**République Algérienne Démocratique et Populaire**

**Ministère de l'Enseignement Supérieur et de la Recherche Scientifique**

**Université M'hamed Bougara – Boumerdès**

---

**Faculté des Sciences**

**Département d'Informatique**

**Domaine :** Mathématiques et Informatique.
**Filière :** Informatique.
**Spécialité :** SI.
**Année universitaire:**2025/2026
**N° de l'Arrêté d'habilitation de la spécialité : arrêté n° du 04/06/2026.**

# Mémoire de fin d'études

En vue de l'obtention du diplôme de **Licence Académique**

## Thème :

**Stock et Facturation d'une usine d'eau minérale .**

**Présenté par :**
–KHELALFA Sarah.
–ABDESSELAMI Ilhem .
–FERHOUM Sanaa.
–BENFELLAH Ikram.

**Soutenu le :** 04/06/2026

-M.MEFTAH Mustapha :Encadrant.
-M.REZKI Mohamed Amine :Examinateur.

# *Dédicace*

*Nous dédions ce modeste travail :*

*À nos chers parents,*
*pour leur amour, leurs sacrifices, leur patience et leurs encouragements tout au long de notre parcours universitaire.*

*À nos familles et à nos proches,*
*pour leur soutien moral et leur confiance.*

*À nos amis(es) et collègues,*
*pour leur aide précieuse et leur motivation.*

# Remerciements

Tout d'abord, nous remercions Allah le Tout-Puissant de nous avoir accordé la force, la patience et la volonté nécessaires pour mener à bien ce travail.

Nous exprimons nos sincères remerciements à l'ensemble du corps enseignant du département d'informatique de l'Université M'hamed Bougara de Boumerdès pour la qualité de la formation reçue et leur accompagnement tout au long de notre parcours universitaire.

Nous tenons à adresser nos vifs remerciements à notre encadrant, Monsieur Meftah, pour son encadrement, sa disponibilité, ses conseils précieux et ses orientations qui ont contribué à la réalisation de ce mémoire.

Nous remercions également nos familles pour leur soutien moral, leurs encouragements et leur patience durant toute la période de nos études.

Enfin, nous adressons nos remerciements à toutes les personnes qui ont contribué, de près ou de loin, à l'élaboration de ce travail.

# Contents

**Table des Matières** **4**

**Liste des Figures** **6**

**Liste des Tableaux** **7**

**Liste des abréviations** **7**

**1 Introduction Générale** **8**
1.1 Contexte et justification du projet . . . . . 8

**2 Chapitre 1 : Présentation générale de l'usine d'eau minérale** **10**
2.1 Présentation de l'usine . . . . . 10
2.2 Processus de production de l'eau minérale . . . . . 10
2.3 Organisation interne . . . . . 10
2.4 Problèmes rencontrés dans la gestion actuelle . . . . . 11
2.5 Besoins fonctionnels de l'usine . . . . . 11

**3 Chapitre 2: Étude de l'existant et analyse des besoins** **12**
3.1 Étude du système actuel . . . . . 12
3.2 Inconvénients du système existant . . . . . 12
3.3 Identification des utilisateurs du système . . . . . 12
3.4 Analyse des besoins fonctionnels . . . . . 12
3.4.1 Gestion des produits . . . . . 12
3.4.2 Gestion du stock . . . . . 13
3.4.3 Gestion des clients . . . . . 13
3.4.4 Gestion des factures . . . . . 13
3.5 Analyse des besoins non fonctionnels . . . . . 13
3.6 Cahier des charges (résumé) . . . . . 13

**4 Chapitre 3 :Conception du système** **14**
4.1 Modélisation des processus : . . . . . 14
4.1.1 Diagramme de cas d'utilisation Vue générale : . . . . . 14
4.1.2 . Diagramme de cas d'utilisation Gestion de la facturation : . . . . . 15
4.2 Diagramme de classe : . . . . . 16
4.2.1 Description détaillée du diagramme de classes : . . . . . 17
4.2.2 Description des relations : . . . . . 19
4.2.3 Logique fonctionnelle globale . . . . . 19
4.2.4 Conclusion : . . . . . 20
4.3 Diagramme de séquence : . . . . . 21
4.3.1 Description du diagramme de séquence : . . . . . 21
4.4 Modélisation de la base de données : . . . . . 22
4.4.1 Modèle Conceptuel de Données (MCD) : . . . . . 22
4.4.2 Entité Client : . . . . . 22
4.4.3 Entité Commande : . . . . . 22
4.4.4 Entité Ligne Commande : . . . . . 22
4.4.5 Entité Produit : . . . . . 22
4.4.6 Entité Facture : . . . . . 23
4.4.7 Entité Paiement : . . . . . 23
4.4.8 Entité Responsable : . . . . . 23

4.5 Modèle logique de données (MLD) : . . . . . 24
4.5.1 Explication du MLD – Gestion des commandes : . . . . . 24

**5 Chapitre 4 : Réalisation de l'application 26**
5.1 Environnement de développement . . . . . 26
5.2 Architecture de l'application . . . . . 27
5.3 Description des principales interfaces . . . . . 28
5.3.1 Interface de Gestion du Stock . . . . . 28
5.3.2 Interface de Gestion des Clients . . . . . 29
5.3.3 Interface de Gestion de la Facturation . . . . . 29
5.4 Implémentation de la base de données . . . . . 30
5.5 Sécurité et gestion des accès . . . . . 32
5.6 Tests et résultats . . . . . 33

**Conclusion Générale 34**

**Références 36**

# Liste des Figures


1 Diagramme de cas d'utilisation du système . . . . . 14
2 Diagramme de cas d'utilisation Gestion de la facturation . . . . . 15
3 Diagramme de classe du système . . . . . 16
4 Diagramme de Séquence du Traitement des Commandes . . . . . 21
5 Modèle Conceptuel de Données (MCD) . . . . . 22
6 Modèle Logique de Données (MLD) . . . . . 24
7 Technologies et outils de développement utilisés . . . . . 26
8 Architecture technique de l'application . . . . . 27
9 Interface de gestion du stock de l'usine . . . . . 28
10 Interface de gestion des clients . . . . . 29
11 Interface globale de facturation et de suivi des paiements (AQUASTOCK) . . . 30
12 Structure de la base de données `gestion_stock_eau` sous phpMyAdmin . . . . . 31

# Liste des abréviations

| Abréviation | Signification |
|---|---|
| **CSS** | Cascading Style Sheets |
| **FAO** | Food and Agriculture Organization |
| **FK** | Foreign Key (Clé Étrangère) |
| **HACCP** | Hazard Analysis Critical Control Point |
| **HT** | Hors Taxe |
| **HTML** | HyperText Markup Language |
| **ISO / IEC** | International Organization for Standardization / International Electrotechnical Commission |
| **JS** | JavaScript |
| **MCD** | Modèle Conceptuel de Données |
| **MLD** | Modèle Logique de Données |
| **MPD** | Modèle Physique de Données |
| **OMS** | Organisation Mondiale de la Santé |
| **PFE** | Projet de Fin d'Études |
| **PHP** | PHP: Hypertext Preprocessor |
| **PK** | Primary Key (Clé Primaire) |
| **SI** | Système d'Information |
| **SGBDR** | Système de Gestion de Base de Données Relationnelle |
| **TVA** | Taxe sur la Valeur Ajoutée |
| **TTC** | Toutes Taxes Comprises |
| **UML** | Unified Modeling Language |
| **XAMPP** | Cross-platform Apache, MariaDB, PHP, Perl |

# 1 Introduction Générale

## 1.1 Contexte et justification du projet

Avec le développement rapide des technologies de l'information, les entreprises industrielles cherchent à moderniser leurs systèmes de gestion afin d'améliorer leur efficacité et leur performance. Dans ce contexte, les systèmes informatiques jouent un rôle essentiel dans l'automatisation des processus et la gestion des ressources.

Dans le domaine agroalimentaire, notamment dans les usines d'eau minérale, la gestion des produits, du stock et de la facturation constitue un élément clé pour assurer la continuité de la production et la satisfaction des clients. Selon l'Organisation mondiale de la Santé (OMS), la traçabilité et le contrôle des produits alimentaires sont indispensables pour garantir la qualité et la sécurité sanitaire.

De plus, le Codex Alimentarius souligne l'importance d'une gestion rigoureuse des produits alimentaires, depuis leur production jusqu'à leur distribution.

Ainsi, la mise en place d'une application de gestion du stock et de la facturation devient nécessaire pour améliorer l'organisation interne et assurer un suivi fiable des opérations.

### Problématique

Dans une usine d'eau minérale, la gestion traditionnelle (manuelle ou semi-informatisée) du stock et de la facturation présente plusieurs difficultés, telles que :

- le manque de précision dans le suivi des quantités en stock,
- les erreurs humaines dans la création des factures,
- l'absence de traçabilité des produits,
- la difficulté d'accès rapide à l'information,
- la mauvaise coordination entre les services.

Selon la FAO (Food and Agriculture Organization), une mauvaise gestion des ressources peut entraîner des pertes économiques importantes et une diminution de la performance de l'entreprise.

Face à ces problèmes, il devient essentiel de proposer une solution informatique permettant d'automatiser et de sécuriser ces processus.

### Objectifs du projet

L'objectif principal de ce projet est de concevoir et développer une application web dédiée à la gestion du stock et de la facturation d'une usine d'eau minérale.

Les objectifs spécifiques sont :

- Assurer une gestion efficace des produits et du stock,
- Automatiser la création et le suivi des factures,
- Réduire les erreurs humaines,
- Améliorer la traçabilité des opérations,

- Faciliter l'accès aux informations en temps réel,
- Offrir une interface simple et intuitive pour les utilisateurs.

## Méthodologie de travail

Pour mener à bien ce projet, nous avons adopté une démarche structurée basée sur les étapes suivantes :

- Analyse des besoins,
- Modélisation avec UML (diagrammes de cas d'utilisation, de classes, de séquence),
- Conception de la base de données (MCD, MLD, MPD),
- Développement de l'application web (Html, CSS , js, PHP, MySQL),
- Tests et validation du système.

