

# SYSTÈMES DISTRIBUÉS

MASTER INTELLIGENCE  
ARTIFICIELLE ET ANALYSE  
DES DONNÉES

RÉALISÉE PAR :  
ANOADA NOHAYLA

ENCADRÉ PAR :  
M. MOHAMED YOUSFI

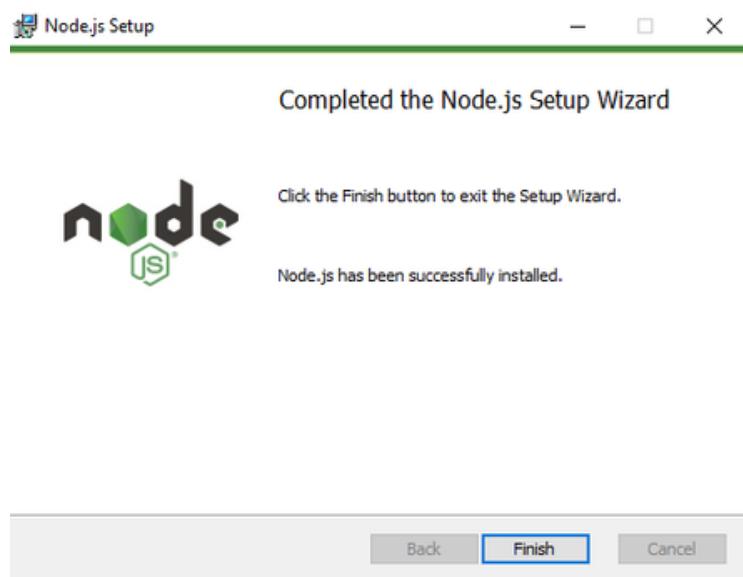


# ACTIVITÉ PRATIQUE N° 4

## ANGULAR - SPRING

### PARTIE 1 :

#### 1. TÉLÉCHARGEMENT ET INSTALLATION DE NODE.JS



téléchargement et installation de Node.js depuis le site officiel. L'installation a été confirmée par l'assistant d'installation.

#### 2. VÉRIFICATION DE L'INSTALLATION DE NPM

```
C:\Users\nohayla>npm --version  
10.5.2
```

### 3. VÉRIFICATION DE L'ANGULAR CLI

```
C:\Users\nohayla>ng version

Angular CLI: 17.3.7
Node: 20.13.1
Package Manager: npm 10.5.2
OS: win32 x64

Angular: undefined
...


| Package                    | Version             |
|----------------------------|---------------------|
| @angular-devkit/architect  | 0.1703.7 (cli-only) |
| @angular-devkit/core       | 17.3.7 (cli-only)   |
| @angular-devkit/schematics | 17.3.7 (cli-only)   |
| @schematics/angular        | 17.3.7 (cli-only)   |


```

## 4. INSTALLATION DE JSON-SERVER

```
C:\Users\nohayla>npm i -g json-server@0.17.4  
added 116 packages in 15s
```

## PARTIE 2:

# 1. CRÉATION D'UN PROJET ANGULAR

```
C:\Users\nohayla\Desktop\study\school\Master\S2\Systèmes_Distribués\Tp\Anoada_Nohayla_Tp4_1
>ng new Anoada_Nohayla_Tp4_1
? Which stylesheet format would you like to use? CSS [CSS]
https://developer.mozilla.org/docs/Web/CSS
? Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? Yes
CREATE Anoada_Nohayla_Tp4_1/angular.json (2908 bytes)
CREATE Anoada_Nohayla_Tp4_1/package.json (1305 bytes)
CREATE Anoada_Nohayla_Tp4_1/README.md (1098 bytes)
CREATE Anoada_Nohayla_Tp4_1/tsconfig.json (889 bytes)
CREATE Anoada_Nohayla_Tp4_1/.editorconfig (290 bytes)
CREATE Anoada_Nohayla_Tp4_1/.gitignore (629 bytes)
CREATE Anoada_Nohayla_Tp4_1/tsconfig.app.json (342 bytes)
CREATE Anoada_Nohayla_Tp4_1/tsconfig.spec.json (287 bytes)
CREATE Anoada_Nohayla_Tp4_1/server.ts (1759 bytes)
CREATE Anoada_Nohayla_Tp4_1/.vscode/extensions.json (134 bytes)
CREATE Anoada_Nohayla_Tp4_1/.vscode/launch.json (490 bytes)
CREATE Anoada_Nohayla_Tp4_1/.vscode/tasks.json (980 bytes)
CREATE Anoada_Nohayla_Tp4_1/src/main.ts (256 bytes)
CREATE Anoada_Nohayla_Tp4_1/src/favicon.ico (15086 bytes)
CREATE Anoada_Nohayla_Tp4_1/src/index.html (316 bytes)
CREATE Anoada_Nohayla_Tp4_1/src/styles.css (81 bytes)
CREATE Anoada_Nohayla_Tp4_1/src/main.server.ts (271 bytes)
CREATE Anoada_Nohayla_Tp4_1/src/app/app.component.html (20239 bytes)
CREATE Anoada_Nohayla_Tp4_1/src/app/app.component.spec.ts (987 bytes)
CREATE Anoada_Nohayla_Tp4_1/src/app/app.component.ts (329 bytes)
CREATE Anoada_Nohayla_Tp4_1/src/app/app.component.css (0 bytes)
CREATE Anoada_Nohayla_Tp4_1/src/app/app.config.ts (330 bytes)
CREATE Anoada_Nohayla_Tp4_1/src/app/app.routes.ts (80 bytes)
CREATE Anoada_Nohayla_Tp4_1/src/app/app.config.server.ts (361 bytes)
CREATE Anoada_Nohayla_Tp4_1/src/assets/.gitkeep (0 bytes)
? Packages installed successfully.
warning: in the working copy of 'package-lock.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'package.json', LF will be replaced by CRLF the next time Git touches it
    Successfully initialized git.
```

## 2. LANCER LE SERVEUR DE DÉVELOPPEMENT

```
PS C:\Users\nohayla\Desktop\study\school\Master\S2\Systèmes_Distribués
\Tp\Anoada_Nohayla_Tp4\Anoada_Nohayla_Tp4_1> ng serve
  ✓ Browser application bundle generation complete.

  Initial Chunk Files | Names          | Raw Size
  vendor.js           | vendor        |  2.04 MB |
  polyfills.js        | polyfills     | 313.74 kB |
  styles.css, styles.js | styles       | 208.86 kB |
  main.js             | main         | 48.77 kB |
  runtime.js          | runtime      |  6.54 kB |

  | Initial Total |  2.61 MB

  Build at: 2024-05-20T21:13:45.021Z - Hash: 8feadb1b1da1cc0a - Time: 79
  608ms

  ** Angular Live Development Server is listening on localhost:4200, ope
  n your browser on http://localhost:4200/ **
```

## 3. INSTALLATION DE BOOTSTRAP

```
PS C:\Users\nohayla\Desktop\study\school\Master\S2\Systèmes_Distribués\Tp\
Anoada_Nohayla_Tp4\Anoada_Nohayla_Tp4_1> npm i bootstrap bootstrap-icons
added 3 packages, removed 1 package, and audited 987 packages in 27s

  styles.css ×
  1  /* You can add global styles to this file, and also
  2   @import "bootstrap-icons/font/bootstrap-icons.css";

  "styles": [
    "src/styles.css",
    "node_modules/bootstrap/dist/css/bootstrap.min.css"
  ],
  "scripts": [
    "node_modules/bootstrap/dist/js/bootstrap.bundle.js"
  ]
```

## 3. INSTALLATION DE BOOTSTRAP

```
PS C:\Users\nohayla\Desktop\study\school\Master\S2\Systèmes_Distribués\Tp
\Anoada_Nohayla_Tp4\Anoada_Nohayla_Tp4_1> ng g c home
CREATE src/app/home/home.component.html (19 bytes)
CREATE src/app/home/home.component.spec.ts (585 bytes)
CREATE src/app/home/home.component.ts (194 bytes)
CREATE src/app/home/home.component.css (0 bytes)
UPDATE src/app/app.module.ts (467 bytes)
PS C:\Users\nohayla\Desktop\study\school\Master\S2\Systèmes_Distribués\Tp
\Anoada_Nohayla_Tp4\Anoada_Nohayla_Tp4_1> ng g c products
CREATE src/app/products/products.component.html (23 bytes)
CREATE src/app/products/products.component.spec.ts (613 bytes)
CREATE src/app/products/products.component.ts (210 bytes)
CREATE src/app/products/products.component.css (0 bytes)
UPDATE src/app/app.module.ts (557 bytes)
```

```

PS C:\Users\nohayla\Desktop\study\school\Master\S2\Systèmes_Distribués\Tp\
Anoada_Nohayla_Tp4\Anoada_Nohayla_Tp4_1> ng g c new-product
CREATE src/app/new-product/new-product.component.html (26 bytes)
CREATE src/app/new-product/new-product.component.spec.ts (628 bytes)
CREATE src/app/new-product/new-product.component.ts (221 bytes)
CREATE src/app/new-product/new-product.component.css (0 bytes)

```

## 4. BASE DE DONNÉE

```

{
  "users": [
    {
      "id": "user1",
      "password": "MTIzNA==",
      "roles": [
        "USER"
      ],
      "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ1c2VyMSIsImlhCI6MTUxNjIyMjIwMDA."
    },
    {
      "id": "nohayla",
      "password": "MTIzNA==",
      "roles": [
        "USER"
      ],
      "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ1c2VyMiIsImlhCI6MTUxNjIyMjIwMDA."
    },
    {
      "id": "admin",
      "password": "MTIzNA==",
      "roles": [
        "USER",
        "ADMIN"
      ],
      "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJhZG1pbkiIsImlhCI6MTUxNjIyMjIwMDA."
    }
  ],
  "products": [
    {
      "id": "1",
      "name": "Computer",
      "price": 8000,
      "checked": true
    },
    {
      "id": "2",
      "name": "Computer",
      "price": 7000,
      "checked": true
    },
    {
      "id": "3",
      "name": "Smart phone",
      "price": 3000,
      "checked": false
    },
    {
      "id": "5",
      "name": "Smart phone",
      "price": 4500,
      "checked": true
    }
  ]
}

```

Le fichier db.json sert de base de données pour les utilisateurs et les produits. Il est utilisé par le serveur JSON pour stocker et gérer les données.

