

# Sushi Shop

## Table of Contents

Présentation .....	1
Fonctionnalités .....	1
Utilisation .....	2
Codes .....	3
Produits .....	3
Détails .....	4
Panier .....	5
Service API .....	6
Service panier .....	7
App Routing .....	8
Box .....	9
Produit (box) HTML .....	9
Détails (box) HTML .....	10
Environnement API .....	11
Code de l'API .....	11
Diagramme .....	12
Diagramme d'utilisation .....	12
Diagramme des tiers .....	12
Structure JSON .....	12
Cybersécurité .....	13
Conclusion .....	14
Technologies utilisées .....	14
Contributeurs .....	14

## Présentation

Il s'agit d'un site pour une prise de commande au niveau d'un point de vente de sushis. Avec l'utilisation du framework Angular pour le développement de l'application et en utilisant une Api afin de collecter les données des différentes boxes de sushi de l'entreprise SushiShop afin d'approcher le plus d'une réalité commerciale.

## Fonctionnalités





Ce projet propose une application qui permet à l'utilisateur de commander des boxes de sushis et de les transmettre à la production (cuisine). Les fonctionnalités implémentées sont les suivantes :

- Une page de produits qui affiche tous les box disponibles dans l'API.


- Pour chaque box, un bouton permet d'accéder aux détails de la box (page de détails).
- Sur la page de détails, un bouton permet d'ajouter une box au panier (page panier).
- Sur la page panier, un bouton permet de passer une commande théorique pour les boxes.
- Accès à l'historique des commandes.

## Utilisation

La page "Produits" présente la liste des boxes de sushis disponibles à l'achat. Si vous souhaitez en savoir plus sur une box en particulier, il suffit de cliquer sur le bouton "Détails" correspondant.

<b>SushiShop</b> <a href="#">Accueil</a> <a href="#">Produit</a> <a href="#">Contact</a> <a href="#">Avis</a> <span>id</span> <span>Panier</span>				
Nom	Prix	Pieces	Image	Détails
Tasty Blend	12.5€	12		<a href="#">Voir le détail</a>
Amateur Mix	15.9€	18		<a href="#">Voir le détail</a>
Saumon Original	12.5€	11		<a href="#">Voir le détail</a>
Salmon Lovers	15.9€	18		<a href="#">Voir le détail</a>

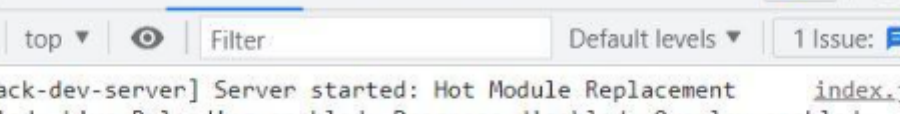
La page "Détails" affiche les informations détaillées de la box de sushis que vous avez sélectionnée. Vous pouvez également ajouter cette box à votre panier en cliquant sur le bouton "Ajouter au panier" au panier".

<b>SushiShop</b> <a href="#">Accueil</a> <a href="#">Produit</a> <a href="#">Contact</a> <a href="#">Avis</a> <span>id</span> <span>Panier</span>				
				
<b>Tasty Blend</b>				
<b>Prix :</b> 12.5 €				
<b>Saveurs :</b> avocat,saumon,cheese				
<b>Aliments :</b> <ul style="list-style-type: none"> <li>• California Saumon Avocat (3)</li> <li>• Sushi Saumon (3)</li> <li>• Spring Avocat Cheese (3)</li> <li>• California pacific (3)</li> <li>• Edamame/Salade de chou (1)</li> </ul>				
<a href="#">Ajouter au panier</a>				
À propos   Contact   Politique de confidentialité Tous droits réservés © 2023 Rodrigues_Loic				

SushiShop [Accueil](#) [Produit](#) [Contact](#) [Avis](#) [id](#) [Panier](#)

À propos    Contact    Politique de confidentialité

Tous droits réservés © 2023 Rodrigues\_Loïc



The screenshot shows the Chrome DevTools Console with the 'Console' tab selected. The output of the `webpack-dev-server` command is visible, indicating that the server has started with Hot Module Replacement disabled, Live Reloading enabled, Progress disabled, and Overlay enabled. It also shows that Angular is running in development mode. Below this, a recap of the command is shown, followed by a list of items in a shopping cart: 'Box : Tasty Blend - Prix : 12.5 €' and 'Box : Salmon Lovers - Prix : 15.9 €'. The file paths for each log entry are shown on the right side of the console.

```
[webpack-dev-server] Server started: Hot Module Replacement disabled, Live Reloading enabled, Progress disabled, Overlay enabled.  
Angular is running in development mode. Call enableProdMode() to enable production mode.  
Récapitulatif de la commande :  
Box : Tasty Blend - Prix : 12.5 €  
Box : Salmon Lovers - Prix : 15.9 €
```

