rapport de travail db 2

February 19, 2024

1 Besoins fonctionnels

Nous avons besoin d'une base de données pouvant gérer la liste d'utilisateurs d'une application. Chaque utilisateur a un pseudo-unique, un mot de passe, une liste d'amis, et des messages échangés avec ceux-ci.
Le pseudo et mot de passe sont obligatoires à l'inscription. La liste d'amis se complète une fois connecté.

Les opérations suivantes doivent être possibles :

- Inscrire un nouvel utilisateur:
  vérifier que son pseudo est disponible et que son mot de passe est conforme, sinon retourner une erreur.
- Login un utilisateur déjà inscrit :
  vérifier si son mdp et pseudo correspondent au data d'un utilisateur, retourner une erreur si le mot de passe ou pseudo ne correspond pas.
- Ajouter un ami à sa liste (une amitié est bidirectionnelle et unique).
- Supprimer un ami de sa liste (l'amitié disparait dans les deux sens).
- Recupérer les messages d'un utilisateur.
- Récupérer les messages échangé entre deux utilisateurs.
- Modifier son mot de passe
- Modifier son pseudo

2 Choix de librairie

Nous avons choisit d'utiliser la librairie SQLite3 pour le points suivants :

- Base de données embarquée: elle fonctionne directement à l'intérieur du processus de l'application
- Sans serveur : ne nécéssite pas de connexion a un serveur distinct, ce qui facilite son utilisation dans les applications client-serveur.
- Zéro configuration.
- Stockage dans un fichier unique: Les bases de données sont stockées dans un simple fichier sur le système de fichiers de l'hôte, ce qui facilite leur déploiement et leur gestion.

Commande d'instalation sur Debian/Ubuntu:

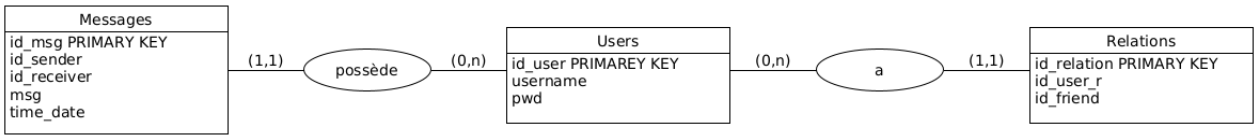
```
sudo apt-get install sqlite3
```

Include:

```
#include <sqlite3.h>
```

3 Architecture proposée pour la db

3.1 Shéma relationnel



Contraintes:
-les amitiés sont bidirectionnelle.
-envoyeur et destinateur doivent etre différents.
-utilisateur et ami doivent etre différents.
-mot de passe doit avoir min 8 caractères.
-le user name est unique.

3.2 Implémentation

La classe Database a pour rôle de fournir une interface pour interagir avec une base de données SQLite. Elle encapsule les opérations de base telles que l'exécution de requêtes SQL, l'insertion, la mise à jour et la suppression de données dans la base de données. Elle permet également de gérer la connexion à la base de données et de gérer les erreurs qui peuvent survenir lors de l'exécution des requêtes.

La classe Queries agit comme une couche d'abstraction supplémentaire au-dessus de la classe Database. Elle fournit des méthodes spécifiques pour effectuer des opérations sur la base de données qui sont spécifiques à l'application, telles que l'enregistrement et la connexion des utilisateurs, l'envoi de messages, l'ajout et la suppression d'amis, etc.

Nous utiliserons des objets de la **classe Enum DbError** pour facilité la gestion des erreurs possibles lors des opérations réalisées.

Et des objets de la **structure QueryResult** afin de retourner le résultat des requètes. Une instance de QueryResult a pour attribut un vecteur ou chaque élément est une entrée de la database correspondant a la requète et un objet DbError afin de savoir ce qui n'a pas été avec la requète en cas d'écheque. Si tout c'est bien déroulé, DbError = OK.

NB: Le résultat des requètes et affichage des erreurs est gérée dans la classe Queries, c'est une solution temporaire afin de visualiser le résultat. Idéalement les methodes de la classe Queries devraient retourner un QueryResult.

3.3 Compilation et lancement

Les options de compilation suivantes sont nécéssaires: **-lsqlite3 -lstdc++**

Je n'ai fais de makefile, j'ai compiler avec la commande suivante :

```
g++ -o my_db main.cpp database.cpp queries.cpp -lsqlite3 -lstdc++
```

Le programme se lance simplement via a la commande:

```
./my_db
```

3.4 Autres fichiers

Vous trouverez également un fichier DDLuserdb.sql, c'est le fichier contenant le code SQL de la déclaration des tables de la data base.

Et un fichier main.cpp, je l'ai utiliser pour réaliser des testes sur chaque methodes de la classe Queries. Les résultats sont imprimer en ligne de commande.