ArcelorMittal

Rapport du cas d'étude : ArcelorMittal

Dossier de conception

Louis BERTHIER – Nicolas LYOT – Adrien RUGGIERO – Nathan WITKOWICZ

TABLE DES MATIERES

I. Le dossier de conception 3
II. Le modèle d'architecture global 3
III. Le modèle d'architecture détaillé 4
1. Les diagrammes de classes 4
A. Le package « data_types » 4
B. Le package « model » 5
C. Le package « controller » 6
D. Le package « view » 6
2. Les diagrammes de séquence 7
A. Connexion à l'application 7
B. Affichage des graphiques et droits d'un technicien 8
C. Modification des accès à un atelier de production et du mode de calcul 8
D. Création d'un utilisateur 9
E. Administration et modification des droits 9
IV. Le modèle conceptuel de données 10

I. Le dossier de conception

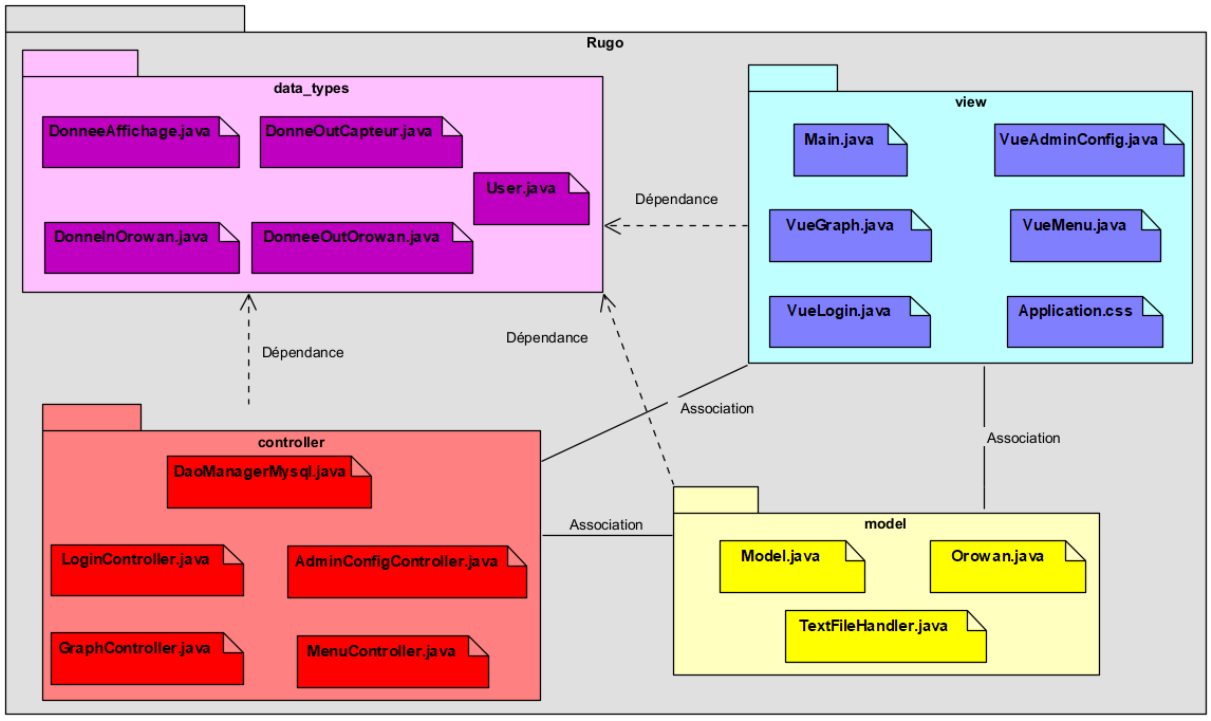
Ce document a pour but de présenter une **vision d'ensemble et globale** ainsi qu'une **vision plus approfondie** sur le déroulement des étapes au sein du projet *(Rugo)* que nous proposons en réponse à la problématique d'ArcelorMittal.

Voici l'ensemble des documents qui seront proposés par la suite afin de présenter ce dossier de conception :

- Un modèle d'architecture global
- Un modèle d'architecture détaillé
- Un modèle conceptuel de données

II. Le modèle d'architecture global

Voici le **diagramme de packages** que nous proposons pour représenter une **architecture générale** de Rugo :



On retrouve une architecture **modèle-vue-contrôleur** (MVC) classique à laquelle nous venons rajouter un nouveau package : « data_types » qui va nous permettre de **clairement définir les données** que nous traitons ainsi que les **instances associées**.