- **Users:** Contient les informations des utilisateurs, y compris les identifiants, les mots de passe (encodés en Base64), les rôles (USER, ADMIN) et les tokens JWT pour l'authentification.
- **Products:** Contient les détails des produits avec des attributs tels que id, name, price, et checked (indiquant si le produit est coché ou non).

## 5. LANCER L'APPLICATION

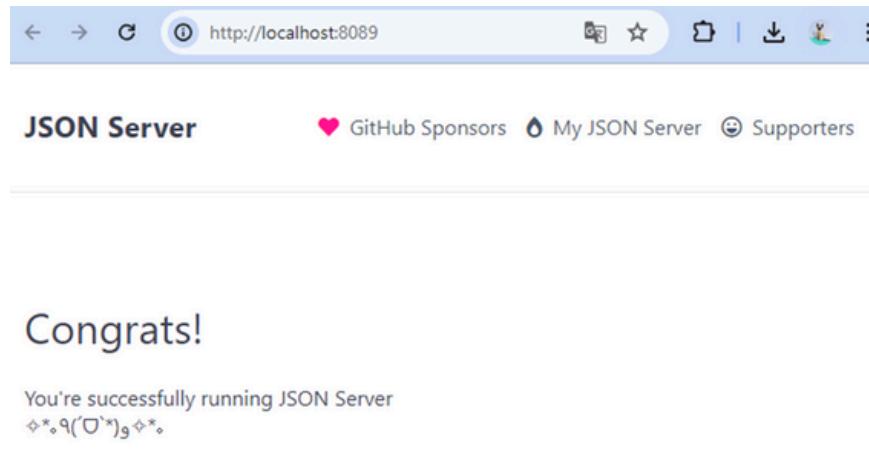
### 5.1 LANCER LE SERVEUR ANGULAR:

```
\Anoada_Nohayla_Tp4\Anoada_Nohayla_Tp4_1> ng serve
Browser bundles
main.server.mjs      | main.server          | 428.13 kB |
chunk-VPSODEBW.mjs   | -                  | 2.51 kB |
render-utils.server.mjs | render-utils.server | 423 bytes |

Lazy chunk files      | Names           | Raw size
chunk-OTT6LQ5K.mjs    | xhr2            | 39.10 kB |

Application bundle generation complete. [45.933 seconds]

Watch mode enabled. Watching for file changes...
→ Local: http://localhost:4200/
→ press h + enter to show help
```



## 5.2 LANCER LE SERVEUR JSON:

```
\Anoada_Nohayla_Tp4\Anoada_Nohayla_Tp4_1> json-server -w data/db.json -p 8089

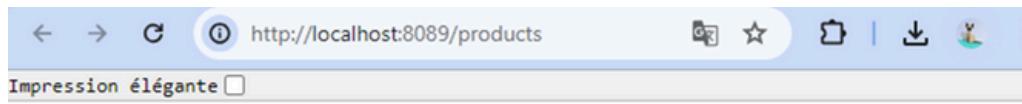
\{^_^}/ hi!

Loading data/db.json
Done

Resources
http://localhost:8089/users
http://localhost:8089/products

Home
http://localhost:8089

Type s + enter at any time to create a snapshot of the database
Watching...
```



The screenshot shows a web browser window with the URL <http://localhost:8089/products> in the address bar. The page content displays a JSON array of product items.

```
[{"id": "1", "name": "Computer", "price": 8000, "checked": true}, {"id": "2", "name": "Computer", "price": 7000, "checked": true}, {"id": "3", "name": "Smart phone", "price": 3000, "checked": false}, {"id": "4", "name": "Laptop", "price": "12000", "checked": true}, {"id": "5", "name": "Smart phone", "price": "4500", "checked": true}, {"id": "uP8J0zs", "name": "Smart phone", "price": "12000", "checked": true}]
```

```

[{"id": "user1", "password": "MTIzNA==", "roles": ["USER"], "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIoiJ1c2VyMSIsImhdCI6MTUxNjIzOTAyMiwiZXhwIjoxNTE2MjMSMDiyLCJpc3MiOjIodHRwOi8vbG9jYWxob3N0Ojg4MDgvYXV0aCIsIm5iZiI6MTUxNjIzOTAyMiwiZmlyc3ROYW1lIjoiTW9oYW1tZWQiLCJsYXN0TmFtZSI6I1lPVVNTRkkilCJlbWFpbCI6Im1lZEByb2xlcyI6WyJVU0VSIl19.kATxOoGzNi6NOralr_2k2i-ekK2gI0w7G7ty0t031n8"}, {"id": "nohayla", "password": "MTIzNA==", "roles": ["USER"], "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIoiJ1c2VyMiIsImhdCI6MTUxNjIzOTAyMiwiZXhwIjoxNTE2MjMSMDiyLCJpc3MiOjIodHRwOi8vbG9jYWxob3N0Ojg4MDgvYXV0aCIsIm5iZiI6MTUxNjIzOTAyMiwiZmlyc3ROYW1lIjoiTW9oYW1tZWQiLCJsYXN0TmFtZSI6I1lPVVNTRkkilCJlbWFpbCI6Im1lZEByb2xlcyI6WyJVU0VSIl19.RhY-9z5rywbJH0qwGO5PPxQafUx1ZSCnMXJ2y8qy7w"}, {"id": "admin", "password": "MTIzNA==", "roles": ["USER", "ADMIN"], "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIoiJhZG1pbisImhdCI6MTUxNjIzOTAyMiwiZXhwIjoxNTE2MjMSMDiyLCJpc3MiOjIodHRwOi8vbG9jYWxob3N0Ojg4MDgvYXV0aCIsIm5iZiI6MTUxNjIzOTAyMiwiZmlyc3ROYW1lIjoiTW9oYW1tZWQiLCJsYXN0TmFtZSI6I1lPVVNTRkkilCJlbWFpbCI6Im1lZEByb2xlcyI6WyJVU0VSIlwiQUFRNSU4iXX0.g6LoXcTSM3y6ZxaRBCDJplvJ-KSrqjtCNf93JC7SoKrA"}]

```

## G. COMPOSANTS

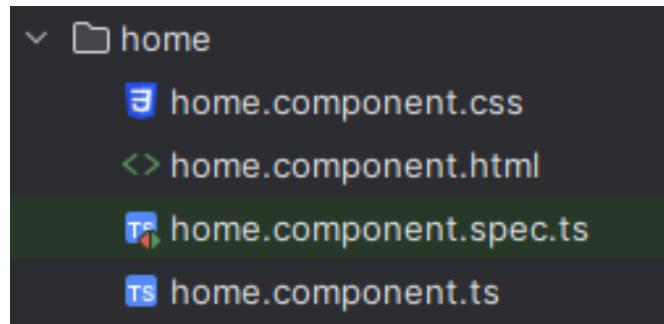
### G.1 HOME

- CRÉATION DÉ COMPOSANT

```
\Anoada_Nohayla_Tp4\Anoada_Nohayla_Tp4_1> ng g c home
CREATE src/app/home/home.component.html (20 bytes)
CREATE src/app/home/home.component.spec.ts (605 bytes)
CREATE src/app/home/home.component.css (0 bytes)
```

Cette commande permet de générer un composant angulaire nommé "home" à l'aide de Angular CLI

la commande génère les fichiers suivants :



- **src/app/home/home.component.ts:**  
Fichier TypeScript pour la logique du composant.
- **src/app/home/home.component.html**  
Template HTML pour le composant.
- **src/app/home/home.component.css)**  
Feuille de style pour le composant.
- **src/app/home/home.component.spec.ts**  
Fichier de test unitaire pour le composant.
- **TYPESCRIPT (HOME.COMPONENT.TS)**

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

1+ usages
@Component({
  selector: 'app-home',
  standalone: true,
  imports: [],
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css'
})
export class HomeComponent
```

- **TEMPLATE HTML  
(HOME.COMPONENT.HTML)**

```
<div class="p-3">
  <div class="card">
    <div class="card-body">
      <h3>Home Component</h3>
    </div>
  </div>
</div>
```

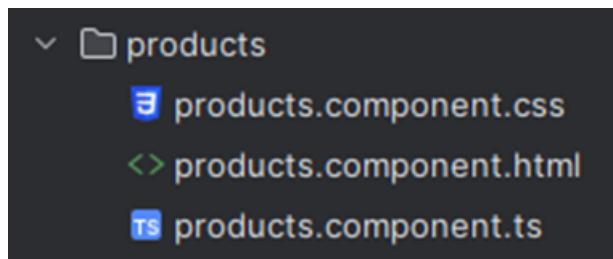
Home Component

## 6.2 PRODUCTS

- **CRÉATION DE COMPOSANT**

```
\Anoada_Nohayla_Tp4\Anoada_Nohayla_Tp4_1> ng g c products
CREATE src/app/products/products.component.ts (254 bytes)
CREATE src/app/products/products.component.css (0 bytes)
```

Cette commande vous permet de générer un composant angular nommé "products" à l'aide de Angular CLI  
la commande génère les fichiers suivants :



- **src/app/products/products.component.ts:**  
Fichier TypeScript pour la logique du composant.
- **src/app/products/products.component.html**  
Template HTML pour le composant.
- **src/app/products/products.component.css**  
Feuille de style pour le composant.
- **src/app/products/products.component.spec.ts**  
Fichier de test unitaire pour le composant.

## • TYPESCRIPT (PRODUCTS.COMPONENT.TS)