Cette méthodologie permet d'assurer une conception claire et une réalisation progressive du projet.

## Organisation du mémoire

Ce mémoire est structuré comme suit :

- Chapitre 1 : Présentation générale de l'usine d'eau minérale,
- Chapitre 2 : Étude de l'existant et analyse des besoins
- Chapitre 3 : Conception du système (UML et base de données),
- Chapitre 4 : Réalisation et implémentation de l'application
- Conclusion générale.

# 2 Chapitre 1 : Présentation générale de l'usine d'eau minérale

## 2.1 Présentation de l'usine

Une usine d'eau minérale est une unité industrielle spécialisée dans l'exploitation, le traitement, le conditionnement et la distribution de l'eau destinée à la consommation humaine.

Selon l'Organisation mondiale de la Santé (OMS), l'eau destinée à la consommation doit respecter des normes strictes de qualité afin de garantir la sécurité sanitaire des consommateurs.

L'eau minérale naturelle provient d'une source souterraine protégée et se caractérise par sa pureté ainsi que par sa composition minérale stable. D'après le Codex Alimentarius, elle ne doit subir aucun traitement susceptible de modifier sa composition essentielle.

Les principales activités de l'usine incluent la production, le stockage, la distribution et la gestion administrative, nécessitant une organisation rigoureuse et un suivi permanent.

## 2.2 Processus de production de l'eau minérale

Le processus de production suit plusieurs étapes conformes aux normes internationales de qualité alimentaire.

- **Captage de l'eau** : l'eau est extraite directement de la source naturelle dans des conditions protégées.
- **Filtration et contrôle** : l'eau est filtrée et analysée afin de garantir sa qualité microbiologique et chimique.
- **Embouteillage** : l'eau est conditionnée dans différents formats (0.5L, 1.5L, etc.).
- **Étiquetage et emballage** : chaque produit reçoit une étiquette contenant les informations nécessaires (date, lot, composition).
- **Stockage** : les produits finis sont stockés dans des conditions appropriées.

Selon la FAO, ces étapes doivent respecter les principes du système HACCP afin d'assurer la sécurité des produits alimentaires.

## 2.3 Organisation interne

L'usine est structurée en plusieurs services :

- **Service de production** : responsable de l'embouteillage et du conditionnement.
- **Service de stock** : assure la gestion des produits et le suivi des quantités disponibles.
- **Service de vente** : gère la relation avec les clients et les commandes.
- **Service administratif** : s'occupe de la facturation, des documents et du suivi financier.

La coordination entre ces services permet d'assurer un fonctionnement efficace de l'usine.

## 2.4 Problèmes rencontrés dans la gestion actuelle

Malgré l'organisation existante, plusieurs problèmes peuvent être observés :

- Gestion manuelle du stock entraînant des erreurs,
- Absence de suivi en temps réel des quantités disponibles,
- Difficultés dans la création et la gestion des factures,
- Manque de traçabilité des produits,
- Perte de temps dans la recherche d'informations.

Selon la Food and Agriculture Organization (FAO), une mauvaise gestion des ressources peut engendrer des pertes économiques et réduire l'efficacité globale de l'entreprise.

## 2.5 Besoins fonctionnels de l'usine

Pour remédier aux problèmes identifiés, l'usine exprime plusieurs besoins fonctionnels :

- Gestion informatisée des produits et du stock,
- Suivi en temps réel des quantités disponibles,
- Automatisation de la facturation,
- Gestion des clients et des commandes,
- Amélioration de la traçabilité des opérations,
- Accès rapide et fiable aux informations.

La mise en place d'un système d'information adapté permet de répondre à ces besoins et d'améliorer les performances de l'entreprise.

# 3 Chapitre 2: Étude de l'existant et analyse des besoins

## 3.1 Étude du système actuel

Actuellement, l'usine d'eau minérale utilise un système de gestion manuel ou partiellement informatisé. Les opérations liées au stock sont souvent enregistrées sur des supports papier ou à l'aide de fichiers simples tels que des feuilles Excel. La facturation est réalisée soit manuellement, soit via des outils basiques ne permettant pas une automatisation complète du processus.

Selon l'Organisation mondiale de la Santé (OMS), la gestion et la traçabilité des produits alimentaires nécessitent des systèmes fiables afin d'assurer la qualité et la sécurité sanitaire. Cependant, l'utilisation de méthodes traditionnelles limite l'efficacité du suivi et augmente les risques d'erreurs.

## 3.2 Inconvénients du système existant

Le système actuel présente plusieurs limites :

- **Risque élevé d'erreurs humaines** : saisie manuelle des données pouvant entraîner des incohérences.
- **Perte de données** : absence de mécanismes de sauvegarde fiables.
- **Absence de suivi en temps réel** : difficulté à connaître l'état exact du stock.
- **Difficulté d'accès à l'information** : recherche lente et non optimisée.
- **Manque de sécurité** : absence de contrôle d'accès et de protection des données.

Selon la Food and Agriculture Organization (FAO), une mauvaise gestion des données peut entraîner une baisse de productivité et des pertes économiques importantes.

## 3.3 Identification des utilisateurs du système

Le système proposé sera utilisé par plusieurs acteurs ayant des rôles différents :

- **Administrateur** : responsable de la gestion globale du système et des utilisateurs.
- **Responsable du stock** : chargé du suivi des produits et des quantités disponibles.
- **Responsable de facturation** : responsable de la création des factures et du suivi des paiements.

Chaque utilisateur dispose de droits spécifiques selon ses responsabilités.

## 3.4 Analyse des besoins fonctionnels

Les besoins fonctionnels décrivent les services que le système doit offrir.

### 3.4.1 Gestion des produits

- Ajouter, modifier et supprimer des produits.
- Consulter les informations : nom, type, volume, prix.

### 3.4.2 Gestion du stock

- Suivi des entrées et sorties des produits.
- Consultation du stock en temps réel.
- Génération d'alertes en cas de rupture de stock.

### 3.4.3 Gestion des clients

- Enregistrement et mise à jour des informations clients.
- Consultation de l'historique des commandes.
- Suivi des paiements effectués.

### 3.4.4 Gestion des factures

- Génération automatique des factures.
- Impression et archivage des factures.
- Suivi des paiements et des factures impayées.

## 3.5 Analyse des besoins non fonctionnels

Les besoins non fonctionnels définissent les contraintes de qualité du système.

- **Sécurité** : mise en place d'un système d'authentification, gestion des rôles et protection des données.
- **Performance** : temps de réponse rapide et traitement efficace des données.
- **Fiabilité** : sauvegarde régulière des données et disponibilité continue du système.

Selon la norme ISO/IEC 25010, les critères de qualité d'un système logiciel incluent la sécurité, la performance et la fiabilité.

## 3.6 Cahier des charges (résumé)

Le système à développer doit répondre aux exigences suivantes :

- Être simple et facile à utiliser.
- Assurer la sécurité des données.
- Garantir un suivi précis du stock et des ventes.
- Fournir des rapports clairs et exploitables.
- Être accessible uniquement aux utilisateurs autorisés.

# 4 Chapitre 3 :Conception du système

## 4.1 Modélisation des processus :

### 4.1.1 Diagramme de cas d'utilisation Vue générale :

Le diagramme de cas d'utilisation est un outil essentiel dans la modélisation des systèmes. Il permet de représenter les interactions entre les acteurs et les différentes fonctionnalités offertes par le système. Dans notre cas, ce diagramme illustre les principales opérations réalisées par l'administrateur, telles que la gestion des produits, des clients, des commandes et des paiements.

