MasterOfDL
–
Nuit de l'info

6 décembre
2013

Document de conception respectant le processus CS-Neptune.
Document réalisé pour l'équipe MasterOfDL.

Défi CS

MASTER DÉVELOPPEMENT LOGICIEL

Master IHM

Sommaire

Introduction

Lors de la nuit de l'informatique 2013, nous avons effectué la conception d'une application en lien avec le e-commerce. Une description de notre application sera énoncée dans un premier temps, puis nous présenterons sa conception.C'est en suivant la méthode CS-Neptune que nous avons conçu et modélisé l'application nommée Trade-Game.

Description de l'application

Sujet

Le sujet de la nuit était le suivant : *Faite qu'Y-Commerce devienne une réalité, la nuit est à vous !*
Il s'agit donc d'apporter des innovations au commerce sur internet, faire en sorte que le produit vienne à l'utilisateur et plus l'inverse.

Interprétation du sujet

Nous partons du principe qu'une application web peut parfaitement proposer de manière séduisante un produit à un consommateur. Cependant nous avons distingués deux types d'utilisateur :
- L'utilisateur décidé : celui qui sait ce qu'il veut acheter, il est venu sur ce le site pour finaliser une idée déjà présente dans son esprit.
- L'utilisateur à séduire : Cet utilisateur est sur l'application car il veut trouver un produit sans cependant en avoir une idée précise.

Nous devons donc trouver des mécanismes permettant à au client de trouver un produit qui lui plaît parmi notre catalogue, sans passer par les étapes habituelles et fastidieuses de filtres / champs de recherche.

Objectifs de l'application : Trade-Game

Nous avons conçu une application qui met en place un jeu d'échange. Il s'agit d'une fonctionnalité complémentaire d'un site marchand normal : de CDiscount à Amazon. L'idée est d'avoir en plus des menus "rubriques" horizontal , un menu latéral où le Trage-Game pourra prendre place.

Description du Trade-Show

Le trade show est un jeu d'échange de produit entre les utilisateurs connectés et authentifiés a l'application.

Le trade-show démarre a un instant T et se termine à la fin d'un timer qui se veut long, une semaine par exemple. A l'instant T un **Produit** est attribué à l'utilisateur, provenant du catalogue de nos produits. Nous verrons par la suite que ce produit pourra être échangé plusieurs fois.
A la fin du temps imparti, une récompense s'applique sur le **TypeDeProduit** que l'utilisateur possède a ce moment précis.

Exemple : L'utilisateur Adolf, démarre une session de de trade Game, le timer indique une semaine. On lui attribue le produit suivant : Clé-USB de 4Go de marque A. Si le timer se termine et qu'il possède ce produit à ce moment là, une récompense lié au produit de Type : Clé-USB lui sera délivré, nous n'avons en l'occurrence pas définit de récompense précise (réduction en %, bon d'achat etc…).

Chaque jour, le produit courant d'un utilisateur pourra être échangé avec un autre selon un nombre limité de fois : **point d'échange**, l'application se chargera lorsque l'utilisateur de connecte, de proposer un échange avec un pair. Cette proposition d'échange peut être acceptée ou refusée. Exemple :

Adolf et Joesph sont deux utilisateurs connectés de mon application, Adolf détient le produit clé-USB 4Go de marque A et Joseph détient le best-seller du moment. Le système propose l'échange au deux clients :

