

Thème :

Application des Gestions de Stock pour ***TGCC***

***Encadré Par :*** ***Hamza Ouanzi***

***Réalisé Par : Adam Belghazi***

Rapport de Stage d'observation :

1ère année cycle d'Ingénieur :

**Génie Informatique**

# Remerciements

Au terme de ce stage d’observation, je tiens à exprimer ma profonde gratitude à toutes les personnes qui ont contribué, de près ou de loin, à son bon déroulement.

Mes remerciements les plus sincères vont tout particulièrement à **M. Hamza OUANZI**, mon encadrant au sein de l’entreprise, pour sa disponibilité, ses conseils pertinents et son accompagnement tout au long de cette expérience. Son encadrement m’a permis d’approfondir mes connaissances et de mieux comprendre les réalités du développement d’applications web en milieu professionnel.

Je remercie également l’ensemble de l’équipe de **TGCC** pour l’accueil chaleureux, la bienveillance et l’environnement de travail stimulant qu’elle m’a offert durant toute la période du stage.

Enfin, j’espère que l’application de gestion de stock développée durant ce stage répondra aux besoins exprimés et constituera une base utile pour de futures évolutions.

**Cordialement,  
Adam BELGHAZI**

# Résumé

Le sujet de mon stage d’observation portait sur le développement d’une **application web de gestion de stock** au sein de l’entreprise **TGCC**. Ce projet s’est articulé autour de plusieurs phases : une phase d’analyse des besoins, suivie d’une phase de conception, puis une phase de développement et de tests.

Encadré par **M. Hamza Ouanzi**, j’ai eu l’opportunité de mettre en pratique les connaissances acquises au cours de ma formation, tout en découvrant des outils professionnels utilisés en entreprise. Ce stage m’a permis de renforcer mes compétences techniques, mais aussi de mieux comprendre les enjeux liés à la gestion de stock dans un contexte réel.

Ce rapport retrace les différentes étapes du développement de cette application, les technologies utilisées, ainsi que les principales fonctionnalités mises en œuvre.

### Mots clés :

**1. Gestion de Stock**

* **Article** : Élément physique (produit, équipement, matière…) géré par l’application.
* **Fournisseur** : Entité externe responsable de la livraison des articles.
* **Entrée/Sortie** : Opérations de mouvement de stock (approvisionnement, consommation).
* **Inventaire** : État global et à jour des quantités disponibles en stock.

**2. Technologies**

* **Frontend** : Interface utilisateur développée avec **React JS**, facilitant la navigation et la gestion des données.
* **Backend** : Développé avec **Spring Boot**, assurant le traitement des données et la communication avec la base.
* **Base de données** : Utilisation de **MySQL** pour le stockage des données relatives aux articles, fournisseurs et mouvements.

**3. Fonctionnalités**

* **CRUD complet** : Création, lecture, modification et suppression des articles, fournisseurs et mouvements.
* **Recherche & filtre** : Pour une gestion rapide et efficace des stocks.
* **Sécurité & Authentification** : Accès restreint via une interface de connexion sécurisée.
* **Dashboard** : Vue d’ensemble dynamique sur l’état du stock et les indicateurs clés.

# Abstract

The topic of my observation internship was the development of a **web-based stock management application** for the company **TGCC**. The work was structured into several phases: a requirements analysis phase, followed by design, development, and testing.

Thanks to the support of my supervisor, **Mr. Hamza Ouanzi**, I was able to apply my academic knowledge in a real-world setting, while discovering new tools and concepts used in professional software development. This internship was a valuable opportunity to improve my technical and organizational skills.

In this report, I present in detail the different steps taken throughout the project, the technologies used, and the main features implemented in the stock management application.

### Keywords

**1. Stock Management**

* **Item**: A physical product or material tracked by the system.
* **Supplier**: External entity responsible for providing items to the stock.
* **Stock Entry/Exit**: Operations representing the movement of items in and out of inventory.
* **Inventory**: Real-time overview of available stock quantities.

**2. Technologies**

* **Frontend**: User interface developed using **React.js**, ensuring a smooth and responsive user experience.
* **Backend**: Server-side logic handled by **Spring Boot**, responsible for data processing and communication with the database.
* **Database**: **MySQL** used to store structured data related to stock, suppliers, and operations.

**3. Features**

* **CRUD Operations**: Complete management (Create, Read, Update, Delete) of items and suppliers.
* **Search and Filter**: Advanced tools for locating and organizing stock records.
* **Authentication**: Secure login system for authorized access to the application.
* **Dashboard**: Overview of stock statistics and key performance indicators.

# Liste des abréviations

**1. Technologies & Développement**

* **HTML** : HyperText Markup Language
* **CSS** : Cascading Style Sheets
* **JS** : JavaScript
* **React.js** : Bibliothèque JavaScript pour construire des interfaces utilisateur
* **Spring Boot** : Framework Java pour le développement backend rapide
* **SQL** : Structured Query Language
* **CRUD** : Create, Read, Update, Delete (Opérations de base sur les données)
* **UML** : Unified Modeling Language
* **MCD** : Modèle Conceptuel de Données (Méthodologie Merise)
* **JSON** : JavaScript Object Notation (Format léger d’échange de données)
* **API** : Application Programming Interface

**2. Composants du Projet**

* **Article** : Élément ou produit géré dans le stock
* **Fournisseur** : Entité externe fournissant des articles
* **Entrée/Sortie** : Mouvement d’un article vers ou depuis le stock
* **DB / BD** : Database / Base de Données
* **ID** : Identifiant unique associé à un article ou utilisateur
* **Dashboard** : Interface de synthèse affichant les indicateurs clés du stock

**3. Sécurité et Accès**

* **HTTPS** : HyperText Transfer Protocol Secure
* **JWT** : JSON Web Token (pour l’authentification sécurisée)
* **Auth** : Authentication (Processus de connexion)

**4. Gestion & Statistiques**

* **Qte** : Quantité
* **Stock min.** : Stock minimum (seuil d’alerte)
* **Mvt** : Mouvement (opération d’entrée ou sortie de stock)

# Introduction Générale

Dans le cadre de ma formation à l’ENSA de Tétouan, j’ai eu l’opportunité d’effectuer un stage d’observation au sein de l’entreprise TGCC, acteur majeur du secteur du BTP au Maroc. Ce stage avait pour objectif de me permettre de découvrir le monde professionnel tout en mettant en pratique les compétences acquises durant ma formation.

Durant cette période, j’ai travaillé sur le développement d’une application web de gestion de stock, un outil essentiel pour assurer un suivi précis des mouvements d’articles (entrées, sorties, inventaire) et optimiser l’approvisionnement. Ce projet m’a permis d’approfondir mes connaissances en développement web, notamment dans l’architecture frontend/backend, ainsi que dans la modélisation des données et l’interaction avec une base de données relationnelle.

L’application propose une interface intuitive et responsive permettant à un utilisateur administrateur de gérer l’ensemble des opérations liées au stock : ajout, modification, suppression d’articles, enregistrement des mouvements de stock, gestion des fournisseurs, etc. Un tableau de bord présente également des indicateurs en temps réel pour faciliter la prise de décision.

Dans ce rapport, je présenterai dans un premier temps le contexte du projet et les objectifs poursuivis, avant de détailler la démarche adoptée depuis la phase d’analyse jusqu’à l’implémentation. J’expliquerai ensuite les choix techniques effectués, les difficultés rencontrées ainsi que les solutions apportées. Enfin, je décrirai les principales fonctionnalités développées et le résultat final obtenu.

