

Royaume du Maroc



Ministère du Tourisme, du Transport Aérien,  
de l'Artisanat et de l'Economie Sociale



# **Rapport du mini projet**

**Sujet : Réalisation d'une application desktop de gestion  
des stocks pour un magasin d'ascenseurs**

**Réaliser par**

**ANAS SAHRAOUI**

# Sommaire

## 1. Contexte du Projet

- 1.1 Objectif
- 1.2 Périmètre du Projet

## 2. Fonctionnalités Détaillées

- 2.1 Gestion des Produits
- 2.2 Suivi des Stocks
- 2.3 Gestion des Commandes Clients
- 2.4 Administration des Ventes et Achats

## 3. Planification et Organisation

- 3.1 Diagramme de Gantt
- 3.2 Description du Diagramme de Gantt
- 3.3 Tableau des Exigences Techniques

## 4. Conception et Réalisation

- 4.1 Acteurs
- 4.2 Cas d'Utilisation Principaux
- 4.3 Cas d'Utilisation Complémentaires
- 4.4 Matrice de Priorités et Risques
- 4.5 Tableau Récapitulatif des Cas d'Utilisation
- 4.6 Diagramme de cas d'utilisation
- 4.7 Diagramme de classe
- 4.8 Modèle logique de données relationnel

## 5. Conception des cas d'utilisations

- 5.1 Cas d'utilisation : S'identifier
  - 5.1.1 Format textuel
  - 5.1.2 Diagramme de Séquence
  - 5.1.3 Diagramme de classes participantes
  - 5.1.4 Diagramme d'activité

- 5.1.5 Diagramme d'interaction
  - 5.1.6 Réalisation
- 5.2 Cas d'utilisation : Gestion des Commandes Fournisseurs
  - 5.2.1 Format textuel
  - 5.2.2 Diagramme de Séquence
  - 5.2.3 Diagramme d'activité
  - 5.2.4 Diagramme de classes participantes
  - 5.2.5 Diagramme d'interaction
  - 5.2.6 Réalisation
- 5.3 Cas d'utilisation : Gérer les produits
  - 5.3.1 Format textuel
  - 5.3.2 Diagramme de Séquence
  - 5.3.3 Diagramme de classes participantes
  - 5.3.4 Diagramme d'activité
  - 5.3.5 Diagramme d'interaction
  - 5.3.6 Réalisation
- 5.4 Cas d'utilisation : Gestion des clients et fournisseurs
  - 5.4.1 Format textuel
  - 5.4.2 Diagramme de Séquence
  - 5.4.3 Diagramme de classes participantes
  - 5.4.4 Diagramme d'activité
  - 5.4.5 Diagramme d'interaction
  - 5.4.6 Réalisation
- 5.5 Cas d'utilisation : Gestion des Commandes Clients
  - 5.5.1 Format textuel
  - 5.5.2 Diagramme de Séquence
  - 5.5.3 Diagramme d'activité
  - 5.5.4 Diagramme de classes participantes

- 5.5.5 Diagramme d'interaction
- 5.5.6 Réalisation

## **6. Tests et Validation**

- 6.1 Introduction
- 6.2 Outils utilisés
- 6.3 Configuration Maven
- 6.4 Méthodes JUnit utilisées
  - 6.4.1 AdministrateurDAOTest
  - 6.4.2 ClientTest
  - 6.4.3 CommandeClientTest
  - 6.4.4 DAOTest
  - 6.4.5 CommandeFournisseursTest
  - 6.4.6 FournisseurTest
  - 6.4.7 GestionDesCommandesClientsDAOTest
  - 6.4.8 GestionDesProduitsDAOTest
  - 6.4.9 GestionDesCommandesFournisseursDAOTest
  - 6.4.10 GérerLesClientsDAOTest
  - 6.4.11 GérerLesFournisseursDAOTest
  - 6.4.12 ProduitTest
- 6.5 Rapport de couverture de code (JaCoCo)
- 6.6 Conclusion des tests

## **7. Conclusion Générale**

- 7.1 Synthèse des réalisations
- 7.2 Bilan technique

## Table des Figures

**Figure 1 :** Diagramme de Gantt du projet

**Figure 2 :** Diagramme de cas d'utilisation générale

**Figure 3 :** Diagramme de classe générale

**Figure 4 :** Modèle logique de données relationnel

**Figure 5 :** Diagramme de Séquence - S'identifier

**Figure 6 :** Diagramme de classes participantes - S'identifier

**Figure 7 :** Diagramme d'activité - S'identifier

**Figure 8 :** Diagramme d'interaction - S'identifier

**Figure 9 :** Interface de S'identifier

**Figure 10 :** Diagramme de cas d'utilisation - Gestion des Commandes Fournisseurs

**Figure 11 :** Diagramme de Séquence - Gestion des Commandes Fournisseur

**Figure 12 :** Diagramme d'activité - Gestion des Commandes Fournisseur

**Figure 13 :** Diagramme de classes participantes - Gestion des Commandes Fournisseur

**Figure 14 :** Diagramme d'interaction - Gestion des Commandes Fournisseur

**Figure 15 :** Interface de Gestion des Commandes Fournisseur

**Figure 16 :** Diagramme de cas d'utilisation - Gérer les produits

**Figure 17 :** Diagramme de Séquence - Gérer les produits

**Figure 18 :** Diagramme de classes participantes - Gérer les produits

**Figure 19 :** Diagramme d'activité - Gérer les produits

**Figure 20 :** Diagramme d'interaction - Gérer les produits

**Figure 21 :** Interface de Gérer les produits

**Figure 22 :** Diagramme de cas d'utilisation - Gestion des clients et fournisseurs

**Figure 23 :** Diagramme de Séquence - Gestion des clients et fournisseurs

**Figure 24 :** Diagramme de classes participantes - Gestion des clients et fournisseurs

**Figure 25 :** Diagramme d'activité - Gestion des clients et fournisseurs

**Figure 26 :** Diagramme d'interaction - Gestion des clients et fournisseurs

**Figure 27 :** Interface de Gestion des clients et fournisseurs

**Figure 28 :** Diagramme de cas d'utilisation - Gestion des Commandes Clients

**Figure 29 :** Diagramme de Séquence - Gestion des Commandes Clients

**Figure 30 :** Diagramme d'activité - Gestion des Commandes Clients

**Figure 31 :** Diagramme de classes participantes - Gestion des Commandes Clients

**Figure 32 :** Diagramme d'interaction - Gestion des Commandes Clients

**Figure 33 :** Interface de Gestion des Commandes Clients

**Figure 34 :** Couverture de code du package entite (rapport JaCoCo)

**Figure 35 :** Couverture de code du package dao (rapport JaCoCo)

# **Cahier des Charges - Module de Gestion des Stocks magasin d'ascenseurs.**

## **1. Contexte du Projet**

### **1.1 Objectif**

Développer une application desktop de gestion des stocks pour un magasin d'ascenseurs.

- **Automatiser la gestion des stocks et des commandes.**
- **Améliorer le suivi des ventes et des achats.**

### **1.2 Périmètre du Projet**

L'application couvrira l'ensemble des processus de gestion des stocks, **incluant** :

- **Gestion des produits**
- **Suivi des stocks**
- **Gestion des commandes clients**

## **Fonctionnalités Détaillées**

### **1. Gestion des Produits**

Cette fonctionnalité permet à l'administrateur de gérer les produits du magasin. Elle inclut :

- **Ajout d'un produit** : Enregistrement d'un nouveau produit avec ses détails (nom, description, prix, quantité, catégorie, fournisseur, etc.).
- **Modification d'un produit** : Mise à jour des informations d'un produit existant.
- **Suppression d'un produit** : Suppression définitive d'un produit de la base de données.
- **Catégorisation des produits** : Classement des produits en différentes catégories pour une meilleure organisation et recherche.

### **2. Suivi des Stocks**

Cette fonctionnalité assure un contrôle précis du niveau des stocks :

- **Consultation du stock disponible** : Affichage des quantités de chaque produit en stock.

### 3. Gestion des Commandes Clients

Cette fonctionnalité permet de gérer les commandes passées par les clients :

- **Création d'une commande** : Enregistrement d'une nouvelle commande avec les produits sélectionnés et leurs quantités.
- **Suivi des commandes** : Consultation des commandes en cours, livrées ou annulées.

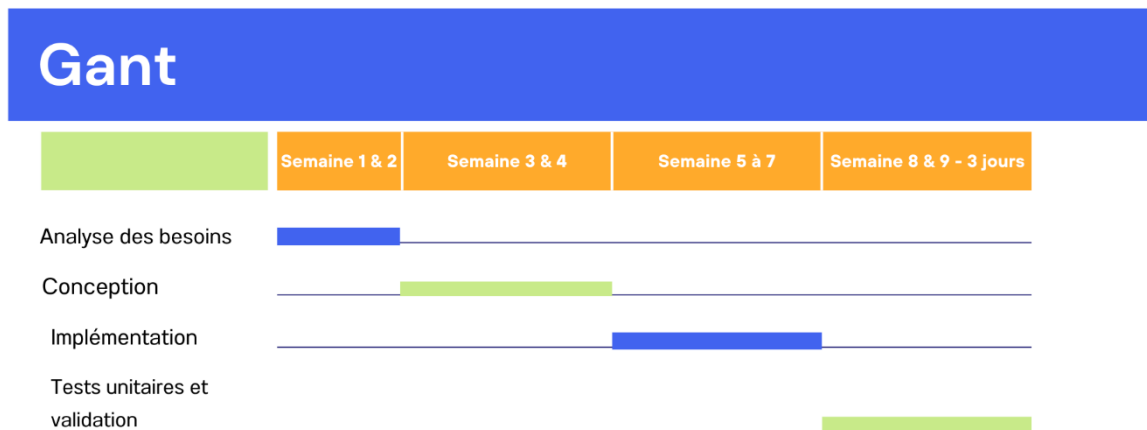
### 4. Administration des Ventes et Achats

Cette fonctionnalité englobe l'ensemble des processus liés aux transactions commerciales :

- **Gestion des fournisseurs** : Ajout et suppression des fournisseurs avec leurs informations.
- **Gestion des clients** : Enregistrement des clients avec leurs détails pour un suivi personnalisé .

## Diagramme de Gantt

Le **diagramme de Gantt** ci-dessous présente la planification des principales étapes du projet :



**Figure 1** : Diagramme de Gantt du projet

## Description du Diagramme de Gantt

Le **diagramme de Gantt** présenté illustre la planification du projet en suivant une approche structurée et séquentielle. Il couvre l'ensemble des phases du développement

## **Répartition des tâches sur 9 semaines(-3jr) :**

### **1 Semaine 1 & 2 – Analyse des besoins**

- Identification des exigences fonctionnelles et techniques.
- Rédaction du cahier des charges.
- Étude des besoins utilisateurs.

### **2 Semaine 3 & 4 – Conception**

- Élaboration des diagrammes UML (cas d'utilisation, classes, séquences...).
- Définition du MLDR pour structurer la base de données MySQL.
- Validation de la conception technique.

### **3 Semaine 5 à 7 – Implémentation**

- Développement de l'interface avec Java Swing.
- Intégration de la base de données MySQL.
- Ajout des fonctionnalités principales (gestion des stocks, commandes...).

### **4 Semaine 8 & 9 (3 jours) – Tests unitaires et validation**

- Tests avec JUnit pour vérifier les fonctionnalités.
- Analyse du code avec PMD, SonarLint et JaCoCo.
- Déploiement et corrections des éventuels bugs.



### Tableau des Exigences Techniques

Catégorie	Technologie / Outil	Description
Langage de programmation	Java (Swing)	Interface utilisateur en Java avec Swing pour une application desktop.
Tests unitaires	JUnit	Vérification du bon fonctionnement des classes et méthodes.
Méthodologie	RUP (Rational Unified Process)	Processus de développement structuré en 4 phases : Inception, Élaboration, Construction, Transition.
Modélisation	UML	Diagrammes UML pour représenter les cas d'utilisation, classes, séquences, etc.
Base de données	MySQL	Stockage des données avec un modèle logique et physique bien défini.
Conception BD	MLDR	Modèle logique des données relationnel pour structurer la base.
Qualité du code	PMD / SonarLint	Outils d'analyse statique du code pour détecter les mauvaises pratiques et les erreurs.
Couverture du code	JaCoCo	Génération de rapports sur la couverture des tests unitaires.
Environnement IDE	NetBeans	Développement et intégration du projet avec un environnement adapté à Java.
Gestion de version	GitHub	Hébergement du code source et suivi des modifications.

### Conclusion :

Ce cahier des charges formalise les exigences fonctionnelles, techniques et organisationnelles pour le développement du module de gestion des stocks.

# Conception et Réalisation

## Système de Gestion des Stocks

### 1. Acteurs

#### Administrateur

- Rôle principal : Supervision et gestion globale du système
- Responsabilités :

Gestion complète des produits

Validation et contrôle

### 2. Cas d'Utilisation Principaux

#### Gestion des Produits

Ajouter un produit

Saisie des informations du produit

Validation des données

Enregistrement en base de données

#### 1. Modifier un produit

Recherche du produit

Mise à jour des informations

Validation des modifications

#### 2. Supprimer un produit

Vérification des droits

Confirmation de suppression

#### 3. Catégoriser les produits

Création de catégories

Affectation des produits aux catégories

#### 4. Suivre les stocks

Suivi des niveaux de stock

### 3. Cas d'Utilisation Complémentaires

#### Gestion des Commandes Clients

1. Gestion des commandes
  - Création de commandes
  - Mise à jour du statut

#### Gestion des Fournisseurs

1. Gérer les fournisseurs
  - Ajout de fournisseurs
  - Mise à jour des informations
  - Suivi des approvisionnements

### 4. Matrice de Priorités et Risques

Cas d'Utilisation	Priorité	Risque	Itération
S'identifier	Haute	Moyen	1
Ajouter un produit	Haute	Bas	1
Modifier un produit	Haute	Bas	1
Suivre les stocks	Haute	Moyen	2
Catégoriser les produits	Moyenne	Bas	2
Supprimer un produit	Moyenne	Bas	2
Gestion des commandes	Moyenne	Bas	3
Gestion des fournisseurs	Moyenne	Bas	3

## 5. Tableau Récapitulatif des Cas d'Utilisation

Acteur	Cas d'Utilisation	Priorité	Risque	Itération	Description
Administrateur	UC1 : S'identifier	Haute	Moyen	1	Authentification au système
Administrateur	Gérer les produits	Haute	Moyen	1	Gestion complète des produits
Administrateur	Ajouter un produit	Haute	Bas	1	Ajout de nouveaux produits
Administrateur	Modifier un produit	Haute	Bas	1	Mise à jour des informations produit
Administrateur	Suivre les stocks	Haute	Moyen	2	Suivi des niveaux de stock
Administrateur	Catégoriser les produits	Moyenne	Bas	2	Organisation des produits
Administrateur	Supprimer un produit	Moyenne	Bas	2	Retrait de produits du système
Administrateur	Gestion des commandes clients	Moyenne	Bas	3	Suivi et gestion des commandes
Administrateur	Gestion des Commandes Fournisseurs	Moyenne	Bas	3	Suivi et gestion des commandes
Administrateur	Gestion des clients et fournisseurs	Moyenne	Moyen	3	Supervision des transactions

