Colin Troisemaine
Guillaume Bouchard

03/12/2019

Liens SGBD-LOO : Dossier partagé

Note : Pour accéder à l'application sans connexion à la base de données, il existe un compte dont le nom de compte et le mot de passe est **root**.

A) Contraintes :

1) Taille des fichiers :

La taille maximum d'un fichier importé varie en fonctions de trois paramètres : La limite imposée par le SGBD (par exemple 2048Ko sur phpMyAdmin de base), la taille maximum des fichiers manipulés dans le projet java et enfin la taille limite du type de BLOB utilisé (ici nous avons choisi un "Medium BLOB", donc 16Mo). Ainsi si l'utilisateur essaie de télécharger un fichier plus grand que cette limite sur la base, il ne se passera rien.

2) Noms d'utilisateur unique :

Il est impossible de créer deux comptes utilisateurs avec le même login. En effet, si nous l'autorisions, un utilisateur voulant créer un compte avec le même login qu'un autre recevrait une erreur si celui-ci essayait de créer son compte avec le même mot de passe puisqu'il serait alors impossible de différencier les deux comptes à la connexion. Cela lui révélerait donc le mot de passe de l'autre utilisateur.
Nous avons donc décidé de limiter l'usage d'un même login à un seul et unique compte.

3) Noms de dossiers :

Nous avons choisi de faire en sorte qu'il soit impossible d'avoir deux dossier avec le même nom au sein d'un même dossier pour pouvoir les différencier facilement. Il est en revanche possible que deux fichiers aient le même nom dans un même dossier.

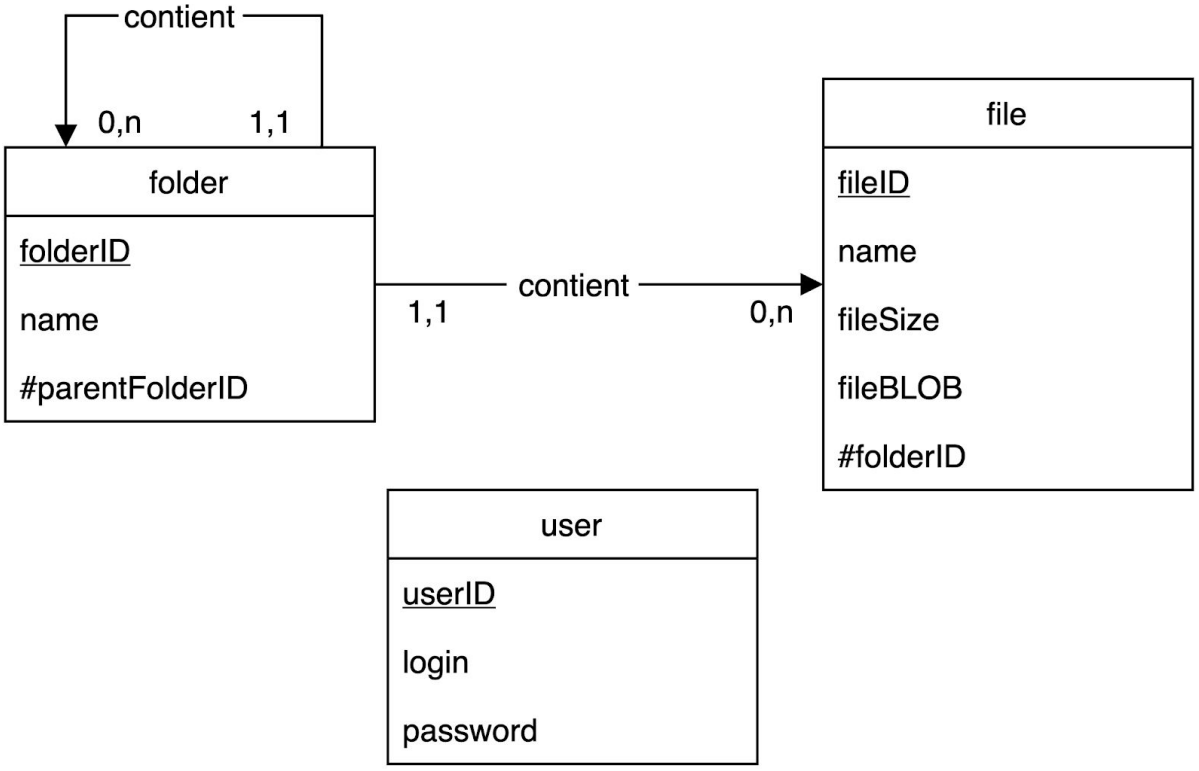
4) Connexion à la base de données :

Il est possible de lancer l'application (que ce soit la fenêtre de login ou l'explorateur de fichier lui-même) sans avoir de connexion à la base de données. Cependant dès qu'une action requérant une connexion est effectuée, l'application essayera de se connecter. Si la connexion est impossible, un pop-up d'erreur apparaîtra et l'action sera annulée.
Comme nous l'avons précisé plus haut, utiliser le compte root/root permet la connexion sans faire appel à la base de données.