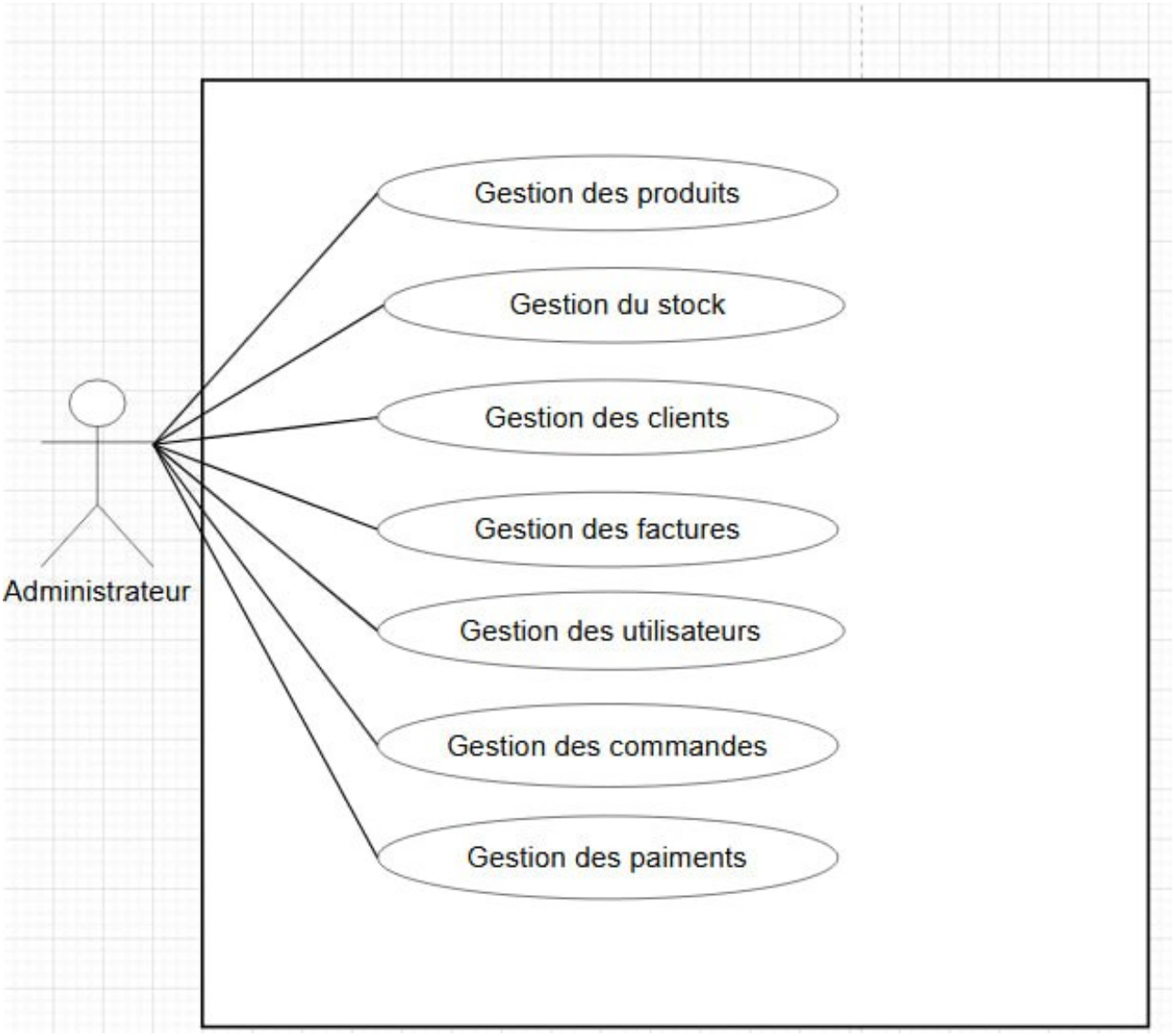



Figure 1: Diagramme de cas d'utilisation du système

## Description des cas d'utilisation

| Cas d'utilisation | Description |
|---|---|
| Gestion des produits | Permet à l'administrateur d'ajouter, modifier, supprimer et consulter les informations des produits (nom, prix, description, etc.). |
| Gestion du stock | Permet de gérer les quantités disponibles des produits, incluant l'entrée, la sortie et la mise à jour du stock. |
| Gestion des clients | Permet d'ajouter, modifier, supprimer et consulter les informations des clients. |
| Gestion des factures | Permet de créer, modifier, supprimer et consulter les factures associées aux commandes. |
| Gestion des utilisateurs | Permet de gérer les comptes utilisateurs, y compris la création, la modification et la suppression. |
| Gestion des commandes | Permet de créer, modifier, suivre et annuler les commandes des clients. |
| Gestion des paiements | Permet d'enregistrer et de suivre les paiements effectués par les clients. |

Table 1: Description des cas d'utilisation

### 4.1.2 . Diagramme de cas d'utilisation Gestion de la facturation :

Ce diagramme décrit le fonctionnement détaillé du processus de facturation au sein du système. Il met en évidence les différentes étapes nécessaires à la création d'une facture ainsi que les options de paiement associées.

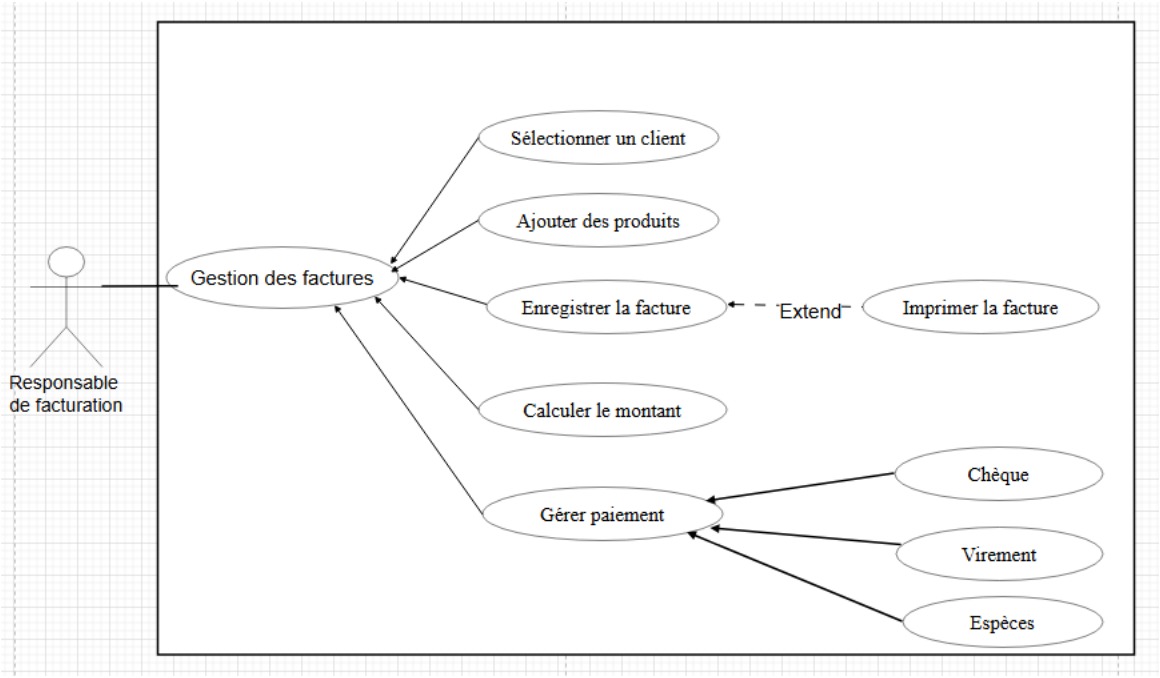



Figure 2: Diagramme de cas d'utilisation Gestion de la facturation

| Élément | Description |
|---|---|
| Acteur | Responsable de facturation |
| Cas d'utilisation principal | Gestion des factures |
| Sélectionner un client | Permet de choisir un client avant de créer une facture. |
| Ajouter des produits | Permet d'ajouter des produits à la facture. |
| Calculer le montant | Permet de calculer le montant total de la facture. |
| Enregistrer la facture | Permet de sauvegarder la facture dans le système après sa création. |
| Imprimer la facture | Permet d'imprimer la facture après son enregistrement. |
| Gérer paiement | Permet de traiter le paiement de la facture. Ce cas inclut plusieurs modes de paiement : *chèque*, *virement* et *espèces*. |
| Paiement par chèque | Permet d'effectuer le paiement via chèque. |
| Paiement par virement | Permet d'effectuer le paiement par virement bancaire. |
| Paiement en espèces | Permet d'effectuer le paiement en espèces. |

Table 2: Description détaillée du cas d'utilisation : Gestion des factures

## 4.2 Diagramme de classe :

Ce diagramme représente la structure statique du système en montrant les classes, leurs attributs, leurs méthodes ainsi que les relations entre elles. Il constitue l'ossature de la base de données.

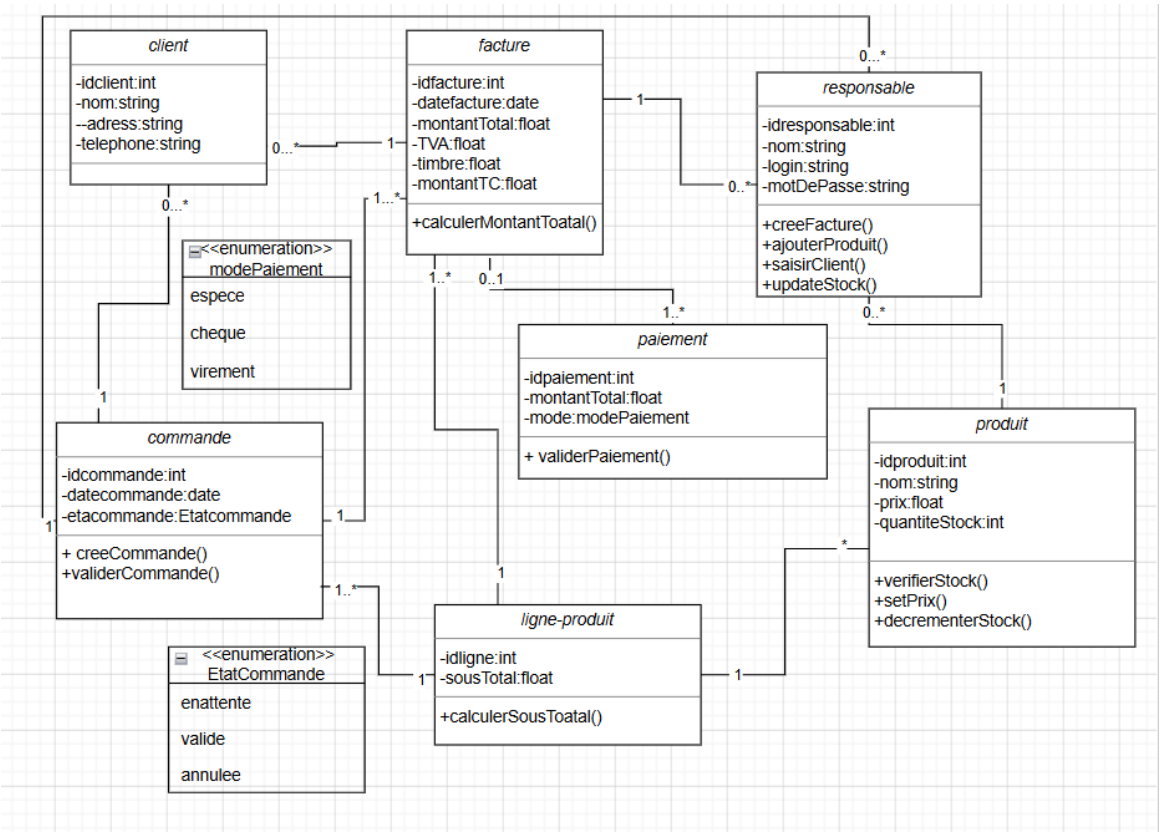



Figure 3: Diagramme de classe du système

### 4.2.1 Description détaillée du diagramme de classes :

**Vue générale :** Ce diagramme de classes représente le système de gestion de commandes, factures, clients et produits. Il modélise les entités principales du système ainsi que leurs attributs, méthodes et relations afin de décrire le fonctionnement logique de l'application.

