Architecture des Applications Réparties
Rapport de Projet
Gestion d'un tournoi de football

Geoffrey CROCHET, Zo RABARIJAONA, Willy FRANÇOIS

8 décembre 2013

Table des matières

Introduction

Le projet demandé consiste à développer une application permettant la gestion d'un tournoi de football. Pour ce faire, nous nous appuierons sur la technologie J2EE et l'utilisation des EJB 3.0 afin de pouvoir persister des objets dans la base de données interne du serveur d'application JBoss.

Ce rapport présentera tout d'abord une analyse UML de l'ensemble de notre application en visualisant les diagrammes de classes, le diagramme de cas d'utilisation ainsi qu'un schéma synthétique de l'architecture du projet. Nous évoquerons ensuite comment le temps de travail a été géré et la répartition des différentes tâches au sein du groupe. Enfin, un guide d'utilisation sera présenté expliquant le fonctionnement général de l'application.

Chapitre 1

Analyse UML

1.1 Diagrammes de classes

Notre application se compose de trois principaux packages : le modèle contenant des objets POJO, les EJB permettant la persistance des objets et les Value Objects utilisés dans les vues de l'application.

1.1.1 Le modèle

Le schéma ci-dessous représente le diagramme de classes de notre modèle.

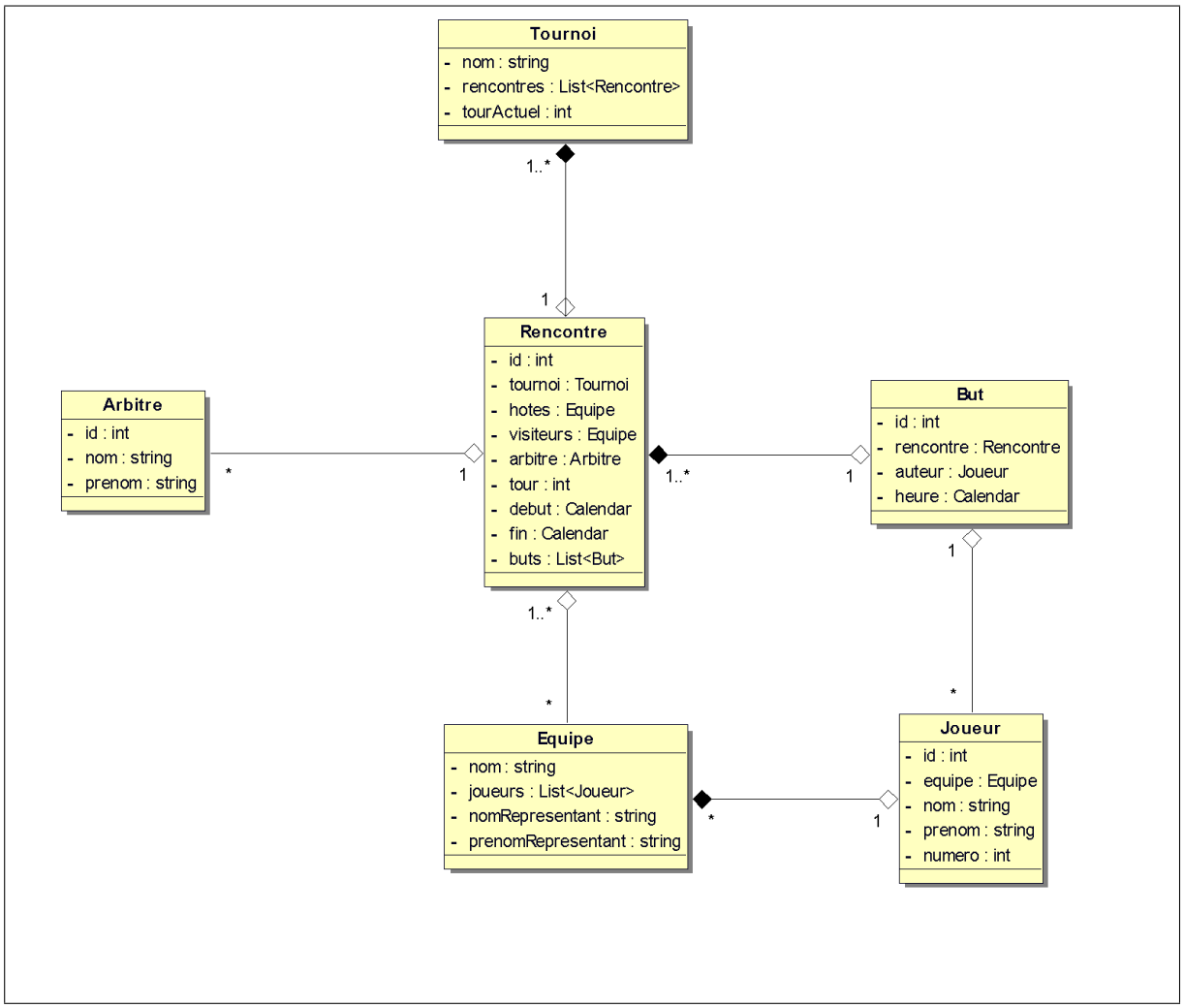


Diagramme de classes du modèle

Un *Tournoi* est composé d'un nom, d'une liste de rencontres et le tour actuel afin de pouvoir l'incrémenter en temps voulu pour les nouvelles rencontres.

