



Écoles nationales des sciences appliquées

Rapport de projet de fin d'études

Réalisé par :

Alaoui lamrani Khanssae

El ouahabi Firdaouss

2022-2023

Table des matières

Remerciement	2
Introduction	3
Cahier de Charge	4
1. Application de gestion d'achat en ligne	4
a. Partie Utilisateur :	4
b. Partie Administrateur :	5
Conception et analyse du projet	6
1. Analyse:	6
2. Cas d'Utilisation : Passer une Commande	7
3. Diagramme de Séquence : Passer une Commande	8
Conception architectural	9
1. Architecture logique:	9
2. Architecture physique:	10
La conception de la base de données	11
1. Entités Principales :	11
2. Relations :	12
Réalisation	13
1. Environnement logiciel	13
2. Les outils de développement utilisés	13
3. Structure du projet	14
Interface du site web	19
1. Le logo:	19
2. Quelques interfaces de notre application :	19
a. La première page	19
b. Interface d'authentification:	20
c. Page d'inscription :	24
Conclusion	30

Remerciement

C'est avec une grande fierté et un profond sentiment d'accomplissement que je présente ce rapport de projet de fin d'études en développement informatique. Cette étape marque la fin d'une période intense et passionnante de ma vie académique, et je tiens à exprimer ma sincère gratitude envers tous ceux qui ont contribué à cette réalisation.

Tout d'abord, je voudrais adresser mes remerciements les plus chaleureux à tous les enseignants de la formation DCA Génie Informatique à l'ENSA de Tanger, pour leur dévouement et leur assistance tout au long de cette année.

Et enfin je tiens également à remercier mes camarades de classe méritent également une mention spéciale. Le partage de connaissances, les discussions techniques et l'esprit d'équipe que nous avons cultivés ont été des éléments clés de mon apprentissage.

Merci pour cette expérience collaborative enrichissante.

Introduction

Le développement informatique est un domaine en constante évolution, jouant un rôle crucial dans la résolution de problèmes complexes et la création de solutions innovantes pour répondre aux besoins croissants de la société numérique. Dans ce contexte, le présent rapport de projet de fin d'études (PFE) se penche sur une étude approfondie et une mise en œuvre pratique d'un projet de développement informatique qui a pour objectif de résoudre un problème spécifique et de fournir une contribution significative au domaine.

Ce rapport présente une analyse détaillée des besoins, la conception et le développement du système, ainsi que les résultats obtenus et les leçons apprises tout au long du processus. Il met en évidence les défis rencontrés et les solutions apportées, ainsi que les technologies et les méthodologies utilisées pour parvenir à la réalisation du projet.

L'objectif principal de ce rapport est de documenter de manière exhaustive toutes les étapes du projet, de partager les connaissances acquises et de fournir une perspective critique sur les résultats obtenus. Il s'agit également de mettre en lumière l'importance de l'innovation et de l'expertise en développement informatique dans un monde de plus en plus numérique.

Dans les sections suivantes, nous aborderons en détail les aspects clés de ce projet, de la planification initiale à la mise en œuvre, en passant par l'analyse des résultats et les recommandations pour des travaux futurs. En fin de compte, ce rapport vise à démontrer l'engagement et la compétence des auteurs dans le domaine du développement informatique tout en contribuant à l'enrichissement de la base de connaissances de ce secteur en constante évolution.

Cahier de Charge

1. Application de gestion d'achat en ligne

Un cahier des charges pour une application de gestion d'achat en ligne, comprenant à la fois la partie utilisateur et administrateur, devrait inclure les éléments suivants :

a. Partie Utilisateur :

1. Inscription et Connexion :

- Formulaire d'inscription avec vérification d'email.
- Connexion sécurisée avec mot de passe.

2. Profil Utilisateur :

- Possibilité de mettre à jour les informations personnelles.
- Historique des commandes.

3. Recherche de Produits :

- Barre de recherche pour trouver des produits.
- Filtres pour affiner les résultats (catégorie, prix, etc.).

4. Catalogue de Produits :

- Affichage la liste des produits descriptions et prix.

5. Panier d'Achat :

- Ajout de produits au panier.
- Modification du panier (ajouter/supprimer des produits).
- Calcul du montant total.

6. Passer Commande :

- Processus de commande en plusieurs étapes.

b. Partie Administrateur :

1) Tableau de Bord :

- Vue d'ensemble des ventes, des commandes en attente, des produits populaires, etc.

2) Gestion des Produits :

- Ajout, modification et suppression de produits.
- Gestion des stocks et des catégories.

3) Gestion des Commandes :

- Traitement des commandes (marquer comme expédiées, annuler, etc.).
- Suivi des commandes en cours.

4) Gestion des Utilisateurs :

- Gestion des comptes utilisateur (blocage, suppression).
- Vue d'ensemble des informations utilisateur.

5) Statistiques et Rapports :

- Génération de rapports sur les ventes, les produits populaires, etc.

6) Support Client :

- Possibilité de répondre aux demandes d'assistance.

7) Système de Paiement :

- Intégration de passerelles de paiement sécurisées.

8) Sécurité et Protection des Données :

- Sécurisation des données utilisateur et des transactions.

9) Gestion des Promotions :

- Ajout de codes de réduction et d'offres spéciales.

10) Système de Gestion des Retours :

- Gestion des demandes de retour et de remboursement.

Conception et analyse du projet

1. Analyse:

- I. Analyse des Besoins :
 - Identifier les besoins des utilisateurs et des administrateurs.
 - Étudier la concurrence pour définir des fonctionnalités uniques.
 - Définir les objectifs du projet et les indicateurs de performance clés.
- II. Spécifications Fonctionnelles :
 - Créer une liste détaillée de toutes les fonctionnalités de l'application.
 - Organiser ces fonctionnalités en modules ou en sections logiques.
 - Définir les flux de travail pour les utilisateurs et les administrateurs.
- III. Conception de la Base de Données :
 - Modéliser la structure de la base de données pour stocker les informations sur les produits, les commandes, les utilisateurs, etc.
 - Choisir la technologie de base de données appropriée (SQL, NoSQL, etc.).
- IV. Architecture Logicielle :
 - Concevoir l'architecture logicielle, y compris la pile technologique (langages de programmation, frameworks, etc.).
 - Planifier la manière dont les différents modules interagiront entre eux.
- V. Conception de l'Interface Utilisateur (UI) :
 - Créer des maquettes ou des prototypes de l'interface utilisateur.
 - Assurer une expérience utilisateur conviviale (UI/UX).
- VI. Sécurité :
 - Élaborer une stratégie de sécurité pour protéger les données utilisateur, les transactions et l'application contre les menaces.

- Mettre en place l'authentification et l'autorisation appropriées.

VII. Plan de Test :

- Définir un plan de test pour chaque fonctionnalité de l'application.
- Effectuer des tests d'unité, d'intégration et de convivialité.
- Corriger les bogues et les problèmes identifiés.

2. Cas d'Utilisation : Passer une Commande

Acteurs :

- Utilisateur (Client)
- Système

Description : L'acteur Utilisateur souhaite passer une commande pour acheter des produits à travers l'application de gestion d'achat en ligne.

Flux Principal :

1. L'Utilisateur se connecte à son compte.
2. L'Utilisateur peut afficher la liste des produits.
3. L'Utilisateur peut consulter sa commande dans son historique de commandes(il peut ajouter ou supprimer des produits là-dessus) ou supprimer tout la commande .
4. L'Utilisateur accède à sa commande et vérifie les produits sélectionnés.
5. Le Système génère un numéro de commande et envoie une confirmation de commande à l'Utilisateur par e-mail.