Chapitre 1

Contexte général du projet

Ce chapitre présente notre **application de gestion de stock**, ses objectifs ainsi que la démarche adoptée pour sa réalisation. Nous y exposons les **fondements du projet**, le **contexte dans lequel il s’inscrit**, ainsi que les **besoins fonctionnels** qu’il vise à satisfaire au sein de l’entreprise TGCC. Cette première partie permet de mieux comprendre l’intérêt de l’application, les enjeux de gestion rencontrés en milieu professionnel, et la manière dont notre solution y répond.

## 1. Principaux concepts traités

### 1.1 Introduction

Dans un monde professionnel de plus en plus automatisé, la **gestion des stocks** représente un enjeu crucial pour les entreprises. Une bonne gestion permet non seulement de **réduire les pertes**, mais aussi d’**optimiser l’approvisionnement** et de **suivre efficacement les mouvements des articles**.  
Ce projet s’inscrit dans cette logique d’amélioration en proposant une **application web dédiée à la gestion du stock** de l’entreprise **TGCC**. Il met en œuvre plusieurs concepts fondamentaux, tels que les **systèmes d'information**, les **technologies web modernes**, la **sécurité des données**, ainsi que le **suivi en temps réel des articles** et fournisseurs.

### 1.2 Systèmes de gestion de stock existants

De nombreuses entreprises utilisent aujourd’hui des solutions de gestion de stock pour **automatiser le suivi des articles**, des mouvements, et des niveaux d’inventaire. Cependant, ces outils sont souvent **coûteux**, **complexes à déployer**, ou **inadaptés aux besoins spécifiques** d’un chantier ou d’un département donné.  
Il devient donc pertinent de concevoir un système **simple, personnalisé et facile à maintenir**, capable de répondre aux exigences de l’entreprise tout en restant **léger et évolutif**.

### 1.3 Technologies web modernes (Frontend / Backend)

Le développement de cette application web repose sur deux volets complémentaires :

* Le **frontend**, qui correspond à l’interface utilisateur visible (développée avec **React.js**, HTML, CSS, etc.),
* Et le **backend**, qui gère la logique métier, les accès à la base de données et les traitements côté serveur (implémenté en **Spring Boot** avec une base de données **MySQL**).

Pour faciliter le développement et le test local, l’environnement **Spring Boot CLI** et **Node.js** ont été utilisés, avec intégration de **Axios** pour la communication entre frontend et backend.

### 1.4 Sécurité des données

La sécurité est un aspect essentiel, en particulier pour les **données critiques** liées aux stocks et aux fournisseurs. Il est impératif d’éviter toute perte, modification non autorisée ou fuite d’information.  
Des techniques de sécurité ont été mises en œuvre :

* **Validation des formulaires côté frontend et backend**,
* **Protection contre les injections SQL**,
* **Authentification par identifiants**,
* **Gestion des droits d’accès** selon le rôle de l’utilisateur (ex. : admin uniquement).

## 2. Présentation du projet

### 2.1 Problématique

Dans de nombreuses entreprises, la gestion manuelle des stocks reste une tâche chronophage, sujette aux erreurs humaines, et souvent mal optimisée. L’absence d’un système centralisé rend difficile le **suivi en temps réel** des articles, la **traçabilité des mouvements**, et la **gestion des fournisseurs**.  
Face à ces limites, il devient essentiel de proposer une solution numérique capable de **centraliser toutes les informations liées au stock**, de **sécuriser les accès**, et de **simplifier les opérations quotidiennes de gestion**.

### 2.2 Description du projet

Ce projet a pour but de développer une **application web de gestion de stock** destinée à l’usage interne de l’entreprise **TGCC**.  
Elle permet notamment :

* D’ajouter, modifier et supprimer des **articles** et des **fournisseurs**,
* D’enregistrer les **mouvements de stock** (entrées/sorties),
* De suivre en temps réel les **quantités disponibles**,
* De visualiser des **statistiques globales** via un tableau de bord.

L’administrateur peut ainsi gérer efficacement le stock, tandis que les utilisateurs autorisés peuvent consulter les informations essentielles via une **interface claire et responsive**.

### 2.3 Objectifs généraux et spécifiques

**Objectif général :**  
Mettre en place une **application web sécurisée, performante et ergonomique** pour la gestion du stock et le suivi des mouvements d’articles au sein de TGCC.

**Objectifs spécifiques :**

* Concevoir une **interface intuitive** pour l’administrateur.
* Implémenter une **base de données relationnelle** pour structurer les données liées aux articles, fournisseurs et mouvements.
* Mettre en place un **système d’authentification** pour sécuriser l’accès.
* Fournir un **dashboard dynamique** affichant les principaux indicateurs de stock (quantités, seuils, alertes, etc.).
* Faciliter la **recherche et le filtrage** d’articles grâce à une interface optimisée.

### Conclusion

Ce chapitre a permis de poser les bases du projet, en exposant **le contexte professionnel**, **la problématique rencontrée**, ainsi que **les objectifs visés**.  
Cette présentation permet de mieux appréhender les **choix techniques et fonctionnels** qui seront abordés dans les chapitres suivants, depuis la conception de la base de données jusqu’à la mise en œuvre de l’application web.

Chapitre 2

Analyse & Conception

Ce chapitre décrit l’architecture de l’application, les choix technologiques, la modélisation des données ainsi que les principales fonctionnalités. Il présente aussi la conception technique, notamment la base de données et les aspects de sécurité

### Introduction

Ce chapitre présente l’**analyse des besoins** et la **conception détaillée** de notre application de gestion de stock développée dans le cadre de mon stage au sein de **TGCC**.  
Après avoir défini les objectifs du projet dans le chapitre précédent, nous abordons ici une phase essentielle, au cours de laquelle nous **structurons la solution technique** pour répondre aux **exigences fonctionnelles** (gestion des articles, fournisseurs, mouvements de stock…) et **non fonctionnelles** (sécurité, accessibilité, performance, etc.).

Nous commencerons par l’**analyse des besoins** à travers des **diagrammes UML** (cas d’utilisation, séquences), afin de modéliser les interactions entre les acteurs (administrateur, utilisateur interne) et le système.  
Ensuite, nous détaillerons l’**architecture logicielle retenue** (choix des technologies, schéma de la base de données), ainsi que les **maquettes des interfaces** utilisateur.

Cette étape de **conception** permet d’établir une base solide pour le développement, en veillant à ce que l’application soit **sécurisée, évolutive et intuitive**. Les décisions prises dans cette phase influenceront directement la qualité de la réalisation technique, que nous aborderons dans le chapitre suivant.

## 1. Architecture de l’application

### 1.1 Choix technologiques

Pour la réalisation de cette application web de gestion de stock, nous avons sélectionné des technologies modernes et robustes, couramment utilisées en entreprise :

* **Frontend** :  
  L’interface utilisateur a été développée à l’aide de **React.js**, combiné à **HTML**, **CSS** et **JavaScript**, afin d’offrir une expérience fluide, responsive et facile à utiliser.
* **Backend** :  
  La logique métier a été implémentée avec **Spring Boot**, un framework Java performant, qui facilite la gestion des requêtes, la connexion à la base de données et la mise en place d’une architecture REST.
* **Base de données** :  
  Nous avons utilisé **MySQL** comme système de gestion de base de données relationnelle, pour sa fiabilité et sa compatibilité avec Spring Boot. Les schémas et les tables ont été conçus pour refléter fidèlement les entités manipulées dans l’application (articles, fournisseurs, mouvements…).
* **Environnement de développement** :  
  Le projet a été développé localement à l’aide de **IntelliJ IDEA** pour le backend Spring Boot, et **Visual Studio Code** pour le développement du frontend React. L’outil **Postman** a été utilisé pour tester les API, et **Axios** a été intégré côté React pour assurer la communication avec le serveur.