Une *Rencontre* garde une référence vers l'équipe hôte et l'équipe visiteur qui vont s'affronter. Elle possède aussi l'arbitre, une liste de buts et les dates de début et de fin. La date de fin est calculée en fonction des buts marqués. On détecte automatiquement s'il y a eu prolongation ou tirs aux buts.

Une *Equipe* contient une liste de joueurs et le nom et prénom du représentant.

Un *But* contient le joueur qui l'a marqué et la date (heure) associée.

Le *Joueur* et l'*Arbitre* sont simplement composés d'un nom et d'un prénom.

1.1.2 Les EJB

Le schéma ci-dessous représente le diagramme de classes de l'EJB Utilisateur créé.

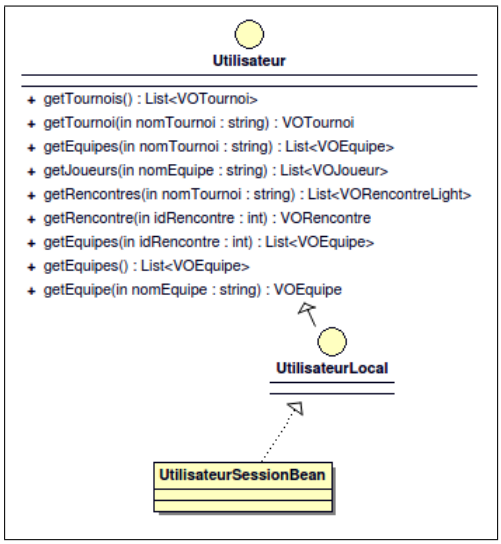
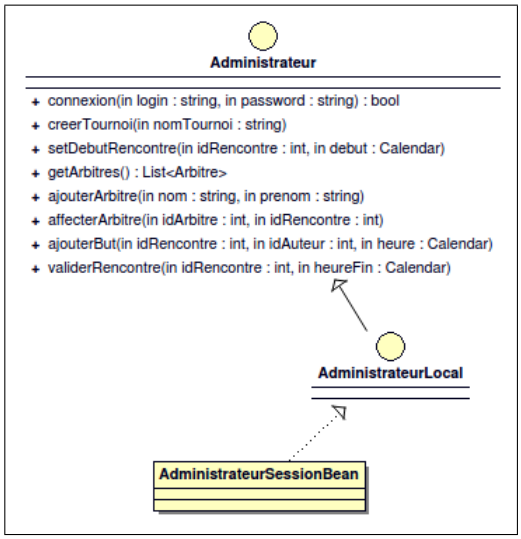


Diagramme de classes de l'EJB Utilisateur

L'EJB Utilisateur se compose de deux interfaces (*Utilisateur* et *UtilisateurLocal*) et d'une classe d'implémentation (*UtilisateurSessionBean*). L'interface *Utilisateur* définit les profils des méthodes implémentées par la classe *UtilisateurSessionBean*. L'interface *UtilisateurLocal* correspond à l'interface locale de l'EJB utilisée par la facade de notre application. L'ensemble des méthodes définies dans cet EJB retournent un Value Object ou une liste de Value Objects. En effet, l'utilisateur ne doit pas être en mesure de connaitre l'ensemble des attributs et méthodes des classes du modèle.

