

# Technologie Web 2

## Rapport de Projet

January 22, 2018

## Contents

<b>I</b>	<b>Introduction</b>	<b>3</b>
<b>1</b>	<b>Fonctionnalités principales</b>	<b>3</b>
1.1	La création et la gestion de comptes . . . . .	3
1.2	Recherche d'un produit . . . . .	3
1.3	Actions disponibles sur le site . . . . .	4
1.4	Valorisation de comptes . . . . .	4
<b>2</b>	<b>Liste des acteurs</b>	<b>4</b>
<b>3</b>	<b>Contraintes matérielles et logicielles</b>	<b>4</b>
<b>II</b>	<b>Spécifications</b>	<b>5</b>
<b>1</b>	<b>Diagrammes de cas d'utilisation</b>	<b>5</b>
<b>2</b>	<b>Diagramme de navigation</b>	<b>10</b>
<b>III</b>	<b>Conception préliminaire</b>	<b>10</b>
<b>1</b>	<b>Diagrammes de sequence système</b>	<b>11</b>
<b>2</b>	<b>Diagramme de séquence</b>	<b>14</b>
<b>3</b>	<b>Diagramme de classe</b>	<b>15</b>
<b>IV</b>	<b>Conception détaillée</b>	<b>16</b>
<b>V</b>	<b>Implémentation et tests</b>	<b>16</b>

<b>1</b>	<b>Les différentes étapes du développement</b>	<b>16</b>
1.1	Construction des pages avant-plan (front-end) . . . . .	16
1.1.1	Choix techniques et bibliothèques . . . . .	17
1.2	Conception de base de données . . . . .	17
1.2.1	Choix techniques et bibliothèques . . . . .	19
1.3	Développement Arrière-plan(DAO) . . . . .	19
1.3.1	Choix techniques . . . . .	22
1.4	Développement Arrière-plan(servlets et filtrages) . . . . .	22
<b>2</b>	<b>Guide utilisateur</b>	<b>25</b>
2.1	Déploiement de l'application web . . . . .	25
2.2	Guide Admin . . . . .	25
<b>3</b>	<b>Présentation des résultats</b>	<b>25</b>
<b>VI</b>	<b>Conclusions et perspectives</b>	<b>32</b>

## List of Figures

1	cas d'utilisation : création/gestion de compte . . . . .	5
2	cas d'utilisation : rechercher un produit . . . . .	6
3	cas d'utilisation : connexion . . . . .	6
4	cas d'utilisation : consulter un produit . . . . .	7
5	cas d'utilisation : ajouter/supprimer un produit . . . . .	7
6	cas d'utilisation : commenter un produit . . . . .	8
7	cas d'utilisation : gestion du panier d'achat . . . . .	8
8	cas d'utilisation : paiement . . . . .	9
9	cas d'utilisation : noter un produit . . . . .	9
10	Diagramme de navigation . . . . .	10
11	DSS - gestion de compte . . . . .	11
12	DSS - création de compte . . . . .	12
13	DSS - consultation d'un produit . . . . .	12
14	DSS - Suppression d'un produit . . . . .	13
15	Diagramme de séquence - gestion du panier . . . . .	14
16	Diagramme de classe . . . . .	15
17	Diagramme de classes participantes . . . . .	16
18	Base de données MySQL . . . . .	18
20	Communication avec DAO . . . . .	19
19	Communication uniquement avec JDBC . . . . .	19
21	précisions sur le DAO . . . . .	20
22	exemple 1 méthode DAO . . . . .	21
23	exemple 2 méthode DAO . . . . .	22
24	Filtre ForeAuthFilter . . . . .	23
25	méthode update pour ProductServlet . . . . .	24

26	Screenshot 1: Page de login du site OpenWallet . . . . .	26
27	Screenshot 2: Page d'accueil du site OpenWallet . . . . .	26
28	Screenshot 3: Page d'inscription au site OpenWallet . . . . .	27
29	Screenshot 4: Page d'un produit du site OpenWallet . . . . .	27
30	Screenshot 5: Page de produit classé par catégorie. Ici c'est la catégorie Homme. . . . .	28
31	Screenshot 6: Page d'accueil du site OpenWallet, après le login. .	28
32	Screenshot 7: Page d'accueil du site OpenWallet, avec affichage des catégories. . . . .	29
33	Screenshot 8: Page de la saisie de l'adresse d'expédition. . . . .	29
34	Screenshot 9: Page montrant le contenu du chariot. . . . .	30
35	Screenshot 10: Page Admin listant les utilisateurs du site Open-Wallet. . . . .	30
36	Screenshot 11: Page Admin listant tous les produits du site Open-Wallet. . . . .	31
37	Screenshot 12: Page Admin listant tous les commandes du site OpenWallet . . . . .	31
38	Screenshot 13: Page Admin listant toutes les catégories du site OpenWallet. . . . .	32

## Part I

# Introduction

L'objectif de ce projet est de réaliser un site de vente en ligne, sur le même modèle principe que des sites comme [www.amazon.com](http://www.amazon.com) ou [leboncoin.fr](http://leboncoin.fr).

## 1 Fonctionnalités principales

On retrouvera sur le site les fonctionnalités présentés ci-après :

### 1.1 La création et la gestion de comptes

Elle permettra la création/édition d'un compte via les informations données par l'utilisateur (identité, email, pseudo et mot de passe). Il existera aussi un compte administrateur qui pourra modifier le contenu du site en ligne.

### 1.2 Recherche d'un produit

La recherche d'un produit peut se faire de deux manières :

1. via la barre de recherche
2. via la recherche avancée, qui sera multi-critères (nom/ type d'objet / prix / note de l'objet / nombre de ventes réalisées / réputation du vendeur)

### **1.3 Actions disponibles sur le site**

Les autres fonctions du site sont listés ci-après :

- deux types de connexions : utilisateur simple (acheteur ou vendeur) ou administrateur,
- consulter un produit,
- acheter un produit,
- vendre un produit,
- système de paiement,
- commenter et noter un produit,
- accéder au support client

### **1.4 Valorisation de comptes**

Le système de valorisation de comptes qui sera utilisée sera le suivant :

- un compte Premium ayant des avantages supplémentaires lors de l'achat de produits par rapport à l'utilisateur simple.
- un compte Super Premium ayant encore plus d'avantages.

## **2 Liste des acteurs**

Cette liste reprend et résume les éléments concernant les types d'utilisateur énoncés dans la partie 1 Fonctionnalités principales

1. Utilisateur simple : cet utilisateur peut être soit acheteur ou vendeur sur le site; il n'a aucun droit de modification sur le site.
2. Administrateur : cet utilisateur a tous les droits sur le site, il peut en modifier le contenu à volonté.

## **3 Contraintes matérielles et logicielles**

1. Le site doit être accessible depuis les navigateurs suivants : Google Chrome, Mozilla Firefox.
2. Résolutions d'écran supportées : le site web doit pouvoir s'adapter à tous types d'afficheurs.

# Part II

## Spécifications

Dans cette partie, nous présenterons les différents diagrammes qui découlent de l'analyse fonctionnelle du problème.

### 1 Diagrammes de cas d'utilisation

Nous présentons ici une liste non-exhaustive des diagrammes d'utilisation issue de la compréhension du besoin métier.