## 6. Diagramme de cas d'utilisation :

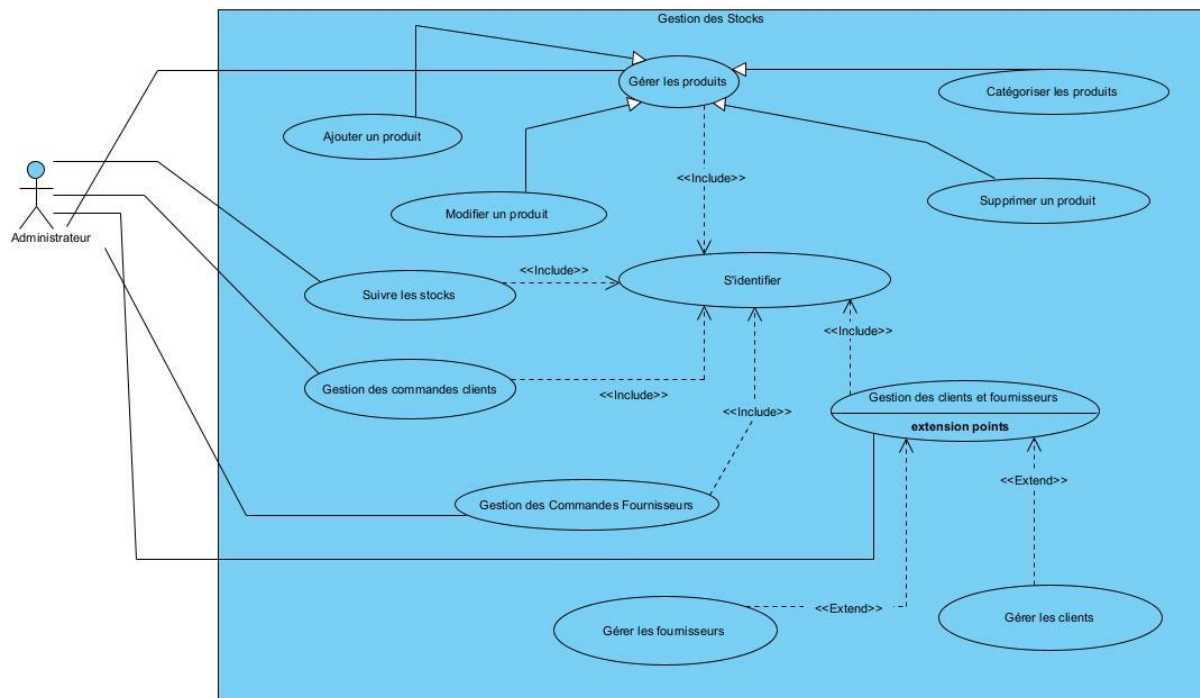


Figure 2 : diagramme de cas d'utilisation générale

Ce diagramme met en évidence les cas d'utilisation du système de gestion des stocks, en définissant clairement les actions possibles pour l'administrateur et les relations entre les fonctionnalités de gestion de produits, suivi des stocks et administration des ventes.

## 7. Diagramme de classe :

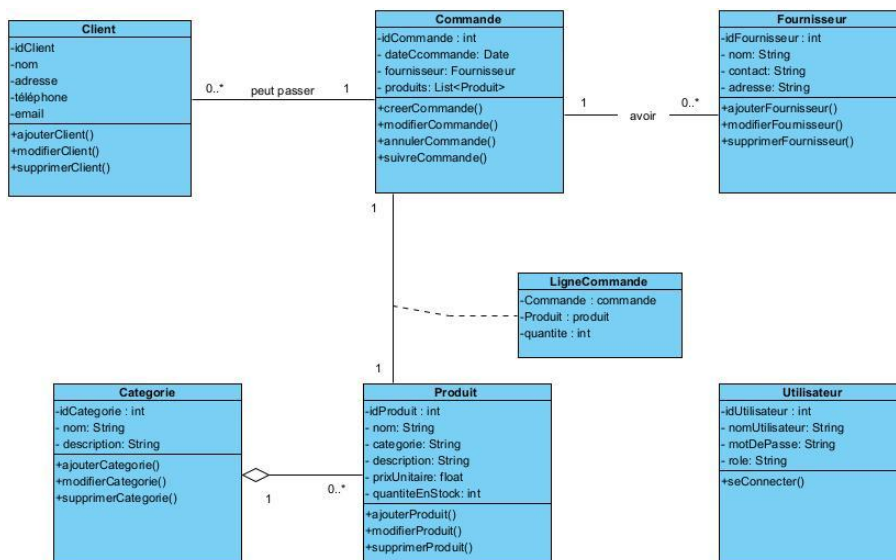


Figure 3 : diagramme de classe générale

Ce diagramme de classe définit clairement la structure des entités du système de gestion des stocks et les relations entre elles.

## 8. Modèle logique de données relationnel :

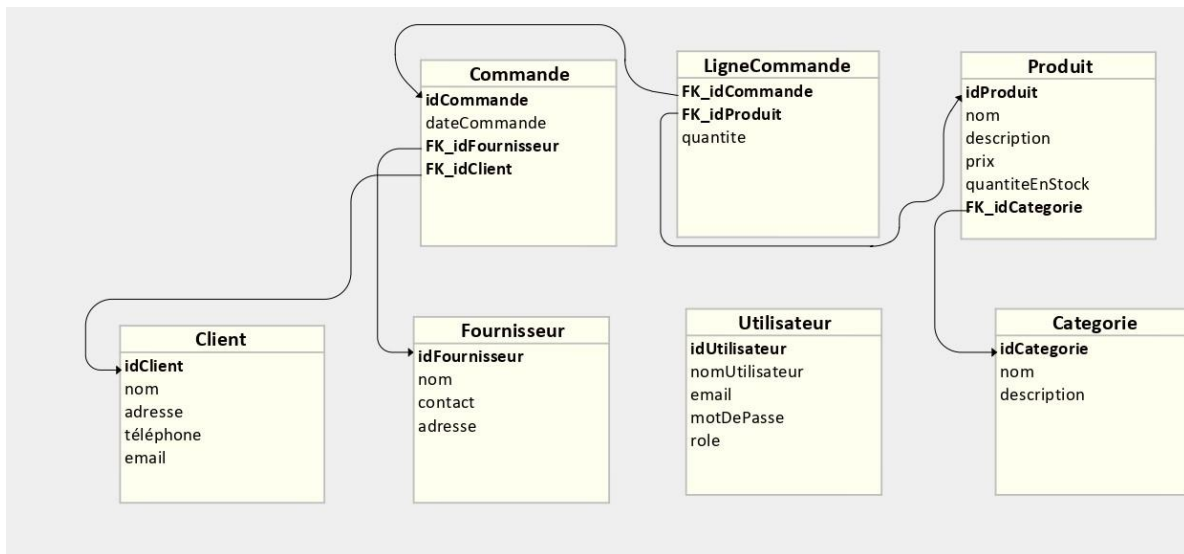


Figure 4 : Modèle logique de données relationnel

Le Modèle Logique des Données (MLDR) est une représentation normalisée et structurée des entités et des relations d'un système de gestion des stocks.

## Conception des cas d'utilisations :

### Administrateur :

#### 1 : Cas d'utilisation (S'identifier) :

Cas d'utilisation format textuelle :

#### Cas d'utilisation : Se connecter (Administrateur Gestion de Stock)

<b>Titre</b>	Connexion Administrateur - Gestion de Stock
<b>Objectif</b>	Permettre à l'administrateur de se connecter à l'interface de gestion de stock via une JFrame de login
<b>Acteur principal</b>	Administrateur
<b>Acteur secondaire</b>	Système
<b>Préconditions</b>	<ul style="list-style-type: none"> <li>- L'administrateur doit avoir un compte créé dans le système</li> <li>- Les identifiants de connexion doivent être connus de l'administrateur</li> </ul>

<b>Scénario nominal</b>	<ol style="list-style-type: none"> <li>1. L'administrateur lance l'application de gestion de stock</li> <li>2. La JFrame de login s'affiche</li> <li>3. L'administrateur saisit son identifiant (login)</li> <li>4. L'administrateur saisit son mot de passe</li> <li>5. L'administrateur clique sur le bouton de connexion</li> <li>6. Le système vérifie les identifiants</li> <li>7. Si les identifiants sont valides, la JFrame AdminDashboard s'ouvre</li> </ol>
<b>Scénario alternatif (Identifiants incorrects)</b>	<ol style="list-style-type: none"> <li>1. Le système vérifie les identifiants</li> <li>2. Si les identifiants sont incorrects : <ul style="list-style-type: none"> <li>- Un message d'erreur s'affiche</li> <li>- L'administrateur reste sur la JFrame de login</li> <li>- L'administrateur peut réessayer de se connecter</li> </ul> </li> </ol>
<b>Post-conditions</b>	<ul style="list-style-type: none"> <li>- Si connexion réussie : l'administrateur accède à l'interface AdminDashboard</li> <li>- Si connexion échouée : l'administrateur reste sur l'écran de login</li> </ul>

## **Diagramme de Séquence (S'identifier)**

### *Flow of Events*

1 : administrateur ouvrir l'application de Gestion de Stock

- Administrateur lance l'application

1.1: charger l'interface du login

- Système prépare l'interface de connexion

1.2: afficher l'interface du login

- Système affiche l'écran de connexion

2: saisir leur email et leur mot de passe

- Administrateur entre ses identifiants

2.1: vérifier les identifiants

- Système valide les informations de connexion

[alt - Identifiants valides] 2.2: charger l'interface du admin.

- Système prépare l'interface du admin

2.3: afficher l'interface du admin

Système affiche l'écran du admin.

[alt - Identifiants non valides] 2.4: afficher un message d'erreur

- Système affiche un message d'erreur de connexion



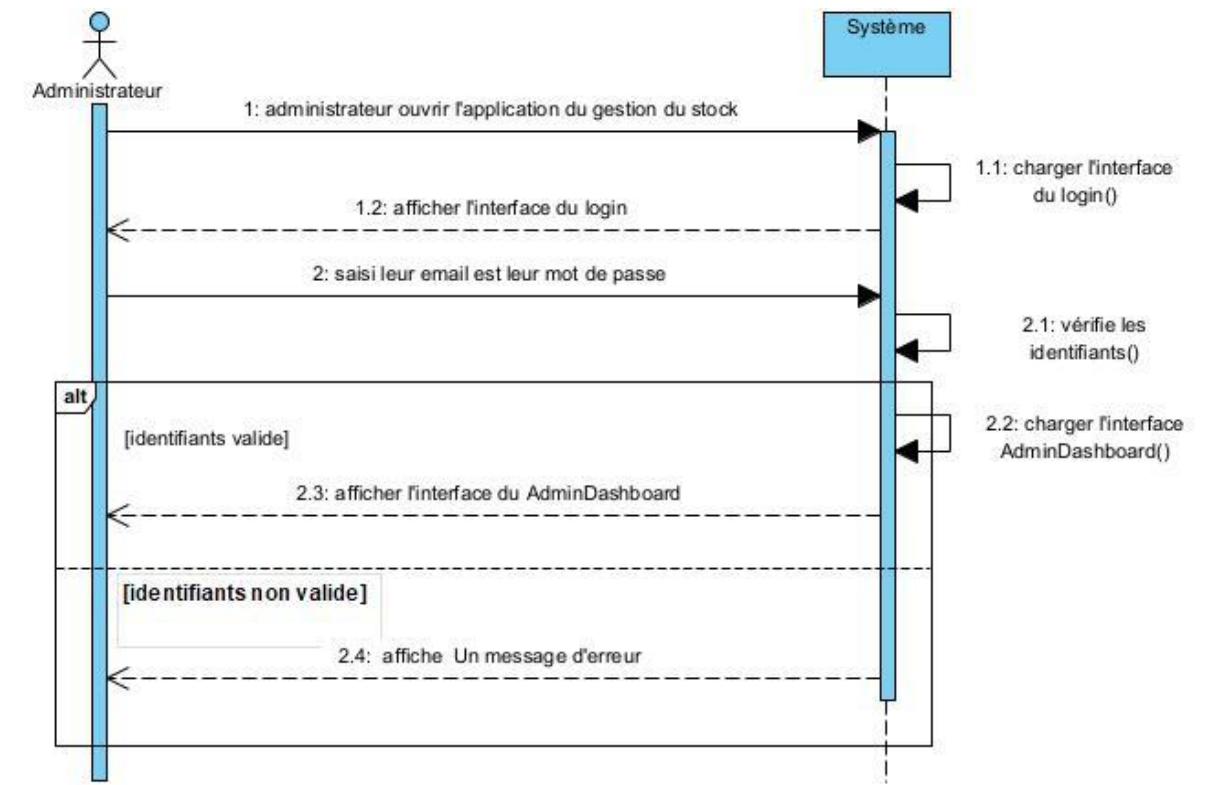


Figure 5 : SD : (S'identifier)

### Diagramme de classes participantes

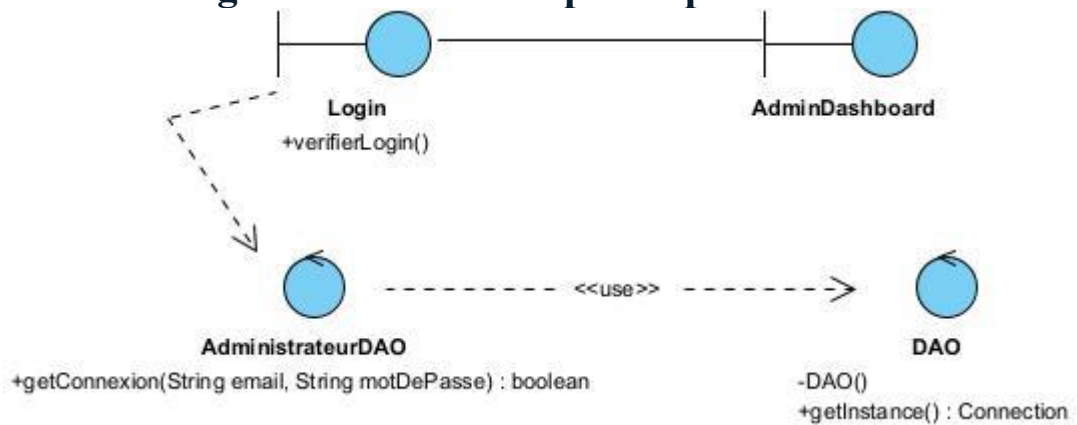


Figure 6 : DCP : (S'identifier)

