Projet SI4 Serveurs d'Entreprise

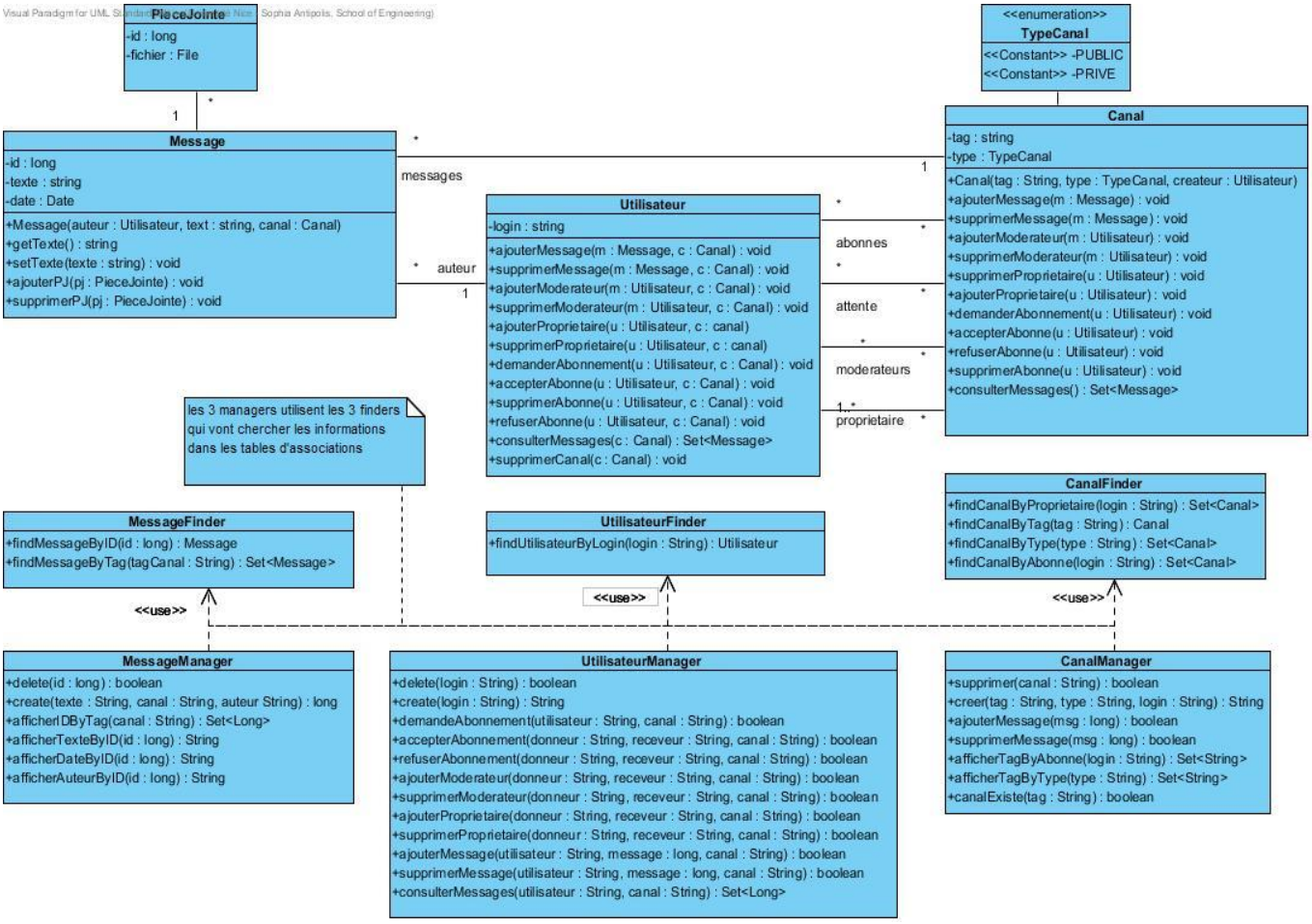
PolyTweet

Martin Alfonsi

Axel Vicard

SI4

Diagrammes de classes



Nous avons supprimé les classes Modérateur, Propriétaire et Administrateur et donc l'héritage. Ces classes complexifiaient le code et n'étaient au final pas nécessaires, les gestions des droits de l'utilisateur étant gérer grâce aux tables d'associations.

De plus, nous avons supprimé la classe PolyTweet ainsi que la classe Utilisateur non connecté. En effet, ces deux classes ne sont pas gérées dans le code Java mais dans la partie .NET. La classe PolyTweet correspond maintenant à notre client lourd WPF et un utilisateur est maintenant toujours connecté. Cependant, il est connecté en tant qu'anonyme et n'a accès qu'aux canaux publics tant qu'il ne s'est pas authentifié.