3. Diagramme de Séquence : Passer une Commande

Acteurs :

- Utilisateur (Client)
- Système

Description : Ce diagramme de séquence représente les interactions entre l'Utilisateur et le Système lorsqu'il passe une commande à travers l'application.

1. L'Utilisateur se connecte à son compte.
2. Le Système vérifie les informations d'identification de l'Utilisateur.
3. Une fois connecté, l'Utilisateur recherche la liste des produits.
4. Le Système renvoie les résultats de la page Home de l'Utilisateur.
5. L'Utilisateur ajoute un produit à sa commande.
6. Le Système met à jour la commande avec le produit ajouté.
7. L'Utilisateur vérifie la commande.
8. Le Système affiche le contenu de la commande.
9. L'Utilisateur confirme la commande.
10. Le Système enregistre la commande.
11. Le Système génère un numéro de commande.

Conception architectural

La conception architecturale d'une application de gestion d'achat en ligne est un aspect crucial du processus de développement. Une architecture bien conçue garantit que l'application est évolutive, maintenable et performante.

Il existe deux types d'architectures logicielles : architecture physique et architecture logique.

1. Architecture logique:

L'architecture logique se concentre sur la structure interne de l'application sans se soucier des détails matériels ou de l'infrastructure sur laquelle elle fonctionne. Elle définit comment les composants de l'application interagissent entre eux pour répondre aux besoins fonctionnels.

Dans le contexte de votre application de gestion d'achat en ligne, l'architecture logique pourrait inclure :

1. **Modèle-Vue-Contrôleur (MVC) :**
 - La séparation conceptuelle des composants Modèle, Vue et Contrôleur, où le Modèle gère la logique métier, la Vue gère l'interface utilisateur et le Contrôleur gère la logique de contrôle.
2. **Couche de Service :**
 - La définition des services qui encapsulent la logique métier complexe, tels que la gestion des paiements, des commandes, etc.
3. **Base de Données :**
 - La structure logique de la base de données, y compris les tables, les relations, les schémas, les procédures stockées, etc.
4. **Sécurité :**
 - La mise en place de mesures de sécurité, telles que l'authentification, l'autorisation et la gestion des sessions, sans se soucier des détails de mise en œuvre.

2. Architecture physique:

L'architecture physique, en revanche, se concentre sur la manière dont l'application est déployée sur l'infrastructure matérielle réelle. Elle concerne les serveurs, les réseaux, les systèmes d'exploitation, les ressources matérielles et les configurations nécessaires pour faire fonctionner l'application.

Pour l'application de gestion d'achat en ligne, l'architecture physique pourrait inclure :

1. Serveurs :
 - La spécification des serveurs sur lesquels l'application sera déployée, par exemple, serveurs web, serveurs de base de données, serveurs d'application, etc.
2. Réseaux :
 - La configuration du réseau pour assurer la connectivité entre les composants de l'application, ainsi que la sécurité du réseau.
3. Systèmes d'Exploitation :
 - Le choix du système d'exploitation pour les serveurs et les configurations associées, telles que la gestion des utilisateurs, les mises à jour, etc.
4. Redondance et Évolutivité :
 - La mise en œuvre de la redondance (tolérance aux pannes) et de l'évolutivité (capacité à gérer une charge croissante) au niveau de l'infrastructure.
5. Sécurité Matérielle :
 - La mise en place de mesures de sécurité au niveau matériel, telles que les pare-feu, les systèmes de détection des intrusions, etc.
6. Plan de Sauvegarde et de Récupération :
 - La stratégie de sauvegarde des données et de récupération en cas de sinistre.

La conception de la base de données

1. Entités Principales :

1. Client :

- ♦ ID (clé primaire)
- ♦ Adresse e-mail
- ♦ Nom d'utilisateur
- ♦ Prénom d'utilisateur
- ♦ Numéro de téléphone
- ♦ Adresse
- ♦ Mot de passe

2. Commande :

- ♦ ID (clé primaire)
- ♦ Date de création de commande
- ♦ Facture
- ♦ Client_id

3. Produit :

- ♦ ID (clé primaire)
- ♦ Nom du produit
- ♦ Prix
- ♦ Stock disponible (quantité)
- ♦ Fournisseur_id

4. Fournisseur :

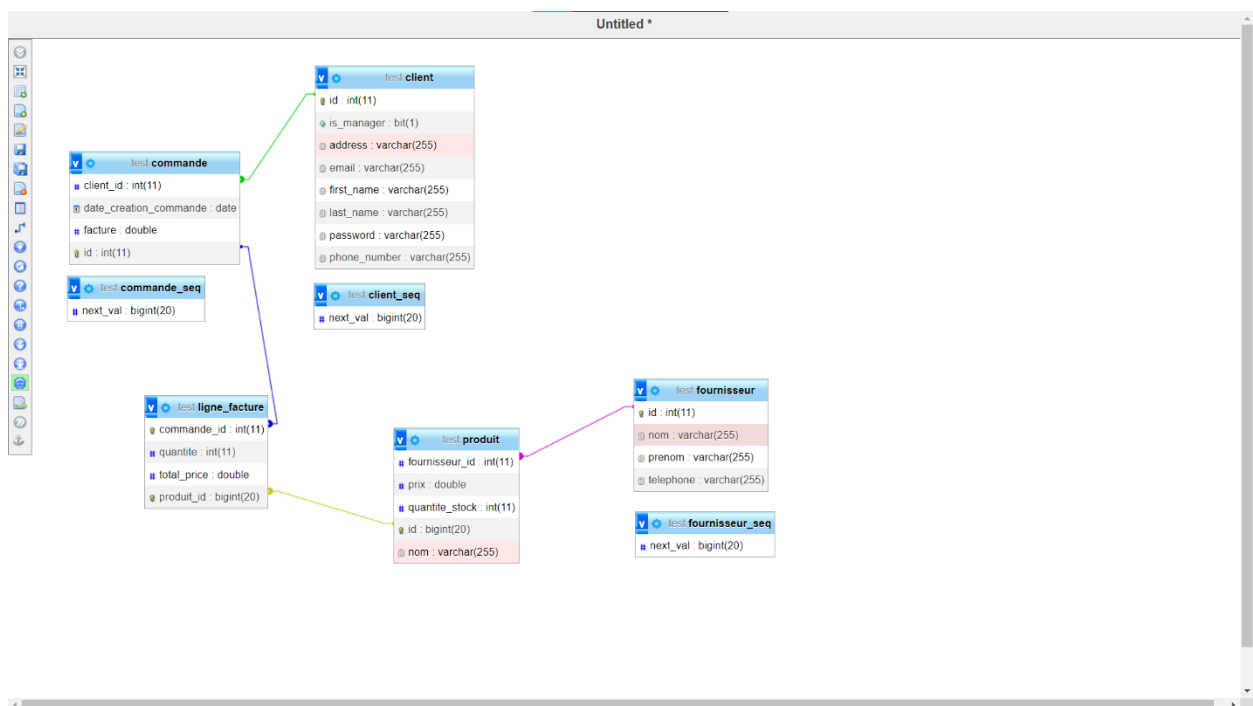
- ♦ ID (clé primaire)
- ♦ Nom
- ♦ Prénom
- ♦ Téléphone

5. Ligne_facture :

- ♦ ID de la commande (référence à une commande)
- ♦ ID du produit (référence à un produit)
- ♦ Quantité commandée
- ♦ Total prie

2. Relations :

- Un client peut passer plusieurs commandes, mais chaque commande appartient à un seul utilisateur (relation un-à-plusieurs entre client et Commande).
- Une commande peut contenir plusieurs produits, et chaque produit peut apparaître dans plusieurs commandes (relation plusieurs-à-plusieurs entre Commande et Détail de la Commande).
- Un fournisseur peut fournir plusieurs produits (relation un-à-plusieurs entre fournisseur et produits).



