Création d'une application Web Angular + Symfony + MySQL

Objectif :

- Créer un backend en Symfony connecté à MySQL ;
- Développer un frontend en Angular 19 ;
- Connecter les deux via une API REST ;
- Échanger des données (affichage, ajout, etc.) entre le backend et le frontend.

Partie 1 — Le concept général

Une **application Web moderne** est souvent composée de **deux parties** :

Partie	Description	Technologie utilisée
Backend (serveur)	Gère les données, les enregistre dans la base MySQL et expose des API (endpoints)	Symfony
Frontend (client)	Interface visible par l'utilisateur (HTML, formulaires, listes, boutons...)	Angular 19

Partie 2 — Création du backend Symfony + MySQL

1. Configuration du projet

Créez un projet Symfony :

```
D:\CFITECH\Angular\cours angular 2025\Angular-Symfony\projet>symfony new backend --webapp
 INFO  A new Symfony CLI version is available (5.15.1, currently running 5.8.11).
```

```
D:\CFITECH\Angular\cours angular 2025\Angular-Symfony\projet>cd backend

D:\CFITECH\Angular\cours angular 2025\Angular-Symfony\projet\backend>
```

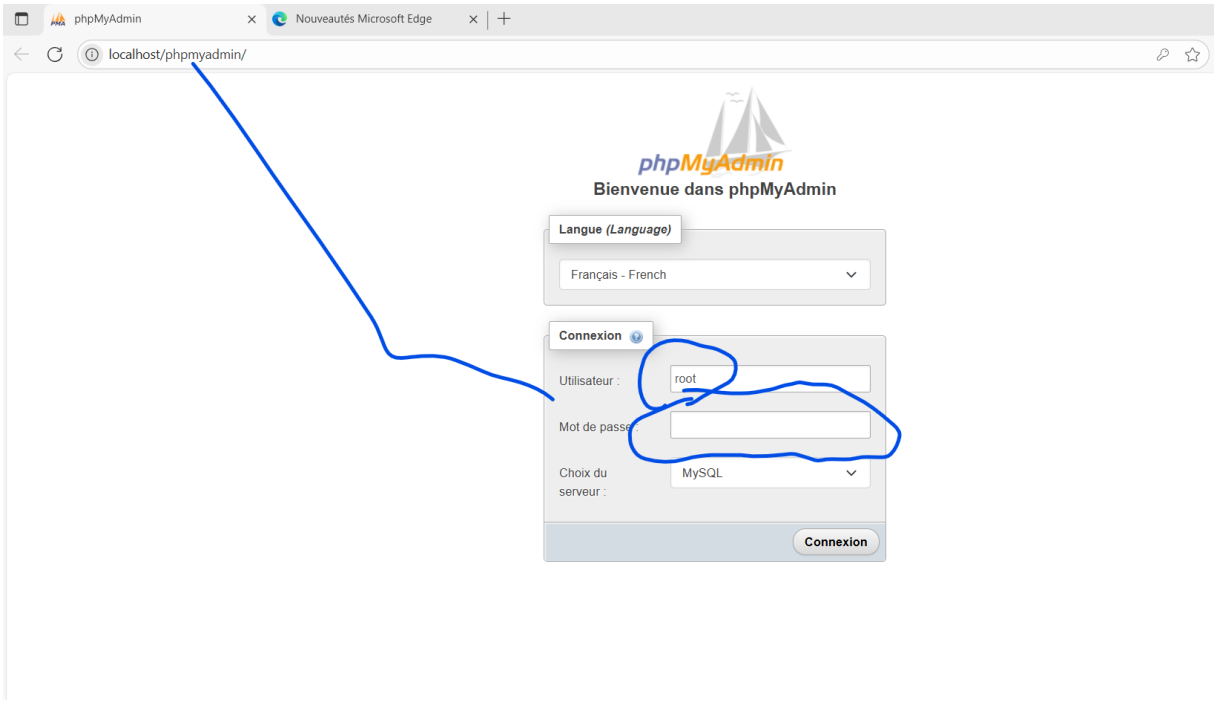
Installez la base de données MySQL :

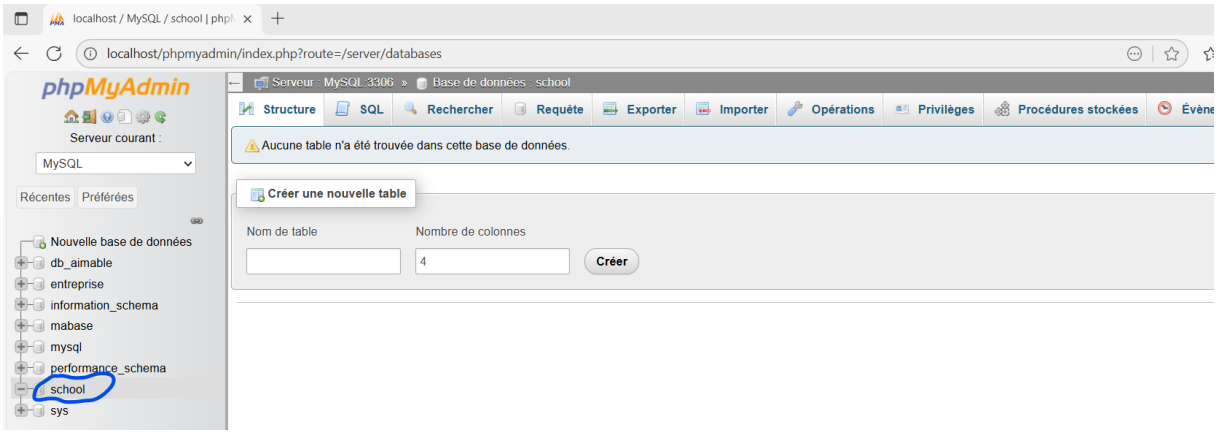
```
D:\CFITECH\Angular\cours angular 2025\Angular-Symfony\projet\backend>composer require symfony/orm-pack
./composer.json has been updated
Running composer update symfony/orm-pack
Loading composer repositories with package information
```

```
D:\CFITECH\Angular\cours angular 2025\Angular-Symfony\projet\backend>composer require symfony/maker-bundle --dev
./composer.json has been updated
Running composer update symfony/maker-bundle
Loading composer repositories with package information
Updating dependencies
```

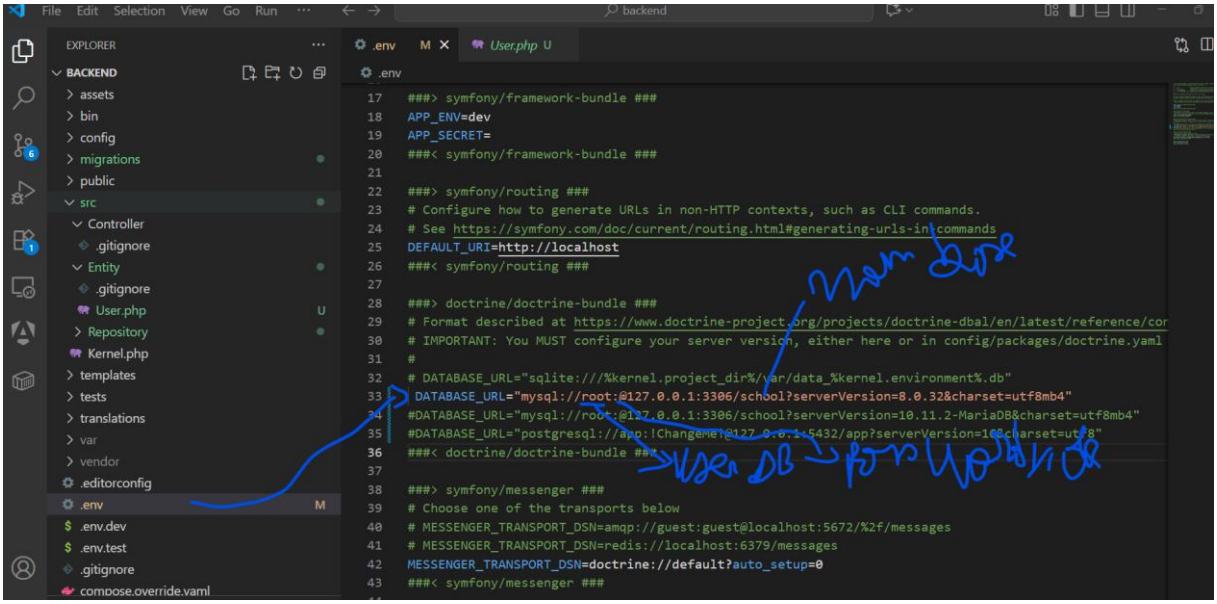
Dans le fichier .env, configurez la connexion MySQL :

Je commence par créer la base de données





Et puis dans fichier .env



2. Création de l'entité User

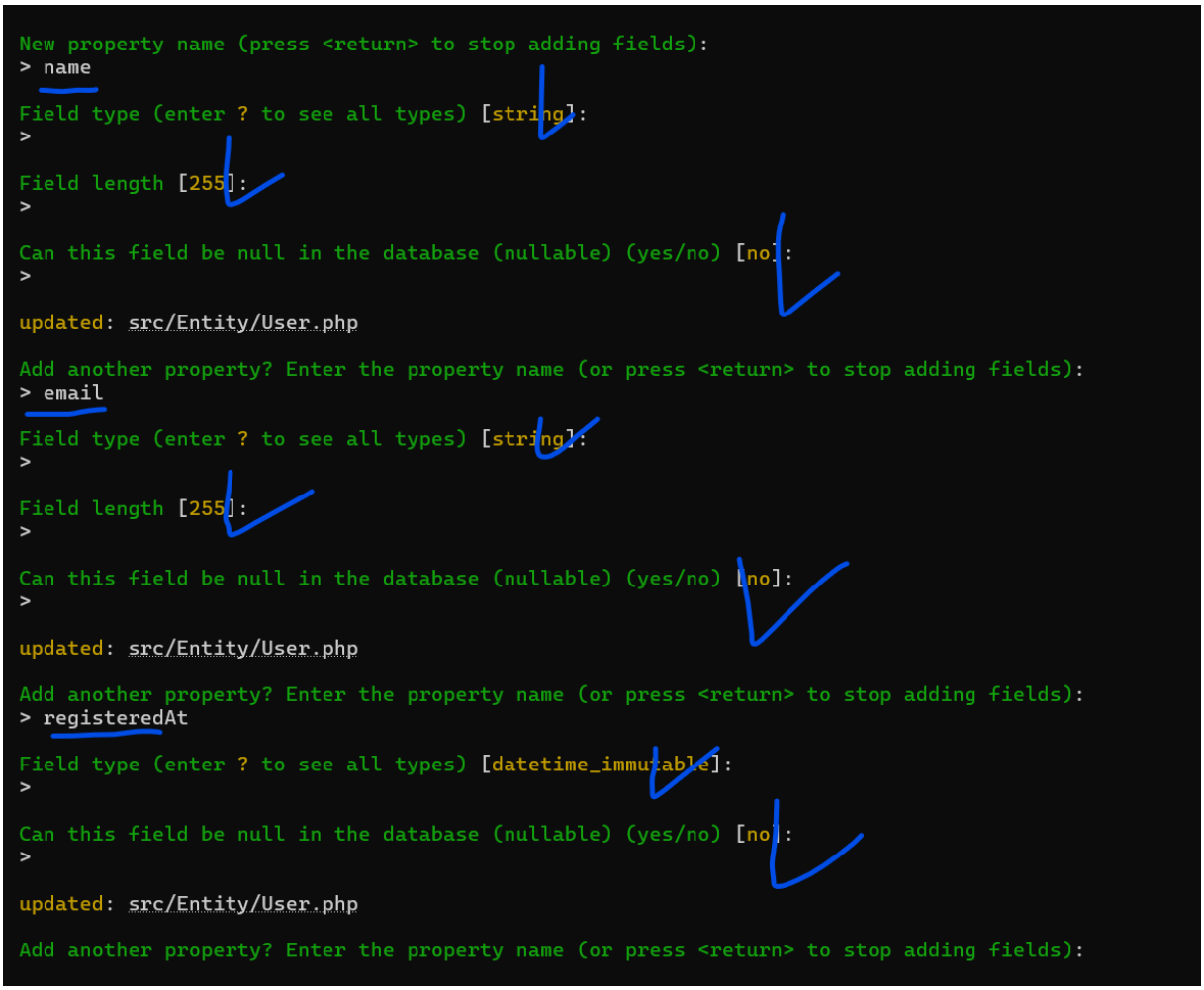
Une **entité** représente une **table** dans la base de données.