```
@Component({
  selector: 'app-products',
  standalone: true,
  imports: [
    HttpClientModule,
    NgForOf,
    AsyncPipe,
    FormsModule,
    NgIf,
    NgClass
  ],
  templateUrl: './products.component.html',
  styleUrls: ['./products.component.css'
})
export class ProductsComponent implements OnInit
{
  no usages
  constructor(private productService:ProductService,
              private router: Router,
              public appState: AppStateService)
  {}
  no usages
  ngOnInit(): void
  searchProducts(): void
  {
    /*this.appState.setProductState({
      status : "LOADING"
   });*/
    this.productService.searchProducts(
      this.appState.productsState.keyword,
      this.appState.productsState.currentPage,
      this.appState.productsState.pageSize)
      .subscribe( observerOrNext: {
        next: (resp : HttpResponse<Object>) : void  => {
          let products : Product[] = resp.body as Product[];

          let totalProducts: number = parseInt(resp.headers.get('x-total-count'))!;

          // this.appState.productsState.totalProducts=totalProducts;

          let totalPages : number = Math.floor( totalProducts / this.appState.productsState.pageSize);

          if (totalProducts % this.appState.productsState.pageSize != 0)
          {
            ++totalPages;
          }
          this.appState.setProductState({
            products : products,
            totalProducts : totalProducts,
            totalPages : totalPages,
            status : "LOADED"
          })
        },
        error: err => {
          this.appState.setProductState({
            status : "ERROR",
            errorMessage : err
          })
        }
      })
  }
}
```

```

handleCheckProduct(product: Product) : void
{
  this.productService.checkProducts(product)
    .subscribe( observerOrNext: {
      next: updatedProduct : Product  => {
        product.checked=!product.checked;
        //this.searchProducts();
      },
      error: err => {
        console.log(err);
      }
    });
}

1+ usages
handleDelete(product: Product) : void
{
  if (confirm("Êtes-vous sûr de vouloir supprimer ce produit? "))
    this.productService.deleteProducts(product).subscribe(
      observerOrNext: {
        next:value => {
          this.searchProducts();
          //this.appState.productsState.products = this.appState.productsState.products.filter((p:any)=>p.id!=product.id);
        }
      }
    );
}

1+ usages
handleGotoPage(page: number) : void {
  this.appState.productsState.currentPage=page;
  this.searchProducts();
}

1+ usages
handleEdit(product: Product) : void {
  this.router.navigateByUrl( url: `/admin/editProduct/${product.id}`);
}
}

```

**La classe implémente OnInit pour initialiser les données au chargement du composant.**

### Constructeur :

**Le constructeur injecte les services nécessaires :**

- **ProductService** : Pour les opérations sur les produits.
- **Router** : Pour la navigation.
- **AppStateService** : Pour gérer l'état de l'application.

### Méthodes :

- **ngOnInit** : Cette méthode est appelée après l'initialisation du composant et déclenche la recherche des produits.

- **searchProducts** : Cette méthode recherche les produits en fonction du mot-clé, de la page actuelle et de la taille de la page. Elle met à jour l'état des produits et la pagination.
- **handleCheckProduct** : Cette méthode gère la vérification d'un produit, en inversant son état "checked".
- **handleDelete** : Cette méthode gère la suppression d'un produit après confirmation de l'utilisateur.
- **handleGotoPage** : Cette méthode gère la navigation vers une page spécifique et relance la recherche de produits.
- **handleEdit** : Cette méthode gère la navigation vers la page d'édition d'un produit.

Le composant **ProductsComponent** fournit une interface utilisateur complète pour gérer une liste de produits. Il permet de rechercher, afficher, vérifier, supprimer, éditer des produits et de naviguer entre les pages de résultats. Le composant utilise des services pour effectuer des opérations CRUD et gérer l'état de l'application.

## • TEMPLATE HTML (COMPONENT.HTML)

```
<div class="p-3">
  <div class="card">
    <div class="card-body">
      <div class="card-body">
        {{appState.productsState.keyword}}
        <input type="text" [(ngModel)]="appState.productsState.keyword" >
        <button (click)="searchProducts()" class="btn btn-outline-success ms-1">
          <i class="bi bi-search"></i>
        </button>
      </div>
      <table class="table">
        <thead>
          <tr>
            <th>Name</th> <th>Price</th> <th>Checked</th>
          </tr>
        </thead>
```

```

<tbody>
  <tr *ngFor="let product of appState.productsState.products">
    <td>{{product.name}}</td>
    <td>{{product.price}}</td>
    <td>
      <button (click)="handleCheckProduct(product)" class="btn btn-outline-success">
        <i [class] = "product.checked?'bi bi-check' : 'bi bi-circle '"></i>
      </button>
    </td>
    <td>
      <button (click)="handleDelete(product)" class="btn btn-outline-danger">
        <i class="bi bi-trash"></i>
      </button>
    </td>
    <td>
      <button (click)="handleEdit(product)" class="btn btn-outline-success">
        <i class="bi bi-pencil"></i>
      </button>
    </td>
  </tr>
</tbody>
</table>
<ul class="nav nav-pills" *ngIf="appState.productsState.totalPages && appState.productsState.totalPages > 0">
  <li *ngFor="let page of [].constructor(this.appState.productsState.totalPages); let i=index">
    <button (click)="handleGotoPage(page: i+1)" [ngClass]="appState.productsState.currentPage==(i+1)?'btn-success' : 'btn-outline-success'" class="btn m-1">
      {{i+1}}
    </button>
  </li>
</ul>
</div>
</div>
</div>

```

Ce code HTML crée une interface utilisateur pour gérer une liste de produits. Il permet aux utilisateurs de rechercher des produits par mot-clé, d'afficher les produits dans une table avec des options pour les vérifier, les supprimer ou les éditer, et de naviguer entre différentes pages de résultats. La mise en page utilise des classes Bootstrap pour le style et la présentation.

The screenshot shows a user interface for managing a list of products. At the top is a search bar with a magnifying glass icon. Below it is a table with three columns: Name, Price, and Checked. The table contains three rows of data:

Name	Price	Checked
Computer	8000	<input checked="" type="checkbox"/>
Computer	7000	<input checked="" type="checkbox"/>
Smart phone	3000	<input type="checkbox"/>

Each row has three buttons in the last column: a red trash can icon for deletion, a green pencil icon for editing, and a green checkmark icon for marking as checked. At the bottom of the table is a navigation section with three buttons labeled 1, 2, and 3, representing different pages of results.

com com

Name	Price	Checked		
Computer	8000	✓		
Computer	7000	✓		

1



com com

Name	Price	Checked		
Computer	8000	✓		
Computer	7000	✓		

1

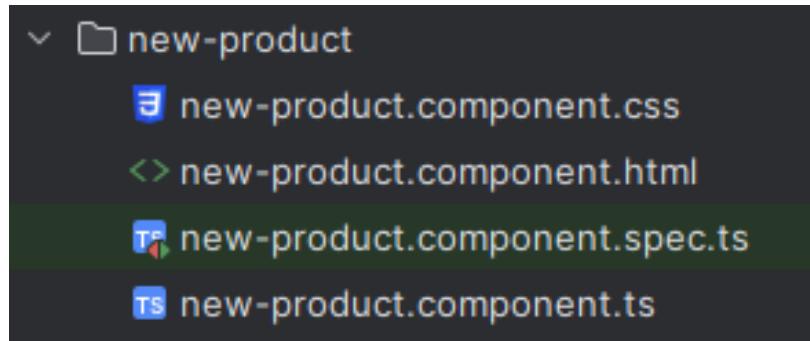
## 6.3 NEW-PRODUCT

### • CRÉATION DE COMPOSANT

```
\Anoada_Nohayla_Tp4\Anoada_Nohayla_Tp4_1> ng g c new-product
CREATE src/app/new-product/new-product.component.html (27 bytes)
CREATE src/app/new-product/new-product.component.spec.ts (648 bytes)
CREATE src/app/new-product/new-product.component.ts (265 bytes)
CREATE src/app/new-product/new-product.component.css (0 bytes)
```

Cette commande vous permet de générer un composant angulaire nommé "new-product" à l'aide de Angular CLI

la commande génère les fichiers suivants :



- **src/app/new-product/new-product.component.ts:**  
Fichier TypeScript pour la logique du composant.
- **src/app/new-product/new-product.component.html**  
Template HTML pour le composant.
- **src/app/new-product/new-product.component.css)**  
Feuille de style pour le composant.
- **src/app/new-product/new-product.component.spec.ts**  
Fichier de test unitaire pour le composant.

## • TYPESCRIPT (NEW-PRODUCT.COMPONENT.TS)

```
@Component({
  selector: 'app-new-product',
  standalone: true,
  imports: [
    ReactiveFormsModule,
    JsonPipe
  ],
  templateUrl: './new-product.component.html',
  styleUrls: ['./new-product.component.css']
})
export class NewProductComponent implements OnInit
{
  public productForm!: FormGroup;
  no usages
  constructor(private fb: FormBuilder, private productService: ProductService)
  {
  }
  no usages
  ngOnInit(): void
  {
    this.productForm=this.fb.group( controls: {
      name: this.fb.control( formState: '' , validatorOrOpts: [Validators.required]),
      price : this.fb.control( formState: 0),
      checked : this.fb.control( formState: false),
    });
  }
}
```