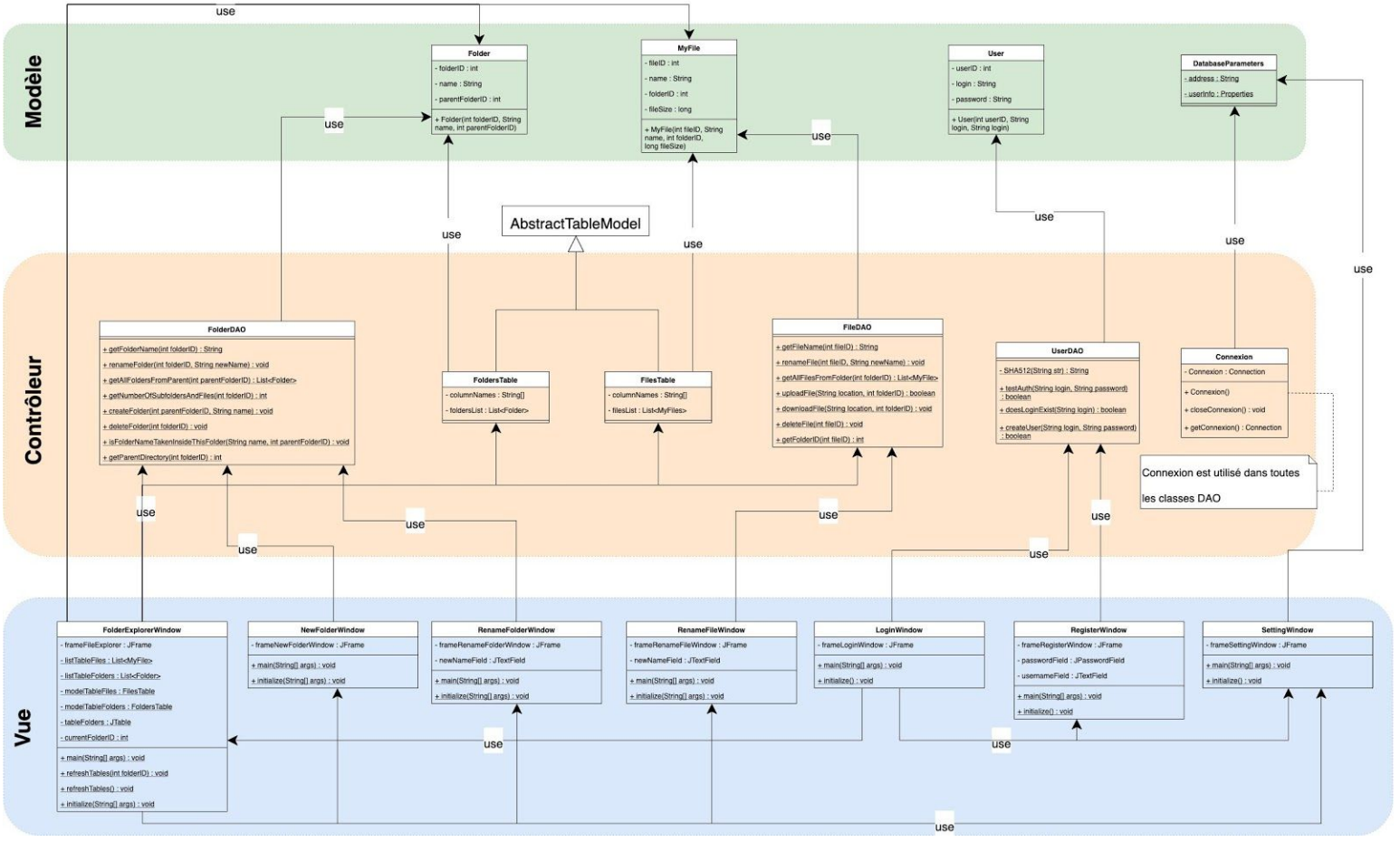
5) Moteur de stockage de la base de données :

Pour gérer le stockage des fichiers, il est nécessaire que le moteur de stockage de la base de données soit InnoDB.

B) Modèle conceptuel de données de la BDD :



C) Architecture du logiciel : (Voir fichier png joint si la qualité est insuffisante)



D) Spécifications techniques :

Identifiants des utilisateurs :

Tous les identifiants sont en auto-increment dans notre base de donnée. De cette manière, l'application Java n'a pas besoin d'avoir à les gérer et ils sont créés automatiquement (de cette manière on évite facilement les doublons).

Affichage séparé dossier, fichier :

Par souci de simplicité, nous avons décidé de séparer les dossiers et les fichiers dans notre affichage. Ainsi il y une JTable pour les dossiers et une JTable pour les fichiers.

