Gestion des enchères

Architecture Logicielle

Tuteur : Olivier Caron, Bernard Carré et Walter Rudametkin

Céline BRUNO – Florian DEZE

Génie Informatique et Statistique

4ème année – Semestre 8

Polytech'Lille

Mai 2017

Table des matières

1 Introduction

Dans le cadre de notre 4ème année de formation en Génie Informatique et Statistique, nous réalisons un projet d'architecture logicielle, nommé « Gestion des enchères » afin d'appliquer les notions vues lors du cours d'Architecture Logicielle. Le but du projet consiste à développer une application JEE de gestion des enchères. Pour cela, nous avons programmé les composants EJB entités nécessaires à l'application suivant le cahier des charges donné. Puis, nous avons programmé le code métier afin d'enrichir l'application par le développement d'un composant session. Enfin, nous avons développé une application cliente. Nous pouvions soit développer une application cliente en mode console soit en mode web et nous avons choisi de développer l'application cliente en mode web car l'application web sera plus ergonomique.

Dans un premier temps, vous verrez nos schémas de modélisation UML, puis dans une deuxième partie, vous verrez le détail des données tests et comment les utiliser. Dans une dernière partie, vous verrez une description de l'application cliente développée.

2 Modélisation UML

1 Schéma des composants EJB entités

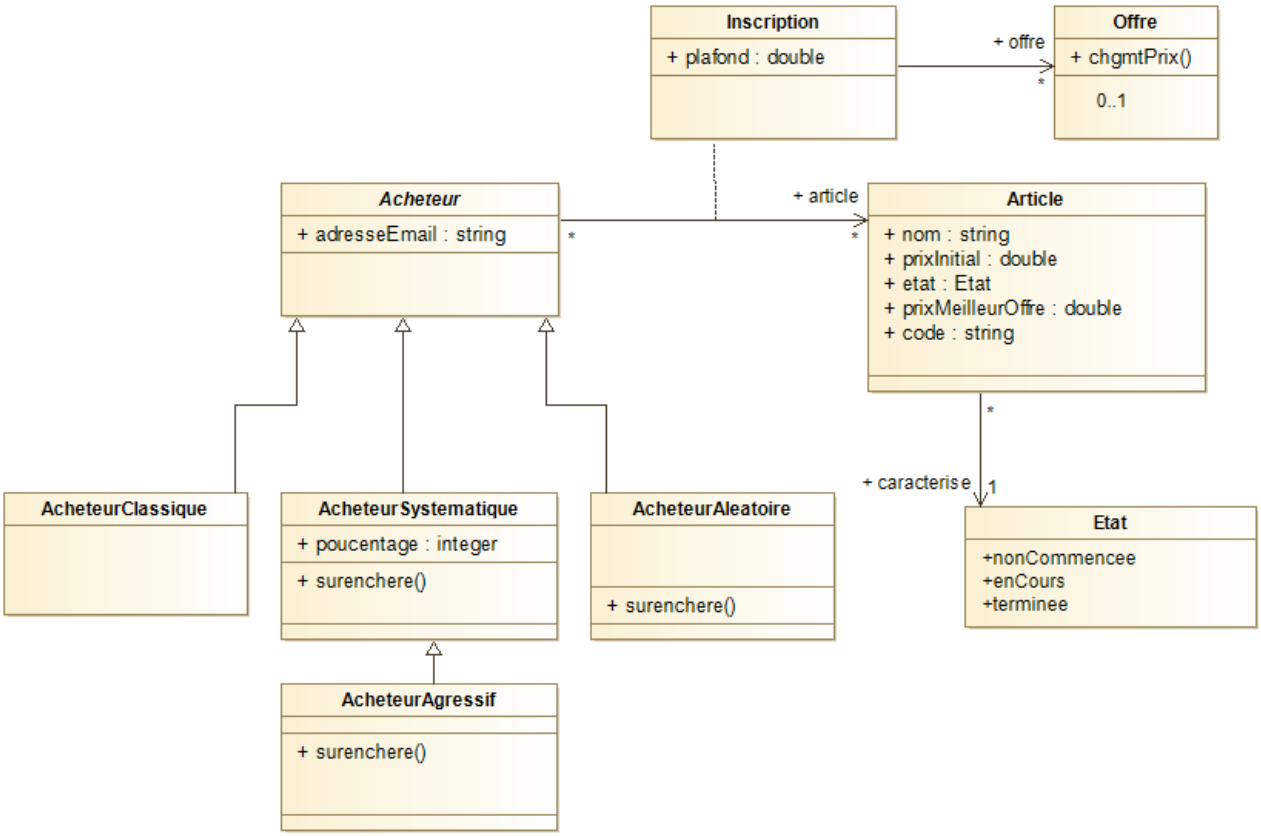