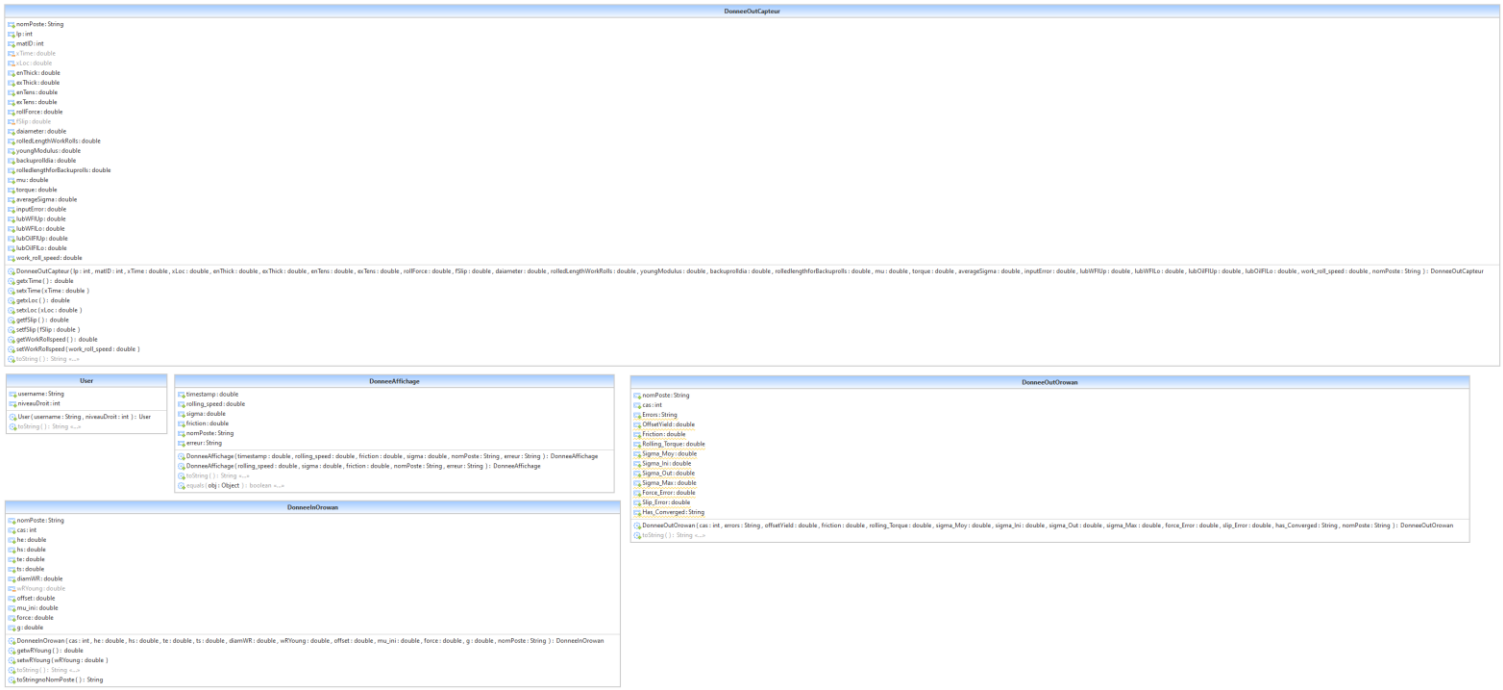
III. Le modèle d'architecture détaillé

Afin de représenter une **architecture plus détaillée** de notre projet, nous proposons de nous intéresser **plus spécifiquement aux packages** présentés précédemment ainsi qu'à différents **diagrammes de séquence** pour étudier **les fonctionnalités** et **le déroulement de notre application**.

1. Les diagrammes de classes

A. Le package « data_types »

Voici le **diagramme de classes** que nous proposons pour représenter l'**architecture détaillée** de notre nouveau package « data_types » :



Même remarque que pour le cahier des charges simplifié, si l'image n'est pas assez lisible, elle est accessible et téléchargeable ici : data_types.png