```
saveProduct() : void
{
  let product:Product=this.productForm.value;
  this.productService.saveProduct(product).subscribe( observerOrNext: {
    next : data : Product  =>{
      alert(JSON.stringify(data));
    }, error : err => {
      console.log(err);
    }
  });
}
```

La classe implémente OnInit pour initialiser les données au chargement du composant.

## Constructeur

Le constructeur injecte les services nécessaires :

- FormBuilder : Pour créer et gérer les formulaires réactifs.
- ProductService : Pour les opérations sur les produits.

## Méthodes :

- ngOnInit : Cette méthode est appelée après l'initialisation du composant et initialise le formulaire du produit.
- saveProduct : Cette méthode est appelée lors de la soumission du formulaire pour sauvegarder le produit.

Le NewProductComponent fournit une interface utilisateur pour créer un nouveau produit via un formulaire réactif. Le composant valide les entrées utilisateur et envoie les données du produit au serveur pour les sauvegarder. Le composant utilise les bonnes pratiques d'Angular pour la gestion des formulaires et l'interaction avec les services.

## • TEMPLATE HTML (NEW-PRODUCT.COMPONENT.HTML)

```
<div class="p-3">
  <div class="card">
    <div class="card-body">
      <div class="row">
        <div class="col-md-6">
          <form [formGroup]="productForm" (ngSubmit)="saveProduct()">
            <div class="mb-3">
              <label class="form-label">Name</label>
              <input class="form-control" formControlName="name">
            </div>
            <div class="mb-3">
              <label class="form-label">Price</label>
              <input class="form-control" formControlName="price">
            </div>
            <div class="mb-3">
              <label class="form-label">Checked</label>
              <input type="checkbox" class="form-check-input" formControlName="checked">
            </div>
            <button [disabled]="productForm.invalid" class="btn btn-success">Save</button>
          </form>
        </div>
      </div>
    </div>
  </div>
</div>
```

Ce code HTML fournit une interface utilisateur propre et intuitive pour créer un nouveau produit, avec des champs de formulaire pour le nom, le prix et l'état vérifié, ainsi qu'un bouton de soumission. Les classes Bootstrap et les directives Angular sont utilisées pour la mise en page et la gestion des formulaires.

The screenshot shows a user interface for creating a new product. It consists of a card with three input fields and a save button. The first field is labeled "Name" and contains an empty input field. The second field is labeled "Price" and contains the value "0". The third field is labeled "Checked" and contains a checkbox that is unchecked. Below these fields is a green "Save" button.

localhost:4200 indique

```
{"name": "lenovo", "price": "15220", "checked": true, "id": "OOOlyMC"}
```

Pages: 1 Si

OK

Name  
lenovo

Price  
15220

Checked

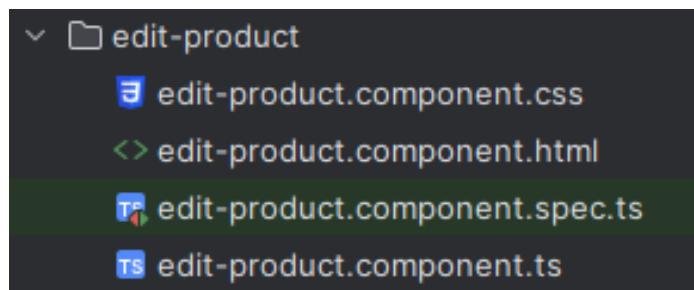
Save

## 6.4 EDIT-PRODUCT

- CRÉATION DÉ COMPOSANT

```
a_Nohayla_Tp4\Anoada_Nohayla_Tp4_1> ng generate component edit-product
CREATE src/app/edit-product/edit-product.component.html (28 bytes)
CREATE src/app/edit-product/edit-product.component.spec.ts (655 bytes)
CREATE src/app/edit-product/edit-product.component.ts (269 bytes)
CREATE src/app/edit-product/edit-product.component.css (0 bytes)
```

Cette commande vous permet de générer un composant angulaire nommé "edit-product" à l'aide de Angular CLI  
la commande génère les fichiers suivants :



- **src/app/edit-product/edit-product.component.ts:**  
Fichier TypeScript pour la logique du composant.
- **src/app/edit-product/edit-product.component.html**  
Template HTML pour le composant.
- **src/app/edit-product/edit-product.component.css)**  
Feuille de style pour le composant.
- **src/app/edit-product/edit-product.component.spec.ts**  
Fichier de test unitaire pour le composant.

## • TYPESCRIPT (EDIT-PRODUCT.COMPONENT.TS)

```
@Component({
  selector: 'app-edit-product',
  standalone: true,
  imports: [
    ReactiveFormsModule,
    NgIf
  ],
  templateUrl: './edit-product.component.html',
  styleUrls: ['./edit-product.component.css'
})
export class EditProductComponent implements OnInit{
  productId!: number;
  productFormGroup!: FormGroup;
  no usages
  constructor(private activatedRoute: ActivatedRoute,
              private productService: ProductService,
              private fb : FormBuilder) {
  }
  no usages
  ngOnInit(): void {
    this.productId=this.activatedRoute.snapshot.params['id'];
    this.productService.getProductById(this.productId).subscribe( observerOrNext: {
      next: (product : Product ) : void =>{
        this.productFormGroup= this.fb.group( controls: {
          id : this.fb.control(product.id),
          name : this.fb.control(product.name, validatorOrOpts: [Validators.required]),
          price : this.fb.control(product.price, validatorOrOpts: [Validators.min( min: 100)]),
          checked : this.fb.control(product.checked),
        });
      },
      error : err => {
        console.log(err);
      }
    });
  }
  1+ usages
  updateProduct(): void {
    let product : Product = this.productFormGroup.value;
    this.productService.updateProduct(product).subscribe( observerOrNext: {
      next : data:Product =>{
        alert(JSON.stringify(data));
      }
    });
  }
}
```

**La classe implémente OnInit pour initialiser les données au chargement du composant.**

### **Constructeur**

**Le constructeur injecte les services nécessaires :**

- **ActivatedRoute:** Pour accéder aux paramètres de la route et obtenir l'ID du produit à éditer.
- **ProductService:** Pour les opérations sur les produits.
- **FormBuilder:** Pour créer et gérer les formulaires réactifs.

### **Méthodes :**

- **ngOnInit :** Cette méthode est appelée après l'initialisation du composant et initialise le formulaire du produit en récupérant les données du produit par ID.
- **updateProduct :** Cette méthode est appelée lors de la soumission du formulaire pour mettre à jour le produit.

**Le EditProductComponent fournit une interface utilisateur pour modifier un produit existant via un formulaire réactif. Le composant valide les entrées utilisateur et envoie les données mises à jour du produit au serveur pour les sauvegarder. Le composant utilise les bonnes pratiques d'Angular pour la gestion des formulaires et l'interaction avec les services.**

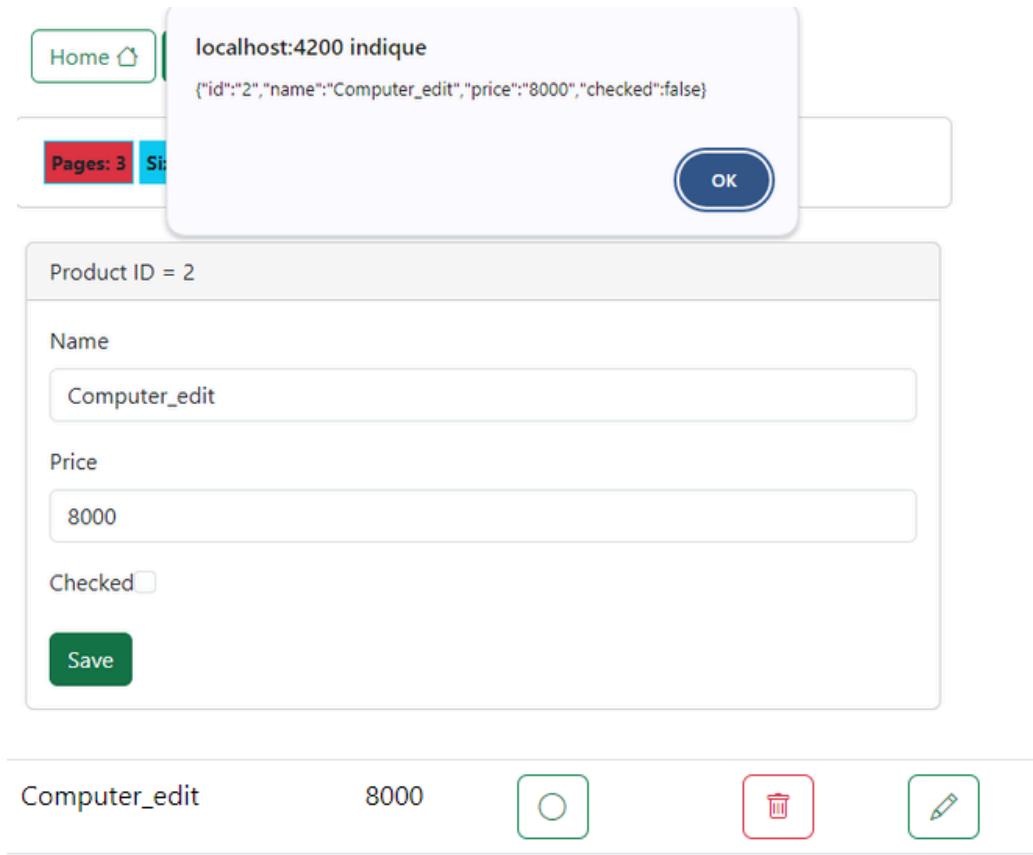
## • TEMPLATE HTML (EDIT-PRODUCT.COMPONENT.HTML)