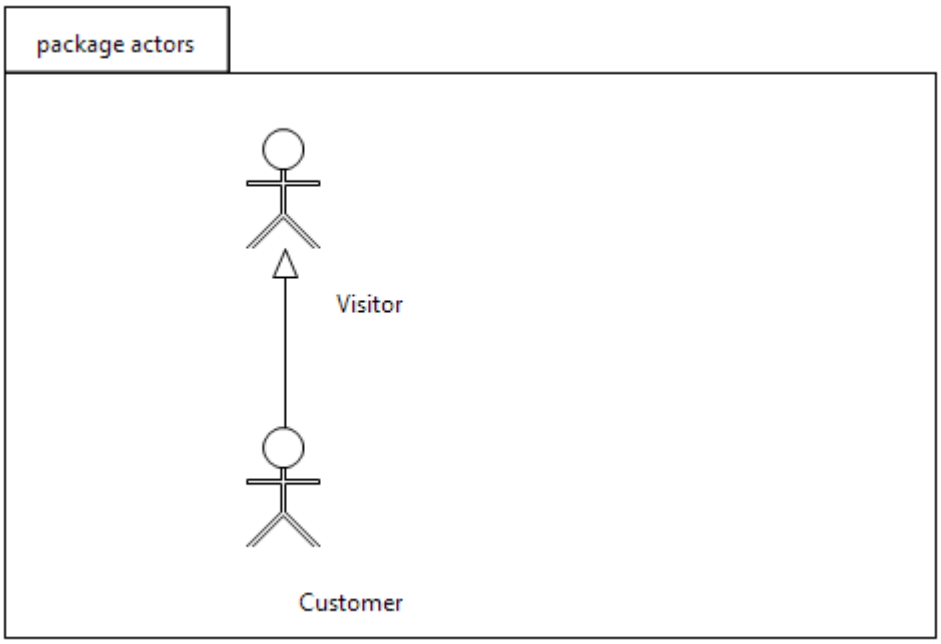
- Adolf se voit proposer le livre : il accepte
- Joseph se voit proposer la clé -USB : il accepte

L'échange est effectué et l'on donne le produit d'Adolf à Joseph et vice versa. De l'échange entre deux utilisateurs nous pourrons déduire des informations : en l'occurrence Adolf préfère l'article livre à l'article clé USB. De cet echange nous mettons en place un tableau de préférence pour chaque utilisateur qui nous servira dans le cadre de notre application normale de vente en ligne.

Nous mettons donc en place un jeu d'échange où l'utilisateur peut échanger son produit afin d'en obtenir un qui l'intéresse plus afin d'obtenir une récompense associée. Nous pensons ce jeu ludique et ainsi donc séduisant au yeux de l'utilisateur, qui consommera sans doute ses points d'échange journalier, et de notre coté il s'agit d'un bon moyen de connaître les

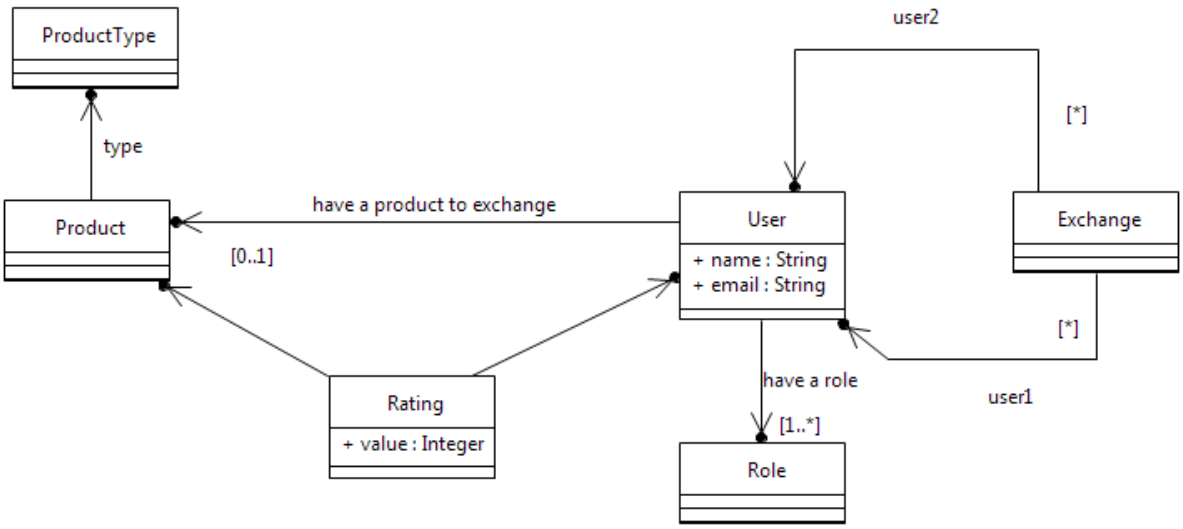
préférences utilisateur afin de lui proposer du contenu pertinent dans le reste de l’application (en plus du jeu de trade game).

