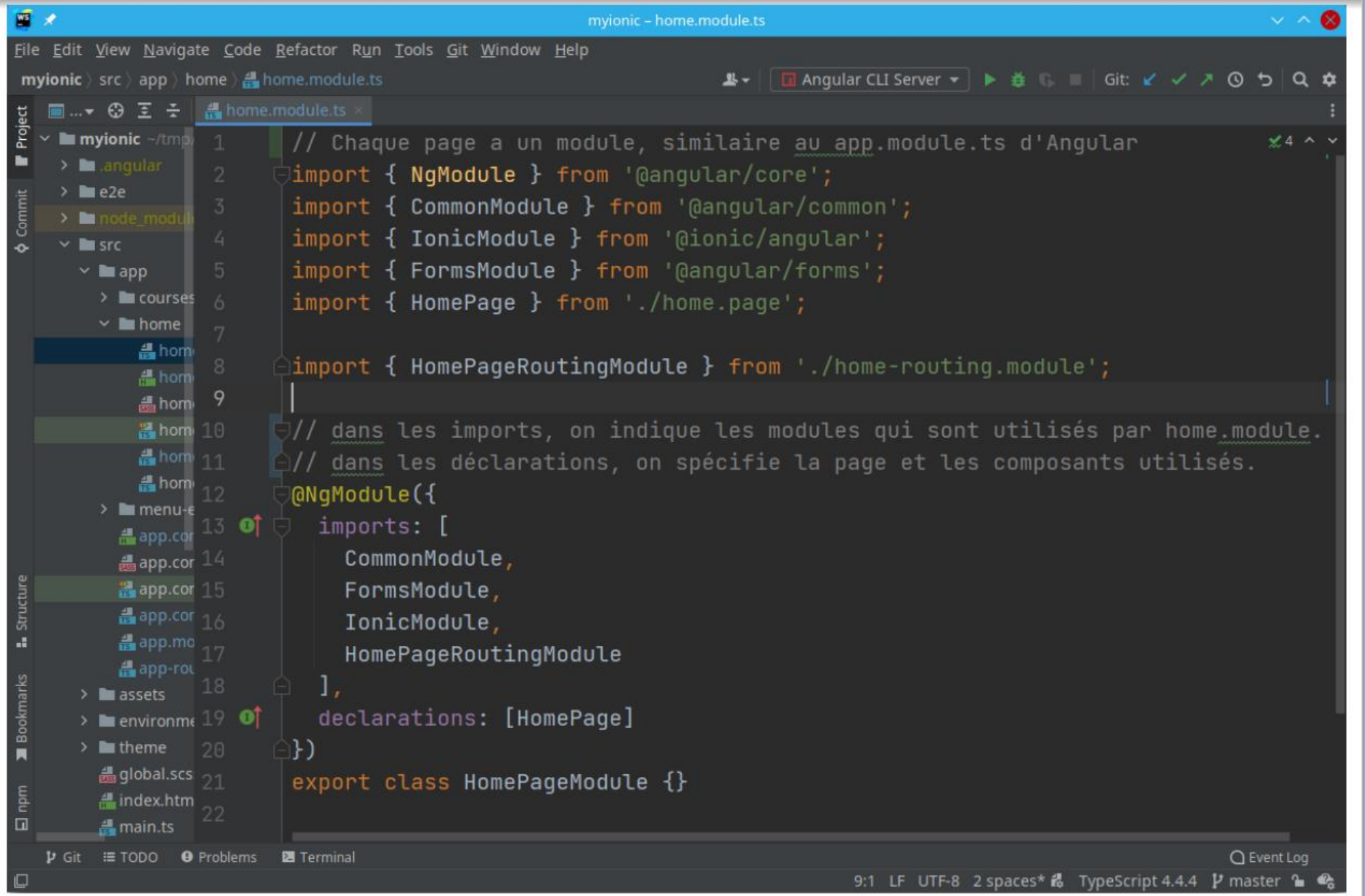
# Création d'une nouvelle page



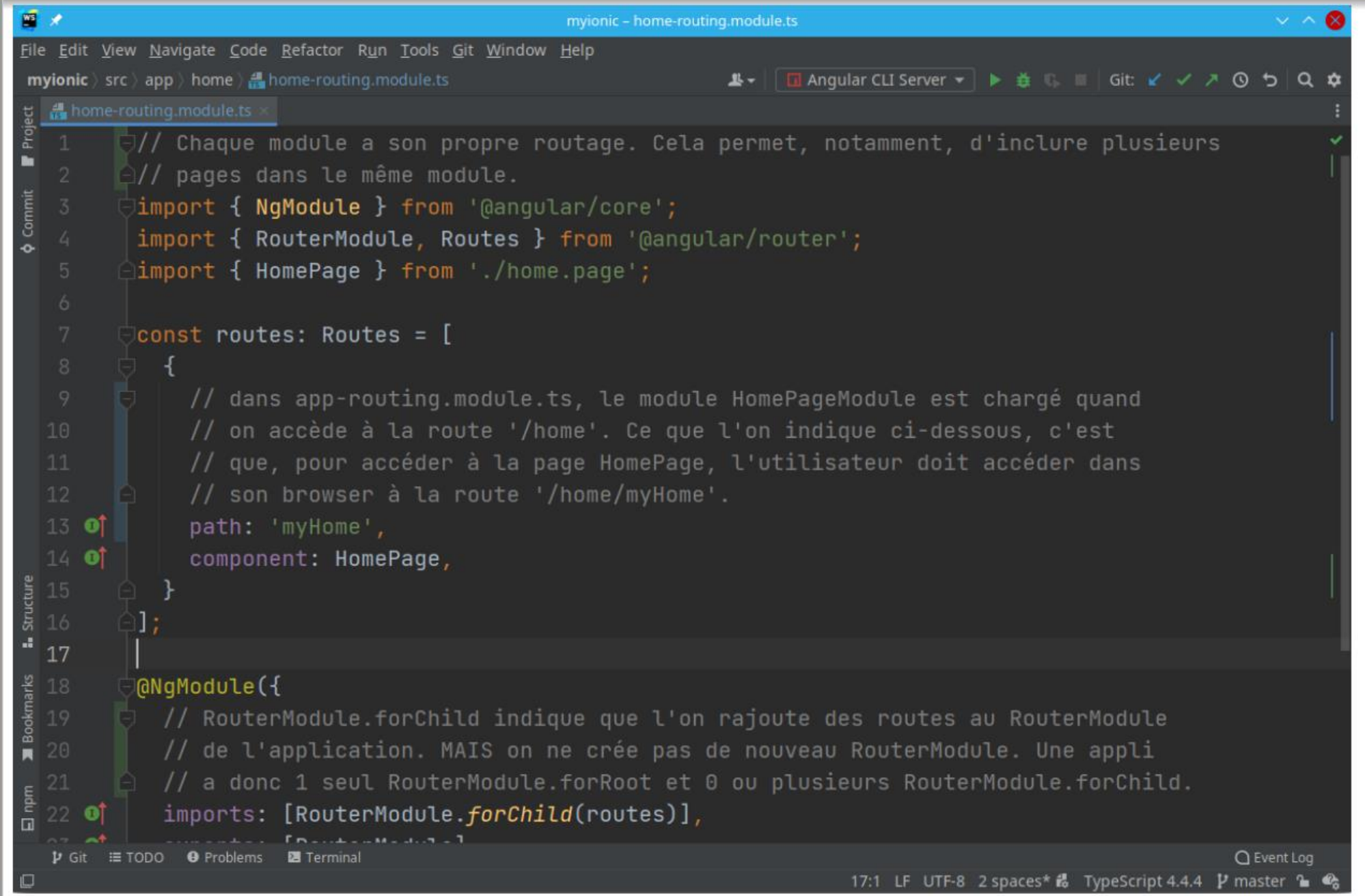
# Chargement des pages (1/3)



# Chargement des pages (2/3)



# Chargement des pages (3/3)



# Résumé sur les pages

- **Création :** `ionic generate page mapage`
  - $\Longrightarrow$ **Extension :** `.page`
  - $\Longrightarrow$ crée le template HTML, le TypeScript, le CSS/SCSS et le module et son routage
- **Rapidité :** Modules des pages chargés dynamiquement
- **Composition :** Les pages contiennent des composants
  - $\Longrightarrow$ `ionic generate component moncomponsant`
  - $\Longrightarrow$ Composant $\neq$ Page

# Quelques composants 

# Création d'une liste

myionic – courses.page.html

File Edit View Navigate Code Refactor Run Tools Git Window Help

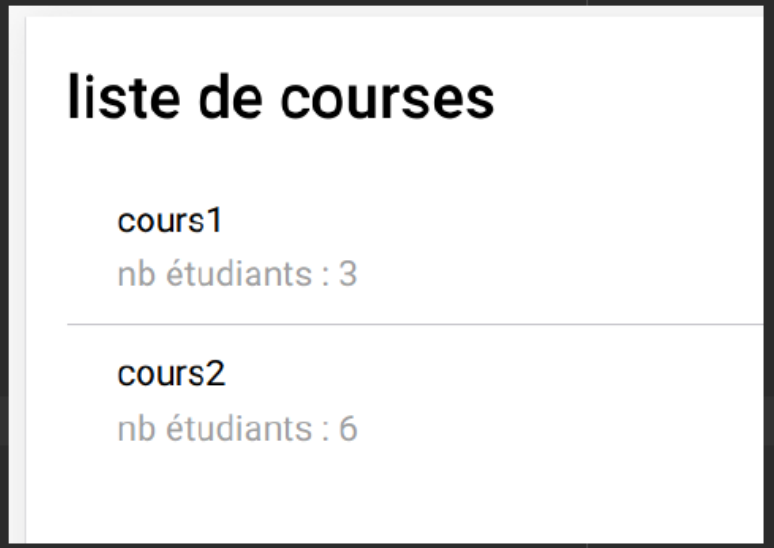
myionic 〉 src 〉 app 〉 courses 〉 courses.page.html

