GL1 & CWA : Projet

Document de conception

Groupe 1 :

CHOUMILOFF Sacha

EL OUALI Ayman

ESCARBAJAL Quentin

GUILLEMIN Sébastien

TRIDARD Jérémy

VERNEY Mathieu

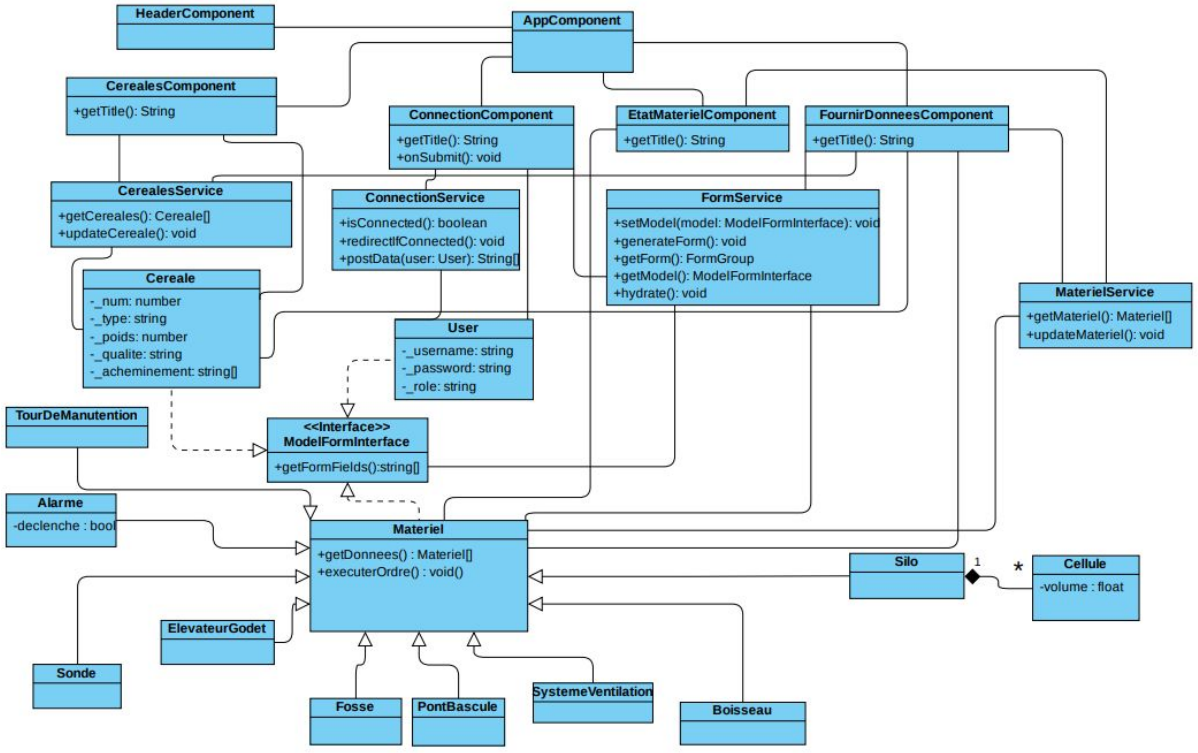
ZOUMAROU WALIS Faïzath Jedida

Ce document présente nos diagrammes de classe ainsi que le modèle conceptuel de notre base de données.

I/ Diagrammes de classes :

Nos diagrammes de classe seront sûrement amenés à évoluer. Nous avons fait deux diagrammes de classes : un pour le serveur NodeJS et un pour la partie Angular.

1) Angular :



La classe principale est “appComponent”. Pour chaque composant Angular il y a une classe, utilisée par “appComponent”. Le composant “HeaderComponent” permet d’afficher la barre de navigation. Chaque composant permet de gérer l’affichage d’un morceau de la page :

- CerealesComponent permet l’affichage des informations sur les céréales.
- EtatMaterielComponent permet d’afficher les informations sur le matériel.
- ConnectionComponent permet aux utilisateurs de se connecter (formulaire de connexion).

- FournirDonneesComponent permet à l'administrateur de saisir des informations sur les céréales et sur le matériel.

Ces composants utilisent des services pour pouvoir interagir avec le serveur ou effectuer des traitements redondants (gestion de formulaires par exemple).