Le schéma ci-dessous représente le diagramme de classes de l'EJB Administrateur créé.



Diagrammes de classes de l'EJB Administrateur

L'EJB Administrateur se compose de deux interfaces (*Administrateur* et *AdministrateurLocal*) et d'une classe d'implémentation (*AdministrateurSessionBean*). L'interface *Administrateur* définit les profils des méthodes implémentées par la classe *AdministrateurSessionBean*. L'interface *AdministrateurLocal* correspond à l'interface locale de l'EJB utilisée par la facade de notre application. Les méthodes définies concernent la connexion de l'administrateur à l'application, la création d'un tournoi ou encore le déroulement des matchs (indication des horaires, affectation des arbitres).

Le schéma ci-dessous représente le diagramme de classes de l'EJB Représentant créé.

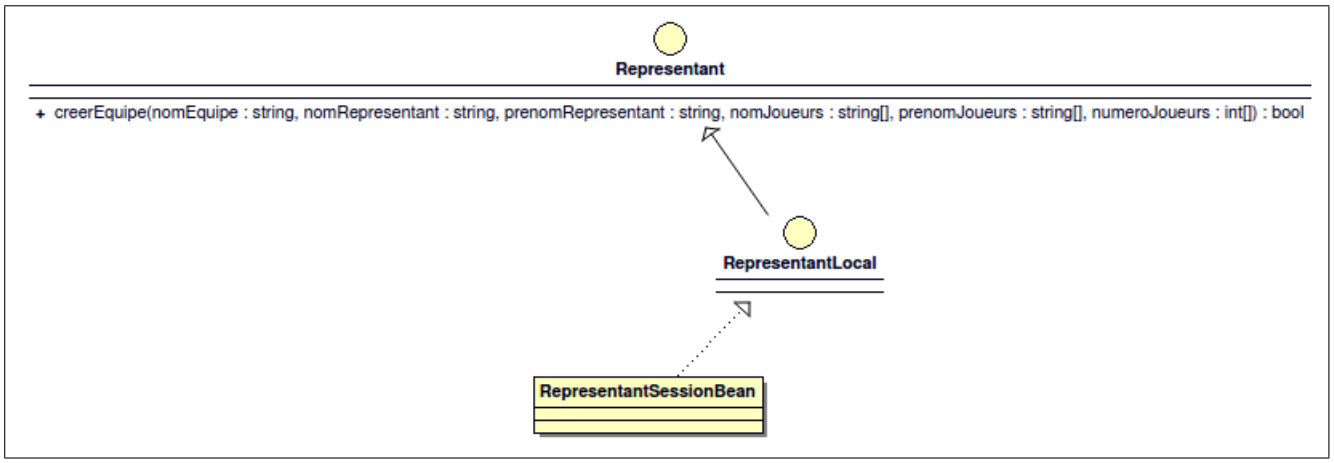


Diagramme de classes de l'EJB Représentant

L'EJB Représentant se compose de deux interfaces (*Representant* et *RepresentantLocal*) et d'une classe d'implémentation (*RepresentantSessionBean*). L'interface *Representant* définit le profil de la méthode implémentée par la classe *RepresentantSessionBean*. L'interface *RepresentantLocal* correspond à l'interface locale de l'EJB utilisée par la facade de notre application. La méthode définie consiste à la création d'une équipe au sein de l'application.

Le schéma ci-dessous représente le diagramme de classes de l'EJB session Facade créé.