## Produits

```
src > app > component > produits > produits.component.ts > ...
You, il y a 20 heures | 1 author (You)
1 import { Component, OnInit } from '@angular/core';
2 import { MyApiService } 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 myApiService: MyApiService) { }
14
15   ngOnInit() {
16     // Appelle la méthode getProduits() du service MyApiService
17     // et souscrit à son Observable pour récupérer les données
18     this.myApiService.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 > detail.component.ts > DetailComponent > ngOnInit > route.paramMap.subscribe
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
18
19     constructor(
20         private route: ActivatedRoute, // Injecte le service ActivatedRoute
21         private myApiService: MyApiService, // Injecte le service MyApiService
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.myApiService.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
36     }
37
38     ajouterAuPanier(box: Box) {
39         this.panierService.ajouterProduit(box); // Appelle la méthode ajouterProduit() du service
40         alert('La boîte a été ajoutée au panier.');// Affiche une alerte pour confirmer l'ajout
41     }
}
```

## Panier

```
src > app > component > panier > panier.component.ts > PanierComponent
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
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 }
```

```

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 }

```

## Service API

```
src > app > service > my-api.service.ts > MyApiService
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 @Injectable({
6   providedIn: 'root'
7 })
8 export class MyApiService {
9
10   constructor(private http: HttpClient) { }
11
12   // Récupération de tous Les produits
13   getProduits() {
14     return this.http.get(`${environment.apiUrl}`);
15   }
16
17   // Récupération des détails d'une boîte spécifique via son id
18   getBoxDetails(id: number) {
19     return this.http.get(`${environment.apiUrl}/${id}`);
20   }
21
22 }
```

## Service panier

```
src > app > service > 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
18        this.panier = JSON.parse(localStorage.getItem('panier') || '[]');
19    }
20    // Fonction qui ajoute un produit au panier
21    ajouterProduit(produit: Box): void {
22        // Ajoute Le produit à La fin du tableau panier
23        this.panier.push(produit);
24        // Emet Le nouveau contenu du panier aux abonnés (sous forme d'un BehaviorSubject)
25        this.panierSubject.next(this.panier);
26        // Stocke Le panier dans Le LocalStorage
27        localStorage.setItem('panier', JSON.stringify(this.panier));
28        // Affiche Le contenu du panier dans La console (pour debug)
29        console.log(this.panier);
30    }
31    // Fonction qui retourne un BehaviorSubject avec Le contenu du panier
32    getProduits(): BehaviorSubject<Box[]> {
33        // Affiche Le contenu du panier dans La console (pour debug)
34        console.log(this.panier);
35        return this.panierSubject;
36    }
37 }
```

## App Routing



```
src > app > app-routing.module.ts > ...
You, il y a 5 jours | 1 author (You)
1 import { NgModule } from '@angular/core';
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 @NgModule({
16   imports: [RouterModule.forRoot(routes)],
17   exports: [RouterModule]
18 })
19 export class AppRoutingModule { }
20 +
```

## Box

```
src > app > models > box.ts > Box
You, il y a 5 jours | 1 author (You)
1 export interface Box {
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

```
src > app > component > produits > 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>Pieces</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

```
src > app > component > detail > 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.qui
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 </div>
24
```

# Environnement API

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

## Code de l'API

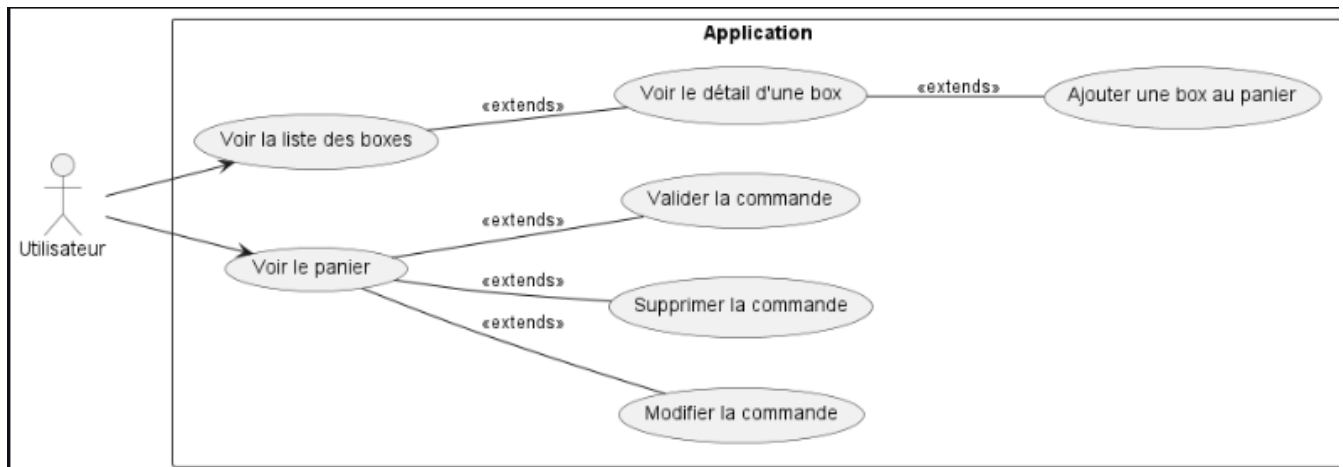
```
1 package org.ldv.sushi.apisushi.controller
2
3 import ...
4
17
18 @CrossOrigin(origins = ["*"])
19 @RestController
20 class ApiController @Autowired constructor(private val boxRepository: BoxRepository) {
21
22   @GetMapping("/api/boxes")
23   fun allBoxes(): ResponseEntity<List<BoxDtoJson>> {
24     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
```

Ces fonctionnalités sont mises à disposition via des endpoints d'une API REST développée en Kotlin, qui permet de fournir des données et des images aux clients.