## Diagramme d'activité

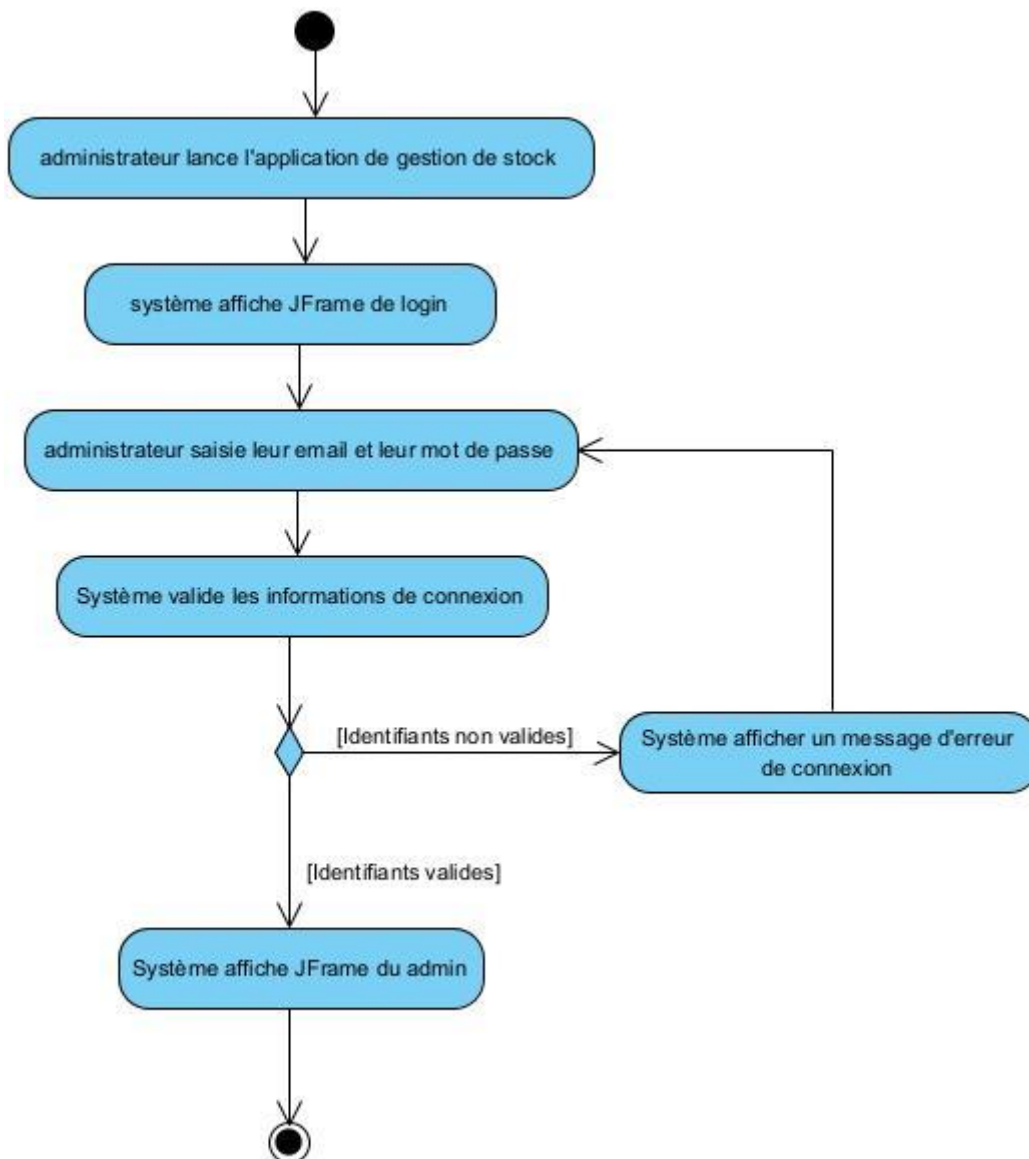


Figure 7 : diagramme d'activité : (S'identifier)

# Diagramme d'interaction

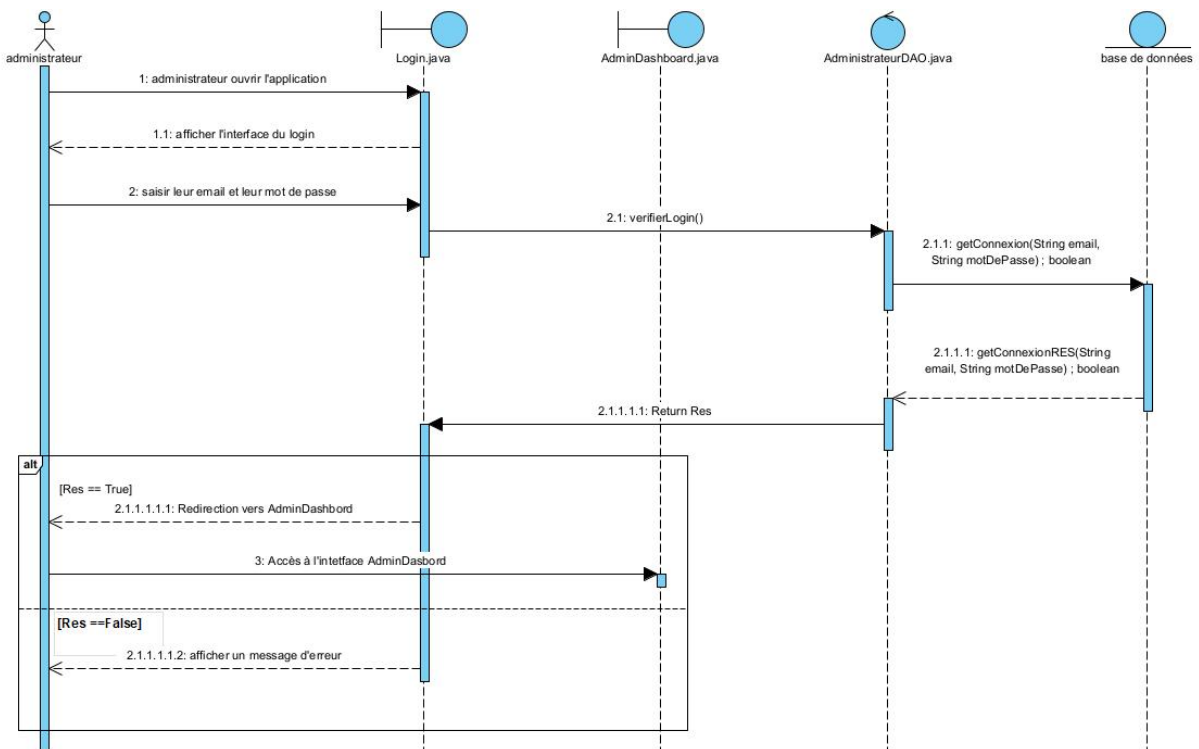


Figure 8 : diagramme d'interaction : (S'identifier)

### Réalisation :

L'interface de S'identifier :

Description :

Structure de l'Interface :

Champs de Saisie :

- Email : Champ obligatoire pour saisir l'adresse email enregistrée.
- Mot de passe : Champ sécurisé masqué par des astérisques pour la saisie du mot de passe.

Bouton d'Action :

- Login : Valide les identifiants et redirige vers AdminDashboard.

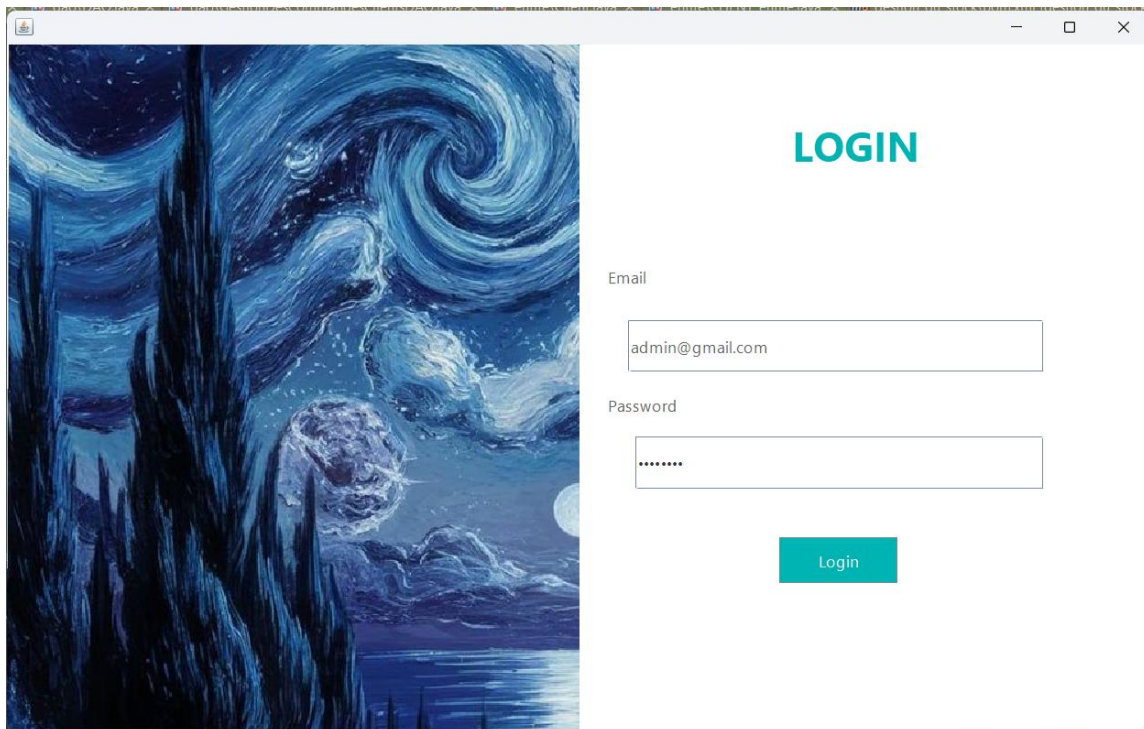


Figure 9 : image de l'interface de S'identifier

## 2 : Cas d'utilisation (Gestion des Commandes Fournisseurs)

Cas d'utilisation format textuelle :

<b>Titre</b>	Gestion des Commandes Fournisseurs
<b>Objectif</b>	Permettre à l'administrateur de gérer les commandes Fournisseurs dans le système de gestion commerciale
<b>Acteur principal</b>	Administrateur
<b>Acteur secondaire</b>	Système
<b>Préconditions</b>	- Administrateur connecté au système
<b>Scénario nominal</b>	<ol style="list-style-type: none"> <li>1. L'administrateur accède à l'interface de gestion des commandes Fournisseurs</li> <li>2. Le système affiche la liste des commandes existantes et le formulaire de saisie</li> <li>3. L'administrateur sélectionne un Fournisseurs dans la liste déroulante</li> <li>4. L'administrateur sélectionne un produit dans la liste déroulante</li> <li>5. Le système affiche automatiquement le prix unitaire du produit sélectionné</li> <li>6. L'administrateur saisit la quantité souhaitée</li> <li>7. L'administrateur valide l'enregistrement de la commande</li> </ol>
<b>Scénarios alternatifs</b>	- Aucune commande dans la base de données
<b>Actions possibles</b>	<ul style="list-style-type: none"> <li>- Ajouter une nouvelle commande (Fournisseurs, Produit, Quantité, Prix)</li> <li>- Consulter la liste des commandes</li> </ul>
<b>Post-conditions</b>	<ul style="list-style-type: none"> <li>- La base de données des (commandes &amp;&amp; ligne commande) est mise à jour</li> <li>- Les listes affichées sont actualisées automatiquement</li> </ul>

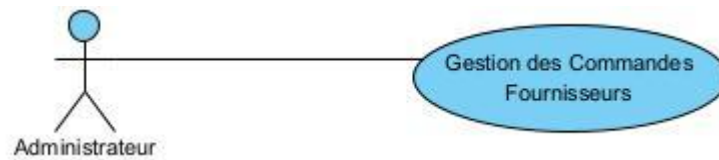


Figure 10 : Diagramme de cas d'utilisation (Gestion des Commandes Fournisseurs)

## Diagramme de Séquence (Gestion des Commandes Fournisseurs)

### Flow of Events - Cas d'utilisation "Gestion des Commandes Fournisseurs "

#### Flux principal

##### Identification et initialisation

1. L'administrateur accède à l'interface de gestion des commandes Fournisseurs.
2. Le système charge la liste des commandes existantes.
3. Le système affiche la liste des commandes existantes à l'administrateur.

##### Gestion des Commandes

1. L'administrateur remplit le formulaire avec les informations de la commande du Fournisseur.
2. L'administrateur clique sur le bouton "Enregistrer".
3. Le système vérifie la validité des données saisies.
4. Si les données sont valides, le système ajoute une nouvelle commande avec les informations (Fournisseurs, Produit, Quantité, Prix).
5. Le système charge à nouveau la liste des commandes existantes.
6. Le système affiche la liste mise à jour des commandes existantes à l'administrateur.

#### Flux alternatifs

##### Données invalides lors de l'ajout

1. Le système détecte des données invalides lors de la validation.
2. Le système affiche un message d'erreur à l'administrateur.
3. L'administrateur doit corriger les données avant de soumettre à nouveau.

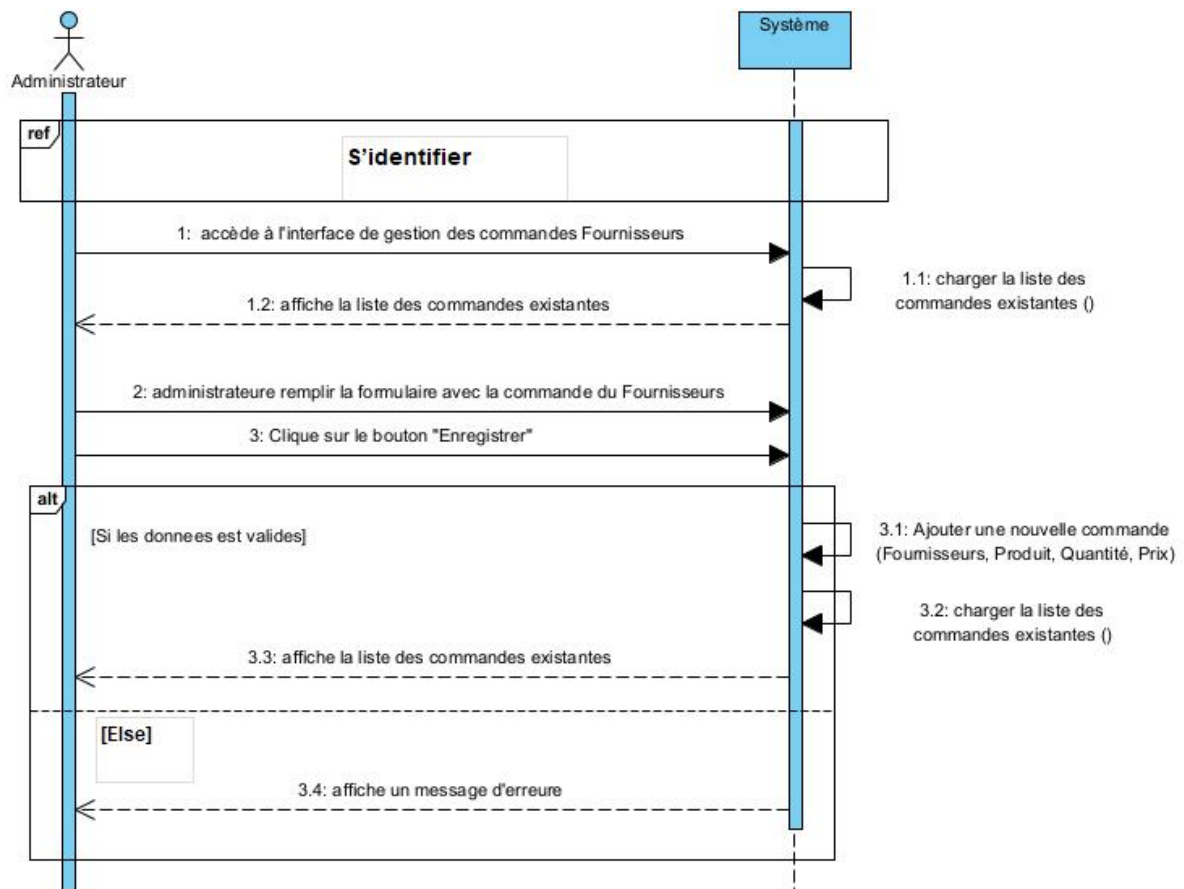


Figure 11 : SD : (Gestion des Commandes Fournisseur)