```
<div class="p-3">
  <div class="card">
    <div class="card-header">
      Product ID = {{productId}}
    </div>
    <div class="card-body" *ngIf="productFormGroup">
      <form [formGroup]="productFormGroup" (ngSubmit)="updateProduct()">
        <div class="mb-3">
          <label class="form-label">Name</label>
          <input class="form-control" formControlName="name">
        </div>
        <div class="mb-3">
          <label class="form-label">Price</label>
          <input class="form-control" formControlName="price">
        </div>
        <div class="mb-3">
          <label class="form-label">Checked</label>
          <input type="checkbox" class="form-check-input" formControlName="checked">
        </div>
        <button [disabled]="productFormGroup.invalid" class="btn btn-success">Save</button>
      </form>
    </div>
  </div>
</div>
```



Product ID = 2

Name	<input type="text" value="Computer"/>
Price	<input type="text" value="7000"/>
Checked	<input checked="" type="checkbox"/>
<input type="button" value="Save"/>	



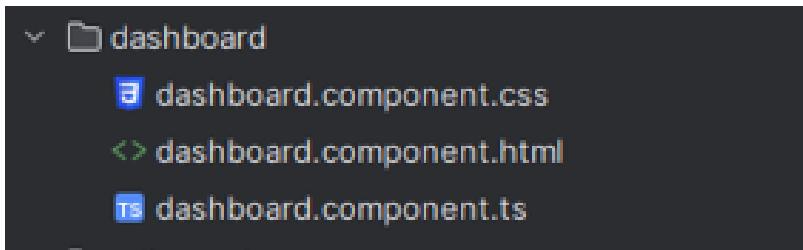
Ce code HTML fournit une interface utilisateur propre et intuitive pour modifier un produit existant, avec des champs de formulaire pour le nom, le prix et l'état vérifié, ainsi qu'un bouton de soumission. Les classes Bootstrap et les directives Angular sont utilisées pour la mise en page et la gestion des formulaires.

## 6.5 DASHBOARD

### • CRÉATION DE COMPOSANT

```
ada_Nohayla_Tp4\Anoada_Nohayla_Tp4_1> ng g c dashboard
CREATE src/app/dashboard/dashboard.component.html (25 bytes)
CREATE src/app/dashboard/dashboard.component.spec.ts (640 bytes)
CREATE src/app/dashboard/dashboard.component.ts (258 bytes)
CREATE src/app/dashboard/dashboard.component.css (0 bytes)
```

Cette commande vous permet de générer un composant angulaire nommé "dashboard" à l'aide de Angular CLI  
la commande génère les fichiers suivants :



- **src/app/dashboard/dashboard.component.ts:**  
Fichier TypeScript pour la logique du composant.
  - **src/app/dashboard/dashboard.component.html**  
Template HTML pour le composant.
  - **src/app/dashboard/dashboard.component.css)**  
Feuille de style pour le composant.
  - **src/app/dashboard/dashboard.component.spec.ts**  
Fichier de test unitaire pour le composant.
- **TYPESCRIPT (DASHBOARD.COMPONENT.TS)**

```
@Component({
  selector: 'app-dashboard',
  standalone: true,
  imports: [],
  templateUrl: './dashboard.component.html',
  styleUrls: ['./dashboard.component.css'
})
export class DashboardComponent
{
  no usages
  constructor(public appState:AppStateService)
  {}
  1+ usages
  totalCheckedProducts()
  {
    let checkedProducts = this.appState.productsState.products.filter((p:any)=>p.checked==true);
    return checkedProducts.length;
  }
}
```

## Constructeur

Le constructeur Injecte le service AppStateService pour accéder à l'état des produits.

## Méthodes :

- **totalCheckedProducts:** Cette méthode filtre les produits pour ne garder que ceux qui sont vérifiés (checked == true), puis il retourne le nombre de produits vérifiés.

## • TEMPLATE HTML (DASHBOARD.COMPONENT.HTML)

```
<small>
  <nav class="p-1 m-1 text-white fw-bold">
    <div class="card">
      <div class="card-body">
        <ul class="nav nav-pills">
          <li>
            <div class="border border-info bg-danger p-1 ms-1">
              Pages: {{appState.productsState.totalPages}}
            </div>
          </li>
          <li>
            <div class="border border-info bg-info p-1 ms-1">
              Size: {{appState.productsState.pageSize}}
            </div>
          </li>
          <li>
            <div class="border border-info bg-success p-1 ms-1">
              Total: {{appState.productsState.totalProducts}}
            </div>
          </li>
          <li>
            <div class="border border-info bg-primary p-1 ms-1">
              Checked: {{totalCheckedProducts()}}
            </div>
          </li>
        </ul>
      </div>
    </div>
  </nav>
</small>
```

Pages: 3    Size: 3    Total: 9    Checked: 2

Ce code HTML crée une section de tableau de bord compact pour afficher des informations sur les produits, telles que le nombre total de pages, la taille des pages, le nombre total de produits et le nombre de produits vérifiés. Il utilise Bootstrap pour le style et Angular pour l'interaction dynamique avec les données.

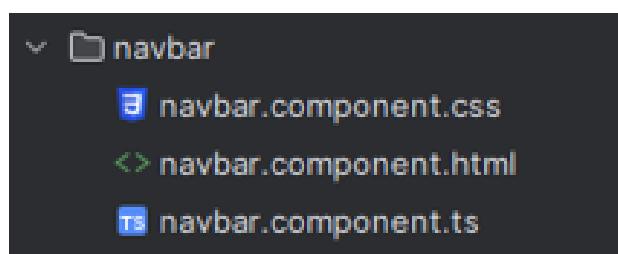
## 6.6 NAVBAR

### • CRÉATION DE COMPOSANT

```
ada_Nohayla_Tp4\Anoada_Nohayla_Tp4_1> ng g c navbar
CREATE src/app/navbar/navbar.component.html (22 bytes)
CREATE src/app/navbar/navbar.component.spec.ts (619 bytes)
CREATE src/app/navbar/navbar.component.css (0 bytes)
```

Cette commande vous permet de générer un composant angulaire nommé "navbar" à l'aide de Angular CLI

la commande génère les fichiers suivants :



- **src/app/navbar/navbar.component.ts:**  
Fichier TypeScript pour la logique du composant.
- **src/app/navbar/navbar.component.html**  
Template HTML pour le composant.
- **src/app/navbar/navbar.component.css)**  
Feuille de style pour le composant.
- **src/app/navbar/navbar.component.spec.ts**  
Fichier de test unitaire pour le composant.
- **TYPESCRIPT (NAVBAR.COMPONENT.TS)**

```
@Component({
  selector: 'app-navbar',
  standalone: true,
  imports: [
    NgForOf,
    RouterLink,
    NgIf,
    AsyncPipe
  ],
  templateUrl: './navbar.component.html',
  styleUrls: ['./navbar.component.css']
})
```

```

export class NavbarComponent {
  actions : Array<any> = [
    {title : "Home", "route":"/admin/home", icon : "house"},
    {title : "Products", "route":"/admin/products", icon : "search"},
    {title : "New product", "route":"/admin/newProduct", icon : "safe"}
  ]
  currentAction : any;
  public isLoading :boolean=false;
  no usages
  constructor(public appState :AppStateService ,
              public loadingService : LoadingService) {
    /*this.loadingService.isLoading$.subscribe({
      next : (value)=>{
        this.isLoading=value;
      }
    })*/
  }
  1+ usages
  setCurrentAction(action: any) : void  {
    this.currentAction = action;
  }
}

```

### Propriétés :

- **actions** : Tableau des actions disponibles dans la barre de navigation, avec leur titre, route et icône.
- **currentAction** : Action actuellement sélectionnée.
- **isLoading** : État de chargement, initialisé à false.

### Constructeur :

- Injecte les services AppStateService et LoadingService.

### Méthode

- **setCurrentAction** : Met à jour l'action actuellement sélectionnée.

Le NavbarComponent fournit une barre de navigation dynamique et réactive avec des liens vers différentes sections de l'application. Il affiche également un indicateur de chargement lorsque des opérations asynchrones sont en cours. Le composant utilise les bonnes pratiques d'Angular pour la gestion des états et des routes, et Bootstrap pour le style.

- **TEMPLATE HTML  
(NAVBAR.COMPONENT.HTML)**

```
<nav class="p-3">
  <ul class="nav nav-pills">
    <li *ngFor="let action of actions">
      <button
        (click)="setCurrentAction(action)"
        routerLink="{{action.route}}"
        [class] = "action == currentAction? 'btn btn-success ms-1' : 'btn btn-outline-success ms-1'"
      >
        {{action.title}}
        <i class="bi bi-{{action.icon}}"></i>
      </button>

    </li>
    <li *ngIf="loadingService.isLoading$ | async">
      <div class="spinner-border text-primary ms-2" role="status">
        </div>
    </li>
  </ul>
</nav>
```



Ce code HTML fournit une barre de navigation dynamique et réactive qui permet de naviguer entre différentes sections de l'application. Il affiche également un indicateur de chargement lorsque des opérations asynchrones sont en cours. Les classes Bootstrap sont utilisées pour le style et la mise en page, et les directives Angular telles que \*ngFor et \*ngIf sont utilisées pour générer dynamiquement les éléments de la liste et afficher conditionnellement le spinner de chargement.

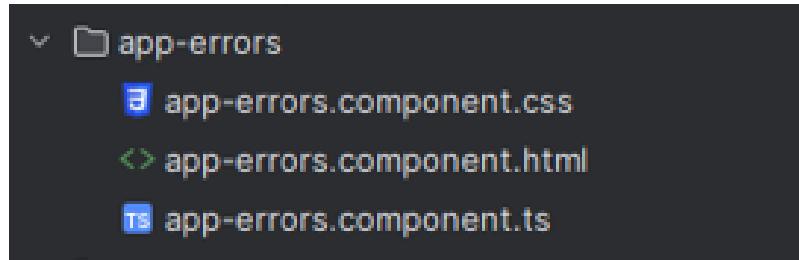
## 6.7 APP-ERRORS

- **CRÉATION DÉ COMPOSANT**

