

Sushi Shop

Table of Contents

Présentation	1
Fonctionnalités	1
Utilisation	2
Codes	4
Produits	4
Détails	5
Panier	5
Service API	6
Service panier	7
App Routing	8
Box	9
Produit (box) HTML	9
Détails (box) HTML	10
Environnement API	10
Code de l'API	11
Diagramme	11
Diagramme d'utilisation	11
Diagramme des tiers	12
Structure JSON	12
Cybersécurité	13
Conclusion	14
Technologies utilisées	14
Contributeurs	14

Présentation

Ce projet consiste en une application front-end pour la prise de commande au niveau d'un point de vente de sushis. Elle utilise le framework Angular et une API existante pour présenter la gamme de produits à la vente.

Fonctionnalités

L'application permet à un utilisateur de passer commande de boxes de sushis en vue d'établir une commande transmissible à la production (cuisine). Voici les fonctionnalités implémentées dans ce projet :

- Une page "home" présentant le projet.

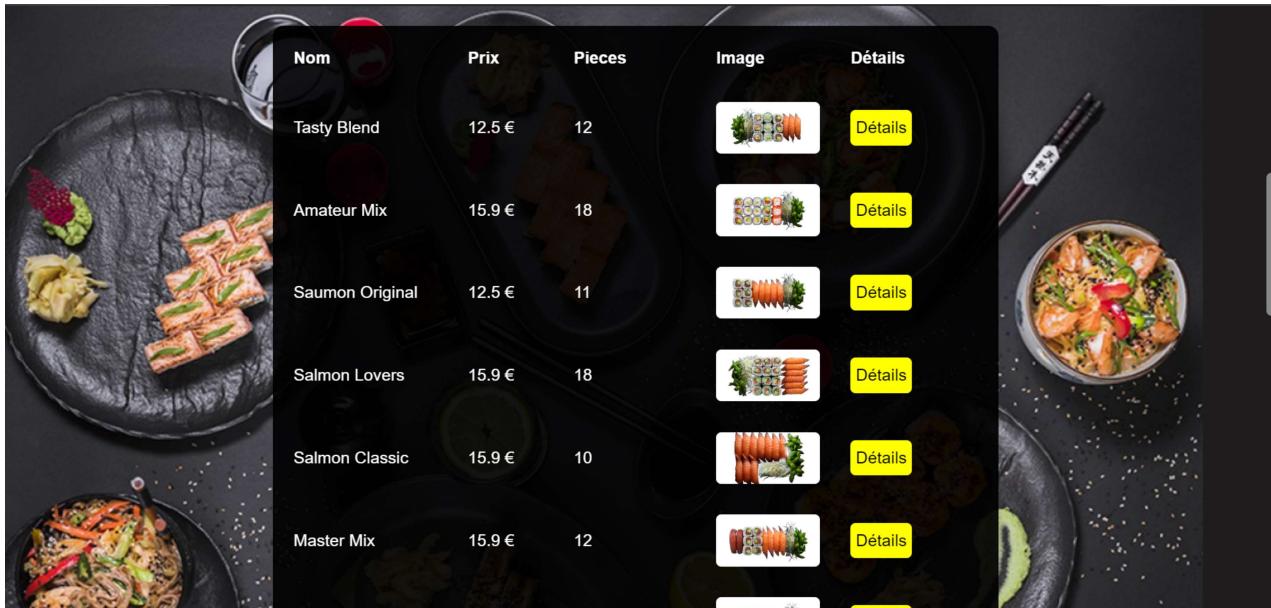
- Une page "produit" qui affiche tous les box dans l'API.
- Pour chaque box, un bouton permet de voir les détails de la box (page détail).
- Dans la page détail, un bouton permet de mettre une box dans notre panier (page panier).
- Dans la page panier, un bouton permet de commander (en théorie) les boxes.
- Accès à l'historique de nos commandes

Utilisation

Lorsque vous accédez à l'application, vous êtes redirigé vers la page d'accueil où vous trouverez une présentation du projet. Pour accéder à la liste des produits, cliquez sur "Produits" dans la barre de navigation.



Sur la page "Produits", vous verrez la liste des boxes de sushis disponibles à la vente. Pour voir les détails d'une box, cliquez sur le bouton "Détails" correspondant.