### 1.2 Modélisation des données

Cette section présente la **modélisation UML** du système. Elle s’appuie sur :

* des **diagrammes de cas d’utilisation** pour décrire les interactions entre les utilisateurs (ex. : administrateur) et le système,
* des **diagrammes de séquence** pour illustrer le déroulement des opérations (ajout d’un article, enregistrement d’un mouvement de stock, etc.).

Ces modèles ont servi de base pour la **conception d’un schéma relationnel fiable**, garantissant la **cohérence des données**, tout en permettant une **évolution future** du système sans perte d’intégrité.

#### **1.2.1 Diagramme de cas d’utilisation**

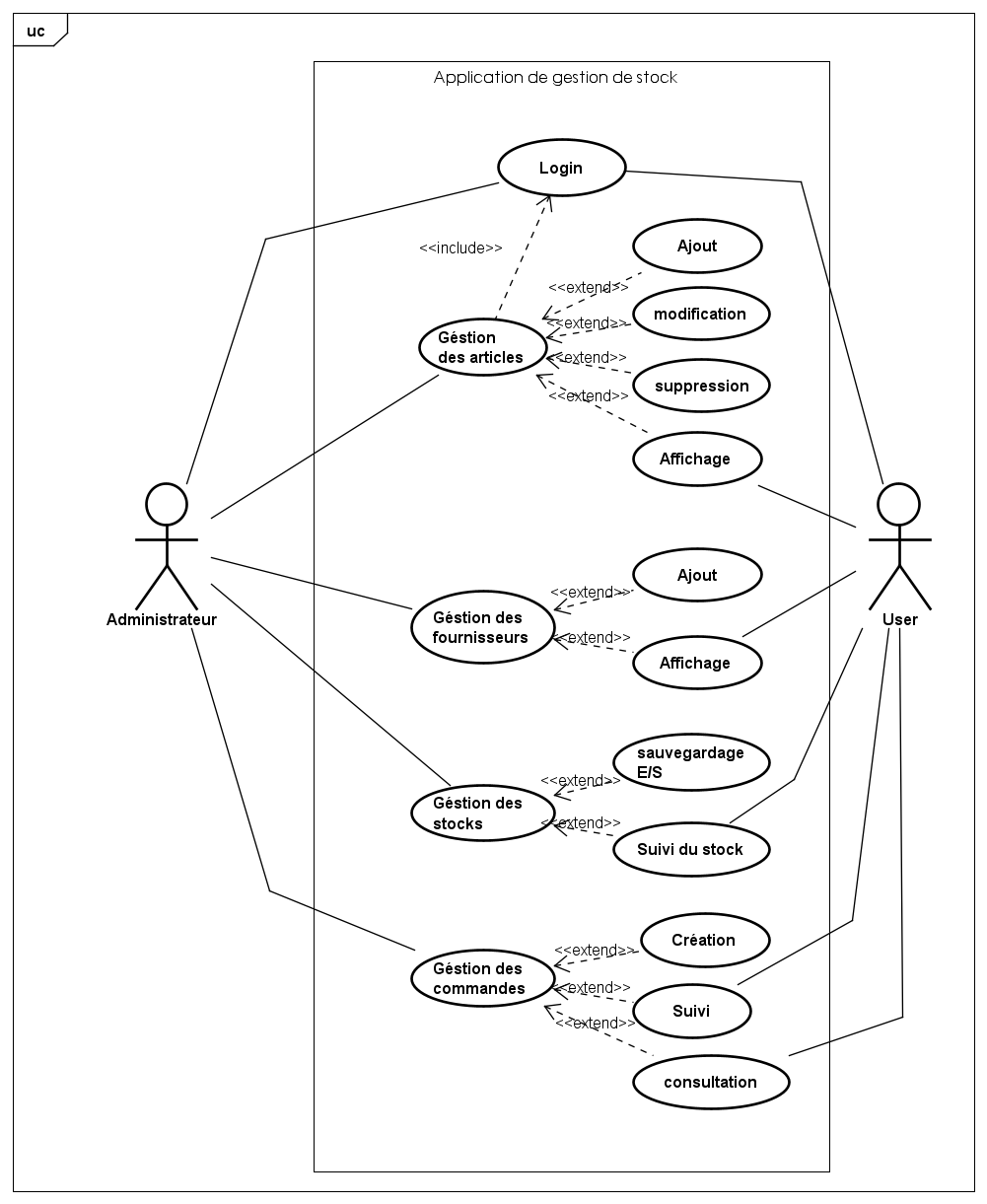
Le diagramme de cas d'utilisation permet de formaliser les interactions entre les acteurs (administrateur, User, etc.) et notre application de gestion de stock. Il met en lumière les **fonctionnalités clés du système**, tout en **clarifiant les droits d'accès** pour chaque profil.

Figure 1 : diagramme de cas d’utilisation

Ce diagramme de cas d'utilisation présente les fonctionnalités clés d'une application web de gestion de stock chez TGCC, avec deux acteurs principaux : l’Administrateur et l’Utilisateur. L’Administrateur peut gérer les articles, fournisseurs, stocks et commandes, tandis que l’Utilisateur a un accès limité à la consultation. Les relations « extend » montrent que certaines actions, comme l’ajout ou la suppression, dépendent de cas d’utilisation principaux. Ce schéma clarifie les interactions entre les utilisateurs et le système.

#### 1.2.2 Diagramme de séquence

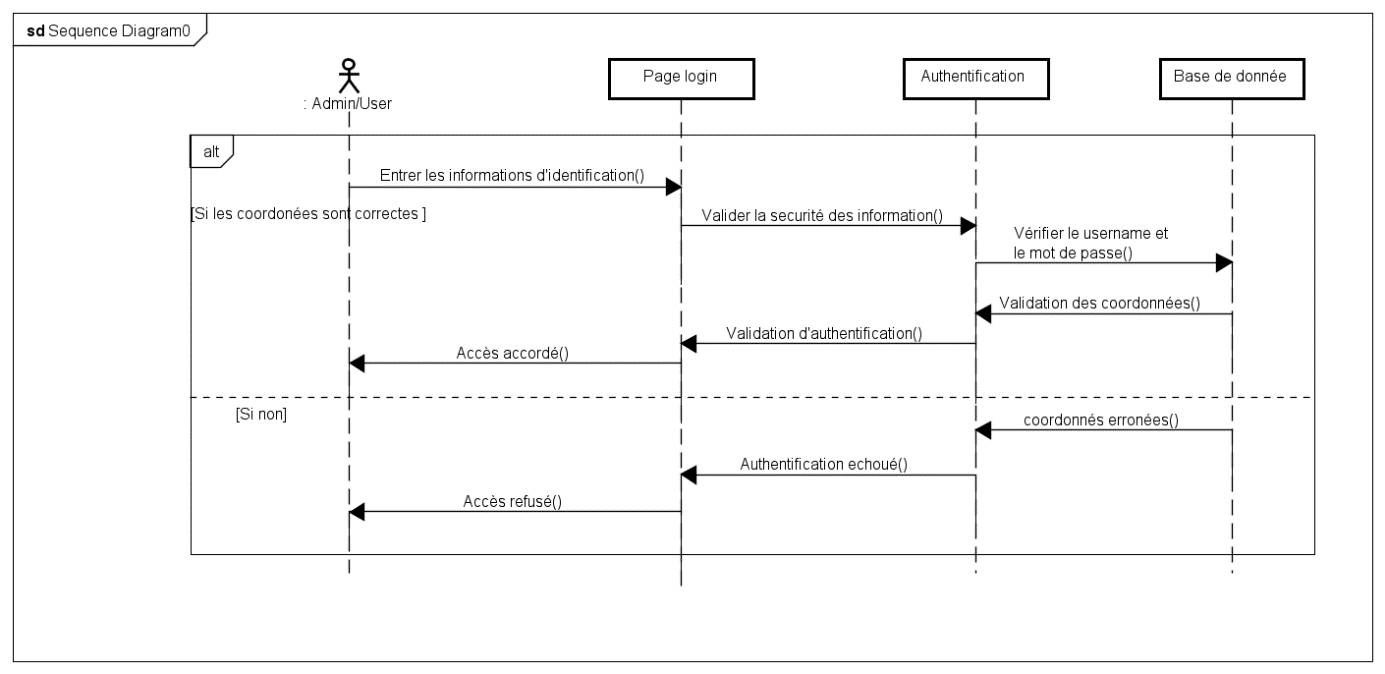
Le diagramme de séquence illustre les interactions dynamiques entre les acteurs (Administrateur, Utilisateur) et les composants principaux de l’application de gestion de stock, dans le cadre d’un scénario précis (ex : ajout d’un article, création d’une commande, suivi de stock). Contrairement au diagramme de cas d’utilisation, qui reste statique, ce diagramme met en évidence l’ordre chronologique des messages échangés entre les entités, ainsi que le comportement interne du système lors de l’exécution d’un processus métier.