```
ada_Nohayla_Tp4\Anoada_Nohayla_Tp4_1> ng g c app-errors
CREATE src/app/app-errors/app-errors.component.html (26 bytes)
CREATE src/app/app-errors/app-errors.component.spec.ts (641 bytes)
CREATE src/app/app-errors/app-errors.component.ts (261 bytes)
CREATE src/app/app-errors/app-errors.component.css (0 bytes)
```

Cette commande vous permet de générer un composant angulaire nommé "app-errors" à l'aide de Angular CLI

la commande génère les fichiers suivants :



- **src/app/app-errors/app-errors.component.ts:**  
Fichier TypeScript pour la logique du composant.
- **src/app/app-errors/app-errors.component.html**  
Template HTML pour le composant.
- **src/app/app-errors/app-errors.component.css)**  
Feuille de style pour le composant.
- **src/app/app-errors/app-errors.component.spec.ts**  
Fichier de test unitaire pour le composant.
- **TYPESCRIPT (APP-ERRORS.COMPONENT.TS)**

```
@Component({
  selector: 'app-app-errors',
  standalone: true,
  imports: [
    JsonPipe,
    NgIf
  ],
  templateUrl: './app-errors.component.html',
  styleUrls: ['./app-errors.component.css']
})
export class AppErrorsComponent
{
  no usages
  constructor(public appState : AppStateService)
  {
  }
}
```

### Constructeur :

- Injecte le service AppStateService pour accéder à l'état des produits.

Le AppErrorsComponent fournit une interface utilisateur simple pour afficher les messages d'erreur lorsque l'état des produits est en erreur. Il utilise les bonnes pratiques d'Angular pour la gestion des états et des conditions.

- TEMPLATE HTML  
(APP-ERRORS.COMPONENT.HTML)

```
<div class="p-3" *ngIf="appState.productsState.status=='ERROR'>
  <div class="alert alert-danger">
    {{appState.productsState.errorMessage.message}}
  </div>
</div>
```

Http failure response for http://localhost:8089/products?  
name\_like=&\_page=1&\_limit=3: 0 undefined

Ce code HTML fournit une manière simple et efficace d'afficher les messages d'erreur lorsque l'état des produits est en erreur. Il utilise les bonnes pratiques d'Angular pour la gestion conditionnelle de l'affichage et Bootstrap pour le style des alertes. Lorsque l'état des produits passe à ERROR, le composant affiche le message d'erreur correspondant dans une alerte rouge, offrant ainsi une expérience utilisateur claire et intuitive.

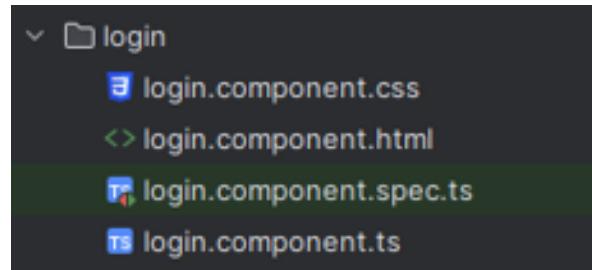
## 6.8 LOGIN

- CRÉATION DÉ COMPOSANT

```
\Anoada_Nohayla_Tp4\Anoada_Nohayla_Tp4_1> ng g c login
CREATE src/app/login/login.component.html (21 bytes)
CREATE src/app/login/login.component.spec.ts (612 bytes)
CREATE src/app/login/login.component.ts (242 bytes)
CREATE src/app/login/login.component.css (0 bytes)
```

Cette commande vous permet de générer un composant angulaire nommé "login" à l'aide de Angular CLI

la commande génère les fichiers suivants :



- **src/app/login/login.component.ts:**  
**Fichier TypeScript pour la logique du composant.**
  - **src/app/login/login.component.html**  
**Template HTML pour le composant.**
  - **src/app/login/login.component.css)**  
**Feuille de style pour le composant.**
  - **src/app/login/login.component.spec.ts**  
**Fichier de test unitaire pour le composant.**
- 
- **TYPESCRIPT (LOGIN.COMPONENT.TS)**

```
@Component({
  selector: 'app-login',
  standalone: true,
  imports: [
    ReactiveFormsModule
  ],
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css'
})
export class LoginComponent implements OnInit
{
  formLogin! : FormGroup;
  no usages
  constructor(private fb :FormBuilder,private router:Router)
  {
  }
  no usages
  ngOnInit() : void
  {
    this.formLogin=this.fb.group( controls: {
      username : this.fb.control( formState: ""),
      password : this.fb.control( formState: "")
    })
  }
}
```

```

handlelogin() : void
{
  console.log(this.formLogin.value);
  if(this.formLogin.value.username=="admin" && this.formLogin.value.password=="1234")
  {
    this.router.navigateByUrl(url: "/admin");
  }
}

```

## Propriété

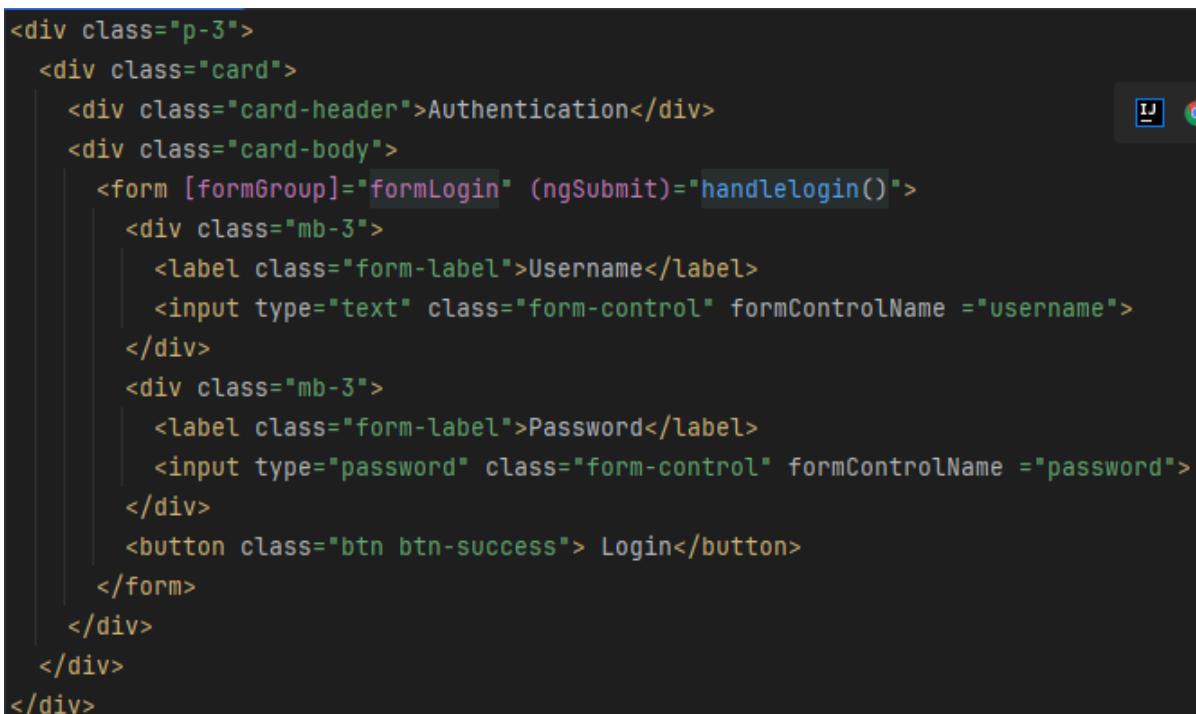
- **formLogin** : Formulaire réactif pour gérer les entrées utilisateur.

## Constructeur

- Injecte FormBuilder pour créer le formulaire et Router pour la navigation.

## Méthode

- **ngOnInit** : Initialise le formulaire avec des champs pour username et password.
- **handlelogin** : Gère la soumission du formulaire. Si les informations d'identification sont correctes, redirige l'utilisateur vers la page d'administration.
- **TEMPLATE HTML (LOGIN.COMPONENT.HTML)**



```

<div class="p-3">
  <div class="card">
    <div class="card-header">Authentication</div>
    <div class="card-body">
      <form [formGroup]="formLogin" (ngSubmit)="handlelogin()">
        <div class="mb-3">
          <label class="form-label">Username</label>
          <input type="text" class="form-control" formControlName ="username">
        </div>
        <div class="mb-3">
          <label class="form-label">Password</label>
          <input type="password" class="form-control" formControlName ="password">
        </div>
        <button class="btn btn-success"> Login</button>
      </form>
    </div>
  </div>
</div>

```

The screenshot shows a clean, modern login interface. At the top, the word "Authentication" is displayed. Below it are two input fields: one for "Username" and one for "Password". Both fields have placeholder text ("Enter Username" and "Enter Password" respectively). A large green button labeled "Login" is centered below the password field.

Ce code HTML fournit un formulaire de connexion élégant et fonctionnel utilisant Angular Reactive Forms pour la gestion des entrées utilisateur et Bootstrap pour le style. Le formulaire comprend des champs pour le nom d'utilisateur et le mot de passe, et un bouton pour soumettre les informations de connexion. Lorsque l'utilisateur soumet le formulaire, la méthode handlelogin() est appelée pour gérer les informations de connexion.

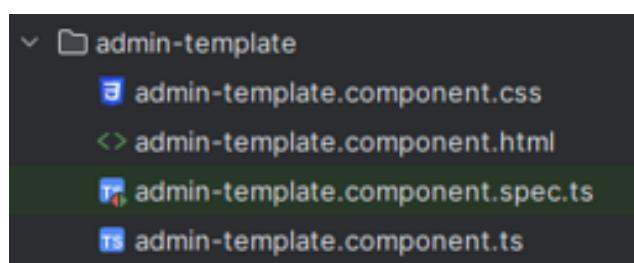
## 6.9 ADMIN-TEMPLATE

### • CRÉATION DE COMPOSANT