Réalisation

1. Environnement logiciel

- Analyse SI v0.80 : Logiciel de modélisation de bases de données.
- Lucidchart : est une plateforme de collaboration en ligne, basée sur le cloud, permettant la création de diagrammes et la visualisation de données, et autres schémas conceptuels. On l'a utilisé pour la création des diagramme UML.
- Wrike : est un logiciel de gestion de projet exclusivement basé sur le cloud. On l'a utilisé pour gérer les taches et les notes concernant notre projet.
- Google Meet : est un service de visioconférence haute qualité. On l'a utilisé pour faire des réunions pour discuter notre progrès.
- Git : Git est un système de contrôle de version open source. C'est un outil qui permet de traquer tous les fichiers d'un projet. Il nous a faciliter la collaboration et le travail sur le même projet.

2. Les outils de développement utilisés

Choix du langage de programmation :

Le langage de programmation utilisé va beaucoup influencer sur le projet et la manière dont celui-ci sera développé, en fonction des avantages et des inconvénients du langage. Il convient de le choisir en considérant les besoins spécifiques du projet.

❖ Spring Boot :

Le **Spring Framework** est très largement utilisé dans la communauté Java. Il permet d'accélérer le développement d'applications d'entreprise (notamment le développement d'applications Web et d'API Web). Mais on trouve des applications basées sur le Spring Framework dans bien d'autres domaines.

Pour comprendre l'origine et l'apport du Spring Framework, il faut savoir que son principal auteur, Rod Johnson, voulait proposer une alternative au modèle d'architecture logiciel proposé par la plate-forme J2EE au début des années 2000.

❖ Vuejs

Vue.js est un framework JavaScript, utilisé pour la partie front-end, qui a l'avantage d'être très accessible. Vue.js se vante d'être un framework avec une courbe d'apprentissage progressive. Il propose une documentation très évoluée et riche. On retrouve, au même endroit, toutes les ressources dont on a besoin. Il a également une communauté en expansion. Qui est très ouverte, et la plus actives, ce qui permet d'obtenir de nombreux guides ou explications sur l'utilisation de ce framework.

❖ MySQL :

MySQL est un serveur de bases de données relationnelles Open Source. Un serveur de bases de données stocke les données dans des tables séparées plutôt que de tout rassembler dans une seule table. Cela améliore la rapidité et la souplesse de l'ensemble. Les tables sont reliées par des relations définies, qui rendent possible la combinaison de données entre plusieurs tables durant une requête. Le SQL dans "MySQL" signifie "Structured Query Language" : le langage standard pour les traitements de bases de données.

3. Structure du projet

1. Dossiers principaux :

- `src/main/java`: Contient le code source Java de votre application.
- `src/main/resources`: Contient les fichiers de configuration, les fichiers de propriétés, les modèles de vues, etc.

2. Dossiers Java :

- `com.votrepackage.application`: L'arborescence des packages Java de votre application.
- `com.votrepackage.application.controller`: Les classes de contrôleur.
- `com.votrepackage.application.service`: Les classes de service.
- `com.votrepackage.application.repository`: Les classes de repository (si vous utilisez une base de données).

3. Classe principale :

- Une classe principale (par exemple, `VotreApplication.java`) qui est annotée avec `@SpringBootApplication` et contient la méthode `main` pour lancer l'application.

4. Dossiers de ressources :

- `static`: Les ressources statiques telles que les fichiers CSS, JavaScript, etc.
- `templates`: Les modèles de vues Thymeleaf ou FreeMarker (si vous utilisez un moteur de templates).
- `application.properties` ou `application.yml`: Fichier de configuration principal pour définir des propriétés spécifiques de l'application.

5. Tests :

- `src/test/java`: Contient les classes de tests unitaires.
- `src/test/resources`: Contient les fichiers de configuration de tests.

6. Dossiers de documentation (optionnels) :

- `docs`: Contient la documentation supplémentaire, telle que des spécifications API ou des guides pour les développeurs.

7. Dossiers de configuration (optionnels) :

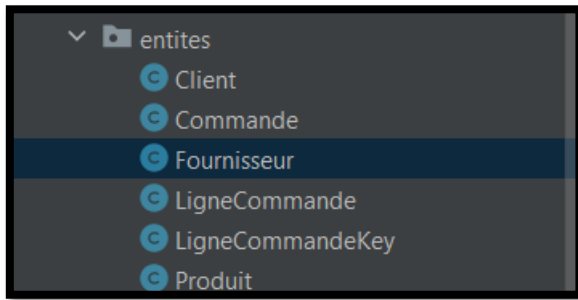
- `config`: Vous pouvez placer des fichiers de configuration supplémentaires ici.

8. Dossiers de gestion de ressources (si applicable) :

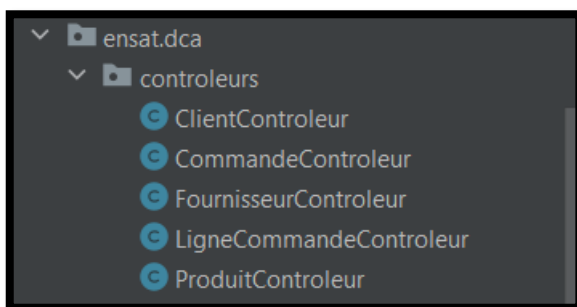
- Par exemple, si vous travaillez avec des fichiers téléchargés, vous pourriez avoir un dossier `uploads` pour stocker les fichiers téléchargés.

9. Autres dossiers et fichiers :

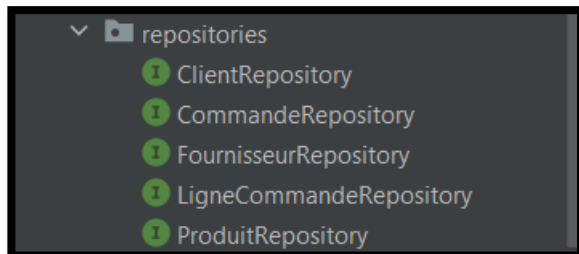
- D'autres dossiers et fichiers peuvent être ajoutés en fonction des besoins spécifiques de votre application.



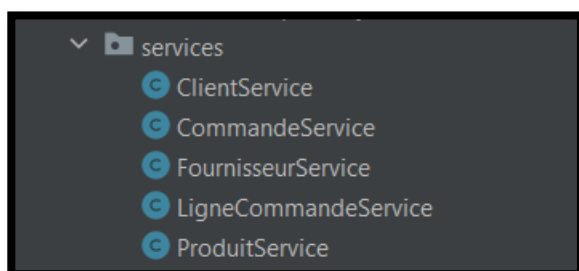
- **Une entité** est généralement une classe Java qui représente un objet métier ou une table de base de données dans le système. Les entités sont utilisées pour modéliser les données avec lesquelles votre application API interagit.



- **Un contrôleur** (ou controller en anglais) est une classe Java qui gère les requêtes HTTP entrantes et les renvoie sous forme de réponses HTTP. Les contrôleurs sont responsables du routage des demandes vers les méthodes appropriées pour effectuer des actions spécifiques, telles que la création, la lecture, la mise à jour ou la suppression de données.

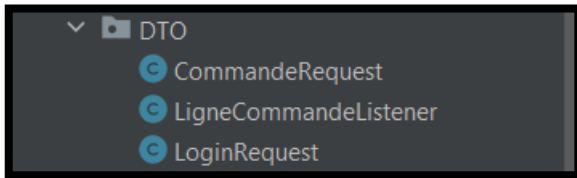


- **Les repositories** sont des composants essentiels qui interagissent avec la base de données pour effectuer des opérations CRUD (Create, Read, Update, Delete) sur les entités de l'application. Spring Data JPA est couramment utilisé pour simplifier la création de ces repositories.