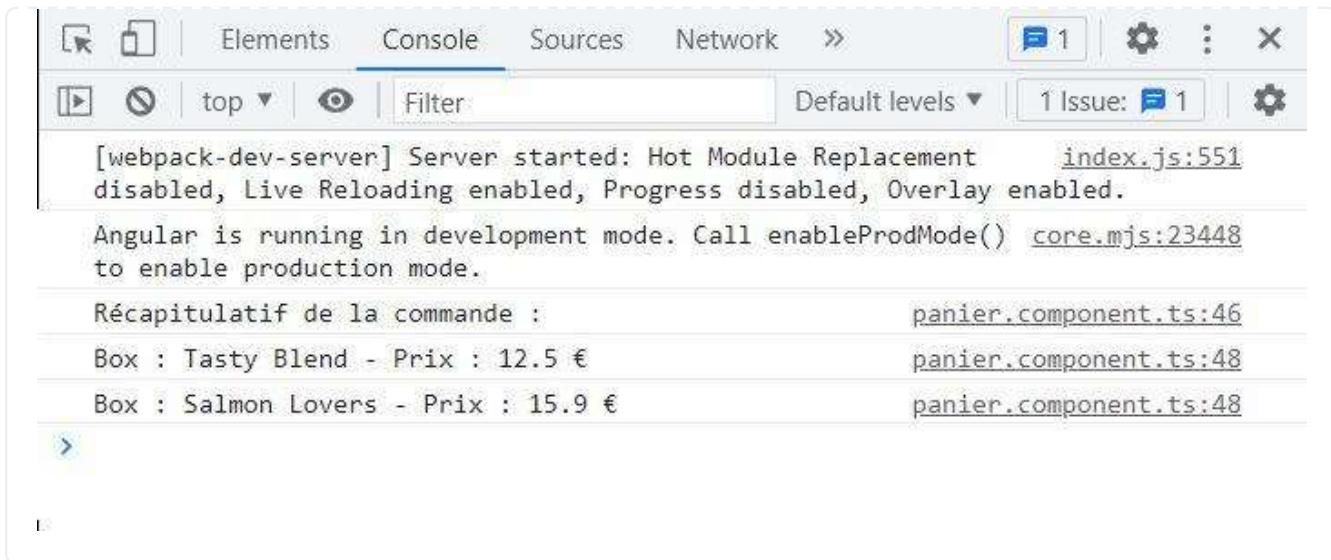
Sur la page "Détails", vous verrez les informations détaillées de la box de sushis sélectionnée. Vous pourrez également ajouter cette box à votre panier en cliquant sur le bouton "Ajouter au panier".

The screenshot shows a product detail page for a sushi box named "Tasty Blend" priced at 12.5 €. The page includes a list of ingredients: Saveurs: avocat, saumon, cheese; Aliments: California Saumon Avocat (3), Sushi Saumon (3), Spring Avocat Cheese (3), California pacific (3), Edamame/Salade de chou (1). A small image of the sushi is shown, along with an "Ajouter au panier" button. The background features a dark, artistic photograph of various sushi dishes.

Sur la page "Panier", vous verrez la liste des boxes de sushis que vous avez ajoutée à votre panier. Vous pourrez également passer votre commande (en théorie) en cliquant sur le bouton "Commander".

The screenshot shows a shopping cart page with two items: "Salmon Lovers" (15.9€) and "Tasty Blend" (12.5€). Each item has a "Supprimer" (Delete) button. To the right, a summary box shows "Coût des produits" (Cost of products) at 28.4€ and a green "Valider ma commande" (Validate my order) button. The background features a dark, artistic photograph of various sushi dishes.

Enfin, vous pourrez accéder à l'historique de vos commandes en cliquant sur "Historique" dans la barre de navigation. (affichage à finaliser)