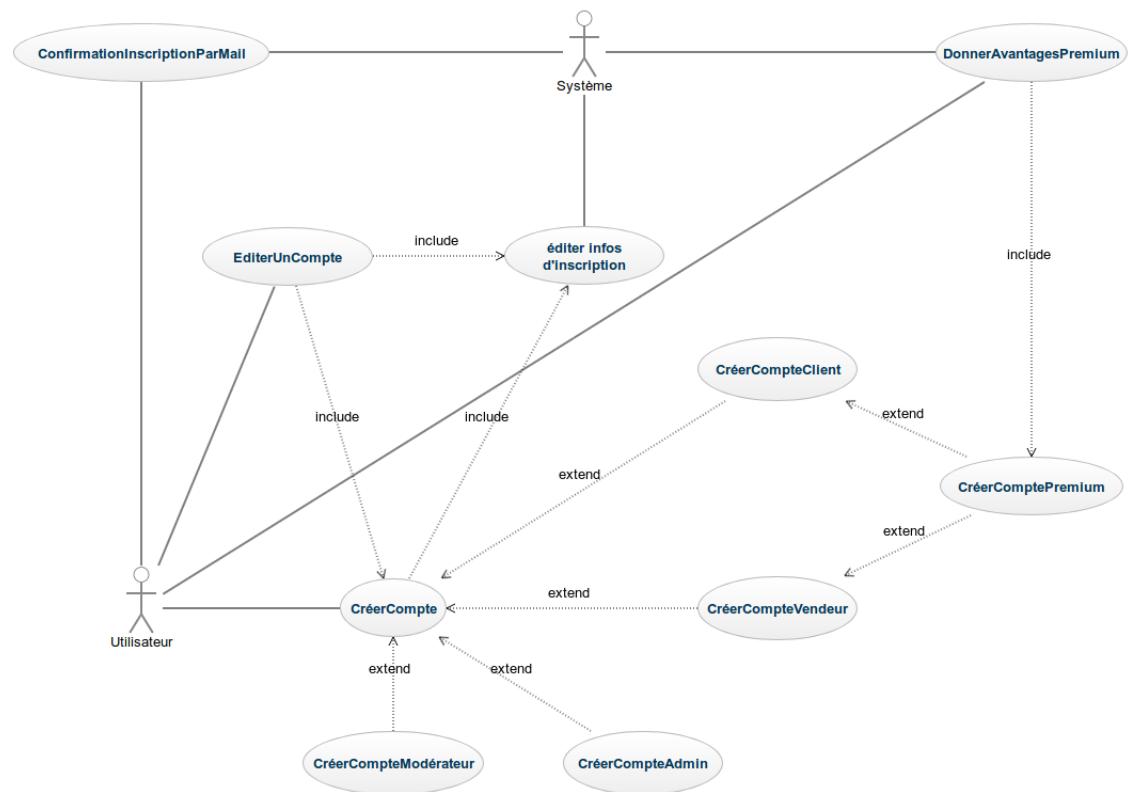


Figure 1: cas d'utilisation : création/gestion de compte

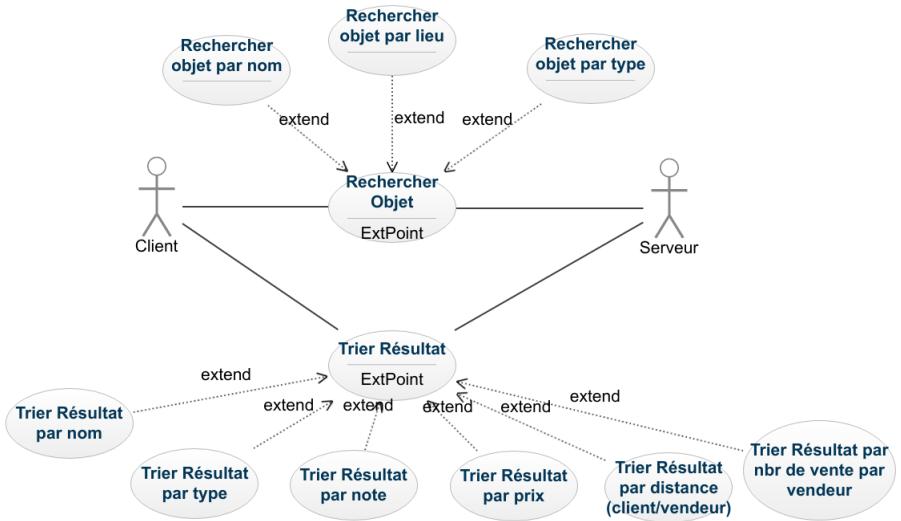


Figure 2: cas d'utilisation : rechercher un produit

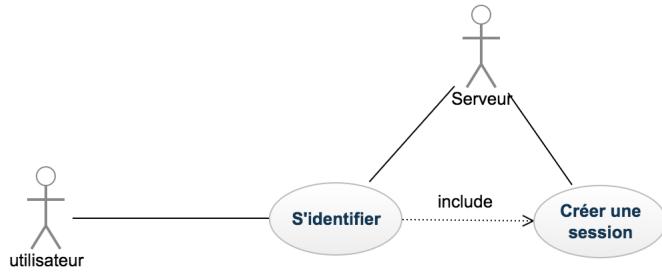


Figure 3: cas d'utilisation : connexion

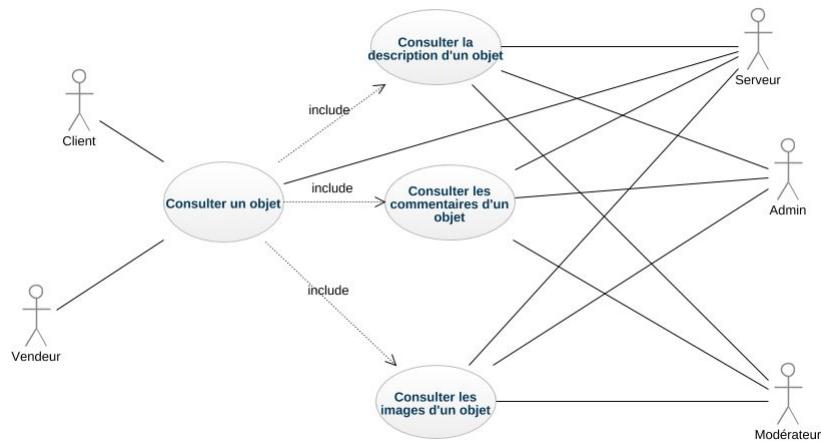


Figure 4: cas d'utilisation : consulter un produit

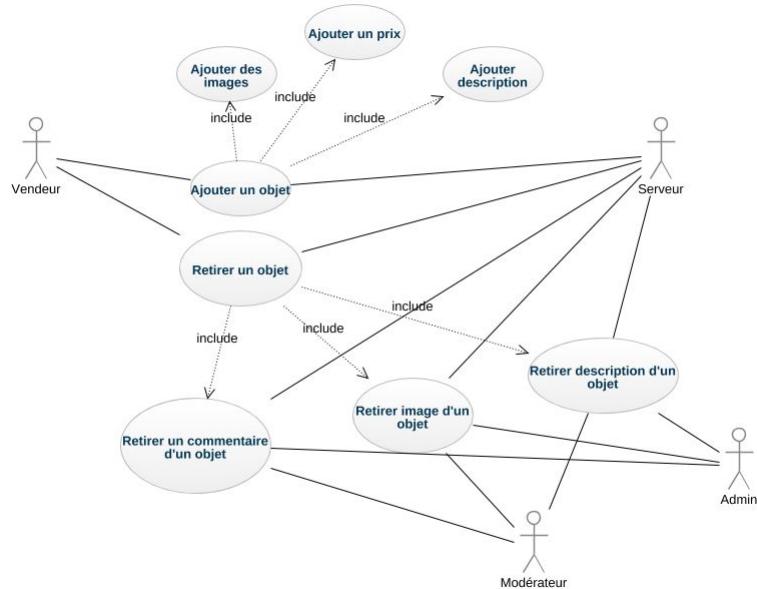


Figure 5: cas d'utilisation : ajouter/supprimer un produit

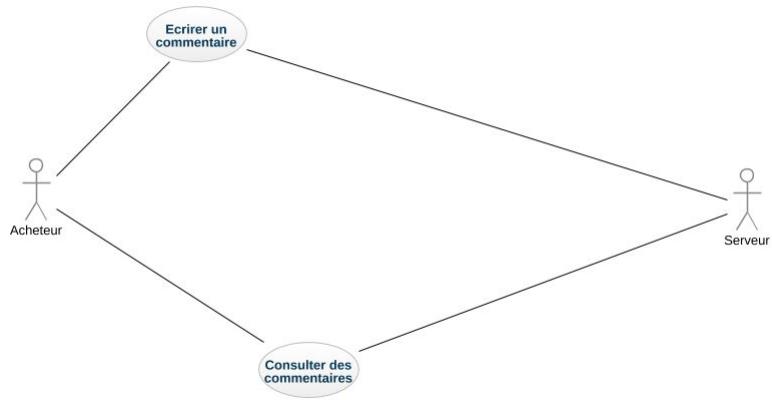


Figure 6: cas d'utilisation : commenter un produit

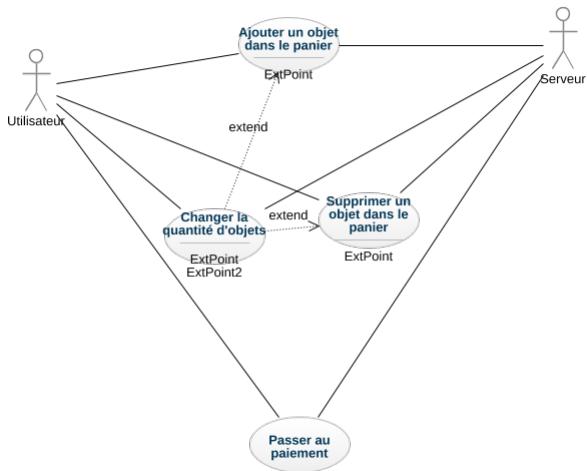


Figure 7: cas d'utilisation : gestion du panier d'achat

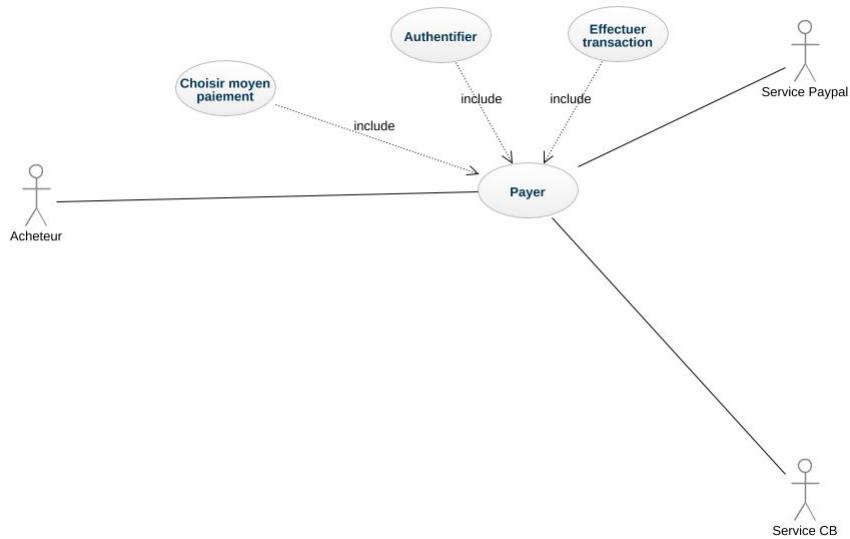


Figure 8: cas d'utilisation : paiement

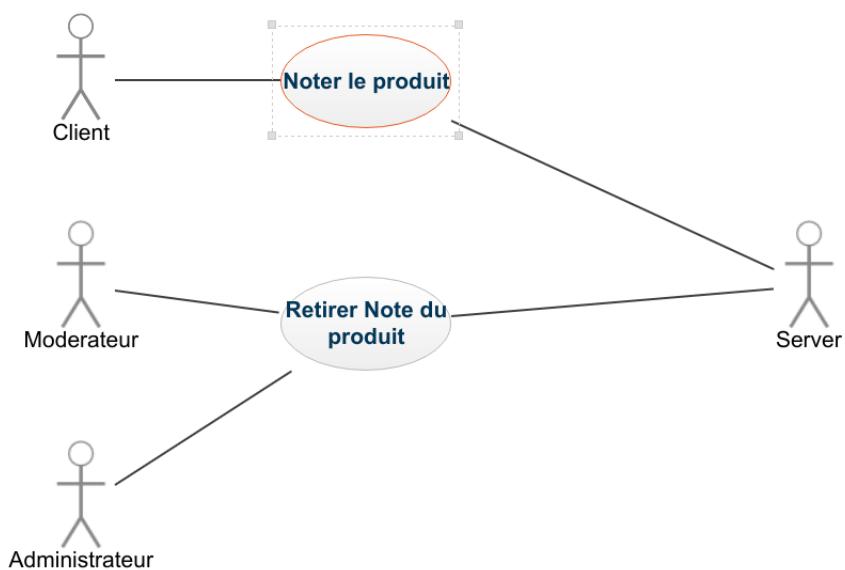


Figure 9: cas d'utilisation : noter un produit

## 2 Diagramme de navigation

Le diagramme de navigation nous permettra de modéliser le parcours d'un utilisateur sur le site.