## Diagramme d'activité



Figure 12 : diagramme d'activité : (Gestion des Commandes Fournisseur)



## Diagramme de classes participantes

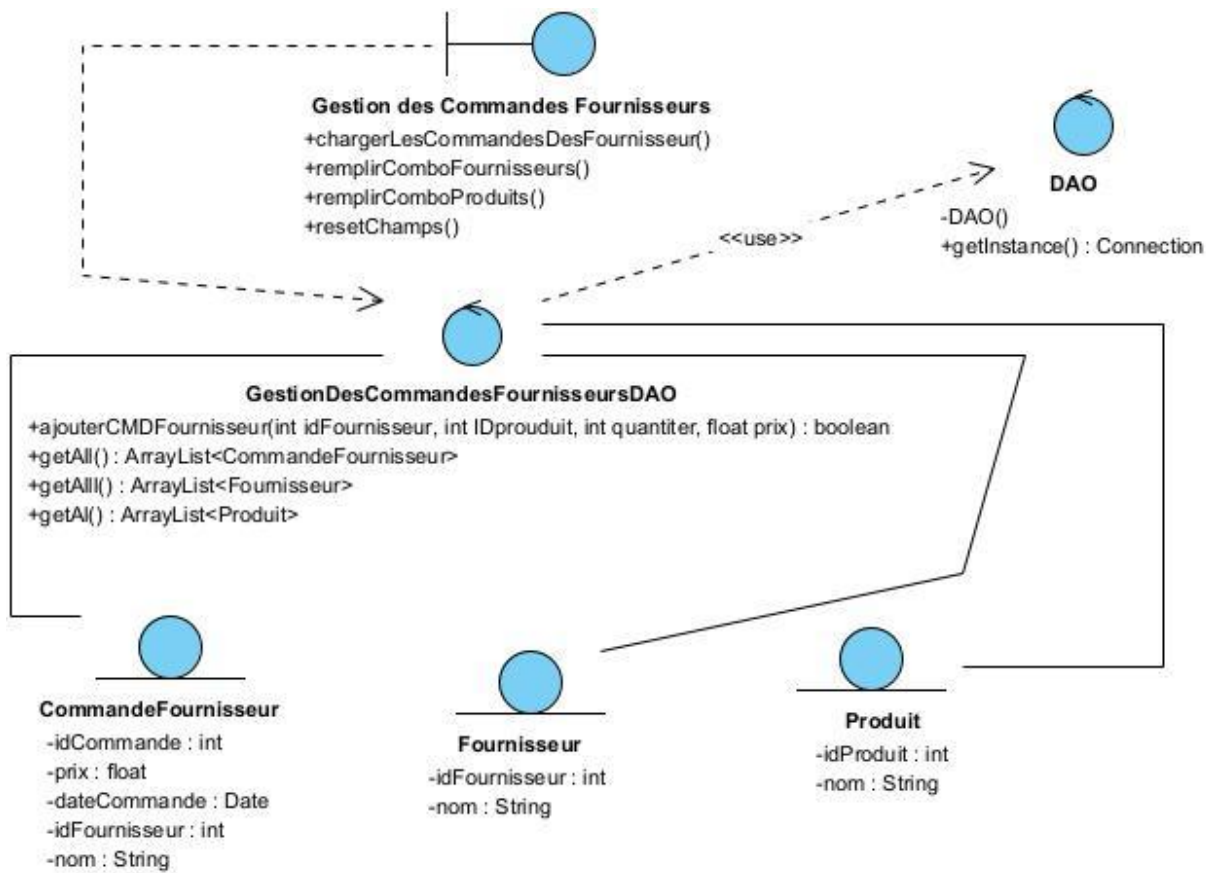


Figure 13 : DCP : (Gestion des Commandes Fournisseur)

# Diagramme d'interaction

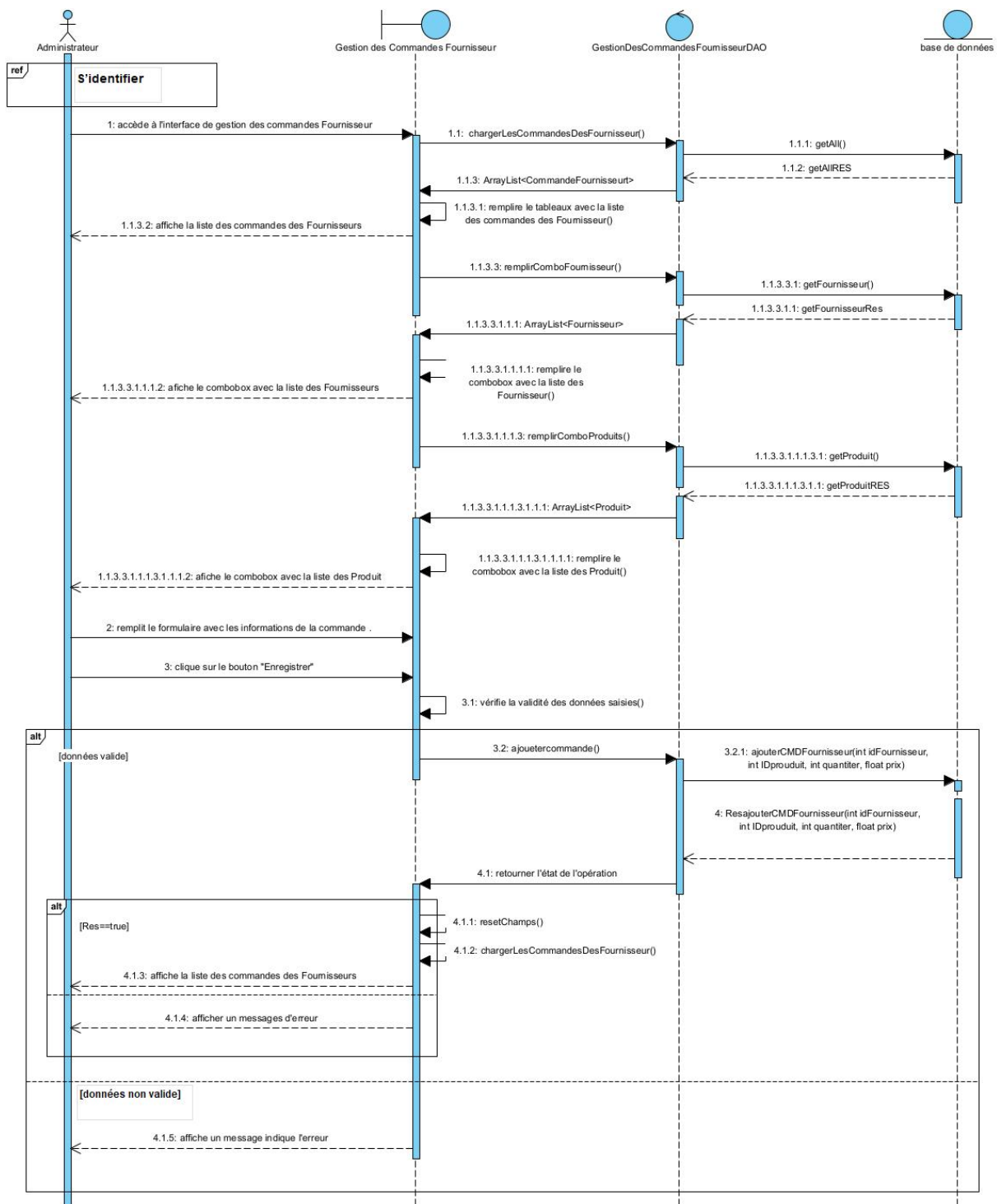


Figure 14 : diagramme d'interaction : (Gestion des Commandes Fournisseur)

Réalisation :

L'interface de Gestion des Commandes Fournisseur :

**Liste des commandes**

Nim de commande	Fournisseur	Date	Montant total
9	messi	2025-03-28	5000.0 DH
10	yassine	2025-03-28	800.0 DH
11	Rachide	2025-03-28	2400.0 DH

**Gestion des Commandes Fournisseurs**

**Liste du fournisseur** : yassine

**Liste des produits** : Rail de guidage 2m

**Quantité** : 22

**Prix** : 120.00

**Enregistrer Commande**

Figure 15 : image de l'interface de Gestion des Commandes Fournisseur

### 3 : Cas d'utilisation (Gérer les produits) :

Cas d'utilisation format textuelle :

#### Cas d'utilisation : Gérer les produits (Administrateur Gestion de Stock)

<b>Titre</b>	<b>Gérer les Produits</b>
<b>Objectif</b>	Permettre à l'administrateur de gérer complètement le catalogue de produits dans le système de gestion de stocks
<b>Acteur principal</b>	Administrateur
<b>Acteur secondaire</b>	Système
<b>Préconditions</b>	- Être connecté au système
<b>Scénario nominal</b>	1. L'administrateur accède à l'interface de gestion des produits 2. Le système affiche la liste des produits existants 3. L'administrateur sélectionne l'action à réaliser (ajouter, modifier, supprimer)
<b>Scénarios alternatifs</b>	- Aucun produit dans le catalogue - Échec de chargement de la liste des produits
<b>Actions possibles</b>	- Ajouter un nouveau produit - Modifier un produit existant - Supprimer un produit
<b>Post-conditions</b>	- Le catalogue de produits est mis à jour

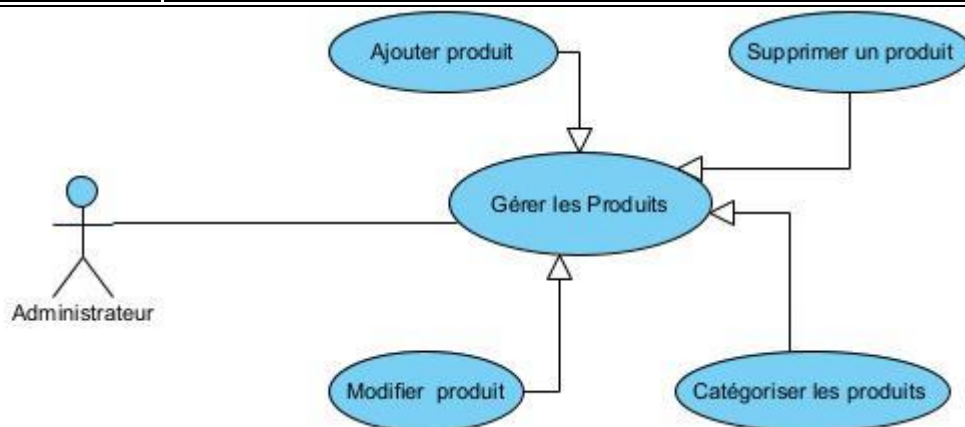


Figure 16 : Diagramme de cas d'utilisation (Gérer les produits)

## **Diagramme de Séquence (Gérer les produits)**

### **Flow of Events - Gestion des Produits**

#### **Acteurs**

- Administrateur
- Système

#### **Flux d'Événements Détaillé**

##### **Étape 1 : Accès à l'Interface de Gestion des Produits**

1.1 L'administrateur accède à l'interface de gestion des produits

1.2 Le système réalise les actions suivantes :

- Charger l'interface de gestion des produits
- Afficher la liste des produits existants

##### **Étape 2 : Sélection de l'Action**

2.1 L'administrateur sélectionne l'action à réaliser

2.2 Le système effectue les opérations correspondantes :

- Mettre à jour le catalogue de produits
- Réafficher la liste des produits mise à jour

#### **Préconditions**

- L'administrateur doit être préalablement identifié (cas d'utilisation "S'identifier")

#### **Postconditions**

- Le catalogue de produits est mis à jour
- La liste des produits est réaffichée avec les modifications

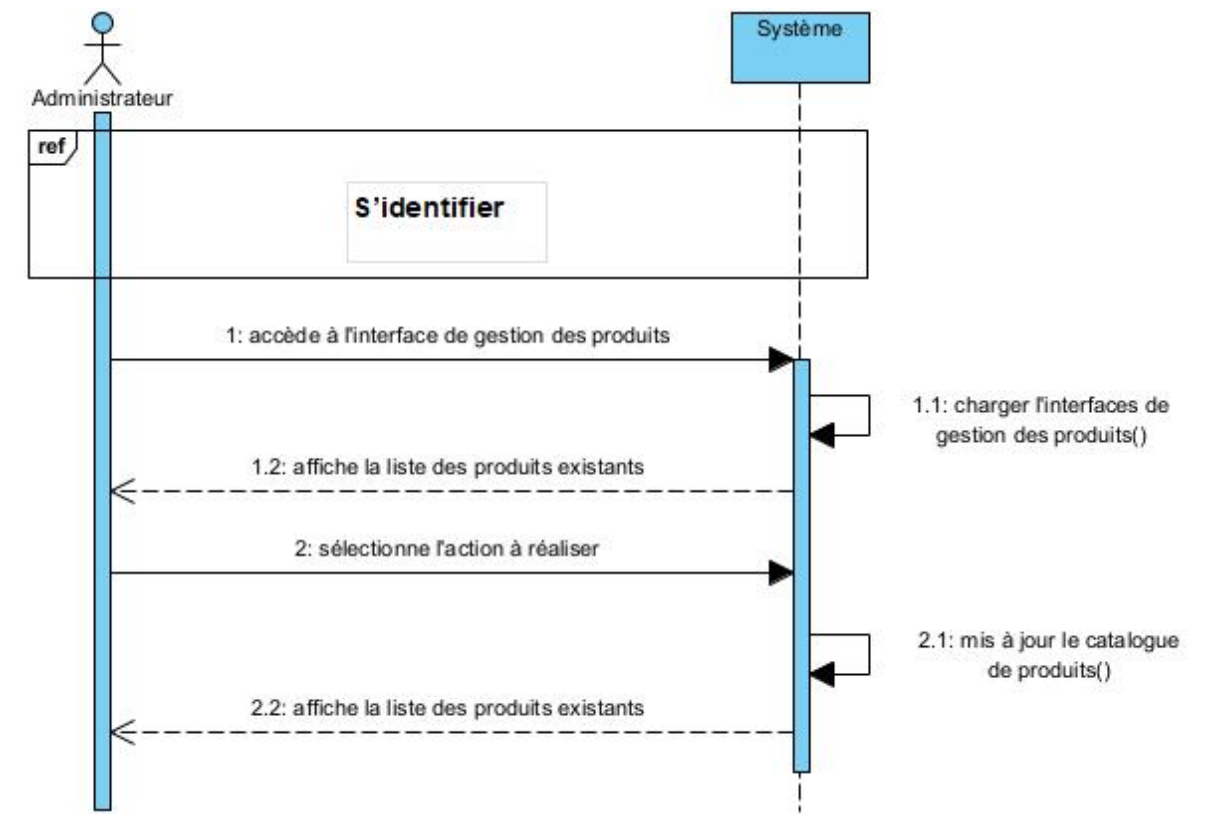


Figure 17 : SD : (Gérer les produits)

