Compte rendu Phase 2

# Démarrage de l’application

D’abord démarrer WAMP.

Importer le fichier script.sql afin de créer la base de données.

Double cliquer sur le fichier stock.jar ou ouvrir un IDE et lancer le main dans le fichier PrincipaleController.java

# Travail effectué

Nous sommes partis d’une base de fichiers comprenant la création d’une fenêtre Swing, d’une connexion à la base de données avec des requêtes de base ainsi qu’un autre fichier permettant de faire le lien.

A partir de là, nous avons adapté ce système au paradigme MVC (expliqué dans la partie suivante) auquel nous avons ajouté les différents modules (article, client, accueil, etc…) au fur et à mesure grâce à la structure MVC qui permettait de les ajouter de façon indépendante.

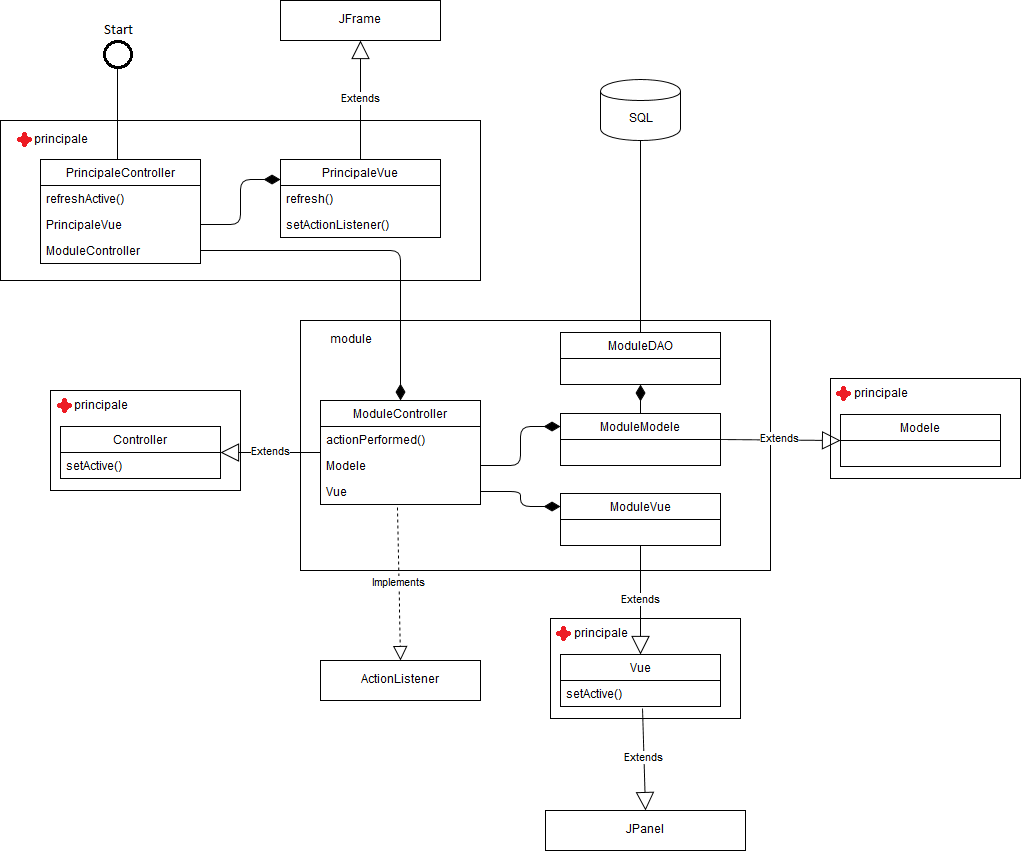
Nous avons donc ajouté les modules suivants :

* Article
* Client
* Fournisseur
* Commande
* Accueil
* Login

Un module a la possibilité d’ajouter, modifier, supprimer ou encore récupérer des éléments associés à sa table dans la base de données.

Nous avons également créé en parallèle la base de données (MCD dans une partie ultérieure) que nous avons remplie avec quelques insertions afin de pouvoir effectuer des tests.