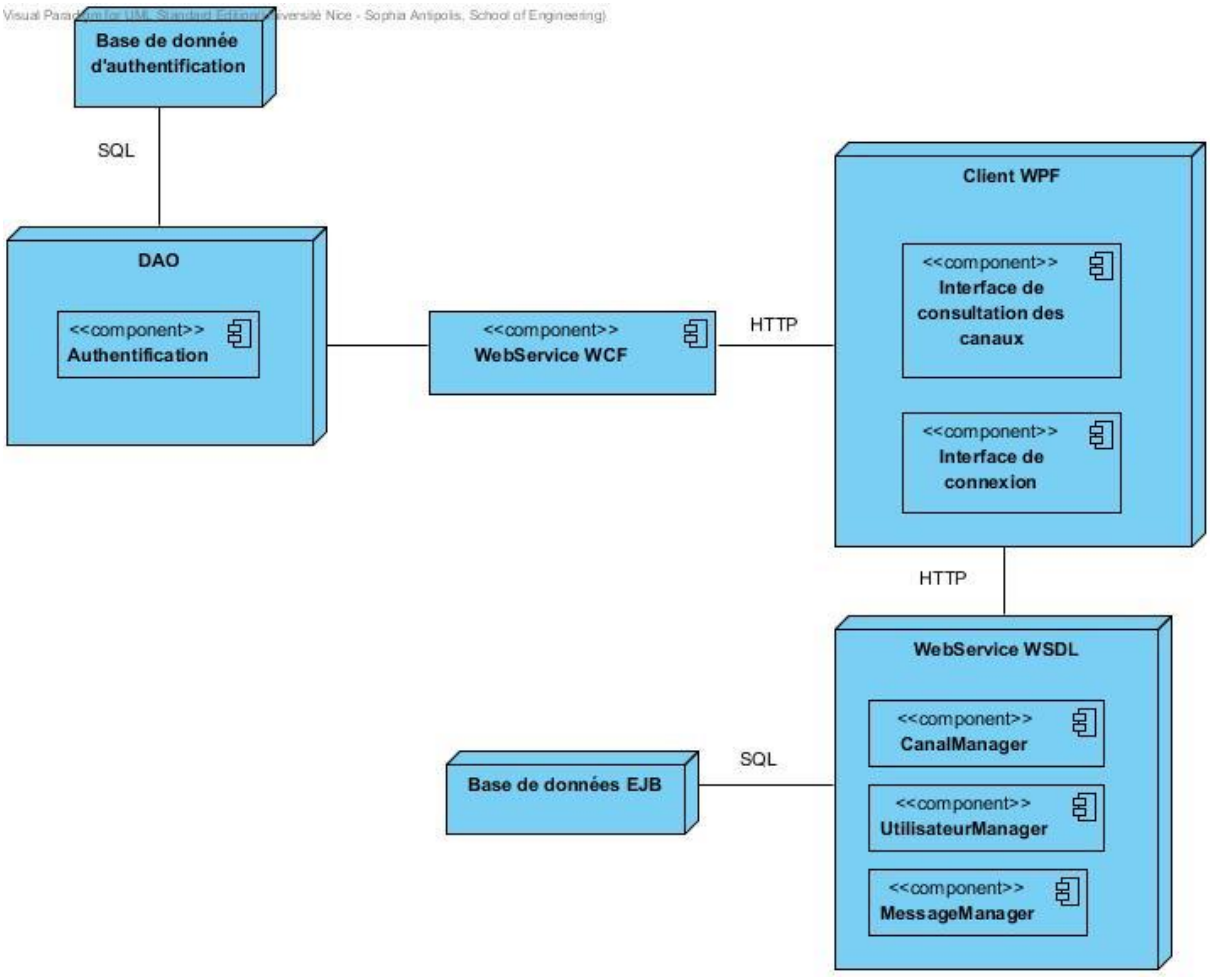
De plus nous avons intégré les Beans à notre projet et donc des Manager et des Finder. Les Finders servent à récupérer les entités (Canal, Message, Utilisateur) de notre base de données. Les Managers utilisent ces Finders afin de pouvoir accéder aux éléments de la base de données et les modifier. Le client WPF utilise les méthodes des Manager alors qu'il n'a pas accès aux entités, les arguments et retour de ces méthodes doivent donc être communes aux 2 environnements, on utilise donc uniquement des String, boolean et long.

Table de la base de données

Dans notre première architecture, nous avions une seule base de données avec toutes les tables nécessaires au projet. Au final, les utilisateurs, les canaux et les messages sont stockés en Java grâce à des EJB entités et gérés par des beans. Cependant, nous avons une seconde base de données créée avec SQL Server Management Studio qui contient les informations nécessaires à la connexion d'un utilisateur, à savoir le login et le password. La liaison entre les utilisateurs des 2 bases de données est faite via le login, qui est le même.

De plus, nous n'utilisons finalement plus le lazy loading mais le eager loading pour les behavorial patterns, car le lazy loading entraînait des problèmes de compilation.

Diagramme de déploiement



A l'origine, nous avions un serveur principal qui contenait tout notre code Java et un serveur de base de données qui communiquait en SQL avec le serveur principal. De plus l'authentification était gérée par le serveur de l'université qui communiquait en TCP/IP avec le serveur principal.

Au final, nous avons un client lourd WPF qui communique avec des WebServices, un WebService WCF pour l'authentification et des WebService WSDL accédant à la base de donnée EJB pour la gestion des utilisateurs, des canaux et des messages.