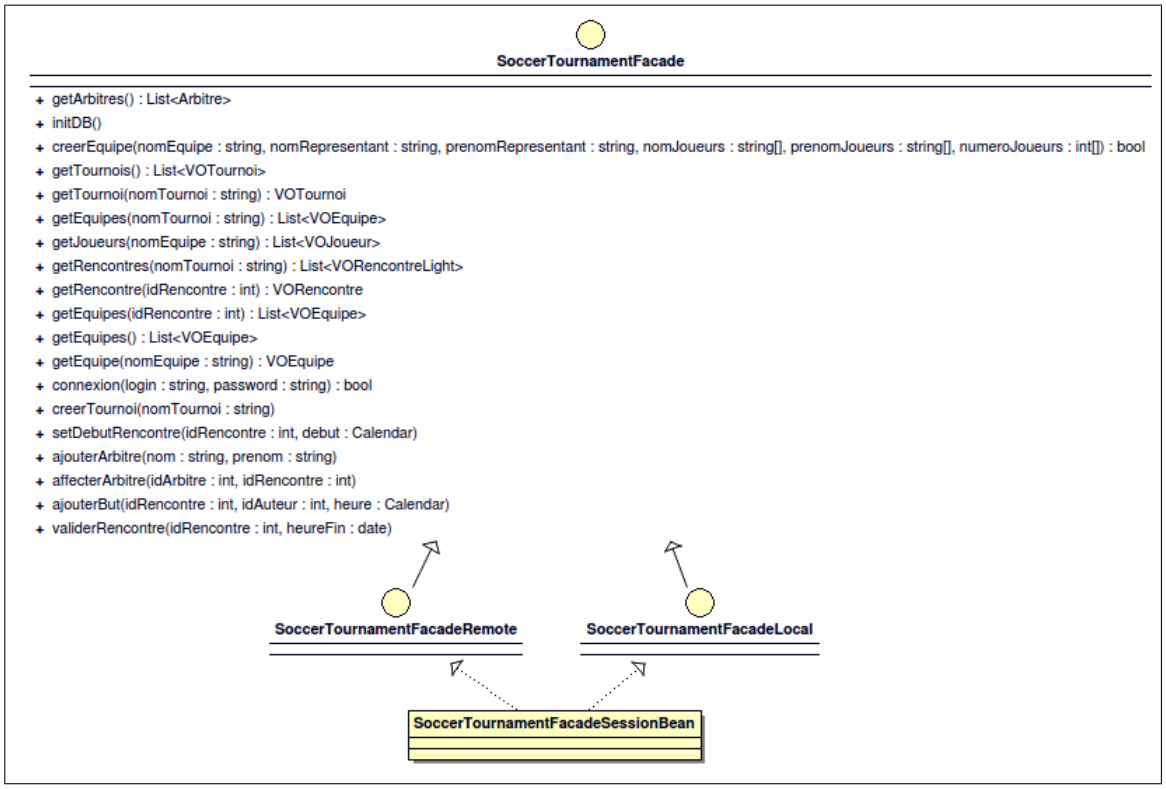


Diagramme de l'EJB session Facade

L'EJB session Facade se compose de trois interfaces (*SoccerTournamentFacade*, *SoccerTournamentFacade-Local* et *SoccerTournamentFacadeRemote*) et d'une classe d'implémentation (*SoccerTournamentFacadeBean*). L'interface *SoccerTournamentFacadeLocal* correspond à l'interface locale de l'EJB utilisée dans notre application. Nous avons également créé l'interface *SoccerTournamentFacadeRemote* dans le cas où nous devions atteindre la façade à distance. Cet EJB regroupe l'ensemble des méthodes définies dans les EJB créés ainsi qu'une méthode *InitDB()* nous permettant d'initialiser la base de données au lancement de l'application. La base de données se charge ainsi, à l'aide de fichiers XML préalablements remplis.

1.1.3 Les Value Objects

Le schéma ci-dessous représente le diagramme de classes des Value Objects créés.

