HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Labo 5 : Bot-tender Persistence

Créé par Christopher Meier.

Objectifs

L'objectif de ce Labo 5 est de compléter le Bot Tender Chatroom crée au labo 4 en ajoutant la persistance des données et des requêtes analytiques. On utilisera la bibliothèque Quill pour gérer la connexion vers la base de données et interroger la base de données.

Indications

Le code source du laboratoire vous est fourni sous la forme d'un fichier `.diff` qui contient les modifications par rapport au Labo 4. Le fichier est disponible sur Cyberlearn.

- Dans ce laboratoire, vous allez continuer votre implémentation du Labo 4 qui sera complété par les modifications appliquées par le fichier `.diff`.
- Ce laboratoire est à effectuer par groupe de 2, en gardant les mêmes formations que pour le Labo 4. Tout plagiat sera sanctionné par la note de 1.
- Ce laboratoire est à rendre en 2 parties (voir section Rendus). Nous vous demandons de continuer sur le repo GitHub créé au Labo 4. Les commits représentant les étapes à rendre seront taggés.
- Il n'est pas nécessaire de rendre un rapport, un code propre et correctement commenté suffit. Faites cependant attention à bien expliquer votre implémentation.
- Faites en sorte d'éviter la duplication de code.
- Préférez un style de programmation fonctionnel.

Description

Dans la première partie de ce laboratoire, vous allez implémenter la persistence des messages et des utilisateurs dans des tables (fournies, voir figure 1) de la base de données. Les pages webs implémentées dans le Labo 4 restent identiques en apparence.

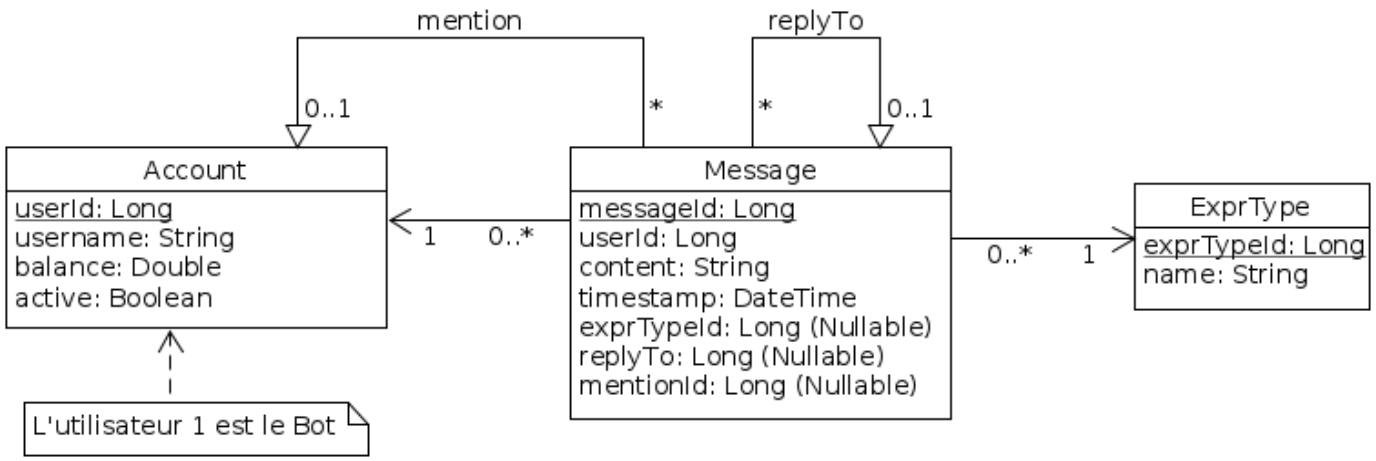

Figure 1: Diagramme des tables de la base de données

Dans la deuxième partie, vous allez implémenter des fonctionnalités de statistiques. Le code d'une page web affichant ces statistiques d'utilisation vous est fourni (voir figure 2).

