Rapport de projet : Gestionnaire de contacts/carnet d'adresses

Amine LADEL

Présentation du projet

Le projet consiste en la création d'un gestionnaire de contacts/carnet d'adresses, en utilisant Qt et SQLite pour la gestion des données. L'application permet aux utilisateurs d'ajouter, de modifier, de supprimer et de rechercher des contacts dans une interface utilisateur. Chaque contact peut inclure des informations telles que le prénom, le nom, l'adresse email, le numéro de téléphone et l'adresse postale.

Instructions pour compiler et exécuter le programme

Prérequis

- Système d'exploitation : Debian 12 "Bookworm"
- Version de Qt : Qt6
- Autres dépendances : SQLite, CMake

Étapes de compilation

1. Installer Qt6 et les autres dépendances requises :

```
sudo apt-get update
sudo apt-get install qt6-base-dev sqlite3 libsqlite3-dev cmake g++
```

2. Cloner le dépôt du projet :

```
git clone https://github.com/amineladel/gestionnaire-de-contacts.git
cd gestionnaire_contacts
```

3. Créer un répertoire de build et compiler le projet :

```
mkdir build
cd build
cmake ..
make
```

Exécution du programme

```
./gestionnaire_contacts
```

Utilisation de l'application

Interface utilisateur

L'interface principale de l'application se compose avec les sections suivantes :

- **Liste des contacts** : Affiche tous les contacts enregistrés.
- **Formulaire/fenêtre de détails** : Permet d'ajouter ou de modifier les informations d'un contact.
- **Barre de recherche** : Facilite la recherche de contacts par nom ou autre critère.
- **Boutons d'action** : Incluent les options pour ajouter, modifier, supprimer et rechercher des contacts.

Fonctionnalités

- **Recherche avancée** : La recherche permet de filtrer les contacts par divers critères (prénom, nom, email, etc.).
- **Gestion de la BDD** : Utilisation de SQLite pour la gestion des données de contacts.

Description Technique

C++/Qt

- **Qt Widgets** : Utilisation de Qt Widgets pour créer l'interface utilisateur.
- **Qt SQL** : Utilisation de Qt SQL pour interagir avec la base de données SQLite.
- **Signal et slots** : Utilisation des mécanismes de signal et slot de Qt pour gérer les interactions utilisateur.
- **QML** : Utilisation de QML pour certaines parties de l'interface utilisateur pour améliorer la réactivité.

Points complexes et résolution

- **Gestion des connexions à la BDD** : Assurer que les connexions à la base de données SQLite sont correctement gérées et fermées après utilisation pour éviter les fuites de mémoire.
- **Synchronisation des données** : Mise en place de mécanismes pour synchroniser l'interface utilisateur avec la base de données en temps réel.
- **Interface** : Intégration de QML avec le backend C++.

Ressources utilisées

- Documentation officielle de Qt : Qt Documentation
- Tutoriels et forums en ligne (Stack Overflow, Qt Forums)
- Connaissances acquises en TP et TD