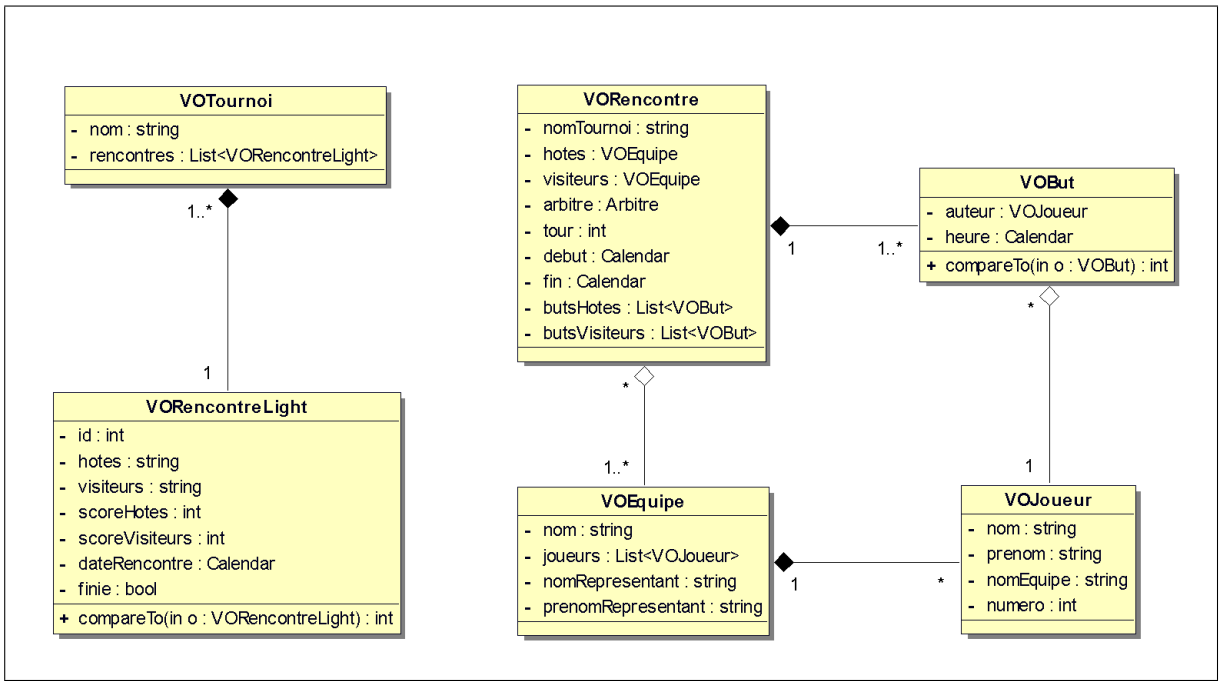


Diagramme de classes des Value Objects

Les Value Objects permettent de ne donner aux JSP que les données nécessaires. Les nôtres sont divisés en deux groupes.

La **VORencontre** possède les informations d'une rencontre et les classes de type **VOEquipe** et **VOBut**. Nous avons gardé la classe **Arbitre** telle quelle car elle n'avait aucun lien vers les autres classes du modèle et ne représentait donc pas un risque. La **VOEquipe** contient une liste de **VOJoueur** et la **VOBut** un **VOJoueur**. Le **VOJoueur** est un joueur dans lequel la référence à l'équipe a été remplacée par son nom.

D'un autre coté, nous avons la **VOTournoi** qui contient une liste de **VORencontreLight**. Cette classe, plus légère, ne contient que le nom des équipes et quelques informations telles que les scores et l'état de terminaison de la rencontre. Il était inutile de garder un lien vers les équipes alors qu'on ne veut afficher que peu d'informations.

1.2 Diagramme de cas d'utilisation

Le schéma ci-dessous représente le diagramme de cas d'utilisation de notre application.