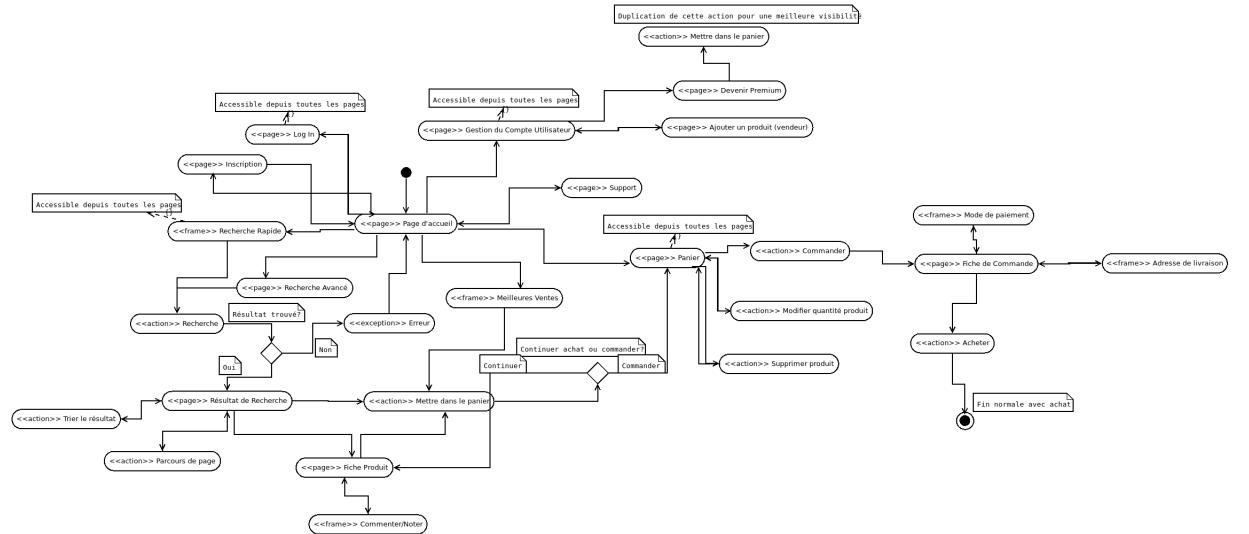


Figure 10: Diagramme de navigation

# Part III

## Conception préliminaire

### 1 Diagrammes de sequence système



Figure 11: DSS - gestion de compte

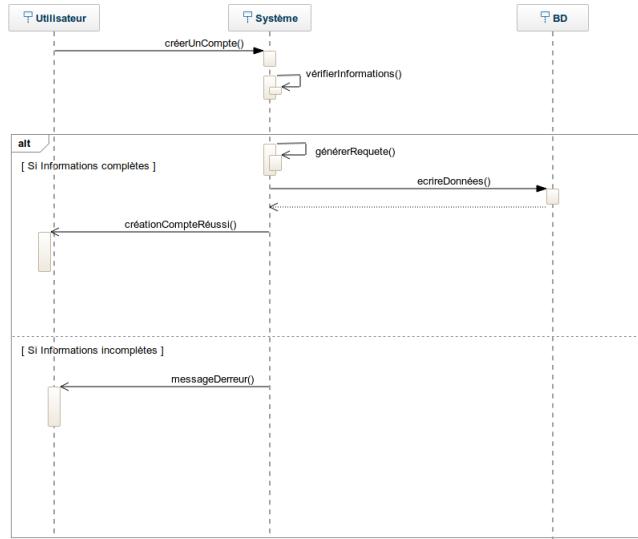


Figure 12: DSS - création de compte

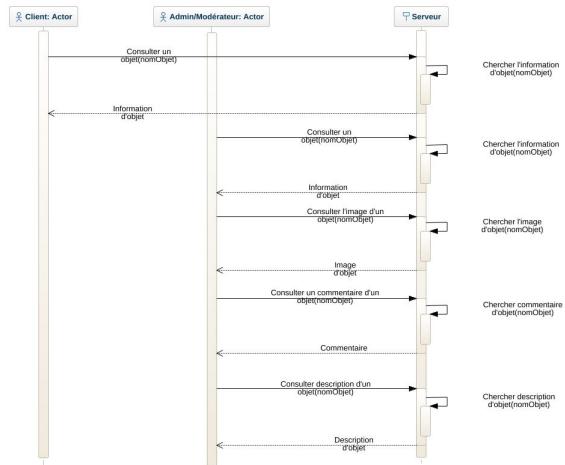


Figure 13: DSS - consultation d'un produit

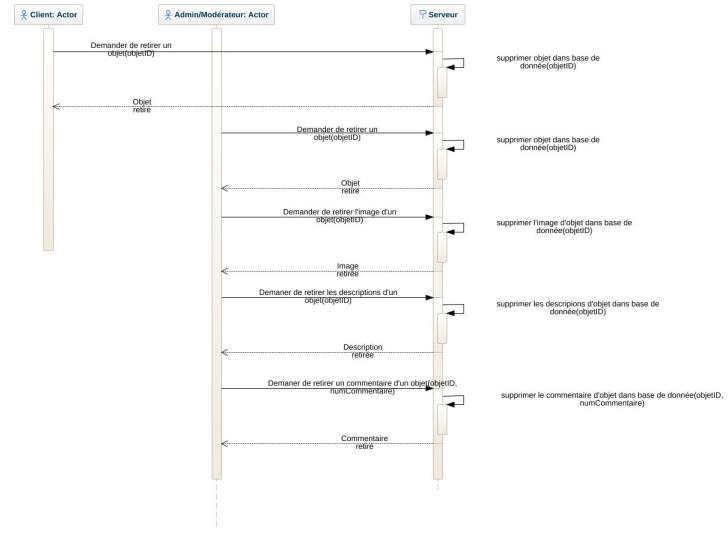


Figure 14: DSS - Suppression d'un produit

## 2 Diagramme de séquence

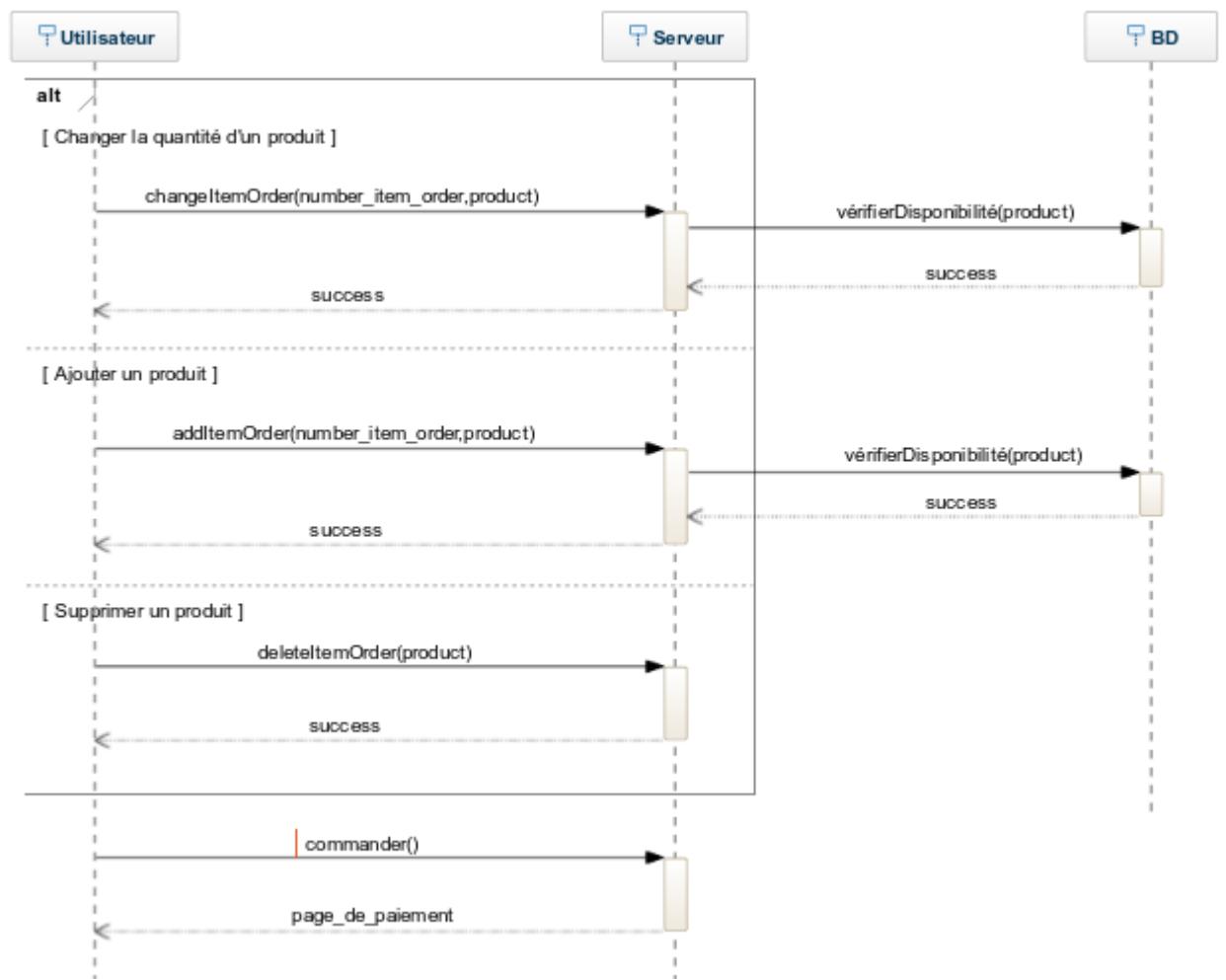


Figure 15: Diagramme de séquence - gestion du panier

### 3 Diagramme de classe

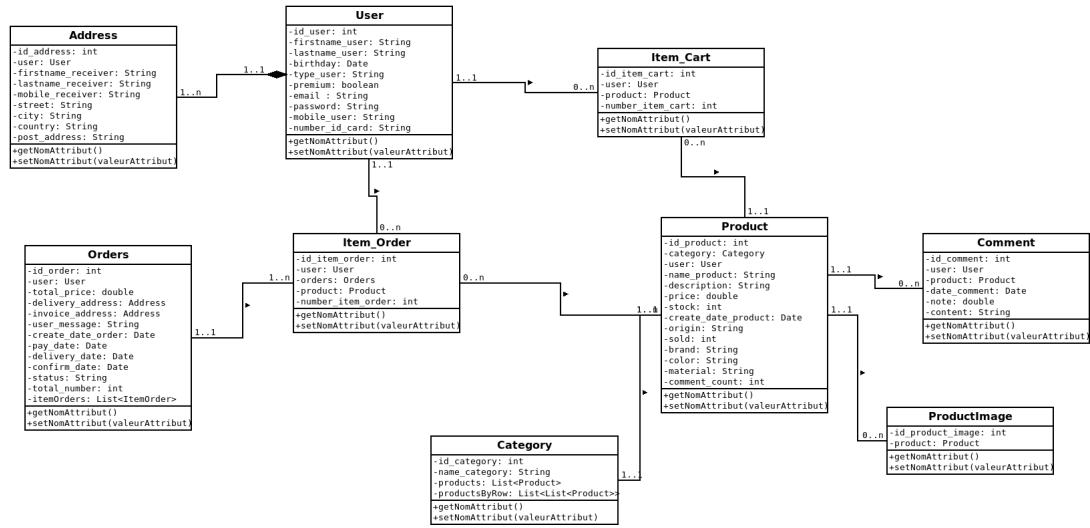


Figure 16: Diagramme de classe

# Part IV

## Conception détaillée

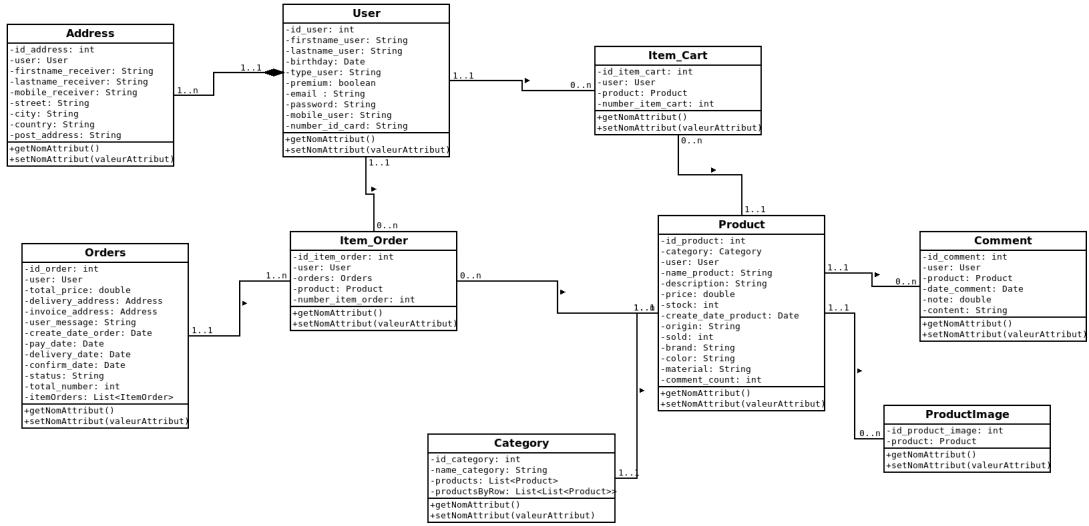


Figure 17: Diagramme de classes participantes

# Part V

## Implémentation et tests

Dans cette partie, nous allons dans un premier temps détailler les étapes de la programmation ainsi que les choix techniques, puis les tests effectués, un aperçu du résultat obtenu et enfin les principales difficultés rencontrées.

### 1 Les différentes étapes du développement

#### 1.1 Construction des pages avant-plan (front-end)

Nous avons dans un premier temps mis en place la structure principale du site avec les diagrammes de conception. Pour réaliser les fonctionnalités dans notre diagramme de navigation, on a déterminé qu'il y a au total 13 pages type à réaliser:

- La page pour gérer le compte utilisateur
- La page pour ajouter en spécifier un produit à vendre

- La page pour faire une recherche avancée dans notre liste des produits
- La page pour gérer le panier et passer au paiement