Figure 2 : diagramme de séquence (Login)

Ce diagramme décrit le processus d'authentification dans l’application de gestion de stock TGCC, couvrant les accès pour l’administrateur et l’utilisateur. Le processus débute par la saisie des identifiants (nom d'utilisateur et mot de passe), suivie d’une vérification auprès de la base de données. Deux cas de figure sont prévus :

● Si les identifiants sont valides, l’accès est autorisé et l’utilisateur est redirigé vers son interface respective (administrateur ou utilisateur).

● En cas d’identifiants incorrects, l’authentification échoue et l’accès est refusé.

Pour sécuriser cette étape critique, plusieurs mesures sont à mettre en œuvre : la validation des champs saisis (filtrage des caractères spéciaux), l’utilisation de requêtes SQL paramétrées pour éviter les injections, et le chiffrement des mots de passe via des algorithmes sécurisés. Ces protections assurent la confidentialité des accès et l’intégrité du système.

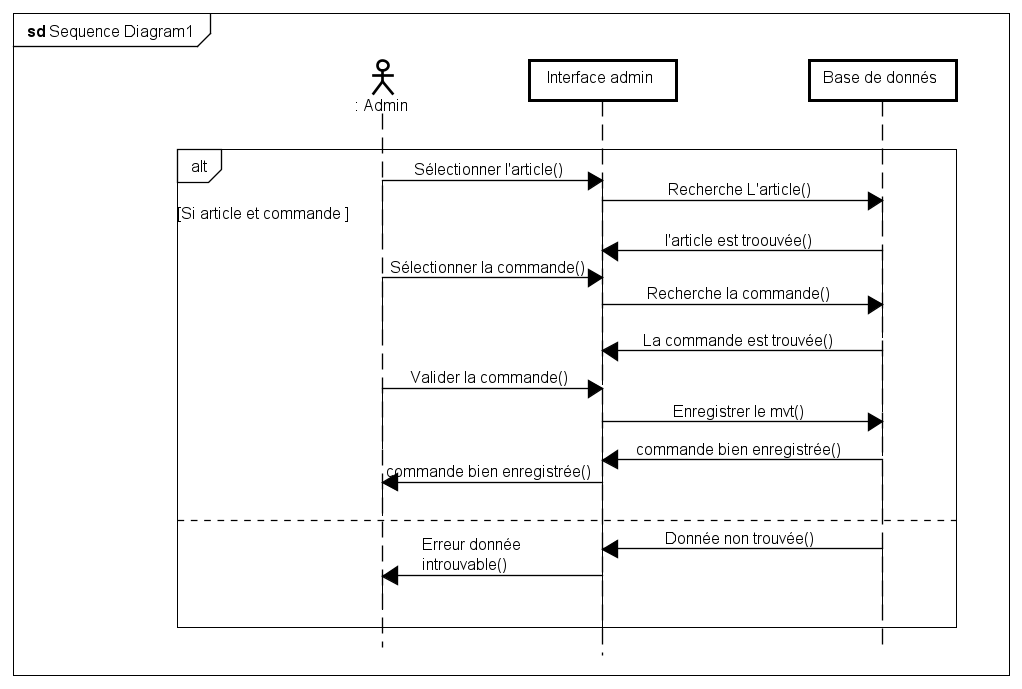


Figure 3 : diagramme de séquence (Ajout de commande)

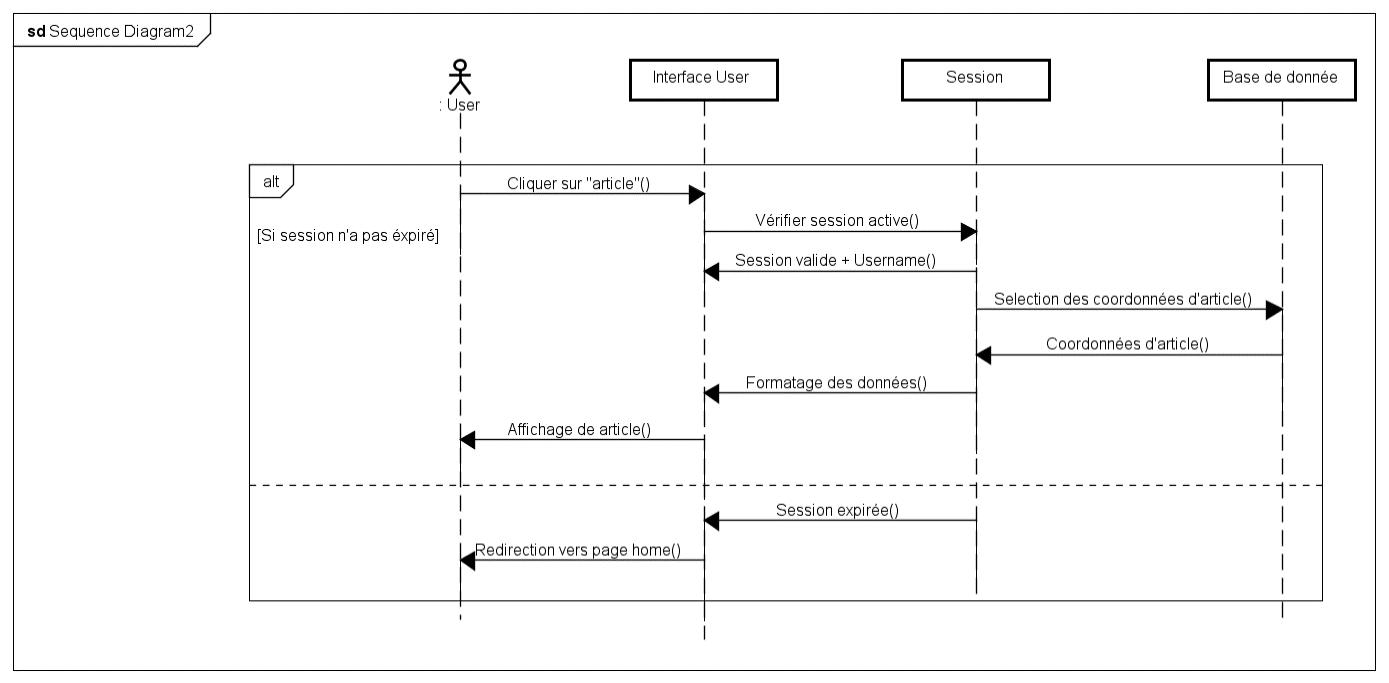
Ce diagramme illustre le processus de gestion des commandes dans une interface d'administration. Le flux débute par la sélection d’un article et d’une commande par l’administrateur. Si les deux éléments sont trouvés dans la base de données, l’interface permet ensuite la validation de la commande. Cette validation déclenche l’enregistrement du mouvement (mvt) dans la base, avec une confirmation de succès transmise à l’utilisateur. En revanche, si l’article ou la commande ne sont pas retrouvés, un message d’erreur approprié est affiché, indiquant que les données sont introuvables.