Définition des acteurs



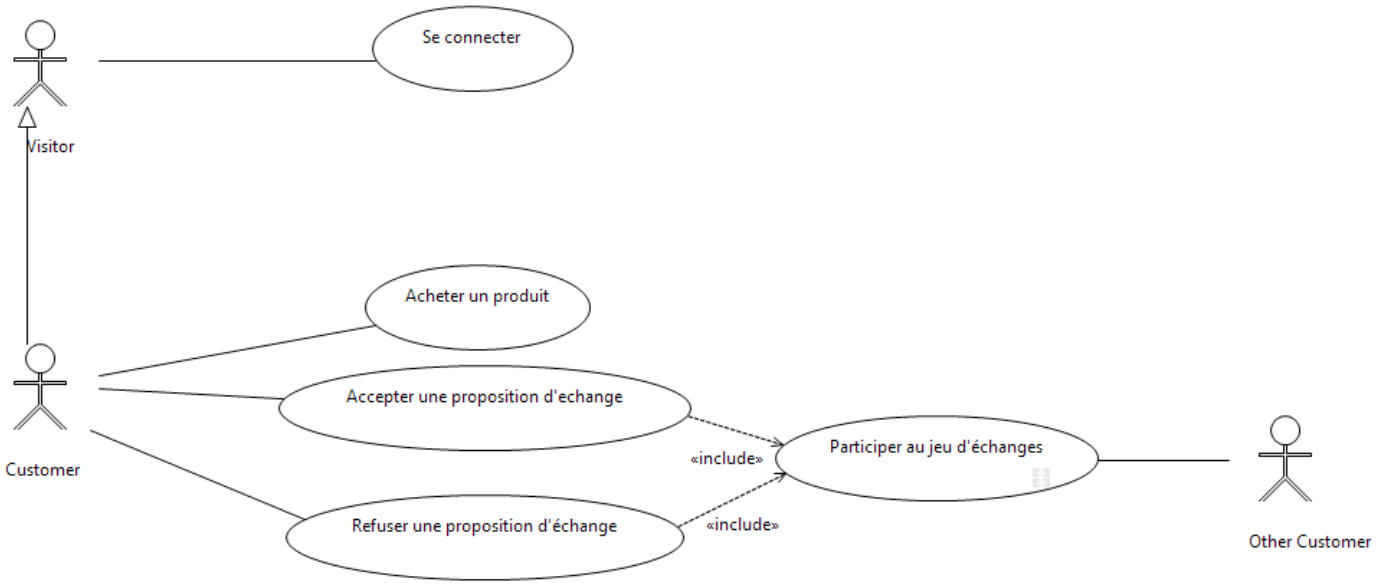
- Visiteur: Simple utilisateur qui n’est pas connecté.
- Customer: Visiteur qui est authentifié sur le site de e-commerce.

