[T03] Méthodologies Agiles / J2EE et webServices

Projet ezBay : Rapport final

Fichier téléchargeable à l'adresse :
http://axel.ammuller.free.fr/ezbay/axlomoso_ezbay_RAPPORT-FINAL.pdf

1. L'équipe : AXLOMOSO – axlomoso@gmail.com

1.1. Les membres

Axel AMMÜLLER (axel.ammuller@free.fr), Lotfi ELGHERBI (lotfi_elgh@yahoo.fr),
Sophie LION (sophie.lion@gmail.com), Mohamed EL-MRABH (elmrabah@yahoo.fr).

1.2. Fonctionnement de l'équipe

Avant Projet : indépendance des membres
La première phase du projet, avant même de réaliser l'itération 00, a été une phase d'acquisition de connaissances. Dans cette phase, chacun des membres a tâché d'effectuer un certain nombre de recherches personnelles, indépendamment des autres membres. Une fois cette phase terminée, nous avons tâché de mettre en commun les savoirs, et de collaborer pour l'exécution de l'itération 00 .

Itérations 00 et 01 : travail en trinôme
Pour les itérations 00 et 01, les développements ont été réalisés conjointement par les 3 membres participants, sur une seule machine. En effet, en raison de contraintes techniques, et en raison de l'absence d'un des membres, nous n'avons pas pu mettre en œuvre le travail en binôme.

Itération 02 : travail individuel
Une fois les problèmes de configuration de machines résolus, l'équipe disposait de 3 plates-formes de développement. Un des membres était toujours absent à l'appel. Ainsi, nous avons fait le choix de partager les tâches afin de les attribuer à chacun des membres pour permettre leur exécution en parallèle.
Au niveau de la collaboration : la plupart du temps, l'équipe était réunie dans la même pièce, un ordinateur par membre, ce qui a permis un échange permanent. Vous trouverez le récapitulatif de l'attribution des tâches en annexe.

2. Livrable

Le livrable est sous la forme d'un fichier .zip : **axlomoso_ezbay.zip**
Il est téléchargeable à l'adresse : http://axel.ammuller.free.fr/ezbay/axlomoso_ezbay.zip
Contenu :

- **readme_first.txt :** notice d'installation
- **datasource :** contient le fichier de configuration de la "datasource" de ezBay
- **ezBayEAR.ear :** application à déployer
- **requis :** contient les fichiers du type librairies nécessaires au fonctionnement du projet
- **sql :** contient le script sql à exécuter pour insérer des enregistrements dans la base de données
- **src :** contient les sources du projet

3. Itérations archivées (repository CVS)

Adresse du repository : http://cvs.berlios.de/cgi-bin/viewcvs.cgi/ezbay/
Vous y trouverez l'ensemble de nos itérations (sources, documents…).

Répertoires importants :

- ezBay : application métier
- ezBayWeb : application cliente
- ezBayDocs : documents du projet (userStories, suivi…)

Configuration pour une utilisation dans un client CVS :
Host : cvs.berlios.de
Repository path : /cvsroot/ezbay
User : axlomoso
Password : iloveejb
Connection type : pserver ou extssh

4. Bilan du projet

4.1. Ce que nous avons appris

Avant EzBay, aucun d'entre nous ne connaissait l'environnement J2EE, ni le framework Struts. Leur utilisation combinée nous a permis d'apprendre de nouvelles techniques de programmation. Au niveau de l'environnement de développement, nous ne connaissions pas très bien Eclipse, ce projet nous a permis de mieux maîtriser cet outil très pratique. Nous n'envisageons plus vraiment l'avenir sans lui maintenant…

4.2. Ce que nous avons aimé

- Le travail d'équipe
- L'apprentissage de nouveaux outils et méthodes
- Se coucher tard…

4.3. Frustrations

Le cadre universitaire du projet ezBay ne nous a pas laissé le temps de le mener totalement à terme. Notre côté perfectionniste en reste insatisfait. De plus, le projet n'a pas été (et ne sera pas) mis en production. Ainsi, nous n'avons pas pu apprécier les facultés de montée en charge de la plate-forme J2EE / EJB.

5. Temps passé global

Prévu : 35 heures/membre, soit 140 heures.
Réalisé : 225 heures pour l'ensemble.
Pourquoi ce décalage ? Nous avons choisi, pour l'implémentation du client web, de mettre en œuvre un élément que nous ne connaissions pas auparavant : Struts. Nous avons du comprendre et apprendre à utiliser le framework avant de commencer réellement les développements du client web. C'est d'ailleurs le développement de ce client web qui a concentré la plus grande partie de nos efforts. Dans un contexte réel de production, il ne serait pas judicieux de choisir de développer une application avec une technologie inconnue. Cependant, nous avons souhaité utiliser le « prétexte » de ce projet universitaire pour acquérir un maximum de compétences, que nous pourront maintenant mettre en pratique lors de notre insertion professionnelle.

Nous tenons également à préciser que les 225 heures consacrées au projet ont été effectuées par 3 étudiants seulement, soit 75 heures /personne. En effet, un des membres a été totalement absent du projet, dans toutes ses phases.

6. Lois de Murphy

Deux lois qui caractérisent quelques unes des difficultés rencontrées.

En hommage à notre cher compagnon CVS, qui nous a réservé certaines « bonnes » surprises :

C'est au moment où l'on a le plus besoin du serveur CVS, que celui–ci est indisponible.

En référence à notre découverte du framework Struts :

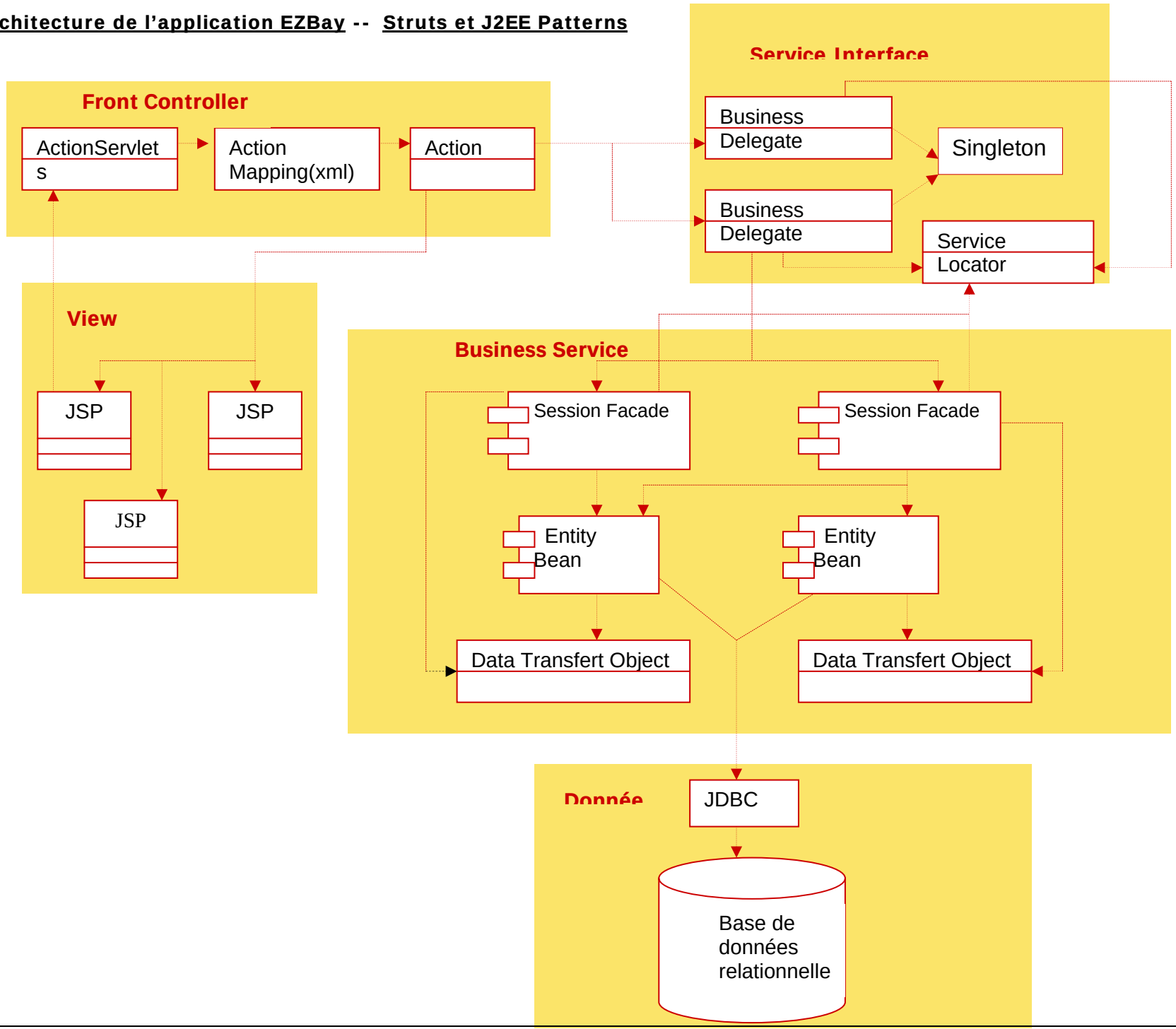
C'est lorsque l'on croit avoir tout compris, qu'on n a finalement rien compris.