Le système permet principalement :

- Gérer les clients et créer des commandes.
- Gérer les produits et le stock.
- Générer des factures et enregistrer les paiements.
- Gérer les responsables (utilisateurs du système).

| Classes | Attributs | Méthodes | Rôles |
|---|---|---|---|
| **Classe Client** : Représente les clients enregistrés. | idClient, nom, adresse, telephone | | Un client peut passer plusieurs commandes et posséder plusieurs factures. |
| **Classe Commande** : Représente une commande effectuée par un client. | idCommande, dateCommande, etatCommande | creerCommande(), validerCommande() | Une commande contient plusieurs lignes de produit. |
| **Classe Ligne Produit** : Représente le détail d'un produit dans une commande. | idLigne, sousTotal | Calculer Sous Total() | Elle fait le lien entre commande et produit (classe d'association). |
| **Classe Produit** : Représente les produits disponibles à la vente. | idProduit, nom, prix, quantiteStock | verifierStock(), setPrix(), decrementerStock(qte) | Permet la gestion du stock et des informations produit. |
| **Classe Facture** : Représente une facture générée après validation. | idFacture, dateFacture, montantTotal | calculerMontantTotal(), paiement() | Une facture regroupe les lignes produits et est liée à un client et un responsable. |
| **Classe Paiement** : Représente les informations financières. | TVA, montantTTC, timbre | | Stocke les valeurs calculées pour le paiement final. |
| **Classe ModePaiement** (Enumération) | Espece, Cheque, Virement | | Décrit les différents moyens de paiement. |
| **Classe Responsable** : Représente l'utilisateur du système. | idResponsable, nom, login, motDePasse | ReerFacture(), AjouterProduit(), SaisirClient() | Le responsable manipule les données du système (Admin ou employé). |

Table 3: Description des classes du système

**Description des classes :**

**Description des relations :**

- **Client — Commande (1..*)** : Un client peut avoir plusieurs commandes.
- **Commande — Ligne-Produit (1..*)** : Une commande est composée d'une ou plusieurs lignes de produits.
- **Produit — Ligne-Produit (1)** : Chaque ligne de produit concerne un produit spécifique.
- **Facture — Paiement (0..1)** : Une facture peut avoir un paiement associé.

### 4.2.2 Description des relations :

- **Client — Commande :**
  - **Relation :** 1 à plusieurs (1..*)
  - Un client peut avoir plusieurs commandes.
  - Une commande appartient à un seul client.
- **Commande — Ligne Produit :**
  - **Relation :** 1 à plusieurs
  - Une commande contient plusieurs lignes.
  - Une ligne appartient à une seule Commande.
- **Ligne Produit — Produit :**
  - **Relation :** plusieurs à un
  - Plusieurs lignes peuvent référencer le même produit.
  - Une ligne correspond à un seul produit.
- **Commande — Facture :**
  - **Relation :** 1 à plusieurs
  - Une commande validée correspond à une seule facture.
  - Une facture correspond à une ou plusieurs commandes.
- **Facture — Client :**
  - **Relation :** 1 à 1
  - Une facture appartient à un seul client.
- **Facture — Responsable :**
  - **Relation :** plusieurs à un
  - Un responsable peut créer plusieurs factures.
  - Une facture est créée par un seul responsable.
- **Facture — Paiement :**
  - **Relation :** 1 à plusieurs
  - Une facture possède un seul paiement.
  - Un paiement correspond à une ou plusieurs factures.

### 4.2.3 Logique fonctionnelle globale

Le fonctionnement global du système suit ce flux :

**Client** → passe → **Commande** → contient → **Lignes** → référencent → **Produits**

**Commande validée** → génère → **Facture** → calculée → **Paiement** → avec → **Mode**

Responsable contrôle toutes les opérations.

### 4.2.4 Conclusion :

**Technique :** Ce diagramme garantit :

- Séparation claire des responsabilités.
- Modularité du système.
- Facilité de maintenance.
- Extensibilité (ajout de modules futurs).

Il respecte les principes UML standard et les règles de conception orientée objet.

## 4.3 Diagramme de séquence :

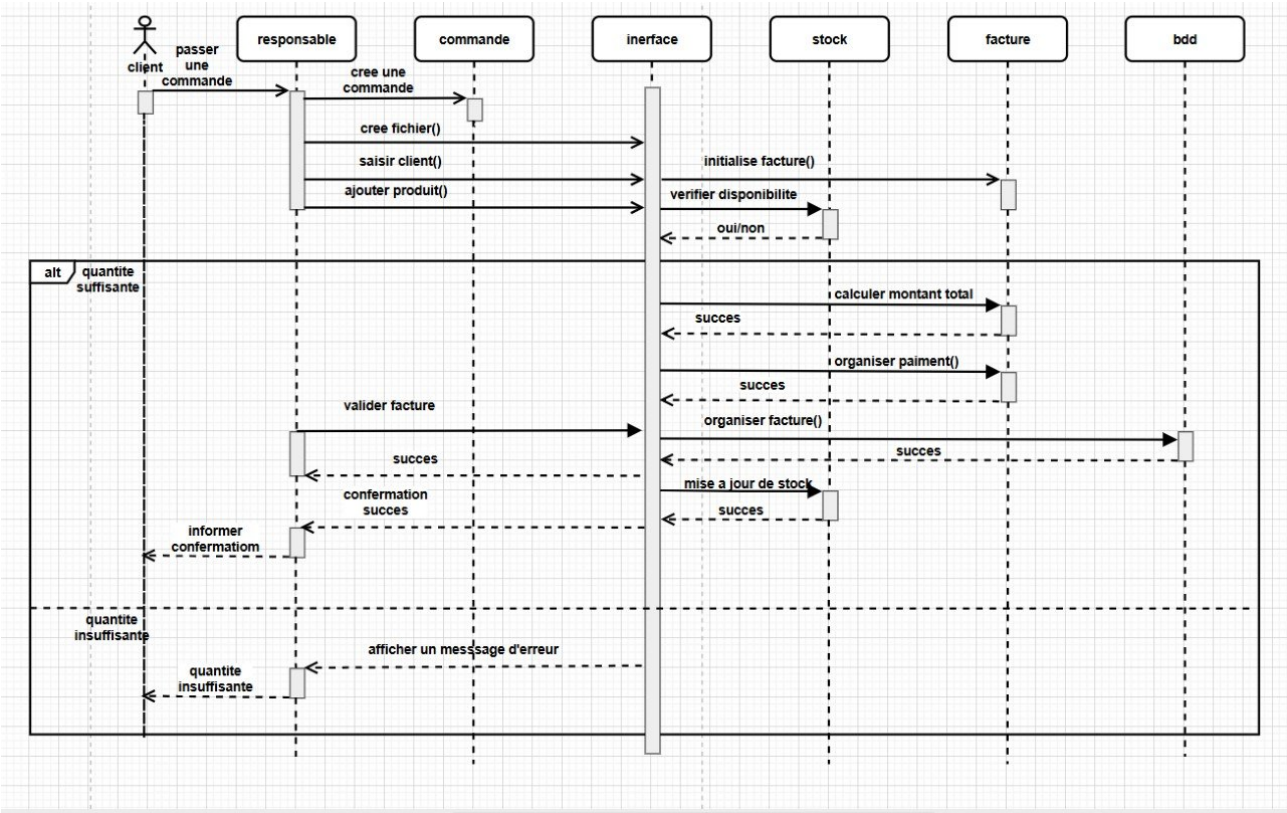



Figure 4: Diagramme de Séquence du Traitement des Commandes

### 4.3.1 Description du diagramme de séquence :

**Gestion d'une commande :** Ce diagramme de séquence illustre le processus de gestion d'une commande dans un système de gestion de stock et de facturation. Il montre les **interactions entre les différents acteurs** du système durant le traitement d'une commande.

Le processus commence lorsque le client passe une commande auprès du responsable. Ce dernier crée une nouvelle commande dans le système à l'aide de l'interface de gestion. Ensuite, il saisit les informations du client et ajoute les produits demandés dans la commande. Après cela, le système vérifie automatiquement la disponibilité des produits dans le stock. Pour représenter les différentes situations possibles, un fragment combiné **alt** est utilisé dans le diagramme.

- **Si la quantité est suffisante :** le système calcule le montant total de la commande, enregistre le paiement, puis génère et valide la facture. La facture est ensuite enregistrée dans la base de données et le stock est mis à jour pour refléter la nouvelle quantité disponible.
- **Si la quantité est insuffisante :** le système affiche un message d'erreur indiquant que les produits demandés ne sont pas disponibles en quantité suffisante.

Enfin, après la mise à jour du stock, le système envoie une confirmation au responsable afin d'informer le client que la commande a été traitée avec succès.

## 4.4 Modélisation de la base de données :

### 4.4.1 Modèle Conceptuel de Données (MCD) :

Ce modèle conceptuel de données représente la gestion des clients, des commandes, des produits, des factures et des paiements dans une usine.

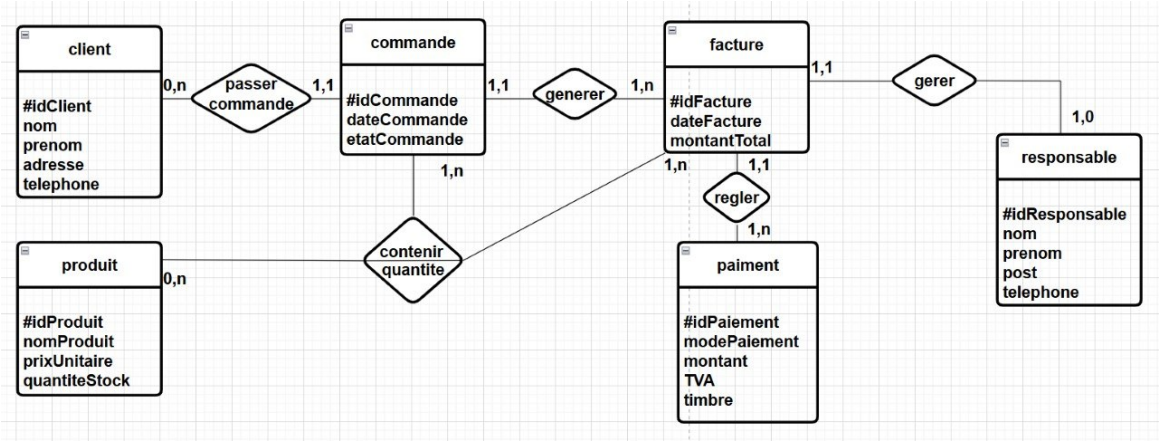



Figure 5: Modèle Conceptuel de Données (MCD)

### 4.4.2 Entité Client :

- **Attributs :** id Client (clé primaire), adresse, telephone, prenom.
- **Relation :** passer commande.
- Un client peut passer 0 à plusieurs commandes (0,n).
- Chaque commande est passée par un seul client (1,1).
- Cela signifie qu'une commande appartient obligatoirement à un client.