Figure 4: diagramme de séquence (Affichage d'article)

Ce diagramme illustre le processus d'affichage des informations d’un article dans le système de gestion de stock. L'utilisateur clique sur l'option "article", ce qui déclenche une vérification de la session en cours. Si la session est encore active, le système récupère les coordonnées de l'article depuis la base de données. Ces données sont ensuite formatées et affichées à l'utilisateur via l'interface. En revanche, si la session a expiré, l'utilisateur est automatiquement redirigé vers la page d'accueil pour se reconnecter.

## 2 Fonctionnalités principales

### 2.1 Gestion des articles (CRUD)

La gestion des articles constitue le cœur fonctionnel de l’application de gestion de stock développée. Elle repose sur une approche CRUD (Create, Read, Update, Delete) qui permet une administration complète et structurée des éléments en stock.

L’administrateur peut :

* **Ajouter** de nouveaux articles en renseignant les informations nécessaires (nom, référence, catégorie, quantité initiale, seuil minimum, etc.) ;
* **Consulter** la liste complète des articles via un tableau interactif avec pagination et filtres (catégorie, disponibilité) ;
* **Modifier** les détails d’un article à tout moment (par exemple, ajuster la quantité ou changer le fournisseur) ;
* **Supprimer** un article en cas d’obsolescence ou de sortie définitive du stock.

Chaque action est sécurisée et validée à la fois côté client (React) et côté serveur (Spring Boot), garantissant l’intégrité des données enregistrées. Cette gestion centralisée permet de maintenir une base de données à jour et fiable, essentielle pour un suivi rigoureux des approvisionnements.

### 2.2 Gestion des mouvements de stock (entrées/sorties)

La fonctionnalité de gestion des mouvements de stock permet de tracer toutes les **entrées** (approvisionnements) et **sorties** (consommations, transferts, pertes) d’articles. Lorsqu’un mouvement est initié, l’utilisateur choisit l’article concerné, indique la nature du mouvement, la quantité, ainsi qu’un commentaire ou une référence à une commande si besoin.

Le système effectue automatiquement :

* La mise à jour du stock disponible ;
* Le contrôle des seuils critiques (stock minimum) ;
* La génération d’un enregistrement horodaté dans l’historique.

En cas de dépassement des seuils (ex. : quantité restante en dessous du stock minimum), une alerte est affichée dans le tableau de bord. Cette gestion rigoureuse améliore la visibilité sur les flux de marchandises et permet une meilleure anticipation des besoins d’approvisionnement.

### 2.3 Gestion des fournisseurs

L’application propose également un module dédié à la gestion des fournisseurs. Chaque article peut être associé à un ou plusieurs fournisseurs référencés avec leurs coordonnées (nom, email, téléphone, localisation). Le CRUD fournisseur permet à l’administrateur de :

* Créer un nouveau fournisseur,
* Modifier ses informations,
* Visualiser la liste complète avec tri et recherche,
* Supprimer un fournisseur (après vérification de non-utilisation dans les articles actifs).

Cette fonctionnalité facilite le suivi des relations commerciales et centralise les contacts nécessaires à la passation des commandes.

### 2.4 Tableau de bord et visualisation

L’application intègre un **dashboard interactif** permettant une **vue d’ensemble** de la situation du stock. Ce tableau de bord affiche :

* Le nombre total d’articles en stock,
* Le nombre d’articles en dessous du seuil critique,
* Le total des mouvements enregistrés sur une période donnée,
* Des graphiques illustrant l’évolution des entrées/sorties par mois.

Ces représentations visuelles (histogrammes, camemberts) aident les responsables à prendre des décisions rapides et éclairées, en mettant en évidence les tendances et anomalies dans la gestion du stock.

### 2.5 Sécurité et authentification

Afin de garantir la **sécurité des données sensibles**, l’application intègre un système d’authentification sécurisé par identifiants. Chaque utilisateur se connecte via une interface login, et les accès sont restreints selon le rôle :

* **Administrateur** : accès complet à toutes les fonctionnalités,
* **Utilisateur** : accès limité à la consultation des articles et mouvements.

Les mots de passe sont chiffrés, et les requêtes SQL sont protégées contre les injections. Des vérifications sont aussi effectuées sur chaque requête afin d’éviter les accès non autorisés ou les manipulations malveillantes.

## 3 Conception de la base de données (MySQL)

Dans le cadre de la conception de notre base de données, nous avons opté pour la méthode **UML (Unified Modeling Language)**, car elle offre une approche souple, moderne et parfaitement adaptée au développement orienté objet. Contrairement à Merise, qui se focalise principalement sur les systèmes d'information traditionnels avec une séparation stricte entre les niveaux conceptuel, logique et physique, UML permet une modélisation globale du système, intégrant à la fois les aspects **structurels** (classes, attributs, relations) et **comportementaux** (interactions, cas d’utilisation).

L’utilisation d’UML nous a permis de concevoir un **diagramme de classes relationnel** servant de base à la structure de notre base de données. Chaque entité métier (article, fournisseur, mouvement, utilisateur…) y est représentée sous forme de classe avec ses attributs, et reliée aux autres via des associations précisant les cardinalités. Ce modèle a ensuite été traduit en **schéma relationnel** dans MySQL, en tenant compte des types de données, des clés primaires/étrangères, ainsi que des règles d’intégrité référentielle.

L’approche UML présente plusieurs avantages dans le contexte de notre projet :

* Elle est bien intégrée aux outils modernes de développement (notamment dans les environnements Java / Spring Boot) ;
* Elle favorise une **meilleure cohérence** entre la modélisation de la base et la logique applicative ;
* Elle permet une **évolution plus fluide** du modèle, notamment en cas d’ajouts futurs de fonctionnalités.

La phase de modélisation s’est appuyée sur une **analyse approfondie des besoins métier** identifiés lors des premières étapes du projet. L’objectif était de structurer les données de manière claire, cohérente et évolutive, tout en assurant leur intégrité et leur performance à l’exécution.

Ainsi, notre base de données relationnelle MySQL a été construite à partir du **modèle UML** comme suit :

* **Entité Article** : stocke les informations sur les produits (nom, référence, quantité, seuil minimum, etc.) ;
* **Entité Fournisseur** : contient les coordonnées des fournisseurs ;
* **Entité Mouvement** : représente les entrées et sorties d’articles (type, date, quantité, remarque) ;
* **Entité Utilisateur** : permet la gestion sécurisée des accès (rôle, identifiants, mot de passe chiffré).

Chaque entité est reliée aux autres via des clés étrangères, selon les relations définies dans le diagramme UML initial. Cette organisation assure une **cohérence stricte** des données tout en permettant une gestion souple et efficace des opérations de stock au sein de l’entreprise TGCC.

### 3.1 Diagramme de classes UML

Le diagramme de classes ci-dessous constitue le **socle conceptuel** de notre base de données. Il permet de visualiser les principales entités métier, leurs attributs, ainsi que les relations entre elles. Ce modèle a servi de base pour la génération du **schéma relationnel MySQL**.