7. Annexes

7.1. L'application

7.1.1. Schéma d'architecture

Architecture de l'application EZBay -- Struts et J2EE Patterns



7.1.2. *Diagrammes UML*

Cas d'utilisation

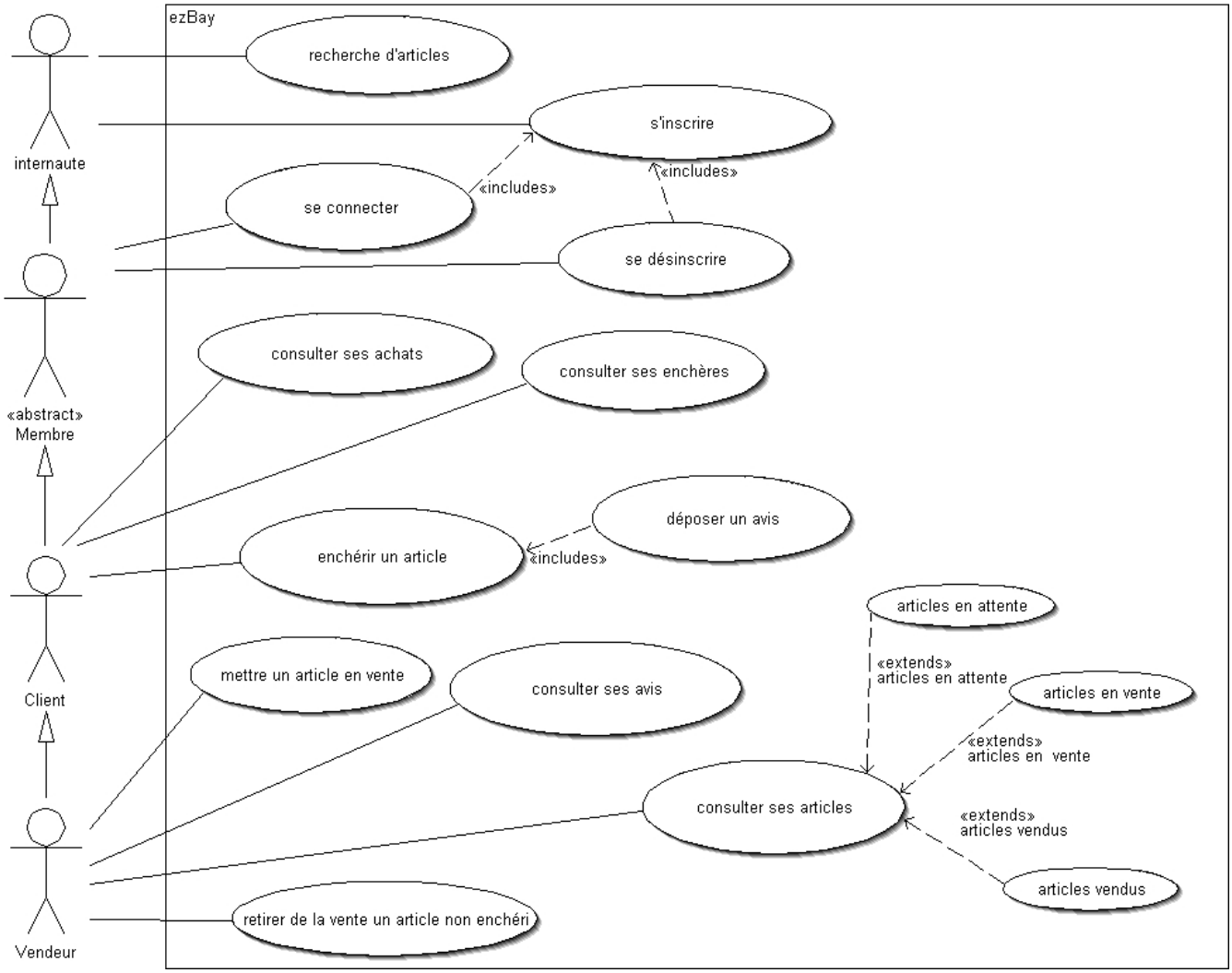


Diagramme de classes technique