Analyse du domaine



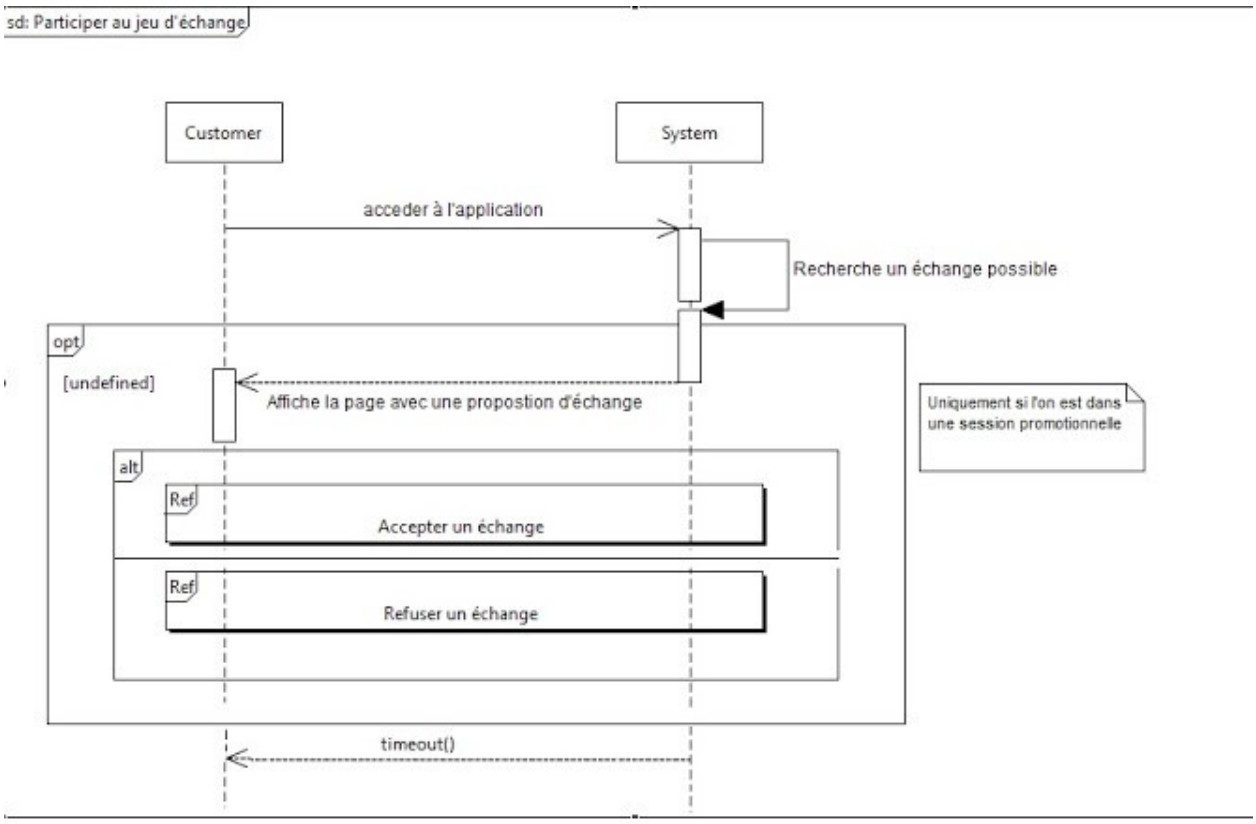
- User: Entité représentant un utilisateur.
- Role: Simplement ici pour permettre la gestion des droits utilisateurs (auto-généré).
- Product: Entité qui représente un produit précis de la boutique de e-commerce.
- ProductType: Représente la catégorie du produit. Cela permet de réunir les produits à l'intérieur d'une famille afin de déclencher la dite récompense sur celle-ci.
- Rating: Cette entité indique le niveau d'intérêt d'un utilisateur pour un produit. Plus ce niveau est élevé et plus l'intérêt de l'utilisateur pour le produit désigné est important. Ce niveau est le résultat des différents échanges effectués par l'utilisateur avec les autres utilisateurs.
- Exchange: Entité utilisée pour effectuer un échange de produit entre deux utilisateurs, permet d'abstraire la communication entre les modules (au lieu de passer par une structure obscure).