Adresse de la bdd :

Nous avons fait le choix de mettre une adresse fixe au démarrage de l'application. Il est cependant possible de la modifier à partir de la fenêtre de connexion ou de l'explorateur de fichiers. Une fois modifiée, l'adresse restera ainsi jusqu'à la fermeture de l'application car elle n'est pas sauvegardée localement.

Nous pourrions dans le futur de cette application permettre à l'utilisateur d'enregistrer cette adresse dans un fichier à côté de l'exécutable.
(Le compte utilisé pour se connecter à la base de données est également paramétrable).

Fonctions de la fenêtre de connexion :

La fenêtre de connexion permet de s'enregistrer à l'aide du bouton "register". Une fois cliqué, une nouvelle fenêtre s'ouvre et demande de renseigner un nom d'utilisateur et un mot de passe. N'importe qui peut donc s'enregistrer et accéder au dossier partagé.

La fenêtre de connexion permet également de se connecter en renseignant les champs "username" et "password". Si l'utilisateur clique sur le bouton "Connection", une requête sera envoyée à la base de données pour vérifier si l'utilisateur existe ou non avant de le faire accéder à l'explorateur de fichier.
Enfin il est possible d'accéder à la fenêtre de paramétrage de l'adresse de la base.

Fonctions de la fenêtre principale :
Pour naviguer à travers tous les dossiers et fichiers de notre base de données, nous avons implémenté les fonctions suivantes :

- Home : Cette fonction renvoie au dossier parent (ou root) du dossier partagé.
- Retour : Renvoie au dossier parent du dossier affiché.
- Refresh : Permet de mettre à jour manuellement les dossiers et fichiers actuellement affichés (peut être utile dans le cas d'un upload long ou encore de l'ajout d'un fichier à la base de données par un autre utilisateur).

Pour manipuler les dossiers : nous avons défini 4 boutons différents :

- "move to folder" : Permet d'accéder au dossier sélectionné. Cette méthode met donc à jour les dossiers et fichiers affichés en conséquence.
- "create folder" : Permet de créer un nouveau dossier dans le dossier courant.
- "delete folder" : Permet de supprimer le dossier sélectionné ainsi que tous ses sous-dossiers ou fichiers.
- "rename folder" : Permet de changer le nom du fichier sélectionné.

Pour manipuler les fichiers :
Nous avons défini 4 boutons différents :

- “download file” : Une fois un dossier choisi, permet de choisir un fichier de destination et de télécharger le fichier dans celui-ci.
- “upload file” : Une fois le fichier choisi, permet de télécharger celui-ci sur la base de données dans le dossier actuellement ouvert.
- “delete file” : Permet de supprimer le fichier sélectionné de la base de données.
- “rename file” : Permet de changer le nom du fichier sélectionné.

Le nom de l’utilisateur actuellement connecté et le nom du dossier actuellement ouvert sont également affichés en haut à gauche de l’explorateur de fichiers.

Actualisation des données affichées :

À chaque fois que l’utilisateur utilise l’une des fonctions permettant de naviguer à travers les dossiers et fichiers, une nouvelle requête est envoyée à la base et récupère tous les dossiers/fichiers actuellement présents sur la base.
Ainsi, si un utilisateur envoie sur la base un nouveau fichier sur un dossier qu’un autre utilisateur affiche actuellement, il ne sera pas immédiatement affiché. Pour régler ce problème, nous avons envisagé deux solutions :

- Soit mettre à jour le dossier affiché automatiquement toutes les secondes.
- Soit ajouter un bouton permettant de mettre à jour manuellement le dossier affiché.

Nous avons donc opté pour la 2ème méthode, car nous craignons que la première ne crée trop de requêtes “inutiles” sur la base de données.

Fonctions de suppression :

FolderDAO.deleteFolder(int folderID) : Pour implémenter cette méthode, nous avons utilisé une approche récursive : On commence par supprimer tous les fichiers du dossier. Puis pour chacun des sous-dossiers du dossier, on regarde s'ils sont vides (dans ce cas là on les supprime) et sinon on applique une nouvelle fois la méthode deleteFolder(int subFolder) sur chacun des sous-dossiers. Puis on termine par la suppression du dossier lui-même.

FileDAO.deleteFile(int fileID) : Cette fonction est bien plus simple que la précédente puisqu’il n’y a pa de contraintes de sous-dossier. Une simple requête “DELETE FROM File” est donc envoyée à la base de données.

Affichage des erreurs :

Pour que l’affichage des erreurs, nous avons évité de faire apparaître une nouvelle fenêtre indiquant l’erreur car nous avons estimé cette méthode d’affichage envahissante puisqu’elle requiert que l’utilisateur clique par exemple sur “Ok” pour la faire disparaître à chaque erreur.

Nous avons donc choisi de faire apparaître les erreurs dans la fenêtre elle même lorsqu’il le fallait. Ainsi, l’utilisateur en prend compte sans avoir à effectuer d’action supplémentaire.