Systèmes distribués -Système de gestion de transactions boursières

# Modèle Métier

Pour répondre au problème du sujet du TP et pour en déduire le schéma de la base de données nous avons modélisé le système en UML.

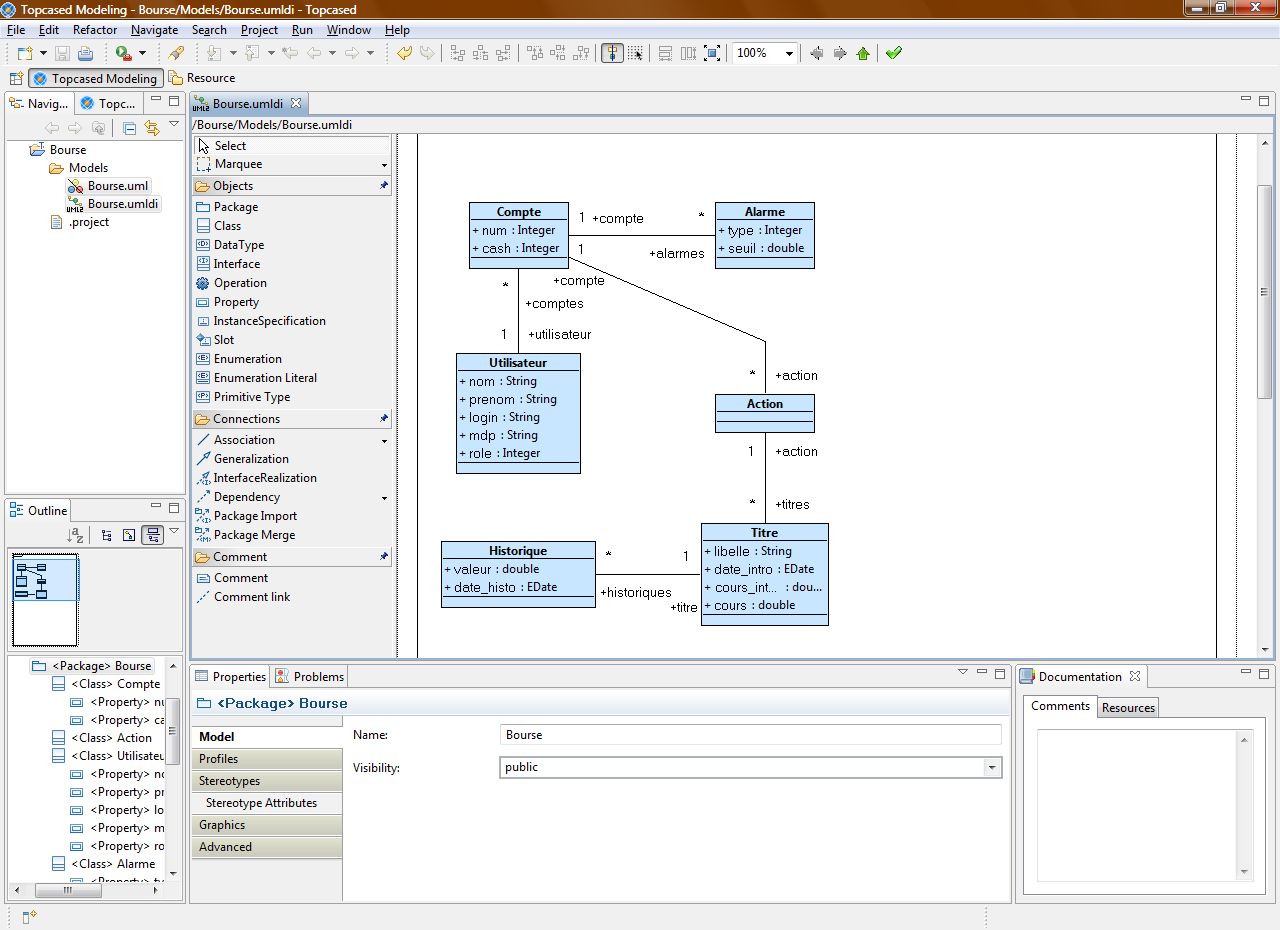


Figure 1 Modèle UML du système

Un compte possède un numéro et du cash. Celui-ci correspond à l’argent courant. Un compte est possédé par un utilisateur.

Celui-ci possède un nom, un prénom, un login d’identification, un mot de passe et un rôle. Le rôle permet de différencier un utilisateur standard d’un administrateur.

Le compte peut être surveillé par plusieurs alarmes qui se déclencheront quand le compte aura franchi un certain seuil. Le type d’alarme définit si l’alarme se déclenché quand le cash du compte est situé en dessous ou au dessus du seuil.

Le titre possède un libellé, une date et un cours d’introduction, ainsi qu’un cours courant. Le titre a aussi un historique qui lui permet de connaître les variations de son cours.

Un compte possède plusieurs actions. Ces actions correspondent en fait à des titres de la bourse.

Du modèle métier décrit en Figure 1, nous en avons déduit un modèle entité-relation décrit ci-dessous.

Figure 2 Schéma entité relationnel

# Realisation

Pour réponde aux exigences du sujet nous avons défini le modèle métier permettant de stocker les informations en base de données mais nous devons aussi modéliser le système corba en prenant en compte les spécificités de la technologie.