FIGURE 1 – schéma UML de la gestion des enchères

Nous avons choisi de créer une classe pour chaque type d'acheteur. En effet, un acheteur peut être Classique ou Systématique ou Aléatoire ou Agressif. En fonction du type de l'acheteur, la surenchère sera différente. La classe abstraite Acheteur contient l'identifiant de l'acheteur : son adresse email.

Afin de faire une offre sur un article, l'acheteur (peu importe son type) doit être inscrit. Lors de l'inscription, l'acheteur précise le prix maximum qu'il s'autorise à engager pour l'article en question.

Nous avons relié les classes Acheteur et Article car l'acheteur doit connaître les caractéristiques de l'article sur lequel il veut enchérir. L'article doit connaître la liste des acheteurs potentiels inscrit.

La classe Etat est une énumération des états possibles que peut prendre un article. L'enchère d'un article est soit non commencée, soit en cours, soit terminée. La classe Etat n'a pas besoin de connaître les informations de l'article.

Nous devons à présent modifier ce schéma UML pour qu'il soit compatible en JPA car les schémas UML sont indépendants de toute caractéristique technologique.

2 Schéma des composants EJB entités compatible JPA

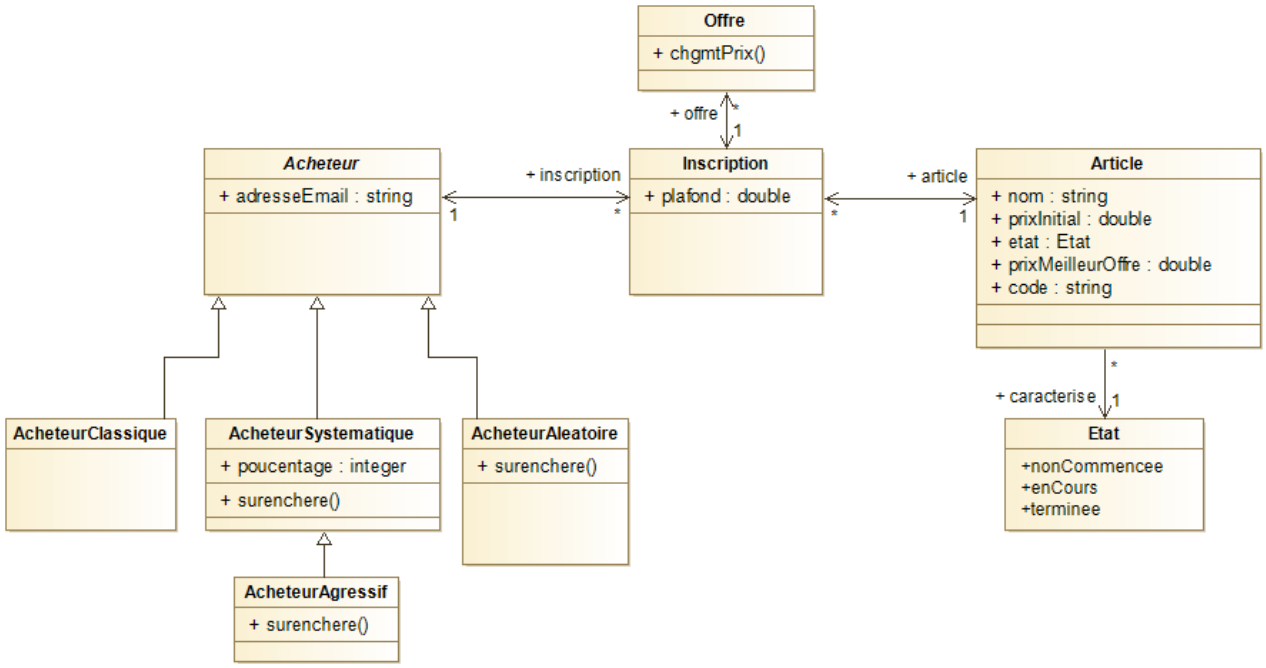

FIGURE 2 – schéma UML de la gestion des enchères, compatible JPA

Afin que notre schéma UML soit compatible avec la technologie JPA, nous devons prêter attention aux adaptations requises :