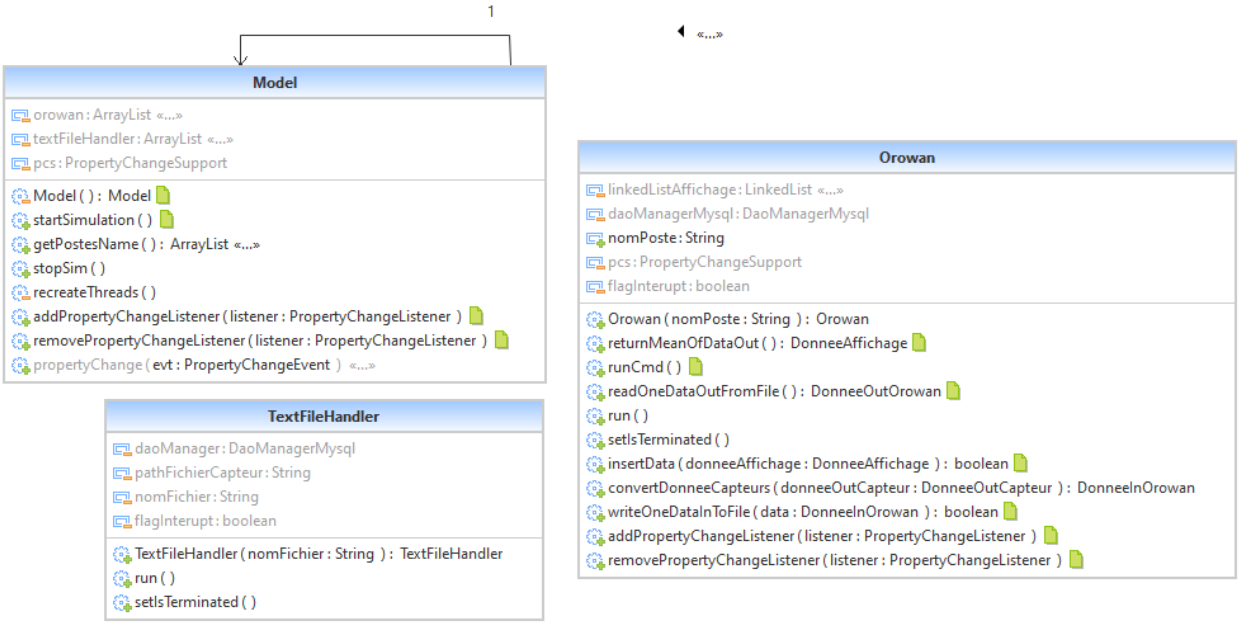
Nous distinguons **5 classes** dont 4 concernent uniquement **les données** que nous traitons. La dernière correspond à **l'instanciation d'un utilisateur** lorsque ce dernier est créé notamment avec son nom et son statut *(technicien ou ingénieur)*. Les autres classes correspondent à **l'instanciation des données** que nous utilisons lors du processus pour encapsuler les données. Par ordre chronologique, nous avons :

- Les **DonneOutCapteur** qui correspondent aux **données brutes directement issues des capteurs** de chaque atelier de production
- Les **DonneInOrowan** qui correspondent aux **informations conservées parmi les DonneOutCapteur** afin d'être utilisées pour le calcul qu'Orowan mène

- Les **DonneOutOrowan** qui sont les **données en sortie d'Orowan donc post-calcul**. On obtient notamment la rhéologie, la vitesse du rouleau et le coefficient de friction comme demandé avec la méthode de calcul inverse *(Back)*
- Les **DonneeAffichage** qui sont tout simplement les mêmes grandeurs que DonneOurOrowan mais nous réalisons **une moyenne chaque seconde** *(soit sur 5 valeurs généralement)*

B. Le package « model »

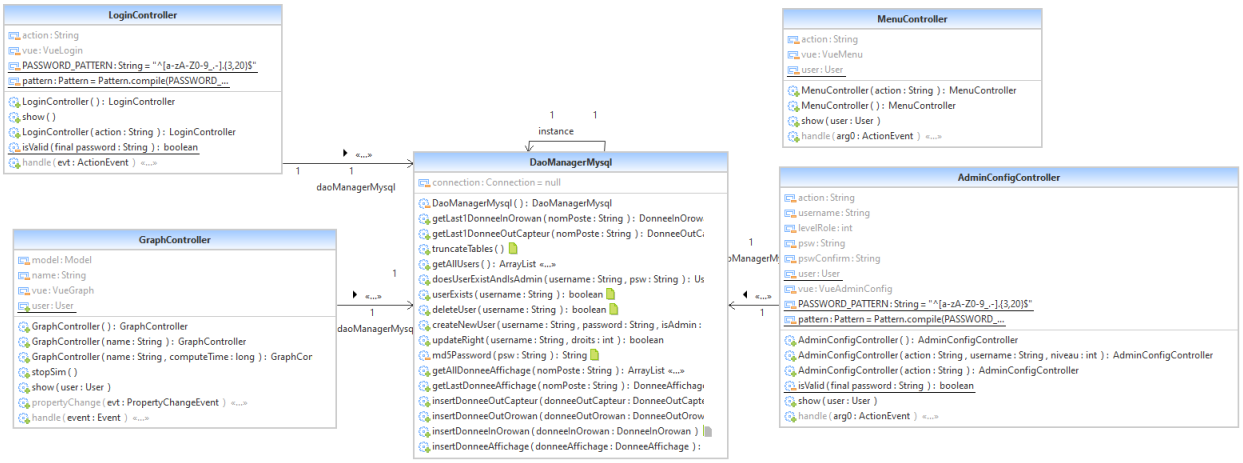
Voici le **diagramme de classes** que nous proposons pour représenter notre package « model » :