```
noada_Nohayla_Tp4\Anoada_Nohayla_Tp4_1> ng g c admin-template
CREATE src/app/admin-template/admin-template.component.html (30 bytes)
CREATE src/app/admin-template/admin-template.component.spec.ts (669 bytes)
CREATE src/app/admin-template/admin-template.component.ts (277 bytes)
CREATE src/app/admin-template/admin-template.component.css (0 bytes)
```

Cette commande vous permet de générer un composant angulaire nommé "admin-template" à l'aide de Angular CLI

la commande génère les fichiers suivants :



- **src/app/admin-template/admin-template.component.ts:**  
Fichier TypeScript pour la logique du composant.
- **src/app/admin-template/admin-template.component.html**  
Template HTML pour le composant.
- **src/app/admin-template/admin-template.component.css)**  
Feuille de style pour le composant.
- **src/app/admin-template/admin-template.component.spec.ts**  
Fichier de test unitaire pour le composant.

## • **TYPESCRIPT (ADMIN-TEMPLATE.COMPONENT.TS)**

```
@Component({
  selector: 'app-admin-template',
  standalone: true,
  imports: [
    AppErrorsComponent,
    DashboardComponent,
    NavbarComponent,
    RouterOutlet
  ],
  templateUrl: './admin-template.component.html',
  styleUrls: ['./admin-template.component.css']
})
export class AdminTemplateComponent
```

## • **TEMPLATE HTML (ADMIN-TEMPLATE.COMPONENT.HTML)**

```
<app-navbar></app-navbar>

<app-dashboard></app-dashboard>

<app-app-errors></app-app-errors>

<router-outlet></router-outlet>
```

Name	Price	Checked
Computer	8000	<input checked="" type="checkbox"/>
Computer_edit	8000	<input type="checkbox"/>
Smart phone	3000	<input type="checkbox"/>

Le composant AdminTemplateComponent sert de modèle principal pour l'interface d'administration de l'application. Il inclut d'autres composants pour la barre de navigation, le tableau de bord et les erreurs, ainsi qu'un espace pour le contenu dynamique géré par le routeur Angular.

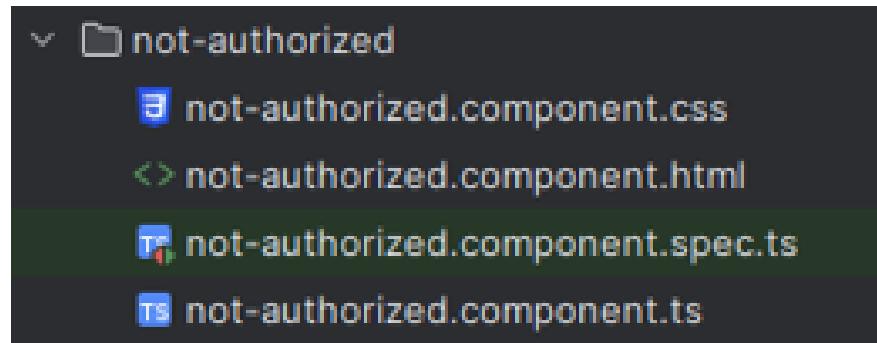
## 6.10 NOT-AUTHORIZED

- CRÉATION DE COMPOSANT

```
ada_Nohayla_Tp4\Anoada_Nohayla_Tp4_1> ng g c not-authorized
CREATE src/app/not-authorized/not-authorized.component.html (30 bytes)
CREATE src/app/not-authorized/not-authorized.component.spec.ts (669 bytes)
CREATE src/app/not-authorized/not-authorized.component.ts (277 bytes)
CREATE src/app/not-authorized/not-authorized.component.css (0 bytes)
```

Cette commande vous permet de générer un composant angulaire nommé "not-authorized" à l'aide de Angular CLI

la commande génère les fichiers suivants :



- **src/app/not-authorized/not-authorized.component.ts:**  
Fichier TypeScript pour la logique du composant.
  - **src/app/not-authorized/not-authorized.component.html**  
Template HTML pour le composant.
  - **src/app/not-authorized/not-authorized.component.css)**  
Feuille de style pour le composant.
  - **src/app/not-authorized/not-authorized.component.spec.ts**  
Fichier de test unitaire pour le composant.
- **TYPESCRIPT  
(NOT-AUTHORIZED.COMPONENT.TS)**

```
@Component({
  selector: 'app-not-authorized',
  standalone: true,
  imports: [],
  templateUrl: './not-authorized.component.html',
  styleUrls: ['./not-authorized.component.css']
})
export class NotAuthorizedComponent
```

- TEMPLATE HTML  
(NOT-AUTHORIZED.COMPONENT.HTML)

```
<div class="p-3">
  <div class="card">
    <div class="card-body">
      <div class="alert alert-danger">
        You are not authorized
      </div>
    </div>
  </div>
</div>
```



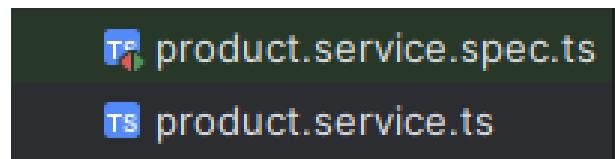
Le NotAuthorizedComponent fournit une interface utilisateur simple et efficace pour afficher un message d'alerte lorsque l'utilisateur n'est pas autorisé à accéder à une certaine partie de l'application. Il utilise Bootstrap pour le style et Angular pour la gestion du composant. Le composant est autonome et peut être facilement intégré dans d'autres parties de l'application pour gérer les erreurs d'autorisation.

# 7. SERVICES

## 7.1 PRODUCT

```
\Anoada_Nohayla_Tp4\Anoada_Nohayla_Tp4_1> ng g s services/product
CREATE src/app/services/product.service.spec.ts (378 bytes)
CREATE src/app/services/product.service.ts (145 bytes)
```

Cette commande va créer deux fichiers :



- **product.service.ts** : Contient la définition du service.
- **product.service.spec.ts** : Contient les tests unitaires pour le service.

## PRODUCT.SERVICE.TS

```
@Injectable({
  providedIn: 'root'
})
export class ProductService
{
  private host : string="http://localhost:8089";
  no usages
  constructor(private http : HttpClient)
  { }
  1+ usages
  public searchProducts(keyword:string="",page :number=1, size:number=4) : Observable<HttpResponse<Object...>>
  {
    return this.http.get(`url: ${this.host}/products?name_like=${keyword}&_page=${page}&_limit=${size}` , options: {observe:'response'});
  }
  1+ usages
  public checkProducts(product : Product):Observable<Product>
  {
    return this.http.patch<Product>( url: `${this.host}/products/${product.id}` ,
      body: { checked: !product.checked });
  }
  1+ usages
  public deleteProducts(product : Product) : Observable<any>
  {
    return this.http.delete<any>( url: `${this.host}/products/${product.id}`);
  }
  1+ usages
  saveProduct(product: Product):Observable<Product>
  {
    return this.http.post<Product>( url: `${this.host}/products` ,product);
  }
  1+ usages
  getProductById(productId: number): Observable<Product>
  {
    return this.http.get<Product>( url: `${this.host}/products/${productId}`);
  }
  1+ usages
  updateProduct(product: Product): Observable<Product>
  {
    return this.http.put<Product>( url: `${this.host}/products/${product.id}` ,product);
  }
}
```

### Propriété host :

- URL de base de l'API.

### Constructeur :

- Injecte le service HttpClient pour effectuer des requêtes HTTP.

### Méthodes :

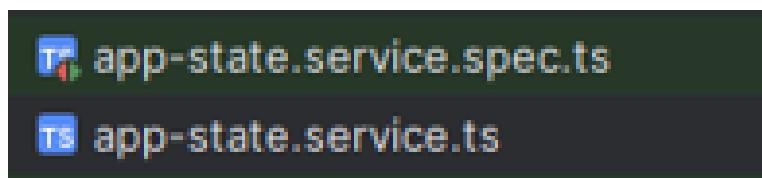
- searchProducts : Rechercher des produits en fonction d'un mot-clé, d'une page et d'une taille de page.
- checkProducts : Vérifier (ou dévérerifier) un produit en inversant son état checked.
- deleteProducts : Supprimer un produit en fonction de son ID.
- saveProduct : Sauvegarder un nouveau produit.
- getProductById : Obtenir un produit en fonction de son ID.
- updateProduct : Mettre à jour un produit existant.

Le ProductService est un service complet pour gérer les produits via une API RESTful. Il permet de rechercher, vérifier, supprimer, sauvegarder, obtenir et mettre à jour des produits. En utilisant HttpClient, il envoie des requêtes HTTP et utilise des observables pour gérer les réponses de manière asynchrone. Ce service est essentiel pour la communication entre le front-end Angular et l'API back-end, assurant une gestion efficace des produits.

## 7.2 APP-STATE

```
ada_Nohayla_Tp4\Anoada_Nohayla_Tp4_1> ng g s services/app-state
CREATE src/app/services/app-state.service.spec.ts (384 bytes)
CREATE src/app/services/app-state.service.ts (146 bytes)
```

Cette commande va créer deux fichiers :



- app-state.service.ts : Contient la définition du service.
- app-state.service.spec.ts : Contient les tests unitaires pour le service.

## APP-STATE.SERVICE.TS

