

Rapport - Projet Serveur

Des élèves et des gommettes

Année : 2021-2022

Table des matières

1	Introduction	1
2	Partie I	1
2.1	Les fonctionnalités qui nous semblent importantes	1
2.2	Les classes principales de l'application	2
2.3	Les tables dont nous avons besoin	2
2.4	Les technologies utilisées	2
3	Partie II	2
3.1	Fonctions Principales C-M-D	2
3.1.1	Create()	2
3.1.2	Delete()	3
3.1.3	Modify()	3
4	Jointure de la EleveGom avec les autres	3
5	Session et Cookies	3
6	Liste des URI de notre programme	3
7	Conclusion	4

1 Introduction

Pour aider Mme Myrtille et M. Framboisier, nous avons conçu une application qui s'occupera de la gestion des gommettes attribuées à chaque élève. La majorité des fonctionnalités de cette application ne pourra être utilisée que par un professeur connecté.

Ici, par défaut, vous pouvez vous connecter en tant que professeur avec le numéro d'identification **1** et son mot de passe associé **mdp123**.

Pour aller sur la page d'accueil, il faut écrire dans la barre de recherche "localhost :8081/index" après avoir démarré le serveur avec `./gradlew run`.

2 Partie I

2.1 Les fonctionnalités qui nous semblent importantes

Voici la liste des fonctionnalités que nous avons implémentées :

- Les professeurs peuvent se connecter (on ne peut ajouter, modifier ou supprimer des gommettes que lorsqu'on est connecté) ;
- Les professeurs peuvent lister, ajouter, supprimer ou modifier un élève ;
- Les professeurs peuvent lister, ajouter, supprimer ou modifier une gommette ;
- Les professeurs peuvent ajouter ou supprimer une gommette à un élève, avec une date et un motif ;
- Les professeurs peuvent lister les gommettes attribuées à un élève, et voir qui a donné chaque gommette ;
- Tout le monde peut voir la liste des gommettes configurées, et la liste des gommettes de chaque élève.

Pour utiliser l'intégralité du site, il faut donc être connecté en tant que professeur. En étant connecté vous pourrez accéder aux fonctions principales du site que sont créer, modifier et supprimer des élèves, des gommettes et attributions de gommettes à un élève.

2.2 Les classes principales de l'application

Nous avons besoin de différentes classes principales pour le bon fonctionnement de celui-ci.

- La classe professeur, représentée par les *ProfEntity()*
- La classe gomme, représentée par les *GomEntity()*
- La classe élève, représentée par les *eleveEntity()*
- La classe Attribution d'une gomme à un élève, soit *EleveGomEntity()*

2.3 Les tables dont nous avons besoin

Pour chaque classe, nous avons créé une table assignée.

Il y a donc une table *eleveGom*, une table professeurs, une table eleves et une table goms.

La table **eleveGom** va avoir des jointures avec la table **eleves**, **professeurs** et **goms** car une attribution de gomme à un élève doit contenir des informations sur l'élève qui a reçu la gomme, sur le professeur qui lui a donné, et sur la gomme elle-même.

La table **professeur** contient les propriétés : id, nom, prénom et mot de passe. Pour se connecter, il aura besoin de son identifiant et de son mot de passe.

La table **élève** contient les propriétés : id, nom et prénom.

La table **gomme** quant à elle est composée des colonnes id, description, couleur.

Enfin, la table d'**attribution eleveGom** est composée d'un id, de la date d'attribution, d'un motif d'attribution, de l'id de l'élève qui reçoit la gomme, de l'id du professeur qui lui donne, et de l'id de la gomme.

2.4 Les technologies utilisées

Pour créer notre application, nous allons utiliser java Spark (qui gèrera notre serveur web) et FreeMarker, pour formater à l'aide de template nos données. Nous utiliserons une base de donnée H2.

Notre code suit le modèle MVC :

- Le modèle encapsule les données, contient les règles métiers, et est composé des fichiers GUI, ENTITY, DAO et CORE.
- Le contrôleur est représenté par le fichier StartServeur
- La vue est représentée par les fichiers ftl dans le dossier view.

3 Partie II

3.1 Fonctions Principales C-M-D

Pour chaque élève et chaque gomme, il est demandé de pouvoir (en tant que professeur) modifier leur table de données respectives, en créant, modifiant ou supprimant certains tuples. Cela peut se faire par l'intermédiaire de trois types de fonctions : Create, Delete et Modify.

Ici, nous allons montrer un échantillon de ces fonctions pour les "Eleves".