### 4.4.3 Entité Commande :

- **Attributs :** id Commande (clé primaire), date Commande, Etat Commande.
- **Relation :** contenir.
- Une commande contient 1 à plusieurs lignes de commande (1,n).
- Chaque ligne de commande appartient à une seule commande (1,1).

### 4.4.4 Entité Ligne Commande :

- **Attributs :** id Ligne Commande (clé primaire), quantite, sous Total.
- **Relation :** concerner.
- Une ligne de commande concerne un seul produit (1,1).
- Un produit peut apparaître dans 0 à plusieurs lignes de commande (0,n).

### 4.4.5 Entité Produit :

- **Attributs :** id Produit (clé primaire), nom Produit, prix Unitaire, quantite Stock.

#### 4.4.6 Entité Facture :

- **Attributs :** id Facture (clé primaire), date Facture, montant Total.
- **Relation :** générer.
- Chaque commande concerne une seule facture (1,1).
- Chaque facture est liée à une ou plusieurs commandes (1,n).

#### 4.4.7 Entité Paiement :

- **Attributs :** id Paiement (clé primaire), mode Paiement, Montant, TVA, timbre.
- **Relation :** régler.
- Une facture est concernée par un seul paiement (1,1).
- Chaque paiement concerne une ou plusieurs factures (1,n).

#### 4.4.8 Entité Responsable :

- **Attributs :** Id Responsable (clé primaire), nom, prenom, telephone, poste.
- **Relation :** gérer.
- Un responsable peut gérer 0 à plusieurs factures (0,n).
- Chaque facture est gérée par un seul responsable (1,1).

Il respecte les règles de gestion et les cardinalités.

## 4.5 Modèle logique de données (MLD) :

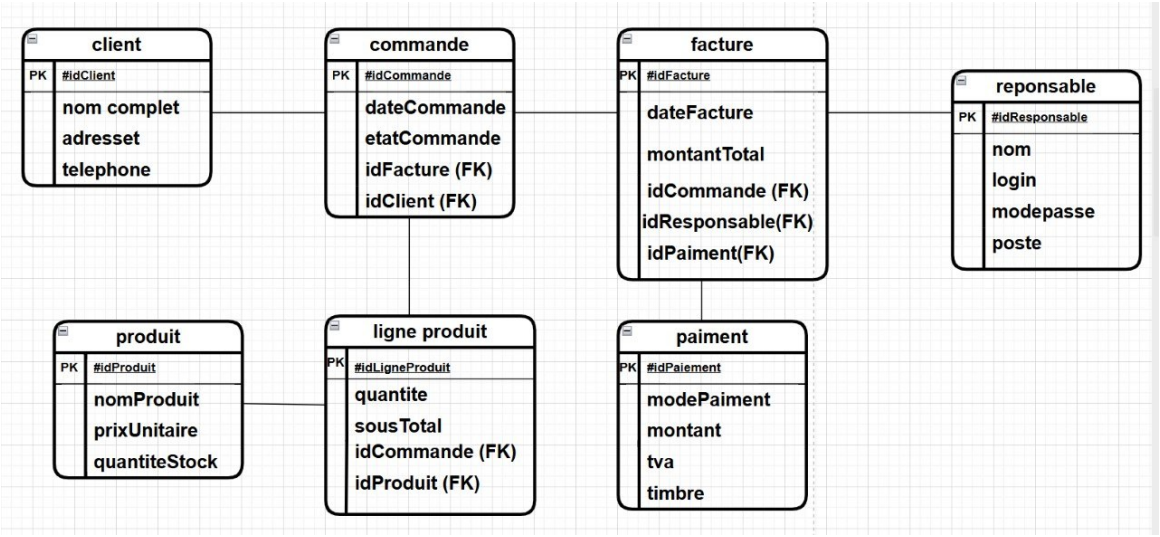



Figure 6: Modèle Logique de Données (MLD)

### 4.5.1 Explication du MLD – Gestion des commandes :

Ce modèle logique de données représente la structure physique de la base de données illustrant la gestion des clients, des commandes, des produits, des factures et des paiements dans une usine.

1. **Entité Client :**
   - **Attributs :** id Client (clé primaire), prenom, adresse, telephone.
   - **Relation :** passer commande.
   - Un client peut passer 0 à plusieurs commandes (0,n).
   - Chaque commande est passée par un seul client (1,1).
   - Cela signifie qu'une commande appartient obligatoirement à un client.
2. **Entité Commande :**
   - **Attributs :** id Commande (clé primaire), date Commande, etat Commande.
   - **Relation :** contenir.
   - Une commande contient 1 à plusieurs lignes de commande (1,n).
   - Chaque ligne de commande appartient à une seule commande (1,1).
3. **Entité Ligne Produit :**
   - **Attributs :** id Ligne Produit (clé primaire), quantite, sous Total.
   - **Relation :** concerner.
   - Une ligne de commande concerne un seul produit (1,1).
   - Un produit peut apparaître dans 0 à plusieurs lignes de commande (0,n).
4. **Entité Produit :**
   - **Attributs :** id Produit (clé primaire), nom Produit, prix Unitaire, Quantite Stock.

5. **Entité Facture :**

   - **Attributs :** id Facture (clé primaire), date Facture, montant Total.
   - **Relation :** générer.
   - Chaque commande concerne une seule facture (1,1).
   - Chaque facture est liée à une ou plusieurs commande (1,n).

6. **Entité Paiement :**

   - **Attributs :** id Paiement (clé primaire), mode Paiement, montant, TVA, timbre.
   - **Relation :** régler.
   - Une facture est réglée par 1 à plusieurs paiements (1,n).
   - Chaque paiement concerne une seule facture (1,1).

7. **Entité Responsable :**

   - **Attributs :** id Responsable (clé primaire), nom, prenom, telephone, poste.
   - **Relation :** gérer.
   - Un responsable peut gérer 0 à plusieurs factures (0,n).
   - Chaque facture est gérée par un seul responsable (1,1).

# 5 Chapitre 4 : Réalisation de l'application

## 5.1 Environnement de développement

L'environnement de développement utilisé pour la réalisation de cette application repose sur plusieurs technologies complémentaires permettant le développement du front-end, du back-end ainsi que la gestion de la base de données.

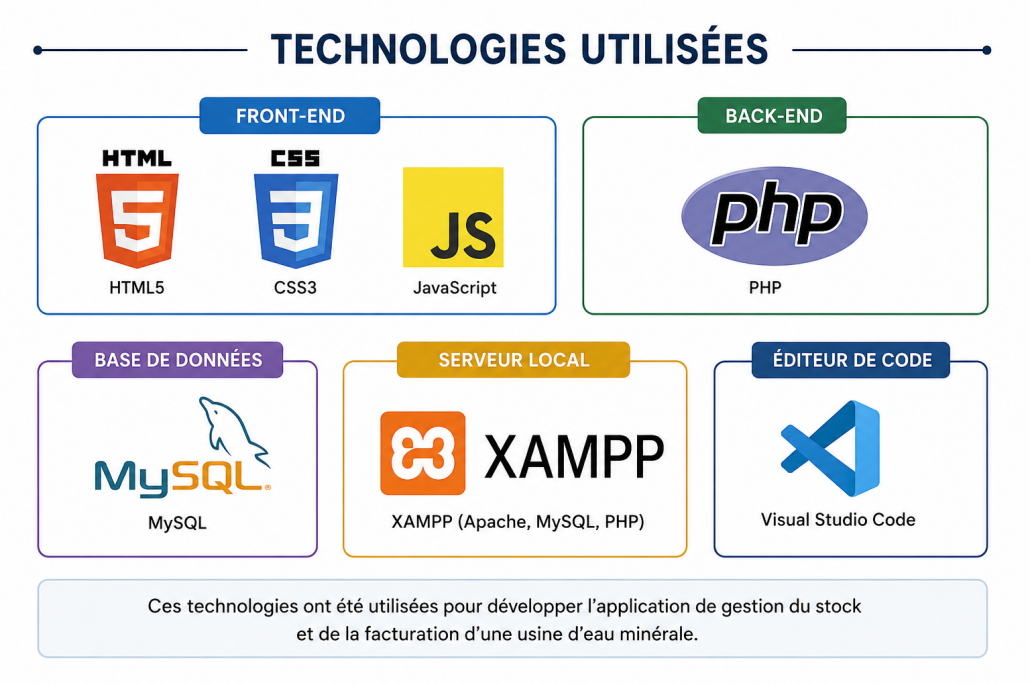



Figure 7: Technologies et outils de développement utilisés

**Pour le développement de l'interface utilisateur (Front-End) :** Nous avons utilisé les technologies standards du web pour garantir une interface fluide et ergonomique :

- **HTML5** : pour structurer le contenu et les éléments des pages web.
- **CSS3** : pour la mise en forme, le design responsive et l'aspect visuel.
- **JavaScript** : pour ajouter des interactions dynamiques et améliorer l'expérience utilisateur.

**Pour la partie serveur (Back-End) :**

- **PHP** : Langage de script côté serveur utilisé pour gérer la logique métier, le traitement des formulaires et la communication sécurisée avec la base de données.

**Pour la gestion des données :**

- **MySQL** : Système de gestion de base de données relationnelle (SGBDR) choisi pour sa performance, sa fiabilité et sa compatibilité avec PHP.

**Environnement de travail et outils :** Enfin, l'environnement local de développement repose sur des outils performants :

- **XAMPP** : Une distribution Apache contenant MySQL et PHP, utilisée comme serveur local pour tester l'application avant son déploiement.
- **Visual Studio Code** : Notre éditeur de code source principal, choisi pour ses nombreuses extensions facilitant le développement et le débogage.