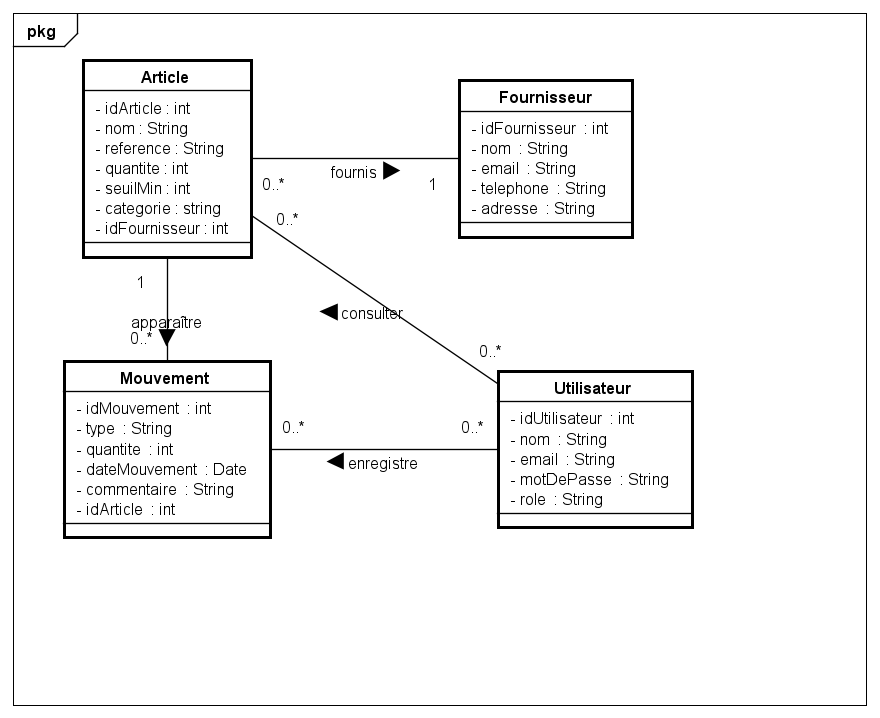


Figure 5:Diagramme de classes UML

### 3.2 Passage au modèle relationnel

À partir de ce diagramme UML, la conversion vers le schéma relationnel MySQL s’est faite de manière systématique :

* Chaque **classe** est devenue une **table**,
* Les **attributs** sont devenus des **colonnes**,
* Les **associations** ont été implémentées via des **clés étrangères**.

Les **contraintes d’intégrité** (clés primaires, clés étrangères, types de données, non-null, etc.) ont été respectées lors de l’implémentation, afin de garantir une base cohérente et performante.

Chapitre 3

Réalisation du projet

Ce chapitre présente notre application de gestion de stock, ses objectifs et notre démarche de réalisation. Nous y exposons les bases du projet et les besoins qu'il vise à satisfaire.

## 1 Environnement de développement

Le développement de l’application a été réalisé dans un environnement adapté au travail web,

Combinant des outils modernes et des technologies efficaces pour le frontend et le backend.

Cette section présente les outils principaux utilisés tout au long du projet.

## **1.1 Outils utilisés**

Pour mener à bien ce projet, plusieurs outils ont été utilisés selon les besoins spécifiques du

Développement web :

● Visual Studio Code : un éditeur de code léger et puissant, utilisé pour écrire et organiser le

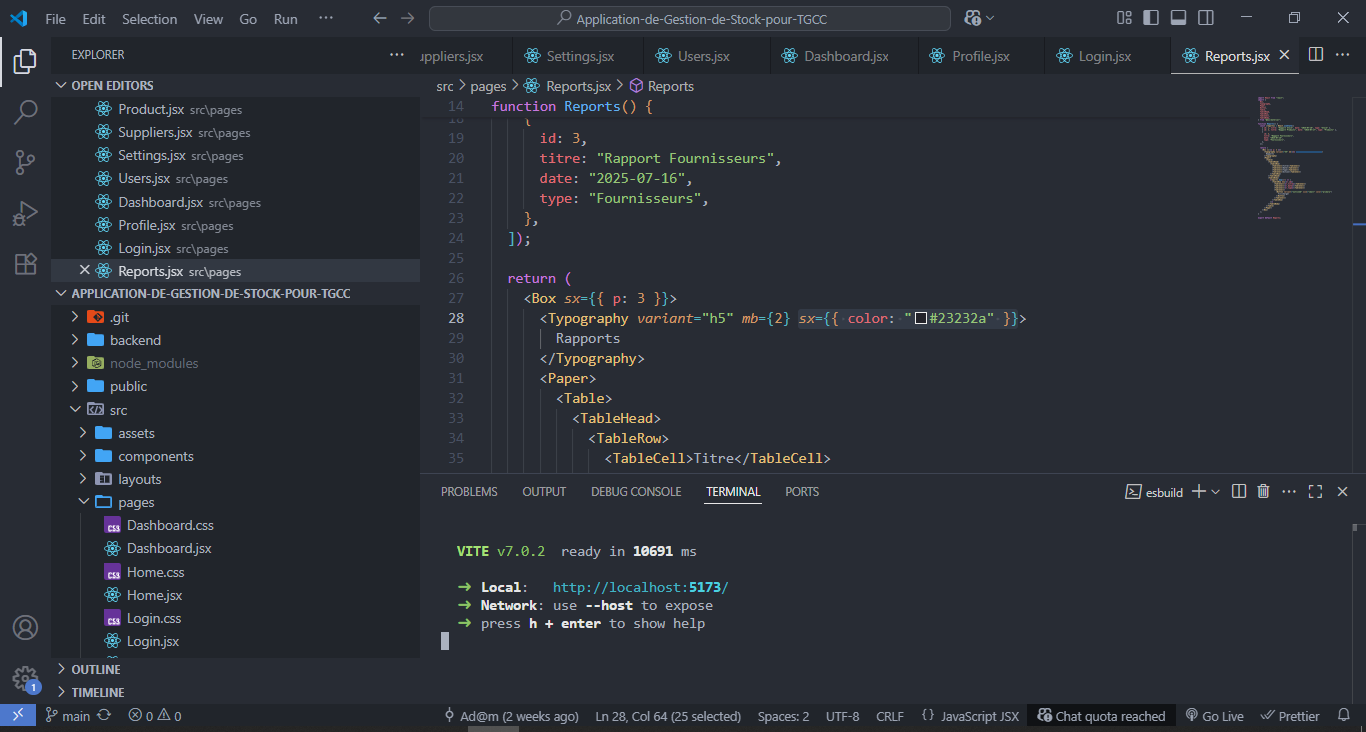
code source.

Figure 6: Interface VS Code

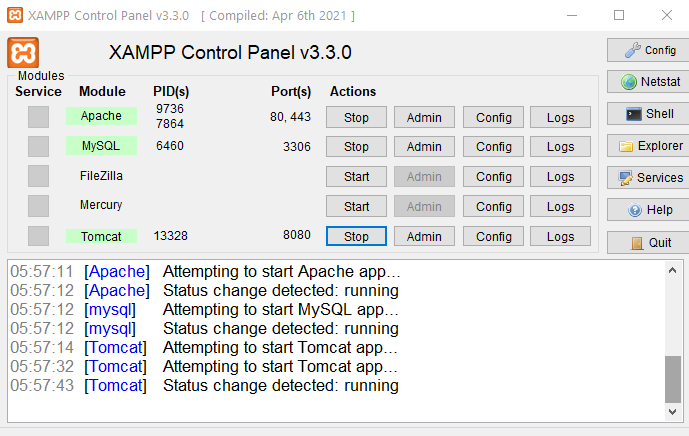
● XAMPP : environnement local de serveur (Apache, MySQL, PHP) pour tester l’application en local.