## Diagramme de classes participantes

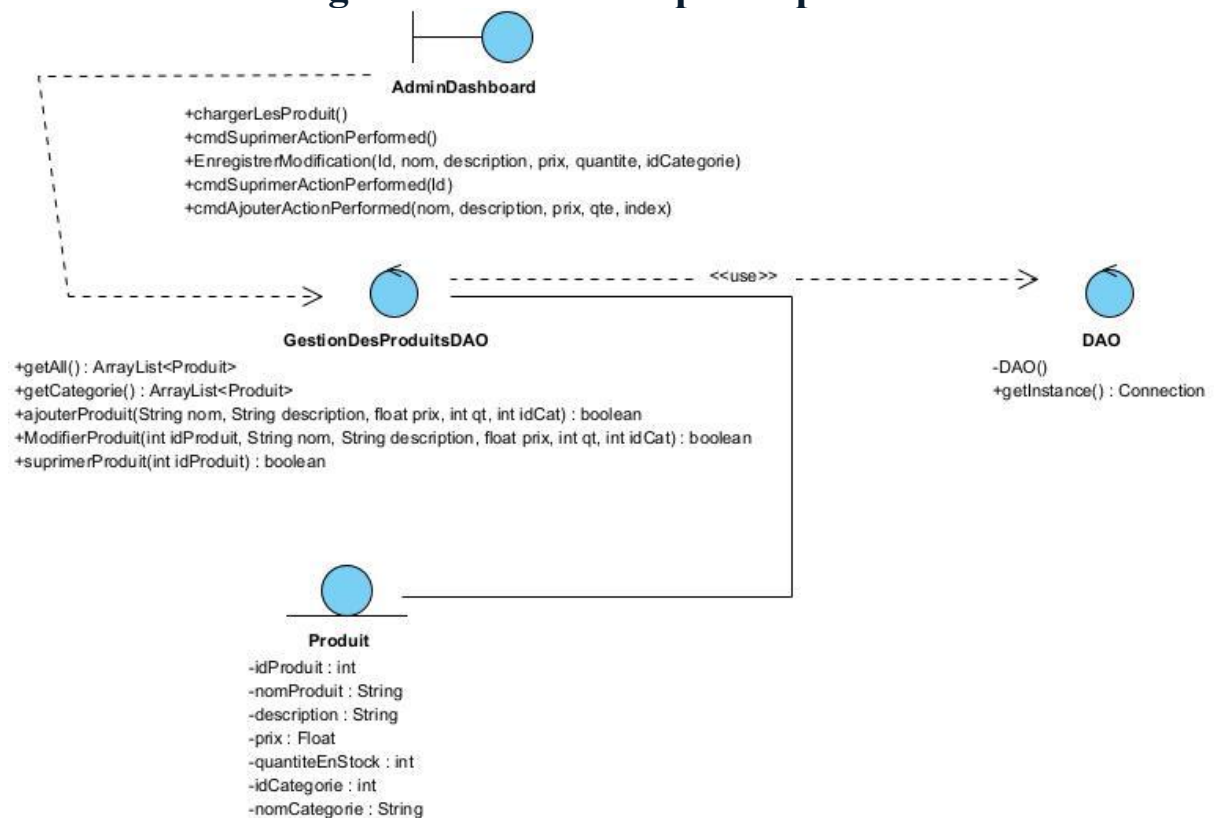


Figure 18 : DCP : (Gérer les produits)

## Diagramme d'activité

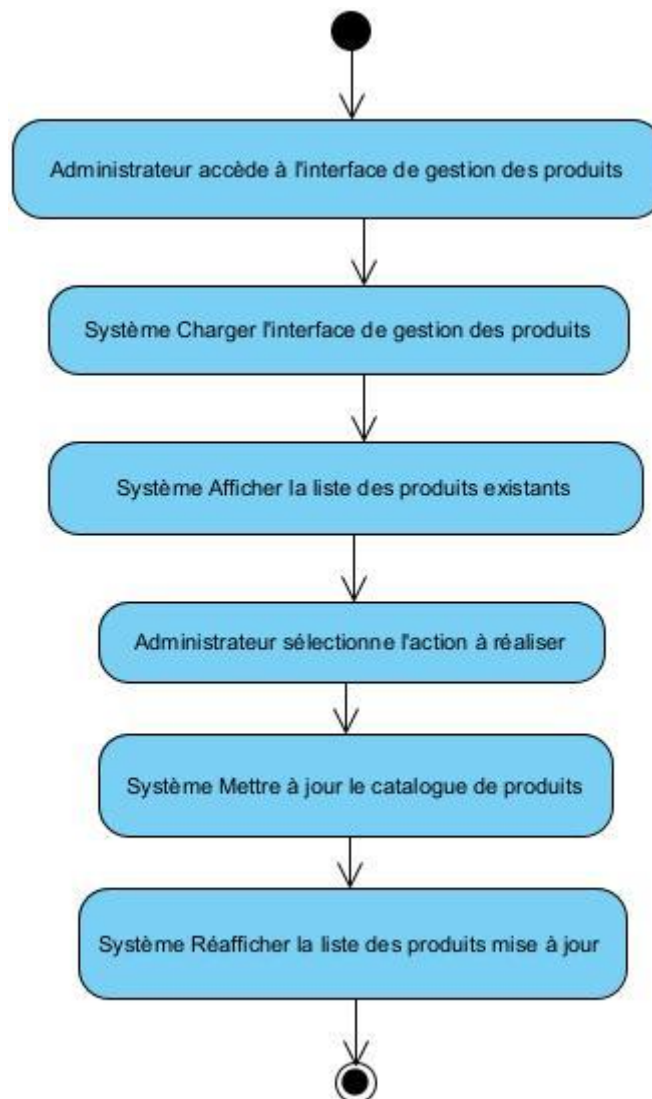


Figure 19 : diagramme d'activité : (Gérer les produits)

# Diagramme d'interaction

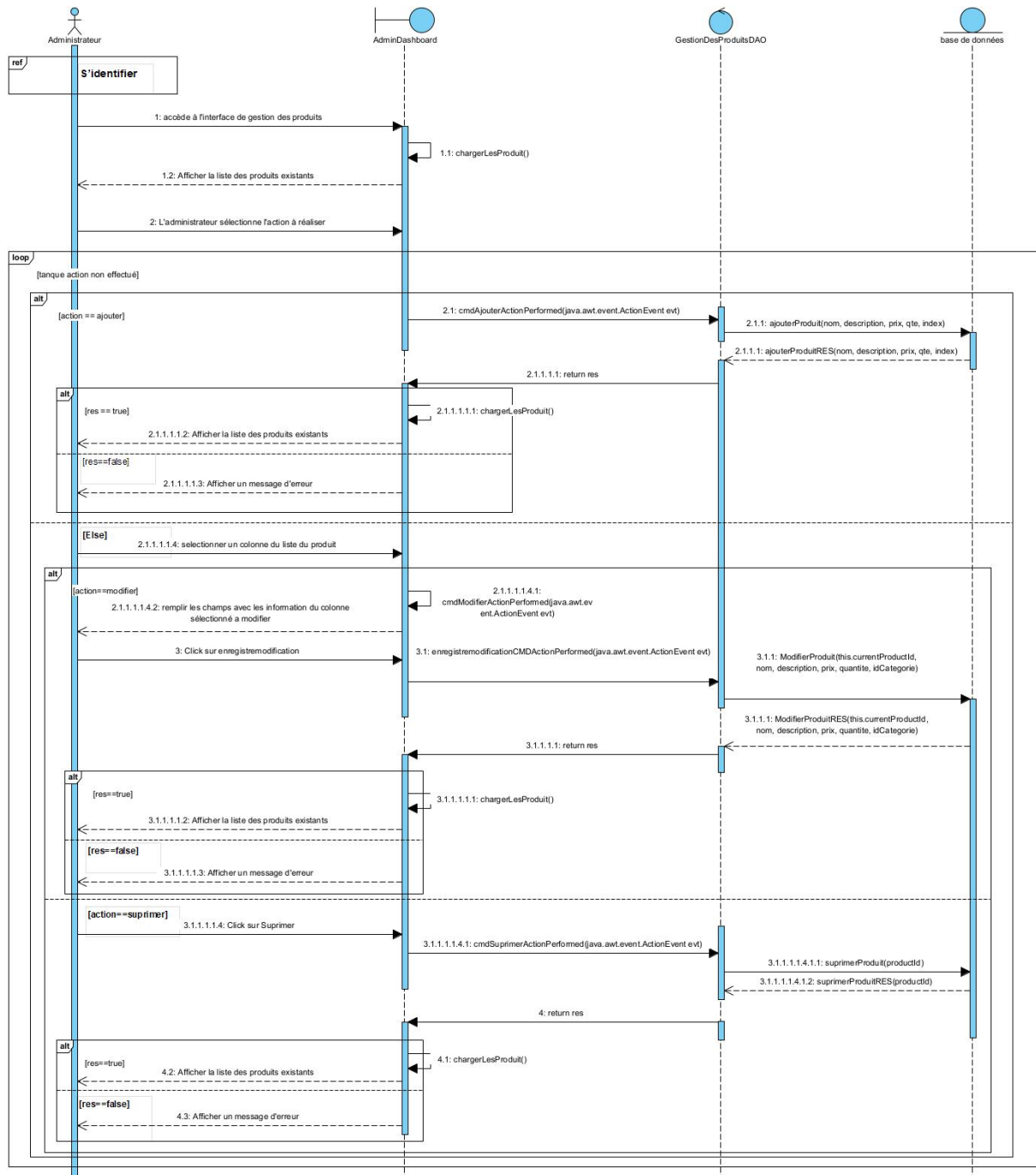


Figure 20 : diagramme d'interaction : (Gérer les produits)



Réalisation :

L'interface de Gérer les produits :

Description :

Structure de l'Interface :

**1. Section de gestion des produits (gauche) :**

- Contient des champs pour entrer les informations d'un produit (Nom, Description, Quantité en stock, Prix, Catégorie).
- Propose des boutons d'action : "Ajouter", "Modifier", "Supprimer" et "Enregistrer Modification".

**2. Section d'affichage des produits (droite) :**

- Une table pour afficher la liste des produits avec les colonnes : Nom, Description, Prix, Quantité en stock, Catégorie, et ID.

The screenshot displays a web application interface for managing products. The left sidebar, titled 'Gestion des Produits', contains a form with the following fields: 'Nom du Produit', 'description', 'quantite En Stock', 'Prix', and 'Categorie' (a dropdown menu currently showing 'Moteurs'). Below these fields are three buttons: 'AJOUTER', 'MODIFIER', and 'SUPPRIMER'. At the bottom of the sidebar is a search bar labeled 'search by category'. The main content area, titled 'La liste du produit', features a table with the following data:

Nom	Description	Prix	quantiteEnStock	Categorie	Id
Moteur 10CV	Moteur électrique 10CV pour ascenseur	2500.0	5	Moteurs	2
Câble en acier 8mm	Câble de traction en acier de 8mm	200.0	50	Câbles	3
Câble en acier 10mm	Câble de traction en acier de 10mm	300.0	40	Câbles	4
Bouton d'appel standard	Bouton d'appel avec éclairage LED	50.0	100	Boutons et Panneaux	5
Panneau de commande dig	Panneau tactile pour ascenseur	500.0	15	Boutons et Panneaux	6
Rail de guidage 2m	Rail en acier de 2 mètres	120.0	30	Rails de Guidage	7
Rail de guidage 3m	Rail en acier de 3 mètres	180.0	20	Rails de Guidage	8
Porte automatique	Porte coulissante automatique	800.0	8	Portes	9
Porte manuelle	Porte battante manuelle	400.0	12	Portes	10

Figure 21 : image de l'interface Gérer les produits.

#### 4 : Cas d'utilisation (Gestion des clients et fournisseurs)

Cas d'utilisation format textuelle :

##### Cas d'utilisation : Gestion des clients et fournisseurs

Titre	Gestion des clients et fournisseurs
Objectif	Permettre à l'administrateur de gérer les données des clients et des fournisseurs dans le système de gestion des stocks
Acteur principal	Administrateur
Acteur secondaire	Système
Préconditions	- Être connecté au système
Scénario nominal	1. L'administrateur accède à l'interface de gestion des clients et fournisseurs 2. Le système affiche les formulaires de saisie et les listes existantes 3. L'administrateur sélectionne l'action à réaliser (ajouter, supprimer un client ou un fournisseur)
Scénarios alternatifs	- Aucun client ou fournisseur dans la base de données - Échec de chargement des listes clients/fournisseurs
Actions possibles	- Ajouter un nouveau client (Nom, Adresse, Téléphone, Email - Supprimer un client existant - Ajouter un nouveau fournisseur (Nom, Contact, Adresse- - Supprimer un fournisseur existant- - Consulter la liste des clients- - Consulter la liste des fournisseurs
Post-conditions	- La base de données clients/fournisseurs est mise à jour - Les listes affichées sont actualisées automatiquement

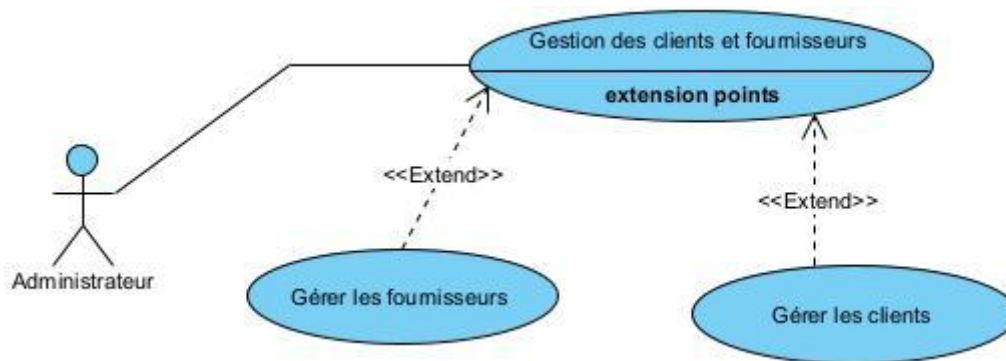


Figure 22 : Diagramme de cas d'utilisation (Gestion des clients et fournisseurs)

### Diagramme de Séquence (Gestion des clients et fournisseurs)

#### **Flow of Events - Cas d'utilisation "Gestion des clients et fournisseurs"**

##### **Flux principal**

##### **Identification et initialisation**

1. L'administrateur s'identifie auprès du système.
2. L'administrateur accède à l'interface de gestion des clients et fournisseurs.
3. Le système charge la liste des clients et la liste des fournisseurs depuis la base de données.
4. Le système affiche les formulaires de saisie et les listes existantes.