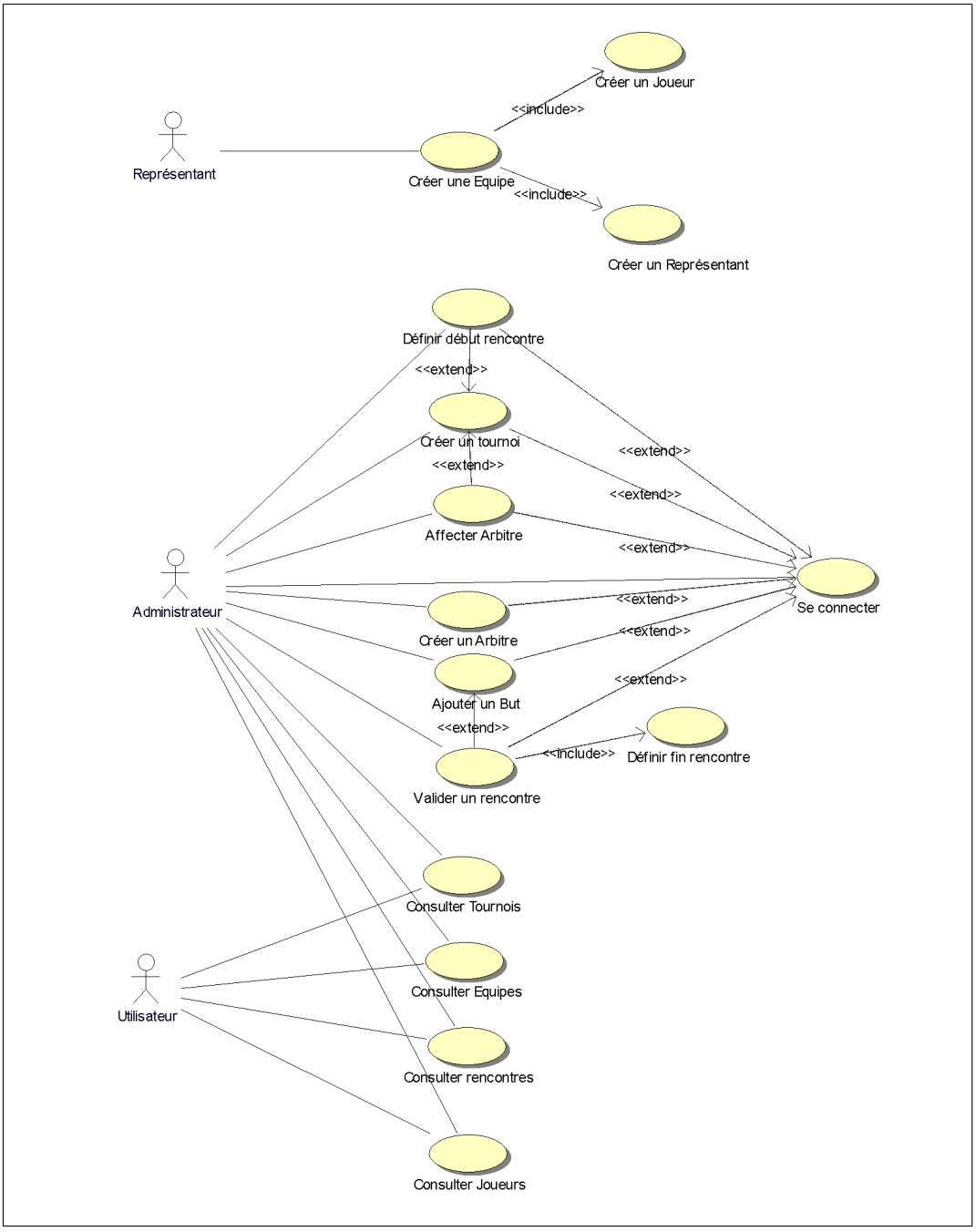


Diagramme de cas d'utilisation

Ce diagramme nous a permis de définir les différents EJB et les méthodes nécessaires à chacun.

Pour commencer, le **Représentant** peut créer une équipe, créant ainsi les joueurs de l'équipe et le représentant.

Ensuite, l'**Administrateur** peut créer un tournoi, définir le début d'une rencontre, y ajouter des buts et la valider. De plus, il peut créer un arbitre et l'affecter à une rencontre. Toutes ces actions ne peuvent se faire que s'il s'est connecté auparavant.

Pour finir, l'**Utilisateur** peut consulter les tournois, les rencontres, les équipes et les joueurs. L'**Administrateur** peut aussi effectuer les actions de l'**Utilisateur**.

1.3 Schéma synthétique de l'application

Le schéma ci-dessous représente le schéma synthétique de l'application développée.