Nous retrouvons donc 3 classes qui vont gérer **toute la partie modèle** de notre système Rugo. Tout d'abord, la classe *TextFileHandler* nous permet de **simuler les capteurs** et **l'envoie des données par les capteurs** à travers la lecture de fichiers texte. On a également la classe *Orowan* qui va nous permettre de réaliser **un large panel d'opérations** comme **l'insertion de données dans une table** à travers un système de base de données relationnelle, **le calcul des grandeurs en sortie** à l'aide du modèle mathématique intégré ou encore **la conversion des données** notamment avec la sélection des données à utiliser pour le calcul d'Orowan à partir des données capteurs *(DonneeOutCapteur => DonneeInOrowan)*. Ainsi la classe *Model* correspond à **« l'intelligence » du système**. En effet, c'est à partir de cette classe que toutes les actions et tâches sont réalisées. De fait, elle **gère les « threads »** associés à la lecture des données capteurs **ainsi que la gestion des instances d'Orowan** et plus précisément, une par poste.

C. Le package « controller »

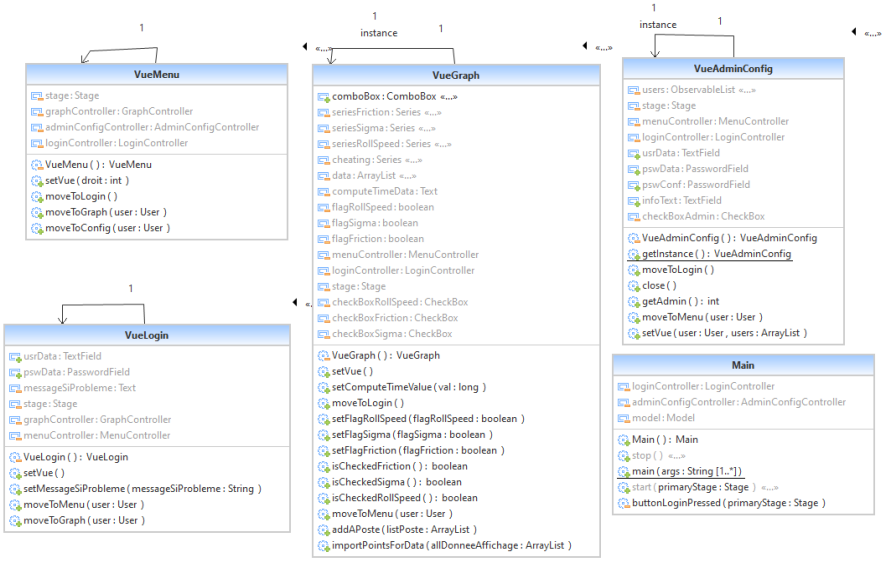
Voici le **diagramme de classes** que nous proposons pour représenter l'architecture de notre package « controller » :



Nous retrouvons ici **5 classes** plus ou moins mis en relation à travers la classe *DaoManagerMysql*. Globalement, ce sont les classes qui vont permettre **de coordonner les éléments appelés** à la suite d'une interaction de la part de l'utilisateur dans la partie interface graphique. On peut prendre l'exemple du *LoginController,* c'est lui qui permet de **définir les règles associées** à la partie connexion d'un utilisateur notamment avec **la restriction sur les caractères spéciaux** ou encore **sur le nombre de caractères requis** pour créer un compte. Finalement, ce package permet de **faire le lien entre la vue et le modèle**. Ici, le DaoManager qui est un singleton permet de **gérer tout l'accès à la base de données**.

D. Le package « view »

Voici le dernier **diagramme de classes** que nous proposons. Il s'agit du package « view » et qui représente toute la partie associée à l'interface graphique de notre application :



On retrouve ici **5 classes** dont 4 vont gérer entièrement **la partie graphique**, et la dernière, la classe *Main* qui va tout simplement nous permettre **d'initialiser Rugo**.