Angular CLI Server

Git:

courses.page.html

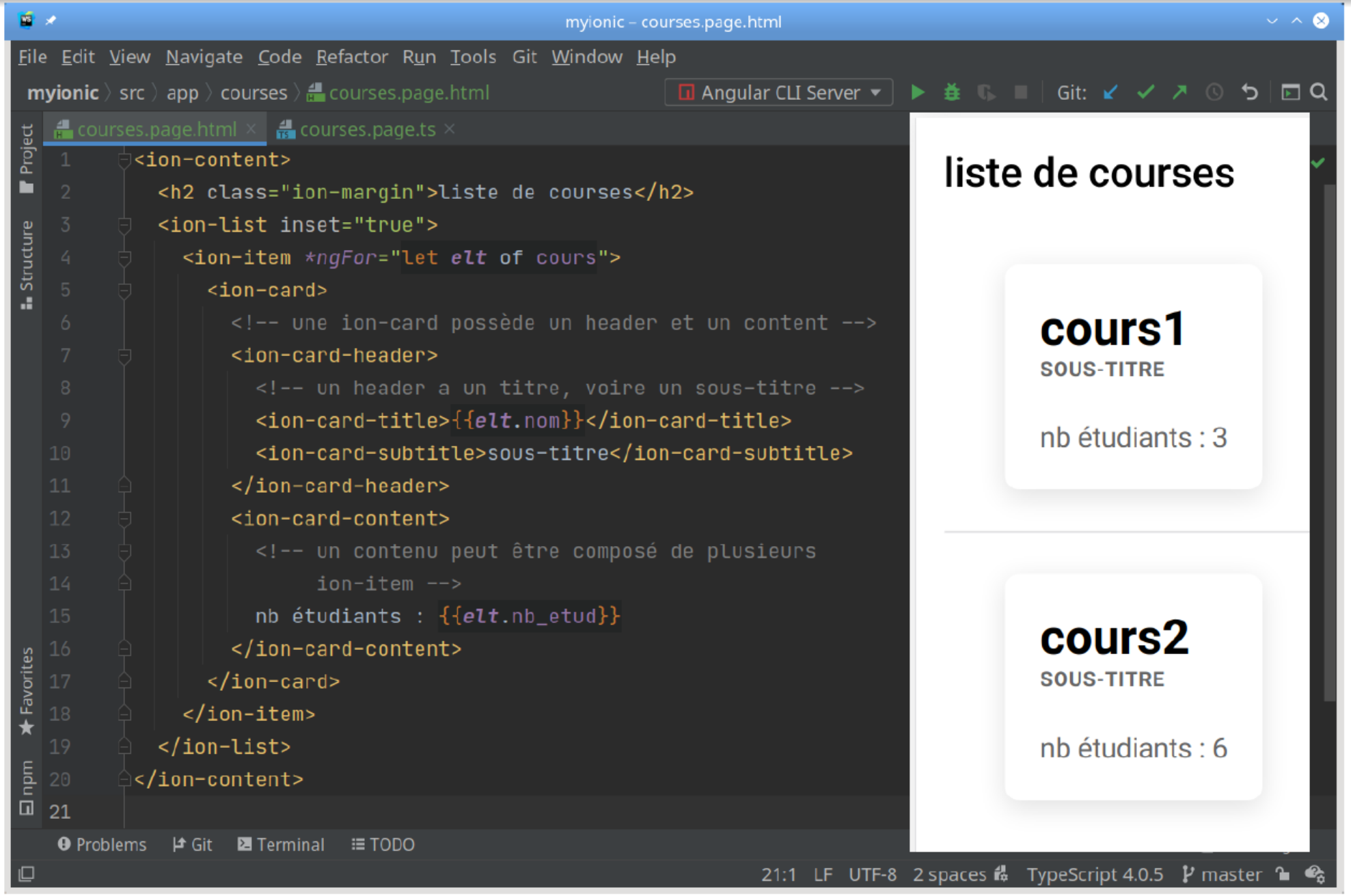
```
1  <ion-content>
2    <h2 class="ion-margin">liste de courses</h2>
3
4    <!-- ion-list : crée une liste avec le look and feel du smartphone (iphone, etc) -->
5    <ion-list inset="true">
6      <!-- chaque élément de la liste est un ion-item -->
7      <ion-item *ngFor="let elt of cours">
8        <!-- pour bien positionner le contenu de l'item, il est pratique d'encapsuler
9             celui-ci dans un ion-label -->
10       <ion-label>
11         <h4>{{elt.nom}}</h4>
12         <p>nb étudiants : {{elt.nb_etud}}</p>
13       </ion-label>
14     </ion-item>
15   </ion-list>
16
17 </ion-content>
18
```




Problems Git Terminal TODO Event Log

18:1 LF UTF-8 2 spaces TypeScript 4.0.5 master

# Création d'une ion-card



# Les ancres, moussaillon !

myionic – courses.page.html

File Edit View Navigate Code Refactor Run Tools Git Window Help

myionic 〉 src 〉 app 〉 courses 〉 courses.page.html

Angular CLI Server

Git:

courses.page.html

```
1  <ion-content>
2    <h2 class="ion-margin">liste de courses</h2>
3    <ion-list inset="true">
4      <ion-item *ngFor="let elt of cours">
5        <ion-label>
6          <h4>{{elt.nom}}</h4>
7          <p>nb étudiants : {{elt.nb_etud}}</p>
8          <p>
9            <!-- en ionic, avec Angular, on peut la balise <a> ou la balise <ion-anchor>
10                (obsolète en ionic 5) mais il faut éviter <ion-router-link>. Les ancres
11                permettent de passer d'une page à l'autre avec une animation. -->
12           <a routerLink='/home'>vers Home</a>
13         </p>
14       </ion-label>
15     </ion-item>
16   </ion-list>
17 </ion-content>
18
```

Project Structure Favorites npm

Problems Git Terminal TODO Event Log

18:1 LF UTF-8 2 spaces TypeScript 4.0.5 master

# Les back buttons

myionic – home.page.html

File Edit View Navigate Code Refactor Run Tools Git Window Help

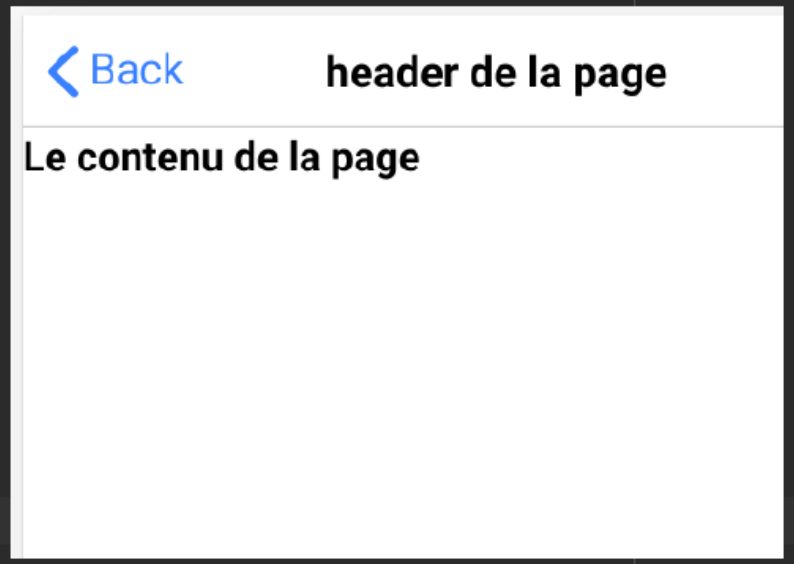
myionic 〉 src 〉 app 〉 home 〉 home.page.html

Angular CLI Server

Git:

home.page.html

```
1  <!-- haut de la page -->
2  <ion-header>
3    <ion-toolbar>
4      <ion-title>header de la page</ion-title>
5
6      <!-- création d'un "back button" pour revenir à la page précédente-->
7      <ion-buttons slot="start">
8        <ion-back-button></ion-back-button>
9      </ion-buttons>
10
11   </ion-toolbar>
12 </ion-header>
13
14 <ion-content>
15   <div>
16     <strong>Le contenu de la page</strong>
17   </div>
18 </ion-content>
19
```




Project Structure Favorites npm

Problems Git Terminal TODO Event Log

19:1 LF UTF-8 2 spaces TypeScript 4.0.5 master

# Les menus

**2 manières de construire les menus :**

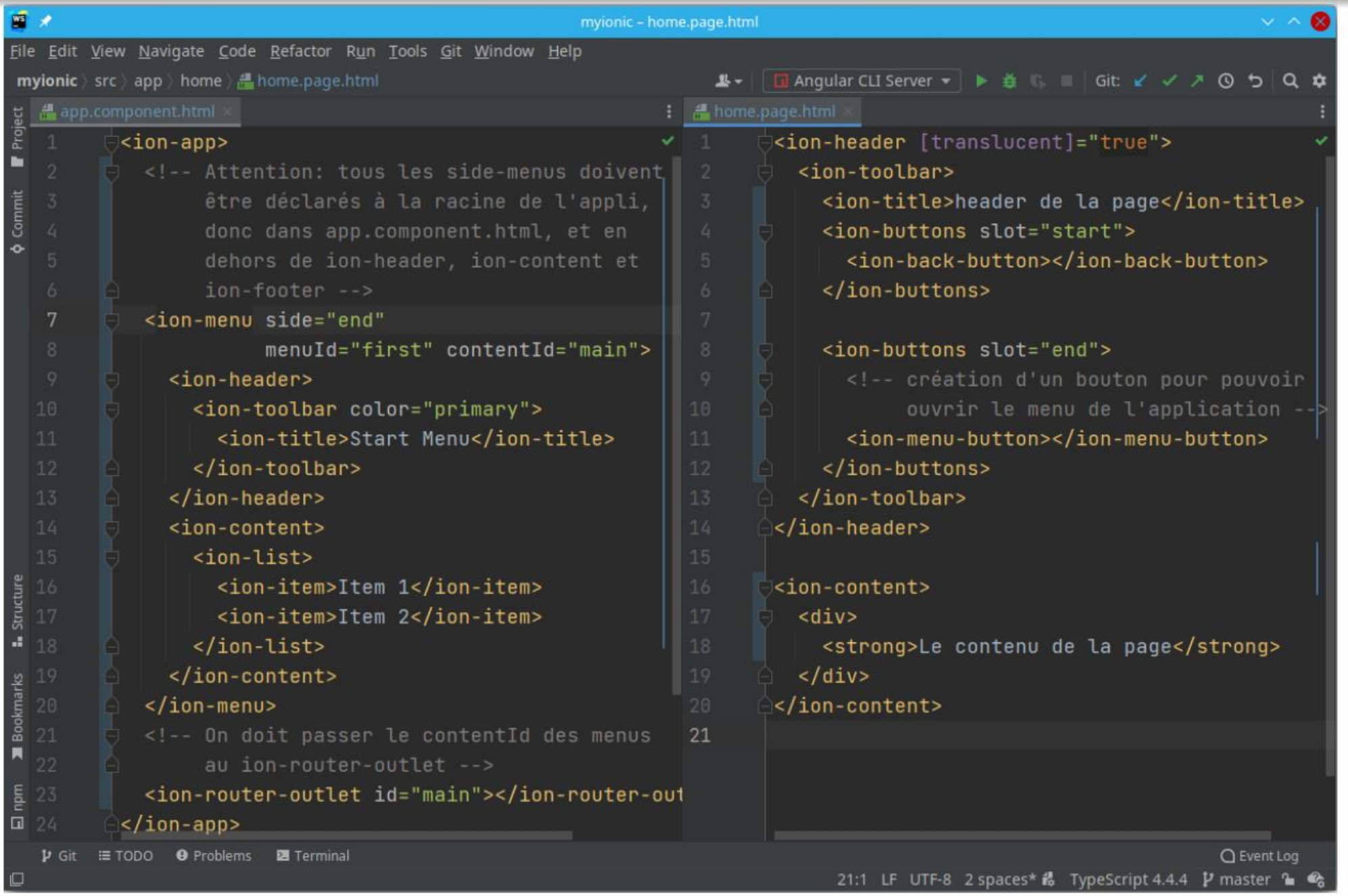
1. Créer des « side »-menus :