# MVC



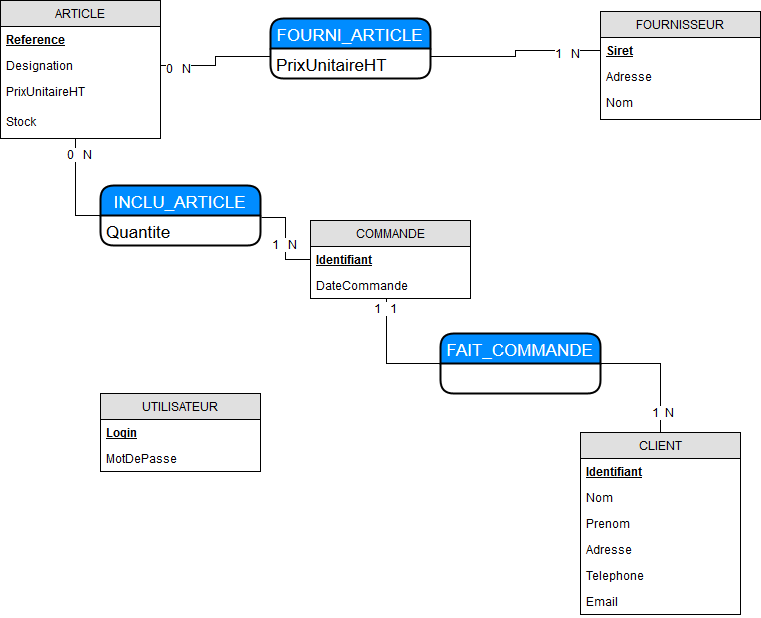
Voici l’architecture du programme avec un Module.

Lors du lancement de l’application, PrincipaleController est lancé, il va ensuite créer la Fenêtre (PrincipaleVue) ainsi que tous les modules.

Lors de l’initialisation d’un module, un modèle et une vue lui sont associés en même temps. Le model du module hérite d’un modèle principal contenant les déclarations des fonctions indispensable à la communication avec la base de données. Il en va de même pour la vue qui hérite d’une vue principale contenant les fonctions de remplissage du contenu de la fenêtre.

Si l’on souhaite afficher un module spécifique, il suffit d’appeler le contrôleur principal qui va rendre actif le module désiré en remplaçant le contenu de la fenêtre.

# MCD

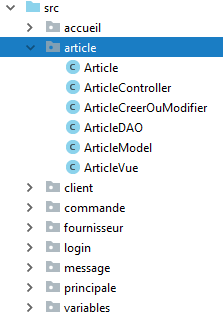


# Module

Comme dit précédemment, tous les modules sont indépendants, cependant, ils ont tous la même forme :

* L’objet à stocker
* Le Contrôleur
* Une page de création/modification
* Le DAO
* Le Model
* Une page principale listant les données existantes (Vue)

Le module de commande possède également une autre page permettant d’ajouter un article à la commande que l’on souhaite créée/modifier.



La formation des pages est également similaire :

IMAGE PAGE

Il y a la liste de tous les éléments associés au module avec des boutons pour supprimer et modifier.