3.1.1 Create()

```
1 public EleveEntity create(EleveEntity obj) {
2     Connection connection = _Connector.getInstance();
3     try {
4         PreparedStatement statement;
5         statement = connection.prepareStatement("INSERT INTO
6         eleves(firstName, lastName) VALUES(?, ?);");
7         statement.setString(1, obj.getFirstName());
8         statement.setString(2, obj.getLastName());
9         statement.executeUpdate();
10    } catch (SQLException e) {
11        System.out.println(e.toString());
12        throw new RuntimeException("could not create database !");
13    }
14    return obj;
15 }
```

L'objectif de cette fonction est de **créer un élève** dans une table existante, en ajoutant ses valeurs aux colonnes associées à la table : *firstName* et *lastName*.

Pour cela, on utilise *PreparedStatement*, qui va nous permettre d'exécuter une requête SQL paramétrée.

3.1.2 Delete()

Pour supprimer un élève on utilise son numéro identifiant, comme on peut le voir dans la requête suivante :

```
1 statement = connection.prepareStatement("DELETE FROM eleves WHERE id=?");
2 statement.setString(1, ""+obj.getId());
3 statement.executeUpdate();
```

3.1.3 Modify()

Pour modifier les informations d'un élève en fonction de son id, on utilise la requête SQL suivante :

```
1 this.connect.prepareStatement
2 ("UPDATE eleves SET firstName=?, lastName=? WHERE id=?");
```

4 Jointure de la EleveGom avec les autres

La table **eleveGom** contient le numéro de l'attribution, l'id de l'élève, l'id du professeur et l'id de la gomme.

Ayant besoin d'informations sur l'élève, le professeur et la gomme, nous avons simulé une jointure. Cela en codant certaines fonctions comme `getOneProf(int idProf)`, elle va chercher dans la table de données les informations sur le professeur que l'on veut à partir de son id. La fonction va renvoyer une `ProfEntity`, et va être appelée dans `EleveGomEntity`. On peut donc obtenir comme attribut dans `EleveGomEntity` la `ProfEntity` correspondant au professeur qui a donné la gomme.

On fait la même chose pour l'élève, grâce à la fonction `getOneEleve`, et pour les gommes avec la fonction `getOneGom`.

5 Session et Cookies

Pour se connecter, on utilise un système de session et de cookies. En effet, une fois sur la page de connexion `/login`, le professeur va remplir un formulaire demandant son id et son mot de passe. Une fois envoyé, on vérifie que le couple (id, mdp) se trouve dans la base de données grâce à la fonction `LoginCore.authentication(id, password)`.

Si le couple est dans la table professeur, on crée un cookie avec l'id de l'utilisateur, et on l'envoie au client. Ce cookie sera renvoyé à chaque requête, et lui donnera accès à l'intégralité des fonctionnalités de l'application.

Sinon, on crée un cookie "incorrect" qui va indiquer que la page html du formulaire doit être réinitialisée et réaffichée, avec en plus un message d'erreur expliquant que l'id ou le mot de passe est incorrect.

Lorsque l'utilisateur va vouloir se déconnecter, il va devoir cliquer sur le bouton déconnection, qui va appeler la page `/removeCookie`. Celle-ci va effacer l'intégralité des cookies envoyés au client, puis rediriger le professeur sur la page du formulaire `/login`.

6 Liste des URI de notre programme

- `/index` : page principale de l'application
- `/removeCookies` : supprime l'intégralité des cookies
- `/eleves` : espace élève
- `/eleves/modify` : permet de modifier un élève
- `/eleves/create` : permet de créer un élève
- `/eleves/delete` : permet de supprimer un élève
- `/professeurs` : espace professeurs
- `/login` : se connecter
- `/gommettes` : espace gommettes
- `/gommettes/delete` : permet de supprimer une gomme
- `/gommettes/create` : permet de créer une gomme
- `/gommettes/modify` : permet de modifier une gomme
- `/elevesGom` : espace attribution d'une gomme à élève par un professeur
- `/elevesGom/create` : un professeur connecté peut assigner une gomme à un élève
- `/elevesGom/delete` : permet de supprimer une attribution

/elevesGom/recherche : permet d'afficher toutes les gommettes possédées par un élève a partir de son id

7 Conclusion

Pour conclure, grâce à cette application, les professeurs présents dans la base de données ont accès à l'intégralité du site, quand les autres utilisateurs ne peuvent que voir les gommettes configurées, et la liste des gommettes de chaque élève.