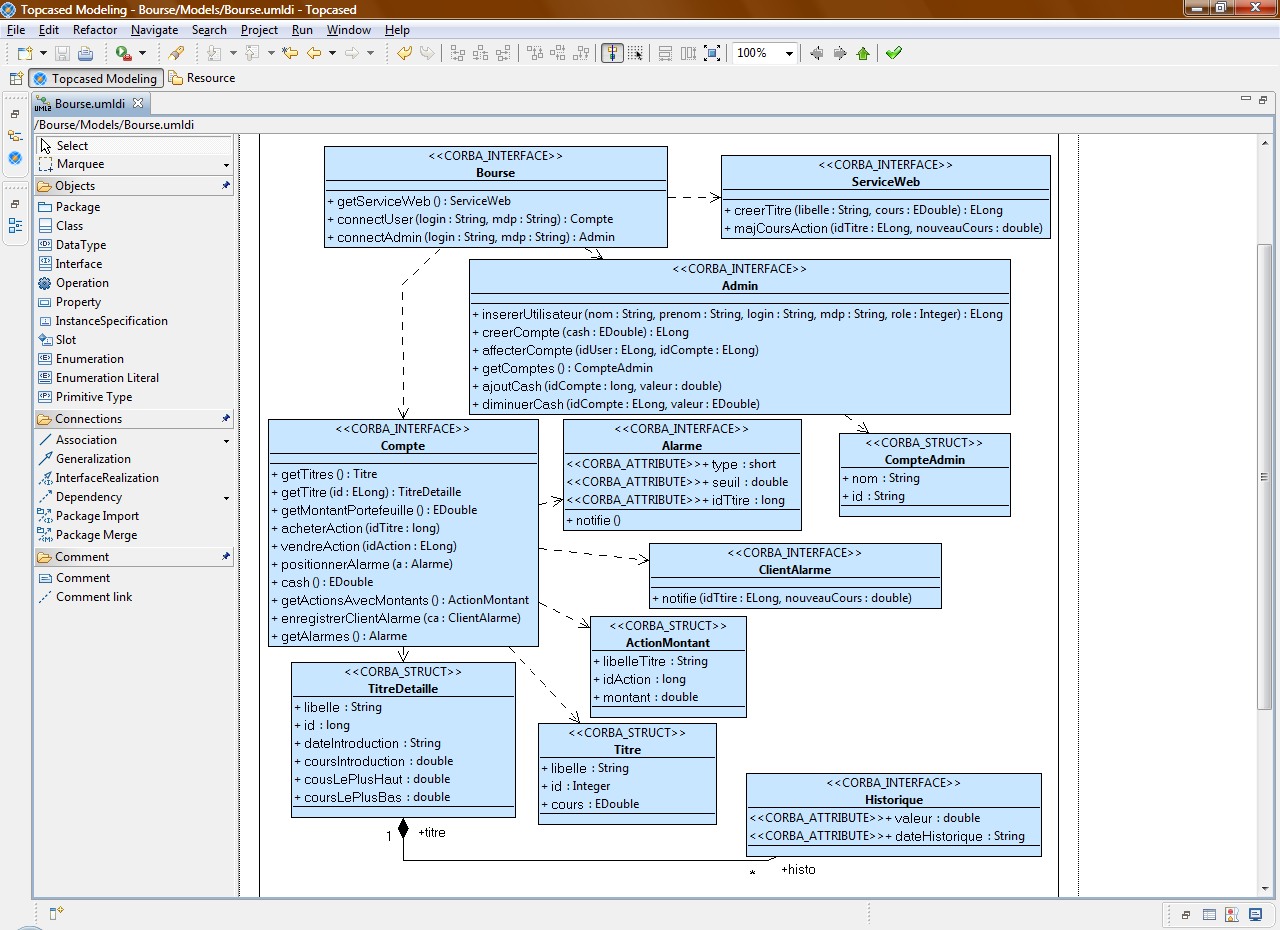


Figure 3 modèle CORBA

Ce modèle possède un élément racine qui est l’interface Bourse. Cette interface permet d’accéder à 3 autres interfaces correspondant aux trois types de services disponibles. Voyons en détail comment ces 3 services répondent aux exigences.

## ADMINISTRATEUR

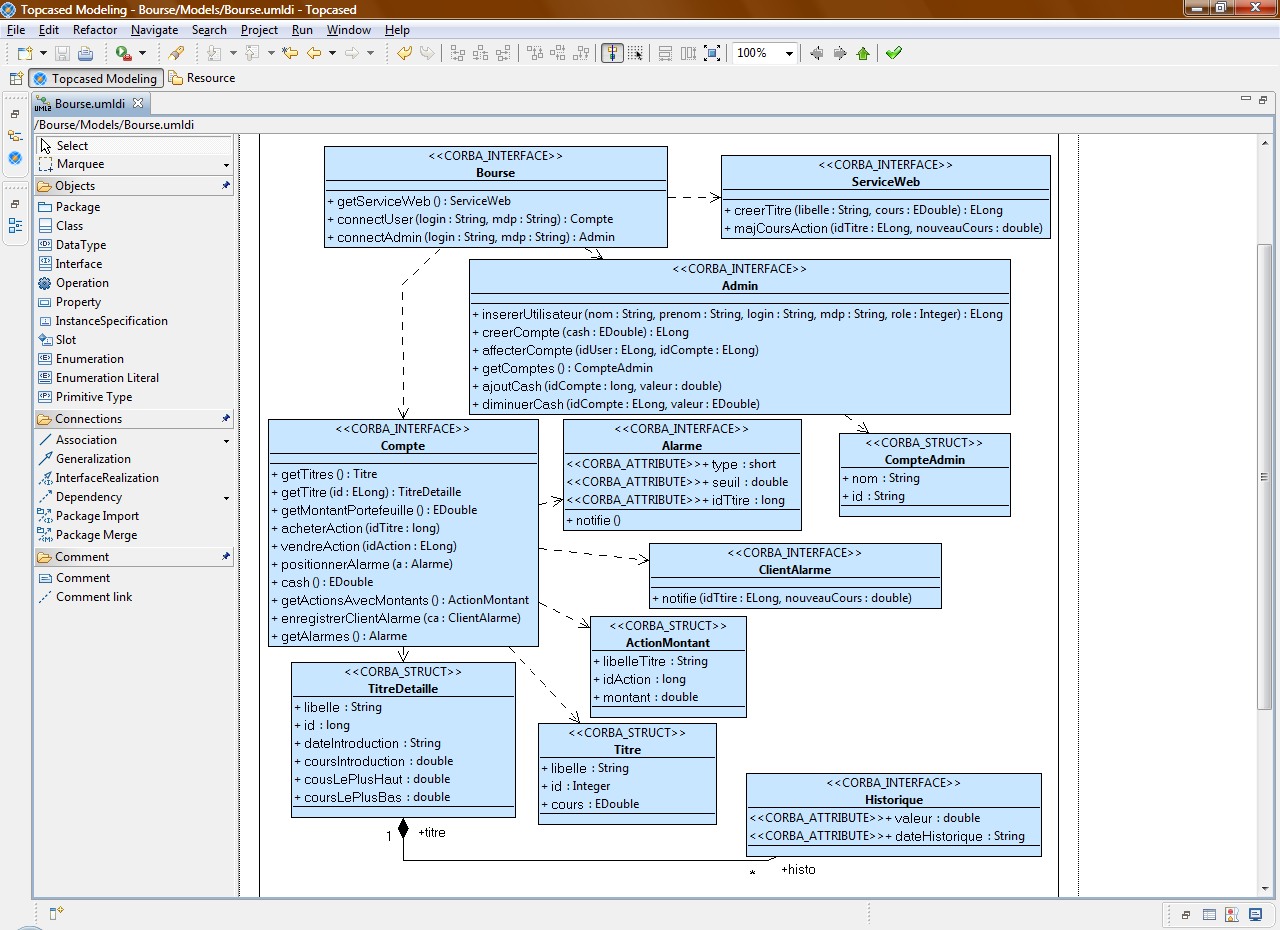


Figure 4 : focus sur la partie Admin

Pour accéder aux fonctions d’administration. Il faut accéder au service connectAdmin de l’interface Bourse. Nous avons souhaité avoir deux services bien distincts pour l’administrateur et l’utilisateur lambda, au lieu de faire un système de gestion par rôles.