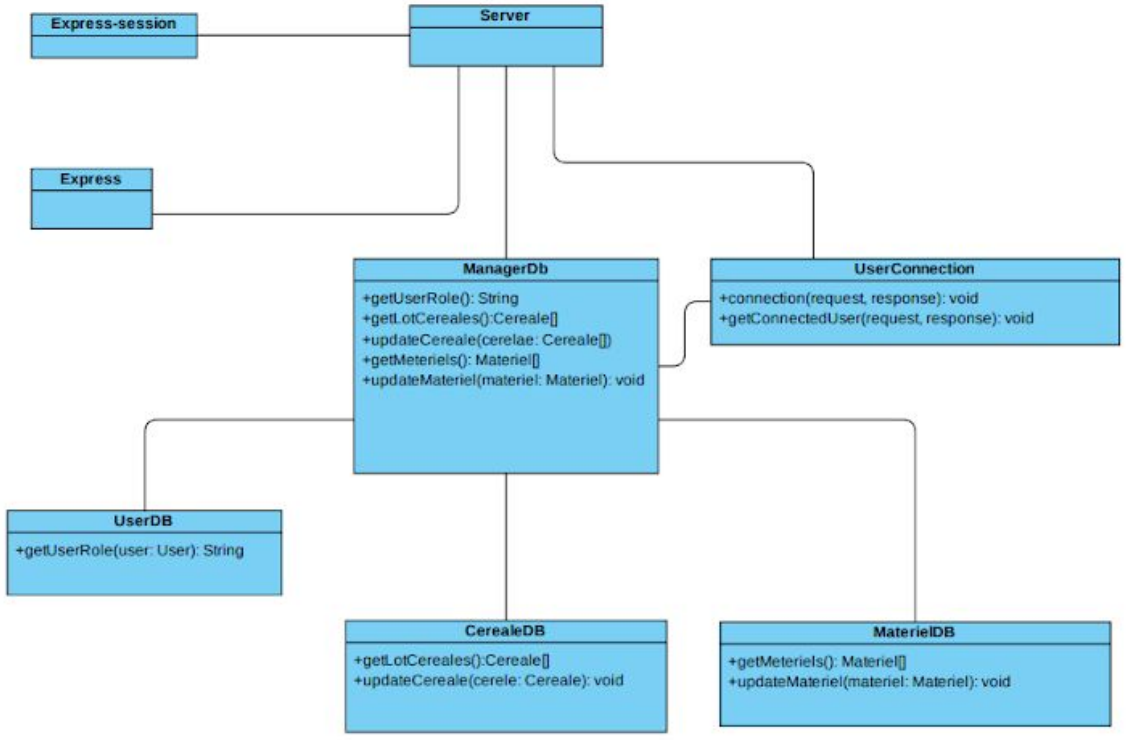
Pour la connexion, on utilise une classe "User" (permettant de représenter un utilisateur), qui possède un nom d'utilisateur, un mot de passe et un rôle ("user" ou "admin"), qui permettra de savoir si l'utilisateur connecté peut fournir les données au serveur ou non (seul un administrateur peut fournir les données).

Le service "FormService" permet de gérer les formulaires. Chaque classe, dont les informations doivent être saisies dans un formulaire, doit hériter de l'interface "ModelFormInterface". Cette interface définit une méthode "getFormFields()" qui retourne la liste des informations devant être saisie dans le formulaire. Le service connaît ainsi quelles informations doivent être fournies et peut vérifier si elles sont valides.

Une classe "Materiel" permet de gérer toutes les autres formes de matériel comme la "Sonde", "Fosse" ... (pour les différents matériels, nous ne sommes pas encore sûrs de quels attributs et méthodes seront nécessaires). Le service "MaterielService" permet de récupérer les données sur les différents matériels depuis le serveur et de mettre à jour ces données.

2) NodeJS :

La partie serveur NodeJS est principalement utilisée pour mettre à jour la base de données et authentifier les utilisateurs.



La classe principale est la classe “Server”, elle utilise les classes “Express” et “Express-session” afin de gérer les requêtes HTTP et les sessions.

La classe “ManagerDb” est un service permettant d’interagir avec la base de données. Cette classe utilise les classes “UserDB”, “CerealeDB” et “MaterielDB” qui gère les interactions avec les tables de la base de données correspondante.

Les classes “User”, “Cereale” et “Materiel” sont les mêmes que les classes utilisées dans la partie Angular.

La classe “UserConnection” permet de connecter un utilisateur (vérifier si les informations fournies sont correctes et créer une session pour l’utilisateur).

II/ Modèle conceptuel

Voici le modèle relationnel de notre base de données. Celle-ci sera implémentée par des fichiers JSON, chaque relation aura un fichier (un pour “Utilisateur”, un pour “Cereale” ...) .

Note: Dans le modèle ci-dessous, le nom de la relation est en gras et ses attributs sont entre parenthèses. Les attributs soulignés sont les clefs primaires. La flèche représente un identifiant externe.

Utilisateur(username, password, role)
Cereale(num, type, poids, qualite, acheminement)
Materiel(id, nom)

Alarme(id_materiel, declenche)

Les attributs des relations correspondent aux attributs des classes. Ainsi, comme nous ne sommes pas encore fixés sur comment implémenter certains points de l’application (notamment pour le matériel), ce modèle relationnel n’est pas définitif, nous serons amenés à ajouter ou modifier des relations.

Cependant, pour chaque matériel, le principe sera le suivant : la relation “Materiel” contiendra l’identifiant et le nom d’un matériel. Suivant le type du matériel la relation du matériel correspondante (“Alarme”, “Silo” ...) contiendra une ligne avec le même identifiant plus les informations spécifiques à ce matériel. Ainsi, chaque relation d’un matériel possède un identifiant externe (id_materiel) sur la relation “Materiel”.