- **Les services** jouent un rôle essentiel pour encapsuler la logique métier de votre application. Ils agissent comme une couche intermédiaire entre les contrôleurs (qui gèrent les requêtes HTTP) et les repositories (qui gèrent l'accès aux données dans la base de données).

- **Les DTO** (Data Transfer Objects) sont des classes spécialement conçues pour transférer des données entre le client et le serveur. Les DTO sont utilisés pour encapsuler des données provenant de l'API et les envoyer au client, ainsi que pour recevoir des données du client et les transmettre à l'API. Ils permettent de définir précisément les données qui sont transférées, ce qui peut aider à réduire la surcharge de données et à améliorer la sécurité.



- **WebMvcConfigurer** sert à personnaliser la configuration de Spring MVC dans une application Spring Boot. Elle vous permet d'ajuster le comportement de votre application pour répondre à des besoins particuliers, tels que la gestion des ressources, les intercepteurs, la conversion des types, etc. Cela vous donne un haut degré de flexibilité pour adapter Spring MVC à vos exigences.

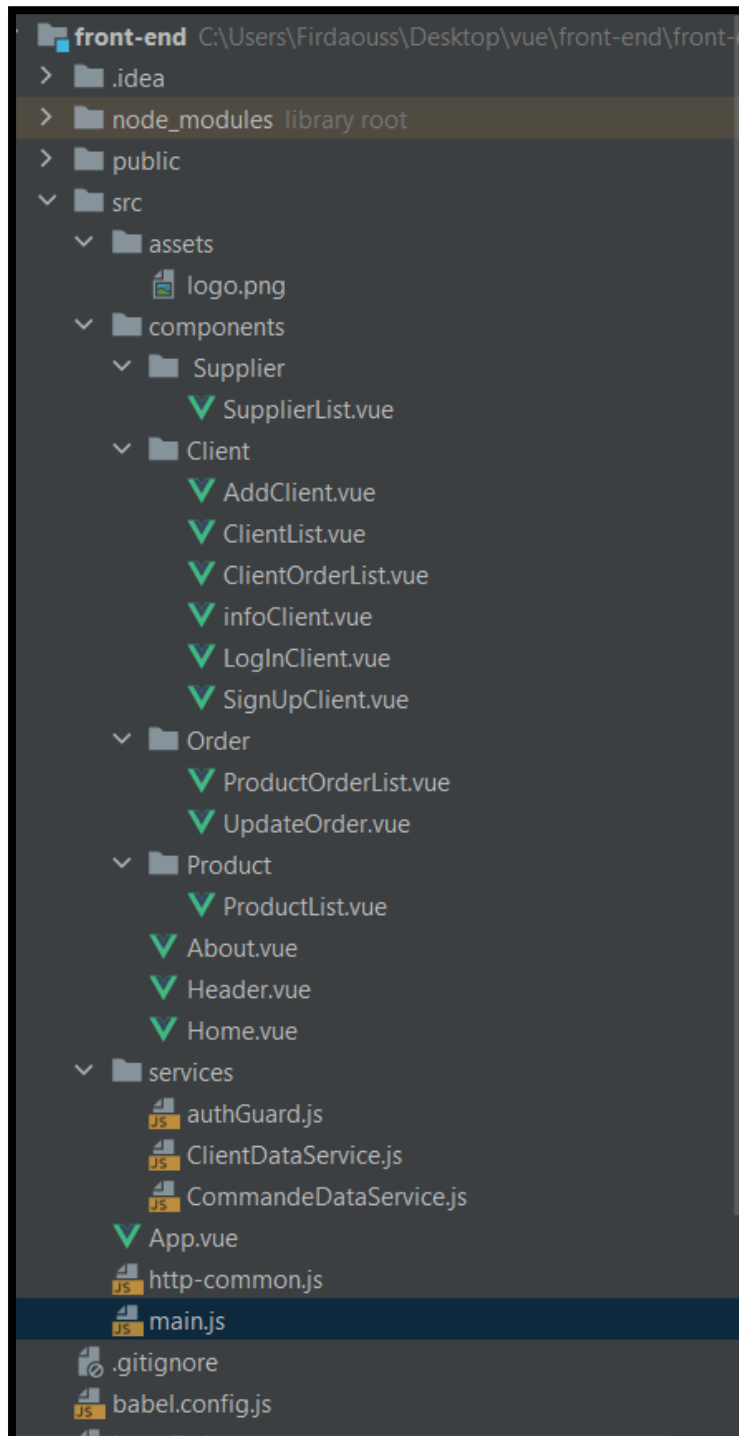
```

1  package ensat.dca;
2  import org.springframework.context.annotation.Configuration;
3  import org.springframework.web.servlet.config.annotation.CorsRegistry;
4  import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
5
6  no usages
7  @Configuration
8  public class CorsConfig implements WebMvcConfigurer {
9      no usages
10     @Override
11     public void addCorsMappings(CorsRegistry registry) {
12         registry.addMapping(pathPattern: "**")
13             .allowedOrigins("http://localhost:8081")
14             .allowedMethods("GET", "POST", "PUT", "DELETE");
15     }
16 }

```

- Vues :

Les vues se sont divisées en parties, la partie admin avec tous les composants VueJs de l'interface de l'administrateur. La partie user avec les composants de la partie utilisateur, les pages Login et Register et la partie de la modification du profil qui sont en commun entre l'utilisateur et l'administrateur.



Interface du site web

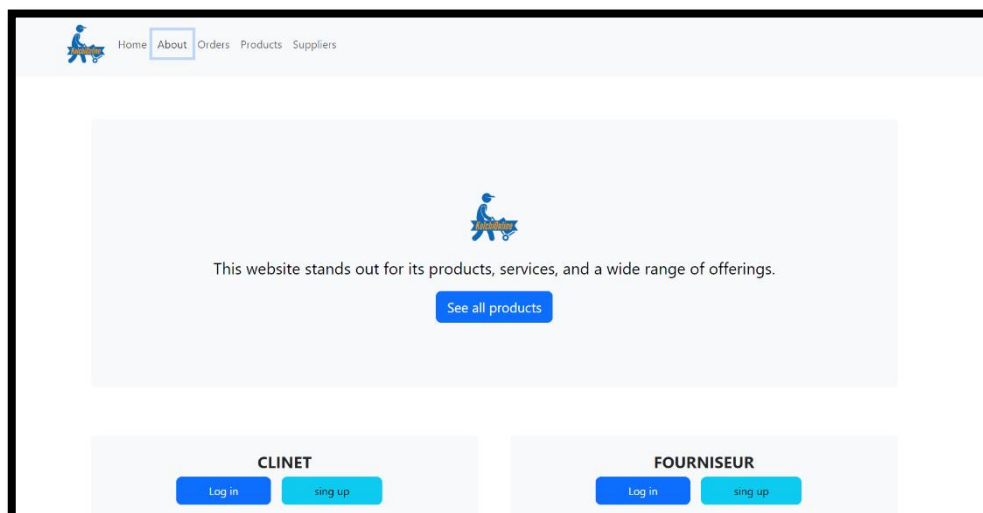
Dans cette partie nous présentons quelques interfaces de notre application.