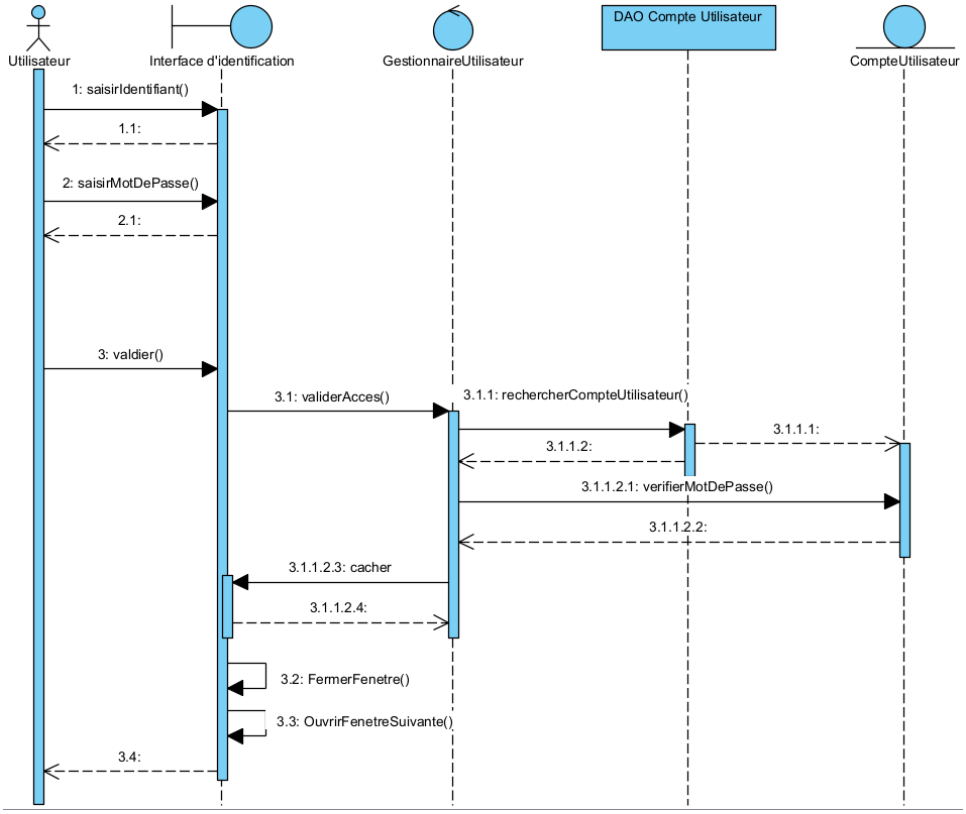
Globalement, on dispose de différentes interfaces graphiques avec par ordre d'apparition :

- **La page d'accueil** à la suite du lancement de l'application
- **La page de connexion** afin que chaque utilisateur puisse renseigner son identifiant et son mot de passe
- **La page de droits** avec les choix suivants : afficher les graphiques ou configurer les droits *(uniquement dans le cas d'un administrateur)*
- **La page des graphiques** qui permet d'afficher le temps de calcul d'Orowan, l'évolution de la rhéologie, de la vitesse du rouleau et du coefficient de friction (simultanément ou non) ou encore de choisir l'atelier de production que l'on souhaite étudier.
- **La page des utilisateurs** qui donne accès à un des administrateurs la liste des utilisateurs renseignées dans la BDD. Ce dernier à la possibilité d'en rajouter, d'en supprimer ou tout simplement de passer un ouvrier administrateur et inversement. Il est cependant impossible de se modifier soi-même pour **éviter le cas où il n'y aurait plus d'administrateurs**.

Les différentes vues sont des singletons et **chacune d'elles est associée à son contrôleur**. De plus, une vue instancie les contrôleurs des pages vers lesquelles on peut naviguer.