   $\Longrightarrow$ Utiliser la balise `<ion-menu>`

2. Créer des menus déroulants :

   $\Longrightarrow$ Utiliser un `popover`

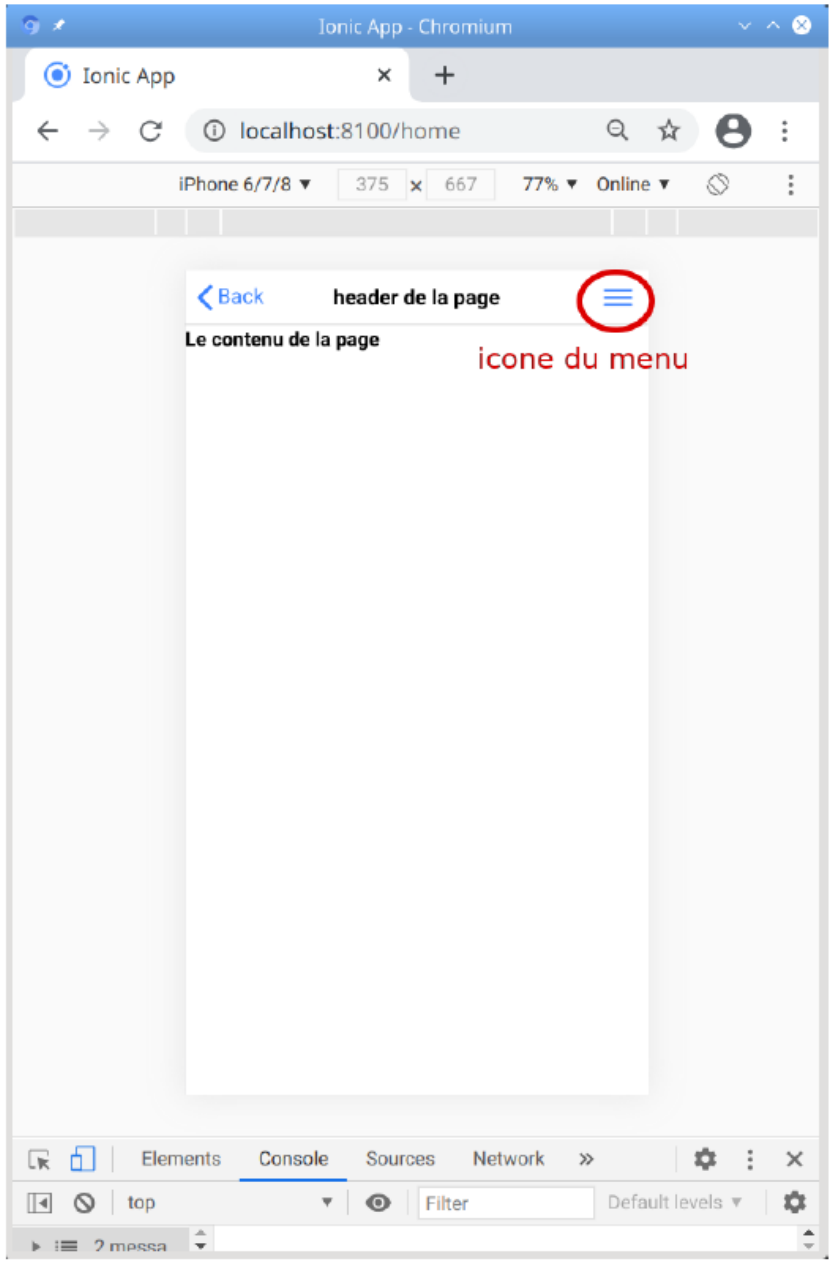
# Un side-menu pour toute l'application (1/2)

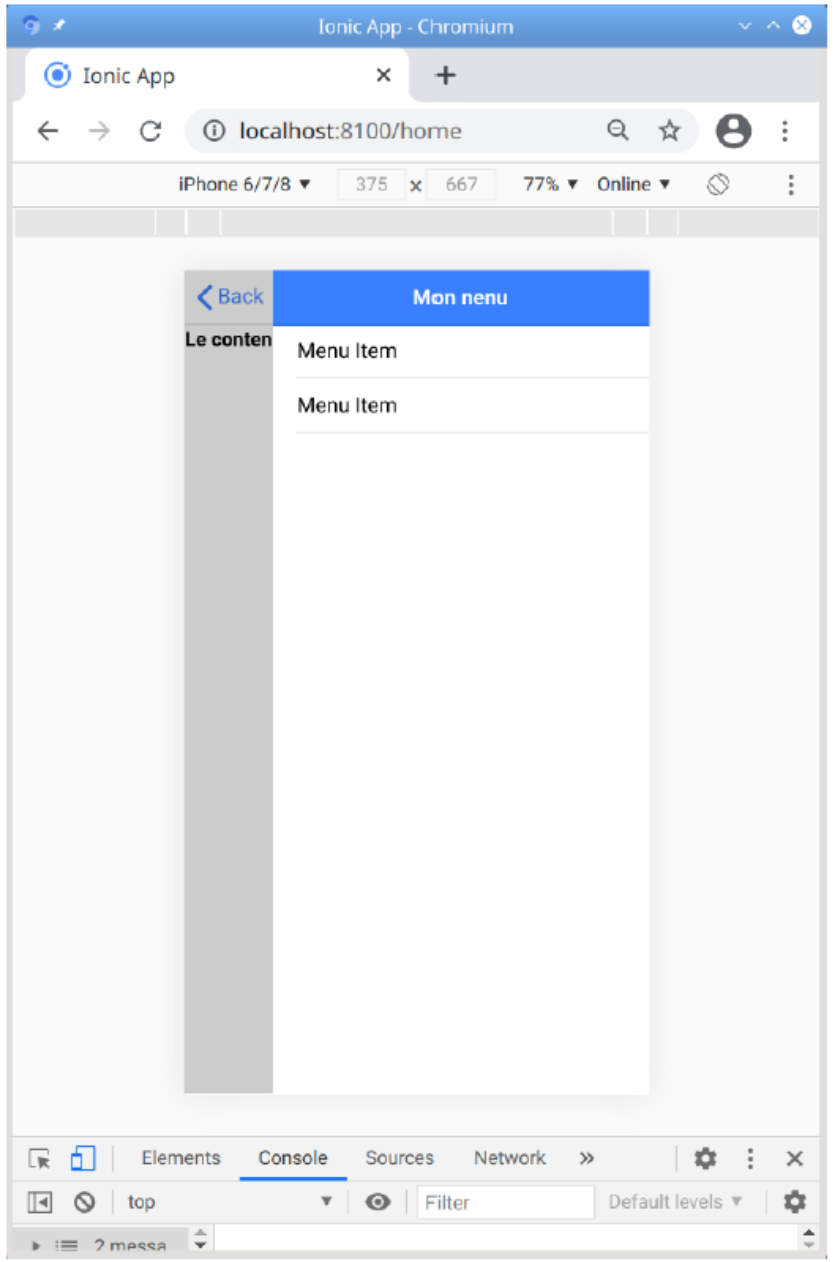



```
<ion-app>
  <!-- Attention: tous les side-menus doivent
       être déclarés à la racine de l'appli,
       donc dans app.component.html, et en
       dehors de ion-header, ion-content et
       ion-footer -->
  <ion-menu side="end"
            menuId="first" contentId="main">
    <ion-header>
      <ion-toolbar color="primary">
        <ion-title>Start Menu</ion-title>
      </ion-toolbar>
    </ion-header>
    <ion-content>
      <ion-list>
        <ion-item>Item 1</ion-item>
        <ion-item>Item 2</ion-item>
      </ion-list>
    </ion-content>
  </ion-menu>
  <!-- On doit passer le contentId des menus
       au ion-router-outlet -->
  <ion-router-outlet id="main"></ion-router-out
</ion-app>
```

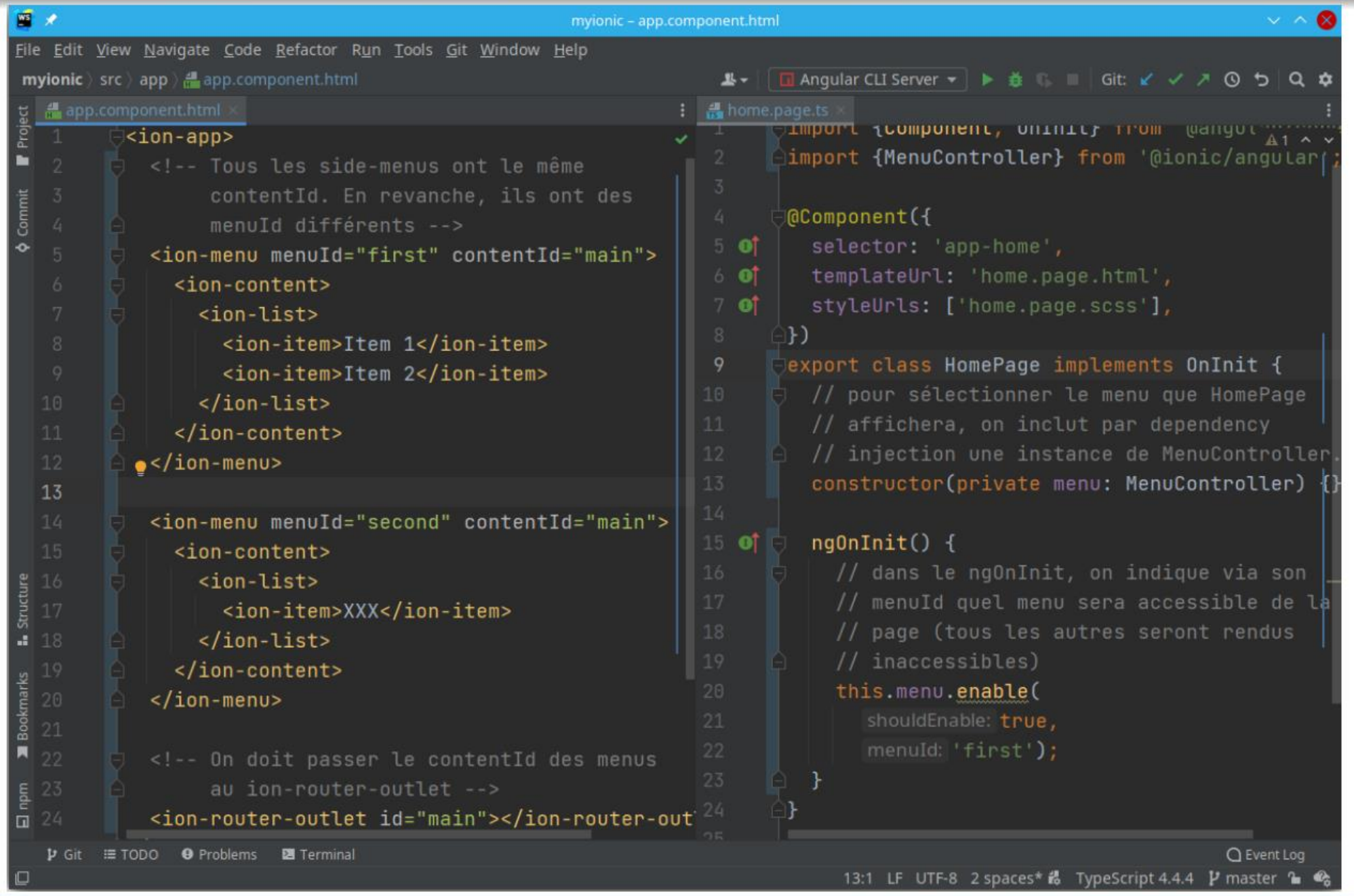
```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>header de la page</ion-title>
    <ion-buttons slot="start">
      <ion-back-button></ion-back-button>
    </ion-buttons>

    <ion-buttons slot="end">
      <!-- création d'un bouton pour pouvoir
           ouvrir le menu de l'application -->
      <ion-menu-button></ion-menu-button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>

<ion-content>
  <div>
    <strong>Le contenu de la page</strong>
  </div>
</ion-content>
```