Figure 7: Interface de XAMPP

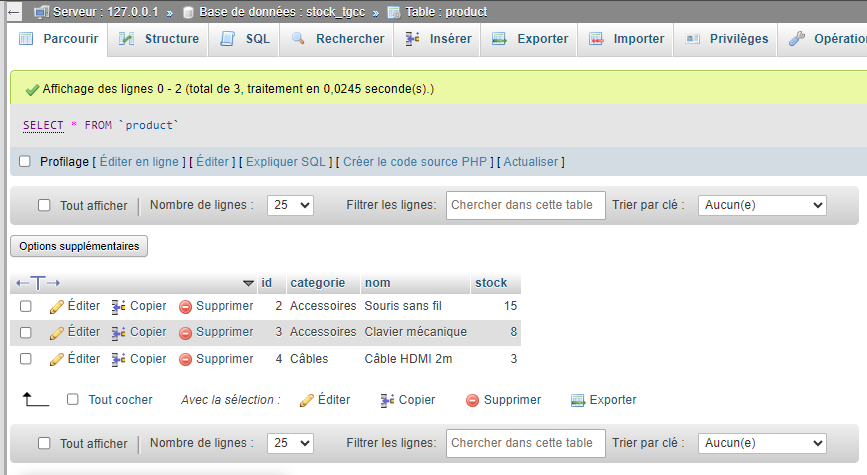
● PhpMyAdmin : outil web permettant de gérer la base de données MySQL de manière simple et visuelle.

Figure 8: Interface phpMyAdmin

● Google Chrome ou Mozilla : navigateur utilisé pour tester l’interface utilisateur.

● Git : système de gestion de version pour suivre l’évolution du code. 

Figure 9: Interface GitHub

### 1.1.1 Frontend

Le développement de la partie visible de l’application (frontend) a été réalisé en utilisant principalement **React.js**, associé à **HTML**, **CSS** et **JavaScript**. Cette combinaison de technologies a permis de concevoir une interface moderne, réactive et facile à utiliser.

* **React.js** a été le cœur du frontend, facilitant la création de composants réutilisables (tableaux, formulaires, modales, etc.) et la gestion de l’état de l’interface.
* **HTML** a servi à structurer les composants et à définir la hiérarchie du contenu.
* **CSS** a été utilisé pour assurer une présentation claire, esthétique et responsive, avec des effets de style personnalisés pour améliorer l’expérience utilisateur.
* **JavaScript**, intégré via React, a permis de gérer les interactions dynamiques (filtrage, tri, gestion d’événements, redirection conditionnelle…).
* La bibliothèque **Axios** a été intégrée pour effectuer les requêtes HTTP et assurer la communication entre le frontend et le backend via une API REST.

Grâce à cette architecture, l’interface utilisateur est fluide, intuitive et entièrement synchronisée avec les données de la base, offrant une expérience ergonomique tant pour l’administrateur que pour les utilisateurs en lecture.

### 1.1.2 Backend

Le backend de l’application a été développé en **Java**, à l’aide du **framework Spring Boot**, qui offre une structure robuste, modulaire et bien adaptée aux applications web modernes. Ce choix technologique a permis d’assurer une **bonne séparation des responsabilités**, une **sécurité accrue** et une **interaction fluide avec la base de données MySQL**.

Le backend permet notamment de :

* Gérer l’**authentification** et les **sessions utilisateurs**, avec une différenciation des rôles (administrateur, utilisateur simple) ;
* Traiter toutes les **requêtes métier** (ajout, suppression, modification, consultation d’articles, fournisseurs, mouvements, etc.) via des **contrôleurs REST** ;
* Assurer la **communication avec la base de données** à l’aide de Spring Data JPA et des entités Java mappées ;
* Valider les données côté serveur, afin d’éviter les entrées invalides ou incohérentes ;
* Protéger l’application via des mécanismes de **sécurité** (gestion des erreurs, validation des entrées, restrictions d’accès…).

Ce backend centralisé offre ainsi une plateforme fiable et évolutive, qui gère efficacement la logique métier tout en garantissant la sécurité et l’intégrité des données manipulées.

### 1.1.2 Base de données

Figure 10: Base de données

## **1.2 Méthodologie**

Pour organiser le travail et assurer une progression régulière, une méthode de type Agile a

été adoptée. Le projet a été découpé en étapes successives, chacune correspondant à une

fonctionnalité précise (gestion des stock, articles, commandes, etc.).

À chaque étape :

● Les besoins étaient analysés.

● Une implémentation partielle est réalisée et testée.

● Des ajustements étaient faits selon les résultats.

# 2 Implémentation des fonctionnalités

Cette section décrit les principales fonctionnalités implémentées dans l’application, en détaillant

leur rôle et leur fonctionnement.

## **2.1 Page de connexion**

Figure 11: Page Login

La page de connexion constitue la porte d’entrée de l’application. Elle permet à deux types d’utilisateurs d’accéder à leur espace respectif : les utilisateurs et l’administration

● L’utilisateur entre son identifiant et son mot de passe.

● Les informations sont vérifiées dans la base de données.

● En fonction du rôle (Utilisateurs ou admin), l’utilisateur est redirigé vers la page d’accueil

Correspondante.

## **2.2 Page d’accueil (Home)**

Figure 12: Page Home

La page d’accueil joue un rôle de transition après la connexion. Elle accueille l’utilisateur dans son espace personnel selon son rôle (administrateur ou utilisateur simple).

* **Pour l’administrateur** : un message de bienvenue personnalisé est affiché, ainsi qu’un aperçu des principales fonctionnalités (accès rapide au Dashboard, gestion des articles, etc.).
* **Pour l’utilisateur** : seule la consultation des articles ou mouvements de stock est disponible.
* L’interface est sobre, avec un menu latéral (ou navbar) pour faciliter la navigation entre les différentes pages.

Cette page est conçue avec React.js et assure une bonne ergonomie dès l’entrée dans l’application.

## **2.3 Page Tableau de bord (Dashboard)**

Figure 13: Page Dashboard

Le **Dashboard** est un composant essentiel pour la supervision globale du stock. Il fournit une visualisation rapide et graphique des données clés :

* Nombre total d’articles en stock ;
* Nombre d’articles en dessous du seuil critique ;
* Volume des entrées et sorties récentes ;
* Représentations graphiques (camemberts, histogrammes…) des mouvements mensuels.

Ces données sont récupérées via des requêtes vers le backend, analysées puis affichées sous forme de widgets interactifs. Ce tableau de bord permet à l’administrateur de prendre des décisions plus rapidement.

## **2.4 Page des produits (Product)**

Figure 14: Page Products

La page **Product** regroupe toutes les informations relatives aux articles stockés. Elle permet d’effectuer un **CRUD complet** sur les produits :

* **Ajout** d’un nouveau produit via un formulaire validé côté frontend et backend ;
* **Modification** des caractéristiques d’un produit existant (quantité, nom, fournisseur…)
* **Suppression** définitive d’un article non utilisé ;
* **Consultation** via un tableau interactif filtrable (nom, catégorie, seuil, disponibilité).