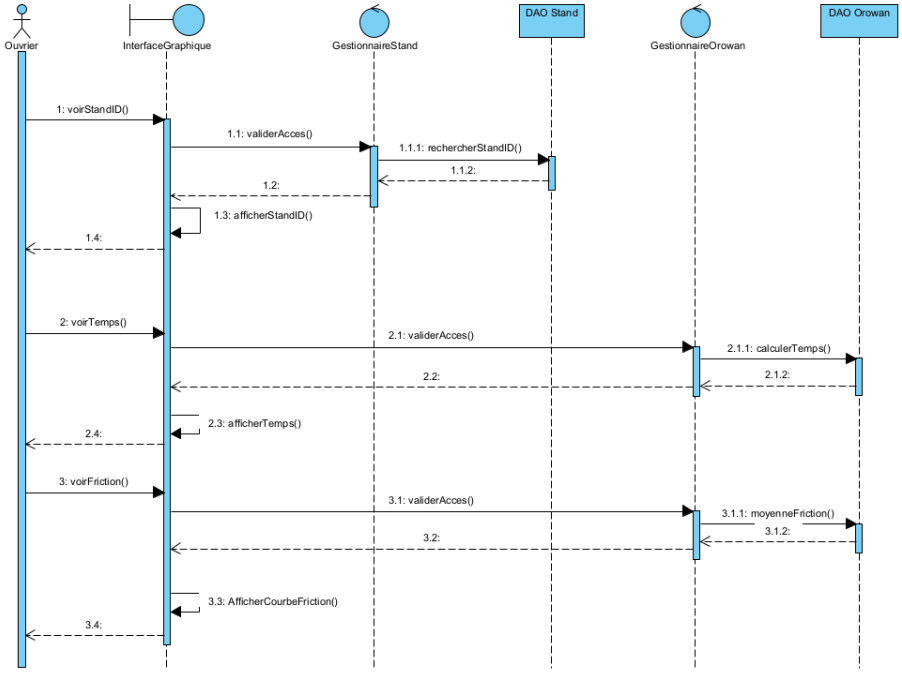
2. Les diagrammes de séquence

A. Connexion à l'application



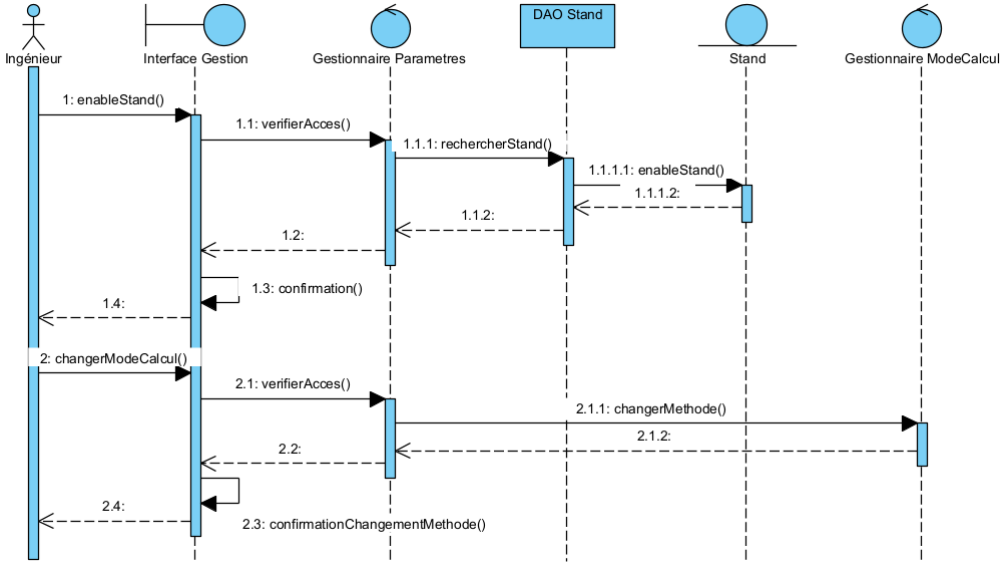
Ici l'utilisateur peut aussi bien être **un ouvrier** *(technicien)* qu'**un ingénieur de processus** *(administrateur)*. Cet utilisateur doit passer par **une phase de connexion** afin d'assurer ses droits, que ce soit la visualisation des courbes ou encore la modification des droits de certains utilisateurs.