# Un side-menu pour toute l'application (2/2)

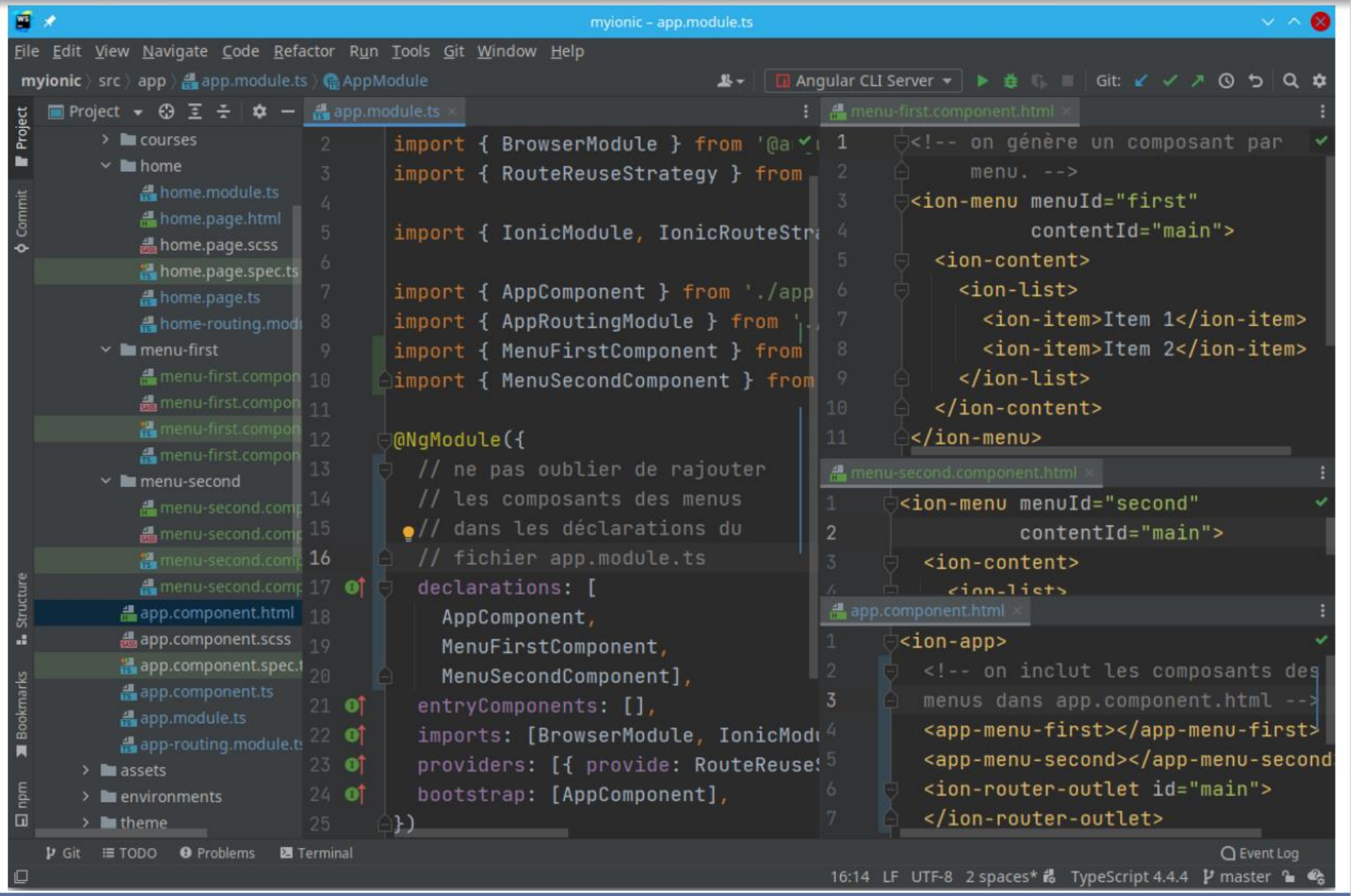


# Plusieurs side-menus



# Organisation possible du code par composants



# Menu déroulant (1/5)

**4 étapes :**

1. Créer un composant contenant le code du menu
2. Rajouter le composant au module de la page
3. Dans le html de la page, créer un bouton pour ouvrir le menu
4. Dans le TypeScript de la page, ajouter une méthode créant un `PopoverController`

# Menu déroulant (2/5)

myionic – menu.component.ts

File Edit View Navigate Code Refactor Run Tools Git Window Help

myionic › src › app › menu › menu.component.ts › MenuComponent

Angular CLI Server

Git:

menu.component.ts

```
9   export class MenuComponent implements OnInit {
10    // penser à rajouter MenuComponent aux déclarations de home.module.ts.
11    // Le champ ci-dessous montre comment le MenuComponent peut transmettre
12    // des informations au menu quand celui-ci s'ouvrira
13    myprop! : string;
14    nb = [1, 2, 3]; // les items du menu déroulant
15
16    constructor(private popover: PopoverController) { }
17
18    ngOnInit(): void {}
19
20    // la méthode dismiss renverra la valeur de event au HomeComponent
21    closePopover(event): void { this.popover.dismiss(event); }
22  }
```

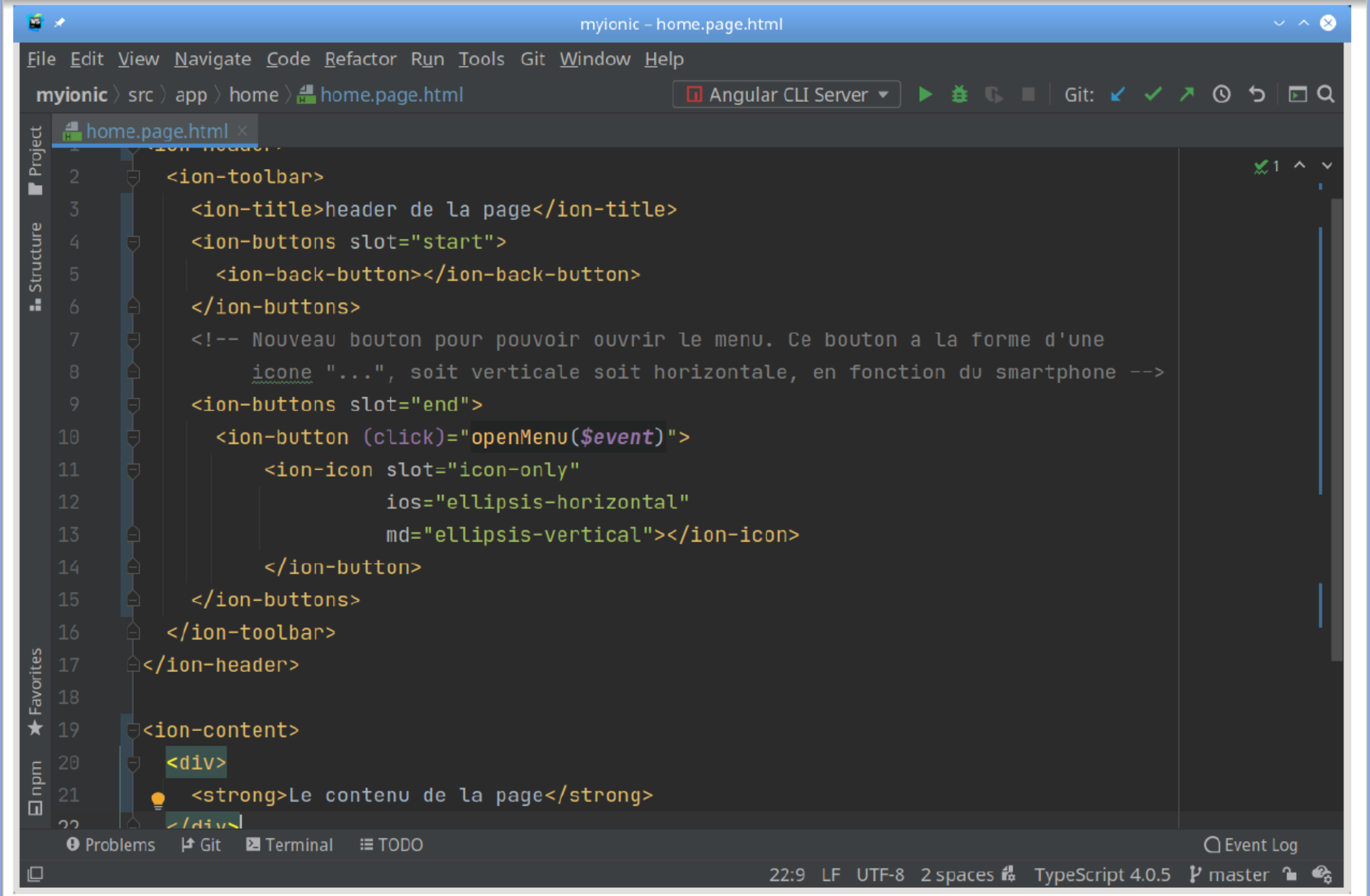
menu.component.html

```
1  <ion-content>
2    <ion-label>Myprop: {{myprop}}</ion-label>
3    <ion-list>
4      <!-- quand on clique, on appelle la fonction qui renvoie l'item sur lequel on clique -->
5      <ion-item *ngFor="let elt of nb" (click)="closePopover(elt)">Menu Item {{elt}}</ion-item>
6    </ion-list>
7  </ion-content>
8
```