- La Homepage
- La page pour inscription
- La page pour s'identifier
- La page pour accéder au information détaillé du commande
- La page du produit à vendre
- La page pour afficher les résultats de recherche
- La page pour devenir compte premium
- La page pour ajouter une adresse de livraison
- La page pour service clientèle

Cette première partie est principalement axée sur le côté HTML/CSS quasiment sans aucun développement back-end. Nous avons développé ces pages pour un style de design uniifié et pour vérifier notre conception.

### 1.1.1 Choix techniques et bibliothèques

- **HTML/CSS:** Le HTML (« HyperText Mark-Up Language ») est un langage dit de « balisage » ou de « structuration » permettant la conception de pages web avec des balises de formatage. Les balises permettent d'indiquer la façon dont doivent être présentés le document et les liens qu'il établit avec d'autres documents. Le CSS (« Cascading Style Sheets » : feuilles de style en cascade) est un langage informatique complétant le HTML. Alors que le HTML structure la page Web, le CSS va la mettre en forme en y apportant du style.
- **Modèle de design Mem[é]nto by Your Inspiration Themes:** On a utilisé un modèle de design web Memento. Memento est un thème brandable et créatif avec une mise en page propre, entreprise et réactive., Il permet de définir n'importe quelle couleur à des éléments tels que des arrière-plans, des liens, du texte, des liens de menu, etc. et changer complètement l'apparence de site en quelques secondes.

## 1.2 Conception de base de données

La réalisation de notre site web a requis la création d'une base de données.