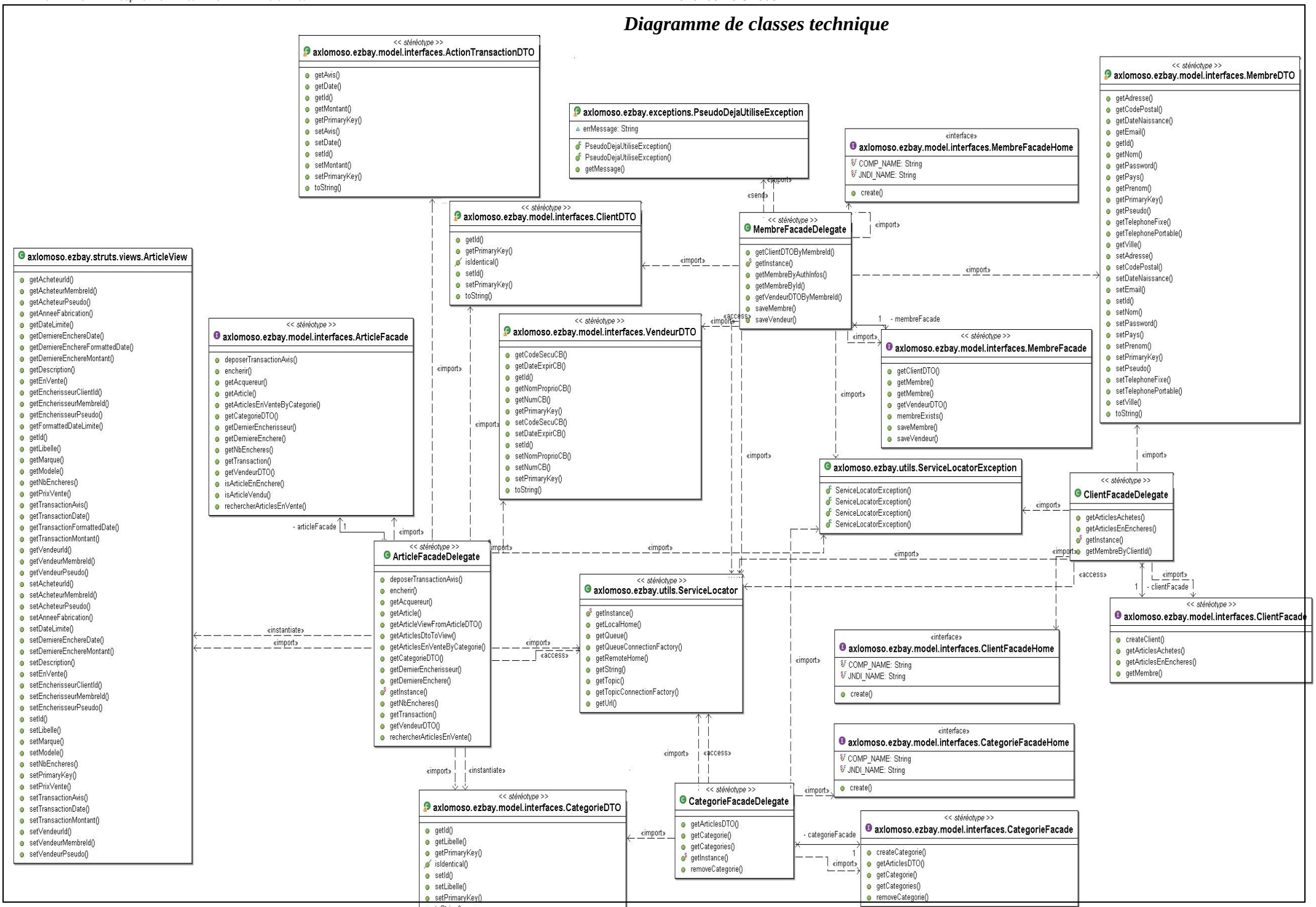


Diagramme de séquences technique : enchérir un article

Axel AMMÜLLER - Sophie LION - Lotfi ELGHERBI - Mohammed EL-MRABH

Lundi 06 mars 2006

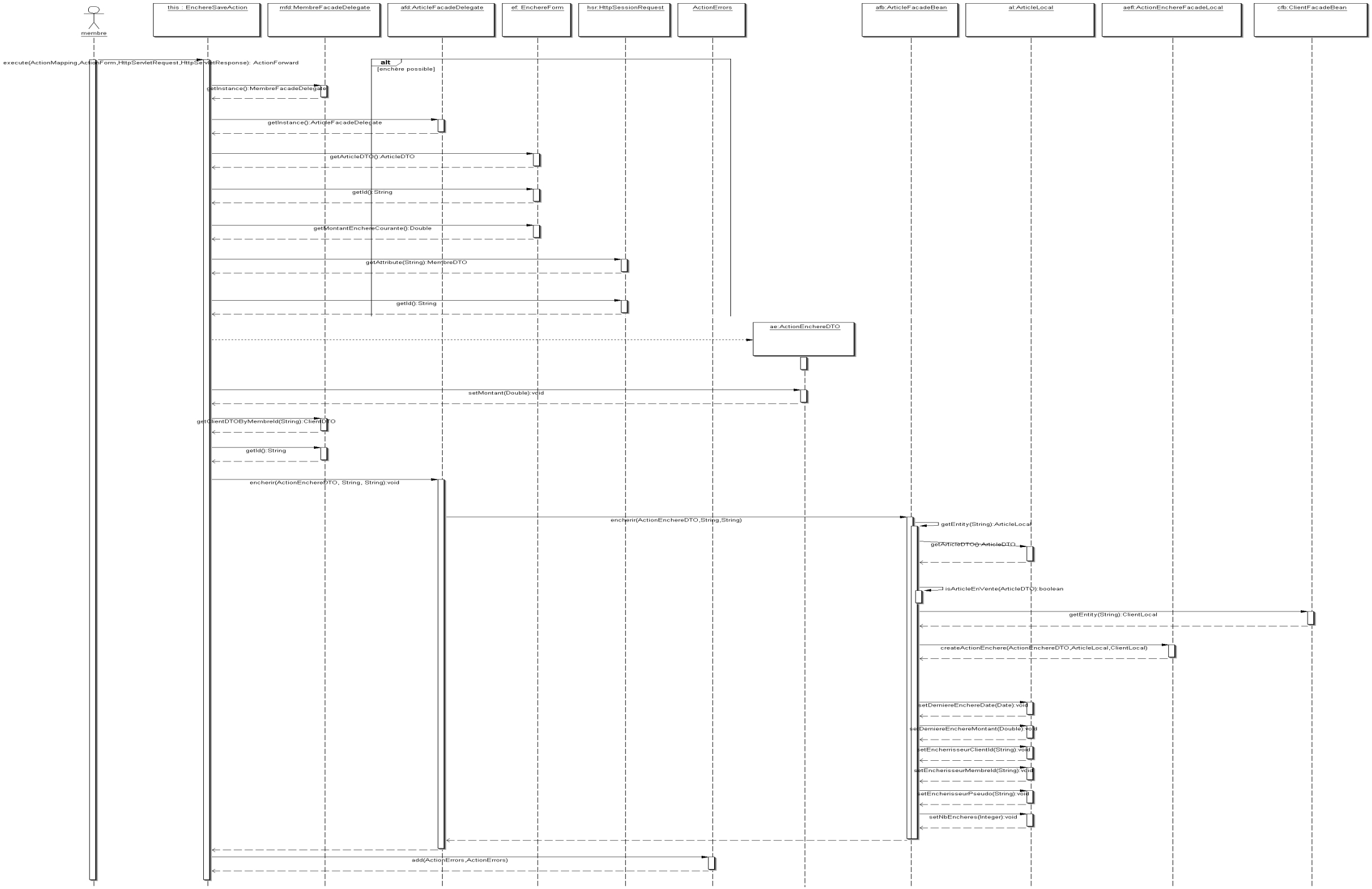


Diagramme d'état : classe Article