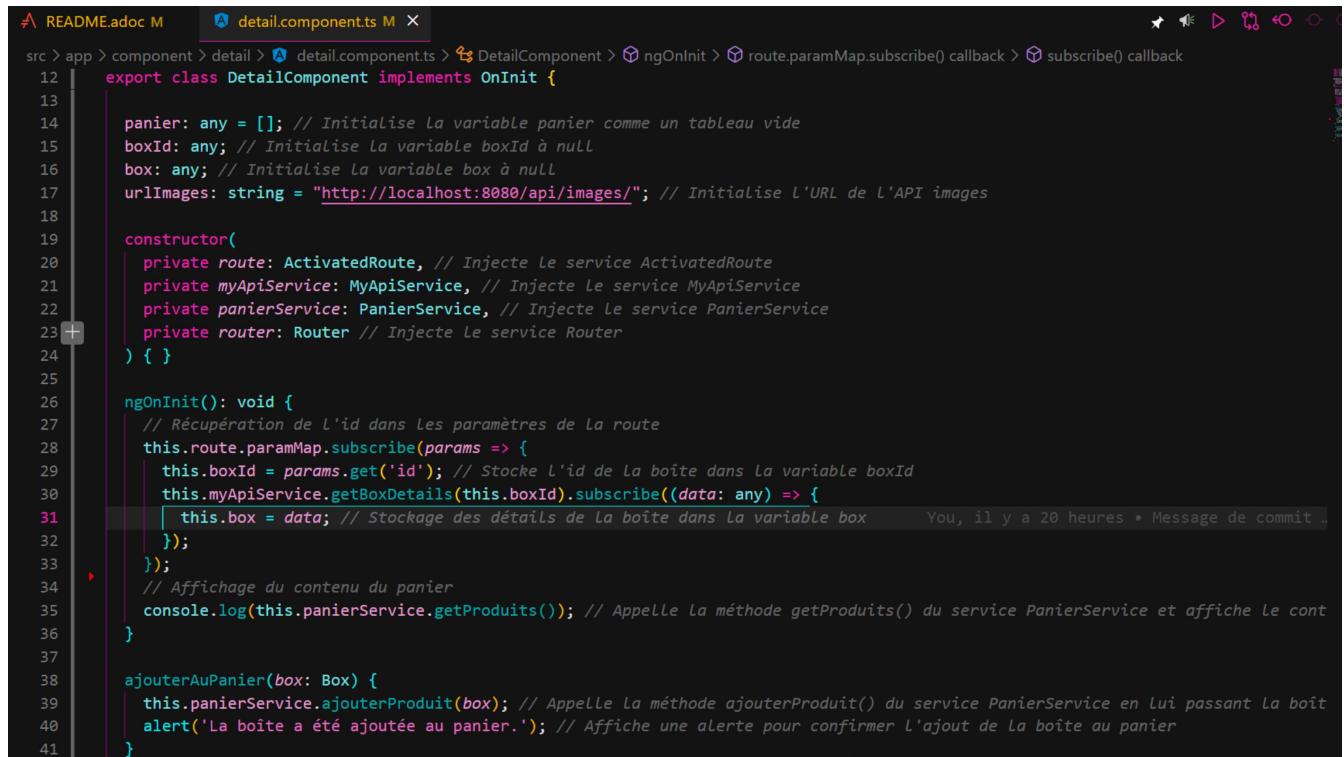
Codes

Produits

The screenshot shows a code editor with the file `produits.component.ts` open. The code is as follows:

```
src > app > component > produits > A produits.component.ts > ...
You, il y a 20 heures | 1 author (You)
1 import { Component, OnInit } from '@angular/core';
2 import { My ApiService } from 'src/app/service/my-api.service';
You, il y a 20 heures | 1 author (You)
3 @Component({
4   selector: 'app-produits',
5   templateUrl: './produits.component.html',
6   styleUrls: ['./produits.component.css']
7 })
8 export class ProduitsComponent implements OnInit {
9
10  produits: any[] = [];
11  urlImages: string = "http://localhost:8080/api/images/";
12
13  constructor(private my ApiService: My ApiService) { }
14
15  ngOnInit() {
16    // Appelle la méthode getProduits() du service My ApiService
17    // et souscrit à son Observable pour récupérer les données
18    this.my ApiService.getProduits().subscribe((data: any) => {
19      |   this.produits = data; // Stocke les données dans la variable "produits"
20    });
21  }
22
23
24 }
```

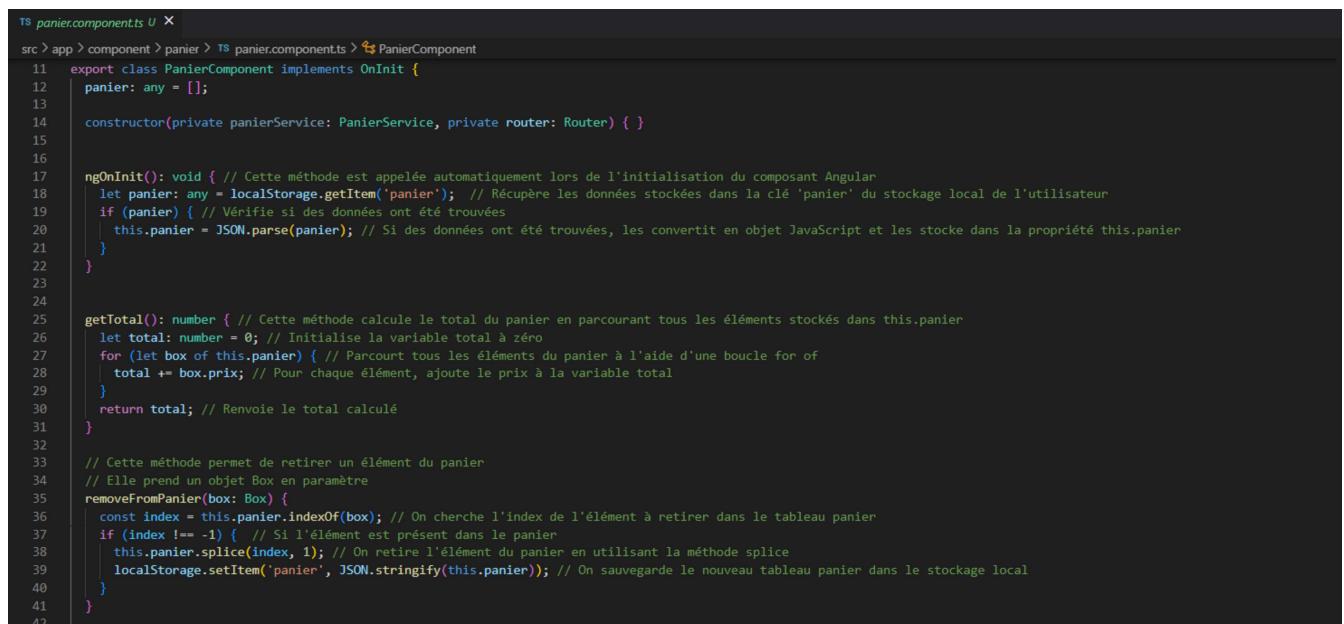
Détails



```
src > app > component > detail > DetailComponent.ts
12 export class DetailComponent implements OnInit {
13
14     panier: any = []; // Initialise la variable panier comme un tableau vide
15     boxId: any; // Initialise la variable boxId à null
16     box: any; // Initialise la variable box à null
17     urlImages: string = "http://localhost:8080/api/images/"; // Initialise l'URL de l'API images
18
19     constructor(
20         private route: ActivatedRoute, // Injecte le service ActivatedRoute
21         private my ApiService: My ApiService, // Injecte le service My ApiService
22         private panierService: PanierService, // Injecte le service PanierService
23         private router: Router // Injecte le service Router
24     ) { }
25
26     ngOnInit(): void {
27         // Récupération de l'id dans les paramètres de la route
28         this.route.paramMap.subscribe(params => {
29             this.boxId = params.get('id'); // Stocke l'id de la boîte dans la variable boxId
30             this.my ApiService.getBoxDetails(this.boxId).subscribe((data: any) => {
31                 this.box = data; // Stockage des détails de la boîte dans la variable box
32             });
33         });
34         // Affichage du contenu du panier
35         console.log(this.panierService.getProduits()); // Appelle la méthode getProduits() du service PanierService et affiche le contenu du panier
36     }
37
38     ajouterAuPanier(box: Box) {
39         this.panierService.ajouterProduit(box); // Appelle la méthode ajouterProduit() du service PanierService en lui passant la boîte
40         alert('La boîte a été ajoutée au panier.'); // Affiche une alerte pour confirmer l'ajout de la boîte au panier
41     }