### Création d’un compte et allocation d’un montant de cash initial.

La méthode créerCompte, permet de créer un compte, de récupérer son identifier et d’initialiser ce compte avec un cash de base.

Dans l’ihm ceci est représenté dans la figure ci-dessous.

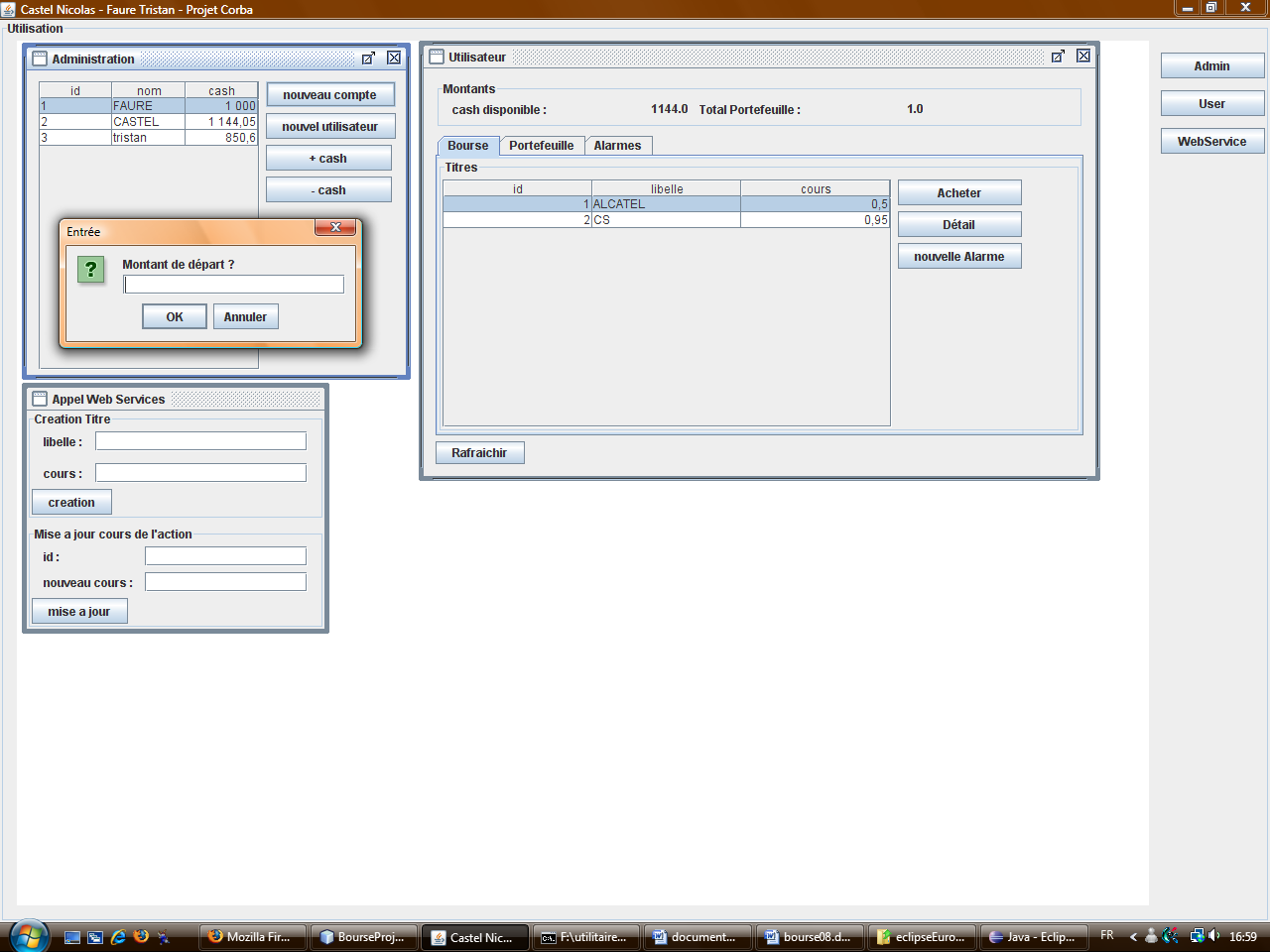


Figure 5: fenêtre de dialogue après creation d’un compte.

La méthode affecterCompte va affecter un compte d’identifiant donné à un utilisateur.