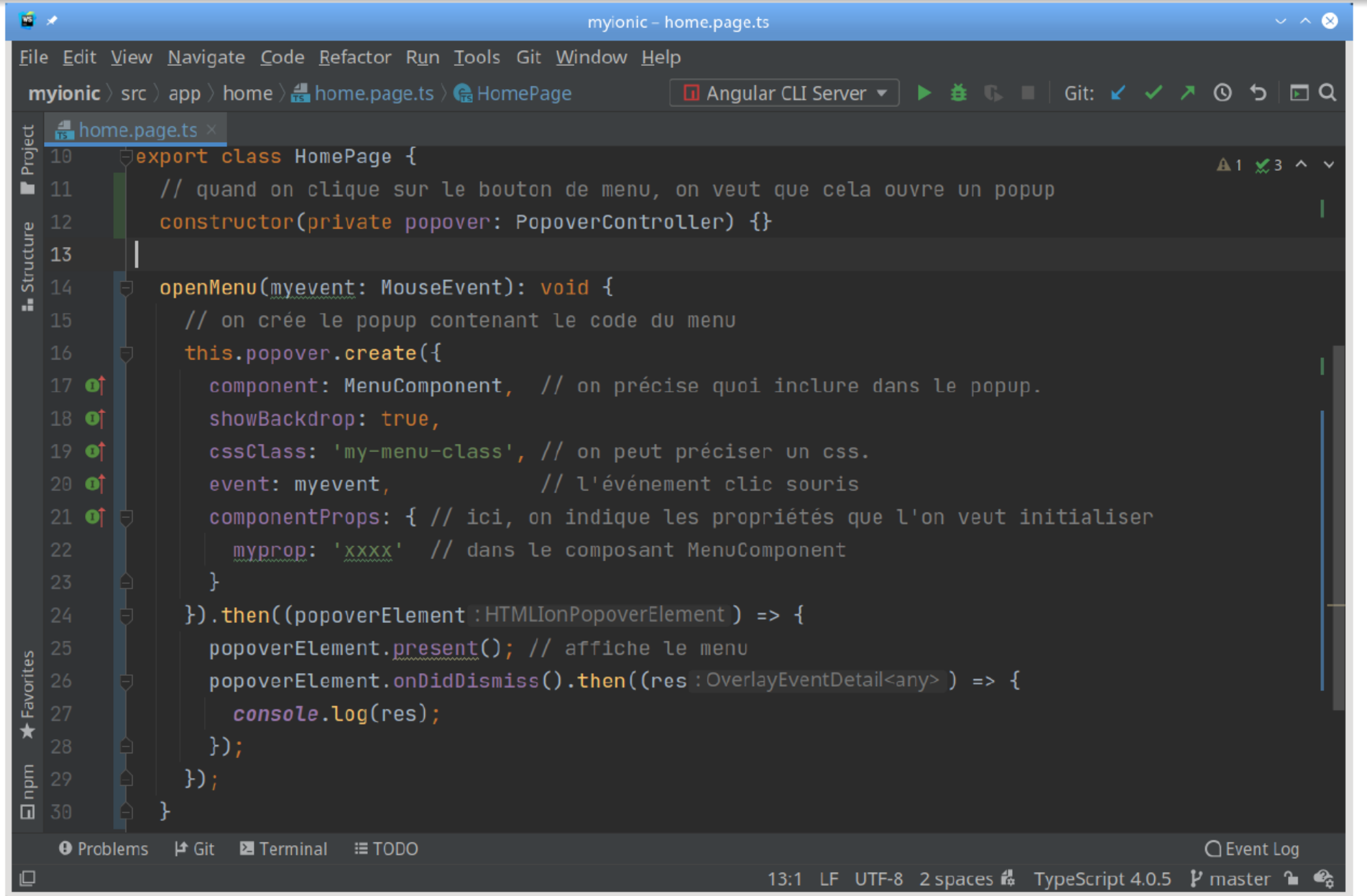
Git TODO Problems Terminal Event Log

15:1 LF UTF-8 2 spaces* TypeScript 4.4.4 master

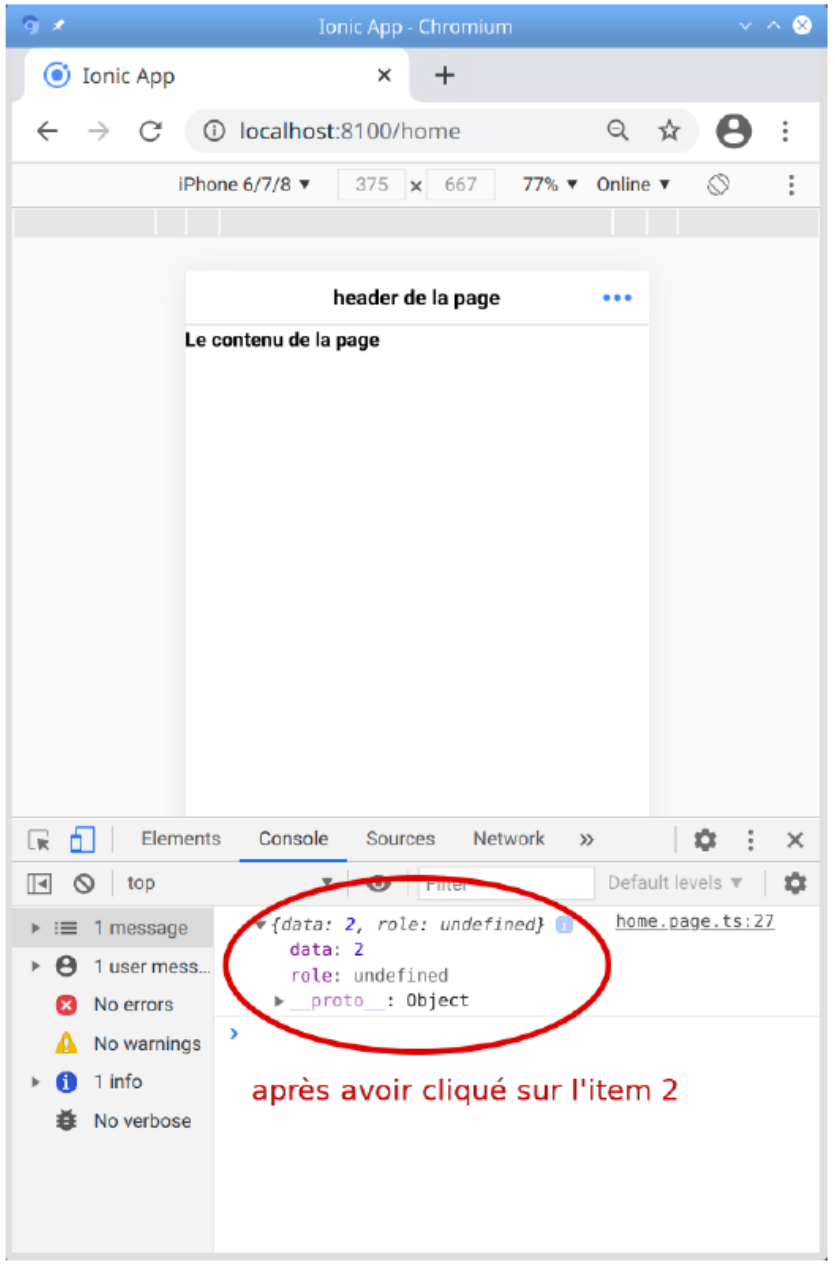
# Menu déroulant (3/5)



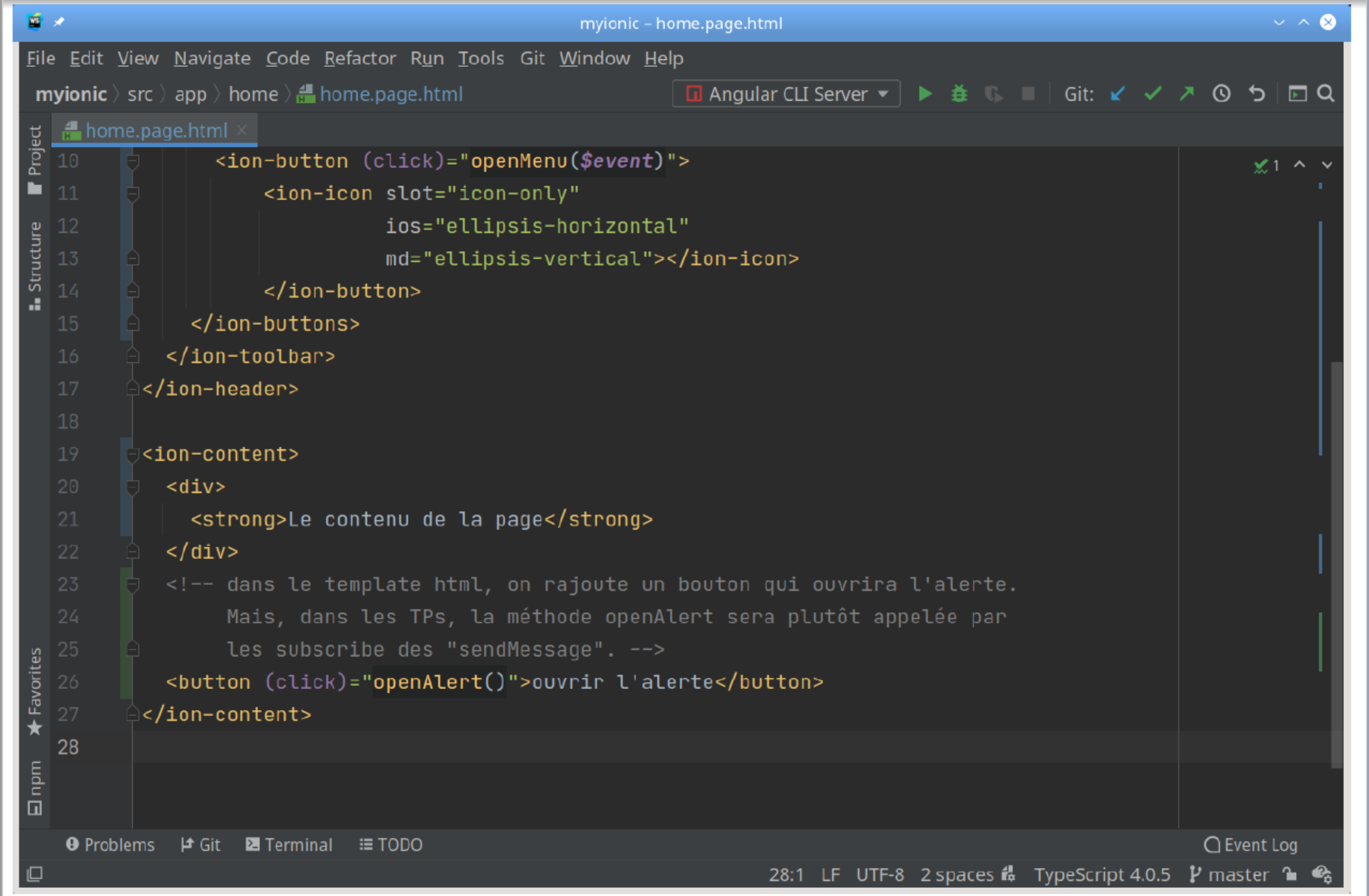
# Menu déroulant (4/5)