Exécutez :

```
D:\CFITECH\Angular\cours angular 2025\Angular-Symfony\projet\backend>php bin/console make:entity User
 created: src/Entity/User.php
 created: src/Repository/UserRepository.php

 Entity generated! Now let's add some fields!
 You can always add more fields later manually or by re-running this command.

 New property name (press <return> to stop adding fields):
 >
```

Ajoutez les champs :

- name (string)
- email (string)
- registeredAt (datetime)



Cela crée le fichier :

src/Entity/User.php

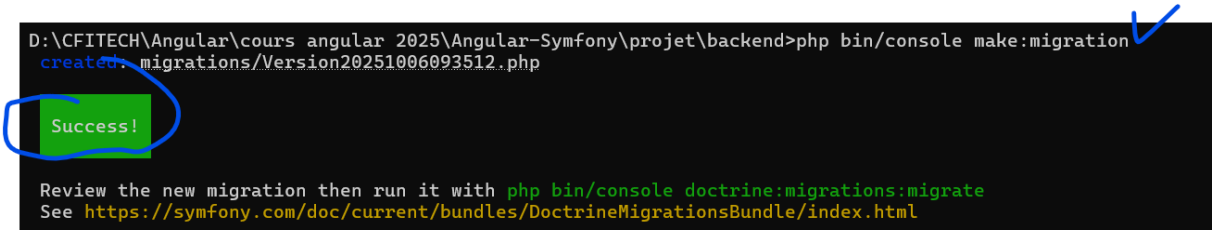
```
<?php

namespace App\Entity;

use App\Repository\UserRepository;
use Doctrine\ORM\Mapping as ORM;

#[ORM\Entity(repositoryClass: UserRepository::class)]
class User
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    private ?int $id = null;

    #[ORM\Column(length: 255)]
    private ?string $name = null;

    #[ORM\Column(length: 255)]
    private ?string $email = null;

    #[ORM\Column]
```

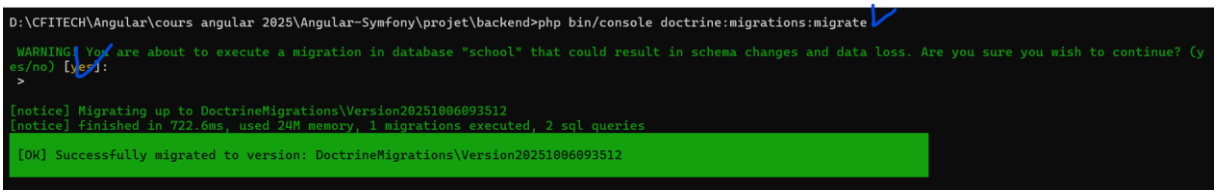
```
private ?\DateTimeImmutable $registeredAt = null;

public function getId(): ?int
{
    return $this->id;
}

public function getName(): ?string
{
    return $this->name;
}

public function setName(string $name): static
{
    $this->name = $name;

    return $this;
}

public function getEmail(): ?string
{
    return $this->email;
}

public function setEmail(string $email): static
```

```
    {
        $this->email = $email;

        return $this;
    }

    public function getRegisteredAt(): ?\DateTimeImmutable
    {
        return $this->registeredAt;
    }

    public function setRegisteredAt(\DateTimeImmutable $registeredAt): static
    {
        $this->registeredAt = $registeredAt;

        return $this;
    }
}
```

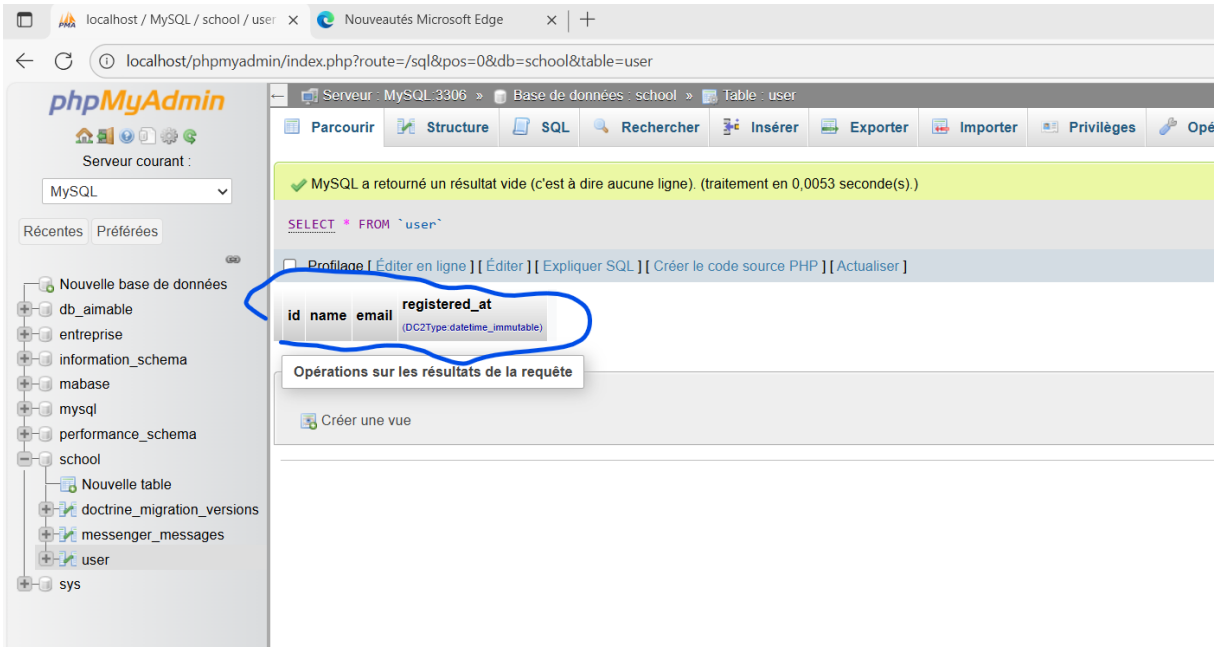
3. Migration et création de la table

Pour créer la table user dans MySQL :





Vous avez maintenant une base de données prête.

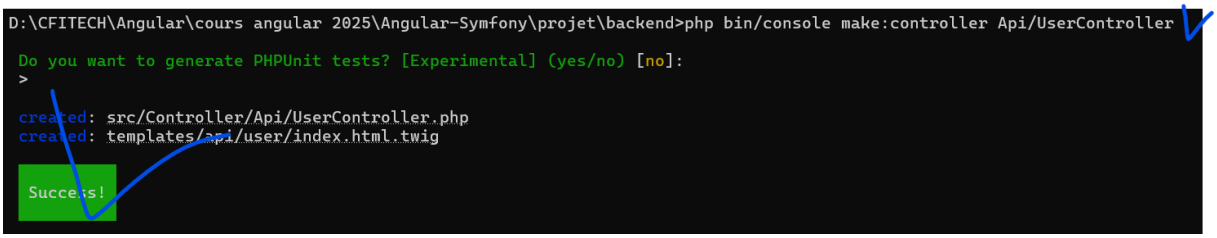


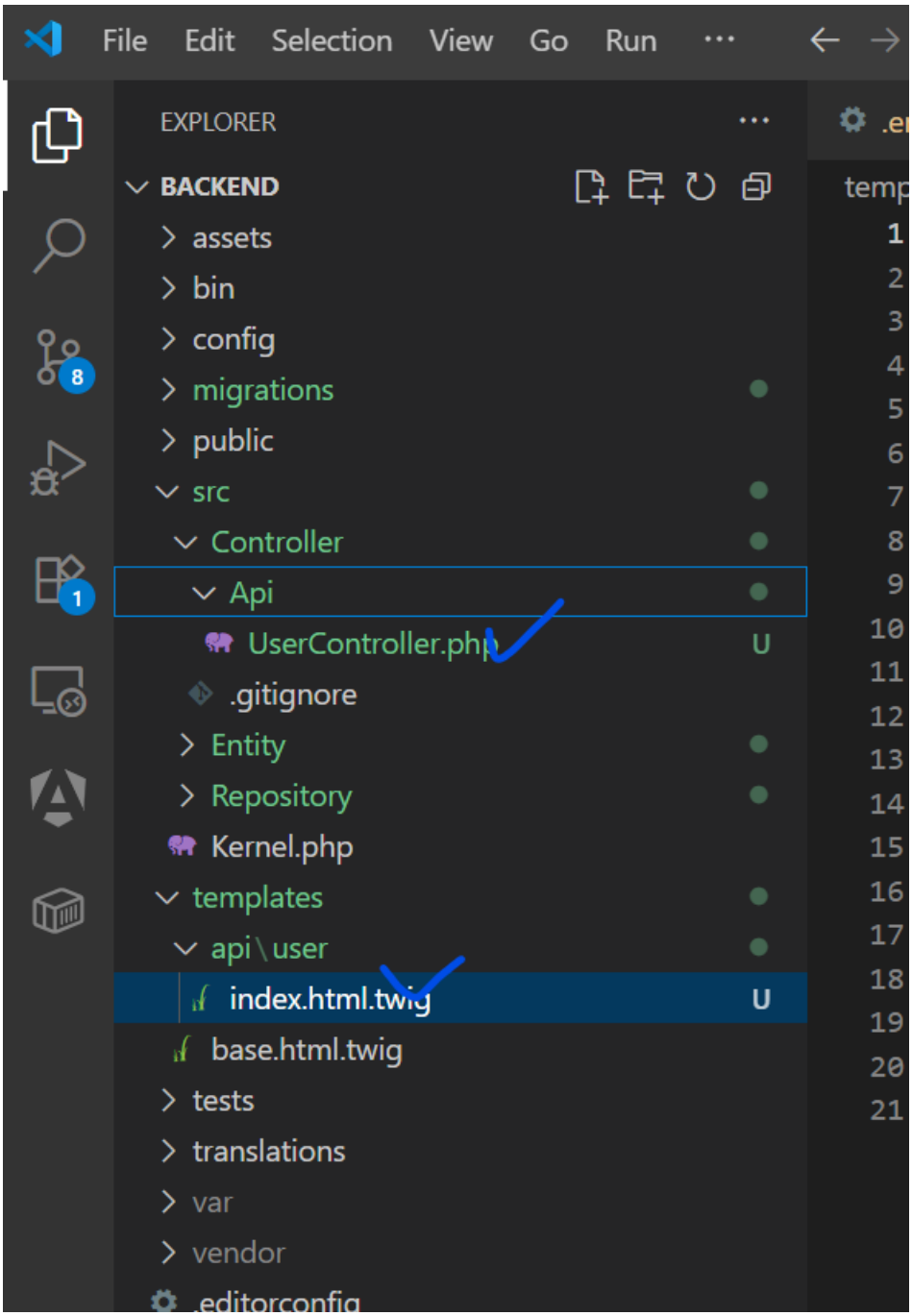
!!!Différence entre DateTime et DateTimeImmutable

Type PHP	Peut être modifié ?	Exemple
`DateTime`	✅ Oui	`$date->modify('+1 day')` change la date
`DateTimeImmutable`	❌ Non	`$date->modify('+1 day')` crée un **nouvel objet** sans changer l'original

4. Création du contrôleur API

Créez un contrôleur pour gérer les utilisateurs :




File
Edit
Selection
View
Go
Run
EXPLORER
BACKEND
assets
bin
config
migrations
public
src
Controller
Api
UserController.php
U
.gitignore
Entity
Repository
Kernel.php
templates
api\user
index.html.twig
U
base.html.twig
tests
translations
var
vendor
.editorconfig
temp
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21

Code du contrôleur src/Controller/Api/UserController.php :

```
<?php
namespace App\Controller\Api;

use App\Entity\User;
use App\Repository\UserRepository;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\JsonResponse;
use Symfony\Component\Routing\Annotation\Route;

#[Route('/api/users')]
class UserController extends AbstractController
{
    #[Route('', methods: ['GET'])]
    public function index(UserRepository $repo): JsonResponse
    {
        $users = $repo->findAll();
        $data = array_map(fn(User $u) => [
            'id' => $u->getId(),
            'name' => $u->getName(),
            'email' => $u->getEmail(),
            'registeredAt' => $u->getRegisteredAt()->format('Y-m-d'),
        ], $users);
```