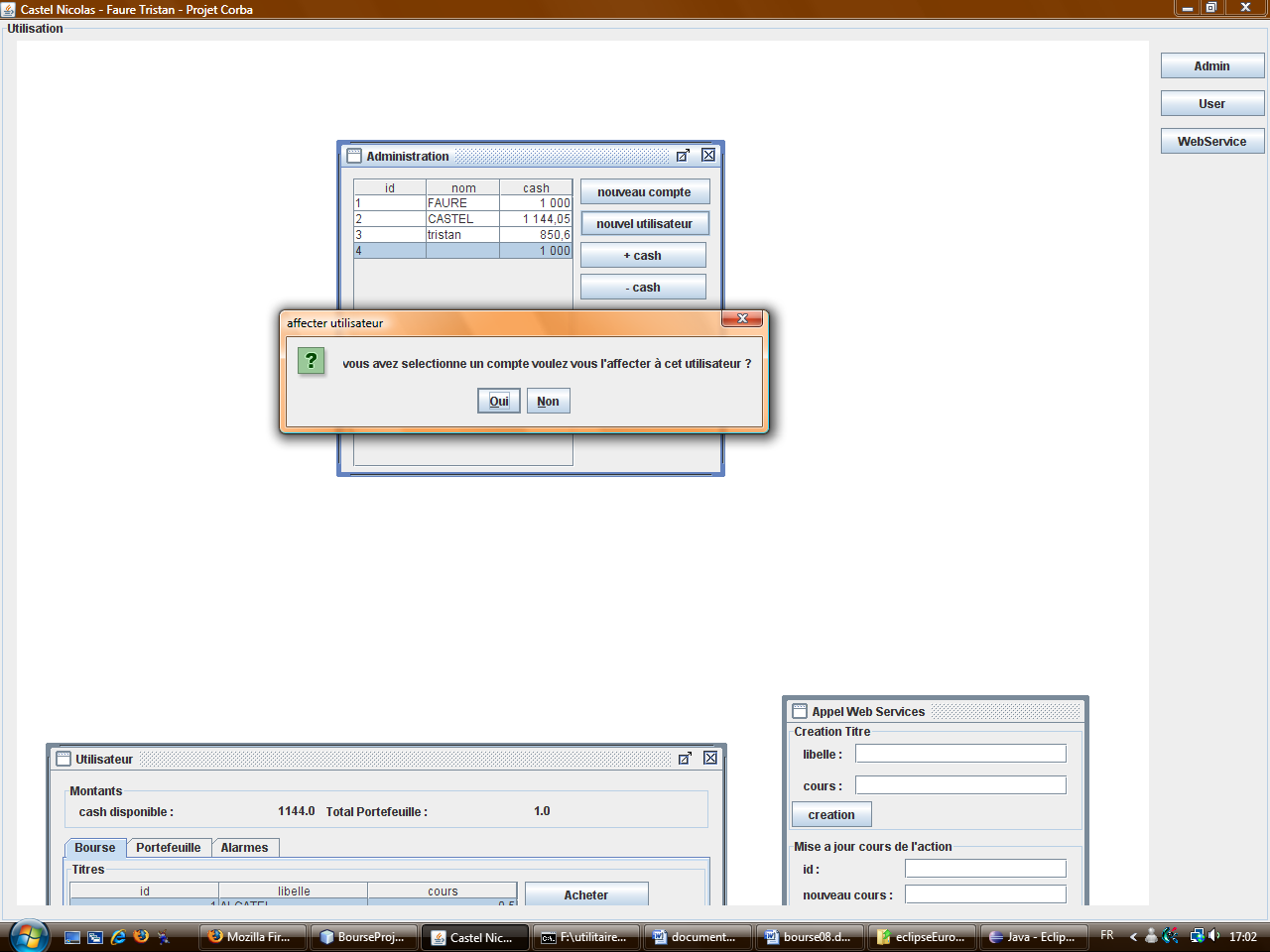


Figure 6 : dialogue d’affectation après creation d’un utilisateur

### Augmentation et diminution du cash.

Les méthodes ajoutCash et diminuerCash permet d’augmenter ou diminuer le cash d’un compte en spécifiant son identifiant.