Modèle des cas d’utilisation

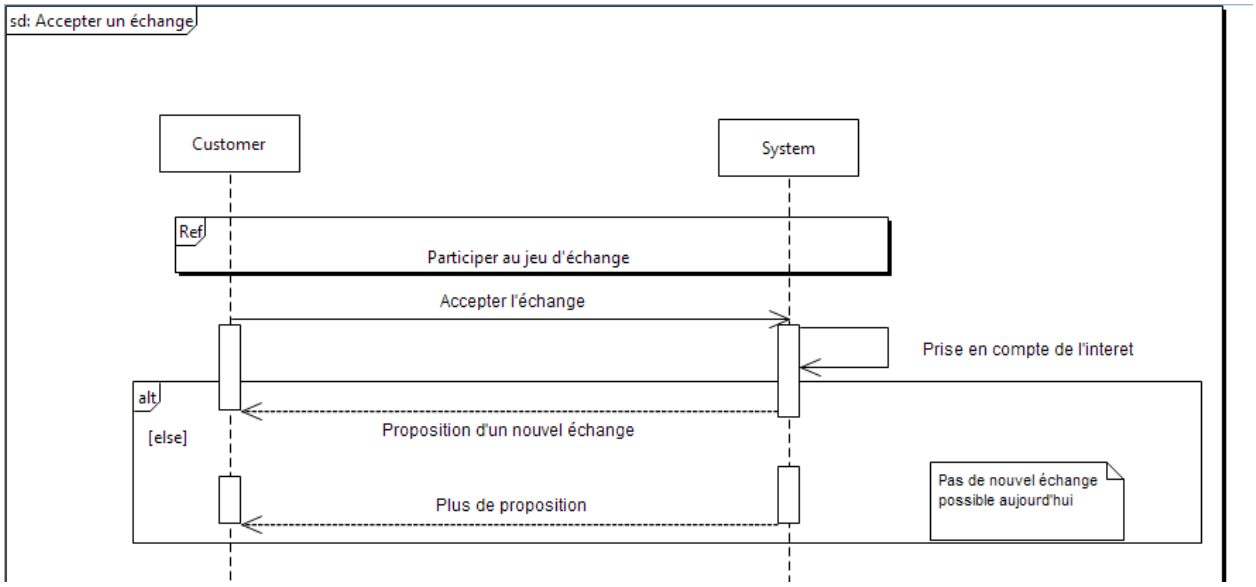


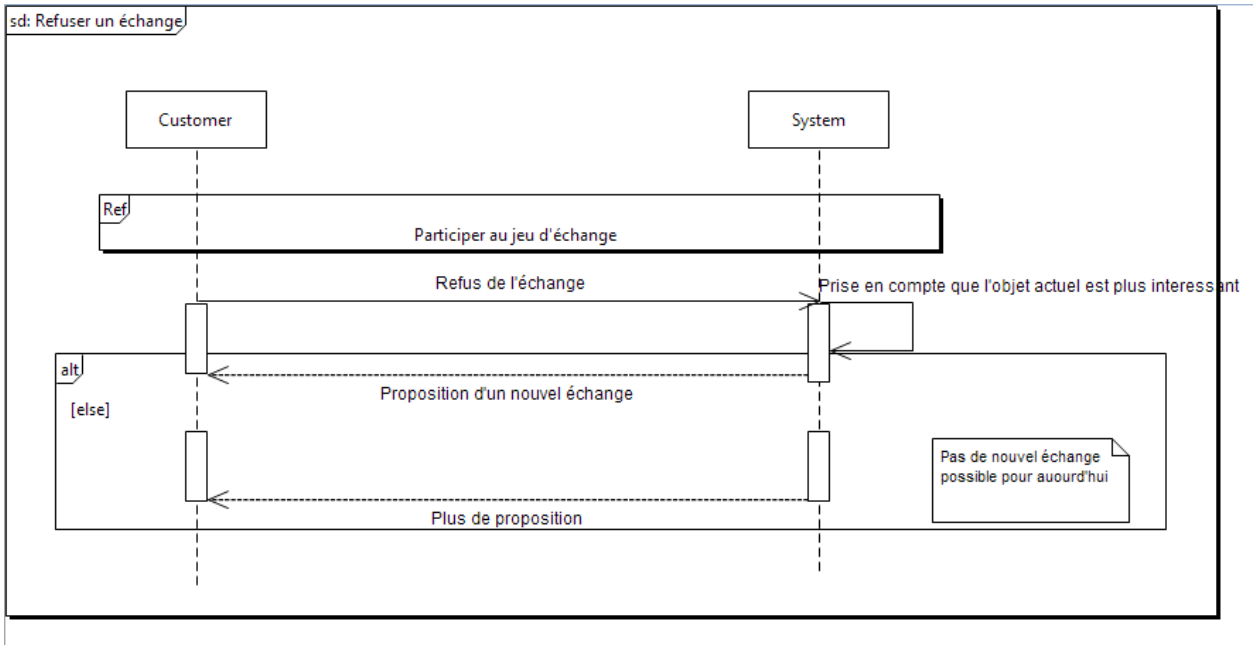
Nous avons décidé, au vu du temps qui nous est imparti, de nous concentrer sur quelques features qui nous semble nouvelles dans le domaine du e-commerce. A savoir celles qui concernent le système d’échange entre les clients.