##### **Gestion des Clients**

##### **Sous-flux: Ajouter un client**

1. L'administrateur saisit les informations du client dans les champs appropriés.
2. L'administrateur clique sur le bouton "Enregistrer".
3. Le système procède à l'insertion des données client dans la base de données.
4. Le système met à jour l'interface avec le nouveau client ajouté.

##### **Sous-flux: Supprimer un client**

1. L'administrateur sélectionne un client dans la liste.
2. L'administrateur clique sur le bouton "Supprimer".
3. Le système procède à la suppression du client de la base de données.
4. Le système met à jour la liste des clients affichée.

## **Gestion des Fournisseurs**

### **Sous-flux: Ajouter un fournisseur**

1. L'administrateur saisit les informations du fournisseur dans les champs appropriés.
2. L'administrateur clique sur le bouton "Enregistrer".
3. Le système procède à l'insertion des données fournisseur dans la base de données.
4. Le système met à jour l'interface avec le nouveau fournisseur ajouté.

### **Sous-flux: Supprimer un fournisseur**

1. L'administrateur sélectionne un fournisseur dans la liste.
2. L'administrateur clique sur le bouton "Supprimer".
3. Le système procède à la suppression du fournisseur de la base de données.
4. Le système met à jour la liste des fournisseurs affichée.

### **Flux alternatifs**

#### **Données invalides lors de l'ajout**

1. Le système détecte des données invalides lors de la saisie.
2. Le système affiche un message d'erreur.
3. L'administrateur doit corriger les données avant de soumettre à nouveau.

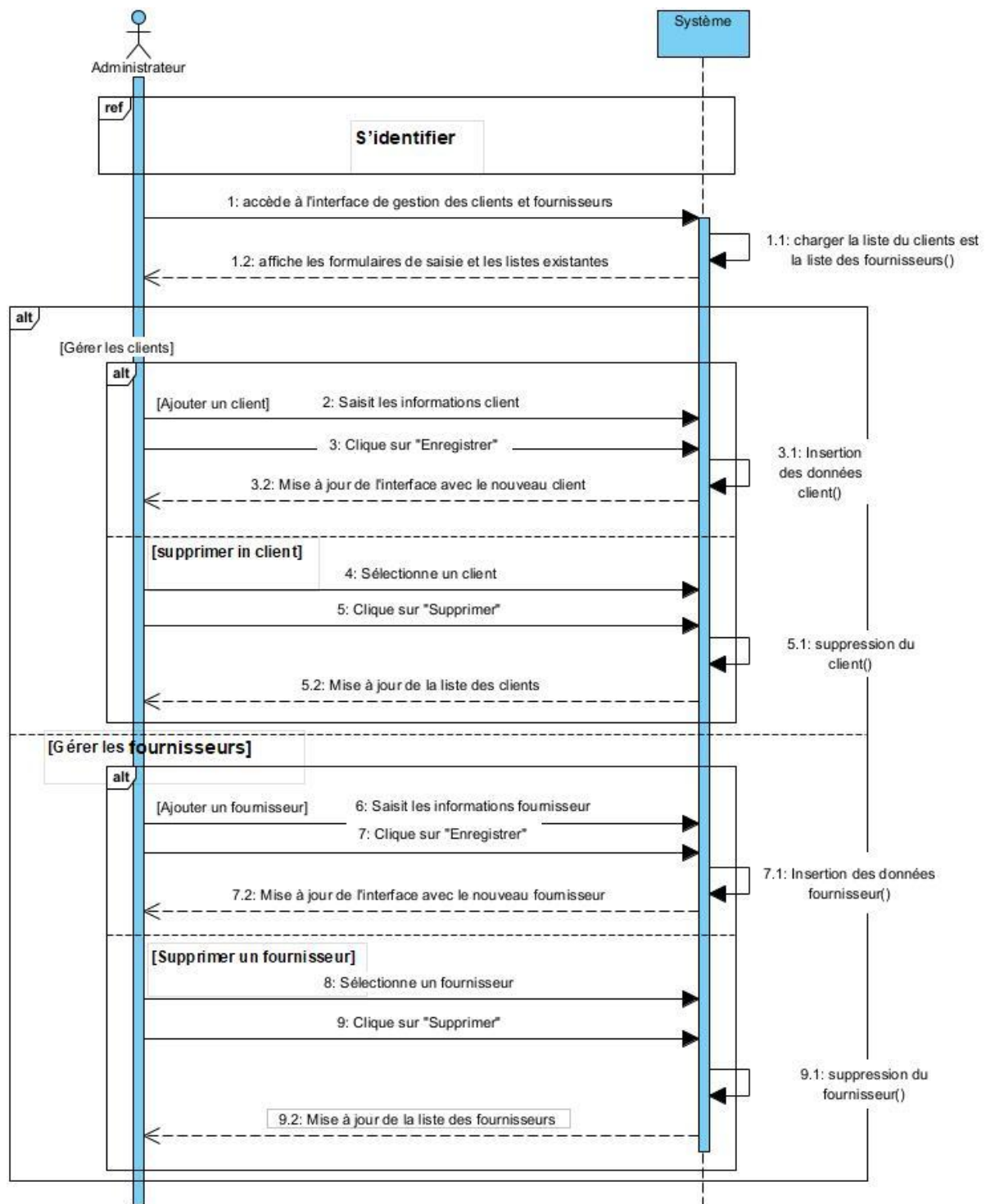


Figure 23 : SD : (Gestion des clients et fournisseurs)

## Diagramme de classes participantes

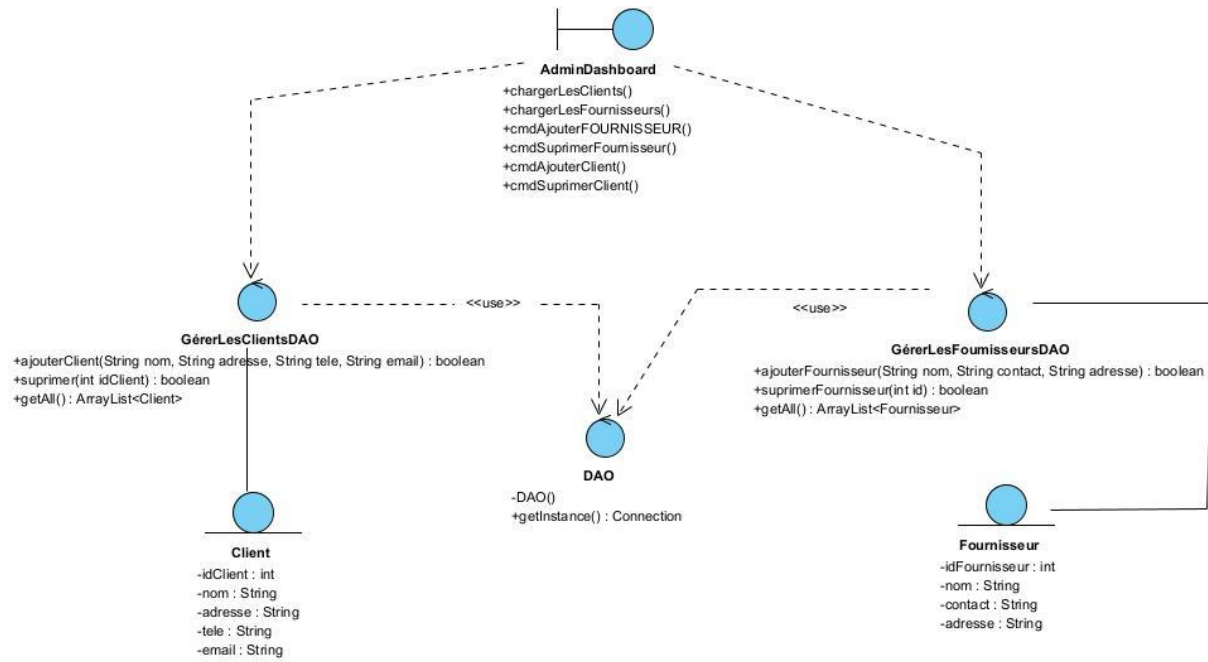


Figure 24 : DCP : (Gestion des clients et fournisseurs)

## Diagramme d'activité

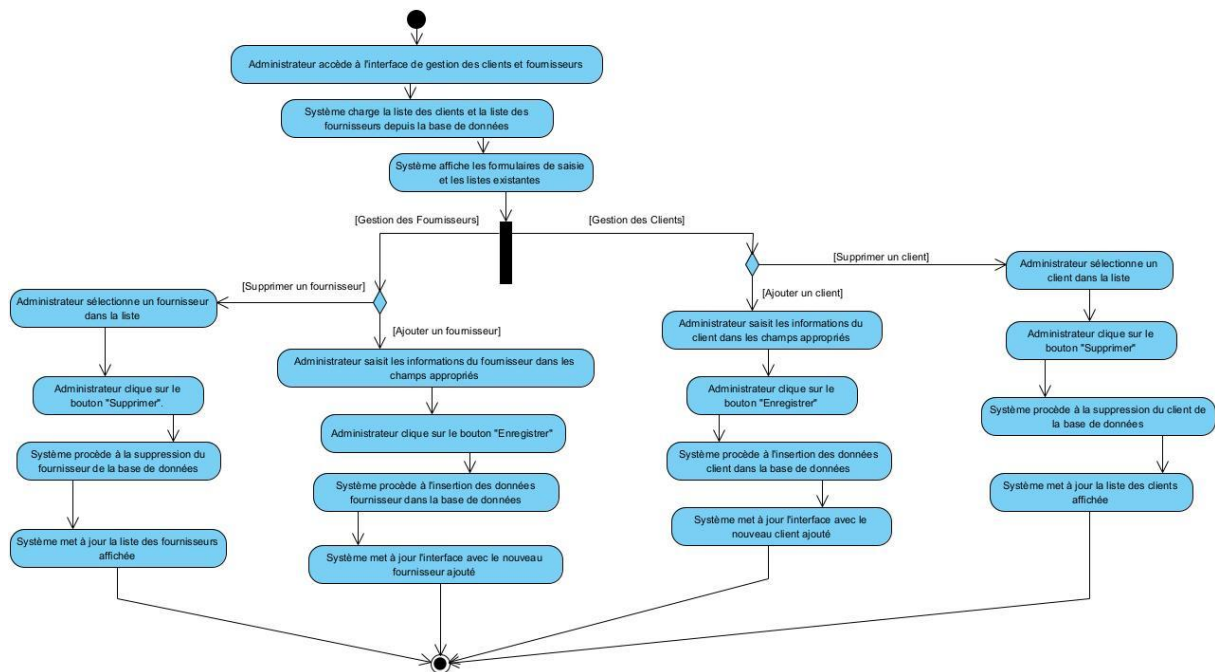
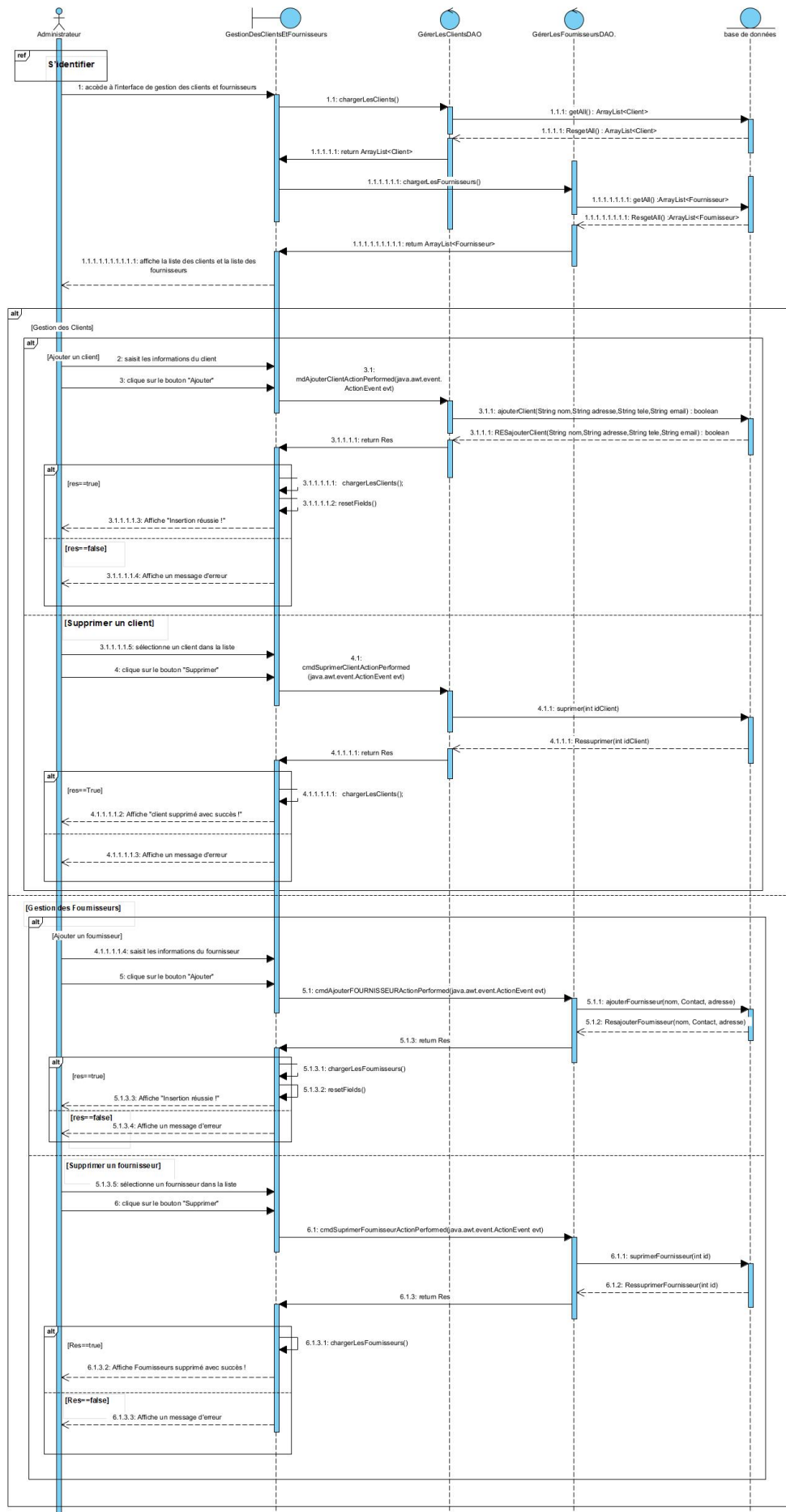


Figure 25 : diagramme d'activité : (Gestion des clients et fournisseurs)

## Diagramme d'interaction





Réalisation :

L'interface de Gestion des clients et fournisseurs :

**Gestion des Client**

Nom

Adresse

Téléphone

Email

Enregistrer Supprimer

**Gestion des Fournisseurs**

Nom

Contact

Adresse

Enregistrer Supprimer

**La liste des Clients**

id	Nom	Adresse	Téléphone	Email
5	anas	kenitra	06292962	anas@gmail.com
6	simo	rabat	0633875912	moad@gmail.com
7	hichame	kenitra	06292962	anas@gmail.com
11	zakaria	el harhoni	0623255896	zakaria@gmail.com

**La liste des Fournisseurs**

id	Nom	Contact	Adresse
4	messi	0625251212	miami
6	yassine	0629292929	kenitra
7	Rachide	0425282930	Tanger

Figure 27 : image de l'interface de Gestion des clients et fournisseurs

## 5 : Cas d'utilisation (Gestion des Commandes Clients)

Cas d'utilisation format textuelle :

<b>Titre</b>	Gestion des Commandes Clients
<b>Objectif</b>	Permettre à l'administrateur de gérer les commandes clients dans le système de gestion commerciale
<b>Acteur principal</b>	Administrateur
<b>Acteur secondaire</b>	Système
<b>Préconditions</b>	- Être connecté au système
<b>Scénario nominal</b>	<ol style="list-style-type: none"><li>1. L'administrateur accède à l'interface de gestion des commandes clients</li><li>2. Le système affiche la liste des commandes existantes et le formulaire de saisie</li><li>3. L'administrateur sélectionne un client dans la liste déroulante</li><li>4. L'administrateur sélectionne un produit dans la liste déroulante</li><li>5. Le système affiche automatiquement le prix unitaire du produit sélectionné</li><li>6. L'administrateur saisit la quantité souhaitée</li><li>7. L'administrateur valide l'enregistrement de la commande</li></ol>
<b>Scénarios alternatifs</b>	- Aucune commande dans la base de données
<b>Actions possibles</b>	<ul style="list-style-type: none"><li>- Ajouter une nouvelle commande (Client, Produit, Quantité, Prix)</li><li>- Consulter la liste des commandes</li></ul>
<b>Post-conditions</b>	<ul style="list-style-type: none"><li>- La base de données des commandes &amp;&amp; ligne commande est mise à jour</li><li>- Les listes affichées sont actualisées automatiquement</li></ul>



Figure 28: Diagramme de cas d'utilisation (Gestion des Commandes Clients)

## Diagramme de Séquence (Gestion des Commandes Clients)

### Flow of Events - Cas d'utilisation "Gestion des Commandes Clients"

#### Flux principal

##### Identification et initialisation

4. L'administrateur accède à l'interface de gestion des commandes clients.
5. Le système charge la liste des commandes existantes.
6. Le système affiche la liste des commandes existantes à l'administrateur.