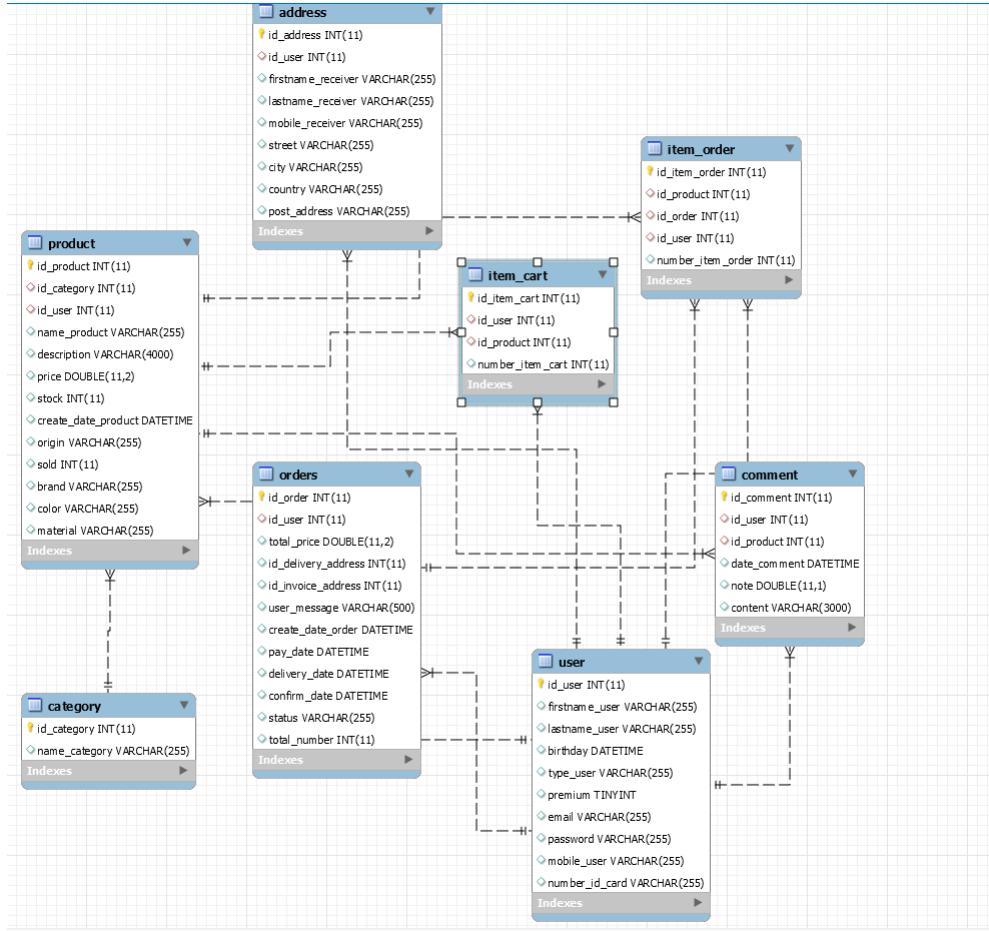


Figure 18: Base de données MySQL

Dans notre cas, il nous a tout d'abord fallu créer ces 8 tables pour la gestion de site web. La table user est destinée à gérer toutes les informations utilisateurs autres que le pseudonyme et le mot de passe. Ensuite on construit les autres 3 tables de base: orders, category et address. Puis, lorsque nous avons commencé category et user, ça nous permet d'initialiser la table produit avec 2 clés étrangères id\_user et id\_category. Après on construit la table orders pour rajouter la possibilité d'insérer plusieurs produits. La table comment est reliée au orders et user. Enfin, pour obtenir un état intermédiaire entre produit, user et orders, il nous a donc fallu rajouter deux tables: item\_cart et item\_order concernant les tableaux récapitulatifs des items perçus par l'association avec les paniers et les commandes.



Figure 20: Communication avec DAO

### 1.2.1 Choix techniques et bibliothèques

- SQL: est un langage informatique normalisé servant à exploiter des bases de données relationnelles. La partie langage de manipulation des données de SQL permet de rechercher, d'ajouter, de modifier ou de supprimer des données dans les bases de données relationnelles.

## 1.3 Développement Arrière-plan(DAO)

Après avoir implémenté les classes Javabeans avec ses propres méthodes et fini le script SQL pour la construction BD, Il a fallu trouver une solution pour la communication. Si on emploie que les méthodes JDBC, on a actuellement une représentation globale comme suit :



Figure 19: Communication uniquement avec JDBC

En théorie, MVC est ainsi bien respecté. Mais avec ce modèle on risque de mélanger le code responsable des traitements métier au code responsable du stockage des données en pratique. Il rend aussi les tests difficile à réaliser (il est impossible de changer de mode de stockage. Que ce soit vers un autre SGBD, voir vers un système complètement différent d'une base de données, cela impliquerait une réécriture complète de tout le modèle, car le code métier est mêlé avec et dépendant du code assurant le stockage). Au lieu de faire communiquer directement nos objets métier avec la base de données, on utilise une autre solution pour améliorer la performance et la robustesse de notre système. C'est le couche DAO qui va ensuite de son côté communiquer avec le système de stockage. On obtient une nouvelle représentation globale comme suit:

Pour l'implémentation du modèle(MVC), nous avons procédé en deux étapes :

1. **Création des classes métier (javabeans)** : En correspondance à notre conception et construction BD, on crée 8 classes java: Address, Category,

Comment, ItemCart, ItemOrder, Orders, Product et User. Chaque classe métiers contient des attribut du classe correspond à celui de BD. Il nous a donc fallu rajouter une classe qui contenait seulement les chemins des images que nous avons identifiés par l'identifiant de produit et celui de l'image. Les classes métiers possèdent que les méthodes pour get et set les attributs correspondant mais ne touche pas la BD.

## 2. Crédation des classes DAO :

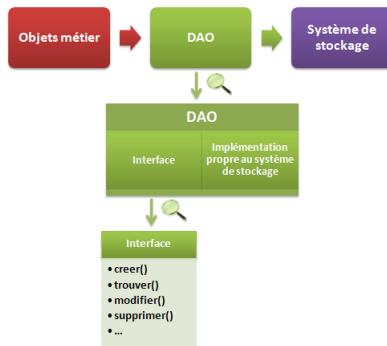


Figure 21: précisions sur le DAO

La couche modèle est constituée de la couche métier (en rouge) et de la couche données (en vert). La couche métier n'a connaissance que des interfaces décrivant les objets de la couche données. Ainsi, peu importe le système de stockage final utilisé, du point de vue du code métier les méthodes à appeler ne changent pas, elles seront toujours celles décrites dans l'interface. C'est uniquement l'implémentation qui sera spécifique au mode de stockage.

Dans notre cas, on construire en premier temps la classe DBUtil pour établir le connection entre la base de données et les classes DAO. Dans chaque classe DAO spécifiée on génère les commandes SQL par retirer les propres attributs depuis notre objets Java Bean et les insérer dans les commandes SQL prédéfinie.

Par exemple:

```

public void add(Product bean) {

    String sql = "insert into Product values(null,?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
    try (Connection c = DBUtil.getConnection(); PreparedStatement ps = c.prepareStatement(sql)) {

        ps.setInt(1, bean.getCategory().getId_category());
        ps.setInt(2, bean.getUser().getId_user());
        ps.setString(3, bean.getName_product());
        ps.setString(4, bean.getDescription());
        ps.setDouble(5, bean.getPrice());
        ps.setInt(6, bean.getStock());
        ps.setTimestamp(7, DateUtil.dtt(bean.getCreate_date_product()));
        ps.setString(8, bean.getOrigin());
        ps.setInt(9, bean.getSold());
        ps.setString(10, bean.getBrand());
        ps.setString(11, bean.getColor());
        ps.setString(12, bean.getMaterial());
        ps.execute();

        ResultSet rs = ps.getGeneratedKeys();
        if (rs.next()) {
            int id = rs.getInt(1);
            bean.setId_product(id);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

Figure 22: exemple 1 méthode DAO

Quand nous avons une nouvelle instance de Product Bean et nous voulons de l'enregistre dans notre base de données, Nous utiliserons la méthode add dans Product DAO. Cette méthode prépare une ligne de commande SQL "insert into Product values(null,?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)" et on le remplit avec les attribut de notre objet Product.

```

public Product get(int id_product) {
    Product bean = new Product();

    try (Connection c = DBUtil.getConnection(); Statement s = c.createStatement();) {

        String sql = "select * from Product where id_product = " + id_product;

        ResultSet rs = s.executeQuery(sql);

        if (rs.next()) {

            int id_category = rs.getInt("id_category");
            Category category = new CategoryDAO().get(id_category);
            int id_user = rs.getInt("id_user");
            User user = new UserDao().get(id_user);
            String name_product = rs.getString("name_product");
            String description = rs.getString("description");
            double price = rs.getDouble("price");
            int stock = rs.getInt("stock");
            Date create_date_product = DateUtil.t2d(rs.getTimestamp("create_date_product"));
            String origin = rs.getString("origin");
            int sold = rs.getInt("sold");
            String brand = rs.getString("brand");
            String color = rs.getString("color");
            String material = rs.getString("material");

            bean.setCategory(category);
            bean.setUser(user);
            bean.setName_product(name_product);
            bean.setDescription(description);
            bean.setPrice(price);
            bean.setStock(stock);
            bean.setCreate_date_product(create_date_product);
            bean.setOrigin(origin);
            bean.setSold(sold);
            bean.setBrand(brand);
            bean.setColor(color);
            bean.setMaterial(material);
            bean.setId_product(id_product);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return bean;
}

```

Figure 23: exemple 2 méthode DAO

Dans le cas où l'information sur un produit spécifique avec son id\_product vient à manquer, on peut quand même générer une commande select et retirer toutes les informations nécessaires. Et puis on instancier un nouvel objet de Product, introduit ses attributs et le retour comme résultat.

### 1.3.1 Choix techniques

- **DAO** : Un objet d'accès aux données (en anglais data access object ou DAO) est un patron de conception (c'est-à-dire un modèle pour concevoir une solution) utilisé dans les architectures logicielles objets.

## 1.4 Développement Arrière-plan(servlets et filtrages)

Après nous avons eu les parties JavaBeans/BD fonctionnelles, Il faut des méthodes pour traiter les requêtes du client et pour la génération de pages jsp dynamiques.

namiques en utilisant le protocole http et donc un serveur web. Pour notre projet, on a choisi les servlets et les filtrages pour réaliser les liaisons avant-plan et arrière-plan.

Les filtres sont des objets qui peuvent transformer une requête ou modifier une réponse. Ce sont des préprocesseur de la requête avant qu'elle atteigne une servlet. Dans notre projet nous avons développé trois filtres pour traiter les requêtes clientèles. L'idée générale pour filtrage est d'extraire l'uri dans HttpServletRequest et trouver les noms de méthode dans l'uri. Par exemple, dans ForeAuthFilter on a :