```

Panier



```
src > app > component > panier > PanierComponent.ts
11 export class PanierComponent implements OnInit {
12     panier: any = [];
13
14     constructor(private panierService: PanierService, private router: Router) { }
15
16
17     ngOnInit(): void { // Cette méthode est appelée automatiquement lors de l'initialisation du composant Angular
18         let panier: any = localStorage.getItem('panier'); // Récupère les données stockées dans la clé 'panier' du stockage local de l'utilisateur
19         if (panier) { // Vérifie si des données ont été trouvées
20             this.panier = JSON.parse(panier); // Si des données ont été trouvées, les convertit en objet JavaScript et les stocke dans la propriété this.panier
21         }
22     }
23
24
25     getTotal(): number { // Cette méthode calcule le total du panier en parcourant tous les éléments stockés dans this.panier
26         let total: number = 0; // Initialise la variable total à zéro
27         for (let box of this.panier) { // Parcourt tous les éléments du panier à l'aide d'une boucle for of
28             total += box.prix; // Pour chaque élément, ajoute le prix à la variable total
29         }
30         return total; // Renvoie le total calculé
31     }
32
33     // Cette méthode permet de retirer un élément du panier
34     // Elle prend un objet Box en paramètre
35     removeFromPanier(box: Box) {
36         const index = this.panier.indexOf(box); // On cherche l'index de l'élément à retirer dans le tableau panier
37         if (index !== -1) { // Si l'élément est présent dans le panier
38             this.panier.splice(index, 1); // On retire l'élément du panier en utilisant la méthode splice
39             localStorage.setItem('panier', JSON.stringify(this.panier)); // On sauvegarde le nouveau tableau panier dans le stockage local
40         }
41     }
42 }
```

```

ts panier.components.ts ×
src > app > component > panier > ts panier.component.ts > PanierComponent
32
33 // Cette méthode permet de retirer un élément du panier
34 // Elle prend un objet Box en paramètre
35 removeFromPanier(box: Box) {
36   const index = this.panier.indexOf(box); // On cherche l'index de l'élément à retirer dans le tableau panier
37   if (index !== -1) { // Si l'élément est présent dans le panier
38     this.panier.splice(index, 1); // On retire l'élément du panier en utilisant la méthode splice
39     localStorage.setItem('panier', JSON.stringify(this.panier)); // On sauvegarde le nouveau tableau panier dans le stockage local
40   }
41 }
42
43 validerCommande() {
44   // Récupérer les informations du panier depuis le localstorage
45   let panier: any = localStorage.getItem('panier');
46   if (panier) {
47     this.panier = JSON.parse(panier);
48
49   // Afficher un récapitulatif de chaque box commandée
50   console.log("Récapitulatif de la commande :");
51   for (let box of this.panier) {
52     console.log("Box : " + box.nom + " - Prix : " + box.prix + " €");
53   }
54
55   // Enregistrer les informations de la commande dans l'historique
56   let historique: any = localStorage.getItem('historique');
57   if (!historique) {
58     historique = [];
59   } else {
60     historique = JSON.parse(historique);
61   }
62   historique.push(this.panier);
63   localStorage.setItem('historique', JSON.stringify(historique));
64
65   // Vider le panier et enregistrer les modifications dans le localstorage
66   this.panier = [];
67   localStorage.setItem('panier', JSON.stringify(this.panier));
68   this.router.navigate(['/historique']);
69 }
70 }
71
72 }