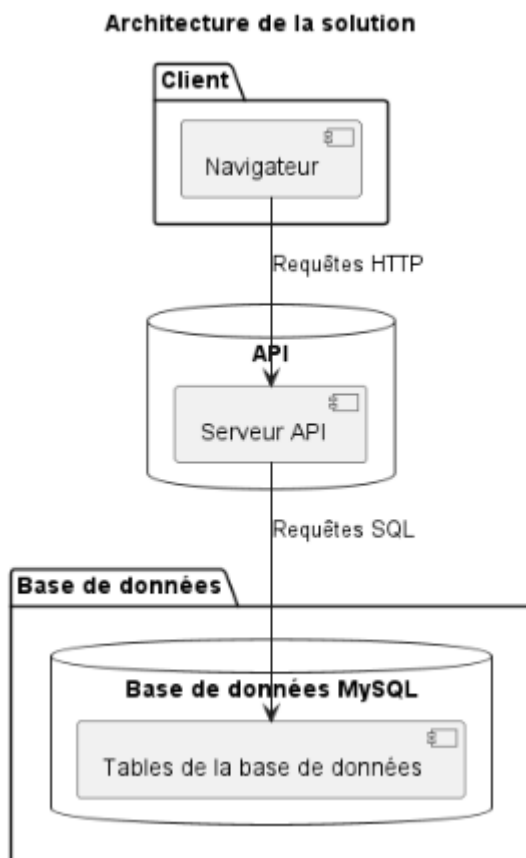
`allBoxes()` permet de renvoyer toutes les boîtes de sushis stockées en base de données, sous forme d'une liste d'objets de type `BoxDtoJson`. `getImage()` permet de récupérer une image stockée dans le dossier `"static/images"` en fournissant son nom, et de renvoyer la ressource correspondante avec un type de contenu `"image/jpeg"`. `getBoxById()` permet de renvoyer une boîte de sushis stockée en base de données, en fournissant son ID, sous forme d'un objet `BoxDtoJson`. Si la boîte est introuvable, une réponse avec le statut `"notFound()"` est renvoyée.

# Diagramme

## Diagramme d'utilisation



## Diagramme des tiers



## Structure JSON

```
{  
  "items": [  
    {
```

```
    "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": "Paul Lambert",  
"adresseLivraison": "23 Rue de la mairie, Paris",  
"telephone": "01 23 45 67 89",  
}
```

## Cybersécurité

Voici une liste des scénarios de sécurité redoutés pour mon projet de site de vente de sushi :

- Le vol de données personnelles des clients, telles que les noms, les adresses, les numéros de téléphone et de cartes bancaires.
- Une attaque par déni de service (DDoS) qui empêche les clients d'accéder au site et de passer des commandes.
- Une injection de code malveillant (malware) dans le site Web, qui permettrait à des pirates informatiques d'intercepter les informations de paiement des clients.
- Une attaque de phishing, où les clients peuvent recevoir des e-mails frauduleux leur demandant de fournir des informations de compte ou de paiement.

Voici quelques contre-mesures (EvilUS) que nous pouvons prendre pour prévenir ces événements redoutés :

- Nous devons utiliser des pratiques de sécurité appropriées pour protéger les données des clients, comme le cryptage des données stockées et l'application d'une politique de mot de passe fort.
- Nous devons mettre en place des mesures de sécurité pour prévenir les attaques DDoS, comme l'utilisation d'un pare-feu et la surveillance de la bande passante pour détecter les pics de trafic suspects.
- Nous devons mettre à jour régulièrement le logiciel et les systèmes d'exploitation pour prévenir les vulnérabilités connues qui pourraient être exploitées par les pirates informatiques.
- Nous devons sensibiliser les clients aux techniques de phishing et leur fournir des informations claires sur les pratiques de sécurité du site pour éviter les fraudes.

# Conclusion

En résumé, la réalisation de ce projet a été bénéfique pour notre développement de compétences en Angular et en développement web. Nous avons acquis une expérience significative dans la conception d'une application d'e-commerce, l'intégration de composants et la création d'un système fonctionnel. Nous avons également amélioré nos capacités en matière de design et de développement d'interfaces utilisateur intuitives.

Ce projet a été une occasion d'appliquer des concepts essentiels tels que la gestion des états, l'utilisation de services, l'interaction avec des API externes et la persistance des données avec le LocalStorage.

Nous sommes fiers du résultat final et espérons que cette application servira de référence pour les futurs projets de développement web que nous entreprendrons.

# Technologies utilisées

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

# Contributeurs

CAFFIAUX Elia et RODRIGUES Loïc