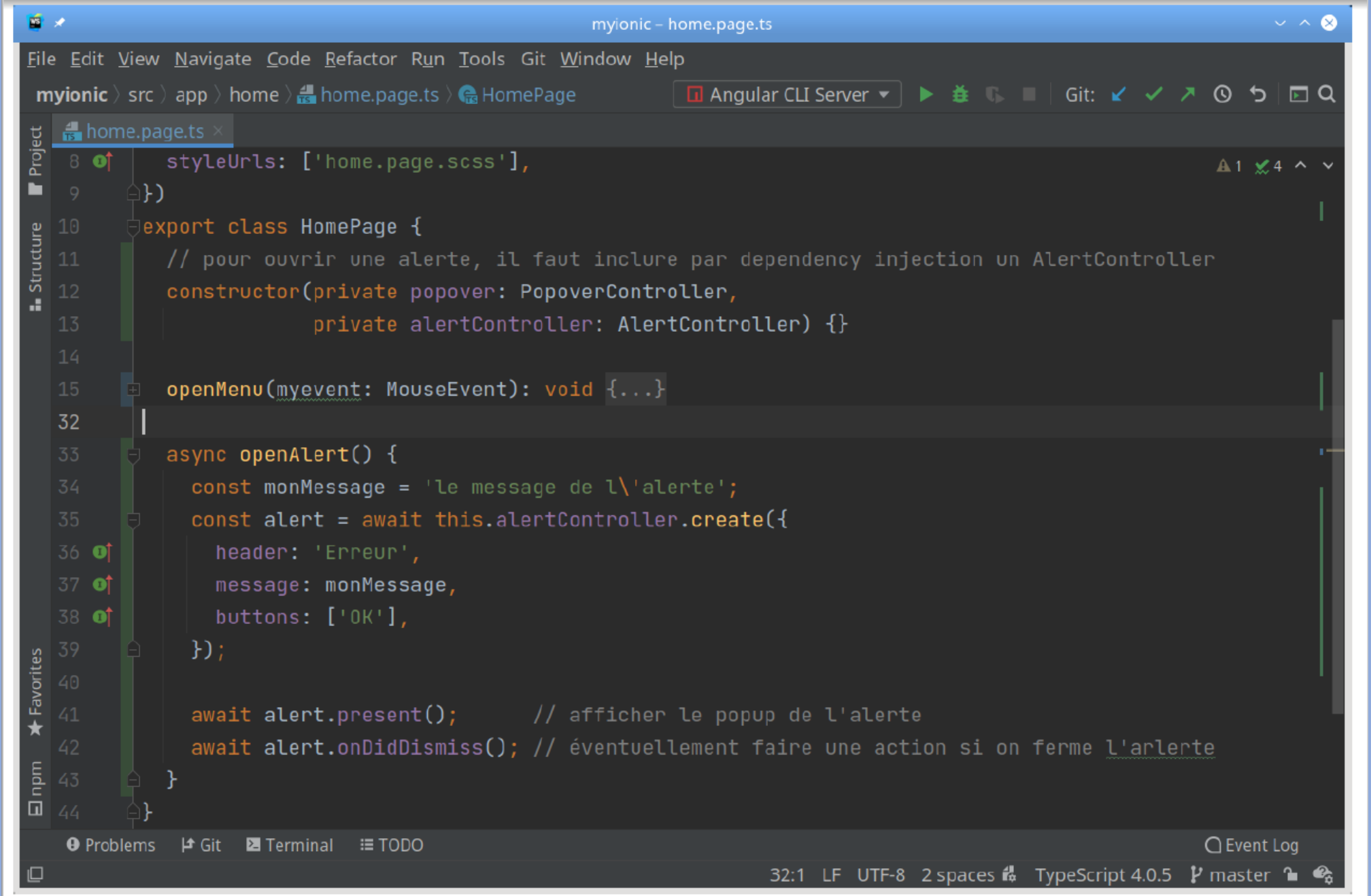
# Menu déroulant (5/5)

# Les alertes (1/3)

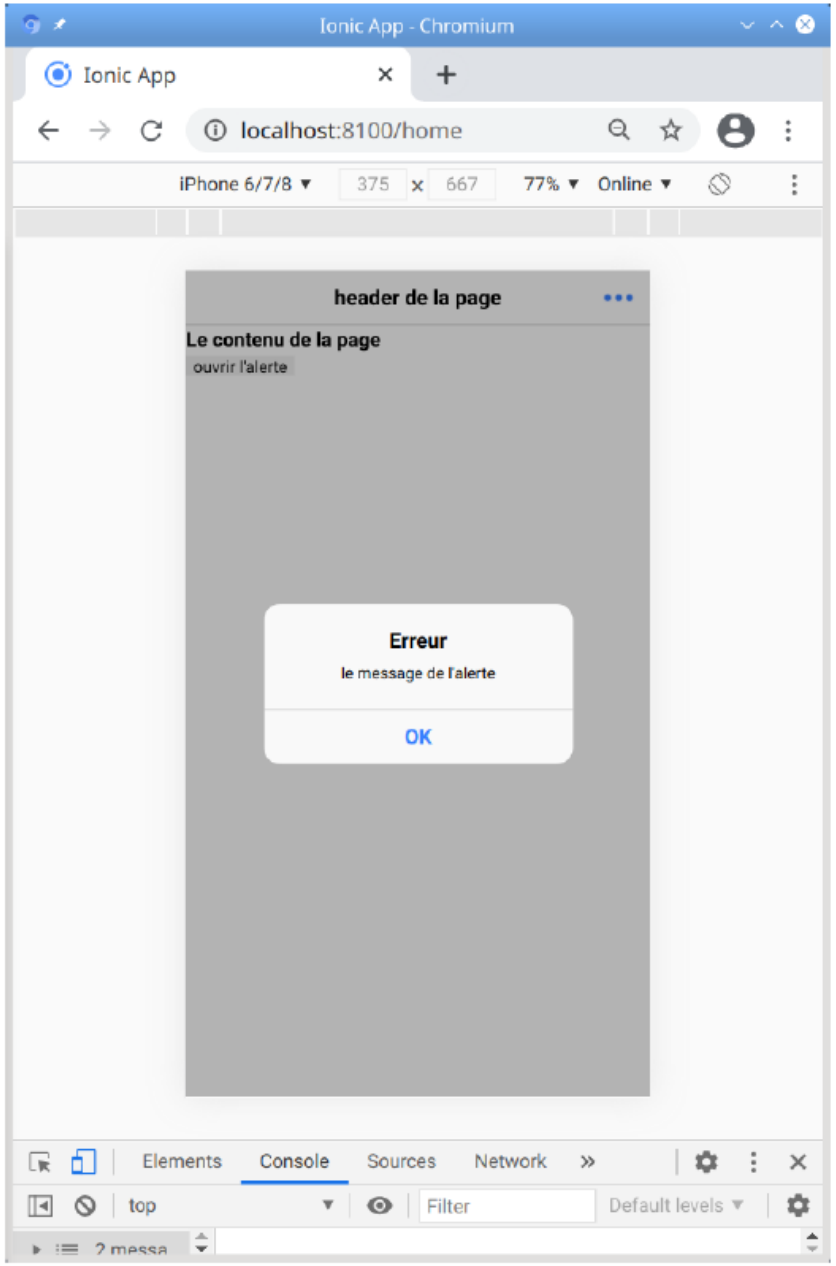


# Les alertes (2/3)

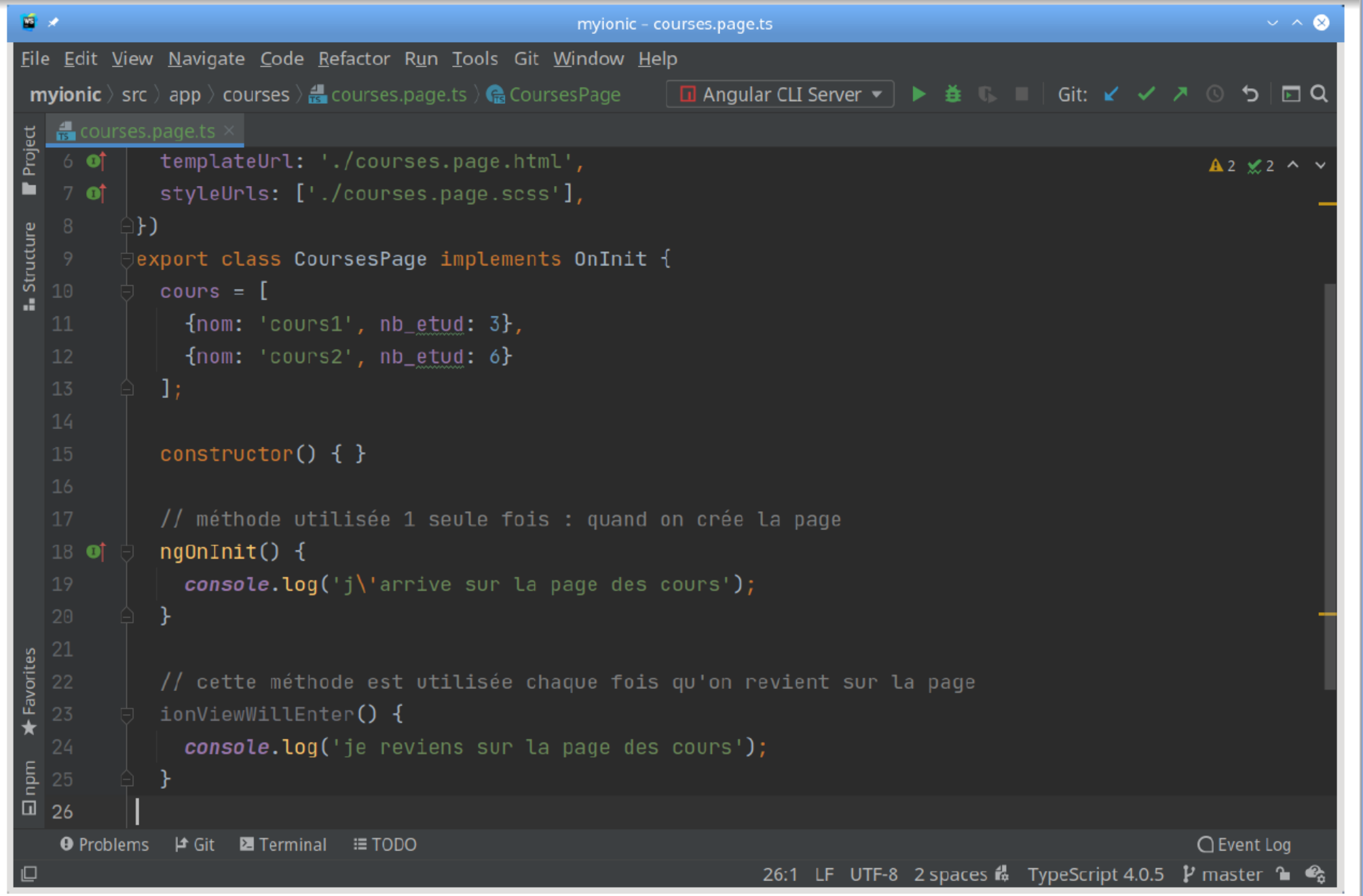



```
8     styleUrls: ['home.page.scss'],
9   })
10  export class HomePage {
11    // pour ouvrir une alerte, il faut inclure par dependency injection un AlertController
12    constructor(private popover: PopoverController,
13                private alertController: AlertController) {}
14
15    openMenu(myevent: MouseEvent): void {...}
32
33    async openAlert() {
34      const monMessage = 'le message de l\'alerte';
35      const alert = await this.alertController.create({
36        header: 'Erreur',
37        message: monMessage,
38        buttons: ['OK'],
39      });
40
41      await alert.present();      // afficher le popup de l'alerte
42      await alert.onDidDismiss(); // éventuellement faire une action si on ferme l'arlerte
43    }
44  }
```

# Les alertes (3/3)



# Faire une action quand on arrive/revient sur une page



# Produire votre application native : le *setup*

## 2 méthodes pour autoriser Ionic à utiliser capacitor :

1. Créer un projet intégrant directement capacitor :

```
ionic start monprojet blank
```

2. Ajouter capacitor *a posteriori* :

```
ionic integrations enable capacitor
```

## Spécifier les plateformes :

- Faire un `ionic build` auparavant ($\Longrightarrow$ crée répertoire `www`)
- Application native  :

```
npm install @capacitor/android
npx cap add android
```