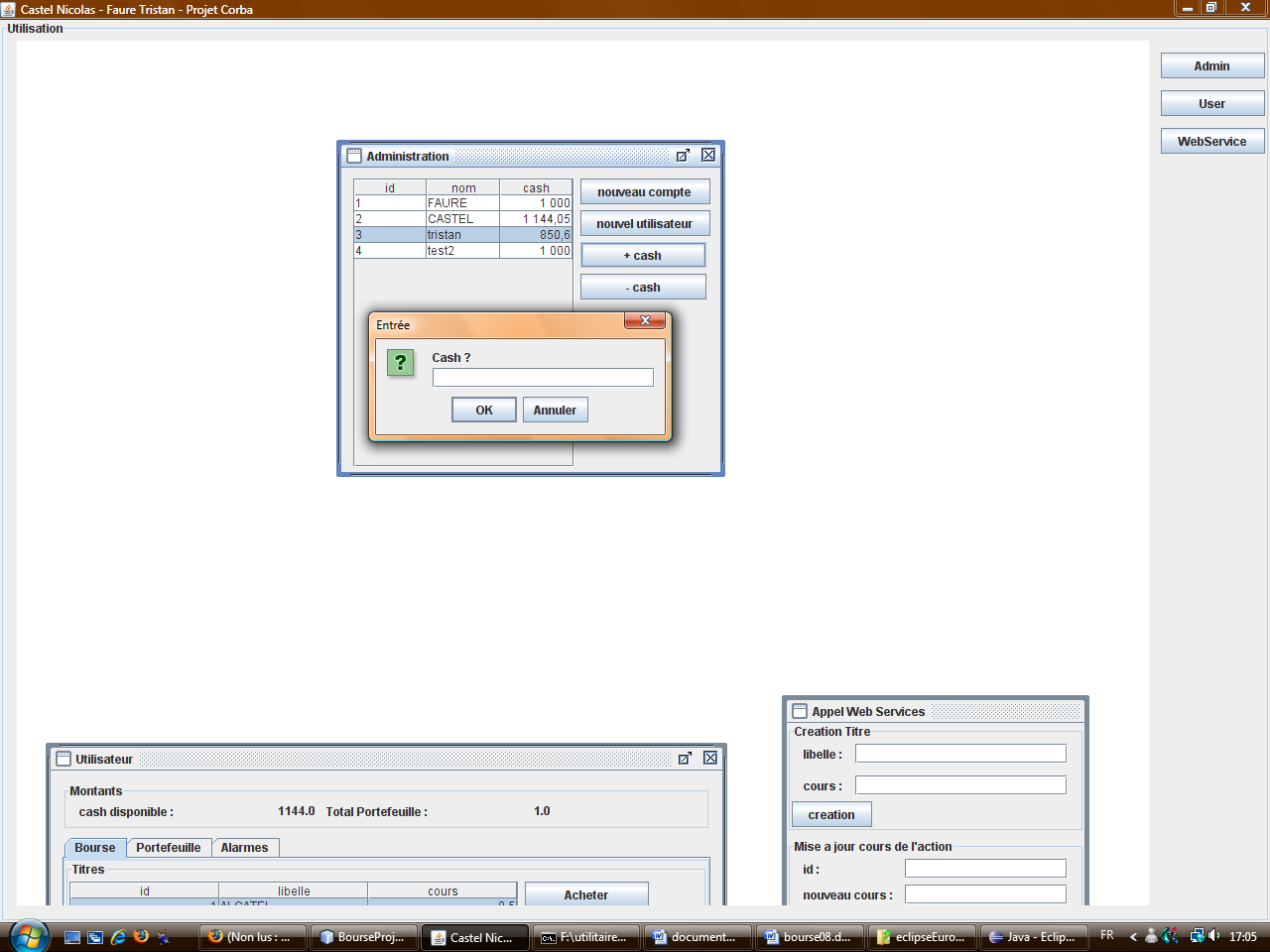
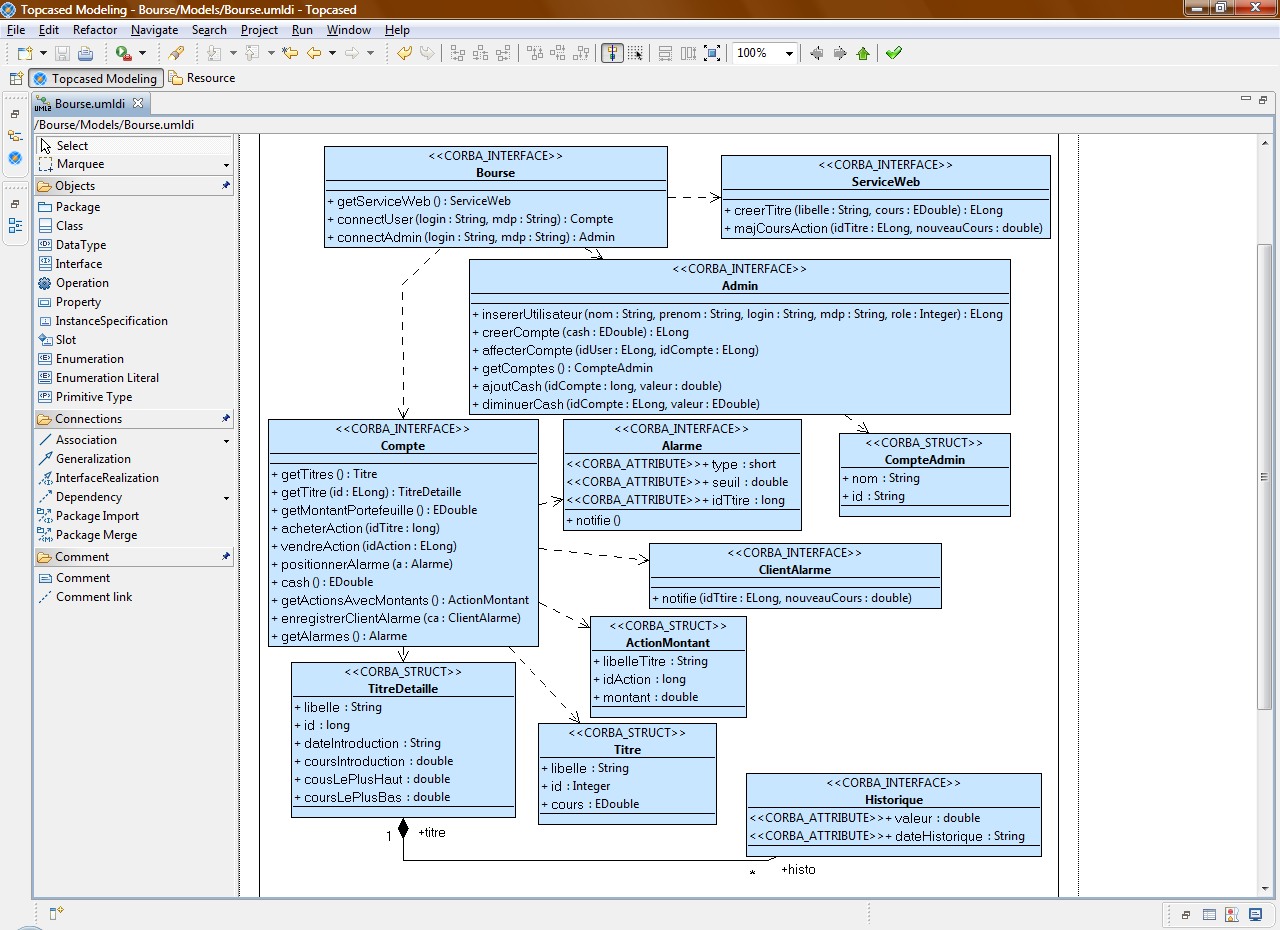


Figure 7 : dialogue pour valeur du cash a ajouter

## Services WEB



Les services web utilisent la technologie axis. Dans notre implémentation nous nous connectons au serveur CORBA nous récupérons l’interface BOURSE et faisons appel au service getServiceWeb.

### Créer un nouveau titre.

Le service creerTitre va créer en base un titre en spécifiant le libellé et le cours. Le service retourne l’identifiant du titre.