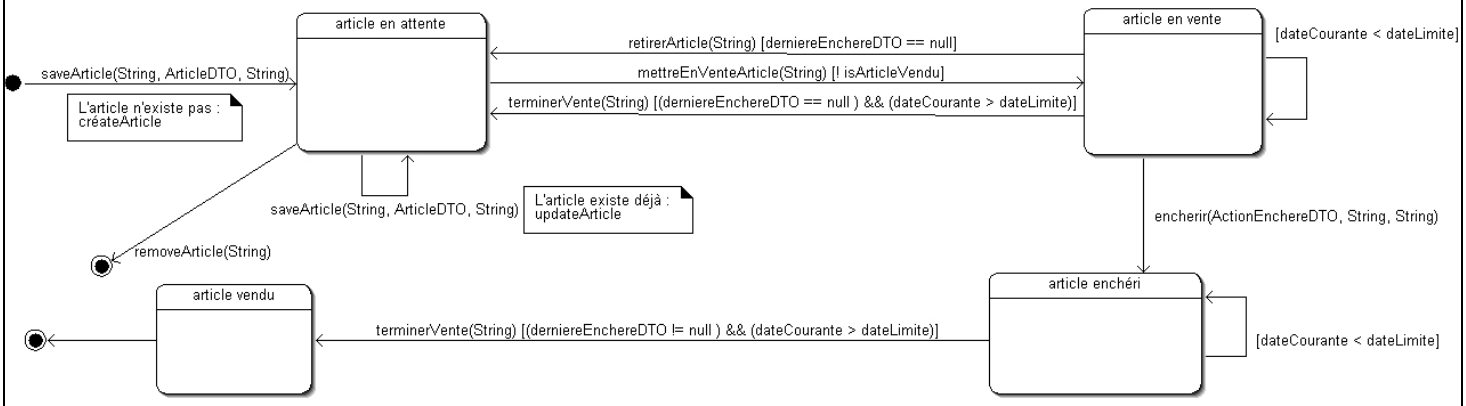
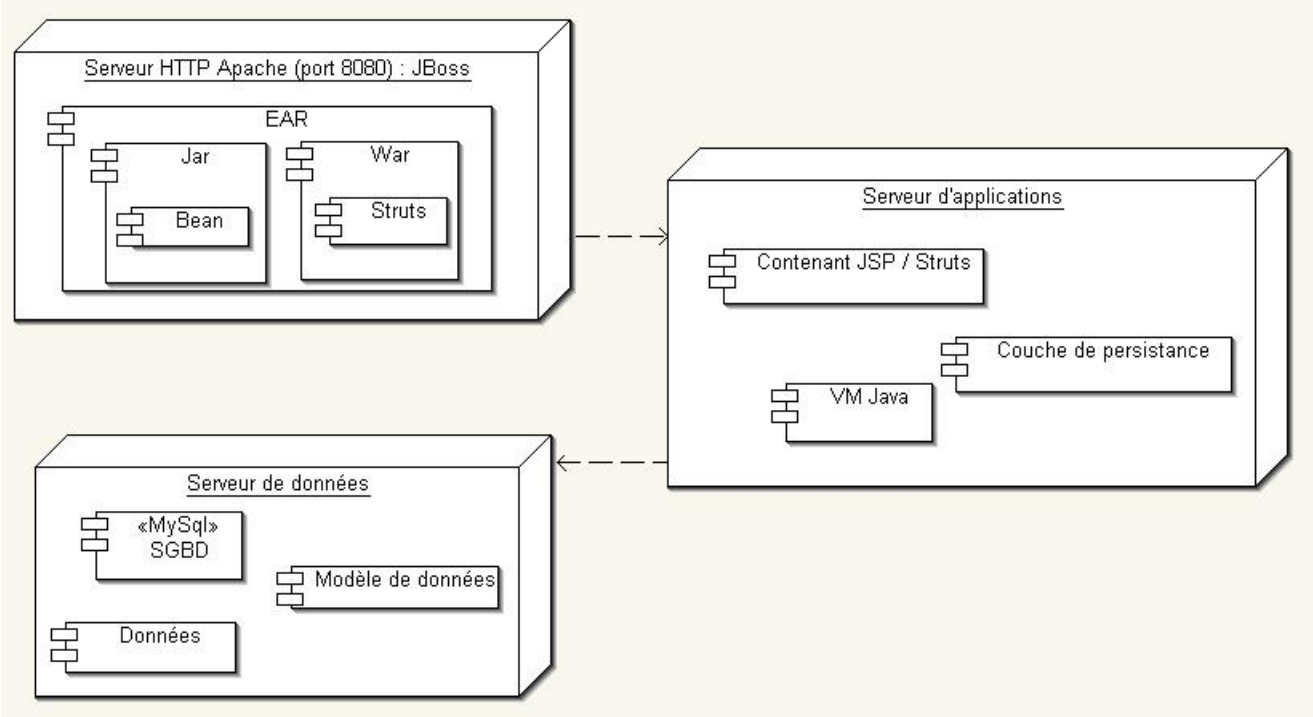


Diagramme de déploiement J2EE



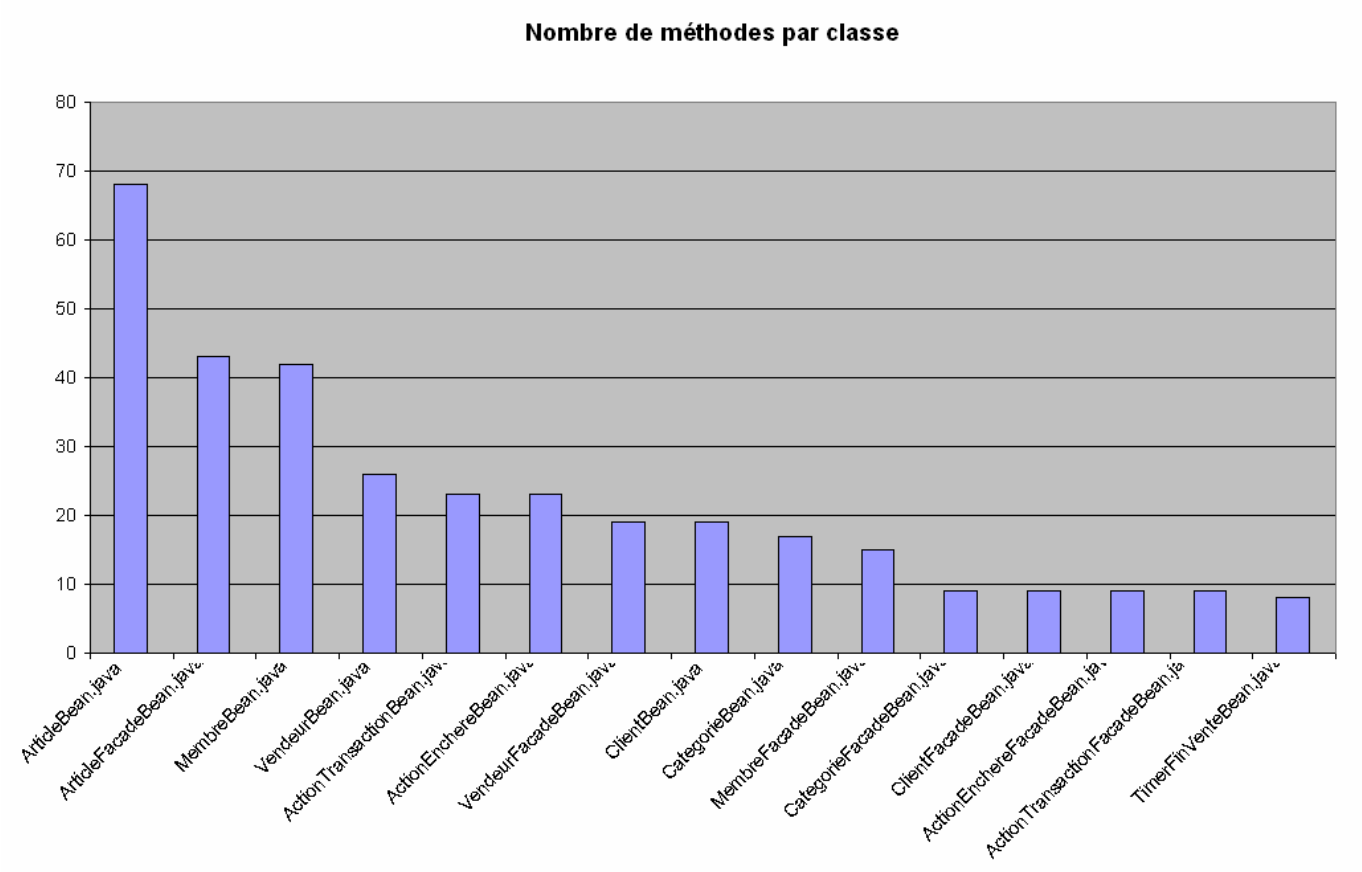
7.1.3. *Métriques*

Dans cette partie, nous étudierons les métriques des packages :