1. Le logo:



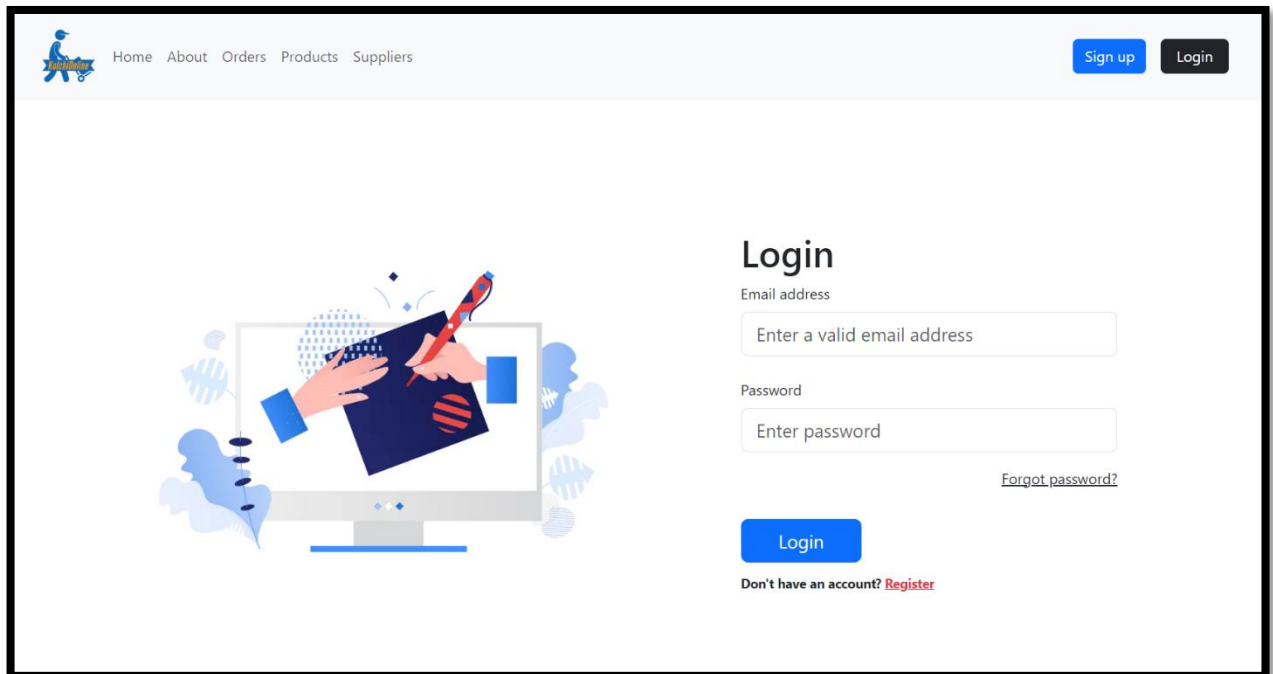
2. Quelques interfaces de notre application :

La première page



Interface d'authentification:

Après authentification l'utilisateur peut accéder aux différentes fonctionnalités de l'application selon son rôle (admin, utilisateur).



Home About Orders Products Suppliers

Sign up Login

Login

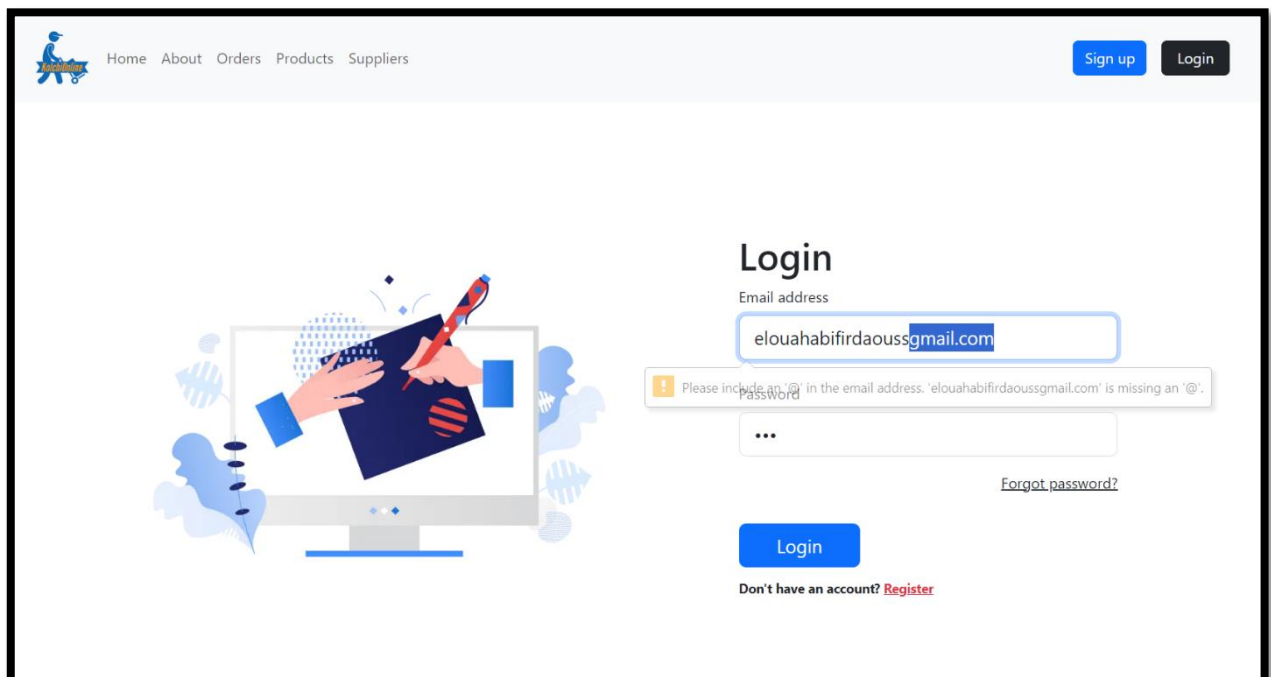
Email address

Password

[Forgot password?](#)

Login

Don't have an account? [Register](#)



Home About Orders Products Suppliers

Sign up Login

Login

Email address

Please include an '@' in the email address: 'elouahabifirdaoussgmail.com' is missing an '@'.

...

[Forgot password?](#)

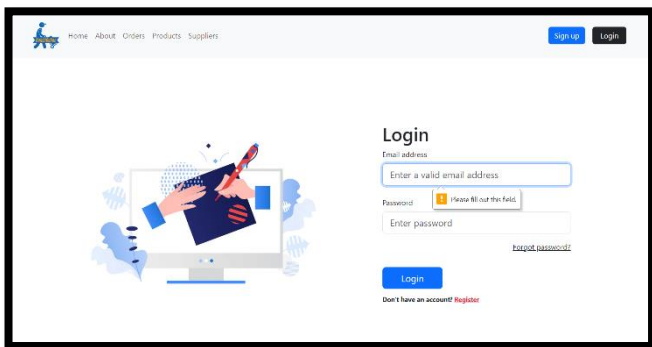
Login

Don't have an account? [Register](#)

Si la structure de l'adresse mail n'est pas respectée

```
<form @submit.prevent="login">
  <!-- Email input -->
  <div class="form-outline mb-4">
    <label class="form-label" >Email address</label>
    <input type="email" v-model="email" required class="form-control form-control-lg"
      placeholder="Enter a valid email address" />
  </div>

  <!-- Password input -->
  <div class="form-outline mb-3">
    <label class="form-label" >Password</label>
    <input type="password" v-model="password" required class="form-control form-control-lg"
      placeholder="Enter password" />
  </div>
</form>
```



Home About Orders Products Suppliers [Sign up](#) [Login](#)

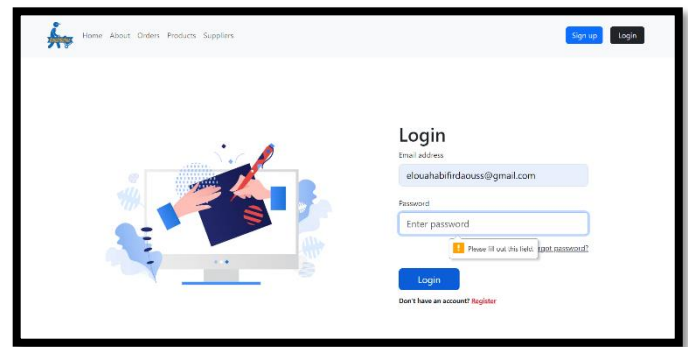
Login

Email address
Enter a valid email address

Password Please fill out this field
Enter password Forgot password?