## 5.2 Architecture de l'application

L'application suit une architecture client-serveur basée sur le modèle à trois niveaux (3-Tier), composée de trois couches principales qui interagissent de manière structurée :

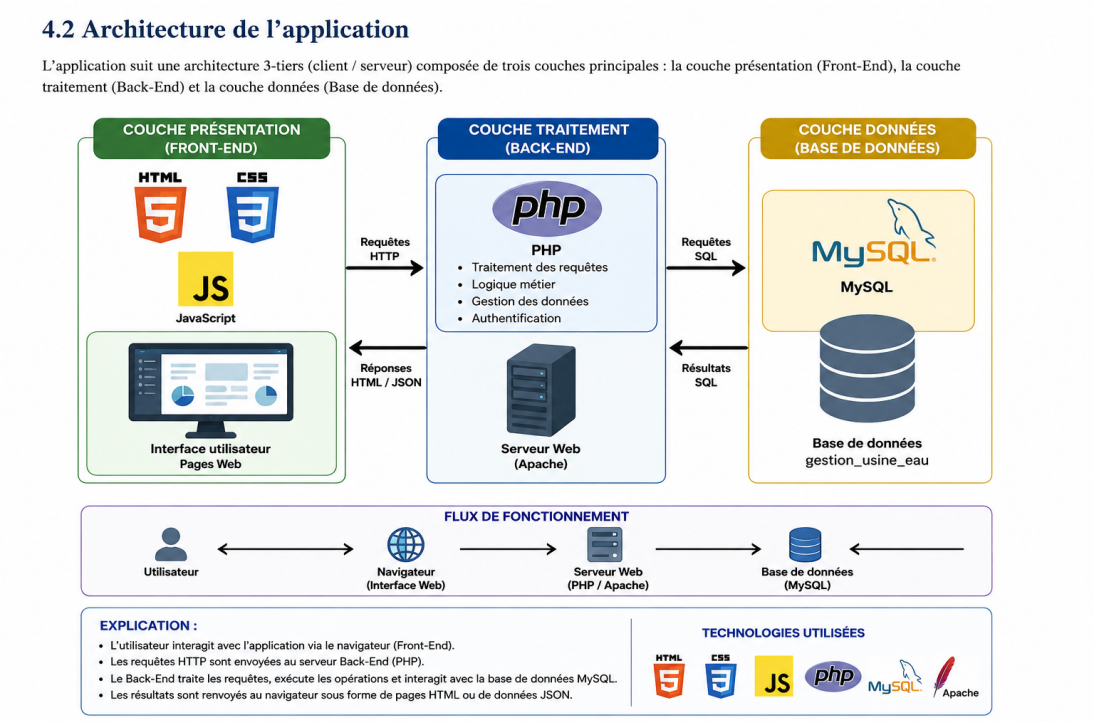



Figure 8: Architecture technique de l'application

1. **Couche Présentation (Front-End) :** Cette couche représente les interfaces graphiques visibles par l'utilisateur. C'est le point d'interaction entre le système et l'utilisateur final. Elle a été développée avec **HTML, CSS et JavaScript** pour assurer une navigation fluide et une présentation claire des données issues du serveur.
2. **Couche Traitement (Back-End) :** Cette couche constitue le cœur de l'application, contenant toute la logique fonctionnelle (logique métier). Elle est développée en **PHP** et assure le lien entre l'interface utilisateur et la base de données. Elle permet notamment :
   - Le traitement des requêtes envoyées par le client.
   - La gestion des droits d'accès des utilisateurs.
   - Le calcul automatique des factures et la mise à jour dynamique du stock.

3. **Couche Données :** Cette couche correspond au serveur de base de données **MySQL**. Elle assure le stockage permanent, la sécurité et l'intégrité des informations relatives aux clients, produits, commandes, factures et paiements. Les données y sont organisées selon le schéma relationnel défini dans le MLD.

**Principe de fonctionnement :** Le client (Front-End) envoie des requêtes via le protocole HTTP. Le serveur (Back-End) traite ces requêtes en interrogeant la base de données si nécessaire, puis renvoie une réponse formatée au client pour affichage.

## 5.3 Description des principales interfaces

Dans cette section, nous présentons les principales interfaces graphiques de notre application. Ces interfaces ont été conçues de manière simple et intuitive pour faciliter la manipulation du système par les différents utilisateurs.

### 5.3.1 Interface de Gestion du Stock

Cette interface constitue l'outil principal pour le responsable du stock afin de suivre l'état des produits en temps réel et d'assurer la disponibilité des ressources de l'usine.

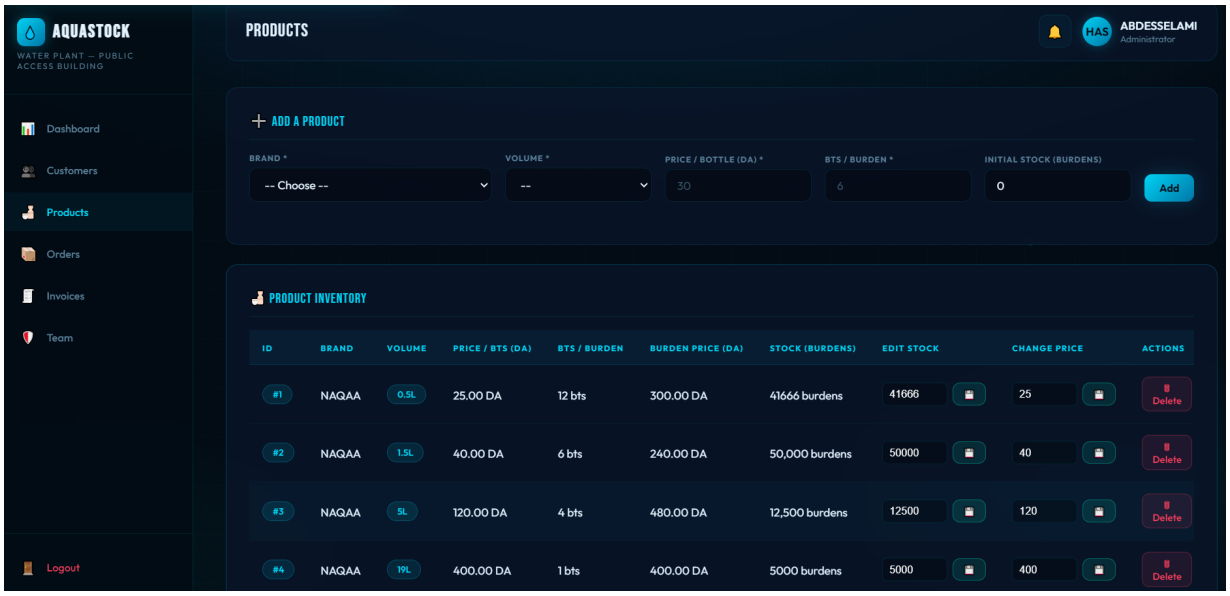



Figure 9: Interface de gestion du stock de l'usine

**Fonctionnalités clés de l'interface :** L'interface de gestion du stock offre un ensemble de fonctionnalités essentielles, notamment :

- **Ajouter un produit :** Permet d'enregistrer un nouveau produit dans la base de données en saisissant son nom, son prix unitaire et sa quantité initiale.
- **Modifier les informations :** Offre la possibilité de mettre à jour les données d'un produit existant (changement de prix ou ajustement manuel du stock).
- **Supprimer un produit :** Permet de retirer définitivement un produit de la liste en cas d'arrêt de sa production ou de son conditionnement.

- **Consulter le stock disponible :** Présente un tableau de bord clair affichant la liste globale des produits avec leurs quantités en temps réel pour éviter les ruptures de stock.

### 5.3.2 Interface de Gestion des Clients

Cette interface permet au responsable d'administrer efficacement le fichier client de l'usine, ce qui constitue une étape indispensable avant l'enregistrement de toute nouvelle commande ou facturation.

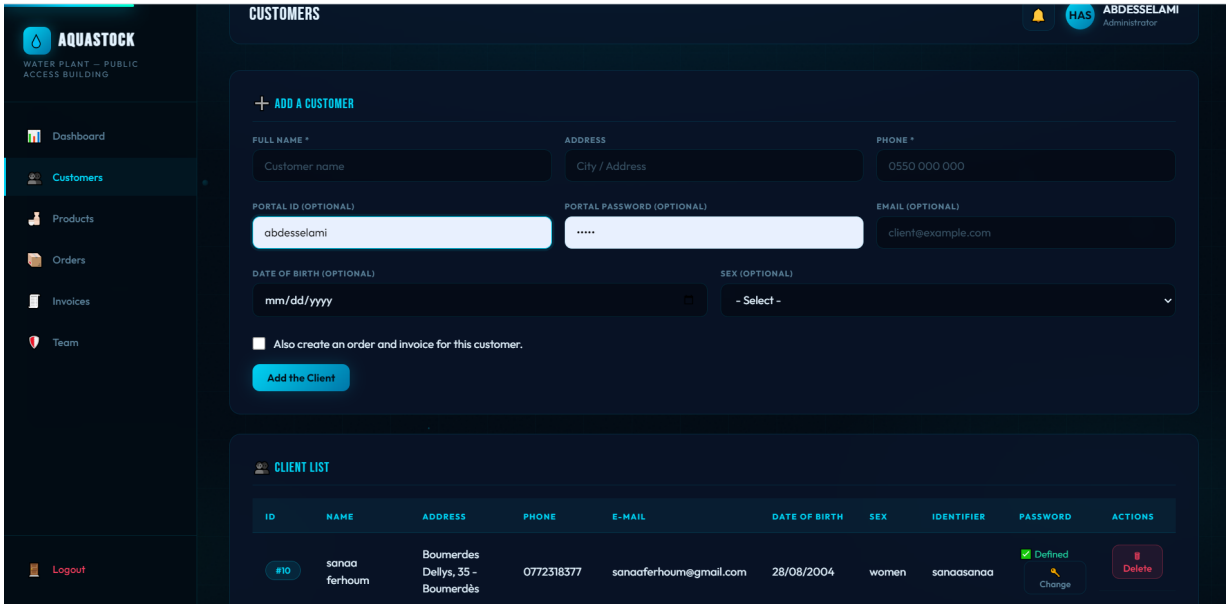



Figure 10: Interface de gestion des clients

**Fonctionnalités clés de l'interface :** L'interface de gestion des clients centralise l'ensemble des opérations nécessaires au suivi des partenaires commerciaux, notamment :

- **Ajout et enregistrement :** Permet d'intégrer un nouveau client dans le système en saisissant ses coordonnées de base.
- **Gestion des informations personnelles :** Assure la mise à jour et le suivi des données d'identification essentielles pour la livraison et la facturation, telles que :
  - **Le nom :** Le nom complet ou la raison sociale de l'entreprise/client.
  - **L'adresse :** L'adresse physique exacte pour l'organisation des livraisons de lots d'eau minérale.
  - **Le téléphone :** Le numéro de contact direct pour assurer un suivi logistique rapide et fiable.
- **Consultation et modification :** Offre une vue d'ensemble sur le répertoire des clients inscrits avec la possibilité de modifier ou de corriger leurs informations en temps réel.