```

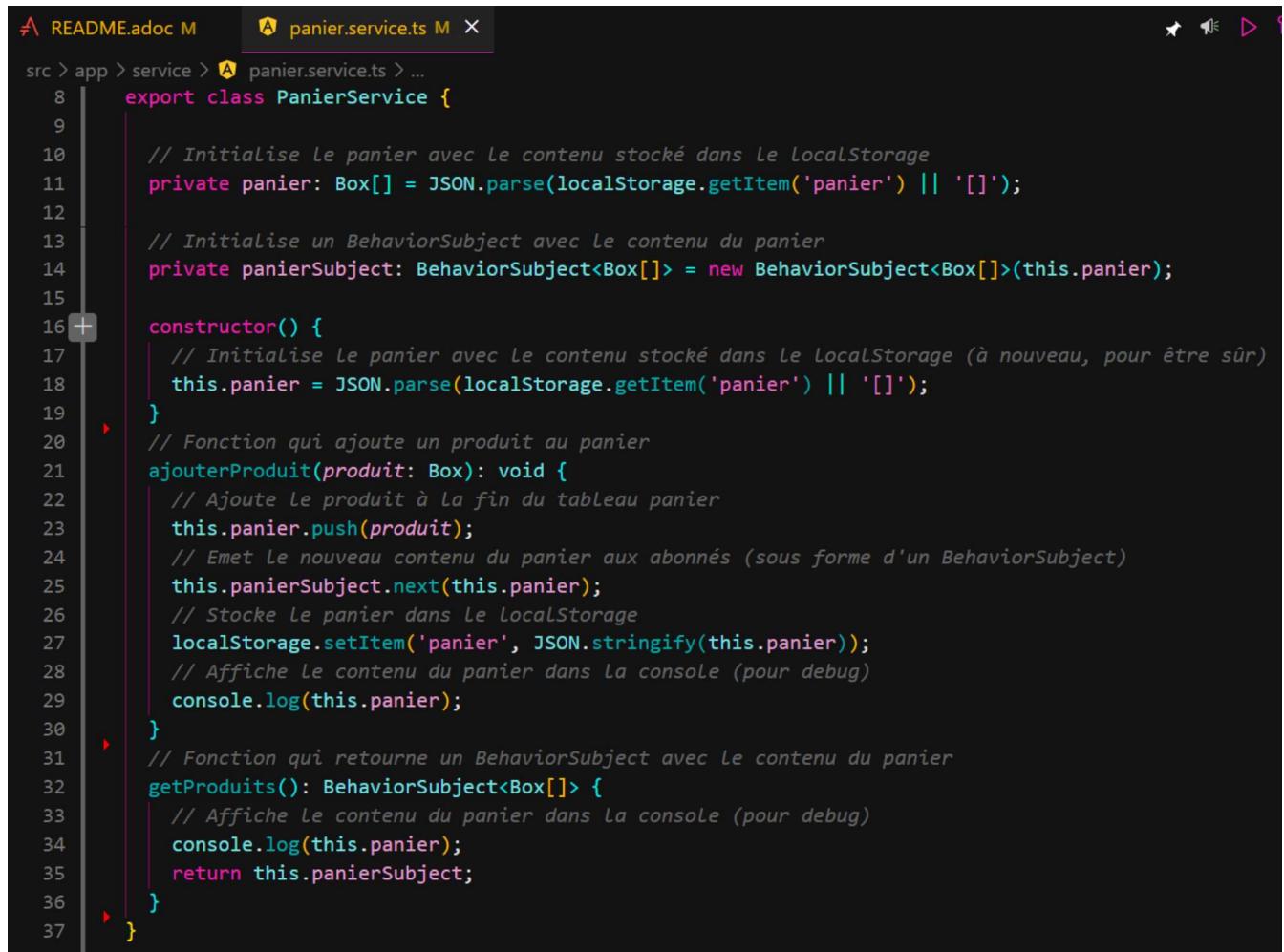
Service API

The screenshot shows a code editor with two tabs: 'README.adoc M' and 'my-api.service.ts'. The 'my-api.service.ts' tab is active, displaying the following TypeScript code:

```
src > app > service > my-api.service.ts > My ApiService
You, il y a 20 heures | 1 author (You)
1 import { Injectable } from '@angular/core';
2 import { HttpClient } from '@angular/common/http';
3 import { environment } from '../../../../../environments/environment';
4
5 You, il y a 20 heures | 1 author (You)
6 @Injectable({
7   providedIn: 'root'
8 })
9 export class My ApiService {
10 +
11   constructor(private http: HttpClient) { }
12
13   // Récupération de tous les produits
14   getProduits() {
15     return this.http.get(`.${environment.apiUrl}`);
16   }
17
18   // Récupération des détails d'une boîte spécifique via son id
19   getBoxDetails(id: number) {
20     return this.http.get(`.${environment.apiUrl}/${id}`);
21   }
22 }
```

A message bar at the bottom right of the editor window says: 'You, la semaine dernière • Message de commit ...'

Service panier



```
src > app > service > A panier.service.ts > ...
8  export class PanierService {
9
10    // Initialise le panier avec le contenu stocké dans le LocalStorage
11    private panier: Box[] = JSON.parse(localStorage.getItem('panier') || '[]');
12
13    // Initialise un BehaviorSubject avec le contenu du panier
14    private panierSubject: BehaviorSubject<Box[]> = new BehaviorSubject<Box[]>(this.panier);
15
16    constructor() {
17        // Initialise le panier avec le contenu stocké dans le LocalStorage (à nouveau, pour être sûr)
18        this.panier = JSON.parse(localStorage.getItem('panier') || '[]');
19    }
20
21    // Fonction qui ajoute un produit au panier
22    ajouterProduit(produit: Box): void {
23        // Ajoute le produit à la fin du tableau panier
24        this.panier.push(produit);
25        // Emet le nouveau contenu du panier aux abonnés (sous forme d'un BehaviorSubject)
26        this.panierSubject.next(this.panier);
27        // Stocke le panier dans le localStorage
28        localStorage.setItem('panier', JSON.stringify(this.panier));
29        // Affiche le contenu du panier dans la console (pour debug)
30        console.log(this.panier);
31    }
32
33    // Fonction qui retourne un BehaviorSubject avec le contenu du panier
34    getProduits(): BehaviorSubject<Box[]> {
35        // Affiche le contenu du panier dans la console (pour debug)
36        console.log(this.panier);
37        return this.panierSubject;
38    }
39}
```

App Routing

src > app > **A** app-routing.module.ts > ...