[Login](#)

Don't have an account? [Register](#)



Home About Orders Products Suppliers [Sign up](#) [Login](#)

Login

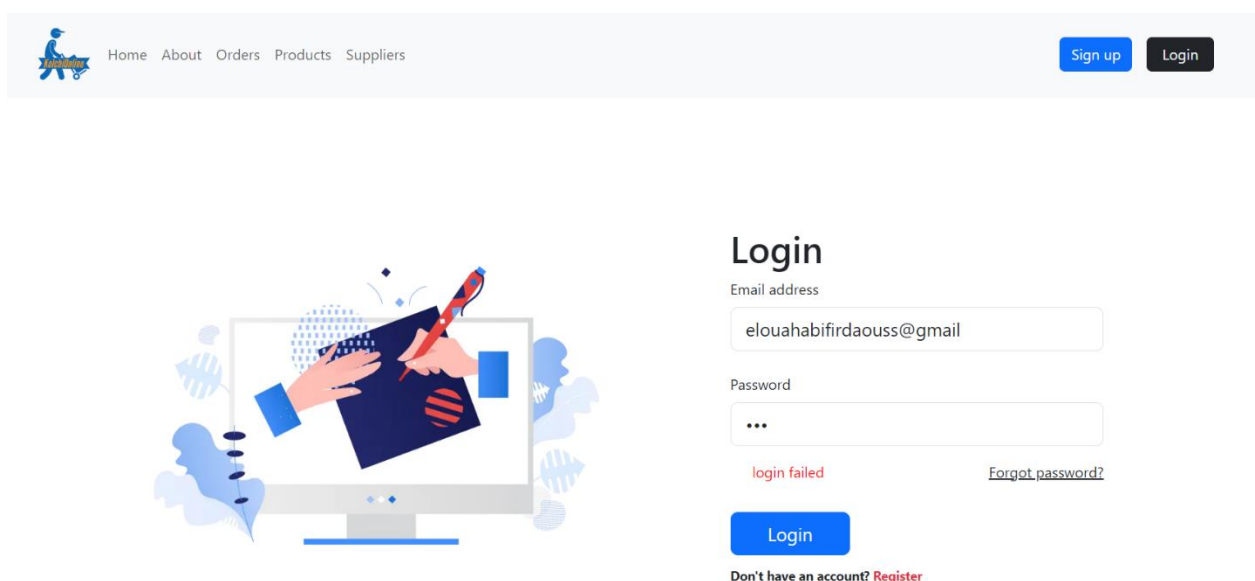
Email address
elouahabifirdaouss@gmail.com

Password Please fill out this field
Enter password Forgot password?

[Login](#)

Don't have an account? [Register](#)

Si l'adresse mail ou le mot de passe est incorrecte.



Home About Orders Products Suppliers [Sign up](#) [Login](#)

Login

Email address
elouahabifirdaouss@gmail

Password
...

login failed [Forgot password?](#)

[Login](#)

Don't have an account? [Register](#)

```

2 usages
3 public class LoginRequest {
4     3 usages
5     private String email;
6     3 usages
7     private String password;
8
9     no usages
10    public LoginRequest() {}
11
12    1 usage
13    public String getEmail() { return email; }
14
15    no usages
16    public void setEmail(String email) {
17        this.email = email;
18    }
19
20    1 usage
21    public String getPassword() { return password; }
22
23    no usages
24    public void setPassword(String password) { this.password = password; }
25
26    no usages
27    public LoginRequest(String email, String password) {
28        this.email = email;
29        this.password = password;
30    }

```

```

57
58 no usages
59 @PostMapping(value = "/client/login")
60 public ResponseEntity<?> login(@RequestBody LoginRequest loginRequest)
61 { //return clientService.login(loginRequest.getEmail(), loginRequest.getPassword());
62     Optional<Client> clientOptional = clientService.login(loginRequest.getEmail(), loginRequest.getPassword());
63
64     if (clientOptional.isPresent()) {
65         Client client = clientOptional.get();
66         return ResponseEntity.status(HttpStatus.OK).body(client);
67     } else {
68         return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("Error: Failed to LogIn !!!");
69     }
70 }
71

```

```

methods: {
  login() {
    const loginData = {
      email: this.email,
      password: this.password,
    };

    axios.post( url: 'http://localhost:8080/Client/login', loginData)
      .then((response) => {
        console.log('login successful!', response.data);
        localStorage.setItem('client', JSON.stringify(response.data));
        localStorage.setItem('loggedIn', true);
        this.$router.push('/Orders');
      })
      .catch((error) => {
        console.error('login failed:', error);
        this.errorMessage='login failed';
      });
  },