```
@Injectable({
  providedIn: 'root'
})
export class AppStateService {
  public productsState :any={
    products:[],
    keyword : "",
    totalPages:0,
    pageSize:3,
    currentPage :1,
    totalProducts :0,
    status : "",
    errorMessage : ""
  }
  public authState :any ={
    isAuthenticated : false,
    username : undefined,
    roles : undefined,
    token : undefined
  }
  no usages
  constructor() { }
  1+ usages
  public setProductState(state :any):void {
    this.productsState={...this.productsState, ...state}
  }
  1+ usages
  public setAuthState(state : any) :void{
    this.authState={...this.authState, ...state};
  }
}
```

### Propriété

**productsState** : Objet contenant l'état des produits avec les propriétés suivantes :

- **products** : Tableau des produits.
- **keyword** : Mot-clé de recherche.
- **totalPages** : Nombre total de pages de résultats.
- **pageSize** : Taille de chaque page de résultats.
- **currentPage** : Page actuelle.
- **totalProducts** : Nombre total de produits.
- **status** : Statut de l'état des produits (e.g., "LOADING", "LOADED", "ERROR").
- **errorMessage** : Message d'erreur si une erreur se produit.

### Constructeur :

- Le constructeur est vide car aucune initialisation spécifique n'est nécessaire.

### Méthode

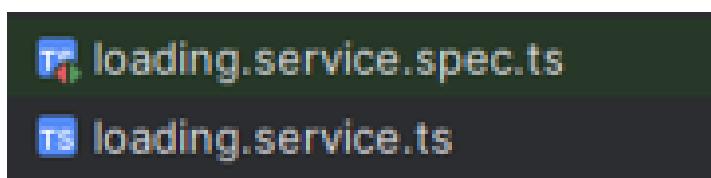
- `setProductState` : Permet de mettre à jour l'état des produits. Elle prend un objet `state` en argument et fusionne ses propriétés avec l'état actuel en utilisant la décomposition d'objet (`{ ...this.productsState, ...state }`).

Le `AppStateService` est un service essentiel pour gérer l'état global de l'application concernant les produits. Il centralise les informations relatives aux produits et fournit une méthode pour mettre à jour cet état. Cela permet à différents composants de l'application de lire et de modifier l'état des produits de manière synchronisée.

## 7.3 LOADING

```
ada_Nohayla_Tp4\Anoada_Nohayla_Tp4_1> ng g s services/loading
CREATE src/app/services/loading.service.spec.ts (378 bytes)
CREATE src/app/services/loading.service.ts (145 bytes)
```

Cette commande va créer deux fichiers :



- `loading.service.ts` : Contient la définition du service.
- `loading.service.spec.ts` : Contient les tests unitaires pour le service.

# LOADING.SERVICE.SPEC.TS

```
@Injectable({
  providedIn: 'root'
})
export class LoadingService {
  public isLoading$ : Subject<boolean> = new Subject<boolean>();
no usages
constructor() { }
1+ usages
showLoadingSpinner():void
{
  this.isLoading$.next( value: true);
}
1+ usages
hideLoadingSpinner():void
{
  this.isLoading$.next( value: false);
}
}
```

## Propriété

- `isLoading$` : Un `Subject` de type `boolean` utilisé pour diffuser des événements de chargement. Les composants peuvent s'abonner à `isLoading$` pour réagir aux changements de l'état de chargement.

## Constructeur

- Le constructeur est vide car aucune initialisation spécifique n'est nécessaire.

## Méthodes

- `showLoadingSpinner` : Appelée pour indiquer que le chargement commence. Elle émet `true` via `isLoading$`.
- `hideLoadingSpinner` : Appelée pour indiquer que le chargement est terminé. Elle émet `false` via `isLoading$`.

## 7.4 AUTH

```
ada_Nohayla_Tp4\Anoada_Nohayla_Tp4_1> ng g s auth
CREATE src/app/auth.service.spec.ts (363 bytes)
CREATE src/app/auth.service.ts (142 bytes)
```

Cette commande va créer deux fichiers :

 auth.service.spec.ts

 auth.service.ts

- **auth.service.ts** : Contient la définition du service.
- **auth.service.spec.ts** : Contient les tests unitaires pour le service.

### AUTH.SERVICE.SPEC.TS

```
@Injectable({
  providedIn: 'root'
})
export class AuthService {
  no usages
  constructor(private http: HttpClient, private appState: AppStateService)
  {}
  1+ usages
  async login(username: string, password: string): Promise<boolean>
  {
    let user: any = await firstValueFrom(this.http.get(url: "http://localhost:8089/users/" + username));
    console.log(password);
    console.log(user.password);
    console.log(atob(user.password));
    if (password == atob(user.password))
    {
      let decodedJwt: any = jwtDecode(user.token);
      this.appState.setAuthState({
        isAuthenticated: true,
        username: decodedJwt.sub,
        roles: decodedJwt.roles,
        token: user.token
      });
      return Promise.resolve(value: true);
    }
    else
    {
      return Promise.reject(reason: "Bad credentials");
    }
  }
}
```

### Propriété

- baseUrl : URL de base de l'API.

### Constructeur

- Injecte les services HttpClient et AppStateService.

### Méthode

- login : Gère l'authentification des utilisateurs.

Le AuthService est un service complet pour gérer l'authentification des utilisateurs dans une application Angular. Il effectue des requêtes HTTP pour récupérer les informations de l'utilisateur, vérifie le mot de passe, décode les tokens JWT, et met à jour l'état d'authentification de l'application. En utilisant HttpClient, firstValueFrom, et jwtDecode, le service offre une gestion efficace et sécurisée de l'authentification des utilisateurs.

## 8. INTERCEPTEURS

### APP-HTTP

Cette commande crée deux fichiers

```
ada_Nohayla_Tp4\Anoada_Nohayla_Tp4_1> ng g interceptor app-http
CREATE src/app/app-http.interceptor.spec.ts (506 bytes)
CREATE src/app/app-http.interceptor.ts (157 bytes)
```

- app-http.interceptor.ts : Contient la définition de l'intercepteur.
- app-http.interceptor.spec.ts : Contient les tests unitaires pour l'intercepteur.

### APP-HTTP.INTERCEPTOR.TS

```
@Injectable()
export class AppHttpInterceptor implements HttpInterceptor
{
  no usages
  constructor(private appState : AppStateService,
             private loadingService:LoadingService) {}
  no usages
  intercept(request: HttpRequest<unknown>, next: HttpHandler): Observable<HttpEvent<unknown>>
  {
```

```

    /*this.appState.setProductState({
      status :"LOADING"
    })*/
    this.loadingService.showLoadingSpinner();
    let req : HttpRequest<?> = request.clone( update: {
      headers : request.headers.set("Authorization","Bearer JWT")
    });
    return next.handle(req).pipe(
      finalize( callback: () : void =>{
        /*this.appState.setProductState({
          status : ""
        })*/
        this.loadingService.hideLoadingSpinner();
      })
    );
  }
}

```

L'intercepteur `AppHttpInterceptor` permet d'ajouter des en-têtes, de gérer des tokens d'authentification, et de gérer l'état de chargement de l'application. Après avoir généré l'intercepteur avec Angular CLI, vous pouvez le configurer dans votre module principal pour qu'il soit utilisé globalement dans votre application. Cela centralise et simplifie la gestion des requêtes HTTP et de l'état de l'application.

## 9. GARDES

### 9.1 AUTHENTICATION

Cette commande crée deux fichiers

```

ada_Nohayla_Tp4\Anoada_Nohayla_Tp4_1> ng g g guards/authentication
? Which type of guard would you like to create? CanActivate
CREATE src/app/guards/authentication.guard.spec.ts (518 bytes)
CREATE src/app/guards/authentication.guard.ts (143 bytes)

```

- `authentication.guard.ts` : Contient la définition du garde de route.
- `authentication.guard.spec.ts` : Contient les tests unitaires pour le garde de route.

## AUTHENTICATION.GUARD.TS

```
@Injectable({
  providedIn: 'root'
})
export class AuthenticationGuard {
  no usages
  constructor(private appState : AppStateService, private router : Router)
  {}
  no usages
  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree
  {
    if (this.appState.authState.isAuthenticated)
    {
      return true;
    }
    else
    {
      this.router.navigateByUrl(url: "/login");
      return false;
    }
  }
}
```

Le garde de route AuthenticationGuard vérifie si un utilisateur est authentifié avant de lui permettre d'accéder à certaines routes. En utilisant AppStateService pour vérifier l'état d'authentification et Router pour la redirection, ce garde assure que seules les personnes authentifiées peuvent accéder aux parties protégées de votre application. Après avoir généré le garde avec Angular CLI.

## 9.2 AUTHORIZATION

Cette commande crée deux fichiers

```
ada_Nohayla_Tp4\Anoada_Nohayla_Tp4_1> ng g g guards/authorization
? Which type of guard would you like to create? CanActivate
CREATE src/app/guards/authorization.guard.spec.ts (514 bytes)
CREATE src/app/guards/authorization.guard.ts (142 bytes)
```

- **authorization.guard.ts** : Contient la définition du garde de route.
- **authorization.guard.spec.ts** : Contient les tests unitaires pour le garde de route.

## AUTHORIZATION.GUARD.TS

```
@Injectable({
  providedIn: 'root'
})
export class AuthorizationGuard {
  no usages
  constructor(private appState : AppStateService, private router : Router ) {
  }
  no usages
  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree
  {
    if(this.appState.authState.roles.includes(route.data['requiredRoles']))
    {
      return true;
    }
    else
    {
      this.router.navigateByUrl(url: "/admin/notAuthorized")
      return false;
    }
  }
}
```

Le garde de route AuthorizationGuard vérifie si un utilisateur possède les rôles requis avant de lui permettre d'accéder à certaines routes protégées de l'application. En utilisant AppStateService pour vérifier les rôles d'autorisation et Router pour la redirection, ce garde assure que seules les personnes autorisées peuvent accéder aux parties protégées de votre application. Après avoir généré le garde avec Angular CL.