

**IStoreApp**

Documentation Technique

Auteurs : CUREAU Melvin

Date : 29/01/2024

**Ce document comporte 5 pages :**

* **Page 1 :** Page de garde ; Sommaire
* **Page 2 :** Introduction ; Architecture ;
* **Page 3 :** Choix techniques ; Structure de données
* **Page 4 :** Structure de données
* **Page 5 :** Références ; Conclusion

# Introduction/

***IStore*** est une application de gestion de magasin en ligne conçue pour permettre aux utilisateurs de gérer efficacement les opérations liées à un magasin.

Son objectif principal est de fournir une plateforme conviviale et fonctionnelle pour la gestion des utilisateurs, des administrateurs, des magasins, des inventaires et des articles associés.

Son architecture modulaire et son utilisation de technologies éprouvées comme ***Java*** et ***MySQL*** garantissent une robustesse et une évolutivité accrues.

Ce document explique les détails techniques de l'implémentation et de la réalisation de notre projet en Java et utilisant la librairie graphique Swing.

# Architecture/

**Structuration du dossier du projet :**

* Un dossier ‘*Docs’* / contenant toutes les documentations nécessaires
* Un dossier ‘*Src’* / contenant tous les packages et les différents fichiers en .java.
* Un fichier ‘LICENSE’ / contient la licence MIT qui régule l’accès à notre projet
* Un fichier ‘README’ / explique brièvement notre projet
* Un dossier ‘sqlDatabase’ / contenant la base de données MySQL utilisé lors du développement de ce projet.

L'application iStore suit une architecture MVC (Modèle-Vue-Contrôleur). Voici une brève description de chaque composant :

**Structuration du dossier source *(‘src’)* de notre projet :**

* Un package ‘*dataAccess* / contenant
* Un dossier ‘*model* / contenant
* Un fichier ‘service / contenant
* Un fichier ‘ui / contenant

# Choix techniques/

* **Langage de programmation :** Java avec la librairie graphique Swing
* **Editeur de code :** IntelliJ IDEA 2023.3.3
* **Système de contrôle de version :** Git et GitHub
* **Système d’exploitation :** Windows

Le code est structuré en plusieurs fichiers et classes pour faciliter la lisibilité et la maintenance. Les fonctions sont découpées de manière modulaire et indépendante pour faciliter la réutilisation du code. L’interface utilisateur est gérée via la librairie graphique Swing.

Pour la réalisation du projet, le langage ***Java*** avec la librairie ***Swing*** a été utilisé.  
Java est un choix judicieux pour la réalisation du projet ***iStore*** en raison de sa portabilité, de sa fiabilité et de sa polyvalence. En tant que langage de programmation multiplateforme, Java permet à l'application iStore d'être exécutée sur divers systèmes d'exploitation sans nécessiter de modifications majeures. De plus, la vaste gamme de bibliothèques et de frameworks Java offre une grande flexibilité pour le développement d'applications robustes et évolutives. La gestion automatique de la mémoire de Java simplifie également le processus de développement en réduisant les risques de fuites de mémoire. Enfin, la popularité du langage Java, ainsi que sa large communauté de développeurs, garantissent un support continu et différentes ressources pour résoudre les problèmes et optimiser les performances de l'application.

Afin de réaliser le projet de façon la plus collaborative possible nous avons utilisés un outil en général : *GitHub,* c’est une plateforme de gestion de code source qui facilite la collaboration entre les développeurs. Les membres peuvent partager leur code, effectuer des modifications, créer des branches pour expérimenter et fusionner les modifications pour les incorporer dans le code principal. Cela facilite grandement la gestion de version, la collaboration et le suivi des modifications apportées au code. Cet outil nous a permis de travailler ensemble de manière plus efficace.

Un diagramme des différentes classes a aussi été réalisé, qui compose le code pour une meilleur compréhension de celui-ci.

# Structure de données/

Le jeu utilise plusieurs structures de données pour stocker les informations nécessaires.

**Algorithmes de gestion de la grille :**

La grille est stockée dans une matrice à deux dimensions. Chaque case contient un entier représentant son état actuel :

* 0 -> représente une case vide
* 1 -> représente l’affichage d’une case disponible (pions noir)
* 2 -> représente l’affichage des pions rouge (renards)
* 3 -> représente affichage des pions vert (oies)
* 4 -> représente l’affichage des futurs déplacements sans capture possible
* 5 -> représente l’affichage des futurs déplacements avec une capture possible

Par le biais de la fonction ***displayWindows***(), chaque entier est représenté graphiquement à une couleur et une forme distincte grâce à la librairie graphique Tkinter.

**Algorithmes implémentant les déplacements et les captures :**

Les déplacements sont effectués par l’intermédiaire de la fonction ***onClick***().

Cette fonction récupère les coordonnées du clic pour afficher ensuite les déplacements possibles pour le pion sélectionné.

Une image contenant texte, Police, capture d’écran, nombre

Description générée automatiquement

Pour effectuer un déplacement, l’algorithme mis en place fait en sorte de remplacer la case choisie parmi celles proposée en modifiant directement dans la matrice l’entier correspondant, comme vu précédemment. Bien sûr une vérification stricte à été mise en place afin de ne pas faire déplacer un pion dans une case vide ou sur un autre pion déjà existant.

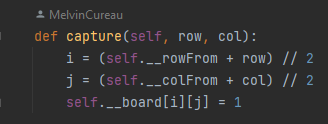
Une seconde fonction **clearBoard()**, permet de nettoyer la grille de tous les pions qui indique au joueurs les éventuels déplacements possibles (voir ci-dessous).

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Pour cela elle parcourt la grille, et remplace tous les entiers égaux à 4 ; 5 ou 6 (représentant les déplacements et captures possibles) par 1 (représentant une case disponible).

Enfin une fonction de capture a été mis en place, celle-ci récupère les coordonnées du déplacement avant (rowFrom et colFrom) et après (row et col) pour remplacer l’entier qui se trouve entre les deux positions dans la matrice.



# Références/

Pour la réalisation de ce projet l’IDE ***IntelliJ 2023.3.3*** a été utilisé.

Le code de ce projet ainsi que les différentes ressources annexes seront publiés sur GitHub.

# Conclusion/

Pour résumer, la réalisation de ce projet a été une expérience enrichissante et stimulante. Cela m'a permis de mettre en pratique les compétences en programmation que j'ai acquises, tout en découvrant de nouveaux outils en Java.

En ce qui concerne l'architecture du projet, j'ai opté pour une structure de code modulaire qui favorise une meilleure organisation et facilite la maintenance, contribuant ainsi à rendre le code plus lisible et plus simple.

Concernant les perspectives d'amélioration, il serait intéressant d'explorer la possibilité d'introduire une fonctionnalité de mise en pause et de reprise des parties, en sauvegardant l'état du plateau de jeu dans un fichier au format .txt, permettant ainsi aux joueurs de reprendre là où ils s'étaient arrêtés.

Cette documentation technique offre donc une vue d’ensemble de l’application. Pour l’avenir il serait intéressant d’y apporter certaines améliorations notamment concernant la modularité du code, la documentation et aux différents tests afin de garantir la robustesse et la maintenabilité à long terme de l’application.

Je reste disponible pour toute question ou commentaire à propos de ce projet.