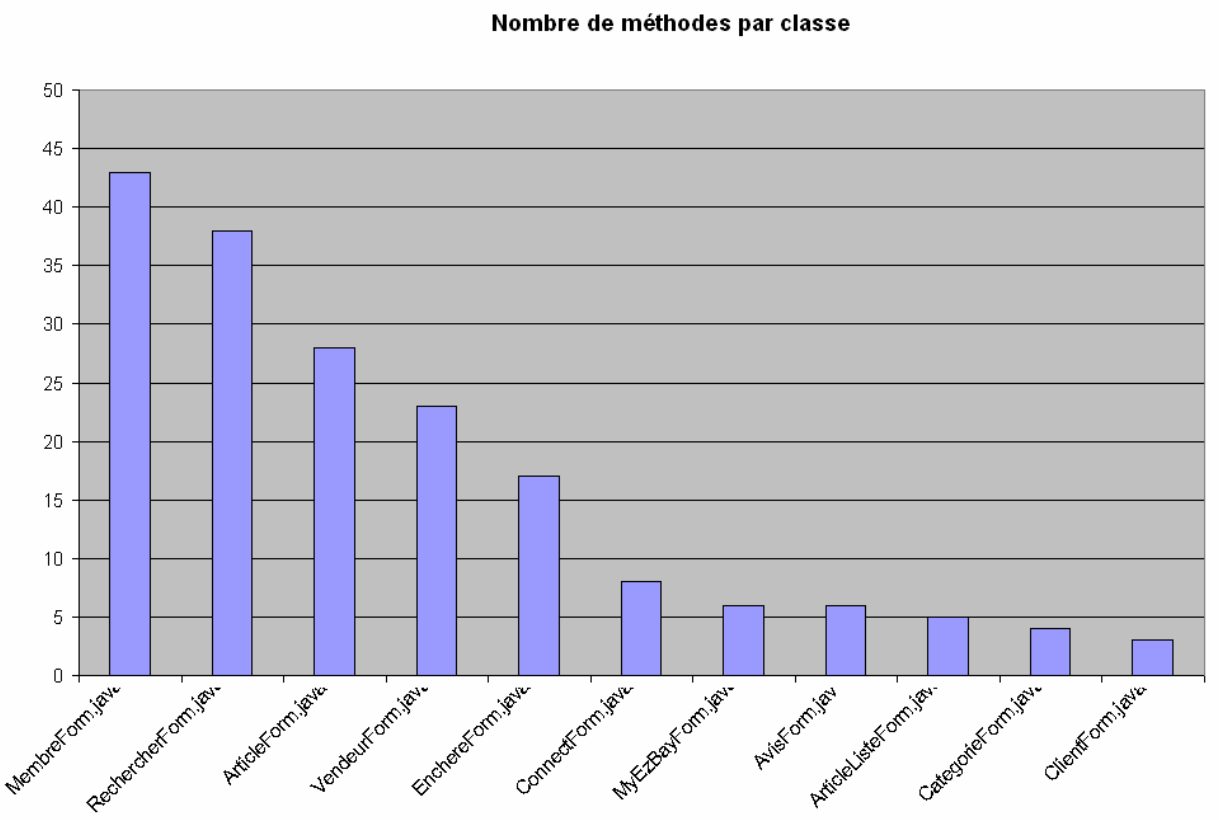
- **axlomoso.ezbay.model.ejb** : contient les différents beans
- **axlomoso.ezbay.struts.action** : contient les « actions » struts.
- **axlomoso.ezbay.struts.form** : contient les « forms » struts.

Nombre de méthodes par classe

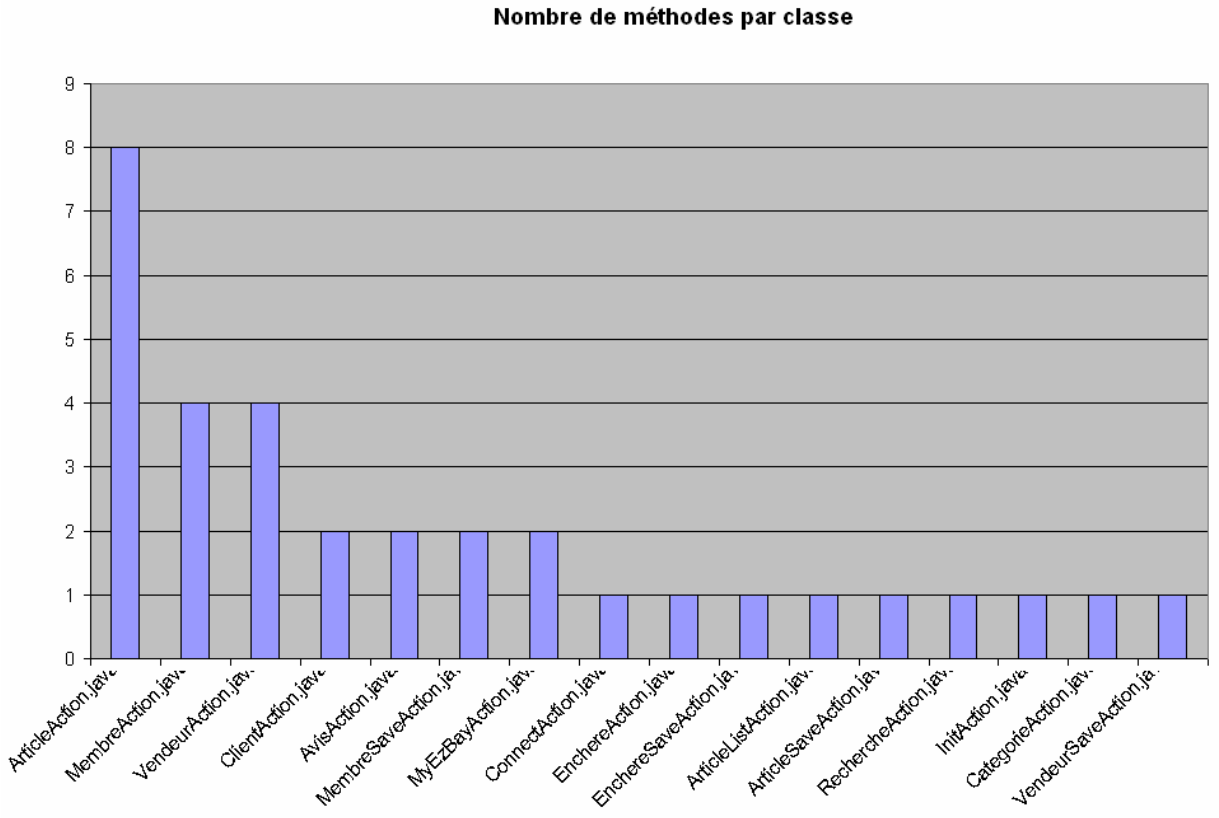
Le package axlomoso.ezbay.model.ejb comporte 558 méthodes, avec un maximum de 68 méthodes dans la classe ArticleBean.java. Ceci s'explique par le fait que l'article est au centre de notre application.



Le package `axlomoso.ezbay.struts.form` comporte 181 méthodes, avec un maximum de 43 méthodes pour la classe `membreForm.java`.



Le package `axlomoso.ezbay.struts.action` comporte 33 méthodes, avec un maximum de 8 méthodes pour la classe `articleAction.java`.