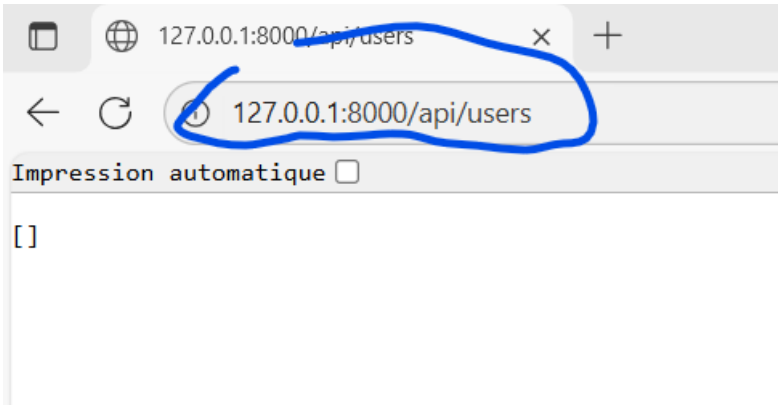
```
    return $this->json($data);
}

#[Route('', methods: ['POST'])]
public function create(Request $request, EntityManagerInterface $em): JsonResponse
{
    $body = json_decode($request->getContent(), true);

    if (!isset($body['name'], $body['email'])) {
        return $this->json(['error' => 'Données invalides'], 400);
    }

    $user = new User();
    $user->setName($body['name']);
    $user->setEmail($body['email']);
    $user->setRegisteredAt(new \DateTimeImmutable());

    $em->persist($user);
    $em->flush();

    return $this->json([
        'id' => $user->getId(),
        'name' => $user->getName(),
        'email' => $user->getEmail(),
```

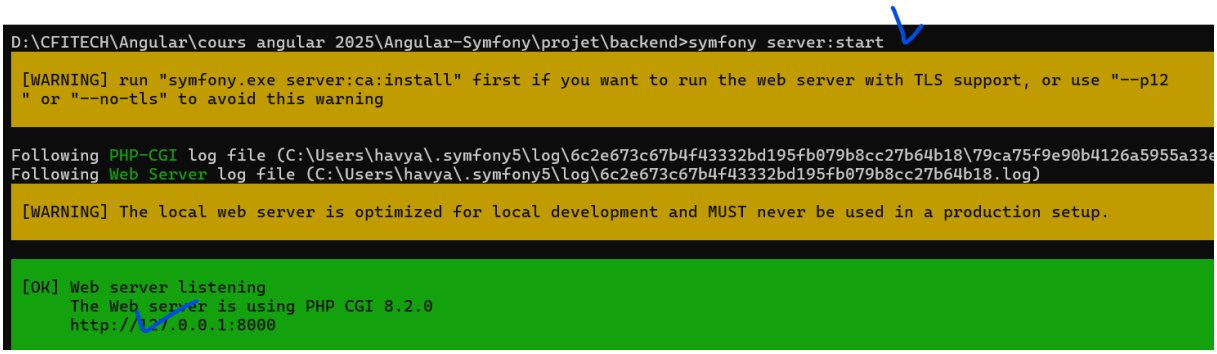
```
            'registeredAt' => $user->getRegisteredAt()->format('Y-m-d'),
        ], 201);
    }
}
```

Le backend expose maintenant deux routes :

- GET /api/users → liste des utilisateurs
- POST /api/users → ajout d'un utilisateur

Lance le serveur Symfony





Partie 3 — Création du frontend Angular 19

1. Création du projet Angular

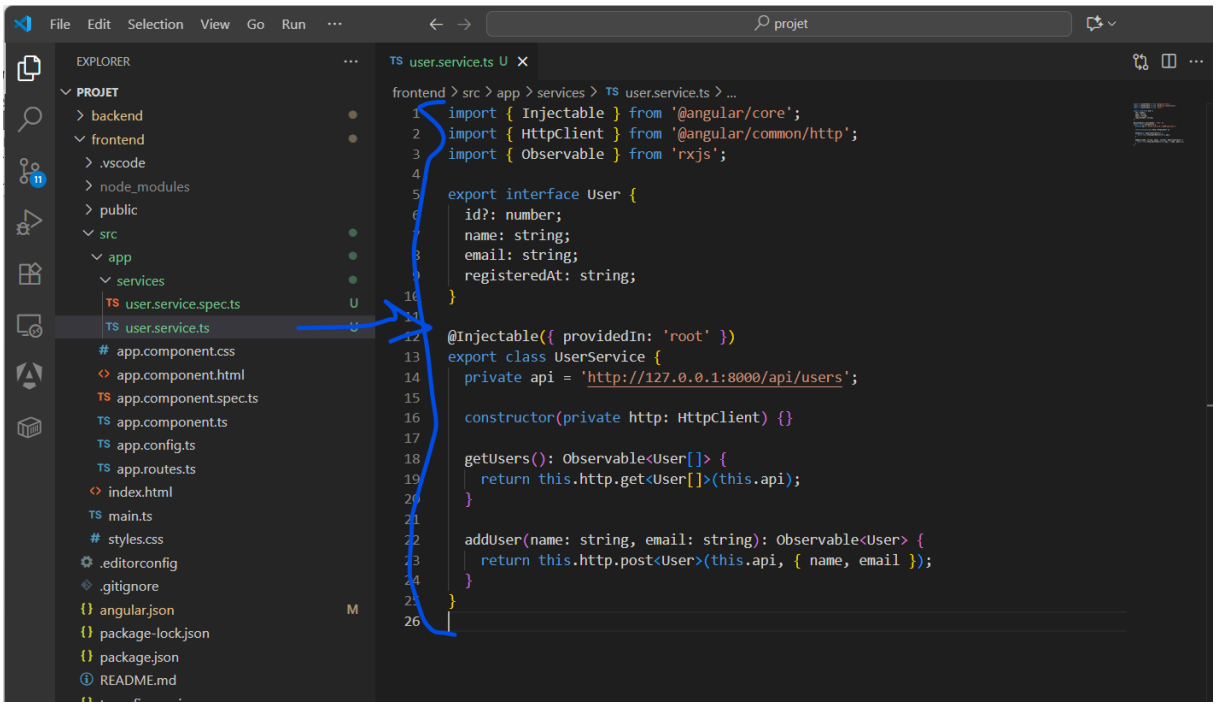
```
D:\CFITECH\Angular\cours angular 2025\Angular-Symfony\projet>ng new frontend --standalone
√ Which stylesheet format would you like to use? CSS                [ https://developer.mozilla.org/docs/Web/CSS
√ Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? No
CREATE frontend/angular.json (2694 bytes)
CREATE frontend/package.json (1040 bytes)
CREATE frontend/README.md (1531 bytes)
CREATE frontend/tsconfig.json (942 bytes)
CREATE frontend/.editorconfig (331 bytes)
CREATE frontend/.gitignore (629 bytes)
CREATE frontend/tsconfig.app.json (439 bytes)
CREATE frontend/tsconfig.spec.json (449 bytes)
CREATE frontend/.vscode/extensions.json (134 bytes)
CREATE frontend/.vscode/launch.json (490 bytes)
CREATE frontend/.vscode/tasks.json (980 bytes)
CREATE frontend/src/main.ts (256 bytes)
CREATE frontend/src/index.html (307 bytes)
CREATE frontend/src/styles.css (81 bytes)
CREATE frontend/src/app/app.component.html (20239 bytes)
CREATE frontend/src/app/app.component.spec.ts (951 bytes)
CREATE frontend/src/app/app.component.ts (296 bytes)
CREATE frontend/src/app/app.component.css (0 bytes)
CREATE frontend/src/app/app.config.ts (318 bytes)
CREATE frontend/src/app/app.routes.ts (80 bytes)
CREATE frontend/public/favicon.ico (15086 bytes)
√ Packages installed successfully.
    Successfully initialized git.

D:\CFITECH\Angular\cours angular 2025\Angular-Symfony\projet>cd frontend

D:\CFITECH\Angular\cours angular 2025\Angular-Symfony\projet\frontend>
```

```
D:\CFITECH\Angular\cours angular 2025\Angular-Symfony\projet>cd frontend

D:\CFITECH\Angular\cours angular 2025\Angular-Symfony\projet\frontend>code .

D:\CFITECH\Angular\cours angular 2025\Angular-Symfony\projet\frontend>ng serve
```

2. Service Angular UserService

```
D:\CFITECH\Angular\cours angular 2025\Angular-Symfony\projet\frontend>ng g s services/user

Would you like to share pseudonymous usage data about this project with the Angular Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details and how to change this setting, see https://angular.dev/cli/analytics.

   Yes

Thank you for sharing pseudonymous usage data. Should you change your mind, the following
command will disable this feature entirely:

    ng analytics disable

Global setting: enabled
Local setting: enabled
Effective status: enabled
CREATE src/app/services/user.service.spec.ts (363 bytes)
CREATE src/app/services/user.service.ts (142 bytes)
```

Code :



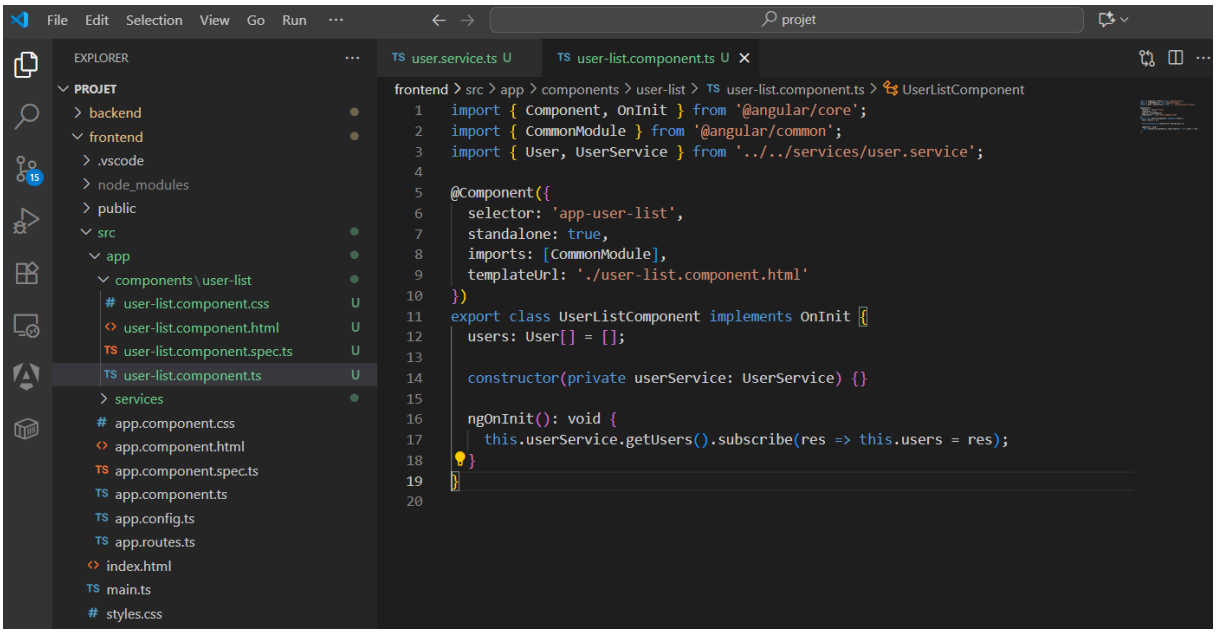

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';


export interface User {
 id?: number;
 name: string;
 email: string;
 registeredAt: string;
}
```

```
@Injectable({ providedIn: 'root' })
export class UserService {
  private api = 'http://127.0.0.1:8000/api/users';

  constructor(private http: HttpClient) {}

  getUsers(): Observable<User[]> {
    return this.http.get<User[]>(this.api);
  }

  addUser(name: string, email: string): Observable<User> {
    return this.http.post<User>(this.api, { name, email });
  }
}
```

3. **Liste des utilisateurs**