```

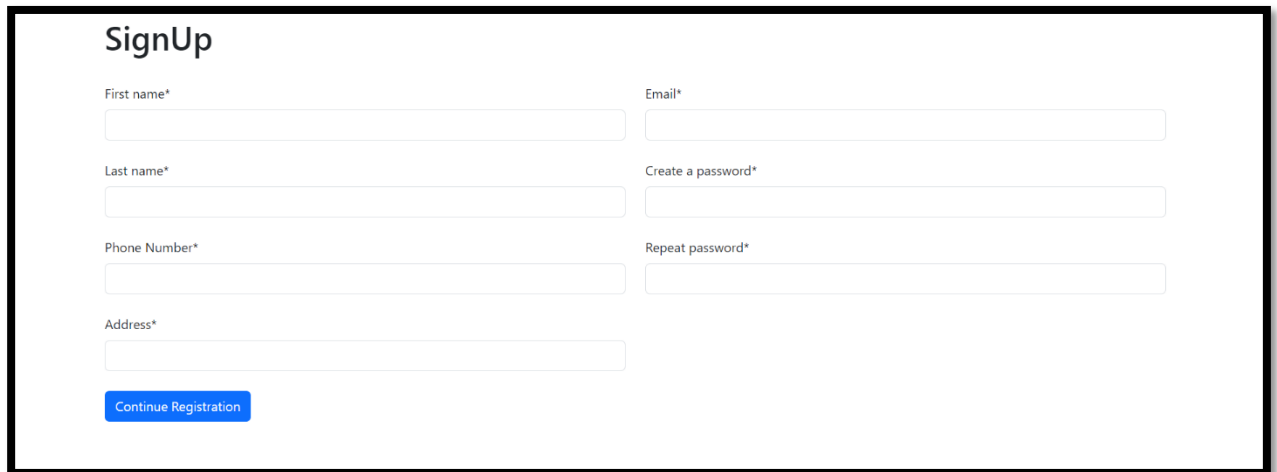
```

<div class="d-flex justify-content-between align-items-center">
  <!-- Checkbox -->
  <div class="form-check mb-0">
    <!--input class="form-check-input me-2" type="checkbox" value="" id="form2Example3"
    <label class="form-check-label" for="form2Example3">
      <div v-if="errorMessage" class="error-message">
        {{errorMessage}}
      </div>
    </label>
  </div>
  <a href="#" class="text-body">Forgot password?</a>
</div>

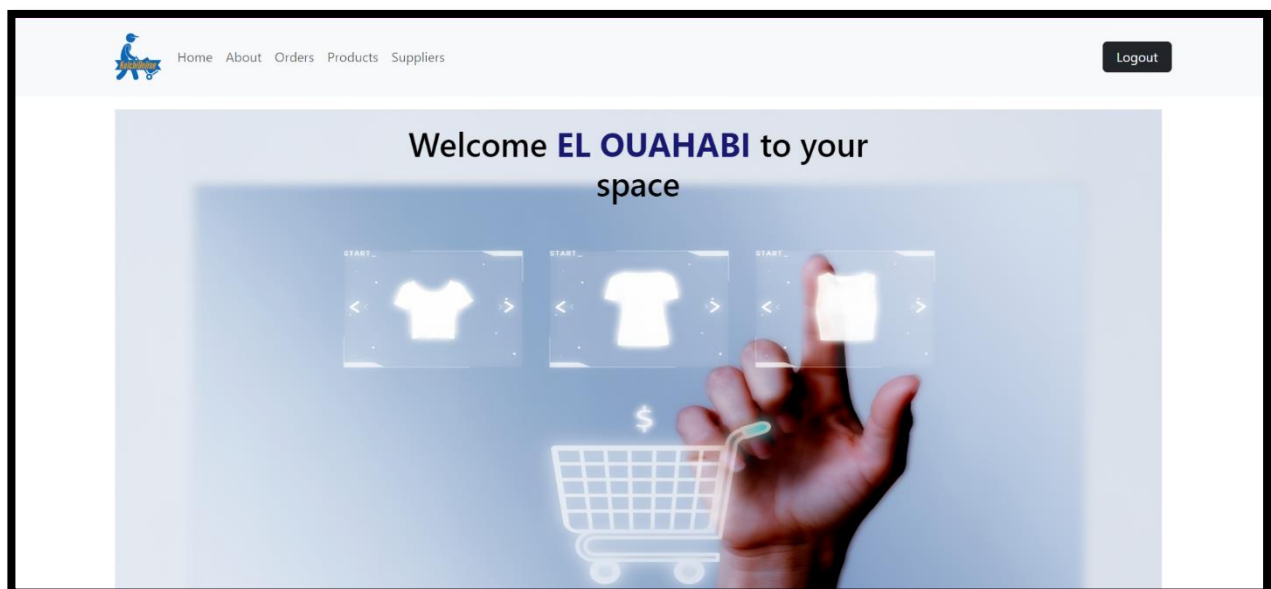
```


Page d'inscription :

A partir de cette page, l'internaute remplit un formulaire d'inscription pour devenir un membre et de pouvoir accéder à l'application

A registration form titled 'SignUp' with a white background and a black border. It contains several input fields for user information: 'First name*', 'Email*', 'Last name*', 'Create a password*', 'Phone Number*', 'Repeat password*', and 'Address*'. Each field is a simple white rectangle with a thin border. At the bottom left, there is a blue button with white text that says 'Continue Registration'.

La page d'accueil des utilisateurs affiche un message de bienvenue avec le nom de l'utilisateur .



```
<div class="container">
  

  <div class="centered"> <h1 class="wlc">Welcome
    <b class="name-cl">{{ client.lastName }}</b> to your space </h1></div>
</div>
</div>
```

Affiche la liste des commandes avec le nom de l'utilisateur et l'adresse mail connecter avec.




[Home](#) [About](#) [Orders](#) [Products](#) [Suppliers](#)

Welcome **EL OUAHABI** to Your Orders List

Email: elouahabifirdaouss@gmail.com

```
<h1>Welcome <b class="name-cl">{{ client.lastName }}</b> to Your Orders List</h1>
<div>
  <p><strong>Email:</strong> {{ client.email }}</p>
</div>
```


[Home](#)
[About](#)
[Orders](#)
[Products](#)
[Suppliers](#)
Logout

Welcome **FIRDAOUSS** To Product List

Email: elouahabifirdaouss@gmail.com

Now you are in Order ID : 6


Get back to list Orders
Reload
Suppliers

All Products

ID	Name	Price	stock quantity	supplier ID	number of items	ADD
1	Produit1	500	50000	2	<input type="text" value="1"/>	ADD
2	Produit2	800	3000	1	<input type="text"/>	ADD
22	Produit3	100	25000	1	<input type="text"/>	ADD
23	Produit4	200	1000	6	<input type="text"/>	ADD
ID	Name	Price	stock quantity	supplier ID	number of items	ADD

Cette interface affiche la liste des produits avec la possibilité d'ajouter un produit à une commande il est mentionné aussi le prix, la quantité, ainsi que le nombre de produit que vous voulez acheter.

Après, on peut consulter la liste des commandes déjà passées


[Home](#)
[About](#)
[Orders](#)
[Products](#)
[Suppliers](#)
Logout

Welcome **EL OUAHABI** to Your Orders List

Email: elouahabifirdaouss@gmail.com

New Order
Reload

Orders

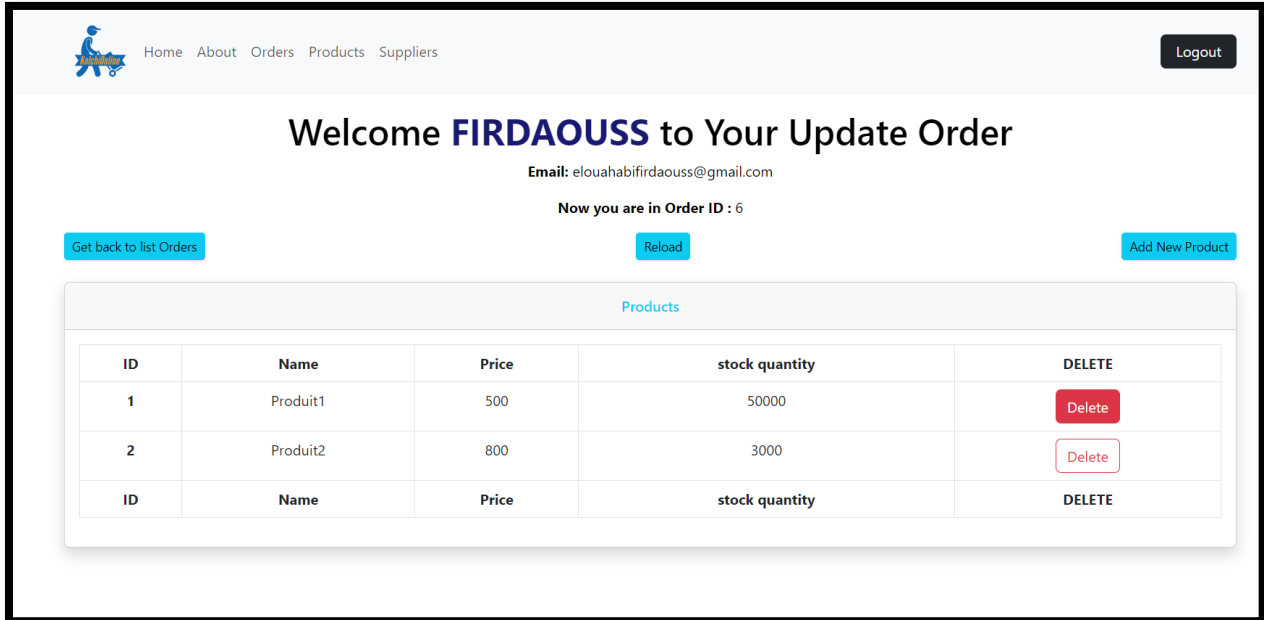
ID	Invoice	Date	Number of products	UPADTE	DELETE	MORE
1	1500	2023-09-06	1	Update	Delete	Show Products
6	6000	2023-09-06	2	Update	Delete	Show Products
9	1300	2023-09-06	2	Update	Delete	Show Products
10	3000	2023-09-06	3	Update	Delete	Show Products
12	5000	2023-09-06	4	Update	Delete	Show Products
ID	Invoice	Date	Number of products	UPADTE	DELETE	MORE

Dans cette interface, on a plusieurs boutons, nouvelle commande nous permet de retourner sur la liste des produits, les boutons modifier supprimer et afficher la commande, ainsi il y a le bouton de se déconnecter.