B. Affichage des graphiques et droits d'un technicien



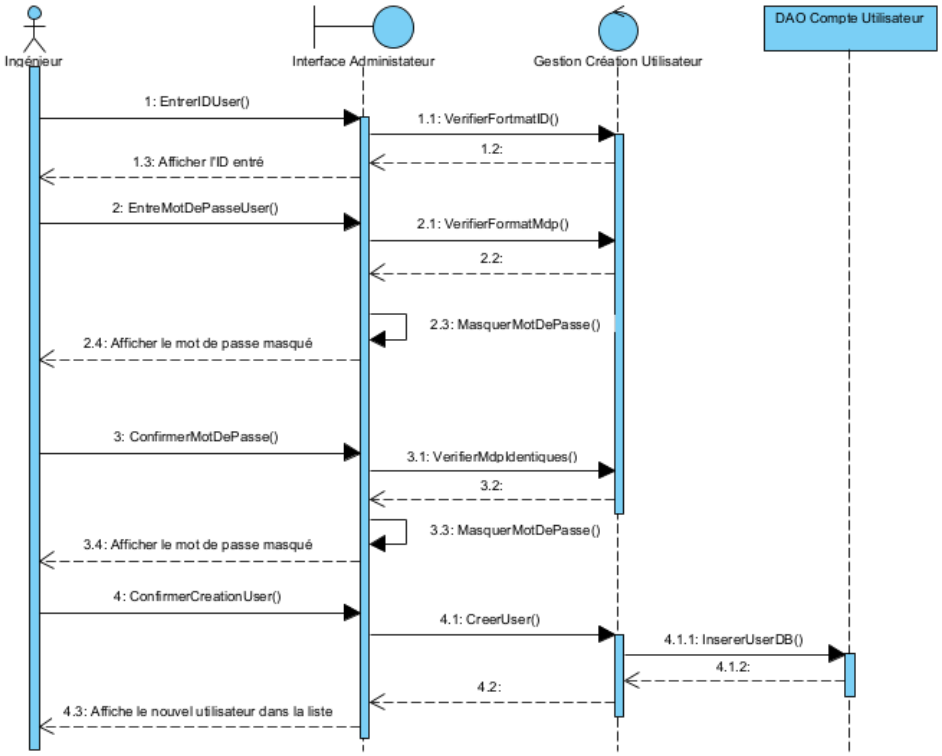
Pour simplifier la représentation et l'affichage graphique, nous traitons uniquement le cas de **la courbe de friction**. En réalité**, deux courbes supplémentaires sont demandées** à savoir : la vitesse du rouleau et la rhéologie. Le principe séquentiel est **exactement le même**, il suffit d'ajuster le nom de la courbe à afficher.

C. Modification des accès à un atelier de production et du mode de calcul



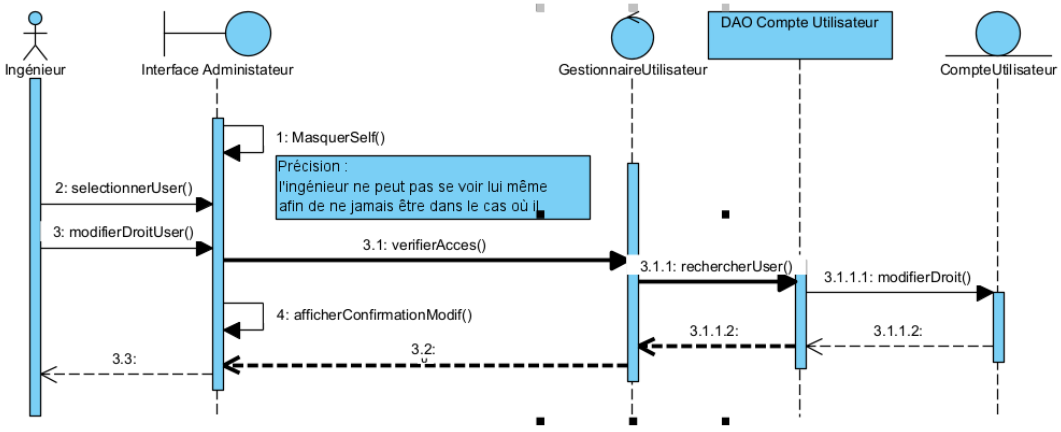
Cette fois-ci, on se place en tant qu'ingénieur pour potentiellement **supprimer ou** au contraire **autoriser l'accès à un atelier** aux autres utilisateurs. Également, l'administrateur doit être capable de **choisir le mode de calcul** *(inverse ou direct)* de l'atelier pour en déduire les grandeurs intéressantes vis-à-vis de l'**optimisation de production**. Dans notre cas, on ne raisonne uniquement que sur du calcul inverse afin de déterminer **le coefficient de friction, la vitesse du rouleau et la rhéologie**.

D. Création d'un utilisateur



On considère qu'un administrateur doit pouvoir ajouter de nouveaux utilisateurs notamment **en cas de recrutement** au sein de l'entreprise par exemple. Evidemment un utilisateur peut être considéré comme **un technicien ou** également comme **un autre administrateur** mais c'est **à déterminer** par une autre action **lors de l'administration des droits**.