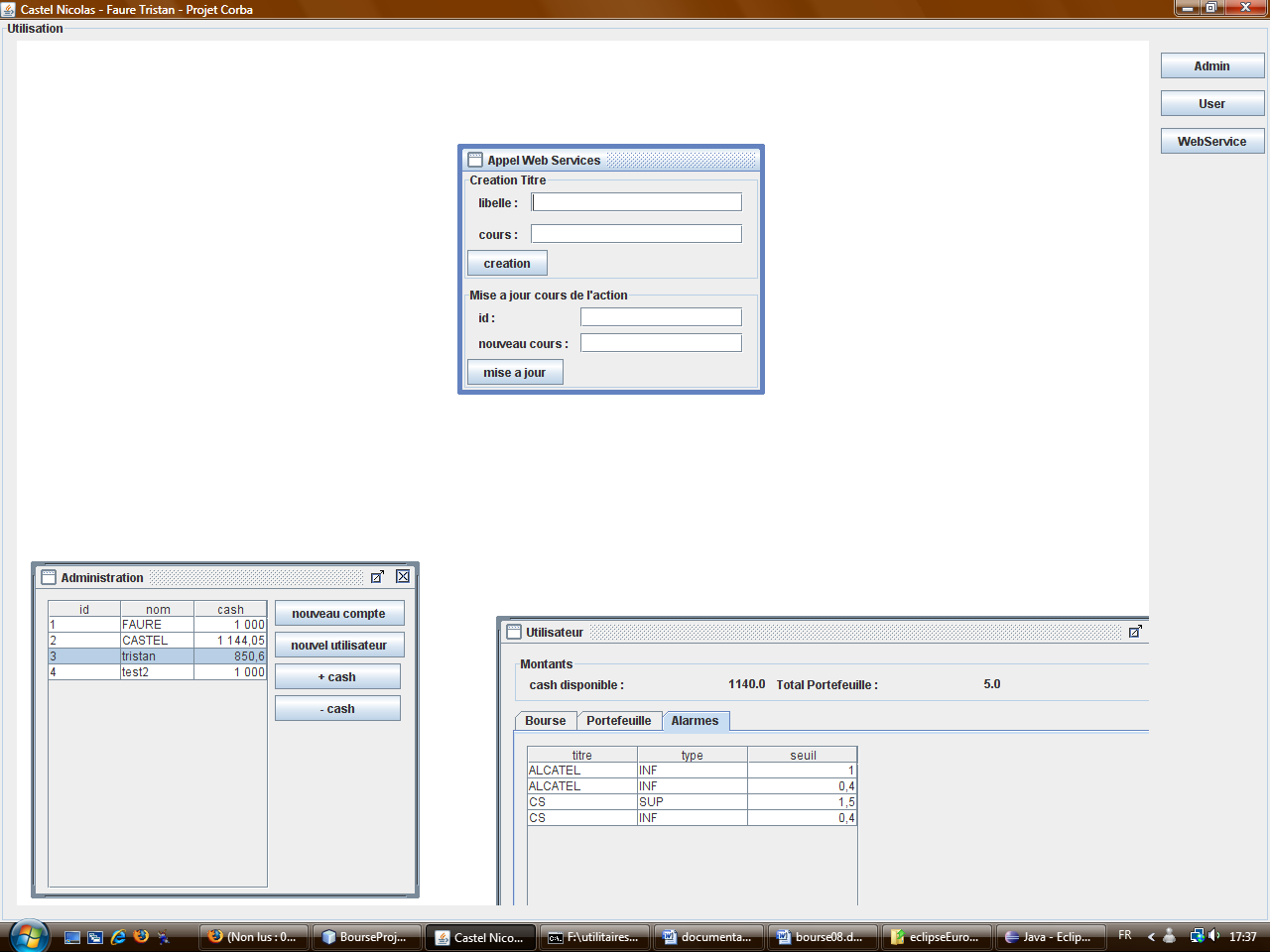


Figure 8 : creation de titre

### Mettre à jour les cours d’une action.

Le service majCoursAction va à partir d’un identifiant affecter un nouveau cours. Lorsque CORBA reçoit cet appel il va à la fois modifier la valeur du cours mais aussi rajouter une entrée dans l’historique et vérifier les alarmes (voir titres plus bas).