```
public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain
    throws IOException, ServletException {
    HttpServletRequest request = (HttpServletRequest) req;
    HttpServletResponse response = (HttpServletResponse) res;
    String contextPath=request.getServletContext().getContextPath();

    String[] noNeedAuthPage = new String[]{
        "homepage",
        "checkLogin",
        "register",
        "loginAjax",
        "login",
        "product",
        "category",
        "search"};

    String uri = request.getRequestURI();
    uri =StringUtils.remove(uri, contextPath);
    if(uri.startsWith("/fore")&&uri.startsWith("/foreServlet")){
        String method = StringUtils.substringAfterLast(uri,"/fore" );
        if(!Arrays.asList(noNeedAuthPage).contains(method)){
            User user =(User) request.getSession().getAttribute("user");
            if(null==user){
                response.sendRedirect("login.jsp");
                return;
            }
        }
    }
    chain.doFilter(request, response);
}
```

Figure 24: Filtre ForeAuthFilter

La fonctionnalité de cette filtre contient deux parties:

1. Il extrait l'uri dans HttpServletRequest, supprime le contextpath dans l'uri et enregistre le nom de méthode dans un String.
- 2)Il vérifie si le nom de méthode est dans la liste des méthode (no need for authentification) , si oui on le laisser passage au filtre suivant. Sinon on vérifier l'information d'utilisateur dans la HttpServletRequest et déterminer si on rédiger la requête vers page d'authentification directement.

Le chaînage des filtres est dans l'ordre suivant: FiltrageBack -FiltrageAuthentification-FiltrageFront Après le filtrage, chaque méthode dans la requête est transformé comme une chaîne de caractère qui redirige vers une servlet nommé “xxxServlet” ou une page jsp.

Pour les servlets on a définit deux classes abstraites: BaseBackServlet et BaseForeServlet qui vont définir les méthodes de base (ajouter, modifier, supprimer, etc) pour les classes de notre modèle Javabeans. Chaque classe bean hérite une classe BackServlet pour traiter les requêtes. Par exemple:

```

public String update(HttpServletRequest request, HttpServletResponse response, Page page) {
    int id_product = Integer.parseInt(request.getParameter("id_product"));
    int id_category = Integer.parseInt(request.getParameter("id_category"));
    Category c = categoryDAO.get(id_category);
    int id_user = Integer.parseInt(request.getParameter("id_user"));
    User u = userDao.get(id_user);
    String name_product = request.getParameter("name_product");
    String description = request.getParameter("description");
    double price = Double.parseDouble(request.getParameter("price"));
    int stock = Integer.parseInt(request.getParameter("stock"));
    String[] date = request.getParameter("create_date_product").split("-");
    GregorianCalendar cal = new GregorianCalendar();
    cal.set(Integer.parseInt(date[0]), Integer.parseInt(date[1]), Integer.parseInt(date[2]));
    Date create_date_product = cal.getTime();
    String origin = request.getParameter("origin");
    int sold = Integer.parseInt(request.getParameter("sold"));
    String brand = request.getParameter("brand");
    String color = request.getParameter("color");
    String material = request.getParameter("material");
    int comment_count = Integer.parseInt(request.getParameter("comment_count"));

    Product p = new Product();

    p.setId_product(id_product);
    p.setCategory(c);
    p.setUser(u);
    p.setName_product(name_product);
    p.setDescription(description);
    p.setPrice(price);
    p.setStock(stock);
    p.setCreate_date_product(create_date_product);
    p.setOrigin(origin);
    p.setSold(sold);
    p.setBrand(brand);
    p.setColor(color);
    p.setMaterial(material);
    p.setComment_count(comment_count);

