Rapport Explicatif sur le Programme d'Interface pour les Avis d'Hôtel

Introduction

L'industrie hôtelière est fortement axée sur la satisfaction des clients, et les avis jouent un rôle essentiel dans l'évaluation de la qualité des services offerts par un établissement. Dans le contexte de cette importance croissante des retours clients, notre projet vise à développer une interface conviviale permettant aux clients de soumettre leurs avis sur différents aspects d'un hôtel, tels que le confort, la sécurité, le divertissement, et la propreté.
L'objectif ultime de cette application est de permettre aux hôtels de recueillir efficacement les retours de leurs clients, favorisant ainsi une amélioration continue de leurs services.

Pour atteindre cet objectif, nous avons mis en place une application Java avec une interface graphique, facilitant ainsi la saisie et la soumission des avis.

Ce rapport détaillera la structure et le fonctionnement du programme, en mettant l'accent sur trois composants principaux : la classe `**testFrame**` qui gère l'interface utilisateur, la classe `**Traitement**` qui prend en charge le traitement des données et leur enregistrement dans la base de données, et enfin, la classe `**Connect**` qui assure la connexion à la base de données MySQL. Chaque composant sera examiné en profondeur pour comprendre son rôle spécifique dans le contexte du projet.

1. `testFrame.java` (Interface Graphique)

1.1. Attributs et Composants

Dans la classe **testFrame.java**, l'interface graphique est définie pour permettre aux utilisateurs de saisir et soumettre leurs avis sur un hôtel. Les composants suivants sont utilisés pour créer une interface conviviale et intuitive :

- **textFieldName**: Ce champ de texte permet aux utilisateurs de saisir leur nom lorsqu'ils soumettent un avis. Cela permet de personnaliser chaque avis en associant un nom à la rétroaction.
- **textAreaComment**: Il s'agit d'une zone de texte étendue où les utilisateurs peuvent rédiger des commentaires détaillés sur leur expérience à l'hôtel. Cette zone de texte offre suffisamment d'espace pour des retours approfondis et informatifs.

- **btnSubmit**: Ce bouton "Submit" sert d'élément déclencheur pour soumettre l'avis une fois que l'utilisateur a rempli les informations nécessaires. Il est associé à une action qui collecte les données saisies et les transmet au système de traitement.
- **Comfort**, **Safety**, **Entertainment**, **Cleanliness**: Ces curseurs (JSlider) permettent aux utilisateurs d'évaluer différents aspects de leur expérience hôtelière, tels que le confort, la sécurité, le divertissement et la propreté. Les curseurs offrent une méthode visuelle et interactive pour attribuer une note à chaque critère, fournissant ainsi des données quantitatives pour évaluer les performances de l'hôtel.
- **jPanel3:** Un panneau Swing utilisé pour afficher le graphique circulaire.

Cette structure d'interface permet aux utilisateurs d'interagir facilement avec l'application, en fournissant des champs de saisie explicites

1.2. Méthodes

Dans cette JFrame, trois méthodes principales contribuent à la création de l'interface :

- **initializeUI()**: Initialise l'interface utilisateur, met en place les composants Swing et défini les actions des boutons.
- **getDataFromDatabase()**: Récupère les données depuis une base de données MySQL pour alimenter un graphique circulaire (pie chart).
- **showPieChart()**: Affiche un graphique circulaire basé sur les données récupérées de la base de données.
- **submitReview()**: Récupère les informations saisies par l'utilisateur et les affiche dans la console. Appelle également une méthode **Traitement.saveRatingToDatabase()** pour sauvegarder les avis dans la base de données.

1.3. Interface Graphique

L'interface utilisateur est structurée avec deux panneaux principaux, `pan1` et `pan2`. Le premier panneau (`pan1`) est dédié aux composants aidant à la saisie d'avis. Le deuxième panneau (`pan2`) est réservé à l'affichage d'une image (commentée dans le code). Cette organisation offre une disposition claire et intuitive pour les utilisateurs, facilitant ainsi la saisie des avis.

1.4. Connexion à la Base de Données

Au démarrage de l'application, la connexion à la base de données est établie via la classe `Connect`. La méthode `getConnection()` de cette classe est utilisée pour récupérer l'objet de connexion (`con`).

Cette approche garantit que la connexion à la base de données est prête à être utilisée tout au long de l'exécution de l'application.

2. ‘Traitement.java’ : traitement des Données

Cette classe gère le traitement des données liées aux avis.

Les attributs `DB_URL`, `DB_USER`, et `DB_PASSWORD` stockent les informations nécessaires à la connexion à la base de données.

Lorsque l'utilisateur clique sur le bouton "Submit", les données saisies sont récupérées et affichées dans la console. Ensuite, la méthode `saveRatingToDatabase` de la classe `Traitement` est appelée pour insérer ces données dans la table "avis" de la base de données Les données telles que le nom, les évaluations et le commentaire sont transmises à la base de données pour être stockées de manière pérenne.

Cette classe joue un rôle essentiel dans la persistance des données relatives aux avis dans le contexte de notre application.

3. `Connect.java` (Connexion à la Base de Données)

1.1. Attributs

- `con`: Un objet de type `Connection` de JDBC utilisé pour représenter la connexion à la base de données. Il est déclaré comme statique pour que toutes les instances de la classe partagent la même connexion.

1.2. Bloc statique

- La classe `Connect` contient un bloc statique qui est exécuté lors du chargement de la classe. Dans ce bloc statique, la classe s'assure de charger le pilote JDBC (`Class.forName("com.mysql.jdbc.Driver")`) nécessaire pour établir une connexion avec la base de données MySQL.

1.3. Initialisation de la Connexion

Après le chargement du pilote JDBC, la classe établit une connexion avec la base de données MySQL en utilisant `DriverManager.getConnection`. Les paramètres de connexion tels que l'URL de la base de données, le nom d'utilisateur et le mot de passe sont spécifiés dans cette méthode.

1.4. Méthode `getConnection()`

- La classe `Connect` fournit une méthode statique `getConnection()` qui renvoie l'instance unique de la connexion à la base de données. Cette méthode est utilisée par d'autres classes, notamment `Traitement.java`, pour obtenir la connexion à la base de données.

En résumé, la classe `Connect.java` encapsule la logique de connexion à la base de données en utilisant le modèle Singleton. Elle fournit une seule instance de connexion qui peut être partagée dans toute l'application.