Projet TravelTheWorld - Documentation

Table des matières

---

1. Description du Projet

1.1 Fonctionnalités Principales

Le projet "TravelTheWorld" offre les fonctionnalités suivantes :

- Gestion des Villes : Les administrateurs peuvent ajouter, modifier et supprimer des villes.
- Gestion des Parcs : Les administrateurs peuvent ajouter, modifier et supprimer des parcs d'attractions. Chaque parc

est associé à une ville spécifique.

- Gestion des Stations : Les administrateurs peuvent ajouter, modifier et supprimer des aéroports ou des gares. Chaque station est associé à une ville spécifique.
- Gestion des Moyens de Transport : Les administrateurs peuvent ajouter, modifier et supprimer des avions ou des trains.
- Gestion des Trajets : Les administrateurs peuvent ajouter, modifier et supprimer des trajet entre 2 stations.
- Planification de Visites : Les utilisateurs peuvent planifier des visites à un parc particulier. Ils peuvent également afficher les visites planifiées et suivre leur historique.
- Planification des Voyages : Les utilisateur peuvent planifier des voyages en train ou en avion. Ils peuvent également afficher les voyages planifiées et suivre leur historique.

1.2 Technologies Utilisées

- Java (11.0.20)
- JavaFX
- Hibernate
- Maven (version 3.6.3)
- Swing
- MariaDB
- Eclipse

2. Architecture du Projet

*2.1 Structure du Code Source

- `main.java` : Contient les classes principales du projet pour l'interface JavaFX.
- `repositories` : Contient les classes pour l'accès aux données.
- `interfaces` : Contient les interfaces.
- `classes` : Contient les classes de modélisation.
- `web` : Contient les classes pour l'interface utilisateur web.
- `resources` : Contient le fichier `hibernate.cfg.xml` .

2.2 Gestion des Données

Script pour la base de données : `structure.sql` .
DB : TravelTheWorld
User : TravelTheWorld
PWD : TravelTheWorld

Exemple : "Créer utilisateur"

```
// Gestionnaire d'événements pour le bouton "Créer
utilisateur"
createButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String username = usernameField.getText();

        // Vérifie si le nom d'utilisateur est vide et
disponible
        if (username.trim().isEmpty()) {
            messageLabel.setText("Le nom d'utilisateur
ne peut pas être vide.");
            return;
        }
        if (userRepository.getByUsername(username) !=
```

```
null) {
            messageLabel.setText("L'utilisateur existe
déjà.");
            return;
        }

        // Crée un nouvel utilisateur et l'ajoute à la
base de données
        User newUser = new User();
        newUser.setNom(username);
        int userId = userRepository.add(newUser);
        if (userId != -1) {
            messageLabel.setText("Utilisateur créé : " +
username);
        } else {
            messageLabel.setText("Error.");
        }
        messageLabel.setText("Utilisateur créé : " +
username);
    }
});
```

MutatorRepository :

```
@Override public int add(T entity) { try { Session
session = this.getSessionFactory().openSession();
session.beginTransaction(); session.persist(entity);
session.getTransaction().commit(); session.close();
return entity.getId(); } catch (Exception e) {
System.err.println(e); return -1; } }
```

2.3 Interface Administrateur/Utilisateur

L'interface administrateur est développée en utilisant JavaFX. Les boîtes de dialogue sont utilisées pour la saisie et la modification des données. (Style du menu principal réalisé avec ChatGPT)
L'interface utilisateur est développée en utilisant Swing.

2.4 Intégration avec Hibernate

L'intégration avec Hibernate est assurée par la configuration d'une session Hibernate au démarrage de l'application. Les classes "Repository" sont responsables de l'accès aux données et de l'exécution des opérations CRUD (Create, Read, Update, Delete).

3. Guide d'Utilisation

3.1 Utilisation de l'Application

3.1.1 Administrateur

- Lancez l'application en exécutant la classe `JavaFXApp`.
  (Si eclipse n'arrive pas lancer l'application => Entrez la commande `mvn javafx:run` dans le repertoire du projet)
- Vous arrivez sur le menu principal qui affiche la liste des utilisateurs et qui permet d'accéder aux différentes pages de gestion.
- Utilisez les boutons "Ajouter", "Modifier" et "Supprimer " pour gérer les villes, parc, stations et moyen de transport.
- `ConsoleGUI.java` : Interface console permettant l'ajout, la modification et la suppression de ville.

3.1.2 Utilisateur

- Lancez l'application exécutant la classe `TravelTheWorldWebApp`.
- La première page invite l'utilisateur à entrer un nom d'utilisateur existant ou à en créer un nouveau.
- Une fois connecté, la liste des visites de parcs déjà réalisé apparait et permet de les modifier ou d'en ajouter d'autres. *(non fonctionnel)*
- Lors de l'ajout d'une visite, sélectionnez une ville pour faire apparaitre les parcs et stations associés et les moyens de transport les reliant aux autres villes. *(non fonctionnel)*

4. Conclusion

4.1 Perspectives d'Amélioration

- Fonctionnalités pour les utilisateurs manquantes :
  - Gestion des visites de parcs (ajout, modification, suppression).
  - Gestion des voyages en train et en avion (ajout, modification, suppression).
- Aucune réelle différence entre "Gare" et "Aéroport" et entre "Avion" et "Train".
- Ajout d'une authentification utilisateur avec mot de passe.
- Ajout d'une authentification Admin.
- Ajout de gestion des utilisateurs et des visites coté Admin.