    productDAO.update(p);
    return "@admin_product_list?id_category="+p.getCategory().getId_category();
}

```

Figure 25: méthode update pour ProductServlet

La figure 24 est la méthode “mettre à jour” pour ProductServlet. Une fois que la servlet reçoit la requête de mise à jour, on extrait les nouveaux paramètres dans la requête, appelle le constructeur de ProductBean et ensuite le met à jour dans BD via productDAO. Après le traitement, la méthode renvoie vers la liste de produits pour afficher les résultats.

Choix technique: Servlet : Une servlet est une classe Java qui permet de créer dynamiquement des données au sein d'un serveur HTTP. Ces données sont le plus généralement présentées au format HTML, mais elles peuvent également l'être au format XML ou tout autre format destiné aux navigateurs web.

## 2 Guide utilisateur

Dans la partie ‘Guide d’utilisateurs’, nous vous présentons comment utiliser notre site web de vente comme l’administrateur.

### 2.1 Déploiement de l’application web

Il est nécessaire de lancer le script `~/OpenWallet/Code/web/compile_linux.sh` afin de compiler l’ensemble du code source et créer l’archive `.jar`. L’archive est alors disponible sous le nom `~/OpenWallet/Code/web/Openwallet.war`. Il suffit ensuite de copier cette archive dans le répertoire qui y est dédié dans serveur pour le déploiement.

### 2.2 Guide Admin

Etant l’admin du site, il est responsable de mettre à jour des produits et faire la gestion des commandes. En haut de toutes les pages d’administration, nous avons mis un navigateur pour entrer la page d’administration de catégorie, utilisateurs et commandes.

`http://127.0.0.1:8080/Openwallet/admin`

C’est le lien pour entrer la page d’administration. Nous allons être redirigés directement à la page d’administration de catégories. Pour ajouter les nouveaux produits, il faut d’abord ajouter sa catégorie. En bas de cette page, vous pouvez ajouter des nouvelles catégories. Il y a aussi le bouton pour éditer les catégories existantes.

Si quelqu’un veut ajouter des produit sous certain catégorie, donc il faut cliquer sur le bouton dans la colonne “Category Management”. Avec ce bouton, nous pouvons entrer la page d’administration de produits sous une catégorie déterminée. Donc, dans la page d’administration de produits, l’administrateur pourra ajouter les produits avec nombreuses propriétés.

## 3 Présentation des résultats

Dans cette partie, nous vous présenterons l’ensemble des pages réalisé sous forme de capture d’écran.

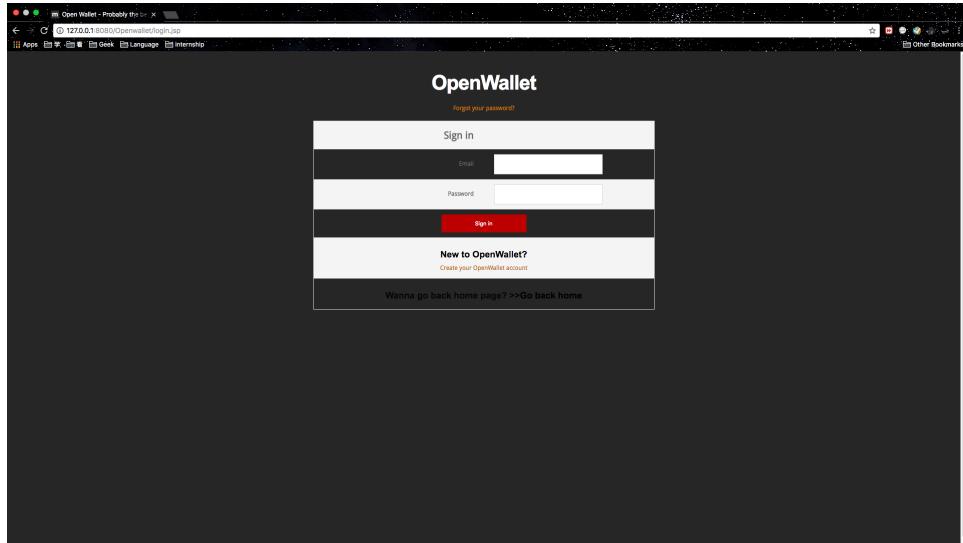


Figure 26: Screenshot 1: Page de login du site OpenWallet

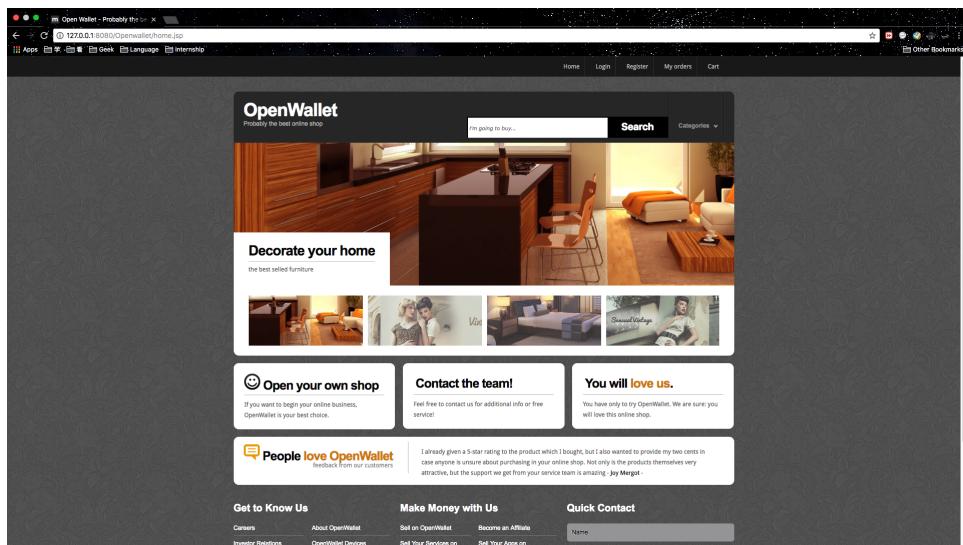


Figure 27: Screenshot 2: Page d'accueil du site OpenWallet

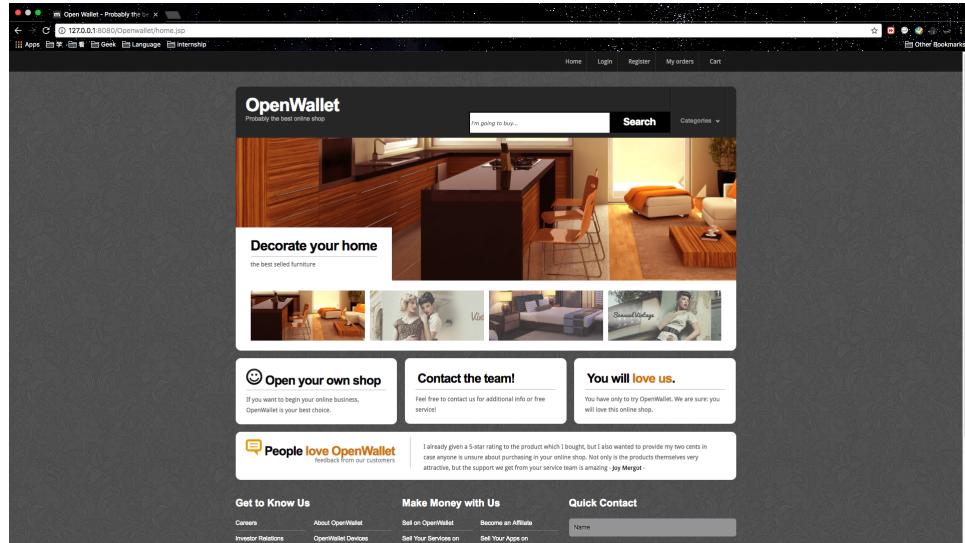


Figure 28: Screenshot 3: Page d'inscription au site OpenWallet

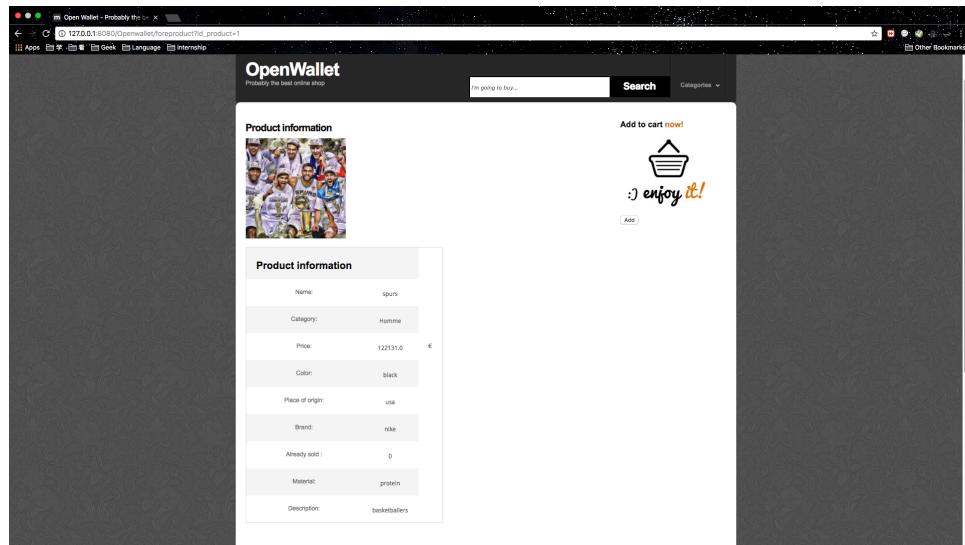


Figure 29: Screenshot 4: Page d'un produit du site OpenWallet