You, il y a 5 jours | 1 author (You)

```
1 import { NgModule } from '@angular/core';          You, il y a 3 semaines • Mess
2 import { RouterModule, Routes } from '@angular/router';
3 import { DetailComponent } from './component/detail/detail.component';
4 import { ProduitsComponent } from './component/produits/produits.component';
5 import { HomeComponent } from './component/home/home.component';
6 import { PanierComponent } from './component/panier/panier.component';
7
8 const routes: Routes = [
9   { path: '', component: HomeComponent },
10  { path: 'box-details/:id', component: DetailComponent },
11  { path: 'produits', component: ProduitsComponent },
12  { path: 'panier', component: PanierComponent }
13];
14
15 You, il y a 3 semaines | 1 author (You)
16 @NgModule({
17   imports: [RouterModule.forRoot(routes)],
18   exports: [RouterModule]
19 })
20 export class AppRoutingModule { }
```

Box

src > app > models > **TS** box.ts > Box

You, il y a 5 jours | 1 author (You)

```
1 export interface Box {          You, il y a 5 jours • Mes
2   id: number;
3   nom: string;
4   description: string;
5   prix: number;
6   image: string;
7   saveurs: string;
8   aliments: { nom: string; quantite: number }[];
9 }
10 +
```

Produit (box) HTML

A README.adoc M B produits.component.html 4, M X

```
src > app > component > produits > B produits.component.html > ...
Go to component | You, il y a 5 jours | 1 author (You)
1 <table style="table-layout: auto;">
2   <thead>
3     <tr>
4       <!--<th>id</th>-->
5       <th>Nom</th>
6       <th>Prix</th>
7       <th>Pièces</th>
8       <th>Image</th>
9       <th>Détails</th>
10    </tr>
11  </thead>
12  <tbody>
13    <tr *ngFor="let produit of produits">
14      <td>{{ produit.nom }}</td>
15      <td>{{ produit.prix }} €</td>
16      <td>{{ produit.pieces }}</td>
17      <td></td>
18      <td><button><a routerLink="/box-details/{{ produit.id }}">Détails</a></button></td>
19    </tr>
20  </tbody>
21 </table>
22 +
```

Détails (box) HTML

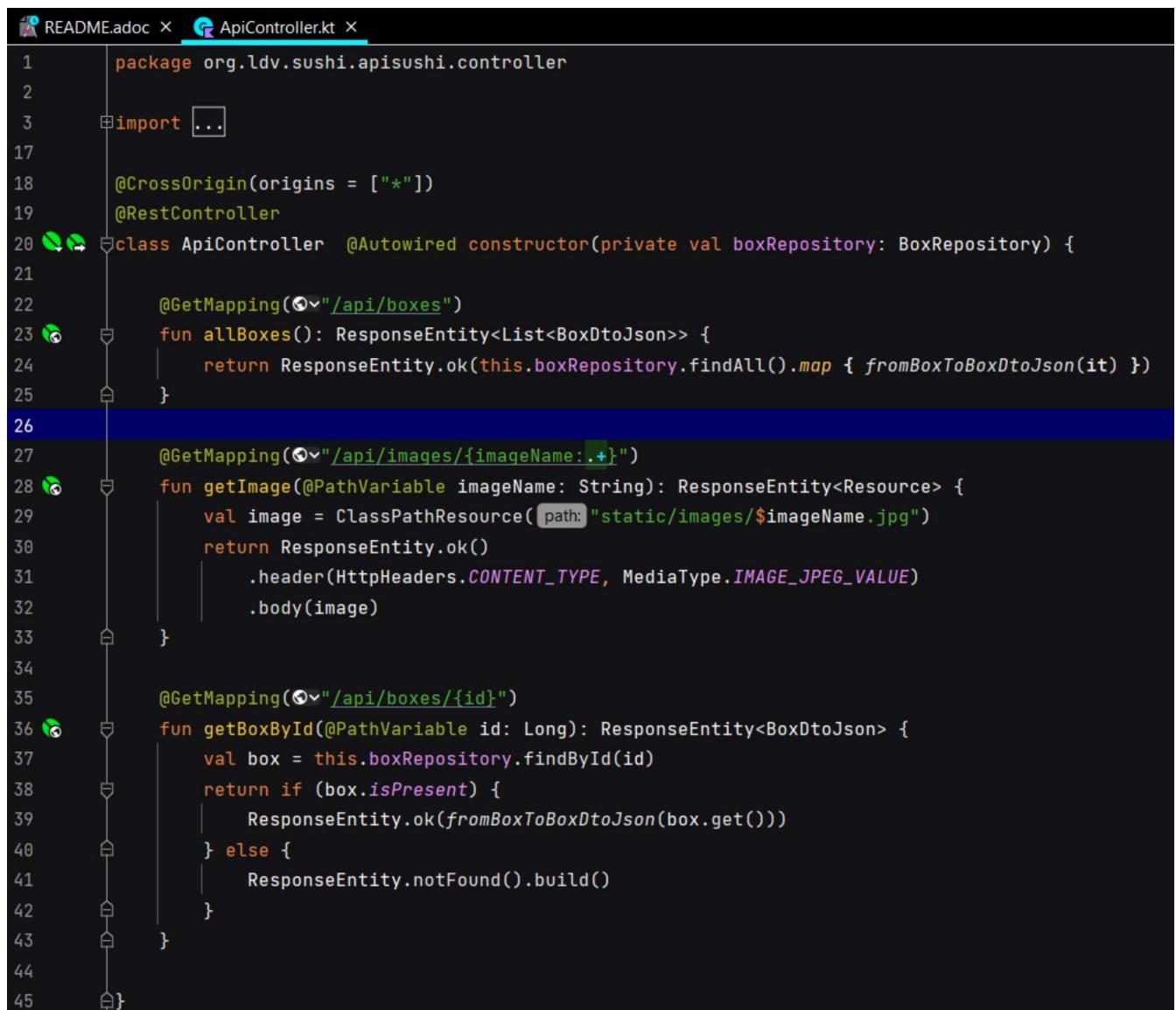
A README.adoc M B detail.component.html 2 X

```
src > app > component > detail > B detail.component.html > Div content
Go to component | You, il y a 4 jours | 1 author (You)
1 <div class="content">          You, il y a 7 jours • Message de commit
2   <div class="box_details">
3     <div class="row">
4       <div class="col-lg-6">
5         <h2>{{ box?.nom }} <span>{{ box?.prix }} €</span></h2>
6
7         <ul>
8           <li><strong>Saveurs:</strong> {{ box?.saveurs }}</li>
9           <li><strong>Aliments:</strong>
10          <ul>
11            <li *ngFor="let aliment of box?.aliments">{{ aliment.nom }} ({{ aliment.quantite }})</li>
12          </ul>
13        </li>
14      </ul>
15
16      <button (click)="ajouterAuPanier(box); goToPanier()">Ajouter au panier</button>
17    </div>
18    <div class="col-lg-6">
19      
20    </div>
21  </div>
22 </div>
23
24 +
```

