A blue and white logo

Description automatically generated

Rapport du projet d’application de gestion de stock

à l’aide de la programmation orientée objet

en Java

**Réalisé par : Encadré par :**

Yassir Daoudi 77 Mourchid Fatima

Soufiane Harzane 90

Ilyacine Manssouri 98

Youssef Baticha 108

Sommaire

[Introduction 3](#_Toc154602805)

[Réalisation 4](#_Toc154602806)

[Structure du projet 4](#_Toc154602807)

[Structure de la base de données : 5](#_Toc154602808)

[Fonctionnalités : 6](#_Toc154602809)

[Conclusion 9](#_Toc154602810)

# Introduction

Ce rapport présente un projet de développement d'une application de gestion des stocks en Java et Java fx dotée de fonctionnalités CRUD basiques.

Nous avons développé cette application dans le but de simplifier la gestion des stocks pour les entreprises. Grâce à son interface conviviale et intuitive, les utilisateurs peuvent facilement gérer leur inventaire, ajouter de nouveaux produits, mettre à jour les informations existantes et supprimer des articles obsolètes.

Nous avons conçu cette application en utilisant Java et JavaFX pour bénéficier de la flexibilité et de l'extensibilité offertes par ces technologies. De plus, nous avons mis en place une architecture solide et modulaire, garantissant une évolutivité future et une maintenance aisée.

Dans ce rapport, nous explorerons la structure et les fonctionnalités clés de cette application.

# Réalisation

## Structure du projet

**A screenshot of a computer

Description automatically generated**

A screenshot of a computer

Description automatically generated

Notre application de gestion de stock en Java est construite avec deux packages principales : **‘db.units’** et **‘example.gestionstock’**. Dans **‘db.units’**, on s'occupe des données avec les classes Produit et Client, qui aident à gérer les infos sur les produits et les clients et assurant la connexion fluide entre l'application et les données. Elles sont réalisée de manière à faciliter les opérations de base comme l’ajout, la mise à jour et la suppression des informations.

Ensuite, dans le package 'example.gestionstock', on trouve tout ce qui concerne l'interface graphique de l'application. Les classes ayant un nom qui commence par "CModal" ou "PModal" sont responsables de l'affichage des fenêtres "Ajouter", "Supprimer", etc. Elles héritent toutes de la classe abstraite ModalUI qui implémente des méthodes communes telles que "Enregistrer/Save". Enfin, la classe controller s'occupe de la gestion des événements initiés par l'utilisateur lors de l'exécution.

La classe GestionApp est la classe principale "Main".

## Structure de la base de données :

* La structure de la table Client :

A screen shot of a computer

Description automatically generated

* La structure de la table Produit:

A blue screen with white text

Description automatically generated

## Fonctionnalités :

* Pour la Table Client :
  + Ajouter : A screenshot of a computer

    Description automatically generated
  + Modifier : A screenshot of a computer

    Description automatically generated
  + Supprimer : A screenshot of a computer

    Description automatically generated
  + La vue :

A screenshot of a computer

Description automatically generated

* Pour la table Produit :
* Ajouter : A screenshot of a computer

  Description automatically generated
* La vue :

A screenshot of a computer

Description automatically generated

# Conclusion

En conclusion, le projet d'application de gestion de stock en Java a été une expérience enrichissante. En utilisant la programmation orientée objet (POO), nous avons structuré l'application de manière méthodique, tirant parti des concepts fondamentaux tels que l'encapsulation, l'héritage et le polymorphisme.

La POO a facilité la gestion du code en le fragmentant en classes distinctes, rendant ainsi la logique de l'application plus compréhensible. L'emploi de deux packages distincts, l'un pour la gestion des données **(‘dbunits’**) et l'autre pour l'interface utilisateur (**‘example.gestionstock’**), a offert une clarté organisationnelle, simplifiant le développement et renforçant la flexibilité de l'application.

Cette approche a également permis une évolution plus aisée de l'application, facilitant l'adaptation à de futurs besoins. L'expérience a renforcé notre compréhension de la POO et son rôle dans la création d'applications robustes et évolutives.