Totally Not Gaps (alias: KING)

> Auteurs: Bastien Potet & Doran Kayoumi

Lancement

La première chose à faire est de définir la configuration de la base de données dans un fichier `.env`. Vous pouvez simplement utiliser le fichier d'exemple que nous avons mis à disposition. Celui-ci contient déjà la configuration pour fonctionner avec le container Docker contenant la base de données.

```
$ cp .env.example .env
```

Maintenant que la configuration de la base de données est faite, vous pouvez démarrer la base de données ainsi que `Totally Not Gaps`.

```
$ docker-compose up -d
$ cargo run
```

> Note: Si vous décidez de ne pas utiliser notre "infrastructure", nous avons mis à disposition un fichier SQL `setup.sql` pour créer les tables nécessaires au bon fonctionnement de l'application et d'insérer des données de test.
>
> Note 2: Si vous utilisez une base de données autre que PostgreSQL, l'application risque de ne pas fonctionner, car nous avons configuré notre ORM pour utiliser PostgreSQL

Changement effectués

En bref

- Ajout d'une base de données postgresql
- Ajout d'un système d'authentification
- Ajout d'un système d'autorisation
- Ajout de logging
- Ajout de tests

Base de données

Nous avons utilisé une base de données postgresql pour stocker notre base de données. Nous avons également utilisé un
ORM qui est `diesel` (https://diesel.rs/) pour nous faciliter les tâches de sélection, de modification et d'insertion de données.

Système d'authentification

Nous avons ajouté un système d'authentification qui a pour unique but de connecter l'utilisateur. Nous avons ajouté des validations d'I/O pour cela nous avons utilisé le crate `read_input` qui nous aide notamment à éviter des problèmes de conversion, puisque celui-ci va refuser une chaîne de caractères, si nous attendons un nombre flottant. Pour éviter qu'un méchant pirate

vienne dump notre base de données et puisse voir les mots de passe des élèves et des enseignants, nous avons donc hashé les mots de passe avec `argon2id` du crate `sodiumoxide`.

Système d'autorisation

Nous avons ajouté un système d'autorisation pour pouvoir vérifier lors d'une action, comme l'ajout de note à un élève si celui-ci est réellement autorisé à effectuer une telle action. Nous avons utilisé le crate `casbin` pour pouvoir gérer des fichiers de contrôle d'accès. Nous avons décidé dans ce projet d'utiliser le modèle RBAC puisqu'il y existe deux rôles bien définis :

- L'enseignant
- L'élève

Il serait ainsi possible à l'avenir d'ajouter de nouveaux rôles, comme l'administrateur.

Logging

Dans le but de suivre le déroulement de notre logiciel, nous avons ajouté du logging à notre programme. Nous avons utilisé le package `simple_logger` pour pouvoir afficher les logs dans le programme. Pour le placement de nos logs, nous avons décidé de les placer là où nous pourrions afficher le plus d'information (i.e. essayé de répondre aux questions `When, What, Where, Who ?`).
Par exemple, si une erreur survient lors de l'insertion du note, nous avons placé un log dans la fonction `enter_grade`, car l'on peut afficher l'enseignant ayant voulu insérer une nouvelle note, l'élève à qui la note a été attribué ainsi que la note. Un autre emplacement aurait été de placer le log dans la fonction du repository qui insère l'entrée dans la base de données, mais il nous manquerait l'information de quel utilisateur a effectué la demande d'insertion.

En plus de la tentative de réponse aux questions `When, What, Where, Who ?` et de la date ajouté automatiquement par `simple_logger`, nous avons aussi ajouté le nom de la fonction qui a généré l'erreur.e.g. `main (see_grades) - Alice tried to acces grades of a non-existing student or a teacher (Bob)`

Tests

Pour vérifier le bon fonctionnement de notre logiciel, nous avons utilisé le crate `rstest` qui facilite grandement le testing. Nous avons également utilisé le crate `mockall` afin de mock notre base de données pour que l'on puisse tester le bon fonctionnement de la récupération et des insertions des données dans la base de données.

Possible amélioration

Le système d'authentification n'est pas capable de prendre en compte l'enregistrement d'un nouvel utilisateur. Nous nous sommes posé la question si il était nécessaire d'ajouter la prise en compte d'un enregistrement, puisqu'il faudra définir si l'utilisateur enregistré est enseignant. Il faudrait alors dans ce cas ajout un dernier rôle comme administrateur qui aurait la capacité de modifier ou d'ajouter des utilisateurs.

Il aurait également possible d'ajouter un système de 2FA pour pouvoir vérifier que l'utilisateur qui se connecte est le bon.

Les logs devraient se trouver dans un fichier dédié au logging, du au fait que notre application est en mode terminal et par conséquent, les entrées et sorties utilisateurs se mélangent avec la sortie de logging. Si celle-ci était sous forme web, desktop ou mobile, il serait possible de laisser les loggings s'afficher dans le terminal, mais il faudrait tout de même ajouter la sortie dans un

fichier de log.

Dans notre implémentation, nous définissons le rôle d'un utilisateur dans la base de données et dans le fichier de policy. Avoir cette redondance risque de causer des problèmes de cohérences si l'on ne fait pas attention lors de l'insertion d'un nouvelle utilisateur.trice. De plus, ce système n'est pas du tout adapté pour des systèmes à plus grandes échelles (e.g. Gaps).