- Application native iOS :

```
npm install @capacitor/ios
npx cap add ios
```

# Produire votre application native : la production

**3 étapes :**

1. Compiler l'appli web : `ionic build`
2. Synchroniser avec l'appli native : `npx cap copy`
3. Ouvrir Android studio ou XCode :

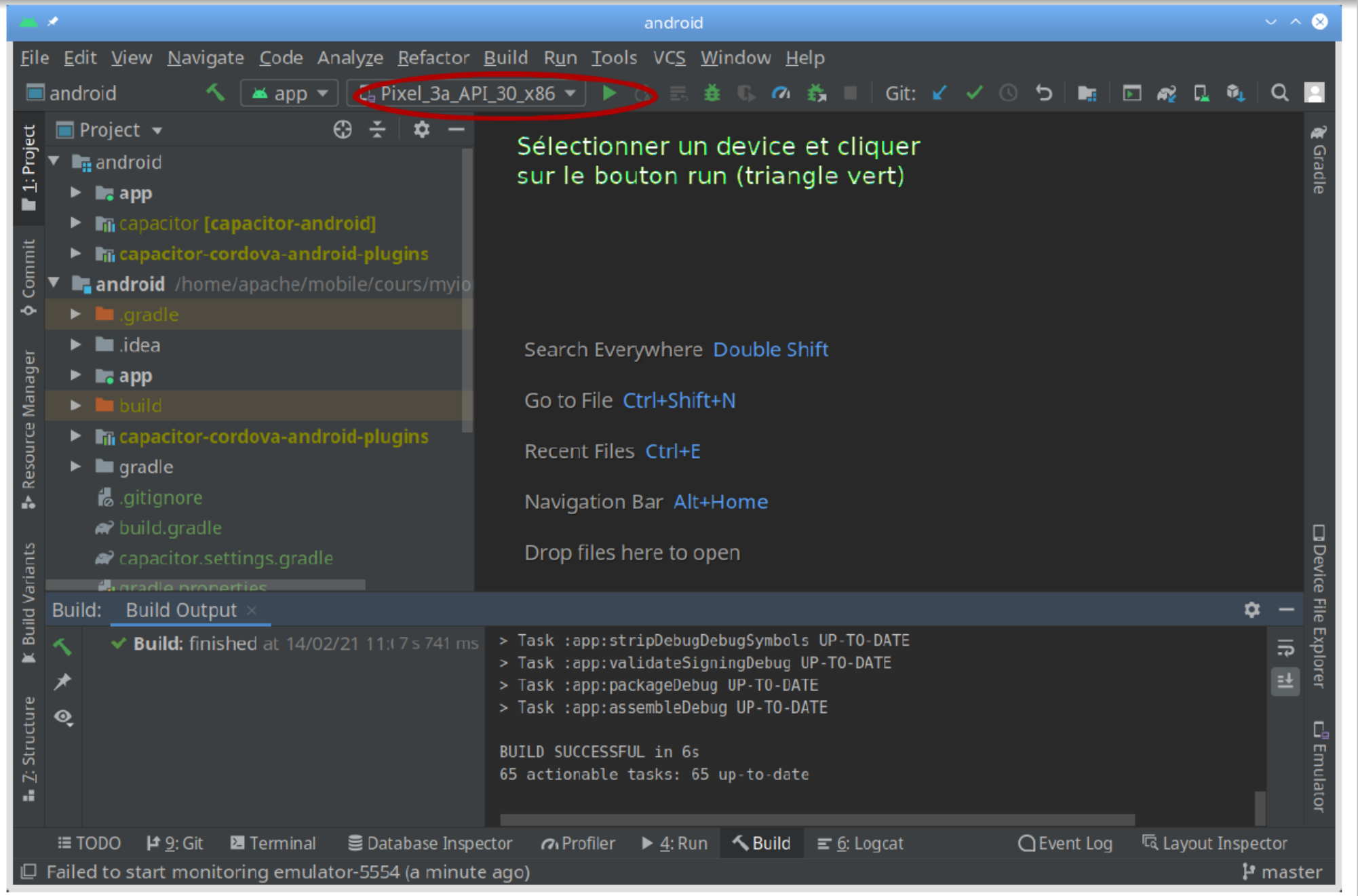
 : `npx cap open android`

 Préciser à la ligne de commande le « path » d'android-studio :

```
export CAPACITOR_ANDROID_STUDIO_PATH=path
```