### 5.3.3 Interface de Gestion de la Facturation

Cette interface représente un élément central et névralgique de notre application **AQUASTOCK**. Elle permet au responsable de suivre la situation financière globale de l'usine, de superviser les montants facturés par client (Payés, En attente), ainsi que de consulter l'historique détaillé de chaque transaction commerciale.

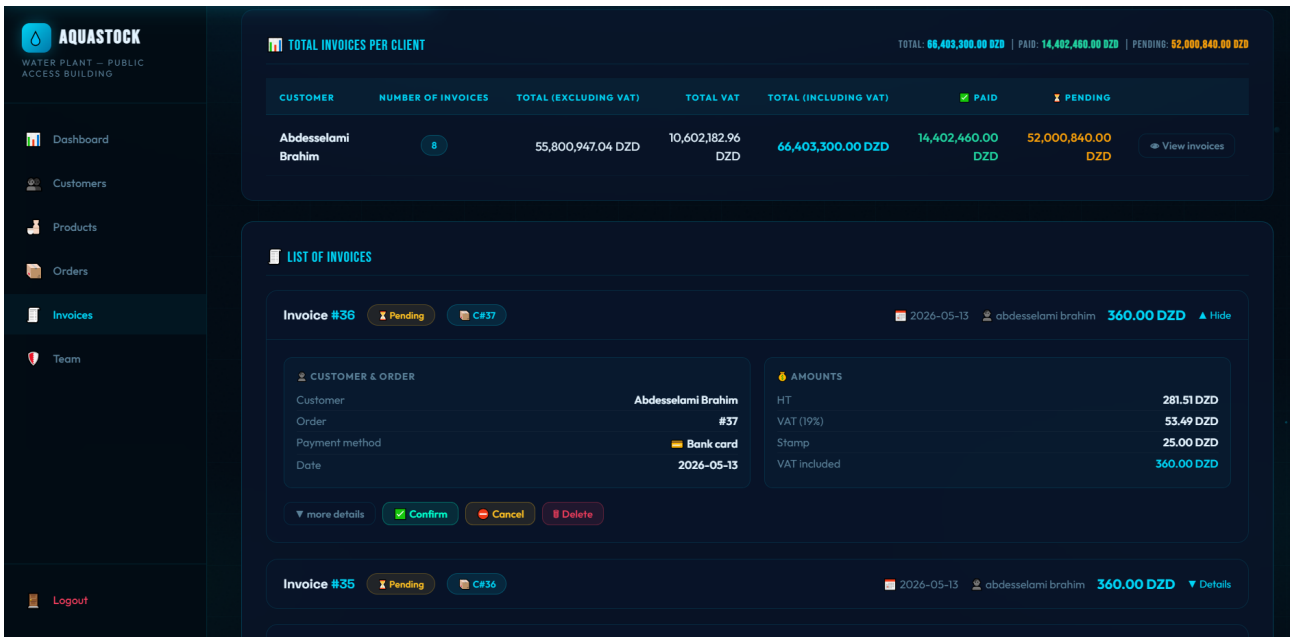



Figure 11: Interface globale de facturation et de suivi des paiements (AQUASTOCK)

**Analyse et Fonctionnalités de l'interface :** L'interface se divise en plusieurs zones clés hautement fonctionnelles :

- **Tableau de bord statistique (Total Invoices per Client) :** En tête de page, le système affiche un récapitulatif financier complet pour un client sélectionné. Il met en évidence le nombre total de factures générées, le montant global Hors Taxes (HT), la TVA cumulée, ainsi que le montant global Toutes Taxes Comprises (TTC) en spécifiant clairement les sommes déjà **payées** (Paid) et celles toujours **en attente** (Pending).
- **Liste et détails des factures (List of Invoices) :** Chaque facture est présentée sous forme de bloc dépliant (Accordion) facilitant la navigation :
  - **Informations Client & Commande :** Elle associe de manière stricte le nom du client (ex: Abdesselami Brahim), le numéro de la commande liée (#37), le mode de paiement (ex: Bank card) et la date d'émission.
  - **Calcul automatisé des montants (Amounts) :** Le système applique automatiquement les règles financières algériennes : calcul du montant **HT**, application de la **TVA à 19%**, ajout du **Droit de Timbre** (Stamp à 25.00 DZD) pour générer instantanément le montant net à payer en Dinars Algériens (**DZD**).
- **Actions de gestion (Workflow) :** Pour chaque facture, l'utilisateur dispose de boutons d'action rapides pour **Confirmer** (Confirm) le règlement, **Annuler** (Cancel) la transaction ou **Supprimer** (Delete) l'élément de l'historique.

## 5.4 Implémentation de la base de données

Après avoir défini le Modèle Logique de Données (MLD), nous avons procédé à l'implémentation physique de la base de données nommée `gestion_stock_eau` au niveau du Système de Gestion de Base de Données Relationnelle (SGBDR) **MySQL**, en utilisant l'interface d'administration **phpMyAdmin**.

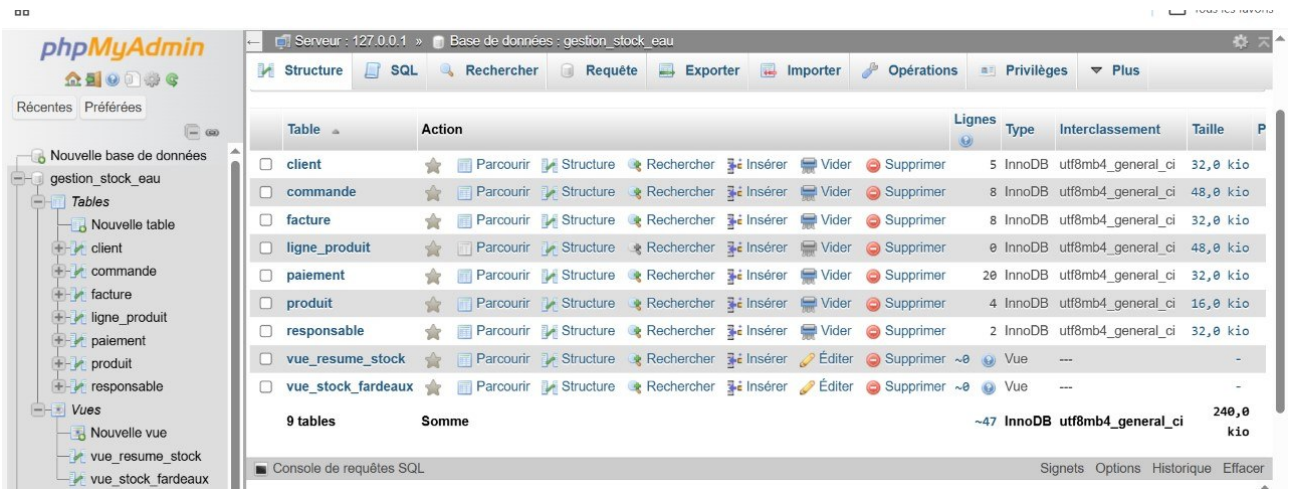



Figure 12: Structure de la base de données `gestion_stock_eau` sous phpMyAdmin

**Structure et composants de la base de données :** Comme l'illustre la figure 12, la base de données est structurée à l'aide du moteur de stockage performant **InnoDB**, qui garantit la gestion des clés étrangères (intégrité référentielle) et la sécurité des transactions. Elle se compose globalement de 9 éléments physiques répartis comme suit :

1. **Les Tables Physiques (Tables) :** Au nombre de 7 tables, elles correspondent strictement aux entités définies lors de la phase de conception :
   - `client` : Stocke les coordonnées des clients (5 enregistrements insérés).
   - `commande` : Contient l'en-tête des commandes clients (8 enregistrements insérés).
   - `facture` : Assure le suivi des factures générées (8 enregistrements insérés).
   - `ligne_produit` : Table d'association gérant le détail des produits inclus dans chaque commande.
   - `paiement` : Enregistre les transactions financières et les montants réglés (20 enregistrements insérés).
   - `produit` : Contient le catalogue des produits (lots d'eau) et les quantités en stock (4 enregistrements insérés).
   - `responsable` : Gère les accès des utilisateurs/administrateurs du système (2 enregistrements insérés).
2. **Les Vues Virtuelles (Vues / Views) :** Une attention particulière a été portée à l'optimisation des requêtes à travers la création de 2 vues virtuelles pour faciliter les calculs et l'affichage rapide des rapports au niveau du Front-End :
   - `vue_resume_stock` : Permet d'obtenir une synthèse claire et instantanée de l'état global du stock disponible.
   - `vue_stock_fardeaux` : Conçue spécifiquement pour afficher et suivre le stock calculé en unités de fardeaux (lots d'eau minérale).