Diagrammes de séquence des cas d'utilisation

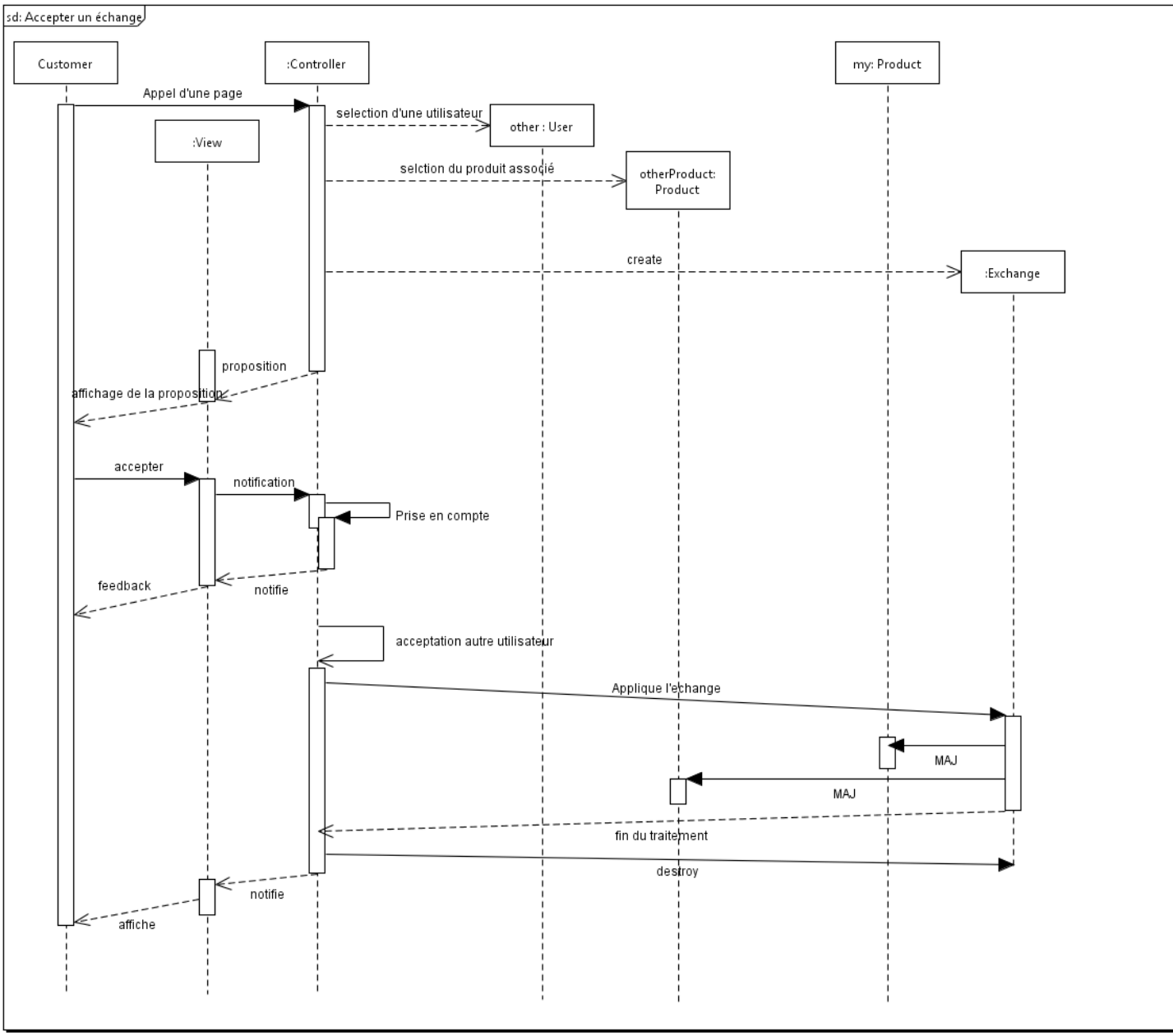


Ce diagramme de séquence système symbolise la mécanique globale du Tade Game qui consiste en l'échange de ristournes sur un produit avec un autre client.





Ces deux diagrammes expriment un comportement entre l'utilisateur et la proposition d'échange de produit : celle-ci peut être acceptée ou refusée, on peut donc ici voir une symétrie évidente.