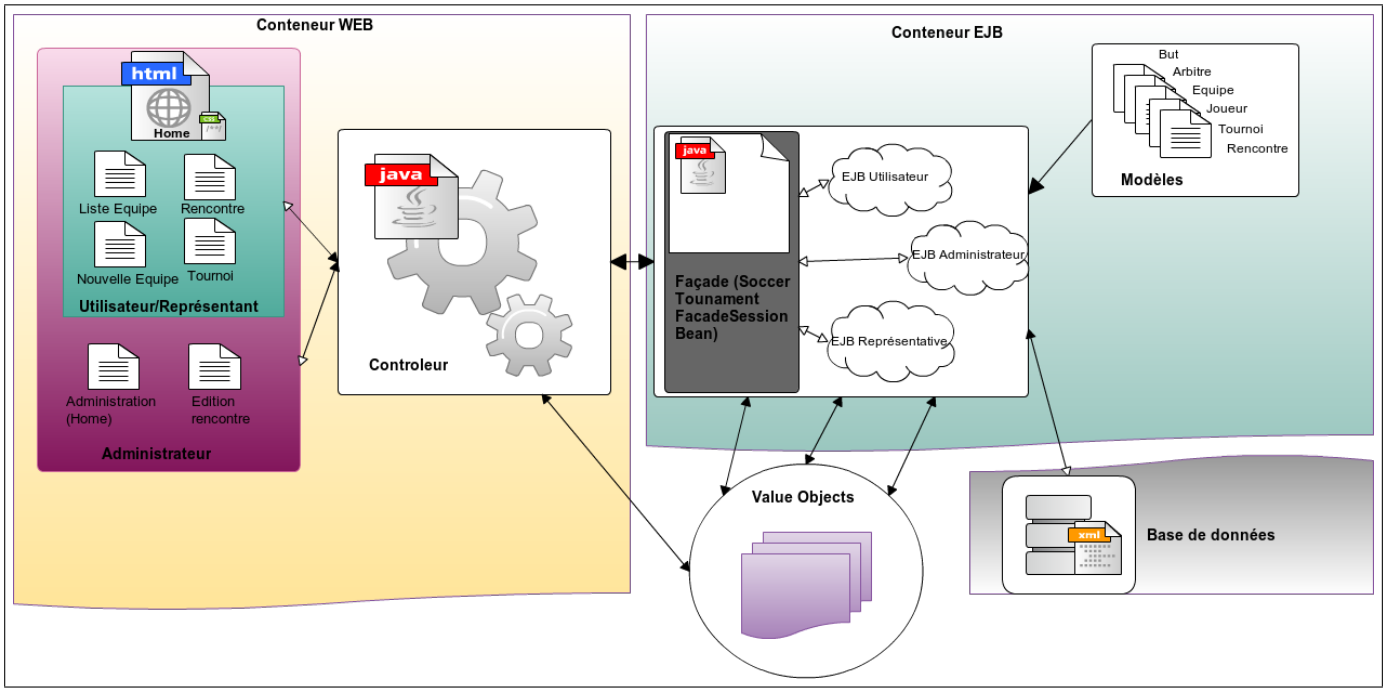


Schéma synthétique de l'application

Ce schéma présente une vision globale de l'application. Elle se décompose en trois grandes parties montrant ainsi les différentes couche de l'architecture.

- Le conteneur Web contient la couche vue « client léger » (JSP) et le controleur. Voici la description des différentes vues :

Vue		**Description**
Utilisateur	**Administrateur**	Les principaux acteurs
index.jsp		La page d'accueil
match.jsp		Affiche les rencontres du tournoi
team.jsp		Affiche la liste des équipes
tournament.jsp		Affiche la liste des tournoi
faq.jsp		Foire aux questions
	adminindex.jsp	Page d'accueil de l'administrateur
	adminmatch.jsp	Page d'administration des rencontres par l'administrateur
	login.jsp	Page d'authentification de l'administrateur
newteam.jsp		Page d'inscription pour une nouvelle équipe (par un représentant)

