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 ; librairie graphique Swing
- **Editeur de code :** IntelliJ IDEA 2023.3.3
- **Système de contrôle de version :** Git et GitHub
- **Système d'exploitation :** Windows

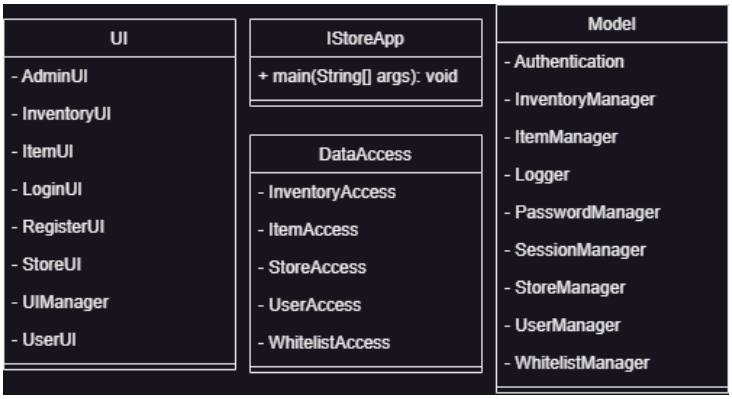
Le code est structuré en plusieurs fichiers et classes pour faciliter la lisibilité et la maintenance. Les fonctions sont découpées de manière modulaire et indépendante pour faciliter la réutilisation du code. L'interface utilisateur est gérée via la librairie graphique Swing.

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 Framework 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.

Vous trouverez ci-joint le diagramme des différents packages composant le projet :



Guide d'installation et de configuration/

Conditions préalables

Avant de commencer l'installation, assurez-vous que votre système dispose des éléments suivants :

- Kit de développement Java (JDK) version 11 ou supérieure.
- MySQL Server ou un autre système de gestion de base de données relationnelle.
- Apache Maven pour la gestion des dépendances et la construction de projets.
- Git pour cloner le référentiel du projet.

Installation

- Clonez le référentiel Git sur votre machine locale
- Configurer la base de données

Assurez-vous que votre serveur MySQL est en cours d'exécution avant de démarrer l'application.

Par défaut lors de la première connexion, un compte 'admin/admin' sera disponible.

Ce compte administrateur par défaut permet un accès rapide aux privilèges administratifs sans avoir besoin d'une inscription manuelle. Cependant, il est important de noter que l'utilisation des informations d'identification « admin/admin » par défaut présente des risques de sécurité si elles ne sont pas modifiées immédiatement après l'installation.

Il est fortement recommandé de modifier les informations d'identification par défaut une fois la configuration initiale terminée pour améliorer la sécurité.

Structure de données/

Architecture globale

1. Modèles de Données

L'application IStoreApp utilise plusieurs modèles de données pour représenter les entités principales de l'application.

1.1 Utilisateur (User)

Attributs :

Identifiant (id) : Identifiant unique de l'utilisateur

Email : Adresse email de l'utilisateur

Pseudo : Nom d'utilisateur de l'utilisateur

Mot de passe : Mot de passe haché de l'utilisateur

Rôle : Rôle de l'utilisateur dans l'application (ex: utilisateur, administrateur)

Fonctionnalités :

Création, lecture, mise à jour et suppression (CRUD) des utilisateurs

Authentification des utilisateurs avec gestion des sessions

1.2 Magasin (Store)

Attributs :

Identifiant (id) : Identifiant unique du magasin

Nom : Nom du magasin

Fonctionnalités :

Création, lecture, mise à jour et suppression (CRUD) des magasins

Association des utilisateurs aux magasins

2. Gestion des Sessions

L'application utilise un système de gestion de sessions pour gérer l'authentification des utilisateurs et leur accès aux fonctionnalités de l'application.

2.1 SessionManager

Fonctionnalités :

Démarrage d'une nouvelle session pour un utilisateur donné

Récupération de l'email de l'utilisateur à partir de l'identifiant de session

Fin d'une session

3. Gestion de la Liste Blanche

Pour permettre l'inscription des utilisateurs, l'application utilise une liste blanche d'adresses email autorisées.

3.1 WhitelistManager

Fonctionnalités :

Ajout d'une adresse email à la liste blanche

Suppression d'une adresse email de la liste blanche

4. Accès aux Données

L'accès aux données est géré par des classes d'accès aux données qui interagissent avec la base de données MySQL.