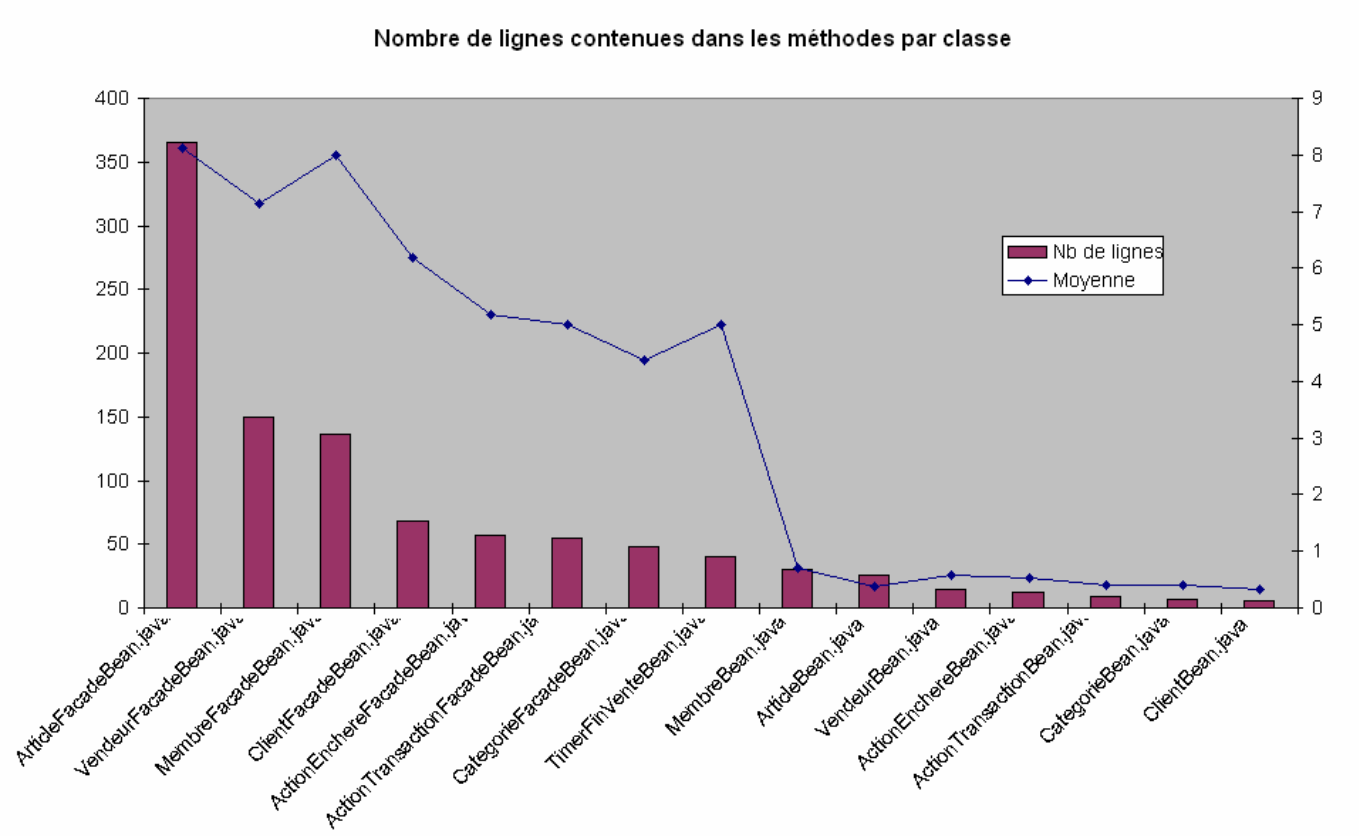
Nombre de lignes de code par méthode

Le package `axlomoso.ezbay.model.ejb` contient 558 méthodes représentant 1207 lignes de code c'est-à-dire en moyenne 2 lignes de code par méthode.

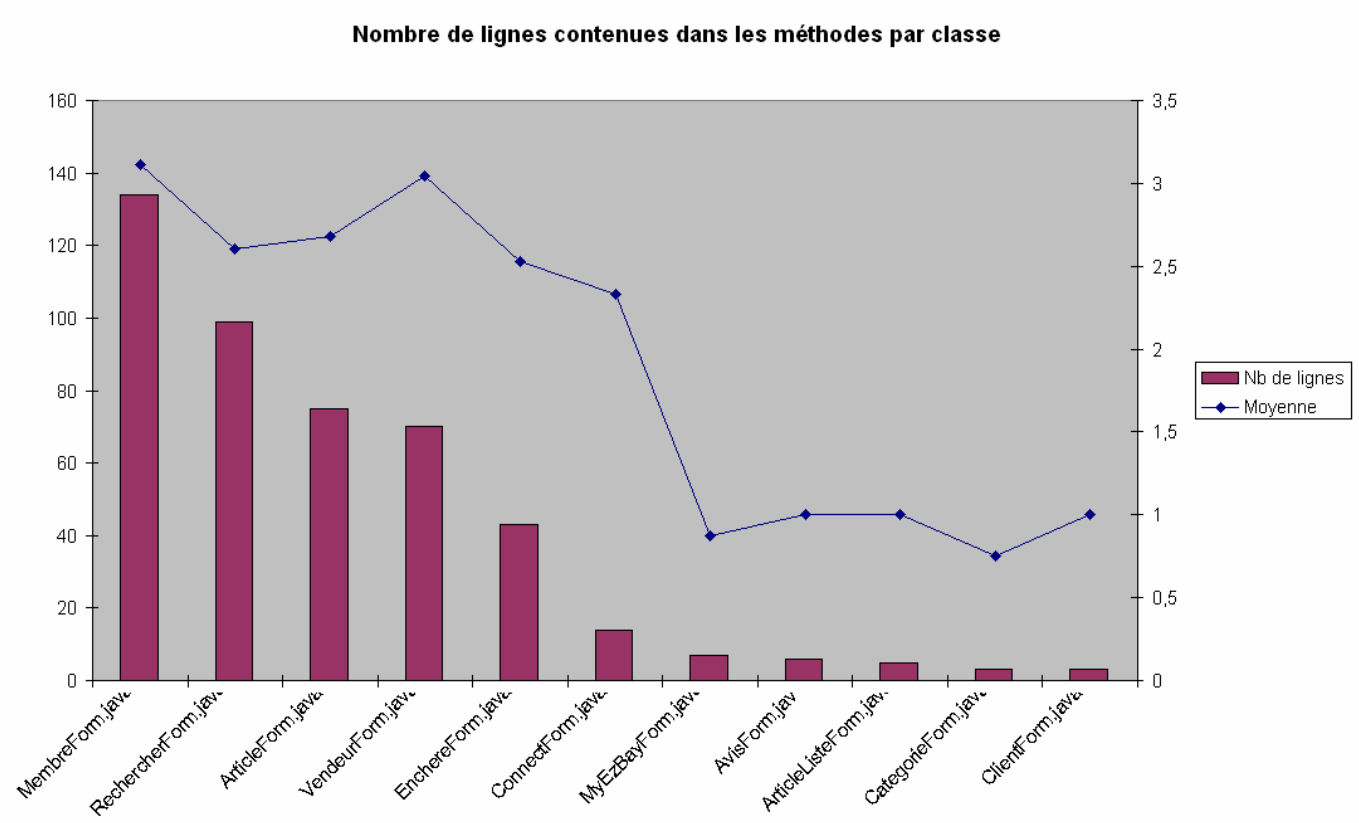
Nous remarquons rapidement que la classe `ArticleFacadeBean.java` possède le plus de lignes de code (360 lignes), avec une moyenne de 8 lignes par méthode.