Description des vues de l'application

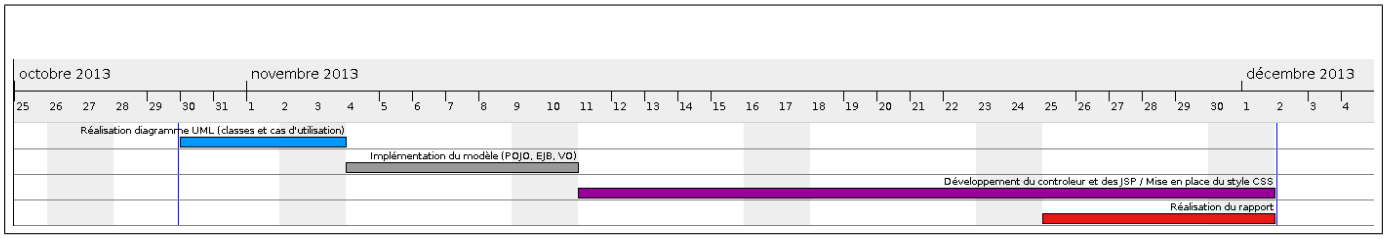
- Le conteneur EJB contient la couche métier (le modèle, les applications métiers et la couche persistante). On y trouve aussi les Value Objects permettant de récupérer des objets spécifiques. La couche métier est constituée d'une façade (*SoccerTournamentFacadeSessionBean*) permettant au controleur d'accéder aux EJB et aux Value Objects. Les EJB à leur tour accèdent à la couche persistante (base de données).

Chapitre 2

Répartition du travail

2.1 Gestion du temps de travail

Le diagramme de Gantt ci-dessous présente la répartition du travail tout au long du projet.



Gestion du temps de travail

La première semaine de travail fut consacrée à l'analyse UML et à l'élaboration des diagrammes de classes et des diagrammes de cas d'utilisation. Une fois cela fait, nous avons débuté l'implémentation du modèle de l'application à savoir les objets POJO, les EJB ainsi que les Value Objects. Le développpement du controleur et des vues utilisateurs et administrateurs a été réalisé ensuite jusqu'à la fin du projet. Le rapport, quant à lui, a été rédigé dans la dernière semaine de travail.

2.2 Répartition des tâches

L'ensemble du groupe a travaillé sur la réalisation de l'analyse UML et la détermination des différentes classes utiles dans l'application. Geoffrey s'est vu attribuer l'EJB Utilisateur alors que Zo s'est vu faire le Représentant et une partie de l'Administrateur. Zo et Willy se sont occupés des entités. Geoffrey et Willy se sont chargés de l'implémentation des Value Objects. Willy a également créé le jeu de test au travers de fichiers XML permettant de charger la base de données et la validation de rencontre.

Au niveau de la partie web, Zo s'est principalement occupé de la mise en place du CSS et du développement des JSP *newteam*, *index*, *login* et *team* pour la visualisation des équipes, des joueurs, la création d'une équipe et la connexion. Willy a travaillé sur *adminmatch*, qui permet à l'administrateur d'éditer une rencontre, une taglib permettant d'afficher le diagramme d'un tournoi et la réalisation d'une foire aux questions. Geoffrey a développé la JSP *match* coté utilisateur, affichant l'ensemble des informations relatifs à une rencontre.

L'ensemble du groupe a contribué à la création du controleur ainsi que tu déboguage des différentes classes et jsp et la rédaction du rapport.

Chapitre 3

Guide d'utilisation

L'application est accessible par l'URL suivante :

http ://localhost :8080/SoccerTournamentWeb/

Vous devez au préalable lancer l'application sur un serveur d'application JBoss. Selon votre configuration, le port de l'application peut varier.

Pour accéder à notre foire aux questions (qui fait office de manuel), il vous suffit de cliquer sur l'icône en forme de point d'interrogation. Des renseignements sont aussi disponibles pour l'administrateur sur cette même page, il faut pour cela vous connecter. Pour vous connecter en tant qu'administrateur, il vous suffit de cliquer sur l'icône en forme de clé et d'entrer les renseignements suivants :

Login : admin
Password : root

Conclusion

Ce projet aura permis de mettre en oeuvre les connaissances vues tout au long du semestre. Le groupe a pu améliorer ses compétences dans l'utilisation de la technologie J2EE et a acquis les principales notions de persistence d'objets, dans une base de données, avec l'usage des EJB. L'intérêt des Value Objects a été compris lors de ce projet et, les membres du groupe ont également renforcés leurs compétences dans le domaine du Web en travaillant sur un controleur et ses JSP.