##### Gestion des Commandes

7. L'administrateur remplit le formulaire avec les informations de la commande du client.
8. L'administrateur clique sur le bouton "Enregistrer".
9. Le système vérifie la validité des données saisies.
10. Si les données sont valides, le système ajoute une nouvelle commande avec les informations (Client, Produit, Quantité, Prix).
11. Le système charge à nouveau la liste des commandes existantes.
12. Le système affiche la liste mise à jour des commandes existantes à l'administrateur.

#### Flux alternatifs

##### Données invalides lors de l'ajout

4. Le système détecte des données invalides lors de la validation.
5. Le système affiche un message d'erreur à l'administrateur.
6. L'administrateur doit corriger les données avant de soumettre à nouveau.

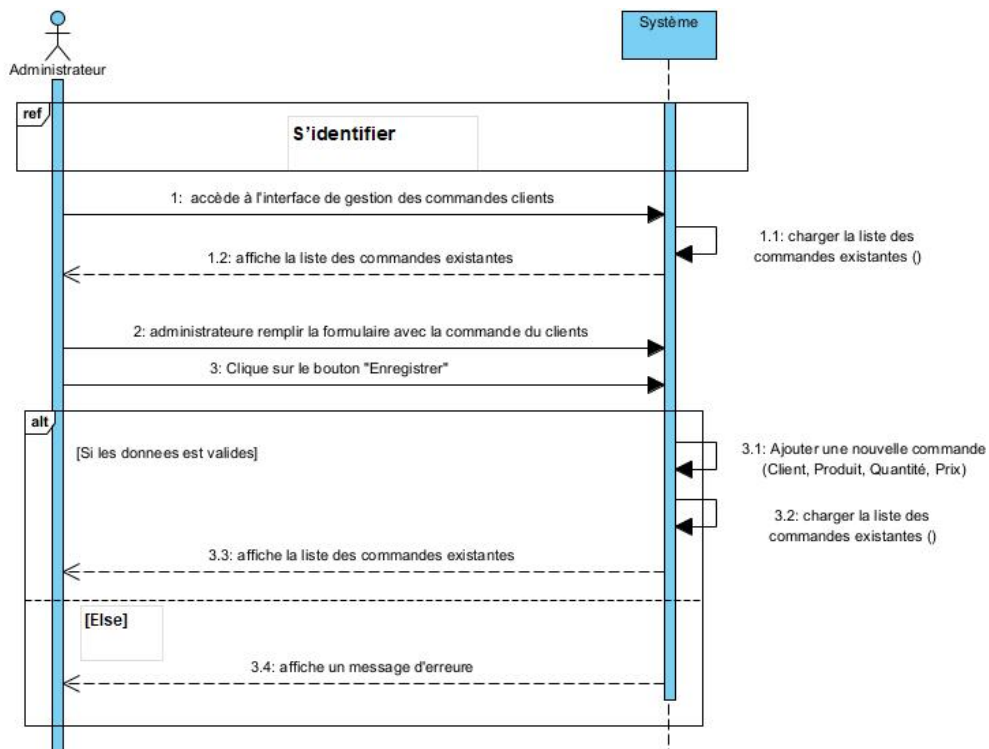


Figure 29 : SD : (Gestion des Commandes Clients)

## Diagramme d'activité



Figure 30 : diagramme d'activité : (Gestion des Commandes Clients)

## Diagramme de classes participantes

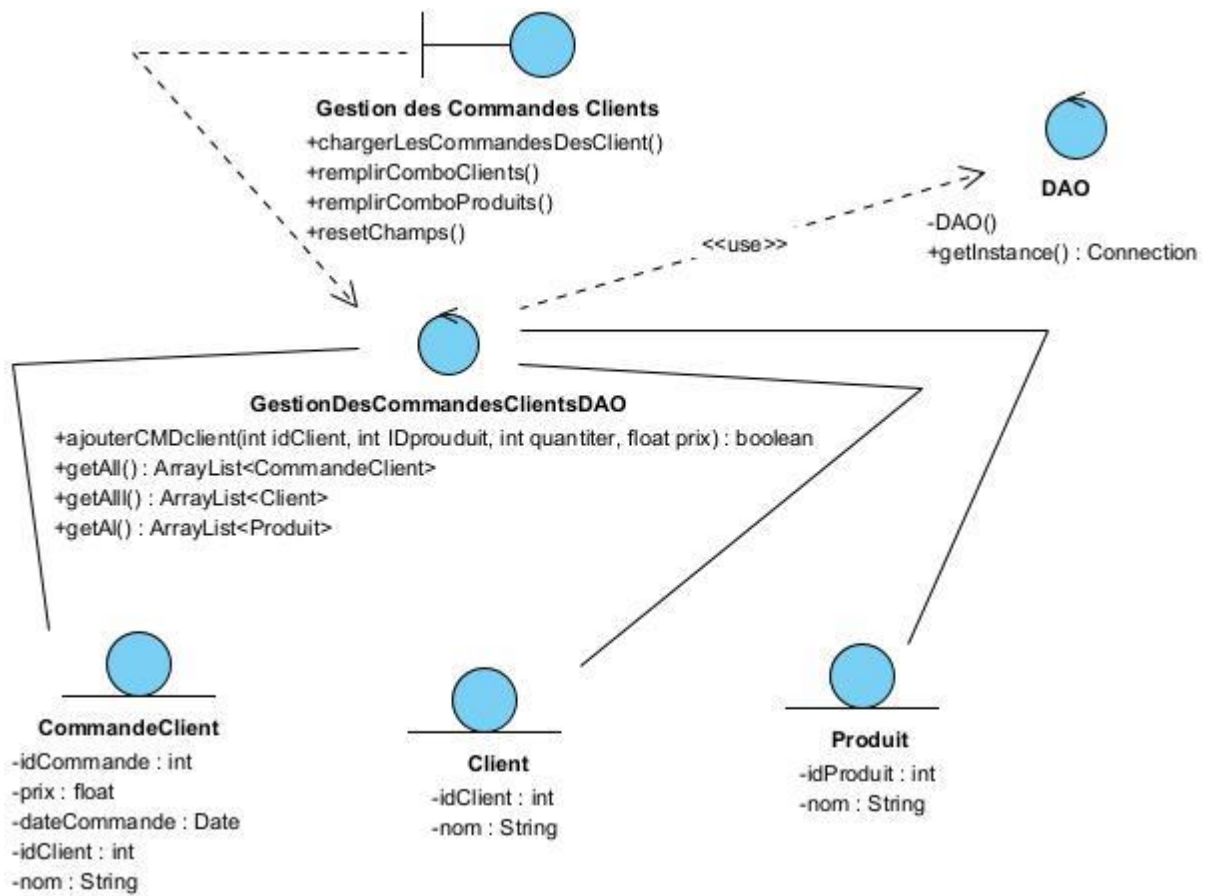


Figure 31 : DCP : (Gestion des Commandes Clients)

# Diagramme d'interaction

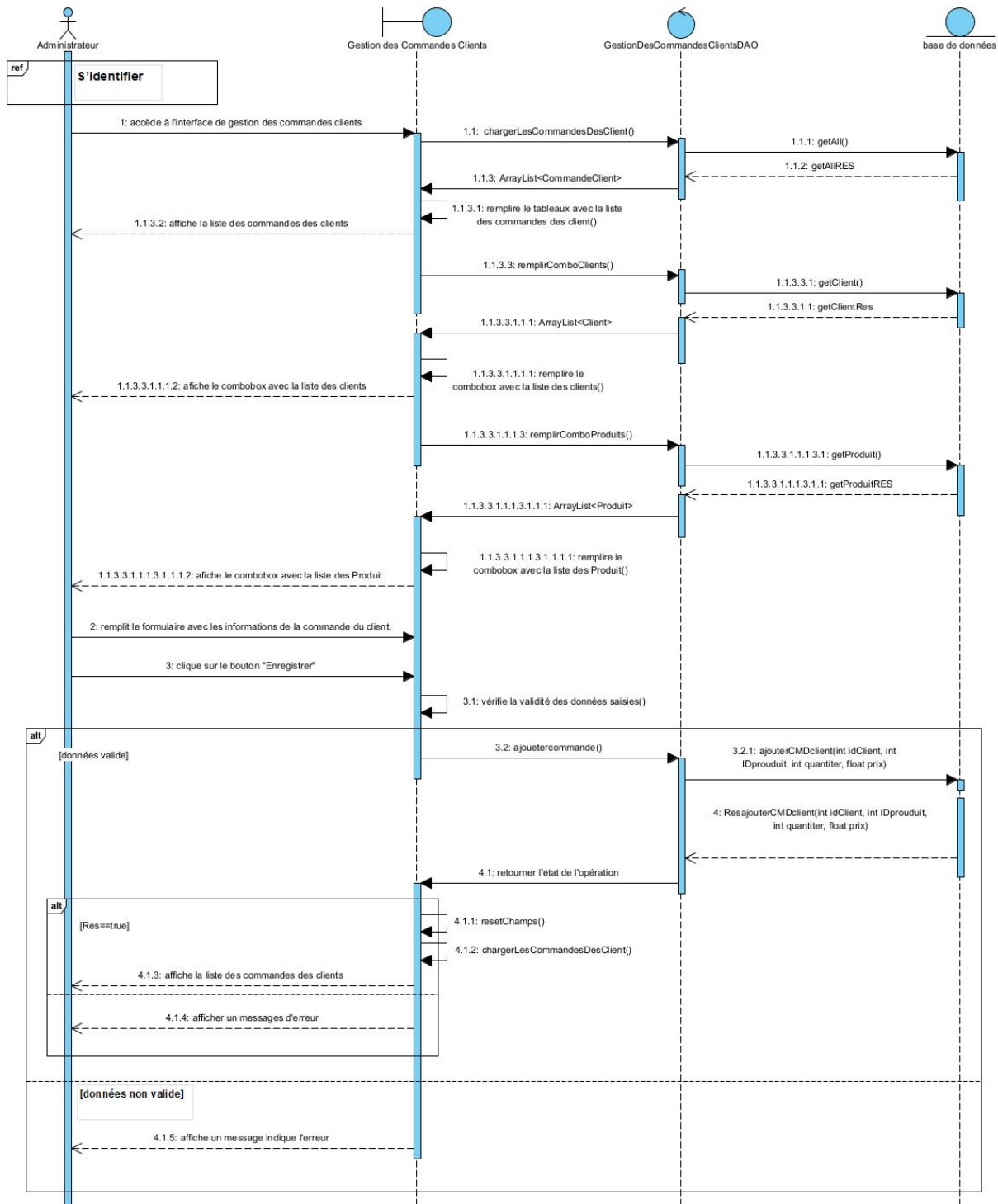


Figure 32 : diagramme d'interaction : (Gestion des Commandes Clients)

Réalisation :

L'interface de Gestion des Commandes Clients :

The interface is titled 'Gestion des Commandes Clients' and is divided into two main sections. On the left, a sidebar contains a table titled 'Liste des commandes'. On the right, a main panel contains a form to register a new order.

**Liste des commandes**

N° de commande	Client	Date	Montant total
6	anas	2025-03-26	600.0 DH
8	anas	2025-03-28	12500.0 DH
7	hichame	2025-03-28	150.0 DH

**Gestion des Commandes Clients**

**Liste des Client :**

**Liste des produits :**

**Quantité :**

**Prix :**

**Enregistrer Commande**

Figure 33 : image de l'interface de Gestion des Commandes Clients



# Tests et Validation

## 1. Introduction

Les tests constituent une étape essentielle dans tout projet logiciel. Ils permettent de garantir la fiabilité, la robustesse et la stabilité de l'application. Dans le cadre de notre projet **de Gestion des Stocks**, nous avons mis en place des tests unitaires afin de valider les différentes fonctionnalités proposées par l'application.

## 2. Outils utilisés

Pour réaliser les tests et analyser la qualité du code, nous avons utilisé les outils suivants :

- **JUnit 5** : Framework de test pour Java, intégré dans notre projet Maven, permettant d'écrire et d'exécuter des tests unitaires.
- **JaCoCo (Java Code Coverage)** : Outil de mesure de couverture de code, utilisé pour générer des rapports indiquant quelles parties du code sont effectivement exécutées lors des tests.
- **SonarLint** : Extension d'analyse statique intégrée à notre environnement de développement (IDE), utilisée pour détecter en temps réel les erreurs potentielles, les mauvaises pratiques et les vulnérabilités dans le code.