```
D:\CFITECH\Angular\cours angular 2025\Angular-Symfony\projet\frontend>ng generate component components/user-list --standalone
CREATE src/app/components/user-list/user-list.component.html (25 bytes)
CREATE src/app/components/user-list/user-list.component.spec.ts (630 bytes)
CREATE src/app/components/user-list/user-list.component.ts (236 bytes)
CREATE src/app/components/user-list/user-list.component.css (0 bytes)
```

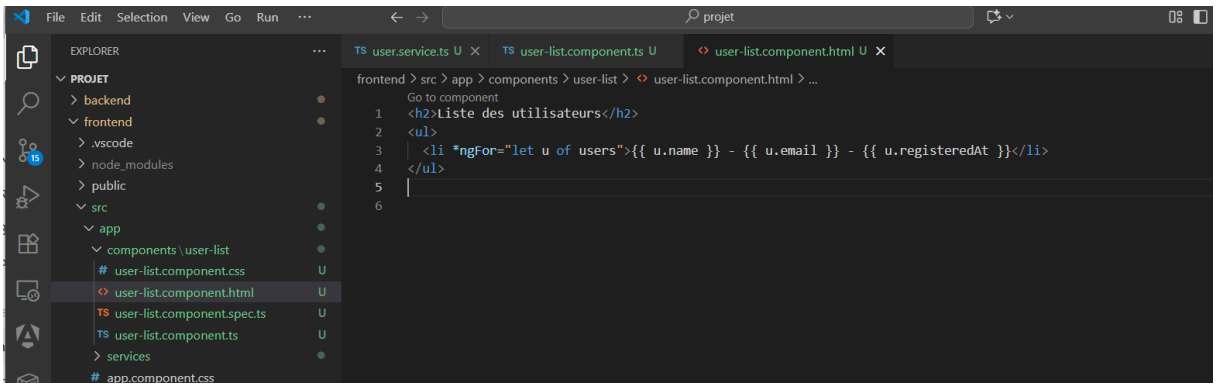
Ts :




```
import { Component, OnInit } from '@angular/core';
import { CommonModule } from '@angular/common';
import { User, UserService } from '../../services/user.service';

@Component({
 selector: 'app-user-list',
 standalone: true,
 imports: [CommonModule],
 templateUrl: './user-list.component.html'
})
export class UserListComponent implements OnInit {
 users: User[] = [];

 constructor(private userService: UserService) {}

 ngOnInit(): void {
```

```
    this.userService.getUsers().subscribe(res => this.users = res);
  }
}
```

Html :

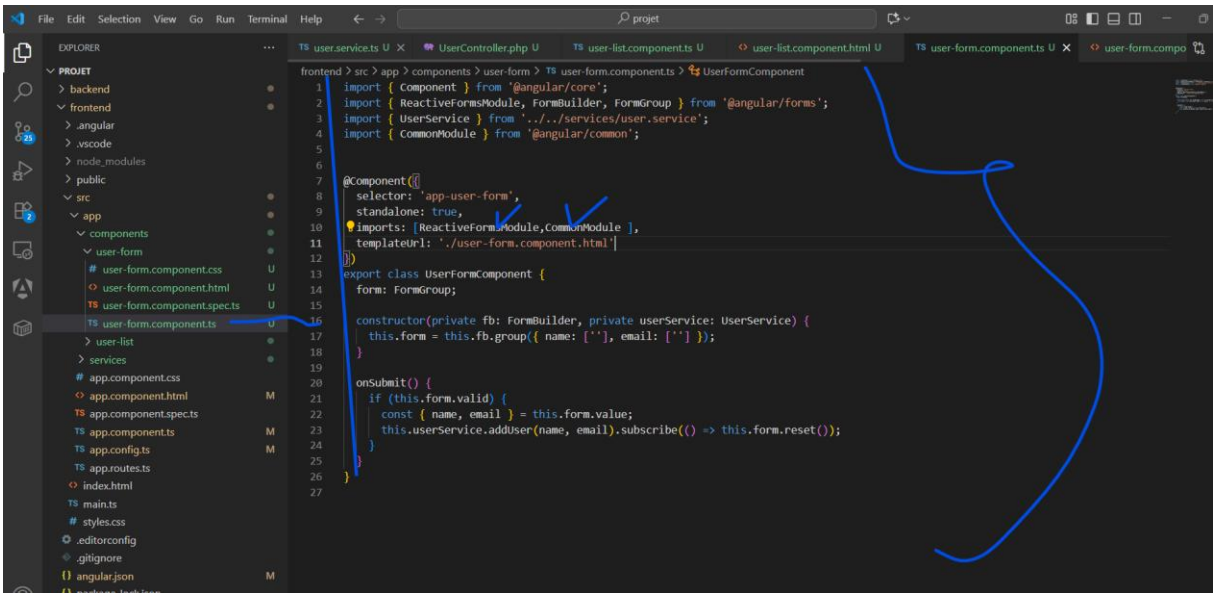



```
<h2>Liste des utilisateurs</h2>
<ul>
  <li *ngFor="let u of users">{{ u.name }} - {{ u.email }} - {{ u.registeredAt }}</li>
</ul>
```

4. Formulaire d’ajout

```
D:\CFITECH\Angular\cours angular 2025\Angular-Symfony\projet\frontend>ng generate component components/user-form --standalone
CREATE src/app/components/user-form/user-form.component.html (25 bytes)
CREATE src/app/components/user-form/user-form.component.spec.ts (630 bytes)
CREATE src/app/components/user-form/user-form.component.ts (236 bytes)
CREATE src/app/components/user-form/user-form.component.css (0 bytes)
```

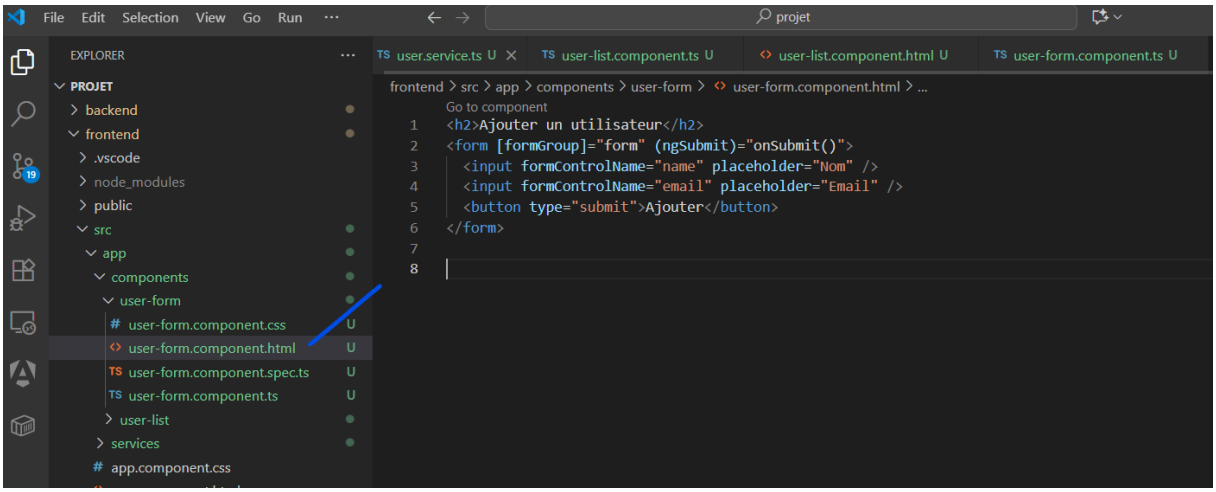
Ts :




```
import { Component } from '@angular/core';
import { ReactiveFormsModule, FormBuilder, FormGroup } from '@angular/forms';
import { UserService } from '../../services/user.service';
import { CommonModule } from '@angular/common';

@Component({
  selector: 'app-user-form',
  standalone: true,
  imports: [ReactiveFormsModule,CommonModule ],
  templateUrl: './user-form.component.html'
})
export class UserFormComponent {
  form: FormGroup;

  constructor(private fb: FormBuilder, private userService: UserService) {
    this.form = this.fb.group({ name: [''], email: [''] });
```

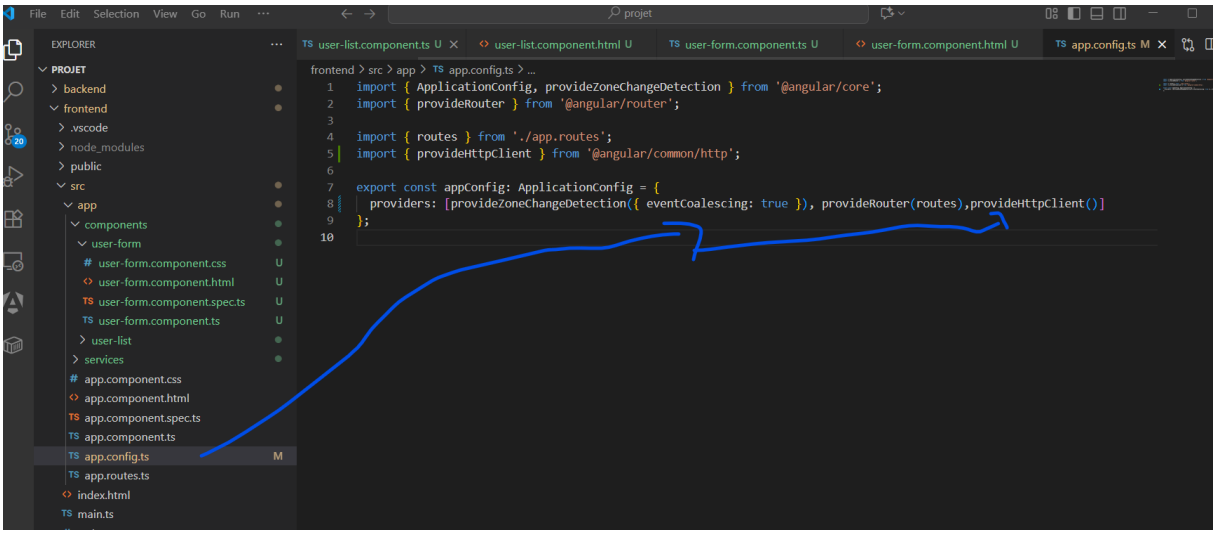
```
  }

  onSubmit() {
    if (this.form.valid) {
      const { name, email } = this.form.value;
      this.userService.addUser(name, email).subscribe(() => this.form.reset());
    }
  }
}
```

Html :

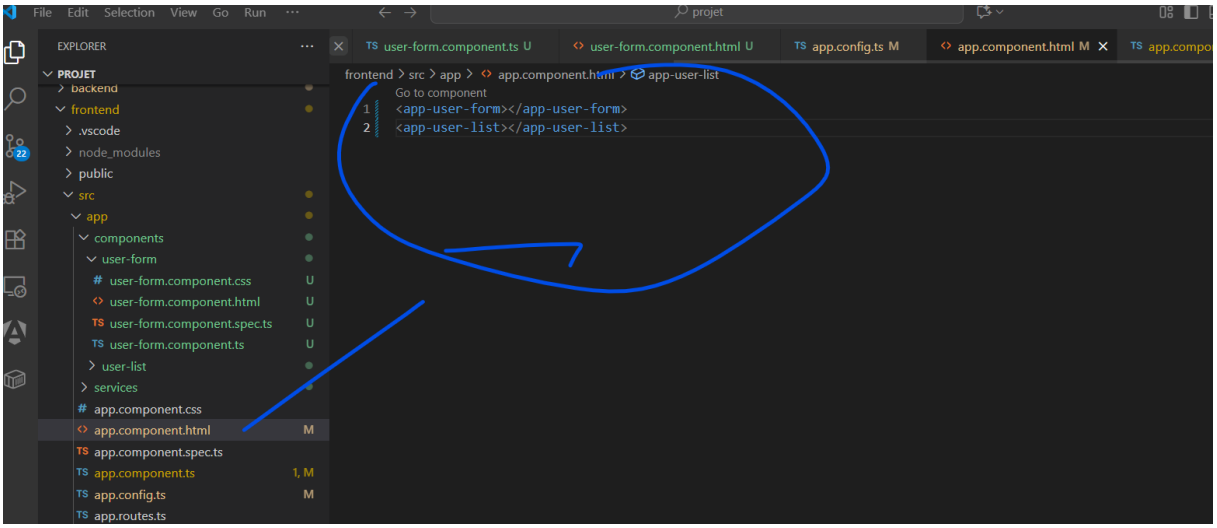