- Les entités JPA ne supporte que les associations binaires. Les associations n-aires (avec $n > 2$) peuvent être transformées en plusieurs associations binaires. Nous n'avons pas d'associations n-aires donc nous n'avons pas de transformation à effectuer pour ce premier point.
- Les entités JPA nécessitent d'avoir un attribut qui joue le rôle de clé primaire. La clé primaire de la classe Acheteur est l'adresse Email. La clé primaire de la classe inscription est un numéro d'inscription. La clé primaire de la classe Offre est un code. Enfin, la clé primaire de la classe Article est le code de l'article.
- Les associations binaires JPA ne peuvent pas avoir de propriétés d'associations. Or, dans notre cas, nous avons une propriété d'association entre les classes Article et Acheteur nommé Inscription. Nous devons donc transformé l'association Inscription en une classe Inscription. La classe est donc reliée à la classe Acheteur et à la classe Article.

Notre schéma est donc compatible avec la technologie JPA. Lors de la programmation des composants EJB entités, nous avons annoté les classe par l'annotation @Entity. La classe dispose d'une fonction constructeur sans paramètres. La classe dispose d'une propriété annotée par @Id (l'annotation se fait uniquement sur la fonction getter de la propriété.

3 Schéma d'architecture logicielle

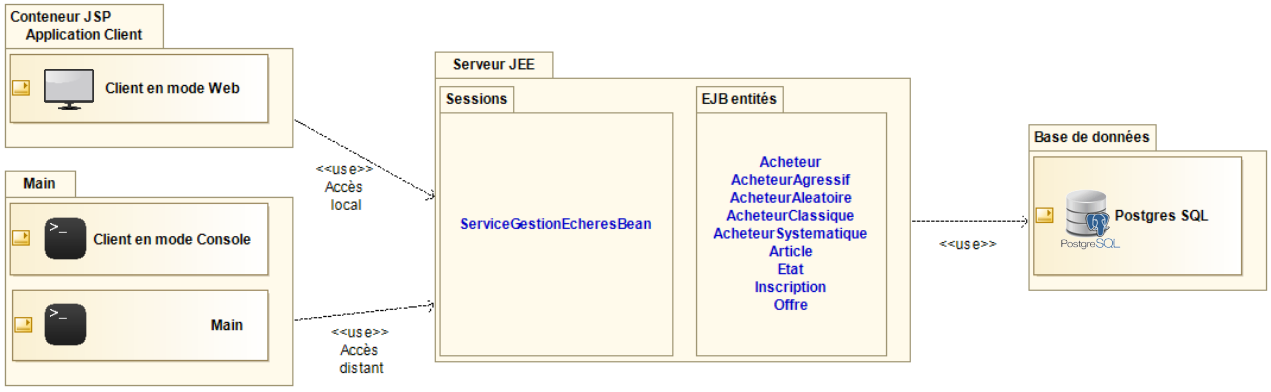

FIGURE 3 – schéma d'architecture logicielle

L'architecture logicielle est une architecture 3-tiers. L'architecture logicielle du système est divisée en trois niveaux (ou couches). La première couche est la couche présentation, la deuxième couche est la couche métier et la troisième couche est la couche accès aux données.

La couche présentation est le conteneur JSP. Elle correspond à la partie de l'application visible et interactive avec les utilisateurs. Dans le conteneur JSP, nous avons l'application cliente en mode web, l'application web en mode console et le main. L'application en mode web est en accès local au serveur JEE alors que l'application cliente en mode console et le main sont en accès distant.

La couche métier est le serveur JEE. Elle correspond à la partie fonctionnelle de l'application, celle qui décrit les opérations que l'application opère sur les données en fonction des requêtes des utilisateurs, effectuées au travers de la couche présentation. Le serveur JEE contient un composant session : ServiceGestionEncheresBean et neuf composants Entités.

La couche accès aux données est le serveur de base de données Postgres SQL. La couche accès aux données consiste en la partie gérant les données du système. Ces données peuvent être propres au système, ou gérées par un autre système. La couche métier n'a pas à s'adapter à ces deux cas, ils sont transparents pour elle, et elle accède aux données de manière uniforme.