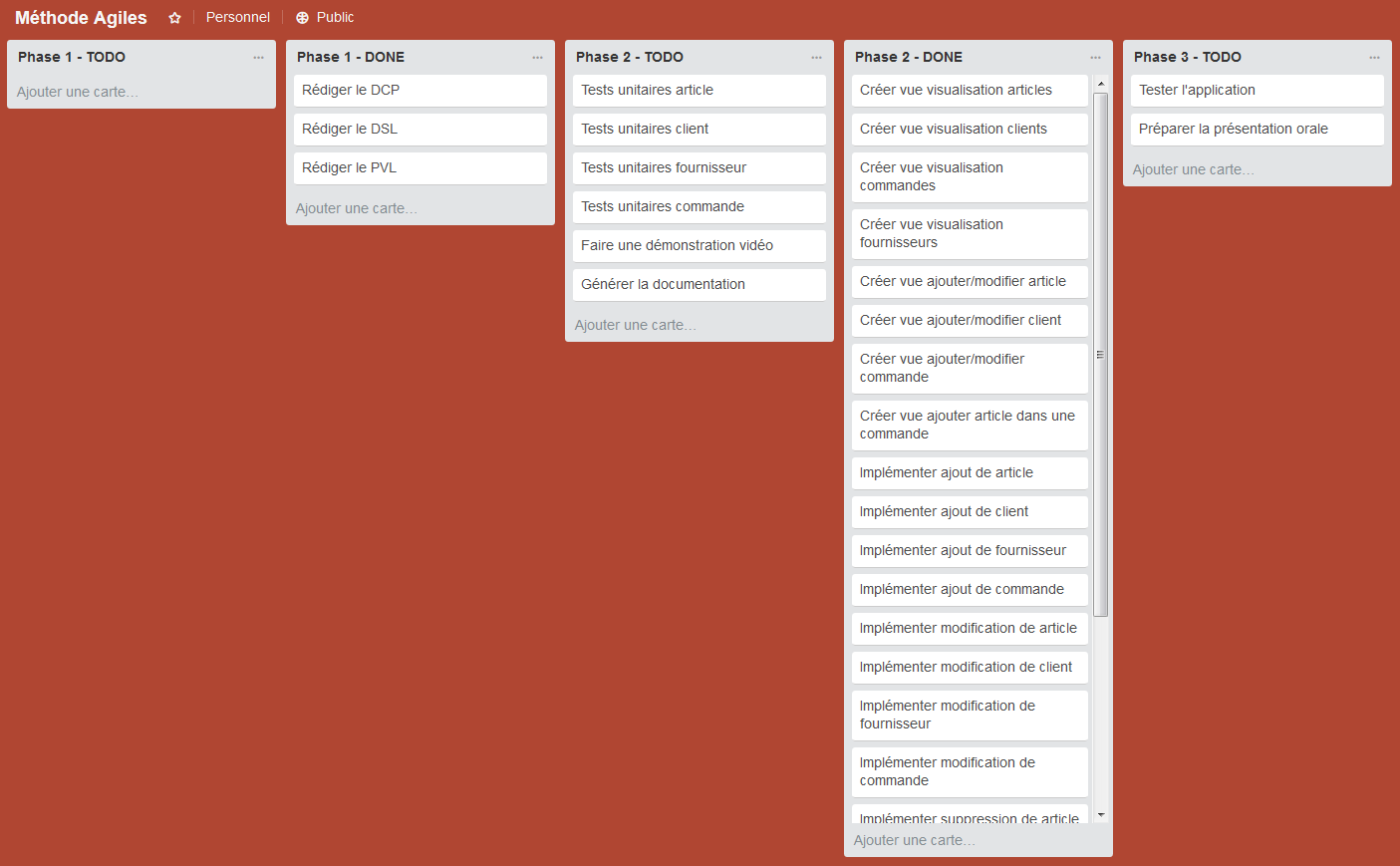
Il y a également un bouton ajouter en base de la fenêtre.  
Lors d’un clic sur modifier ou ajouter, on est renvoyé sur la page XXXCreerOuModifier qui pré-rempli les champs s’il s’agit d’une modification.

La page contient tous les éléments modifiables.

Le module commande, est différent car il faut ajouter des articles à la commande :

IMAGE MODIFIER COMMANDE + AJOUT ARTICLE

# Trello



Nous avons organisé le Trello selon les différentes Phases avec les tâches qui ont été effectuées, et les tâches restantes.

La Phase 1 étant finie, toutes les tâches ont été effectuées, la Phase 2 est encore en cours au moment où l’on écrit ce document, il reste des tâches à faire.

# JavaDoc

Nous avons décidé de commenter les fonctions au fur et à mesure de leur création afin de pouvoir plus facilement créer la documentation plutôt que de devoir écrire toute la documentation d’un coup à la fin.

Nous l’avons générée via :

# Redmine

Etant donné le peu d’heures de travail effectué en cours par rapport à celui effectué en parallèle en dehors des cours, nous n’avons pas pu mettre Redmine à jour au fur et à mesure de notre avancement depuis chez nous. Nous avons donc opté pour la création d’un dépôt Github que nous avons versé sur Redmine en fin de séances.