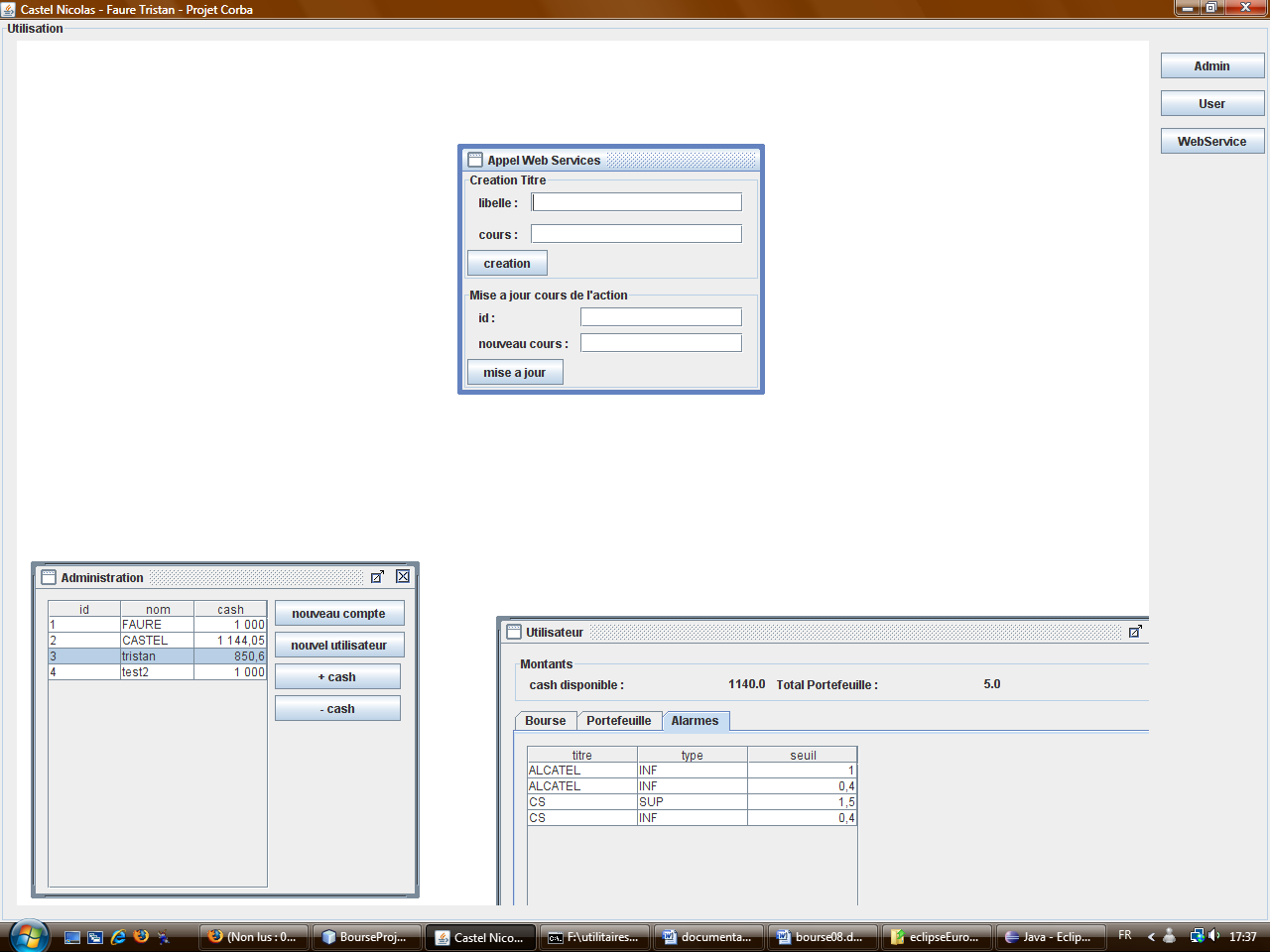


Figure 9 : mise à jour du cour d’une action

## Compte

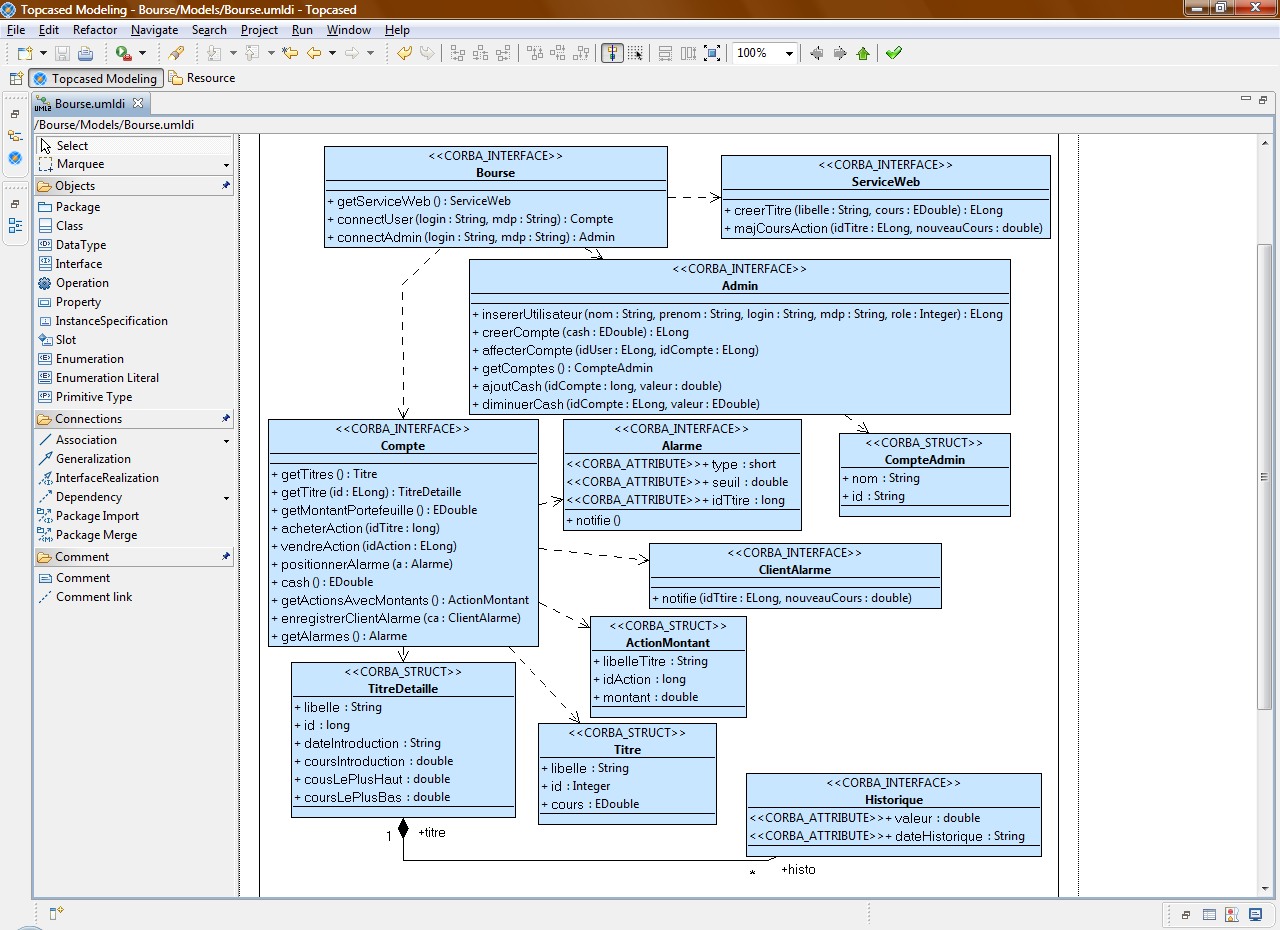


Figure 10 : focus sur la partie compte

### Accéder à un compte existant à partir d’un identificateur et d’un mot de passe

Depuis l’interface Bourse lorsqu’on appelle la méthode distante connectUser en fournissant le login et le mot de passe. On accède au compte associé.

### Accéder à la liste des « titres » (un titre est identifiée par un code et possède un libellé).

La méthode getTitres retourne une collection de Titres. Les éléments « Titre » retounés ne possèdent que le libellé, l’identifiant et le cours du titre. Ceci permet un faible trafic réseau pour avoir la liste des titres.