4.1 UserAccess

Fonctionnalités :

Opérations CRUD sur les utilisateurs dans la base de données

4.2 StoreAccess

Fonctionnalités :

Opérations CRUD sur les magasins dans la base de données

4.3 WhitelistAccess

Fonctionnalités :

Ajout et suppression d'adresses email dans la liste blanche dans la base de données

5. Interface Utilisateur

L'interface utilisateur est réalisée en Java Swing et permet aux utilisateurs d'interagir avec l'application à travers une interface graphique.

Conclusion

La structure de données de l'application IStoreApp est conçue pour fournir une gestion efficace et sécurisée des utilisateurs, des magasins et des sessions utilisateur. En utilisant des modèles de données clairs et des fonctionnalités bien définies, l'application offre une expérience utilisateur fluide et intuitive.

Système de sessions/

Le système de gestion des sessions mis en place dans le fichier *SessionManager.java* permet de gérer les sessions utilisateur dans une application.

Voici une explication détaillée du fonctionnement de ce système :

- **Map de sessions :** Le système utilise une structure de données de type Map pour stocker les sessions actives. Dans ce cas, une HashMap est utilisée pour associer chaque identifiant de session à l'email de l'utilisateur correspondant. Cela permet un accès rapide aux informations de session à partir de l'identifiant de session.

- **Démarrage de session :** Lorsqu'un utilisateur se connecte ou démarre une session, la méthode *startSession(String userEmail)* est appelée. Cette méthode génère un identifiant de session unique à l'aide de la méthode *generateSessionId(),* puis associe cet identifiant à l'email de l'utilisateur dans la map des sessions. Enfin, elle enregistre le début de la session dans les logs en indiquant l'utilisateur et l'identifiant de session.

```
// Méthode pour démarrer une nouvelle session pour un utilisateur donné
public static String startSession(String userEmail) {
    String sessionId = generateSessionId();
    sessions.put(sessionId, userEmail);
    // Enregistrer le début de la session dans les logs
    Logger.log("Session started - User: " + userEmail + ", Session ID: " + sessionId);
    return sessionId;
}
```

- **Récupération de l'email de l'utilisateur :** La méthode *getUserEmail(String sessionId)* permet de récupérer l'email de l'utilisateur à partir de l'identifiant de session. Elle recherche dans la map des sessions l'email associé à l'identifiant de session fourni et le renvoie.

```
// Méthode pour récupérer l'email de l'utilisateur à partir de l'identifiant de session
public static String getUserEmail(String sessionId) {
    return sessions.get(sessionId);
}
```

- **Fin de session :** Lorsque l'utilisateur se déconnecte ou termine sa session, la méthode *endSession(String sessionId)* est appelée. Cette méthode enregistre la fin de la session dans les logs en indiquant l'utilisateur et l'identifiant de session, puis supprime l'entrée correspondante de la map des sessions.

```
// Méthode pour mettre fin à une session
public static void endSession(String sessionId) {
    // Enregistrer la fin de la session dans les logs
    String userEmail = sessions.get(sessionId);
    Logger.log("Session ended - User: " + userEmail + ", Session ID: " + sessionId);
    sessions.remove(sessionId);
}
```

- **Génération d'identifiant de session :** La méthode *generateSessionId()* est une méthode utilitaire privée qui génère un identifiant de session unique à l'aide de la classe UUID. Cet identifiant est une chaîne aléatoire unique qui est utilisée comme clé pour identifier la session de l'utilisateur.

```
// Méthode pour générer un identifiant de session unique
private static String generateSessionId() {
    return UUID.randomUUID().toString();
}
```

En résumé, ce système de gestion des sessions assure le suivi des utilisateurs connectés en associant un identifiant de session unique à chaque session utilisateur, en enregistrant le début et la fin de chaque session dans les logs, et en fournissant des méthodes pour démarrer, récupérer et mettre fin aux sessions. Cela permet à l'application de suivre l'état de connexion des utilisateurs et de maintenir leur contexte d'utilisation tout au long de leur session, il permet notamment de vérifier les droits associés aux utilisateurs en contrôlant leur rôle d’admin ou non, tout en améliorant la sécurité.

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 à l'adresse suivante : https://github.com/MelvinCr1/2JAVA_IStoreApp

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.