Un modèle d'architecture 3-tiers a pour objectif de répondre aux préoccupations suivantes :

- L'allègement du poste de travail client,
- La prise en compte de l'hétérogénéité des plates-formes,
- L'introduction des clients dits « légers »,
- L'amélioration de la sécurité des données, en supprimant le lient entre le client et les données,
- La rupture du lien de propriété exclusive entre l'application et les données,
- La meilleure répartition de la charge entre les différents serveurs d'application.

3 Données tests

Afin de vérifier que tous les cas d'erreurs possibles soient pris en compte, les données tests ont été écrites de manière à avoir un nombre d'enchère important sur un article. Pour cela, la création d'articles à un prix bas et la mise en place d'un plafond élevé pour chaque acheteur ont été nécessaire. A la vue d'un nombre d'enchère important, les corrections apportées au code nous ont permis la vérification des cas suivants :

- L'arrêt des enchères lorsque le plafond de chaque acheteur est atteint ;
- Un acheteur ne peut pas enchérir sur lui-même ;
- La surenchère se calcule à partir de la meilleure offre proposée ;
- Seules les nouvelles offres, qui sont meilleures que les précédentes, sont retenues ;
- Seuls les acheteurs inscrits pour un article donné peuvent faire une offre pour celui-ci.

Notre application client.Main est notre application d'administration, elle permet la création d'articles, d'acheteurs, leurs inscriptions à ces articles en dur et le lancement de deux sessions d'enchères correspondant aux deux articles crées.

Pour notre client en mode web, nous avons mis en place un fichier initBase.sql qui initialise la base, afin d'utiliser notre application sur un navigateur.

Tous les types d'acheteurs ont été implémentés avec un acheteur classique, deux acheteurs agressifs et un acheteur aléatoire. L'acheteur classique ne fait pas de surenchères donc ces offres n'apparaissent pas dans la console, ni sur la page web. Seuls les acheteurs agressifs et aléatoires font des surenchères sans dépasser leurs plafonds respectifs.

Pour lancer l'application client d'administration, il faut :

- Lancer la commande deployAll
- Lancer runClient1

Pour lancer l'application client sur le navigateur il faut :

- Lancer la commande deployAll
- Lancer la commande initBase.sql
- Ouvrir un navigateur à l'adresse suivante :
  localhost:8080/repRacine/GestionEncheres?code=re12

4 Application cliente

Parmi les deux applications clientes proposés, nous avons choisi de développer l'application cliente en mode web. Nous avons un seul composant web-JSP appelé GestionEncheres.jsp. Il accepte 3 paramètres:

- code : la référence de l'article
- adresseEmail: l'adresse mail de l'acheteur
- prix : prix de l'offre de l'acheteur

Si l'email de l'acheteur existe, que celui est inscrit à l'enchère et que le prix est correcte (prix > prix actuel de l'article et prix < plafond de l'acheteur) alors une offre est formulée sur l'article référencé par code.

Deux cas d'affichages d'erreurs sont pris en compte:

- Si l'article n'existe pas : l'article « code » est inconnu et une liste des articles auquel l'acheteur s'est inscrit ;
- Si l'acheteur n'existe pas : l'adresse mail ne correspond n'a aucun acheteur.

S'il n'y a pas d'erreur, la page affiche un descriptif de l'article, ainsi que l'ensemble des offres proposés suite au lancement avec une offre d'un montant de « prix ». Un formulaire est proposé afin de réaliser une nouvelle offre.

5 Conclusion

Notre projet est constitué d'une application d'administration et d'une application cliente en mode web. Nous aurions pu aussi développer l'application en mode console mais cela n'a pas été réalisé par manque de temps. L'architecture de nos composants sessions peut être améliorée, celle-ci nous oblige à faire beaucoup de recherche par des boucles « for » pour obtenir un élément. Pour obtenir la meilleure offre à partir d'un article, nous devons chercher toutes les inscriptions à cet article, puis toutes les offres de chacune de ces inscriptions et enfin trier ces offres et prendre la meilleure. Cependant ces choix d'implémentations furent tout de même le résultat de réflexions et de corrections pour constituer une application qui nous paraissait la plus censée.

Notre application répond à l'ensemble des consignes du projet demandé.