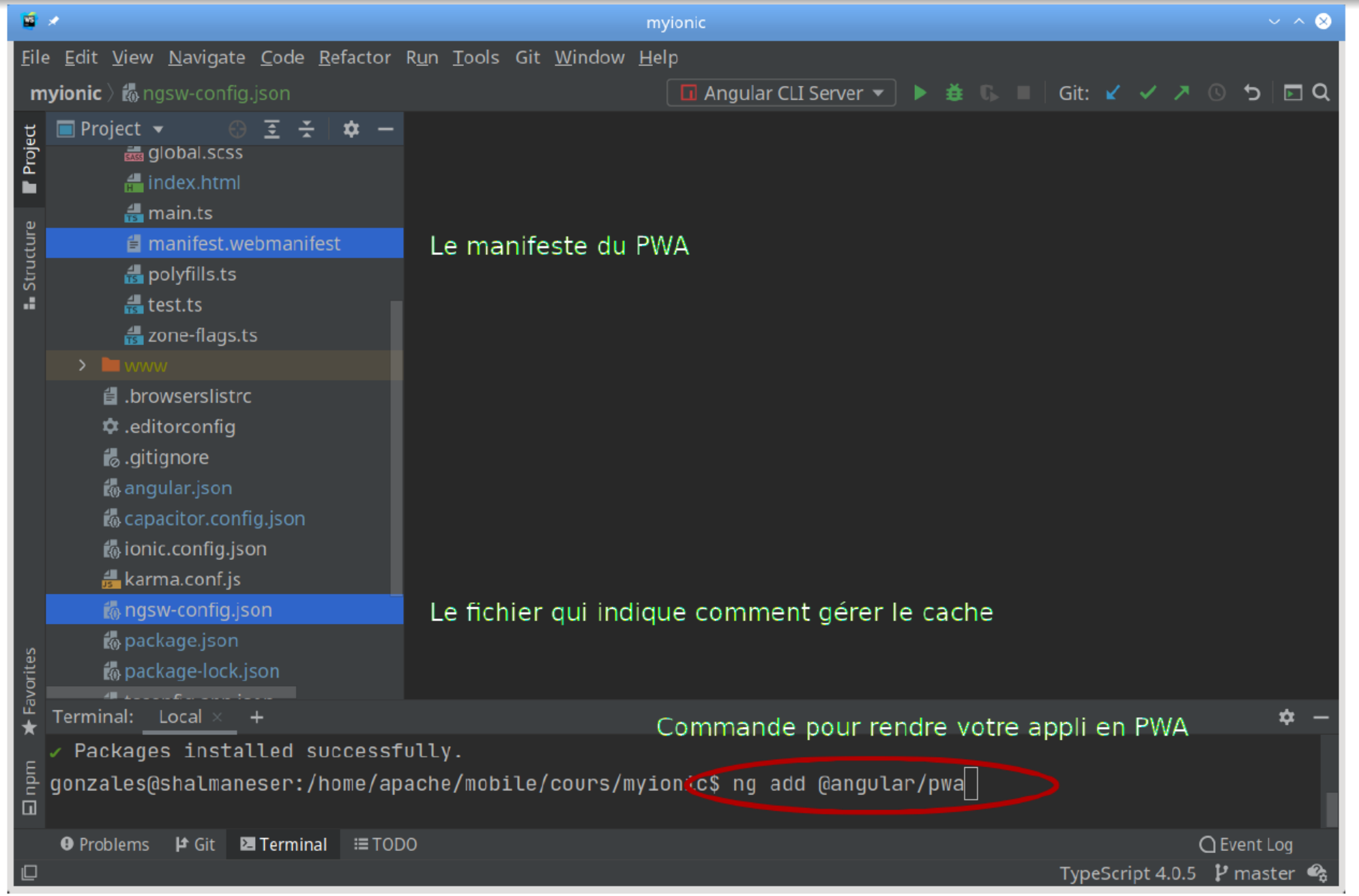
 : `npx cap open ios`

# Production de l'application native : Android studio

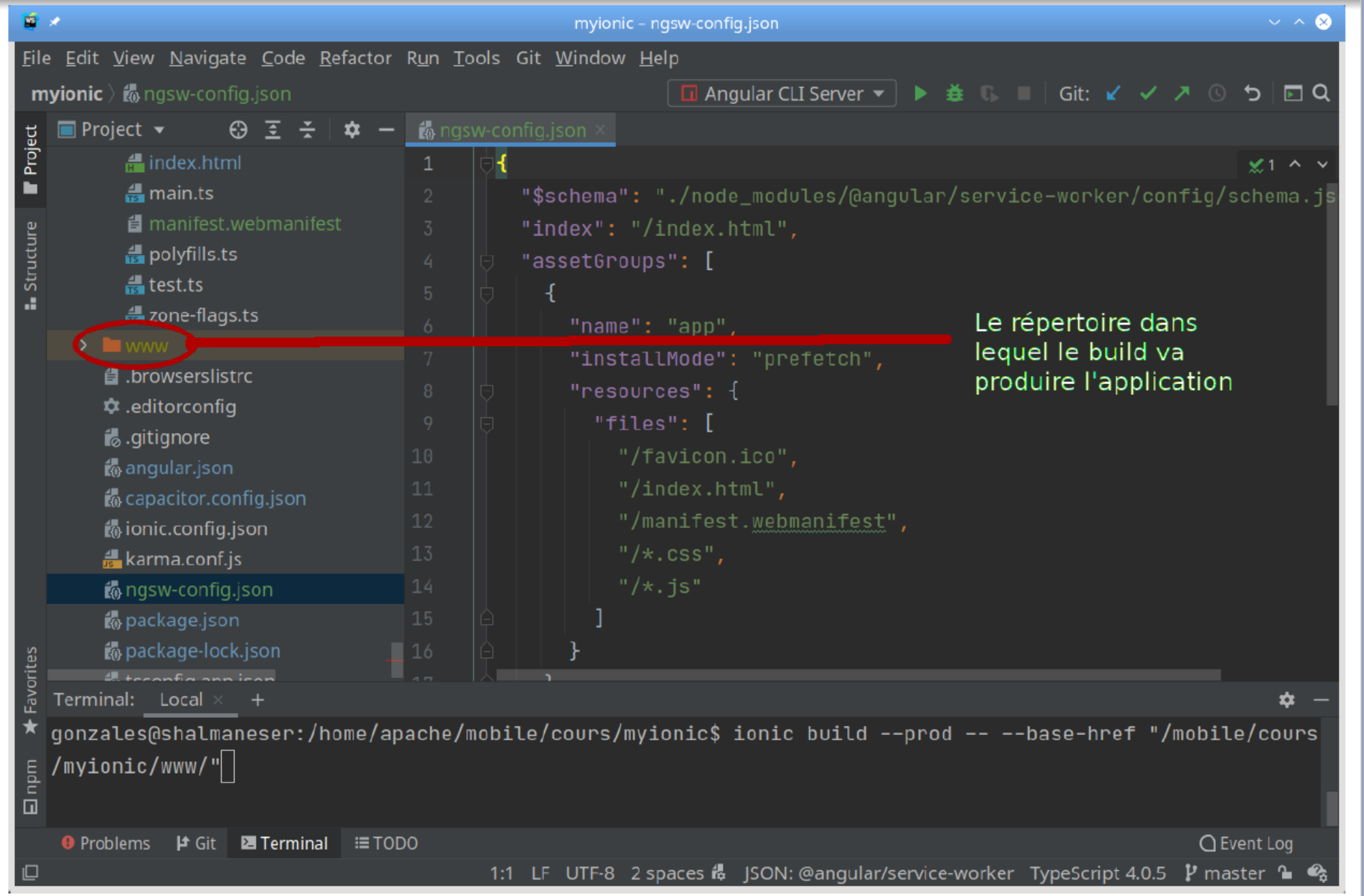


# L'application native

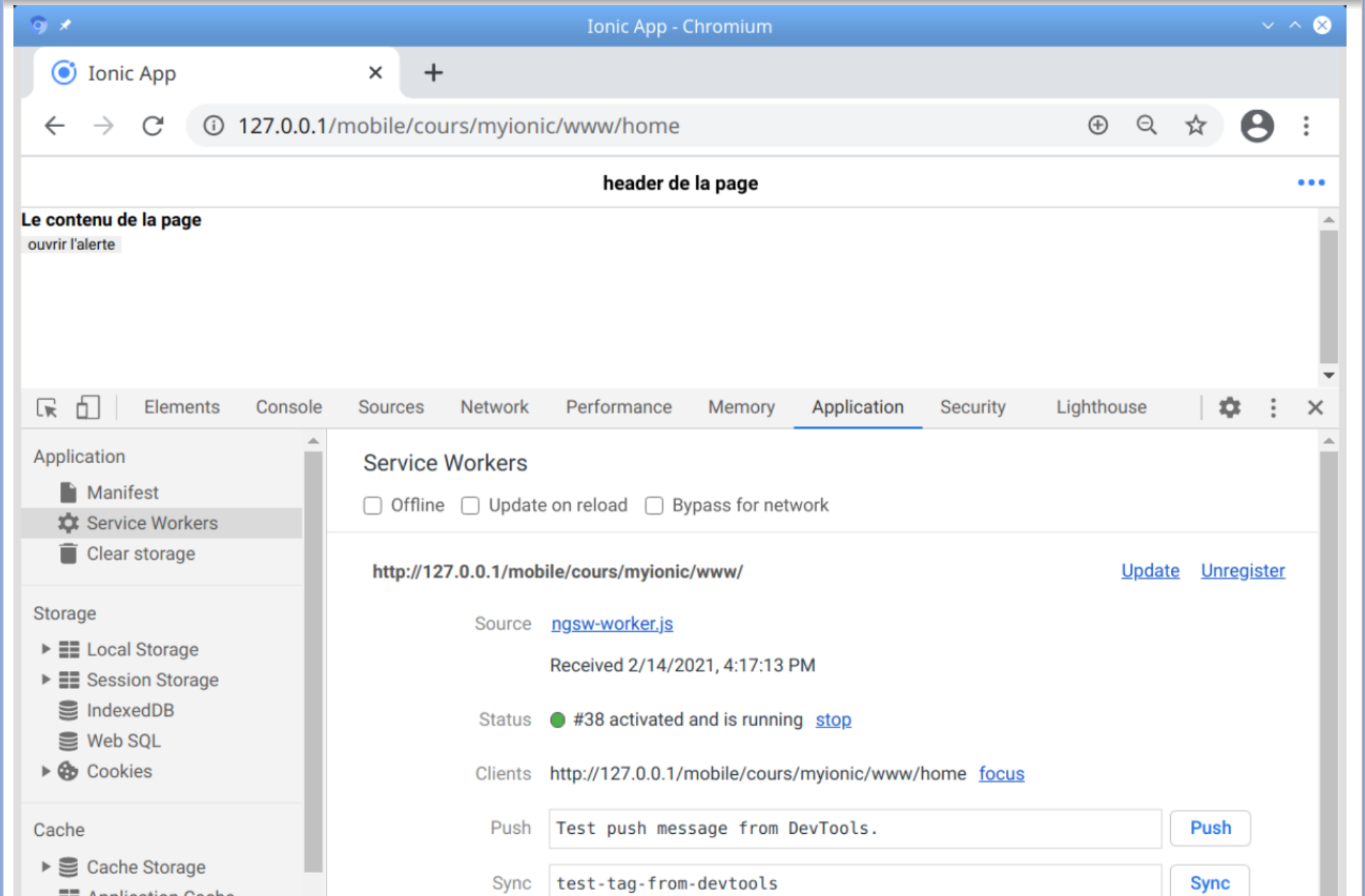
# Création d'une progressive web application (1/3)



# Création d'une progressive web application (2/3)



# Création d'une progressive web application (3/3)



# Un bouton d'installation (1/4)

myionic – home.page.ts

File Edit View Navigate Code Refactor Run Tools Git Window Help

myionic 〉 src 〉 app 〉 home 〉 home.page.ts 〉 HomePage  Angular CLI Server  Git:

home.page.ts

```
10  export class HomePage {
11    // un événement qui sera initialisé dans le constructeur si l'appli n'a pas été installée
12    promptEvent: any = null;
13
14    constructor(private popover: PopoverController,
15                private alertController: AlertController) {
16      // l'événement "beforeinstallprompt" est émis chaque fois que l'on charge la page et que
17      // l'application n'a pas encore été installée. S'il est émis, on met à jour promptEvent
18      window.addEventListener(
19        type: 'beforeinstallprompt', listener: event => {
20          this.promptEvent = event;
21        });
22    }
23
24    // quand on clique sur le bouton d'installation, on exécute cette méthode, qui appelle
25    // la méthode "prompt" du promptEvent : c'est ce qui crée le popup vous proposant
26    // d'installer l'application
27    onInstall() {
28      this.promptEvent.prompt();
29    }
30
```

Problems Git Terminal TODO Event Log

30:1 LF UTF-8 2 spaces TypeScript 4.0.5 master

# Un bouton d'installation (2/4)

myionic – home.page.html

File Edit View Navigate Code Refactor Run Tools Git Window Help

myionic 〉 src 〉 app 〉 home 〉 home.page.html — Angular CLI Server — Git:

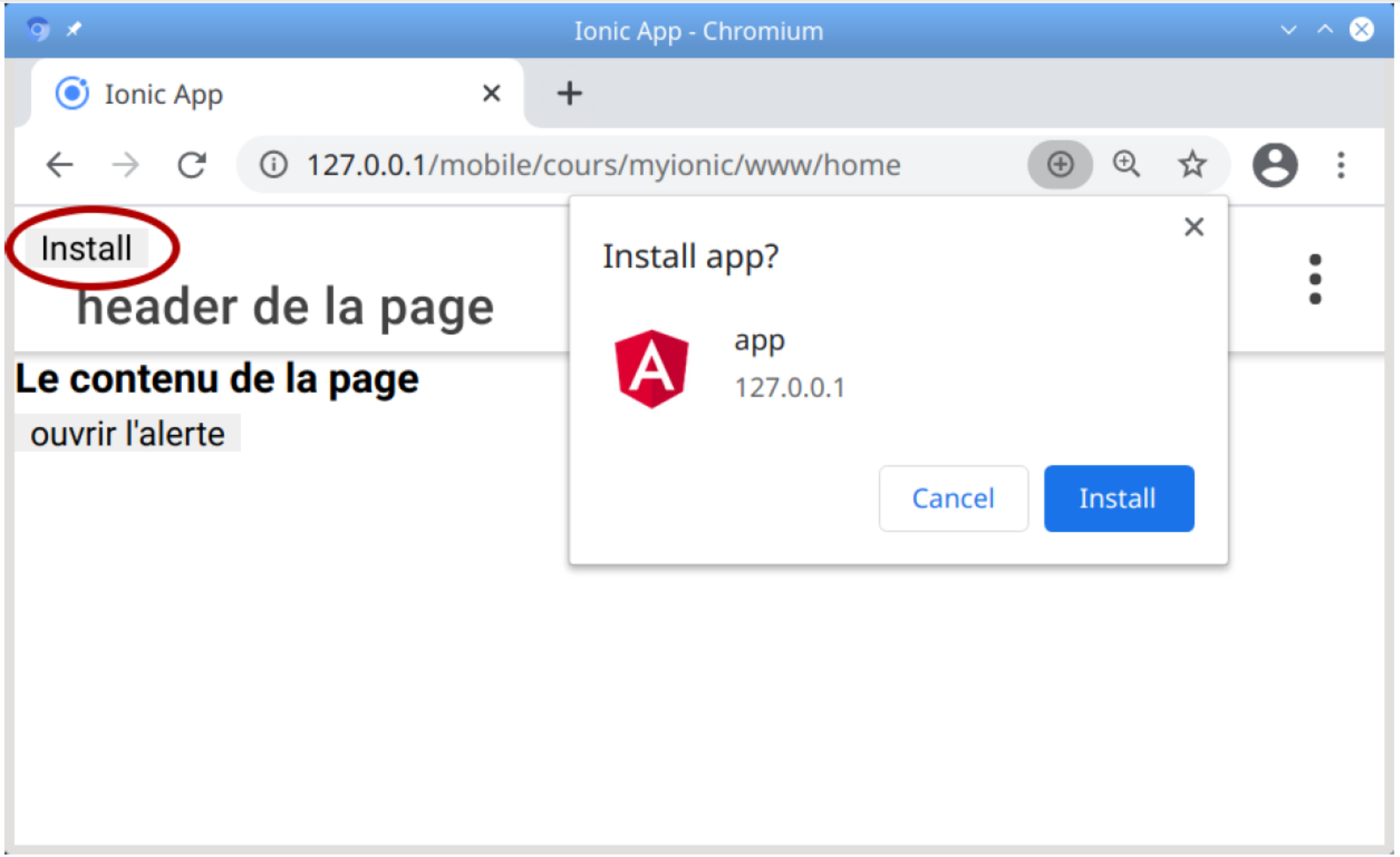
home.page.html

```
<ion-header>
  <ion-toolbar>
    <!-- on n'affiche le bouton d'installation que si l'on sait que l'application
         n'a pas déjà été installée. Si l'on clique sur le bouton, on va appeler la
         méthode onInstall de notre TypeScript, qui va ouvrir le popup d'installation -->
    <button *ngIf="promptEvent" (click)="onInstall()">Install</button>

    <ion-title>header de la page</ion-title>
    <ion-buttons slot="start">
      <ion-back-button></ion-back-button>
    </ion-buttons>
    <!-- Nouveau bouton pour pouvoir ouvrir le menu. Ce bouton a la forme d'une
         icone "...", soit verticale soit horizontale, en fonction du smartphone -->
    <ion-buttons slot="end">
      <ion-button (click)="openMenu($event)">
        <ion-icon slot="icon-only"
                  ios="ellipsis-horizontal"
                  md="ellipsis-vertical"></ion-icon>
        </ion-button>
    </ion-buttons>
  </ion-toolbar>
```

Problems Git Terminal TODO Event Log

5:90 LF UTF-8 2 spaces TypeScript 4.0.5 master

# Un bouton d'installation (3/4)



# Un bouton d'installation (4/4)