Ce constat n'est pas surprenant, puisque l'ensemble de l'application tourne autour des articles. Dans la classe `articleFacadeBean.java`, la méthode `terminerVente` est l'une des plus importante (30 lignes), puisqu'elle permet :

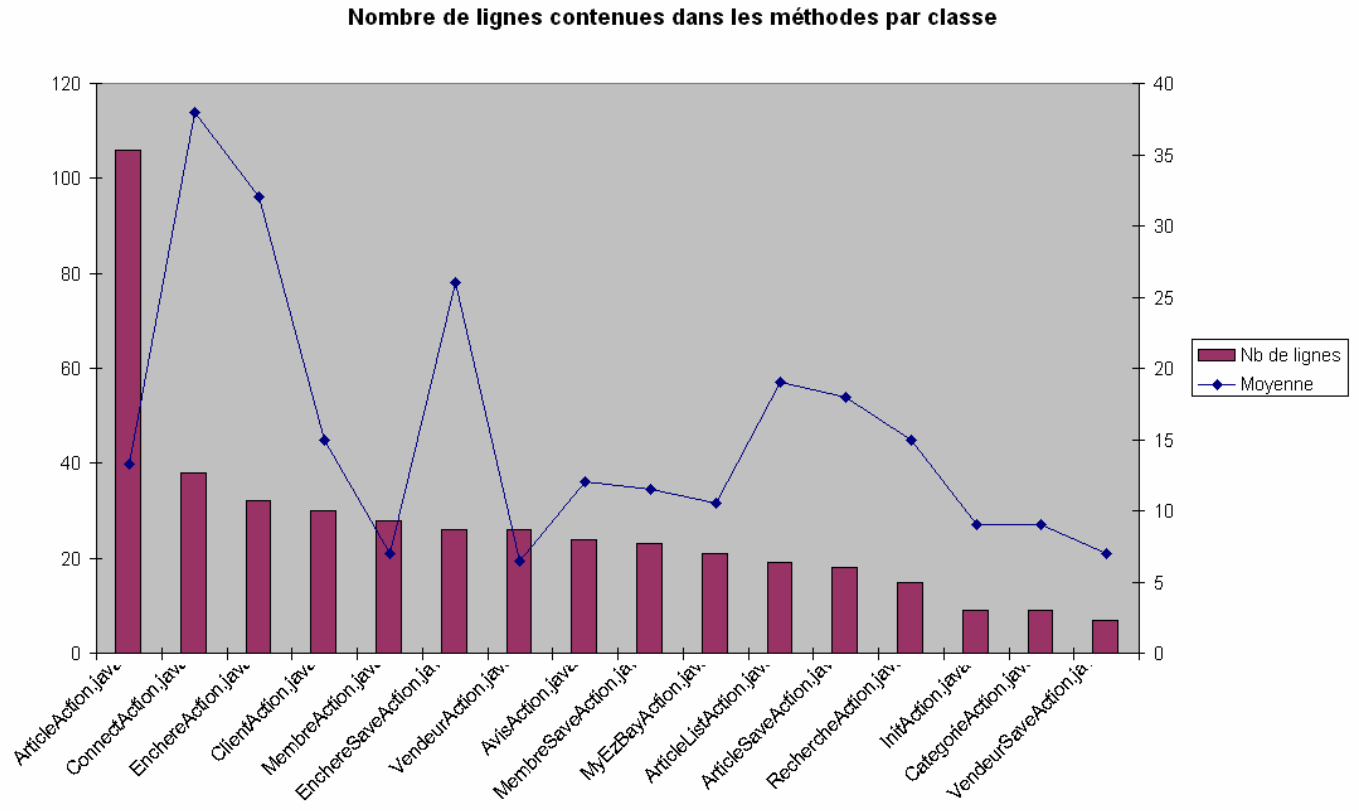
- de passer un article de l'état : `enVente` à l'état : `vendu`
- de créer une nouvelle transaction (et supprimer l'enchère correspondante)
- d'ajouter l'article au panier du dernier enchérisseur



Le package `axlomoso.ezbay.struts.form` contient 181 méthodes représentant 459 lignes de code c'est-à-dire en moyenne 2,5 lignes de code par méthode. La méthode `validate` contenue dans la classe `membreForm.java` possède le maximum de ligne, c'est-à-dire 75 lignes.



Le package `axlomoso.ezbay.struts.action` contient 181 méthodes représentant 431 lignes de code c'est-à-dire en moyenne 13 lignes de code par méthode. La méthode `confirmSupprimerArticle.java` contenue dans la classe `ArticleAction.java` possède le maximum de ligne, c'est-à-dire 28 lignes.



McCabe Cyclomatic Complexity

La complexité "Cyclomatic" (introduit par Thomas McCabe en 1976) mesure le nombre de chemins linéaires indépendants dans un module de programme. Elle est conçue pour être indépendante du langage et du format de langage.

Le tableau ci-dessous représente le nombre de branches conditionnelles dans chaque classe du package `axlomoso.ezbay.model.ejb`. Le nombre Cyclomatic d'une fonction devrait être inférieur à 15. Si une fonction a un nombre Cyclomatic de 15, il y a au moins 15 chemins d'exécution dans son contenu. Dans notre cas, le nombre maximum (dans le package) est de 6, ce qui signifie que le nombre de chemins d'exécution dans les classes est raisonnable. En effet plus le nombre Cyclomatic est élevé, plus le nombre de chemins d'exécution est important ce qui implique que le code sera compliqué à comprendre.

Metric	Total	Mean	Std. Dev.	Maximum
⊟ axlomoso.ezbay.model.ejb		1.257	0.714	6
⊞ ArticleFacadeBean.java		2.289	1.293	6
⊞ VendeurFacadeBean.java		2.2	1.327	5
⊞ MembreFacadeBean.java		2.176	1.15	5
⊞ ActionEnchereFacadeBean.java		1.818	0.936	4
⊞ TimerFinVenteBean.java		1.75	1.09	4
⊞ ClientFacadeBean.java		1.909	0.793	3
⊞ CategorieFacadeBean.java		1.636	0.643	3
⊞ ActionEnchereCMP.java		1.062	0.242	2
⊞ ArticleCMP.java		1.018	0.132	2
⊞ MembreCMP.java		1.029	0.167	2
⊞ VendeurCMP.java		1.053	0.223	2
⊞ ClientCMP.java		1.1	0.3	2
⊞ ActionTransactionFacadeBean.j		1.545	0.498	2
⊞ ActionTransactionCMP.java		1.059	0.235	2
⊞ CategorieCMP.java		1.077	0.266	2
⊞ ClientFacadeSession.java		1	0	1
⊞ ActionTransactionBean.java		1	0	1
⊞ ClientBean.java		1	0	1
⊞ VendeurBean.java		1	0	1
⊞ ArticleBean.java		1	0	1
⊞ TimerFinVenteBeanSession.java		1	0	1
⊞ TimerFinVenteSession.java		1	0	1
⊞ ActionEnchereBean.java		1	0	1
⊞ VendeurFacadeSession.java		1	0	1
⊞ CategorieFacadeSession.java		1	0	1
⊞ CategorieBean.java		1	0	1
⊞ ActionTransactionFacadeSessior		1	0	1
⊞ ArticleFacadeSession.java		1	0	1
⊞ ActionEnchereFacadeSession.ja		1	0	1
⊞ MembreFacadeSession.java		1	0	1
⊞ MembreBean.java		1	0	1

7.2. Bilan de l'application ezBay

7.2.1. *Points forts*

Nous avons cherché à mettre en pratique un certain nombre de règles de bonne conduite afin de réaliser une application proche d'un contexte professionnel. A tout moment, plutôt que de « développer vite », nous avons cherché à « développer bien », en donnant une grande importance au côté pédagogique de l'exercice.