Environnement API

```
1  export const environment = {
2      production: false,
3      apiUrl: 'http://localhost:8080/api/boxes',
4  };
5
```

Code de l'API



The screenshot shows a code editor with the file `ApiController.kt` open. The code is written in Kotlin and defines a REST controller for managing boxes. The file includes imports, annotations for cross-origin requests and REST controllers, and three methods: `allBoxes()`, `getImage()`, and `getBoxById()`. The `getImage()` method returns a static image resource. The `getBoxById()` method returns a box by its ID, returning a found response if present or a not-found response if not.

```
1  package org.ldv.sushi.apisushi.controller
2
3  import ...
17
18  @CrossOrigin(origins = ["*"])
19  @RestController
20  class ApiController @Autowired constructor(private val boxRepository: BoxRepository) {
21
22      @GetMapping("/api/boxes")
23      fun allBoxes(): ResponseEntity<List<BoxDtoJson>> {
24          return ResponseEntity.ok(this.boxRepository.findAll().map { fromBoxToBoxDtoJson(it) })
25      }
26
27      @GetMapping("/api/images/{imageName:.+}")
28      fun getImage(@PathVariable imageName: String): ResponseEntity<Resource> {
29          val image = ClassPathResource(path: "static/images/$imageName.jpg")
30          return ResponseEntity.ok()
31              .header(HttpHeaders.CONTENT_TYPE, MediaType.IMAGE_JPEG_VALUE)
32              .body(image)
33      }
34
35      @GetMapping("/api/boxes/{id}")
36      fun getBoxById(@PathVariable id: Long): ResponseEntity<BoxDtoJson> {
37          val box = this.boxRepository.findById(id)
38          return if (box.isPresent) {
39              ResponseEntity.ok(fromBoxToBoxDtoJson(box.get()))
40          } else {
41              ResponseEntity.notFound().build()
42          }
43      }
44
45  }
```