```
<h2>Ajouter un utilisateur</h2>
<form [formGroup]="form" (ngSubmit)="onSubmit()">
  <input formControlName="name" placeholder="Nom" />
  <input formControlName="email" placeholder="Email" />
  <button type="submit">Ajouter</button>
</form>
```

Et dans app.confi.ts :



Dans appComponent :



Si ça n'affiche pas il faut d'abord autorise Angular à accéder à Symfony (CORS)

Par défaut, Symfony bloque les requêtes venant d'un autre port (par ex. : Angular = 4200, Symfony = 8000).

Installe le CORS Bundle :

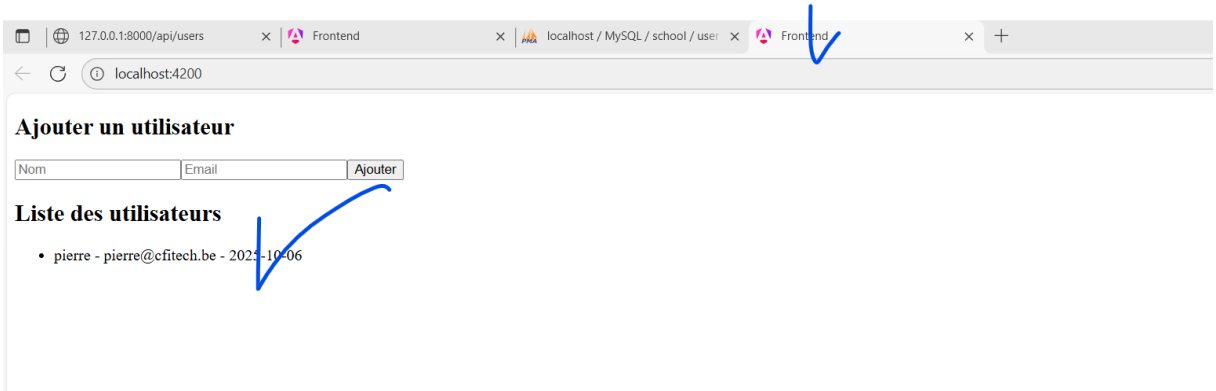
```
D:\CFITECH\Angular\cours angular 2025\Angular-Symfony\projet\backend>composer require nelmio/cors-bundle
./composer.json has been updated
Running composer update nelmio/cors-bundle
Loading composer repositories with package information
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
  - Locking nelmio/cors-bundle (2.5.0)
```

Redémarrer le sereveur symfony :

```
D:\CFITECH\Angular\cours angular 2025\Angular-Symfony\projet\backend>symfony server:start

[WARNING] run "symfony.exe server:ca:install" first if you want to run the web server with TLS support, or use "--p12
" or "--no-tls" to avoid this warning

Following PHP-CGI log file (C:\Users\havya\.symfony5\log\6c2e673c67b4f43332bd195fb079b8cc27b64b18\79ca75f9e90b4126a5955a33ea6a41e
Following Web Server log file (C:\Users\havya\.symfony5\log\6c2e673c67b4f43332bd195fb079b8cc27b64b18.log)

[WARNING] The local web server is optimized for local development and MUST never be used in a production setup.

[OK] Web server listening
     The Web server is using PHP CGI 8.2.0
     http://127.0.0.1:8000
```

Relancer le serveur angular :

```
D:\CFITECH\Angular\cours angular 2025\Angular-Symfony\projet\frontend>ng serve --open
Component HMR has been enabled, see https://angular.dev/hmr for more info.
Initial chunk files | Names         |  Raw size
polyfills.js        | polyfills     |  89.77 kB |
main.js             | main          |  11.53 kB |
styles.css          | styles        |  95 bytes |

                    | Initial total | 101.39 kB

Application bundle generation complete. [2.684 seconds]

▲ [WARNING] TS-998113: RouterOutlet is not used within the template of AppComponent [plugin angular-compiler]

    src/app/app.component.ts:8:12:
      8 |    imports: [RouterOutlet, UserFormComponent,UserListComponent],
                       ~~~~~~~~~~~~~~


Watch mode enabled. Watching for file changes...
NOTE: Raw file sizes do not reflect development server per-request transformations.
  ➜  Local:   http://localhost:4200/
  ➜  press h + enter to show help
```



Mise à jour d’un utilisateur (Angular + Symfony)

Mettre à jour les informations d’un utilisateur existant dans une base de données MySQL via une API Symfony et une interface Angular.

1. Principe général

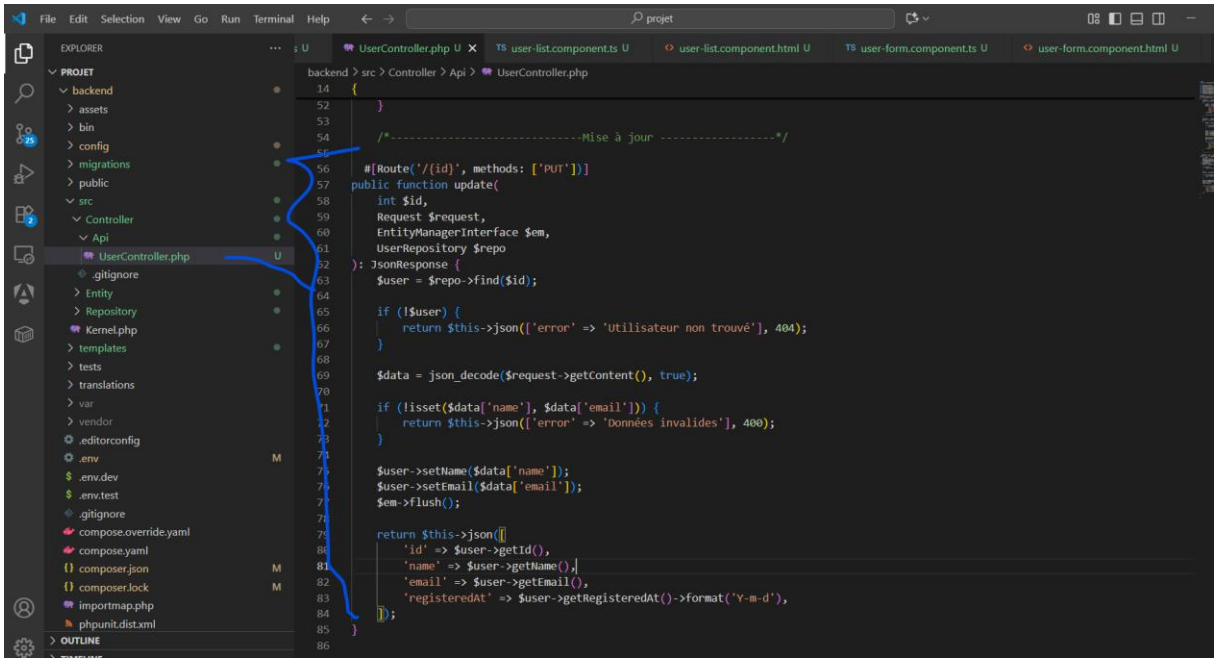
L’application est divisée en deux parties

Backend Symfony	Fournit une API REST (avec des routes `GET`, `POST`, `PUT`, `DELETE`...) qui communique avec la base MySQL.
Frontend Angular	Affiche les utilisateurs et envoie les données au backend pour les créer, les modifier ou les supprimer.

2. Côté backend (Symfony)

Route de mise à jour

On ajoute une nouvelle route dans le contrôleur UserController.php :



<?php

namespace App\Controller\Api;

use App\Entity\User;

```
use App\Repository\UserRepository;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\JsonResponse;
use Symfony\Component\Routing\Annotation\Route;

#[Route('/api/users')]
class UserController extends AbstractController
{
    #[Route('', methods: ['GET'])]
    public function index(UserRepository $repo): JsonResponse
    {
        $users = $repo->findAll();
        $data = array_map(fn(User $u) => [
            'id' => $u->getId(),
            'name' => $u->getName(),
            'email' => $u->getEmail(),
            'registeredAt' => $u->getRegisteredAt()->format('Y-m-d'),
        ], $users);

        return $this->json($data);
    }

    #[Route('', methods: ['POST'])]
```

```
    public function create(Request $request, EntityManagerInterface $em): JsonResponse
    {
        $body = json_decode($request->getContent(), true);

        if (!isset($body['name'], $body['email'])) {
            return $this->json(['error' => 'Données invalides'], 400);
        }

        $user = new User();
        $user->setName($body['name']);
        $user->setEmail($body['email']);
        $user->setRegisteredAt(new \DateTimeImmutable());

        $em->persist($user);
        $em->flush();

        return $this->json([
            'id' => $user->getId(),
            'name' => $user->getName(),
            'email' => $user->getEmail(),
            'registeredAt' => $user->getRegisteredAt()->format('Y-m-d'),
        ], 201);
    }

    /*-----------------------------Mise à jour ------------------*/
```

```
  #[Route('/{id}', methods: ['PUT'])]
public function update(
    int $id,
    Request $request,
    EntityManagerInterface $em,
    UserRepository $repo
): JsonResponse {
    $user = $repo->find($id);

    if (!$user) {
        return $this->json(['error' => 'Utilisateur non trouvé'], 404);
    }

    $data = json_decode($request->getContent(), true);

    if (!isset($data['name'], $data['email'])) {
        return $this->json(['error' => 'Données invalides'], 400);
    }

    $user->setName($data['name']);
    $user->setEmail($data['email']);
    $em->flush();

    return $this->json([
```

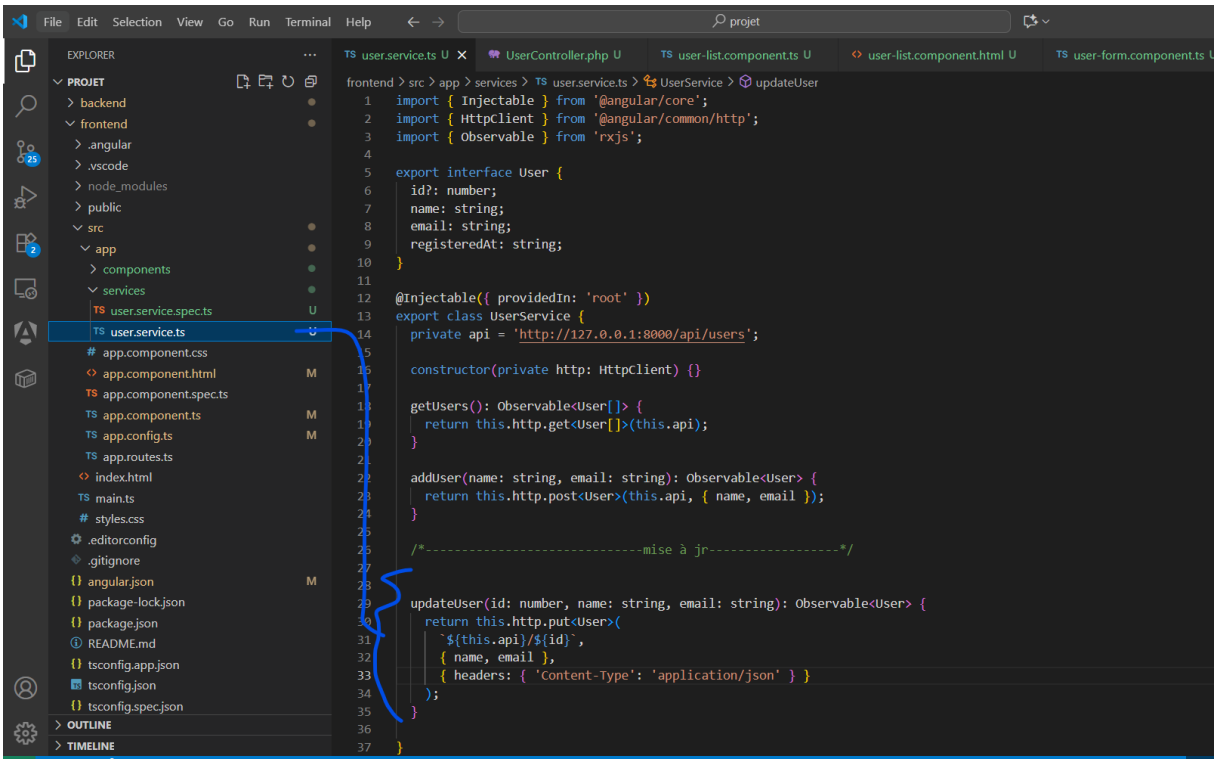
```
        'id' => $user->getId(),
        'name' => $user->getName(),
        'email' => $user->getEmail(),
        'registeredAt' => $user->getRegisteredAt()->format('Y-m-d'),
    ]);
}