Aussi, la page affiche le détail de la commande, le montant total, la date de passation de commande, combien de produits se compose cette dernière.

Boutons modifier :

Là on peut soit supprimer un produit, ou ajouter un autre ou plusieurs, avec un détail sur chaque produit qui compose la commande.



Home About Orders Products Suppliers Logout

Welcome **FIRDAOUSS** to Your Update Order

Email: elouahabifirdaouss@gmail.com

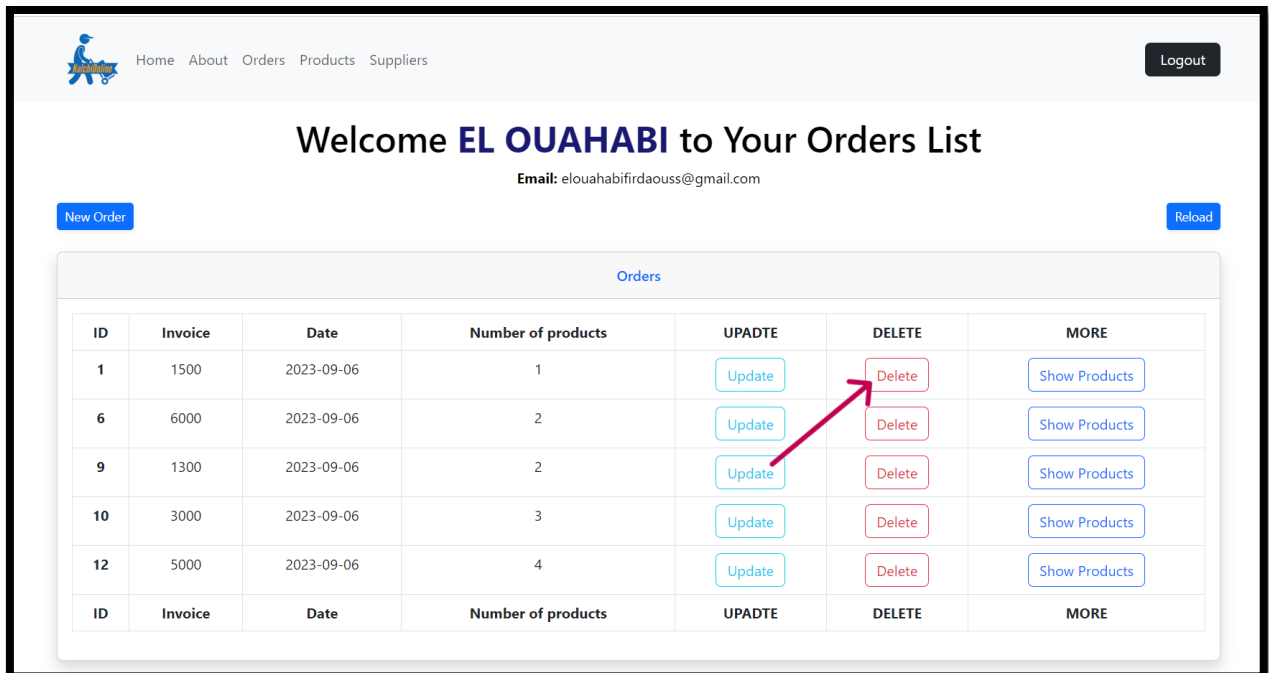
Now you are in Order ID : 6

Get back to list Orders Reload Add New Product

Products

ID	Name	Price	stock quantity	DELETE
1	Produit1	500	50000	Delete
2	Produit2	800	3000	Delete
ID	Name	Price	stock quantity	DELETE

Boutons supprimer :



Home About Orders Products Suppliers Logout

Welcome **EL OUAHABI** to Your Orders List

Email: elouahabifirdaouss@gmail.com

New Order Reload

Orders

ID	Invoice	Date	Number of products	UPADTE	DELETE	MORE
1	1500	2023-09-06	1	Update	Delete	Show Products
6	6000	2023-09-06	2	Update	Delete	Show Products
9	1300	2023-09-06	2	Update	Delete	Show Products
10	3000	2023-09-06	3	Update	Delete	Show Products
12	5000	2023-09-06	4	Update	Delete	Show Products
ID	Invoice	Date	Number of products	UPADTE	DELETE	MORE

```

},
deleteOrder(orderId)
{
    axios.delete( url: `http://localhost:8080/Commande/${orderId}` )
        .then((response) => {
            console.log(response);
            window.location.reload();
        })
        .catch((error) => {
            console.error(" delete failed !!! ", error);
        });
},

```

Boutons affiche détail commande :

En haut de la page il y a l'id de la commande, Aussi il y a le nom des produits, la quantité qu'on a choisi et la quantité de stock.

```

ClientService.java
30 public void deleteClient(Long id) { clientRepository.deleteById(id); }
    1 usage
41 public List<Commande> getListOrderByIdClient(Long id) {
42     Client client = clientRepository.findById(id).orElse( other: null);
43
44     if(client != null){
45         return client.getListCommandes();
46     }
47     return null;
48 }


```

```

ClientContrôleur.java
9
    no usages
0 @GetMapping(value = "/Client/{id}/Commande")
1 public List<Commande> getClientIsCommande(@PathVariable Long id) {
2     return clientService.getListOrderByIdClient(id);
3 }

```



 [Home](#) [About](#) [Orders](#) [Products](#) [Suppliers](#) [Logout](#)

Welcome **FIRDAOUSS** to Your Order ID: 6

Email: elouahabifirdaouss@gmail.com

Now you are in Order ID : 6

[Get back to your Orders list](#) [Reload](#) [Add New Product](#)

Products

ID	Name	Price	stock quantity
1	Produit1	500	50000
2	Produit2	800	3000
ID	Name	Price	stock quantity

Conclusion

Notre projet a consisté à concevoir et mettre en place une application de **gestion d'achat en ligne**. Pour concevoir ce travail, nous avons présenté dans un premier chapitre le projet et ses spécifications. Puis nous avons passé au chapitre conception et analyse qui concerne l'étape de l'analyse et de spécification des besoins afin que nous parvenions à une vue claire des différents besoins pour déterminer les fonctionnalités de l'application.

La phase de conception nous a permis d'entrer plus en profondeur dans l'analyse et de parler de l'architecture de l'application. Par la suite il a fallu modéliser le système, en utilisant les diagrammes de cas d'utilisation et de séquence.

Finalement, au niveau du chapitre réalisation, nous avons présenté l'environnement logiciel de travail, ainsi que les outils de développement utilisés. Sans oublier la structure du projet et les principales interfaces graphiques de l'application. On a pu, grâce à ce projet, améliorer nos compétences sociales et élargir nos connaissances, et de travailler en équipe ce qui nous a permis d'acquérir de nouvelles perspectives, de s'entraider et de communiquer efficacement. La réalisation de ce projet nous a permis d'utiliser nos connaissances en SQL, Vuejs, et spring boot, langages et méthodes que nous avons déjà eu l'occasion de manipuler lors de modules du DCA. Au-delà d'une révision, le projet réalisé nous a permis de découvrir certaines fonctionnalités que nous n'avons jamais rencontrées.