MULAMBA TSHISHIKA Andre Keith

KONNANG FOLACK Ange Andrea

IStore - Documentation Technique

Description

IStore est une application de gestion de boutiques et d'articles réalisée en Java. L'application permet de gérer les boutiques, les articles associés et les employés, tout en assurant l'accès sécurisé par une gestion des utilisateurs. L'application utilise **Swing** pour l'interface graphique et une base de données **MySQL** pour stocker et gérer les informations.

Structure du Projet

Dossier app/

Ce dossier contient les classes principales de l'application.

1. Boutique.java

- **Description** : Entité représentant une boutique. Elle contient les propriétés essentielles comme l'ID de la boutique et son nom.
- **Attributs** :
    - id : Identifiant unique de la boutique (type int).
    - nom : Nom de la boutique (type String).
- **Méthodes** :
    - getId(): Retourne l'identifiant de la boutique.
    - getNom(): Retourne le nom de la boutique.

2. BoutiqueFrame.java

- **Description** : Fenêtre principale qui permet d'interagir avec les boutiques. Elle affiche une liste de boutiques sous forme de boutons hexagonaux.
- **Fonctionnalités** :
    - **Création de boutons hexagonaux** pour chaque boutique à partir de la base de données.
    - **Interaction avec la base de données** pour récupérer et afficher les informations sur les boutiques.
- **Méthodes** :
    - createHexagonButton(): Crée un bouton hexagonal pour chaque boutique.
    - getBoutiques(): Charge les boutiques depuis la base de données via une connexion JDBC.

3. BoutiqueManager.java

- **Description** : Classe responsable de la gestion des boutiques.
- **Fonctionnalités** :
    - **Chargement** des boutiques depuis la base de données.
    - **Ajout, mise à jour et suppression** des boutiques dans la base de données.
- **Méthodes** :
    - loadBoutiques(): Charge toutes les boutiques depuis la base de données.
    - addBoutique(), updateBoutique(), deleteBoutique(): Exécute les requêtes SQL pour manipuler les données dans la base.

4. ArticleFrame.java

- **Description** : Fenêtre dédiée à l'affichage des articles d'une boutique.
- **Fonctionnalités** :
    - **Affichage des articles** associés à une boutique spécifique.
    - **Gestion des articles** (ajout, modification, suppression).
- **Méthodes** :
    - getArticles(): Récupère la liste des articles à partir de la base de données.
    - addArticle(), updateArticle(), deleteArticle(): Effectue des opérations CRUD sur les articles.

5. ArticleManager.java

- **Description** : Responsable de la gestion des articles (ajout, mise à jour, suppression).
- **Méthodes** :
    - addArticle(), updateArticle(), deleteArticle(): Gestion des opérations CRUD sur les articles.

6. CRUDFrame.java

- **Description** : Interface pour les opérations CRUD (Create, Read, Update, Delete) génériques. Utilisée pour gérer les différentes entités du projet.
- **Méthodes** :
    - Gestion des formulaires pour l'ajout, la mise à jour et la suppression d'éléments.

7. EmployeManager.java

- **Description** : Classe qui gère les employés dans l'application.
- **Fonctionnalités** :
    - **Gestion des employés** associés aux boutiques (ajout, mise à jour, suppression).
- **Méthodes** :

- loadEmployes(): Charge les employés depuis la base de données.
- addEmploye(), updateEmploye(), deleteEmploye(): Effectue les opérations CRUD sur les employés.

8. ListBlancheFrame.java

- **Description** : Interface pour la gestion de la liste blanche des employés.
- **Méthodes** :
  - Permet l'ajout, la suppression des employés de la liste blanche.

9. UserListFrame.java

- **Description** : Interface qui permet de visualiser les utilisateurs et leur gestion.
- **Méthodes** :
  - **Affichage et gestion** des utilisateurs de l'application.

10. UserMainPage.java

- **Description** : Page principale qui permet aux utilisateurs de naviguer dans l'application.
- **Fonctionnalités** :
  - Permet à l'utilisateur d'accéder aux différentes sections de l'application (boutiques, articles, etc.).

Dossier auth/

Ce dossier contient les classes responsables de l'authentification des utilisateurs.

1. SignUp.java

- **Description** : Interface permettant à un nouvel utilisateur de s'inscrire.
- **Fonctionnalités** :
  - Inscription via un formulaire qui enregistre un utilisateur dans la base de données.
- **Méthodes** :
  - registerUser(): Valide et insère les informations d'un nouvel utilisateur dans la base de données.

2. LogIn.java

- **Description** : Interface de connexion permettant à un utilisateur existant de se connecter à l'application.
- **Fonctionnalités** :
  - Vérifie les informations de connexion via une requête SQL.
- **Méthodes** :

- loginUser(): Valide les informations d'identification et connecte l'utilisateur à l'application.

Technologies Utilisées

- **Java** : Langage de programmation principal.
- **Swing** : Framework pour la création d'interfaces graphiques en Java.
- **MySQL** : Système de gestion de base de données relationnelle pour le stockage des informations.
- **JDBC** : API Java pour la gestion des connexions et des requêtes SQL à la base de données.

Base de Données

Le projet utilise une base de données MySQL avec les tables suivantes :

Tables

boutique

sql

CopierModifier

```
CREATE TABLE boutique (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nom VARCHAR(255) NOT NULL,
  date_creation DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

article

sql

CopierModifier

```
CREATE TABLE article (
  id INT AUTO_INCREMENT PRIMARY KEY,
  id_boutique INT,
  nom VARCHAR(255) NOT NULL,
  prix DECIMAL(10, 2) NOT NULL,
  FOREIGN KEY (id_boutique) REFERENCES boutique(id)
);
```

employe

sql

CopierModifier

```
CREATE TABLE employe (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nom VARCHAR(255) NOT NULL,
  id_boutique INT,
  FOREIGN KEY (id_boutique) REFERENCES boutique(id)
);
```

liste_blanche

sql

CopierModifier

```
CREATE TABLE liste_blanche (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nom_utilisateur VARCHAR(255) NOT NULL
);
```

Connexion JDBC

Pour connecter l'application à la base de données MySQL, il faut configurer les informations de connexion (URL, utilisateur, mot de passe) dans les classes de gestion de la base de données, comme DbHelper.java.

Exemple de connexion JDBC :

java

CopierModifier

```
public Connection connect() throws SQLException {
  try {
    String url = "jdbc:mysql://localhost:3306/istore";
    String user = "root";
    String password = "password";
    return DriverManager.getConnection(url, user, password);
  } catch (SQLException e) {
    throw new SQLException("Erreur de connexion à la base de données", e);
  }
}
```

Instructions d'Installation

1. **Cloner le projet** depuis le dépôt Git.
2. **Configurer la base de données MySQL** :
    - Créez la base de données istore et exécutez les requêtes SQL pour créer les tables.
3. **Configurer les informations de connexion** dans la classe DbHelper.java (si nécessaire).
4. **Exécuter le projet** à partir de votre IDE Java (par exemple IntelliJ IDEA).