## 5.5 Sécurité et gestion des accès

La sécurité est un critère de qualité fondamental et non négligeable pour notre application **AQUASTOCK**. Étant donné que le système manipule des données sensibles liées à l'usine (informations financières, factures, données clients et niveaux de stock), plusieurs mécanismes de protection avancés ont été implémentés au niveau du code source (Back-End) et de la base de données.

**1. Mécanisme d'authentification et sécurité des mots de passe :** Chaque utilisateur (administrateur ou agent) doit obligatoirement s'authentifier à l'aide d'un identifiant unique (login) et d'un mot de passe.

- **Le problème du stockage en clair :** Stocker les mots de passe sous forme de texte brut dans la table `responsable` présente un risque critique en cas de fuite ou d'accès non autorisé à la base de données.
- **La solution technique implémentée :** Pour garantir une confidentialité absolue, nous avons utilisé un mécanisme de **hachage cryptographique** via la fonction native de PHP `password_hash()`. Nous avons opté pour l'algorithme **BCRYPT**. Contrairement au chiffrement classique, le hachage est une fonction à sens unique (irréversible). Lors de l'authentification, le mot de passe saisi est haché à son tour puis comparé à l'empreinte stockée grâce à la fonction `password_verify()`. De plus, cet algorithme intègre nativement un sel (salt) aléatoire, ce qui protège le système contre les attaques par dictionnaire et les attaques par tables de correspondance (Rainbow Tables).

**2. Protection contre les Injections SQL (SQL Injection) :** L'injection SQL est l'une des failles les plus critiques et les plus courantes dans les applications web utilisant des bases de données relationnelles.

- **Le concept de l'attaque :** Elle consiste pour un attaquant à injecter du code SQL malveillant à travers les champs de saisie de l'application (comme les formulaires de connexion ou de recherche). Si les données ne sont pas vérifiées, le serveur interprète ces entrées comme des instructions SQL directes, permettant à l'attaquant de contourner l'authentification, de modifier ou de vider la base de données.
- **La sécurité mise en place :** Pour neutraliser complètement cette menace, nous avons banni la concaténation directe des variables dans nos requêtes SQL. Nous avons systématiquement utilisé les **requêtes préparées** (Prepared Statements) fournies par l'extension **PDO** (PHP Data Objects). Avec cette méthode, la structure de la requête SQL est compilée à l'avance par le serveur, et les données utilisateurs sont transmises séparément en tant que simples paramètres (Binding parameters). Le SGBD sépare ainsi strictement les instructions exécutables des données textuelles, rendant toute tentative d'injection inoffensive.

**3. Protection contre les attaques XSS (Cross-Site Scripting) :** Les attaques par script intersites (XSS) ciblent principalement l'interface utilisateur (Front-End) et l'intégrité du navigateur des clients.

- **Le concept de l'attaque :** Une faille XSS se produit lorsque l'application inclut des données fournies par l'utilisateur dans la page web finale sans validation ni encodage préalable. Un attaquant peut ainsi injecter des scripts malveillants (généralement en JavaScript) qui s'exécuteront dans le navigateur des autres utilisateurs légitimes, permettant le vol de sessions (cookies) ou le détournement de pages.

- **La sécurité mise en place :** Pour bloquer les attaques XSS, nous avons appliqué le principe de l'échappement et du nettoyage des données (Data Sanitization) avant chaque affichage sur le Front-End. Nous avons utilisé la fonction PHP `htmlspecialchars()`. Cette fonction convertit automatiquement les caractères spéciaux HTML critiques (tels que `<`, `>`, `&`, et `"`) en entités HTML équivalentes (`<`, `>`, etc.). De ce fait, même si un code malveillant est injecté dans le formulaire client ou produit, il sera affiché comme du simple texte inoffensif à l'écran au lieu d'être exécuté par le navigateur.

**4. Gestion des rôles et des privilèges (RBAC) :** En complément de ces protections techniques, l'application applique le modèle de contrôle d'accès basé sur les rôles (*Role-Based Access Control*) :

- **L'Administrateur** possède les privilèges complets pour superviser le système, configurer la base de données et gérer les comptes responsables.
- **Les Opérateurs (Stock / Facturation)** disposent d'un accès restreint aux seules interfaces nécessaires à l'exercice de leurs fonctions (saisie des clients, impression des factures), empêchant toute manipulation malveillante des paramètres généraux du système.

## 5.6 Tests et résultats

Une fois la phase de développement et d'implémentation terminée, nous avons procédé à une phase de tests rigoureux afin de valider la conformité du logiciel par rapport aux spécifications techniques et fonctionnelles.

**Nature des tests effectués :** Plusieurs types de tests fonctionnels ont été réalisés pour vérifier le comportement du système dans des conditions réelles d'exploitation :

- **Test d'ajout des produits** : Vérification de la bonne insertion des articles dans la table `produit` et validation des contrôles de saisie (champs obligatoires, formats des prix).
- **Test de création des factures** : Validation du processus de génération des documents, liaison automatique avec les commandes correspondantes et affichage exact des données.
- **Test de mise à jour du stock** : Contrôle du mécanisme d'ajustement automatique (décrémentation des quantités en stock en temps réel dès la validation d'une commande ou d'une facture).
- **Test des paiements** : Vérification du calcul automatisé des montants (HT, TVA 19%, Timbre fiscal) et enregistrement correct du statut (Paid / Pending) selon le mode de règlement choisi.

**Résultats obtenus :** Les résultats de ces tests se sont révélés pleinement satisfaisants. L'application **AQUASTOCK** réagit de manière stable, les temps de réponse sont optimaux, et l'intégrité des données est préservée à chaque étape. Le système fonctionne correctement et répond fidèlement à l'ensemble des besoins définis en amont dans le cahier des charges.

# Conclusion Générale

Ce projet de fin d'études, intitulé **Stock et Facturation d'une usine d'eau minérale** , nous a permis de concevoir et de réaliser l'application **AQUASTOCK**. Ce travail marque la concrétisation de notre formation en Licence Informatique (Systèmes d'Information).

## 1. Bilan du travail réalisé

Nous avons mis en place une solution informatique complète et fonctionnelle qui automatise les processus clés de l'usine :

- La modélisation rigoureuse du système à travers les diagrammes UML (Cas d'utilisation, Classes, Séquence).
- L'implémentation d'une base de données relationnelle robuste sous **MySQL** via phpMyAdmin.
- Le développement d'une interface web dynamique et moderne (Dark Mode) en utilisant **HTML, CSS, JavaScript** pour le Front-End, et **PHP** pour la logique métier du Back-End.

## 2. Apports du projet

Sur le plan personnel et académique, ce projet a été d'une grande valeur ajoutée :

- **Compétences techniques :** Approfondissement de nos connaissances en développement Full-Stack et en manipulation avancée des bases de données (notamment la création des Vues SQL).
- **Compétences professionnelles :** Initiation au travail d'équipe, à la gestion du temps, et à la modélisation de problèmes réels du secteur industriel (gestion des stocks d'eau, application de la TVA algérienne à 19%, et des droits de timbre).

## 3. Difficultés rencontrées

Durant la réalisation de ce travail, nous avons fait face à quelques défis :

- L'adaptation aux contraintes réelles de la gestion des stocks de fluides et de fardeaux d'eau au niveau de l'usine.
- La synchronisation et l'intégration continue entre le Front-End et le Back-End pour assurer la mise à jour automatique des stocks lors de la validation des factures.
- La maîtrise du temps pour finaliser la rédaction du document LaTeX parallèlement au développement de l'application.

## 4. Perspectives futures

Bien que l'application soit totalement fonctionnelle, elle reste ouverte à plusieurs améliorations futures :

- **Statistiques avancées :** Intégrer des graphes dynamiques pour analyser l'évolution des ventes et des bénéfices par période.

- **Gestion des alertes :** Ajouter un système de notifications automatiques lorsque le stock d'un produit atteint le seuil critique (Stock d'alerte).
- **Extension Mobile :** Développer une application mobile (Android) connectée à la même base de données pour permettre au responsable de suivre le stock à distance.

En conclusion, nous espérons que l'application **AQUASTOCK** répondra efficacement aux attentes de l'usine en optimisant son organisation interne, et que ce document servira de référence utile pour de futurs étudiants.

# References


[1] Apache Friends. (2024). *XAMPP Apache + MariaDB + PHP + Perl* (Version 8.2). https://www.apachefriends.org/ consulté le 04 mars 2026 à 21:12

[2] Codex Alimentarius Commission. (1981). *Standard for Natural Mineral Waters* (Codex Stan 108-1981). https://www.fao.org/fao-who-codexalimentarius/en/ consulté le 07 mars 2026 à 20:56

[3] Food and Agriculture Organization. (2009). *Food hygiene: Basic texts* (4th ed.). Food and Agriculture Organization of the United Nations.

[4] International Organization for Standardization. (2011). *Systems and software engineering — Systems and software quality models* (ISO/IEC Standard No. 25010:2011). https://www.iso.org/standard/35733.html consulté le 11 mars 2026 à 11:34

[5] Microsoft. (2025). *Visual Studio Code documentation*. https://code.visualstudio.com/docs consulté le 15 mars 2026 à 21:42

[6] Nixon, R. (2021). *Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5* (6th ed.). O'Reilly Media.

[7] Oracle Corporation. (2026). *MySQL 8.0 reference manual*. https://dev.mysql.com/doc/ consulté le 20 mars 2026 à 16:34

[8] Organisation mondiale de la Santé. (2022). *Guidelines for drinking-water quality: Small water supplies* (4th ed.). Organisation mondiale de la Santé.

[9] PHP Documentation Group. (2025). *PHP: Hypertext preprocessor official documentation*. https://www.php.net/docs.php consulté le 21 mars 2026 a 19:56

[10] Roques, P. (2018). *UML 2 par la pratique : Études de cas et exercices corrigés* (7e éd.). Éditions Eyrolles.

[11] Soutou, C. (2021). *Modélisation d'une base de données : De l'analyse des besoins au schéma relationnel* (4e éd.). Éditions Eyrolles.

[12] World Wide Web Consortium. (2023). *HTML5 - A vocabulary and associated APIs for HTML and XHTML*. https://www.w3.org/ consulté le 20 mars 2026 à 21:12