Architecture n-tiers : Le projet ezBay a été l'occasion pour nous de mettre en oeuvre ce type d'architecture. Citons en quelques avantages.

- utilisation d'interfaces utilisateurs riches
- séparation nette de tous les niveaux de l'application
- grandes capacités d'extension
- facilité de la gestion des sessions

Design patterns : Nous avons mis en oeuvre un certain nombre de modèles de conception.
Citons :

- Façade : dans la couche métier de l'application, classes permettant de faire abstraction de la structure du modèle de données.
- Business Delegate : dans la couche cliente de l'application, classes donnant accès aux méthodes métier. Ce modèle permet de bien séparer les développements de la couche « métier » et de la couche « client ». La mise en œuvre de ce modèle de conception permet de bien séparer le travail des équipes de développements dans un projet.
- Data Transfer Object (DTO) : diminuer le nombre d'accès en lecture à la base de données . Un objet DTO est un objet qui contient (en mémoire) tous les attributs de persistance d'un objet métier. Ainsi, l'accès à chacun des attributs (nom, prénom d'un membre par exemple) se fait en mémoire et ne nécessite pas de nouvel accès à la base de données.
- Service Locator : Objet chargé de stocker en mémoire les interfaces d'accès à la couche de persistance. Ainsi, au lieu d'instancier chacune des classes d'accès à chaque fois que l'application le nécessite, le Service Locator gère un cache en mémoire, auquel les classes utilisatrices font implicitement appel. Chaque classe d'accès n'est instanciée que lors de sa première utilisation, puis est conservée en mémoire pour un accès ultérieur plus rapide.
- Singleton : Certaines classes sont utilisées plusieurs fois, alors qu'il n'est pas nécessaire (ou qu'il ne faut pas) les ré-instancier. Le pattern Singleton, par l'utilisation d'une méthode gestInstance(), permet de stocker en mémoire une instance de la classe, et de la retourner aux classes utilisatrices, empêchant ainsi son instanciation multiple.
  Les classes « delegate » (voir plus haut) sont utilisées très fréquemment par l'application cliente. Celles-ci implémentent le pattern Singleton.

Best practices : Tout au long des développements du projet, nous avons tâché de nous inspirer de « Best Practices » diffusées sur Internet, notamment par la communauté Open Source.

Framework Struts : L'application cliente a été développée selon le framework de développement Struts. Celui-ci, implémentant le modèle MVC, propose un grand nombre de fonctionnalités natives, dont l'application finale bénéficiera. Citons par exemple un contrôle puissant des saisies de formulaire, une gestion avancées de erreurs (affichage au client), une gestion de mise en mémoire des objets java (« beans »), un mapping avancé des URL, une gestion multi-langue, une gestion implicite des sessions navigateur (avec ou sans cookies).

7.2.2. *Améliorations possibles*

Redondance d'information : Afin d'accélérer l'affichage des informations concernant les articles (nombre d'enchères ; montant, date et client de la dernière enchère …) nous avons pris le parti de dupliquer un certain nombre d'attributs dans la table « Article ». Cette redondance nous a permis de diminuer le nombre de jointures à effectuer pour l'affichage des articles, optimisant ainsi la vitesse d'exécution des requêtes de lecture de la base de données. Ces informations sont évidemment mises à jour à chaque nouvelle écriture dans la base de données. Nous sommes tout à fait conscients que cette solution n'est pas optimale. Dans un contexte de pérennité, nous aurions cherché une solution plus efficace.

Accès à distance : L'application utilise actuellement un accès local à la couche de persistance. En effet, le client web (struts) fait référence au serveur « Localhost ». Nous prévoyons de mettre en place la séparation application cliente / couche de persistance pour la soutenance du projet.

Optimisation des accès à la base de données : Nous n'avons pas pour le moment cherché à optimiser les lectures depuis la base de données. En effet, chaque requête de l'utilisateur via le client web se solde par un accès en lecture à la base de données. Dans un contexte de pérennité, cet aspect aurait été abordé afin de stocker un certain nombre d'informations en mémoire (session navigateur) pour permettre leur réutilisation.

Itération 03 : Les développements ont été stoppés à l'itération 02. L'itération 03 comportait un certain nombre d'améliorations, notamment en terme de l'IHM du client web. Celles-ci ne seront donc pas disponibles.

7.3. Carnet de bord

7.3.1. *Fichier « openPoints » : suivi des tâches*

EzBay : openPoints
Suivi de l'avancement

itération00

userStory	Etat	Reste à faire	Remarque	resp
User_Strories\Iteration00\ezBayit00_us01[Membre].doc	terminé			Axel, Lotfi, Sophie
User_Strories\Iteration00\ezBayit00_us02[Articles].doc	terminé			Axel, Lotfi, Sophie
User_Strories\Iteration00\ezBayit00_us03[Membre-Articles].doc	terminé			Axel, Lotfi, Sophie

itération01

userStory	Etat	Reste à faire	Remarque	resp
ezBayit01_us01[Membre].doc	terminé		A réaliser dans it02	Lotfi, Sophie, Axel
ezBayit01_us02[Articles].doc	en cours	recherche : caractères spéciaux	A réaliser dans it02	Lotfi
ezBayit01_us03[fiches].doc	en cours	a compléter	A réaliser dans it02	Sophie
ezBayit01_us04[design].doc	**non commencé**		**A réaliser dans it02**	**Mohamed**
ezBayit01_us05[Membre_desinscription].doc	**non commencé**		**A réaliser dans it02**	**Mohamed**

itération02

userStory	Etat	Reste à faire	Remarque	resp
User_Strories\Iteration02\ezBayit02_us01[Membre_desinscription].doc	**non commencé**	**désinscription**	**A réaliser dans it03**	**Mohamed**
User_Strories\Iteration02\ezBayit02_us02[design].doc	terminé			Sophie
User_Strories\Iteration02\ezBayit02_us03[encheres].doc	terminé			Axel
User_Strories\Iteration02\ezBayit02_us03bis[transactions].doc	terminé			Axel
User_Strories\Iteration02\ezBayit02_us04[Articles].doc	terminé			Lotfi
User_Strories\Iteration02\ezBayit02_us05[Membre].doc	terminé			Lotfi

itération03

userStory	Etat	Reste à faire	Remarque	resp
User_Strories\Iteration03\ezBayit03_us01[Membre_desinscription].doc	**non commencé**	**désinscription**	**A réaliser dans it03**	**Mohamed**
User_Strories\Iteration03\ezBayit03_us02[Fiche-Membre].doc	**non commencé**			**Lotfi**
User_Strories\Iteration03\ezBayit03_us03[encheres].doc	**non commencé**			**Axel**
User_Strories\Iteration03\ezBayit03_us04[Articles].doc	**non commencé**			**Sophie**

7.3.2. *Itérations et " User Stories "*

Cf. fichier « userStories_recapitulatif »
Fichier téléchargeable à l'adresse :
http://axel.ammuller.free.fr/ezbay/userStories_recapitulatif.pdf

7.3.3. *Itérations et rapports intermédiaires*

Cf. fichier « iterations_rapport_recapitulatif »
Fichier téléchargeable à l'adresse :
http://axel.ammuller.free.fr/ezbay/iterations_rapport_recapitulatif.pdf