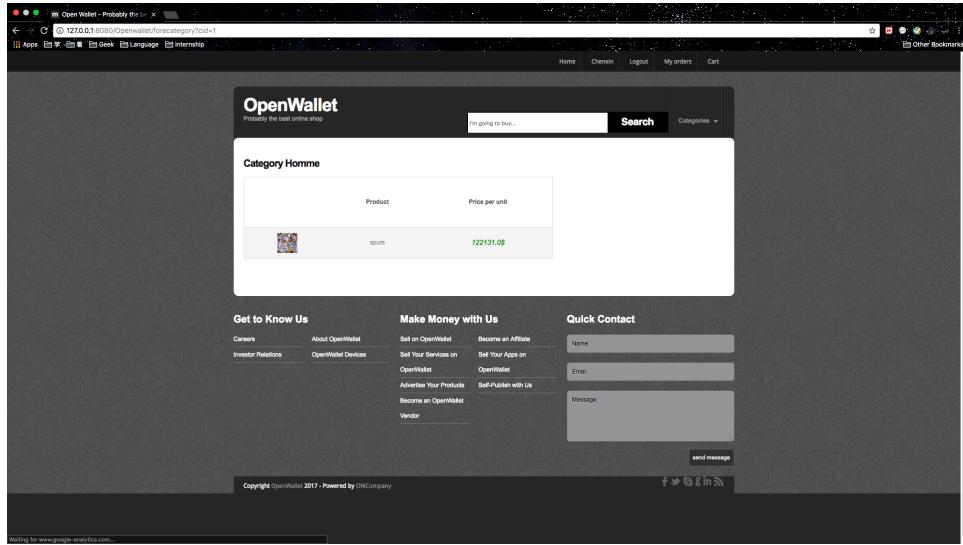


Figure 30: Screenshot 5: Page de produit classé par catégorie. Ici c'est la catégorie Homme.

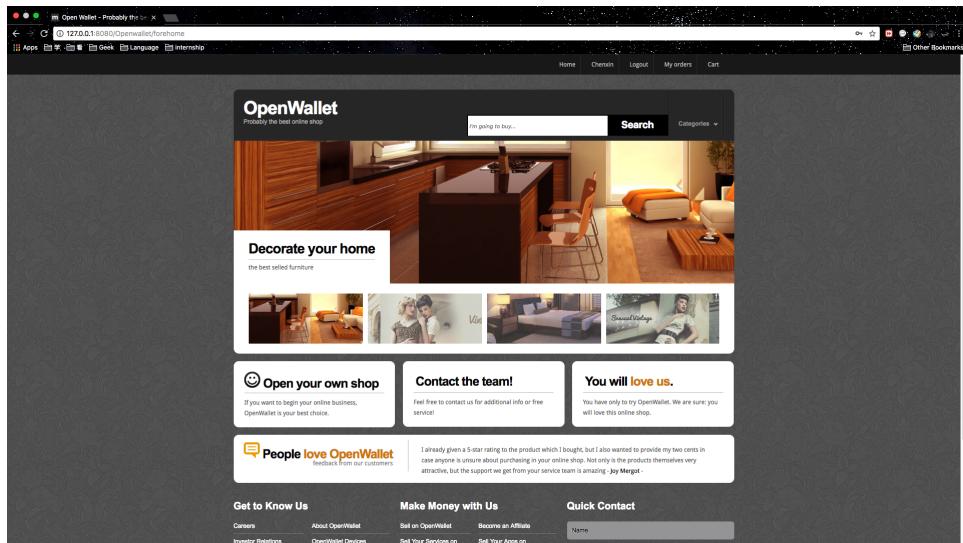


Figure 31: Screenshot 6: Page d'accueil du site OpenWallet, après le login.

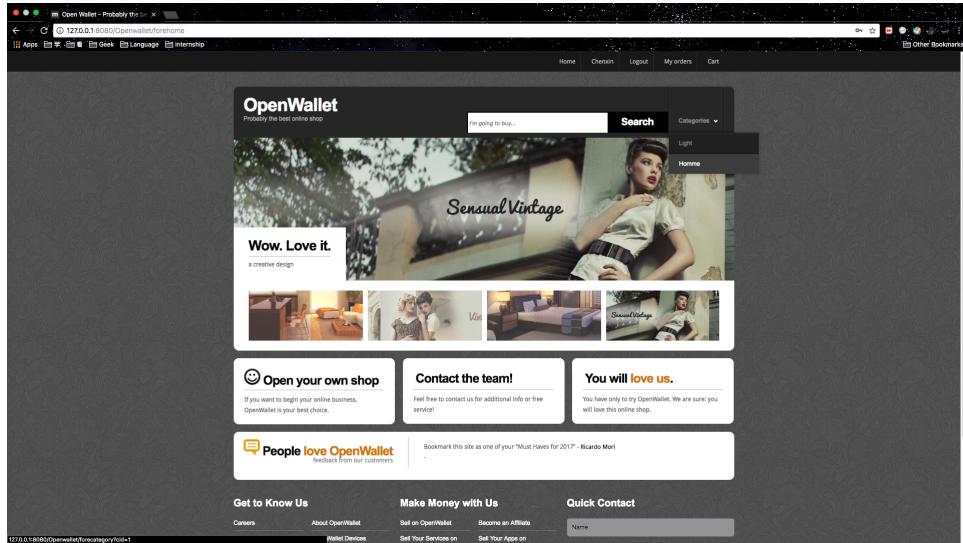


Figure 32: Screenshot 7: Page d'accueil du site OpenWallet, avec affichage des catégories.

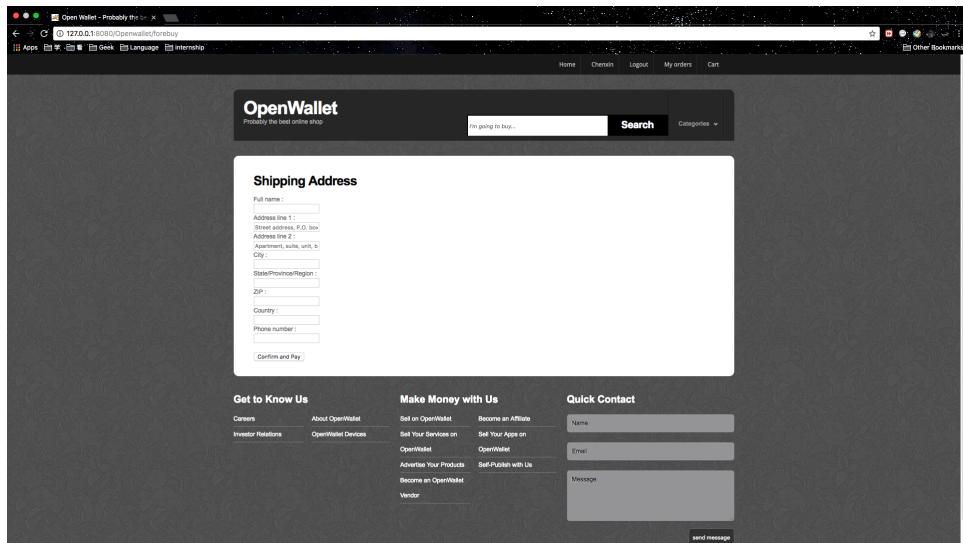


Figure 33: Screenshot 8: Page de la saisie de l'adresse d'expédition.

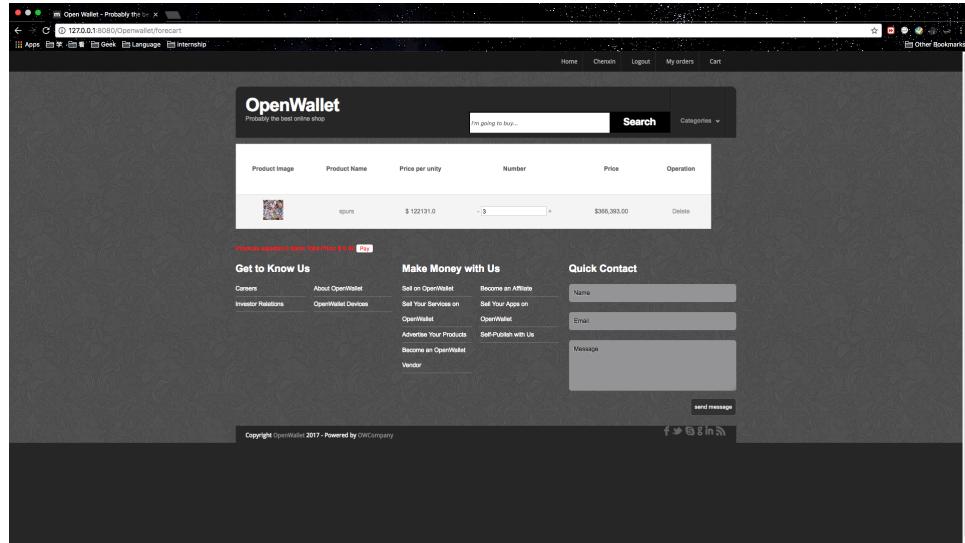


Figure 34: Screenshot 9: Page montrant le contenu du chariot.

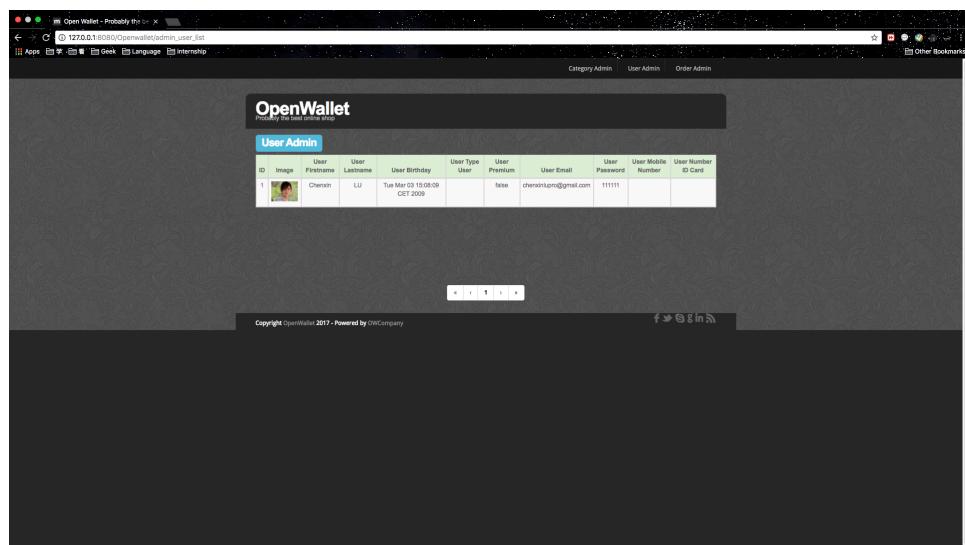


Figure 35: Screenshot 10: Page Admin listant les utilisateurs du site OpenWallet.

ID	Image	Seller	Name	Price	Stock	Create Date	Origin	Sold	Brand	Color	Material	Edit	Delete
1		spurs	122131.0	1		Tue Mar 02 18:08:20 CET 1999	usa	0	riks	black	protein	G	

Figure 36: Screenshot 11: Page Admin listant tous les produits du site Open-Wallet.

ID	Status	Price	Item Number	User Firstname	User Lastname	Creation Date	Pay Date	Delivery Date	Confirm Date	Delivery Address	Invoice Address	Edit

Figure 37: Screenshot 12: Page Admin listant tous les commandes du site OpenWallet

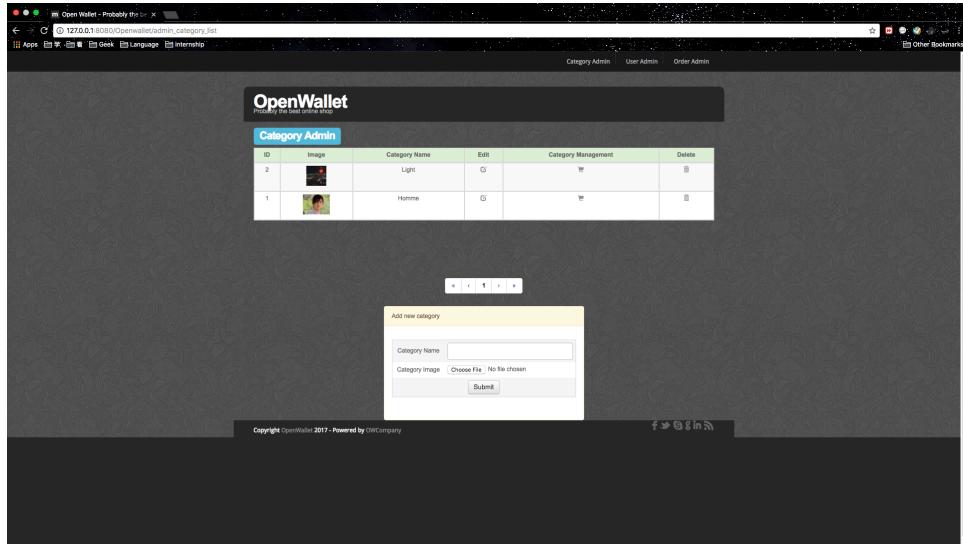


Figure 38: Screenshot 13: Page Admin listant toutes les catégories du site OpenWallet.

## Part VI

# Conclusions et perspectives

Afin de conclure ce projet, nous allons tout d'abord énoncer un bilan de nos réalisations et ensuite présenter les perspectives qui nous sont offertes au vu de nos travaux.

Au cours de ce projet, nous avons pu réaliser le site de ventes en nous basant sur un modèle de template pour le front-end et l'application du motif MVC, via l'implémentation de servlets et du pattern DAO (pour la communication avec une base de données MySQL) pour le back-end.

Les fonctionnalités non-réalisées sont les suivantes :

1. Changer le nombre d'un produit et supprimer un produit dans la page panier. Le code JS étant déjà réalisé, il suffirait ensuite de réaliser la servlet correspondante pour implémenter cette fonctionnalité.
2. Création d'une commande : on peut cliquer sur le bouton Pay dans la page panier, ça redirige vers la page de shipping address mais on ne peut pas récupérer les produit à acheter.
3. Premium, la fonctionnalité a été jugée peu importante. Il y aurait un attribut à ajouter dans la classe correspondante, et créer une page jsp mais la contrainte de temps est trop importante.

4. La fonctionnalité Commentaire n'est pas entièrement opérationnelle, on a une place pour afficher les commentaire s'ils existent. On ne peut pas avoir un commentaire que si on a fini une commande.
5. Pour les utilisateurs, afin de simplifier la tâche, l'administrateur est considéré comme le seul vendeur, il peut gérer les catégories et les produits sur les page dans le répertoire /web/admin.