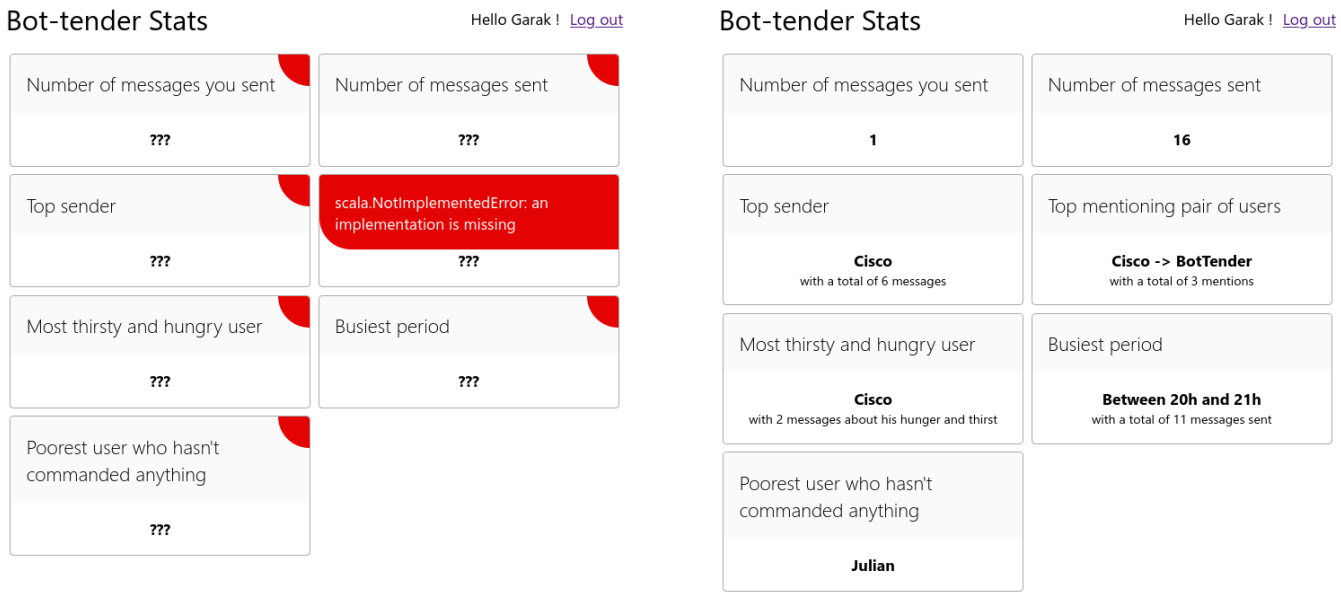

Figure 2: Une capture de la page des statistiques avant et après implémentation des méthodes analytiques

Implémentation

Pour vous guider dans votre implémentation, nous avons établi des étapes.

Étape 0 : Corriger, si besoin, votre implémentation de Bot-tender Chat

Comme ce labo se construit sur votre implémentation du Labo 4, il vous faudra le corriger en fonction du retour que vous recevrez. L'essentiel c'est que votre labo 4 soit fonctionnel.

Étape 1 : Mettre en place PostgreSQL et créer le contexte

- Utiliser le fichier `docker-compose.yml` fournit pour lancer une instance de la base de données
- Utiliser le fichier `tables.sql` fournit pour créer la base de données
- Compléter le fichier `Data/DatabaseContext.scala`

Étape 2 : Établir la relation entre les tables et quill

- Créer les objets Scala représentant les tables

Étape 3 : Modifier votre implémentation actuelle pour utiliser la base de données

- Modifier les méthodes de `UsersInfo` et `MessagesInfo` pour agir sur la base de données.
- Ajouter une méthode `getAccountId` dans `UsersInfo` qui récupère l'id d'utlisateur depuis son nom d'utilisateur.

Étape 4 : Implémenter la suppression de l'historique des messages

- Un `GET` sur l'endpoint `/clearHistory` doit supprimer tout l'historique des messages.

Étape 5 : Implémenter replyTo, mention, exprType

Afin de permettre des requêtes analytiques intéressantes, nous avons enrichi la structure des messages avec des champs supplémentaires:

- `replyTo` Optionnel. Référence un autre message auquel répond le message courant. (Par example, le bot répond aux messages qui lui sont destinés)
- `mention` Optionnel. Référence un autre utilisateur tel que mentionné au début du texte du message. (Par example, le message "`@Julian Hello`" référence l'id du compte de Julian)
- `exprTypeId` Optionnel. Référence le type de la requête envoyé au Bot. (Voir les différents types de la table `exprtype` dans le fichier `tables.sql`)

Ces champs doivent être remplis correctement lors de l'ajout d'un message dans la base de donnée.

Étape 6 : Implémenter des requêtes analytiques

Dans le fichier `StatisticsInfo.scala` implémenter les requêtes suivantes (chaque requête correspond à une méthode) :

1. Récuperer le nombre de messages envoyés.
2. Récuperer le nombre de messages envoyés par l'utilisateur courant.
3. Récuperer, s'il existe, le nom de l'utilisateur ayant envoyé le plus de messages.
4. Récuperer, s'il existe, le nom de l'utilisateur ayant indiqué le plus de fois au Bot qu'il était affamé ou assoiffé. Retourner également ce nombre.
5. Récuperer, s'il existe, le nom de l'utilisateur le plus pauvre qui n'a jamais effectué une commande auprès du Bot.
6. Récuperer, s'il y a des messages, l'heure de la journée pendant laquelle le plus de messages a été envoyé. Retourner également le nombre de messages.
7. Récuperer, s'il y a des messages avec mention, les noms du couple d'utilisateurs dont le premier mentionne le plus le second. Retourner également le nombre de mentions.

L'affichage du résultats des requêtes vous est déjà fournit sur l'endpoint `/statistics`. Il vous faut seulement lier la page au fichier CSS.

Rendus

Pour assurer un suivi continu du laboratoire et pour nous permettre de vous donner des feedbacks au fur et à mesure, nous vous demandons de rendre votre code plusieurs fois.
Lorsque vous êtes prêt à faire un rendu, veuillez suivre les instructions suivantes (adapter au besoin):

1. Commiter le travail que vous voulez rendre (ex. `git commit`).
2. Tagger le commit actuel avec un tag annoté (ex. `git tag -a lab5stepN`). Comme commentaire, indiquez nous si vous avez eu un éventuel problème.
3. Pusher le tag qui vient d'être créé (ex. `git push origin lab5stepN`).

Vous devez effectuer les rendus suivants:

- Pour Dim. 7 Juin 2020, il faut faire les étapes 1, 2, 3 et 4. Utilisez le tag `lab5step4`.
- Pour Dim. 14 Juin 2020, il faut faire les étapes 5 et 6. Utilisez le tag `lab5step6`.