## 3. Configuration Maven

Le fichier pom.xml contient la configuration nécessaire pour utiliser JaCoCo :

```
119 <!-- JaCoCo Plugin for Code Coverage - Configuration améliorée -->
120 <plugin>
121   <groupId>org.jacoco</groupId>
122   <artifactId>jacoco-maven-plugin</artifactId>
123   <version>${jacoco.version}</version>
124   <executions>
125     <execution>
126       <id>default-prepare-agent</id>
127       <goals>
128         <goal>prepare-agent</goal>
129       </goals>
130       <configuration>
131         <destFile>${project.build.directory}/jacoco.exec</destFile>
132         <propertyName>argLine</propertyName>
133       </configuration>
134     </execution>
135     <execution>
136       <id>default-report</id>
137       <phase>prepare-package</phase>
138       <goals>
139         <goal>report</goal>
140       </goals>
141       <configuration>
142         <outputDirectory>${project.build.directory}/site/jacoco</outputDirectory>
143       </configuration>
```

```

144 |
145 |
146 |
147 |
148 |
149 |
150 |
151 |
152 |
153 |
154 |
155 |
156 |
157 |
158 |
159 |
160 |
161 |
162 |
163 |
164 |
165 |
166 |
167 |

```

```

</execution>
<execution>
  <id>jacoco-check</id>
  <goals>
    <goal>check</goal>
  </goals>
  <configuration>
    <rules>
      <rule>
        <element>BUNDLE</element>
        <limits>
          <limit>
            <counter>LINE</counter>
            <value>COVEREDRATIO</value>
            <minimum>0.80</minimum>
          </limit>
        </limits>
      </rule>
    </rules>
  </configuration>
</execution>
</executions>
</plugin>

```

Une fois la configuration faite, les tests sont lancés via :

- mvn clean test
- mvn jacoco:report

Un rapport HTML est généré dans le dossier gestion\_du\_stock\target\site\jacoco.

## 4. Méthodes JUnit utilisées

**Les tableaux suivants présentent les principales méthodes de test et annotations JUnit 5 utilisées dans le cadre de nos tests unitaires.**

Chaque tableau correspond à une classe de test spécifique et décrit les méthodes testées ainsi que leur rôle dans la mise en place, l'exécution ou la validation des fonctionnalités. Ces éléments permettent de s'assurer du bon fonctionnement de chaque composant testé, tout en garantissant la robustesse et la fiabilité du code.

### 1. Tableau des méthodes de test – AdministrateurDAOTest

Méthode de test	Description
testConnexionValide()	Vérifie que la méthode getConnexion retourne true lorsque les identifiants (email et mot de passe) sont corrects.
testConnexionInvalide()	Vérifie que la méthode getConnexion retourne false lorsque les identifiants sont incorrects ou inexistantes.

## 2. Tableau des méthodes de test – ClientTest

Méthode de test	Description
testConstructeurComplet()	Vérifie que le constructeur complet initialise correctement tous les attributs d'un client.
testConstructeurSimple()	Vérifie que le constructeur simple initialise uniquement l'ID et le nom, et laisse les autres attributs à null.
testSetIdClient()	Teste le bon fonctionnement du setter setIdClient().
testSetNom()	Teste la modification du nom via le setter setNom().
testSetAdresse()	Vérifie que l'adresse peut être modifiée correctement avec setAdresse().
testSetTele()	Vérifie la mise à jour correcte du numéro de téléphone avec setTele().
testSetEmail()	Vérifie la modification de l'adresse email avec setEmail().

## 3. Tableau des méthodes de test – CommandeClientTest

Méthode de test	Description
testConstructeurEtGetters()	Vérifie que le constructeur initialise correctement tous les attributs de la commande et que les getters retournent les bonnes valeurs.
testSetters()	Teste la modification de chaque attribut (idCommande, prix, dateCommande, idClient, nom) via les setters.

## 4. Tableau des méthodes de test – DAOTest

Méthode de test	Description
testGetInstanceNotNull()	Vérifie que la méthode getInstance() retourne bien une instance de connexion non nulle.

Méthode de test	Description
testConnectionIsValid()	Vérifie que la connexion à la base de données est valide (test de validité via isValid()).

#### 5. Tableau des méthodes de test – CommandeFournisseursTest

Méthode de test	Description
testConstructeurEtGettersAvecFournisseur()	Vérifie que le constructeur initialise correctement les attributs d'une commande fournisseur et que les getters fonctionnent comme prévu.
testSettersAvecFournisseur()	Teste la mise à jour des attributs d'une commande fournisseur à l'aide des setters (idCommande, prix, dateCommande, idFournisseurs, nom).

#### 6. Tableau des méthodes de test – FournisseurTest

Méthode de test	Description
testConstructeurComplet()	Vérifie que le constructeur complet initialise correctement tous les attributs d'un fournisseur.
testConstructeurPartiel()	Vérifie que le constructeur partiel initialise uniquement l'ID et le nom, en laissant les autres attributs à null.
testSetIdFournisseur()	Teste la mise à jour de l'identifiant du fournisseur via le setter setIdFournisseur().
testSetNom()	Vérifie la modification du nom du fournisseur via setNom().
testSetContact()	Teste la mise à jour du contact (email/téléphone) via le setter setContact().
testSetAdresse()	Vérifie que l'adresse peut être correctement modifiée via setAdresse().

## 7. Tableau des méthodes de test – GestionDesCommandesClientsDAOTest

Méthode de test	Description
testAjouterCMDclient_Reussi()	Vérifie que l'ajout d'une commande client réussit lorsque les données fournies sont valides.
testAjouterCMDclient_EchecClientInexistant()	Vérifie que l'ajout échoue si l'ID client fourni n'existe pas dans la base.
testGetAll_RetourneCommandes()	Vérifie que la méthode getAll() retourne une liste non nulle et non vide.
testGetAll_FormatDonneesCorrect()	Vérifie que chaque commande retournée par getAll() contient des valeurs cohérentes (valeurs positives et non vides).

## 8. Tableau des méthodes de test – GestionDesProduitsDAOTest

Méthode de test	Description
testGetAll_RetourneProduits()	Vérifie que la méthode getAll() retourne une liste de produits non nulle et non vide, avec des champs valides.
testGetByCat_RetourneProduitsCategorie()	Vérifie que la méthode getByCat() retourne uniquement des produits appartenant à la catégorie spécifiée.
testGetCategorie_RetourneCategories()	Vérifie que la méthode getCategorie() retourne une liste de catégories non vide et bien formée.

Méthode de test	Description
testAjouterProduit_Success()	Teste si l'ajout d'un nouveau produit avec des données valides est effectué avec succès.
testModifierProduit_Success()	Vérifie que la modification d'un produit existant avec de nouvelles données réussit.
testSupprimerProduit_Success()	Vérifie qu'un produit ajouté peut être supprimé avec succès à l'aide de son ID.

#### 9. Tableau des méthodes de test -estiondesCommandesFournisseursDAOTest

Méthode de test	Description
testAjouterCMDFournisseurs_Success()	Vérifie que l'ajout d'une commande fournisseur valide réussit et retourne true.
testAjouterCMDFournisseurs_FournisseurInexistant()	Vérifie que l'ajout d'une commande avec un ID fournisseur inexistant échoue et retourne false.
testGetAll_RetourneCommandes()	Vérifie que la méthode getAll() retourne une liste non nulle et non vide de commandes.
testGetAll_FormatDonneesCorrect()	Vérifie que chaque commande retournée par getAll() contient des données cohérentes et valides.

#### 10. Tableau des méthodes de test – GérerLesClientsDAOTest

Méthode de test	Description
testAjouterClient()	Vérifie que l'ajout d'un client avec des informations valides fonctionne correctement.
testGetAllClients()	Vérifie que la méthode getAll() retourne une liste contenant le client récemment ajouté.
testGetClient()	Vérifie que la méthode getClient() retourne une liste filtrée contenant le client ahmad.
testSupprimerClient()	Vérifie que la suppression du client ajouté précédemment est réussie.
testSupprimerClientInexistant()	Vérifie que la suppression d'un client avec un identifiant inexistant échoue comme prévu.

#### 11. Tableau des méthodes de test – GérerLesFournisseursDAOTest

Méthode de test	Description
testAjouterFournisseur()	Vérifie que l'ajout d'un fournisseur valide est effectué avec succès.
testGetAllFournisseurs()	Vérifie que la méthode getAll() retourne une liste contenant le fournisseur récemment ajouté.
testGetFournisseurs()	Vérifie que la méthode getFournisseurs() retourne bien le fournisseur "TestFournisseur".
testSupprimerFournisseur()	Vérifie que la suppression du fournisseur précédemment ajouté réussit.
testSupprimerFournisseurInexistant()	Vérifie que la tentative de suppression d'un fournisseur inexistant échoue comme prévu.

## 12. Tableau des méthodes de test – ProduitTest

Méthode de test	Description
testConstructeurPartiel1()	Vérifie l'initialisation partielle d'un produit avec ID, nom et prix, les autres valeurs par défaut.
testConstructeurPartiel2()	Vérifie l'initialisation partielle d'un produit avec ID catégorie et nom de catégorie.
testConstructeurCompleet()	Vérifie l'initialisation complète d'un produit avec toutes les propriétés.
testSetIdProduit()	Vérifie la modification de l'ID du produit via un setter.
testSetNomProduit()	Vérifie la modification du nom du produit.
testSetDescription()	Vérifie la modification de la description du produit.
testSetPrix()	Vérifie la modification du prix du produit.
testSetQuantiteEnStock()	Vérifie la modification de la quantité en stock du produit.
testSetIdCategorie()	Vérifie la modification de l'ID de la catégorie.
testSetNomCategorie()	Vérifie la modification du nom de la catégorie.

## 5. Rapport de couverture de code (JaCoCo)

**L'exécution des tests unitaires a permis de générer un rapport de couverture,** mettant en évidence le pourcentage de code effectivement testé.

Ce rapport fournit une vision claire de la qualité des tests réalisés et du niveau de couverture obtenu. Voici un aperçu des résultats obtenus :



## Tableau de couverture du code – Package entite

Classe	Instructions couvertes	Méthodes couvertes	Lignes couvertes	Classes testées
Client	95 %	13 / 14	27 / 28	1 / 1
Produit	100 %	17 / 17	39 / 39	1 / 1
CommandeFournisseurs	100 %	11 / 11	22 / 22	1 / 1
CommandeClient	100 %	11 / 11	22 / 22	1 / 1
Fournisseur	100 %	10 / 10	22 / 22	1 / 1
<b>Total</b>	<b>99 % (314 / 317)</b>	<b>62 / 63</b>	<b>132 / 133</b>	<b>5 / 5</b>

gestion\_du\_stock > entite

**entite**






Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed Cxty	Missed Lines	Missed Methods	Missed Classes
Client		95 %	n/a	n/a	1 13	1 27	1 13	0 1
Produit		100 %	n/a	n/a	0 17	0 39	0 17	0 1
CommandeFournisseurs		100 %	n/a	n/a	0 11	0 22	0 11	0 1
CommandeClient		100 %	n/a	n/a	0 11	0 22	0 11	0 1
Fournisseur		100 %	n/a	n/a	0 10	0 22	0 10	0 1
Total	3 of 317	99 %	0 of 0	n/a	1 62	1 132	1 62	0 5

Figure 34 : Couverture de code du package entite (rapport JaCoCo)

Cette capture d'écran présente la couverture des tests unitaires pour les classes du package entite.

Elle montre un très bon taux de couverture global (99 %), indiquant que presque toutes les instructions, méthodes et lignes de code ont été testées efficacement.

## Tableau de couverture du code – package dao

Classe	Couverture instructions	Couverture branches	Méthodes testées	Lignes testées
DAO	40 %	37 %	6 / 8	49 / 81
GestionDesProduitsDAO	74 %	75 %	8 / 10	80 / 104
GérerLesClientsDAO	81 %	100 %	5 / 6	39 / 51
GérerLesFournisseursDAO	87 %	100 %	5 / 6	31 / 40

Classe	Couverture instructions	Couverture branches	Méthodes testées	Lignes testées
GestionDesCommandesClientsDAO	89 %	66 %	3 / 4	40 / 46
GestionDesCommandesFournisseursDAO	90 %	66 %	6 / 7	42 / 47
AdministrateurDAO	80 %	n/a	2 / 3	12 / 15
<b>Total</b>	<b>76 %</b>	<b>59 %</b>	<b>32 / 44</b>	<b>293 / 384</b>

gestion\_du\_stock > dao

dao

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
DAO	<div><div></div></div>	40 %	<div><div></div></div>	37 %	15	22	32	49	2	6	0	1
GestionDesProduitsDAO	<div><div></div></div>	74 %	<div><div></div></div>	75 %	3	12	24	80	2	8	0	1
GérerLesClientsDAO	<div><div></div></div>	81 %	<div><div></div></div>	100 %	1	8	12	39	1	5	0	1
GérerLesFournisseursDAO	<div><div></div></div>	87 %	<div><div></div></div>	100 %	1	8	9	31	1	5	0	1
GestionDesCommandesClientsDAO	<div><div></div></div>	89 %	<div><div></div></div>	66 %	3	6	6	40	1	3	0	1
GestionDesCommandesFournisseursDAO	<div><div></div></div>	90 %	<div><div></div></div>	66 %	3	6	5	42	1	3	0	1
AdministrateurDAO	<div><div></div></div>	80 %	n/a		1	2	3	12	1	2	0	1
Total	225 of 964	76 %	26 of 64	59 %	27	64	91	293	9	32	0	7

Figure 35 : Couverture de code du package dao (rapport JaCoCo)

## 6. Conclusion des tests

Cette capture d'écran présente la couverture des tests unitaires pour les classes du package dao.

Les tests réalisés ont permis de :

- Vérifier le bon fonctionnement des principales fonctionnalités (CRUD, login),
- Identifier et corriger certains bugs (notamment les doublons et erreurs de requêtes),
- Valider l'intégration avec la base de données MySQL,
- Assurer une couverture de code optimale grâce à JaCoCo.

L'intégration des tests dans un projet Maven avec JaCoCo atteste également d'une bonne maîtrise des outils de développement Java, garantissant ainsi la qualité et la fiabilité du code.

## Conclusion Générale :

Le projet de développement du Module de Gestion des Stocks pour un magasin d'ascenseurs a permis de mettre en place une solution complète et fonctionnelle répondant aux exigences définies dans le cahier des charges initial. Cette application desktop développée en Java avec Swing offre désormais au magasin un outil performant pour optimiser sa gestion commerciale et logistique.

### Synthèse des réalisations

La mise en œuvre de ce projet a permis d'accomplir plusieurs objectifs majeurs :

1. **Architecture robuste** : Une conception orientée objet bien structurée avec une séparation claire entre les couches de présentation, métier et données.
2. **Couverture fonctionnelle complète** : Toutes les fonctionnalités prioritaires ont été implémentées avec succès :
  - Gestion complète des produits (ajout, modification, suppression, catégorisation)
  - Suivi précis des stocks
  - Gestion efficace des commandes clients et fournisseurs
  - Administration des clients et fournisseurs
3. **Qualité logicielle** : Les tests unitaires ont démontré un excellent niveau de qualité, avec une couverture de code atteignant 99% pour les entités métier et 76% pour les classes DAO.
4. **Ergonomie soignée** : Les interfaces utilisateur ont été conçues pour être intuitives et efficaces, facilitant le travail quotidien de l'administrateur.

### Bilan technique

L'utilisation de technologies standard comme Java, Swing et MySQL a permis de créer une application performante, maintenable et évolutive. La méthodologie RUP (Rational Unified Process) a structuré efficacement le développement en quatre phases bien distinctes, permettant de livrer un produit de qualité dans les délais impartis.

Les tests unitaires avec JUnit et l'analyse de la qualité du code avec des outils comme PMD et SonarLint ont contribué à garantir la robustesse et la fiabilité de l'application.