}
```

- **Méthode HTTP :** PUT (pour une mise à jour complète).
- **Paramètre dynamique {id} :** permet de cibler l'utilisateur à modifier.
- **json_decode()** : lit les données envoyées par Angular.
- **flush()** : applique les changements dans la base.
- **Retour JSON** : confirme la mise à jour à Angular.

3. Côté frontend (Angular)

a) Méthode du service

Dans user.service.ts, on ajoute :



```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

export interface User {
  id?: number;
  name: string;
  email: string;
  registeredAt: string;
}

@Injectable({ providedIn: 'root' })
export class UserService {
  private api = 'http://127.0.0.1:8000/api/users';

  constructor(private http: HttpClient) {}

  getUsers(): Observable<User[]> {
    return this.http.get<User[]>(this.api);
  }

  addUser(name: string, email: string): Observable<User> {
    return this.http.post<User>(this.api, { name, email });
  }
```

```
 /*------------------------------mise à jr------------------*/



  updateUser(id: number, name: string, email: string): Observable<User> {

    return this.http.put<User>(

      `${this.api}/${id}`,

      { name, email },

      { headers: { 'Content-Type': 'application/json' } }

    );

  }



}
```

Cette méthode envoie une requête HTTP **PUT** à Symfony, avec :

- l'adresse /api/users/{id},
- les nouvelles valeurs à enregistrer (name et email).

b) Intégration dans le composant

Dans le composant Angular (user-form.component.ts ou user-list.component.ts), on ajoute la logique suivante :

```
// src/app/components/user-form/user-form.component.ts

import { Component, OnInit } from '@angular/core';

import { ReactiveFormsModule, FormBuilder, FormGroup } from '@angular/forms';

import { UserService, User } from '../../services/user.service';

import { CommonModule } from '@angular/common';
```

```
@Component({
  selector: 'app-user-form',
  standalone: true,
  imports: [ReactiveFormsModule, CommonModule],
  templateUrl: './user-form.component.html'
})
export class UserFormComponent implements OnInit {
  form!: FormGroup;
  users: User[] = [];
  editingUser: User | null = null;

  constructor(private fb: FormBuilder, private userService: UserService) {}

  ngOnInit(): void {
    this.form = this.fb.group({
      name: [''],
      email: ['']
    });
    this.loadUsers();
  }

  loadUsers() {
    this.userService.getUsers().subscribe({
      next: (data) => (this.users = data),
      error: (err) => {
```

```
      console.error('Erreur getUsers', err);
      alert('Impossible de charger la liste des utilisateurs (voir la console).');
    }
  });
}

startEdit(user: User) {
  this.editingUser = user;
  this.form.patchValue({ name: user.name, email: user.email });
}

cancelEdit() {
  this.editingUser = null;
  this.form.reset();
}

onSubmit() {
  if (!this.form.valid) {
    return;
  }
  const { name, email } = this.form.value;

  if (this.editingUser) {
    // mode édition
    this.userService.updateUser(this.editingUser.id!, name, email).subscribe({
```

```
        next: () => {
          this.loadUsers();
          this.cancelEdit();
        },
        error: (err) => {
          console.error('Erreur updateUser', err);
          alert('Erreur lors de la mise à jour (voir la console).');
        }
      });
    } else {
      // mode ajout
      this.userService.addUser(name, email).subscribe({
        next: () => {
          this.loadUsers();
          this.form.reset();
        },
        error: (err) => {
          console.error('Erreur addUser', err);
          alert('Erreur lors de l\'ajout (voir la console).');
        }
      });
    }
  }
}
```

Html :

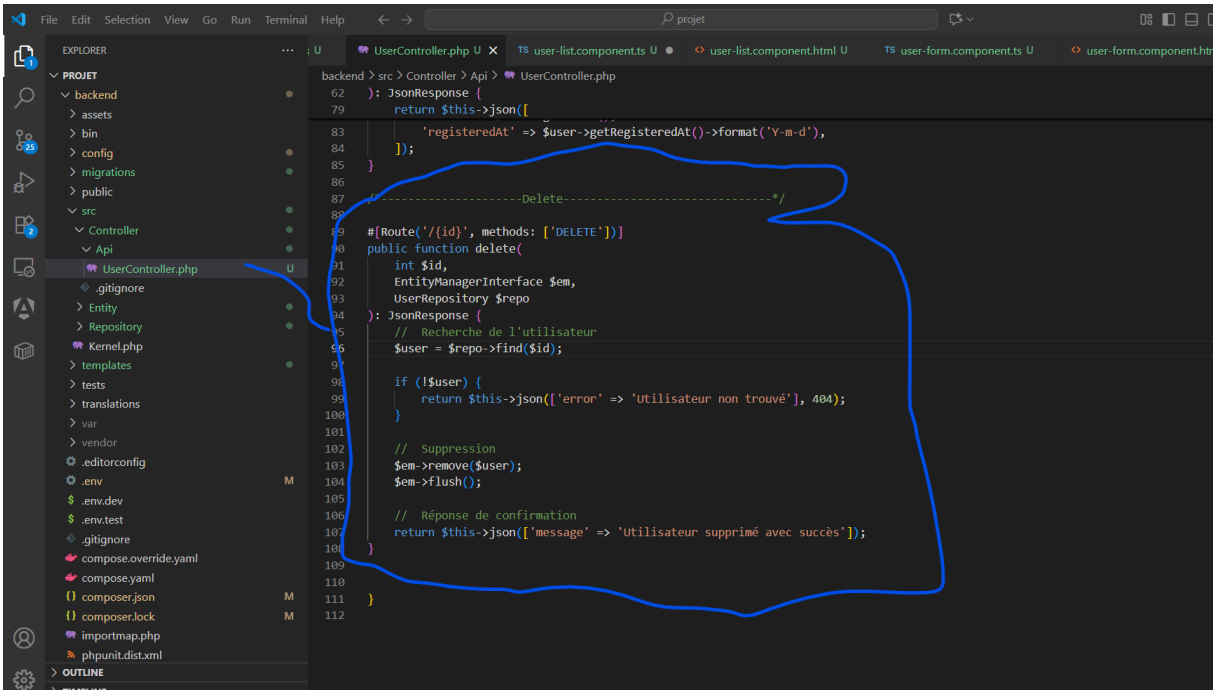
```
<!-- src/app/components/user-form/user-form.component.html -->
<h2>{{ editingUser ? 'Modifier l\'utilisateur' : 'Ajouter un utilisateur' }}</h2>

<form [formGroup]="form" (ngSubmit)="onSubmit()">
  <input formControlName="name" placeholder="Nom" />
  <input formControlName="email" placeholder="Email" />
  <button type="submit">{{ editingUser ? 'Mettre à jour' : 'Ajouter' }}</button>
  <button type="button" *ngIf="editingUser" (click)="cancelEdit()">Annuler</button>
</form>

<hr />

<h3>Liste des utilisateurs</h3>
<ul>
  <li *ngFor="let user of users">
    {{ user.name }} ({{ user.email }})
    <button (click)="startEdit(user)">Modifier</button>
  </li>
</ul>
```

Suppression d’un utilisateur depuis le formulaire (UserFormComponent)

Permettre à l’utilisateur de supprimer la fiche affichée dans le formulaire (UserFormComponent)

1. Côté backend (Symfony)

On ajouté la route suivante :



<?php

namespace App\Controller\Api;

use App\Entity\User;

use App\Repository\UserRepository;

use Doctrine\ORM\EntityManagerInterface;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;

use Symfony\Component\HttpFoundation\Request;

use Symfony\Component\HttpFoundation\JsonResponse;

```
use Symfony\Component\Routing\Annotation\Route;


#[Route('/api/users')]
class UserController extends AbstractController
{
    #[Route('', methods: ['GET'])]
    public function index(UserRepository $repo): JsonResponse
    {
        $users = $repo->findAll();
        $data = array_map(fn(User $u) => [
            'id' => $u->getId(),
            'name' => $u->getName(),
            'email' => $u->getEmail(),
            'registeredAt' => $u->getRegisteredAt()->format('Y-m-d'),
        ], $users);

        return $this->json($data);
    }

    #[Route('', methods: ['POST'])]
    public function create(Request $request, EntityManagerInterface $em): JsonResponse
    {
        $body = json_decode($request->getContent(), true);

        if (!isset($body['name'], $body['email'])) {
```

```
            return $this->json(['error' => 'Données invalides'], 400);
        }

        $user = new User();
        $user->setName($body['name']);
        $user->setEmail($body['email']);
        $user->setRegisteredAt(new \DateTimeImmutable());

        $em->persist($user);
        $em->flush();

        return $this->json([
            'id' => $user->getId(),
            'name' => $user->getName(),
            'email' => $user->getEmail(),
            'registeredAt' => $user->getRegisteredAt()->format('Y-m-d'),
        ], 201);
    }

    /*------------------------------Mise à jour ------------------*/

  #[Route('/{id}', methods: ['PUT'])]
public function update(
    int $id,
    Request $request,
```

```
    EntityManagerInterface $em,
    UserRepository $repo
): JsonResponse {
    $user = $repo->find($id);

    if (!$user) {
        return $this->json(['error' => 'Utilisateur non trouvé'], 404);
    }

    $data = json_decode($request->getContent(), true);

    if (!isset($data['name'], $data['email'])) {
        return $this->json(['error' => 'Données invalides'], 400);
    }

    $user->setName($data['name']);
    $user->setEmail($data['email']);
    $em->flush();

    return $this->json([
        'id' => $user->getId(),
        'name' => $user->getName(),
        'email' => $user->getEmail(),
        'registeredAt' => $user->getRegisteredAt()->format('Y-m-d'),
    ]);
```

```
}

/*----------------------Delete-------------------------------*/

#[Route('/{id}', methods: ['DELETE'])]
public function delete(
    int $id,
    EntityManagerInterface $em,
    UserRepository $repo
): JsonResponse {
    //  Recherche de l'utilisateur
    $user = $repo->find($id);

    if (!$user) {
        return $this->json(['error' => 'Utilisateur non trouvé'], 404);
    }

    //  Suppression
    $em->remove($user);
    $em->flush();