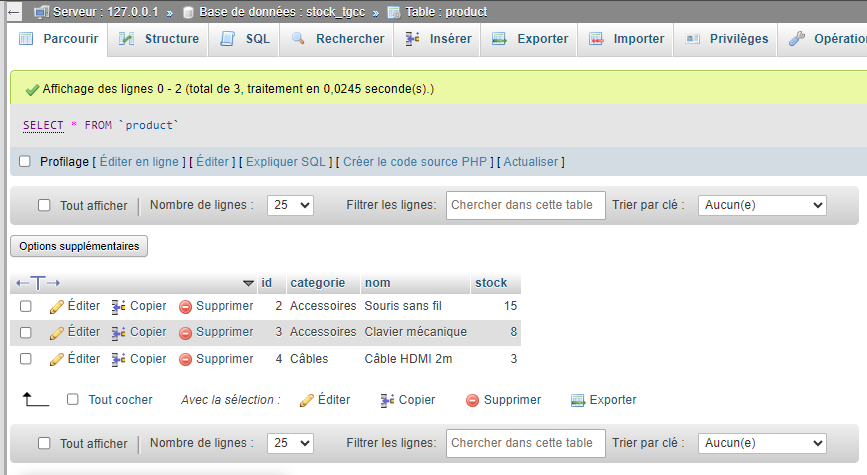
Chaque action est contrôlée et sécurisée par rôle, et les informations sont directement synchronisées avec la base MySQL via Axios et les APIs Spring Boot.

Figure 15: table of Products

## **2.5 Page des fournisseurs (Suppliers)**

Figure 16: Page Suppliers

La page **Suppliers** est dédiée à la gestion des fournisseurs dans l’application. Elle permet à l’administrateur de gérer efficacement les entités partenaires qui approvisionnent les articles en stock.

Les fonctionnalités principales incluent :

* **Ajout d’un fournisseur** avec les informations suivantes : nom, adresse, email, téléphone ;
* **Consultation** de la liste des fournisseurs existants dans un tableau triable et filtrable ;
* **Suppression** d’un fournisseur (avec vérification préalable qu’il n’est pas associé à un article actif).

Chaque fournisseur peut être lié à un ou plusieurs articles. Cette page facilite donc le suivi logistique et commercial, en centralisant toutes les données nécessaires aux commandes.

L’interface est conçue en React.js avec des appels Axios vers l’API backend Spring Boot pour toutes les opérations de gestion. Les entrées sont validées afin d’éviter les erreurs de saisie ou les doublons.

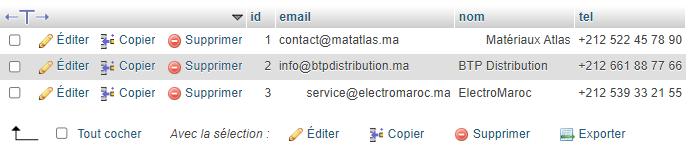


Figure 17: table of Suppliers

## **2.6 Page des projets (Project)**

Figure 18: Page Projects

La page **Project** permet de lier certains mouvements de stock à des projets ou chantiers spécifiques. Elle offre les fonctionnalités suivantes :

* Création d’un nouveau projet (nom, Client, Date Début...) ;
* Association d’un projet à une commande ou mouvement de stock ;
* Visualisation des articles consommés par projet.

Cette fonctionnalité permet un **suivi analytique** des ressources utilisées par chantier, très utile pour les entreprises du BTP comme TGCC.

## **2.7 Page des utilisateurs (Users)**

Figure 19: Page Users

La page **Users** est réservée à l’administrateur. Elle permet la gestion des comptes utilisateurs :

* Ajout d’un nouvel utilisateur avec un rôle spécifique ;
* Visualisation des utilisateurs actifs ;
* Modification ou suppression d’un compte ;
* Contrôle des accès (authentification, permissions, blocage temporaire).

La sécurité y est renforcée, avec une vérification des actions sensibles et l’enregistrement de l’activité utilisateur.

## **2.8 Page des rapports (Reports)**

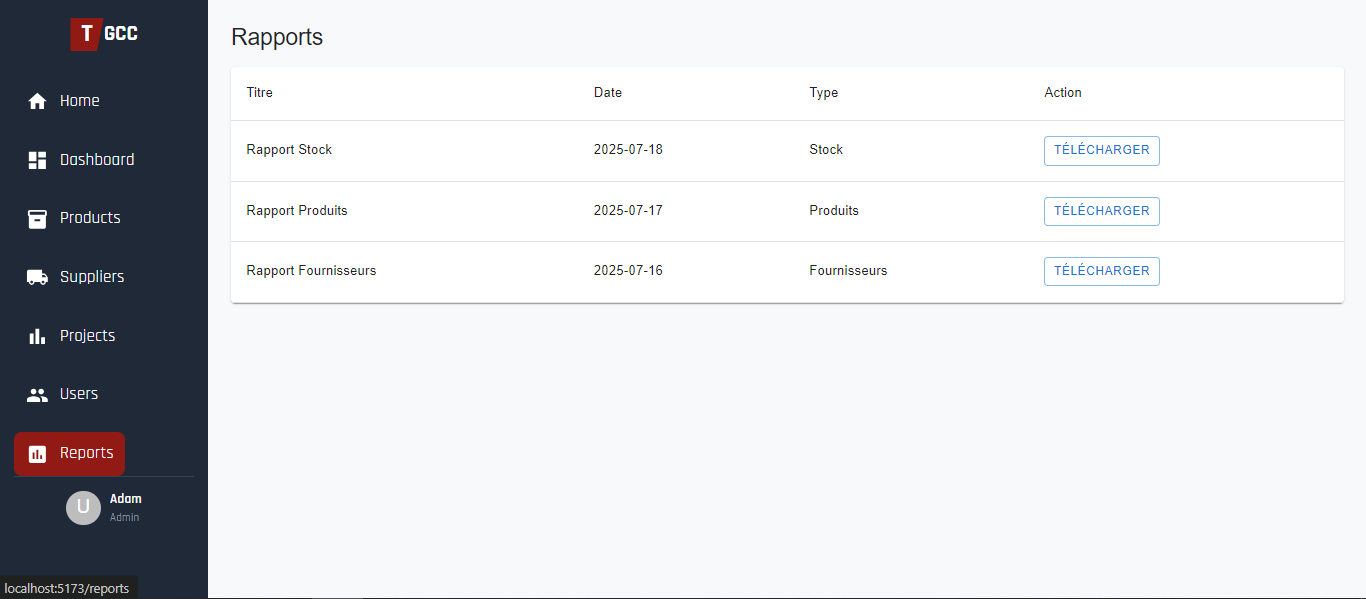
La page **Reports** permet de générer et consulter des **rapports d’activité** liés à la gestion du stock. Elle offre une vision synthétique des opérations passées et permet une **traçabilité complète**.

Figure 20: Page Reports

Fonctionnalités principales :

* **Génération automatique** de rapports périodiques (quotidiens, hebdomadaires, mensuels) sur les mouvements de stock (entrées, sorties).
* **Exportation des rapports** au format PDF ou Excel pour archivage ou impression.
* **Filtrage avancé** : par date, type de mouvement, article ou fournisseur.
* **Affichage dynamique** : les données sont présentées sous forme de tableaux ou de graphiques, selon le besoin de l’utilisateur.

Cette fonctionnalité est particulièrement utile pour les responsables logistiques ou administrateurs souhaitant effectuer un **suivi rigoureux** et disposer d’**indicateurs précis** pour la prise de décision ou les audits.

## **2.9 Page des paramètres (Settings)**

Figure 21: Page Settings

La page **Settings** permet à l’utilisateur (principalement l’administrateur) de gérer les préférences de l’application :

* Modification du mot de passe ;
* Choix du thème (clair/sombre) ;
* Choix de langue (Français, English) ;

Cette page contribue à la personnalisation de l’expérience utilisateur et à la sécurisation du système.