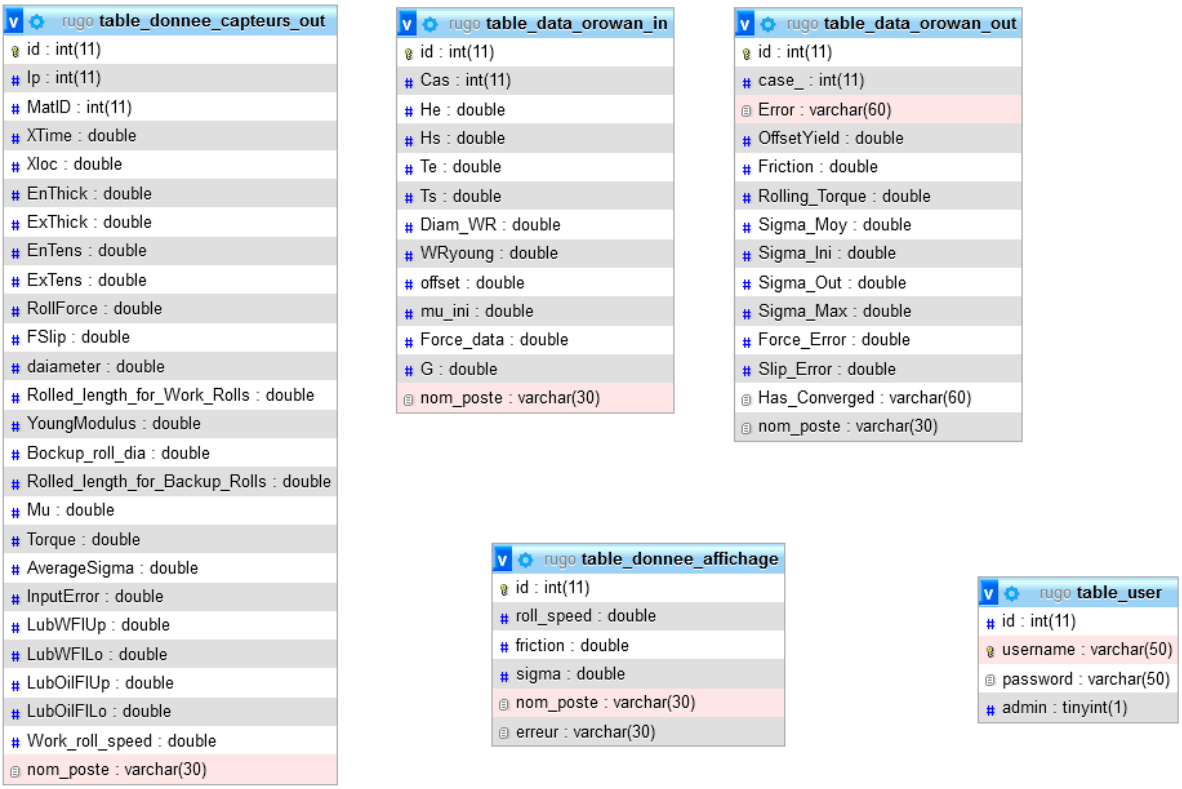
E. Administration et modification des droits



Ici on vient préciser le rôle d'un utilisateur **en ajustant ses droits**. On peut par exemple venir **autoriser l'accès** d'un utilisateur **à un atelier** de production **ou au contraire le restreindre**. Également, on peut le laisser comme un « simple » **technicien ou lui donner les droits équivalents à un administrateur**. On passe également par une **étape de masque** où Rugo vient dissimuler l'utilisateur dans la gestion du personnel pour **éviter de se supprimer** ou tout simplement de **ne plus avoir d'administrateur**.

IV. Le modèle conceptuel de données

On vient stocker nos données dans plusieurs tables à travers une **base de données relationnelle SQL**. Plus précisément, voici le schéma de notre base de données avec les différentes tables :



Les noms des tables parlent d'eux-mêmes, nous avons voulu reprendre une syntaxe similaire aux classes du package « data_types » présenté ici. Pour la table utilisateur, nous retrouvons plusieurs champs :

- **L'id** qui correspond à la clé primaire pour distinguer chaque utilisateur
- **Le nom d'utilisateur**
- **Le mot de passe** associé que l'on vient crypter (MD5)
- **Le statut de l'utilisateur** à savoir : technicien (0) ou ingénieur (1)

Globalement, pour respecter le cahier des charges, nous utilisons **des doubles** afin de traiter les données, avec plus précisément **3 décimales après la virgule**.

Ces tables représentent **l'ensemble des données utilisées par notre système** au cours du processus. On vient stocker l'ensemble des valeurs utilisées et calculées dans différentes tables afin de les **ré-utiliser ultérieurement** notamment pour l'affichage graphique ou tout simplement pour **conserver une trace du déroulement** de notre application.