Diagramme

Diagramme d'utilisation

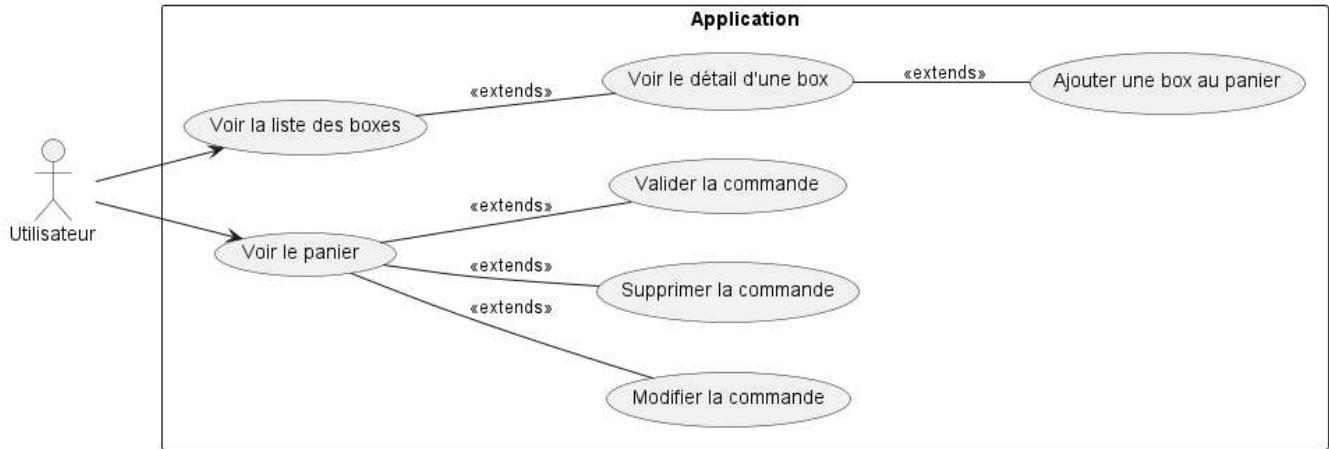
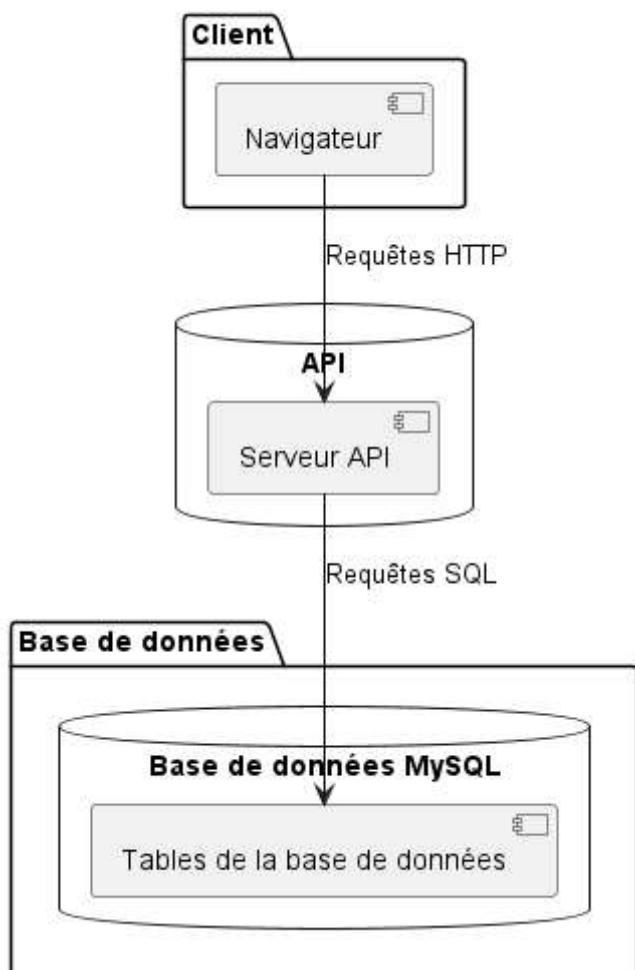


Diagramme des tiers

Architecture de la solution



Structure JSON

```
{
  "items": [
    {
      "id": 1,
      "name": "Box A"
    },
    {
      "id": 2,
      "name": "Box B"
    }
  ]
}
```

```

    "id": 1,
    "date": "02/04/2023",
    "name": "SushiBox1",
    "pieces": 6,
    "quantity": 2
},
{
    "id": 2,
    "date": "05/04/2023",
    "name": "SushiBox2",
    "pieces": 8,
    "quantity": 1
}
]
"nomClient": "Jean Makoumè",
"adresseLivraison": "15 Rue de la Paix, Paris",
"telephone": "01 23 45 67 89",
"montantTotal": 60
}

```

Cybersécurité

Liste d'événements redoutés pour mon projet de site de vente de sushi :

1. Vol de données personnelles des clients tels que noms, adresses, numéros de téléphone et de cartes bancaires.
2. Attaque par déni de service (DDoS) empêchant les clients d'accéder au site et de passer des commandes.
3. Injection de code malveillant (malware) dans le site Web, pouvant permettre à des pirates informatiques d'intercepter les informations de paiement des clients.
4. Attaque de phishing, où les clients peuvent recevoir des e-mails frauduleux leur demandant de fournir des informations de compte ou de paiement.

Contre-mesures (EvilUS) pour ces événements redoutés :

1. Utilisation de pratiques de sécurité appropriées pour protéger les données des clients, telles que le cryptage des données stockées et la mise en œuvre d'une politique de mot de passe fort.
2. Mise en place de mesures de sécurité pour prévenir les attaques DDoS, telles que l'utilisation d'un pare-feu et la surveillance de la bande passante pour détecter les pics de trafic suspects.
3. Mise à jour régulière du logiciel et des systèmes d'exploitation pour prévenir les vulnérabilités connues qui pourraient être exploitées par les pirates informatiques.
4. Sensibilisation des clients aux techniques de phishing et fourniture d'informations claires sur les pratiques de sécurité du site pour éviter les fraudes.

Conclusion

En conclusion, la réalisation de ce projet nous a permis de développer nos compétences en Angular et en développement web. Nous avons appris à concevoir une application d'e-commerce, à créer des composants et à les intégrer dans un système fonctionnel. Nous avons également amélioré nos compétences en matière de design et de développement d'interfaces utilisateur conviviales.

Ce projet nous a également permis de mettre en pratique des concepts clés tels que la gestion des états, l'utilisation des services, l'interaction avec des API externes et la persistance des données avec le LocalStorage.

Nous sommes assez fier du résultat final et nous espérons que cette application pourra être utilisée comme une référence pour les futurs projets de développement web que nous réaliserons.

Technologies utilisées

Angular 13, TypeScript, HTML/CSS, Bootstrap, RxJS.

Contributeurs

TABAR LABONNE Baptiste, LAROUI Chakib, CAFFIAUX Elian, et RODRIGUES Loïc