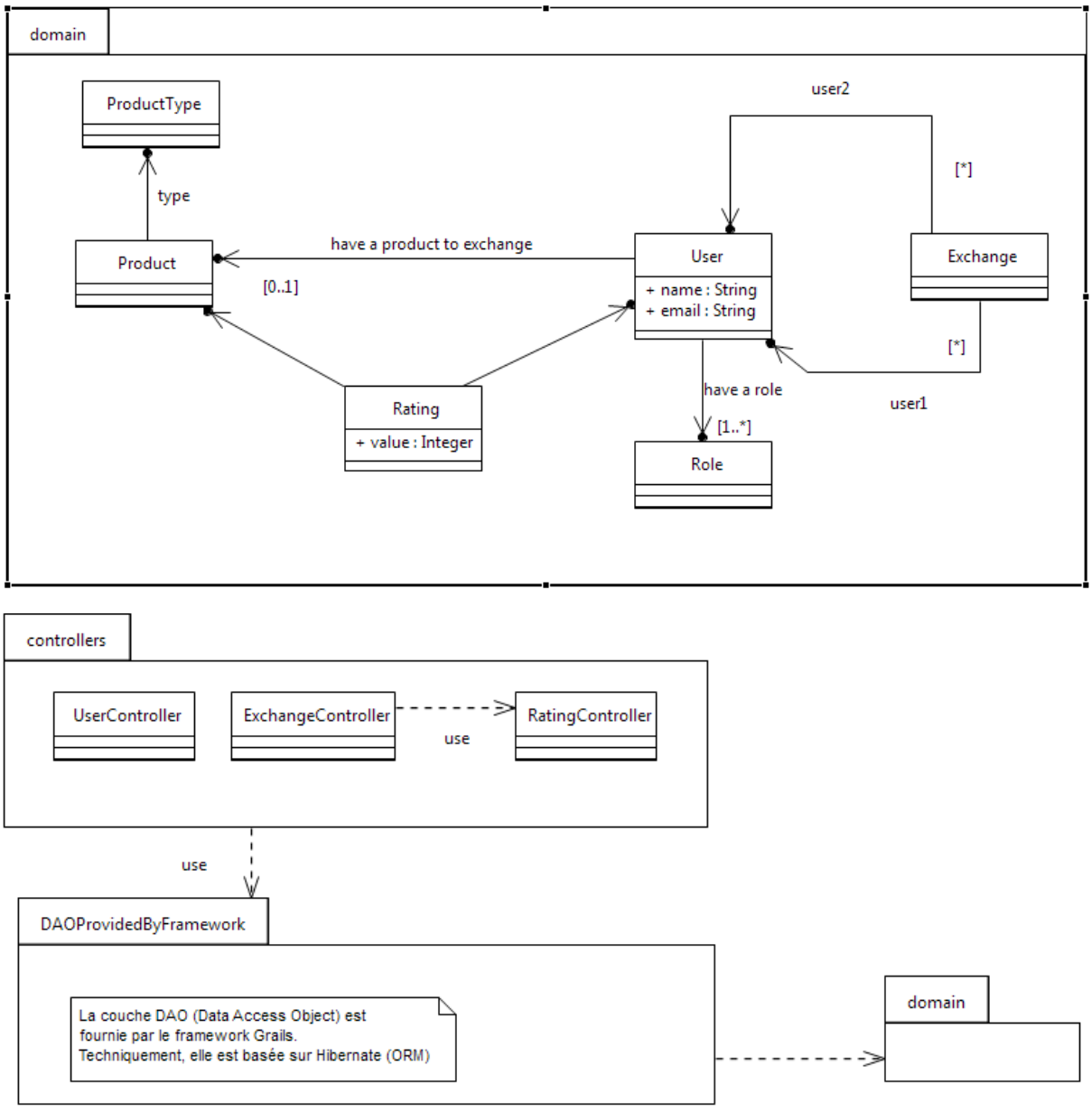


Commentaire : Le cas “accepter un échange” étant symétrique du cas “refuser un échange”, nous avons uniquement modélisé ce dernier. L’important ici est de voir le comportement du système vis a vis des classes du modèle : manipulation des utilisateurs et des produits associés.

Conception de l'architecture

A l'issu de la définition des cas d'utilisation et des diagrammes de séquence, nous avons pu valider le modèle défini précédemment. D'autres entités sont également apparues et il faut maintenant organiser toutes ces entités en packages.

Le choix d'une architecture MVC (plus ou moins imposée par le framework utilisé : Grails), nous donne une organisation préétablie de nos packages.



Conclusion

L’utilisation de la méthode Neptune nous a permis de concevoir cette solution de manière méthodique. De part la nature simple de notre application et compte tenu du temps qu’il nous a été accordé, tous les aspects de la méthode ne sont pas présenté dans ce document. Cependant, nous nous sommes efforcé de respecter au mieux cette méthodologie. Il est a noté que nous avons conservé un spectre de fonctionnalité restreint afin d’être sur de pouvoir livrer une application.