    //  Réponse de confirmation
    return $this->json(['message' => 'Utilisateur supprimé avec succès']);
}
```

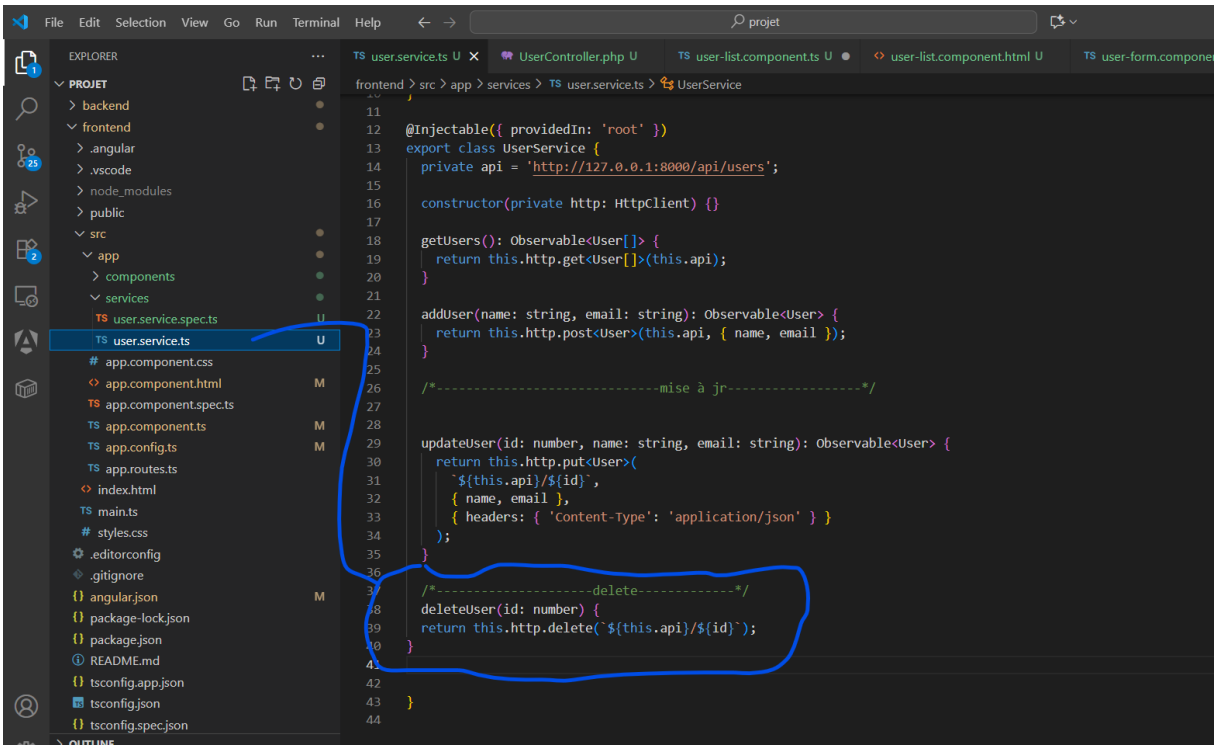
}

-> Cette route est déjà prête à recevoir la requête DELETE depuis Angular.

2. Côté frontend (Angular)

a) Méthode de suppression dans le service

Dans user.service.ts :



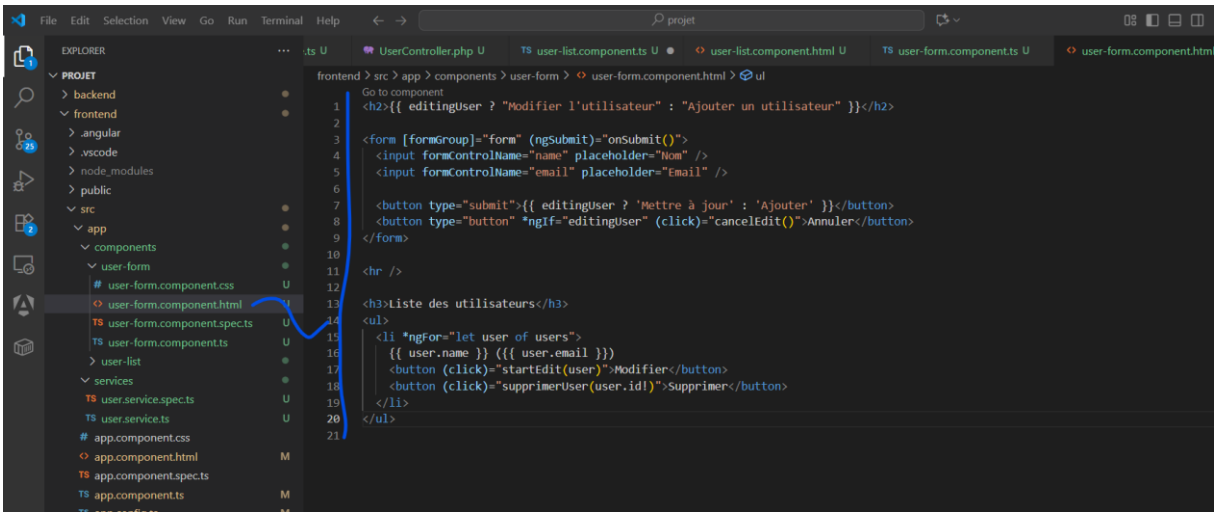

```
import { Injectable } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';


export interface User {

  id?: number;

  name: string;

  email: string;

  registeredAt: string;
```

```
}

@Injectable({ providedIn: 'root' })
export class UserService {
  private api = 'http://127.0.0.1:8000/api/users';

  constructor(private http: HttpClient) {}

  getUsers(): Observable<User[]> {
    return this.http.get<User[]>(this.api);
  }

  addUser(name: string, email: string): Observable<User> {
    return this.http.post<User>(this.api, { name, email });
  }

  /*------------------------------mise à jr------------------*/

  updateUser(id: number, name: string, email: string): Observable<User> {
    return this.http.put<User>(
      `${this.api}/${id}`,
      { name, email },
      { headers: { 'Content-Type': 'application/json' } }
    );
```

```
  }

  /*----------------------delete-------------*/
  deleteUser(id: number) {
  return this.http.delete(`${this.api}/${id}`);
}

}
```

b) Bouton "Supprimer" dans le formulaire

Dans ton fichier **user-form.component.html**, ajoute un bouton “Supprimer” :

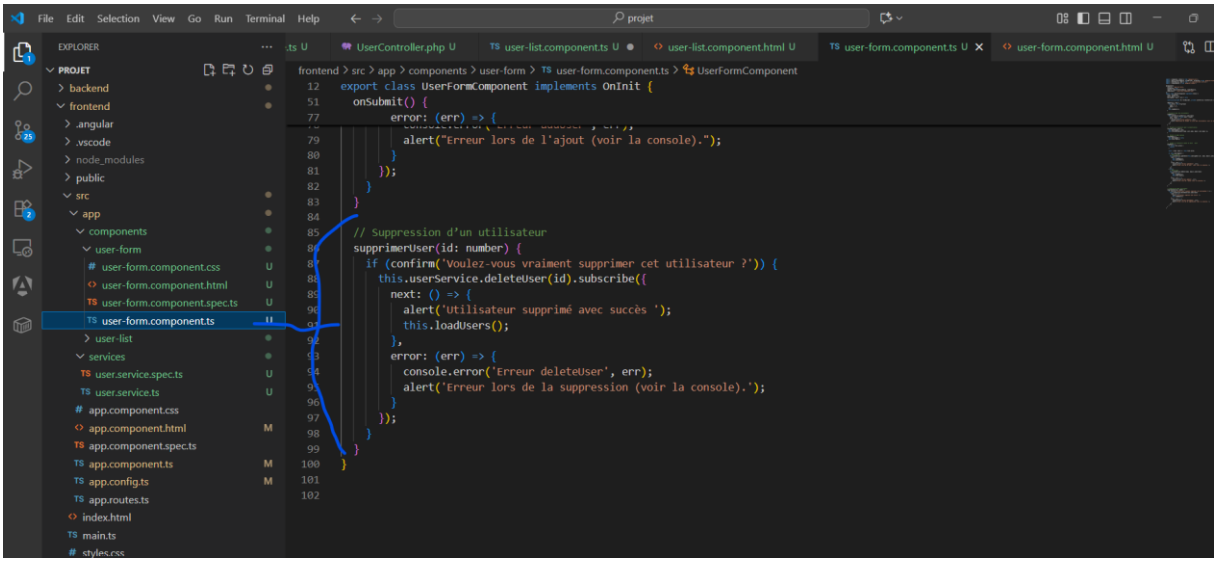



```
<h2>{{ editingUser ? "Modifier l'utilisateur" : "Ajouter un utilisateur" }}</h2>

<form [formGroup]="form" (ngSubmit)="onSubmit()">
  <input formControlName="name" placeholder="Nom" />
  <input formControlName="email" placeholder="Email" />
```

```
  <button type="submit">{{ editingUser ? 'Mettre à jour' : 'Ajouter' }}</button>
  <button type="button" *ngIf="editingUser" (click)="cancelEdit()">Annuler</button>
</form>

<hr />

<h3>Liste des utilisateurs</h3>
<ul>
  <li *ngFor="let user of users">
    {{ user.name }} ({{ user.email }})
    <button (click)="startEdit(user)">Modifier</button>
    <button (click)="supprimerUser(user.id!)">Supprimer</button>
  </li>
</ul>
```

c) Méthode dans user-form.component.ts

Ajoute cette méthode :

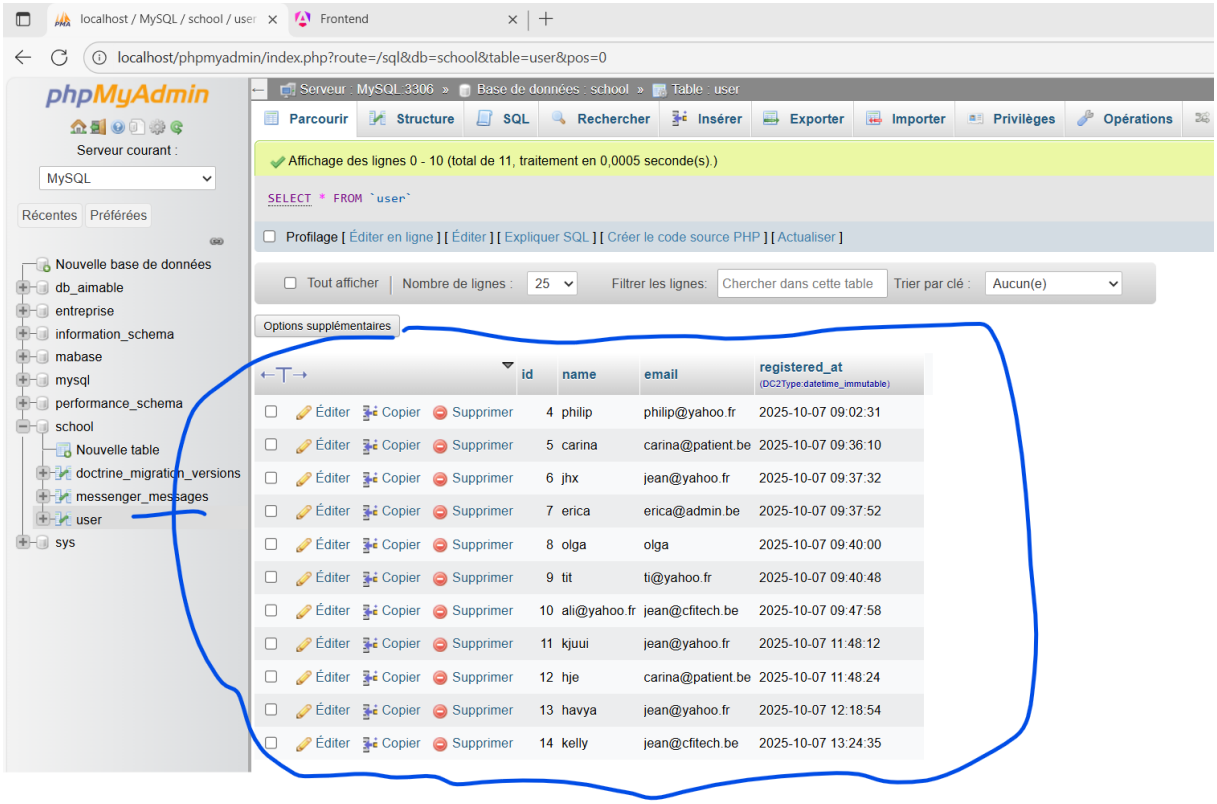


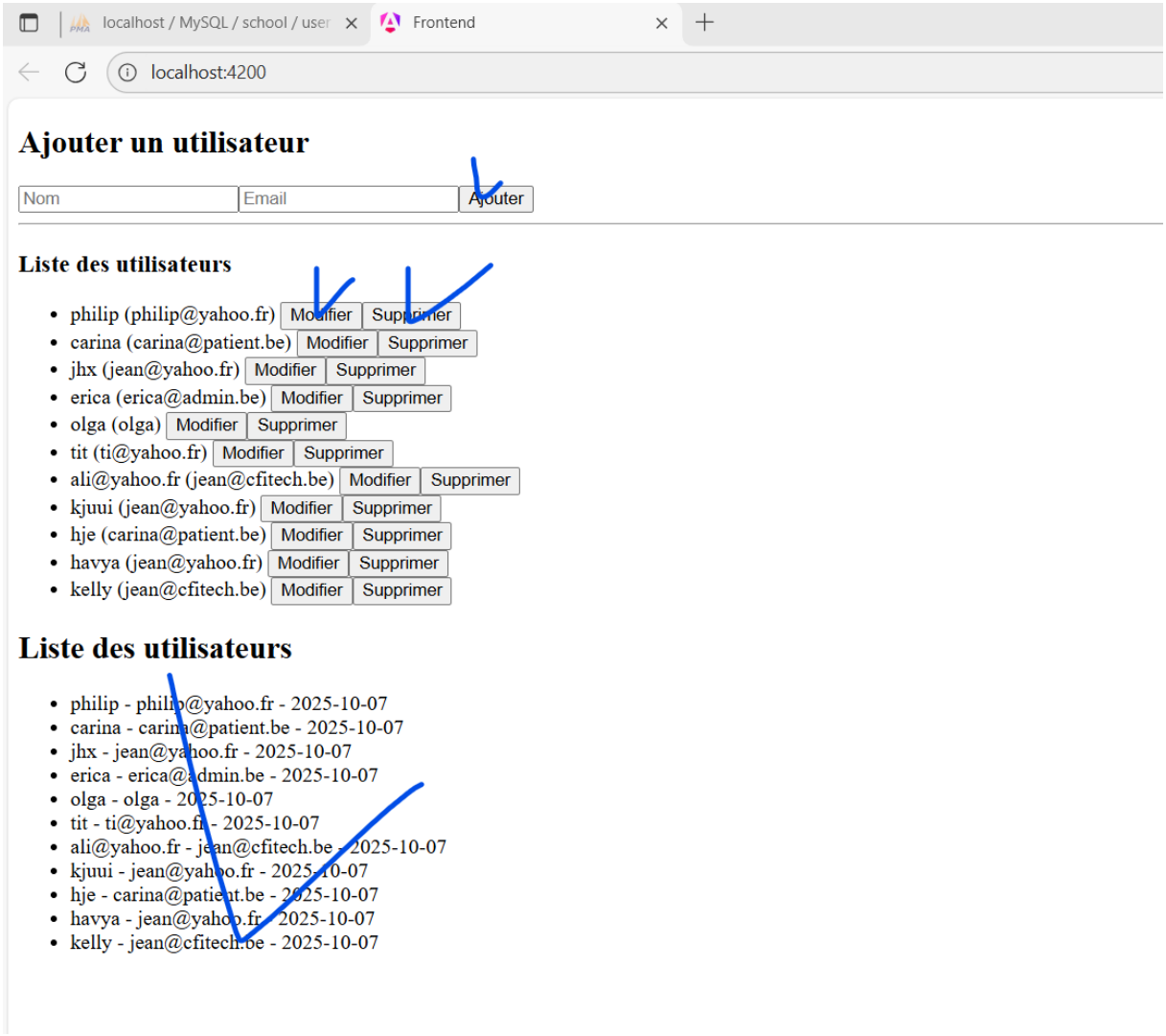
```
import { Component, OnInit } from '@angular/core';
import { ReactiveFormsModule, FormBuilder, FormGroup } from '@angular/forms';
import { UserService, User } from '../../services/user.service';
import { CommonModule } from '@angular/common';

@Component({
  selector: 'app-user-form',
  standalone: true,
  imports: [ReactiveFormsModule, CommonModule],
  templateUrl: './user-form.component.html'
})
export class UserFormComponent implements OnInit {
  form!: FormGroup;
  users: User[] = [];
  editingUser: User | null = null;

  constructor(private fb: FormBuilder, private userService: UserService) {}

  ngOnInit(): void {
    this.form = this.fb.group({
      name: [''],
      email: ['']
    });
    this.loadUsers();
  }
```

```
// Récupère tous les utilisateurs
loadUsers() {
  this.userService.getUsers().subscribe({
    next: (data) => (this.users = data),
    error: (err) => {
      console.error('Erreur getUsers', err);
      alert('Impossible de charger la liste des utilisateurs (voir la console).');
    }
  });
}

//  Prépare le formulaire pour la modification
startEdit(user: User) {
  this.editingUser = user;
  this.form.patchValue({ name: user.name, email: user.email });
}

//  Annule le mode édition
cancelEdit() {
  this.editingUser = null;
  this.form.reset();
}

//  Soumet le formulaire (ajout ou mise à jour)
```

```
onSubmit() {
  if (!this.form.valid) {
    return;
  }

  const { name, email } = this.form.value;

  if (this.editingUser) {
    // Mode édition
    this.userService.updateUser(this.editingUser.id!, name, email).subscribe({
      next: () => {
        this.loadUsers();
        this.cancelEdit();
      },
      error: (err) => {
        console.error('Erreur updateUser', err);
        alert('Erreur lors de la mise à jour (voir la console).');
      }
    });
  } else {
    // Mode ajout
    this.userService.addUser(name, email).subscribe({
      next: () => {
        this.loadUsers();
        this.form.reset();
```

```
      },
      error: (err) => {
        console.error('Erreur addUser', err);
        alert("Erreur lors de l'ajout (voir la console).");
      }
    });
  }
 }

 // Suppression d'un utilisateur
 supprimerUser(id: number) {
  if (confirm('Voulez-vous vraiment supprimer cet utilisateur ?')) {
    this.userService.deleteUser(id).subscribe({
      next: () => {
        alert('Utilisateur supprimé avec succès ');
        this.loadUsers();
      },
      error: (err) => {
        console.error('Erreur deleteUser', err);
        alert('Erreur lors de la suppression (voir la console).');
      }
    });
  }
 }
}
```


localhost / MySQL / school / user
Frontend
localhost/phpmyadmin/index.php?route=/sql&db=school&table=user&pos=0
phpMyAdmin
Serveur courant :
MySQL
Récentes
Préférées
Nouvelle base de données
db_aimable
entreprise
information_schema
mabase
mysql
performance_schema
school
Nouvelle table
doctrine_migration_versions
messenger_messages
user
sys
Serveur : MySQL:3306 » Base de données : school » Table : user
Parcourir
Structure
SQL
Rechercher
Insérer
Exporter
Importer
Privilèges
Opérations
Affichage des lignes 0 - 10 (total de 11, traitement en 0,0005 seconde(s).)
SELECT * FROM `user`
Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]
Tout afficher
Nombre de lignes : 25
Filtrer les lignes:
Chercher dans cette table
Trier par clé :
Aucun(e)
Options supplémentaires
id
name
email
registered_at
(DC2Type:datetime_immutable)
Éditer Copier Supprimer 4 philip philip@yahoo.fr 2025-10-07 09:02:31
Éditer Copier Supprimer 5 carina carina@patient.be 2025-10-07 09:36:10
Éditer Copier Supprimer 6 jhx jean@yahoo.fr 2025-10-07 09:37:32
Éditer Copier Supprimer 7 erica erica@admin.be 2025-10-07 09:37:52
Éditer Copier Supprimer 8 olga olga 2025-10-07 09:40:00
Éditer Copier Supprimer 9 tit ti@yahoo.fr 2025-10-07 09:40:48
Éditer Copier Supprimer 10 ali@yahoo.fr jean@cfitech.be 2025-10-07 09:47:58
Éditer Copier Supprimer 11 kjuui jean@yahoo.fr 2025-10-07 11:48:12
Éditer Copier Supprimer 12 hje carina@patient.be 2025-10-07 11:48:24
Éditer Copier Supprimer 13 havya jean@yahoo.fr 2025-10-07 12:18:54
Éditer Copier Supprimer 14 kelly jean@cfitech.be 2025-10-07 13:24:35



⇨ **Créer un fichier CSS**

Dans le dossier :
src/app/components/user-form/user-form.component.css

```
/* ------------ FORMULAIRE ------- */

.form-title {

  text-align: center;

  margin-bottom: 15px;

  color: #333;

}
```

```
.user-form {
  max-width: 700px;
  margin: 0 auto 30px;
  background: #fff;
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0 3px 10px rgba(0, 0, 0, 0.1);
}

.form-row {
  display: flex;
  align-items: center;
  gap: 10px;
}

.user-form input {
  flex: 1;
  padding: 8px 12px;
  border: 1px solid #ccc;
  border-radius: 6px;
  font-size: 14px;
}

.btn-submit {
  background-color: #007bff;
```

```
  color: white;
  border: none;
  border-radius: 6px;
  padding: 8px 16px;
  cursor: pointer;
  transition: background-color 0.2s;
}

.btn-submit:hover {
  background-color: #0056b3;
}

.btn-cancel {
  margin-top: 10px;
  background-color: #6c757d;
  color: white;
  border: none;
  border-radius: 6px;
  padding: 8px 16px;
  cursor: pointer;
}

.btn-cancel:hover {
  background-color: #5a6268;
}
```

```
/* ===== TITRE DE LISTE ===== */
.list-title {
  text-align: center;
  margin-bottom: 15px;
  color: #333;
}

/* ===== LISTE DES UTILISATEURS ===== */
.user-list {
  max-width: 700px;
  margin: 0 auto;
  display: flex;
  flex-direction: column;
  gap: 10px;
}

.user-card {
  background-color: #fff;
  border: 1px solid #ddd;
  border-radius: 10px;
  padding: 12px 18px;
  display: flex;
  justify-content: space-between;
  align-items: center;
```

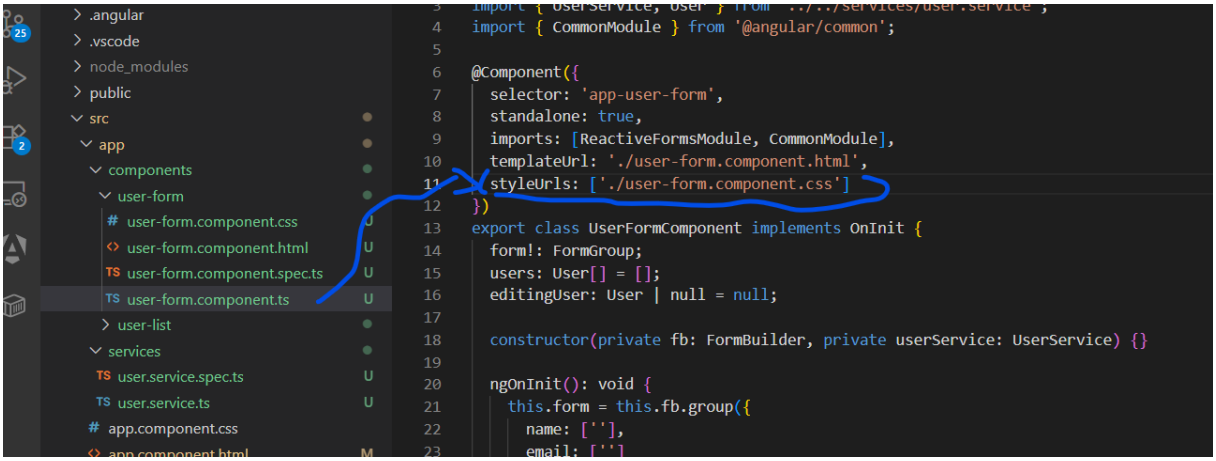
```
  transition: transform 0.2s ease, box-shadow 0.2s ease;
}

.user-card:hover {
  transform: scale(1.01);
  box-shadow: 0 3px 8px rgba(0, 0, 0, 0.1);
}

.user-info {
  display: flex;
  flex-direction: column;
}

.user-info strong {
  font-size: 16px;
  color: #333;
}

.user-info span {
  font-size: 14px;
  color: #555;
}

.user-info small {
  font-size: 12px;
```

```
  color: #888;
}

/* Boutons actions */
.actions {
  display: flex;
  gap: 10px;
}

.btn-edit {
  background-color: #28a745;
  color: white;
  padding: 6px 12px;
  border: none;
  border-radius: 6px;
  cursor: pointer;
}

.btn-edit:hover {
  background-color: #218838;
}

.btn-delete {
  background-color: #dc3545;
  color: white;
```

```
  padding: 6px 12px;
  border: none;
  border-radius: 6px;
  cursor: pointer;
}

.btn-delete:hover {
  background-color: #b02a37;
}
```

Vérifie que le composant utilise bien le CSS

Dans le fichier :
user-form.component.ts :



Modifier HTML

user-form.component.html :

```
<h2 class="form-title">
  {{ editingUser ? 'Modifier un utilisateur' : 'Ajouter un utilisateur' }}
```

```
</h2>

<form [formGroup]="form" (ngSubmit)="onSubmit()" class="user-form">
  <div class="form-row">
    <input type="text" formControlName="name" placeholder="Nom" />
    <input type="email" formControlName="email" placeholder="Email" />
    <button type="submit" class="btn-submit">
      {{ editingUser ? 'Mettre à jour' : 'Ajouter' }}
    </button>
  </div>
  <button
    *ngIf="editingUser"
    type="button"
    (click)="cancelEdit()"
    class="btn-cancel"
  >
    Annuler
  </button>
</form>

<hr />

<h2 class="list-title">Liste des utilisateurs</h2>

<div class="user-list">
```

```
<div *ngFor="let user of users" class="user-card">
  <div class="user-info">
    <strong>{{ user.name }}</strong>
    <span>{{ user.email }}</span>
    <small>{{ user.registeredAt }}</small>
  </div>
  <div class="actions">
    <button class="btn-edit" (click)="startEdit(user)">Modifier</button>
    <button class="btn-delete" (click)="supprimerUser(user.id!)">
      Supprimer
    </button>
  </div>
 </div>
</div>
```