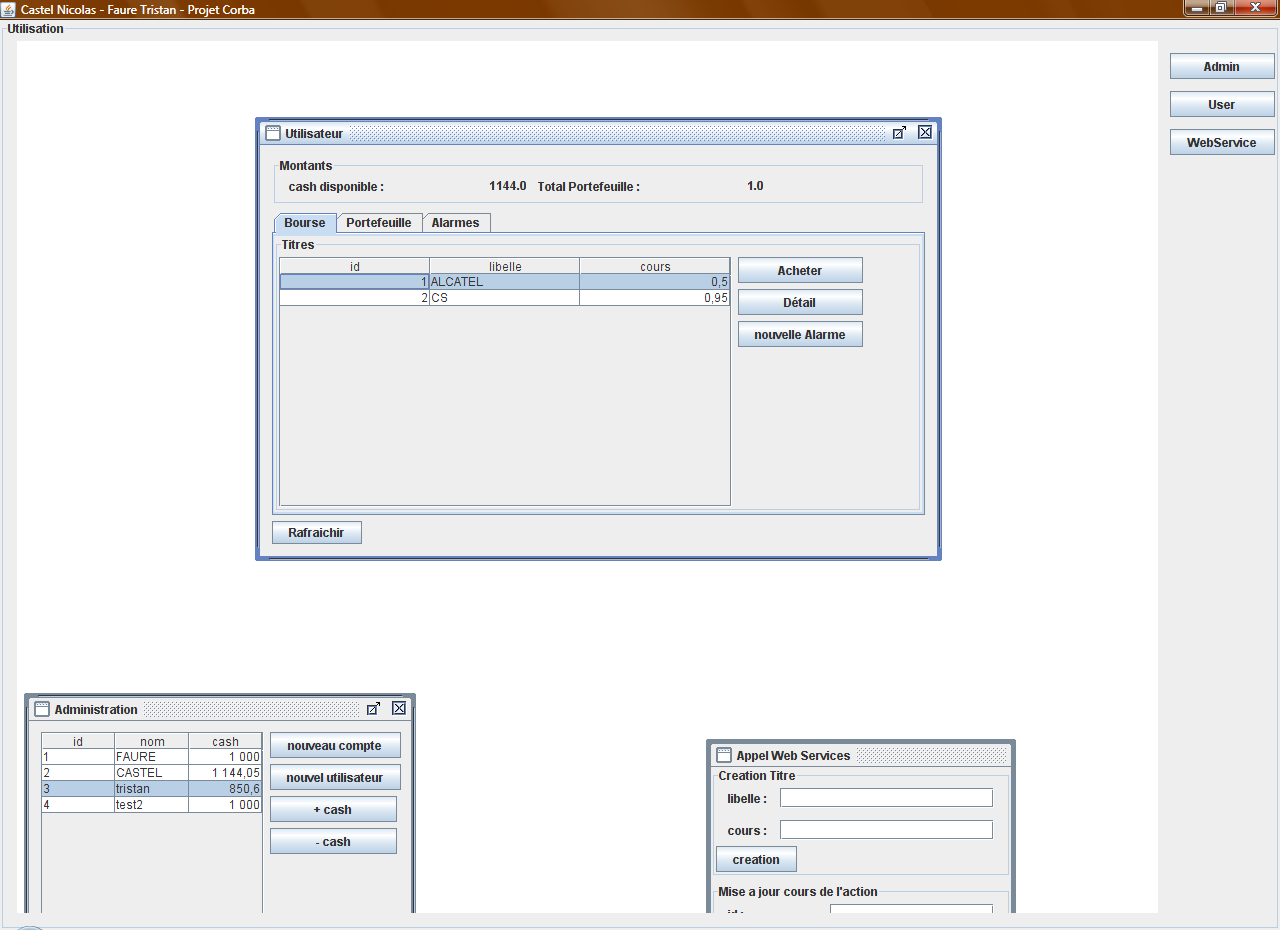


Figure 11 : dans l’ihm la liste des titres est visible sur un onglet

### Accéder aux informations détaillées d’une action donnée (sa date d’introduction en bourse, son cours d’introduction, son cours le plus haut, son cours le plus bas, un historique des cours du jour).

La méthode de l’interface Corba getTitre retourne un Titre détaillé. Cette structure permet d’accéder à toutes les informations spécifiées par l’exigence.

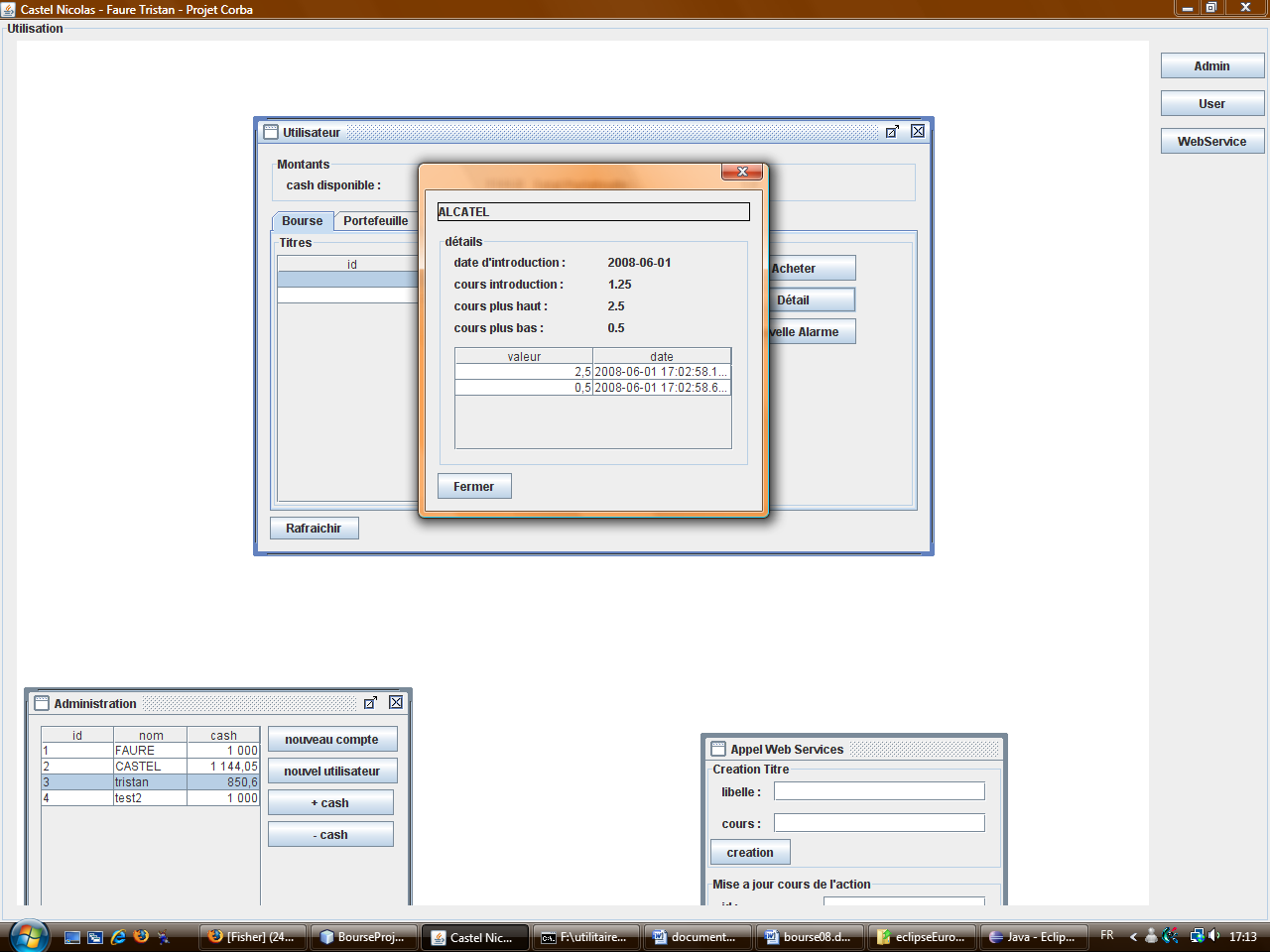


Figure 12 : informations détaillées d’un titre

### Accéder à une valorisation de son portefeuille (montant par actions et montant total).

Le service getMontantPortefeuille retourne le montant total du portefeuille d’actions du compte.

Le service getActionsAvecMontant fournit la liste des actions possédées par le compte et leurs montants.

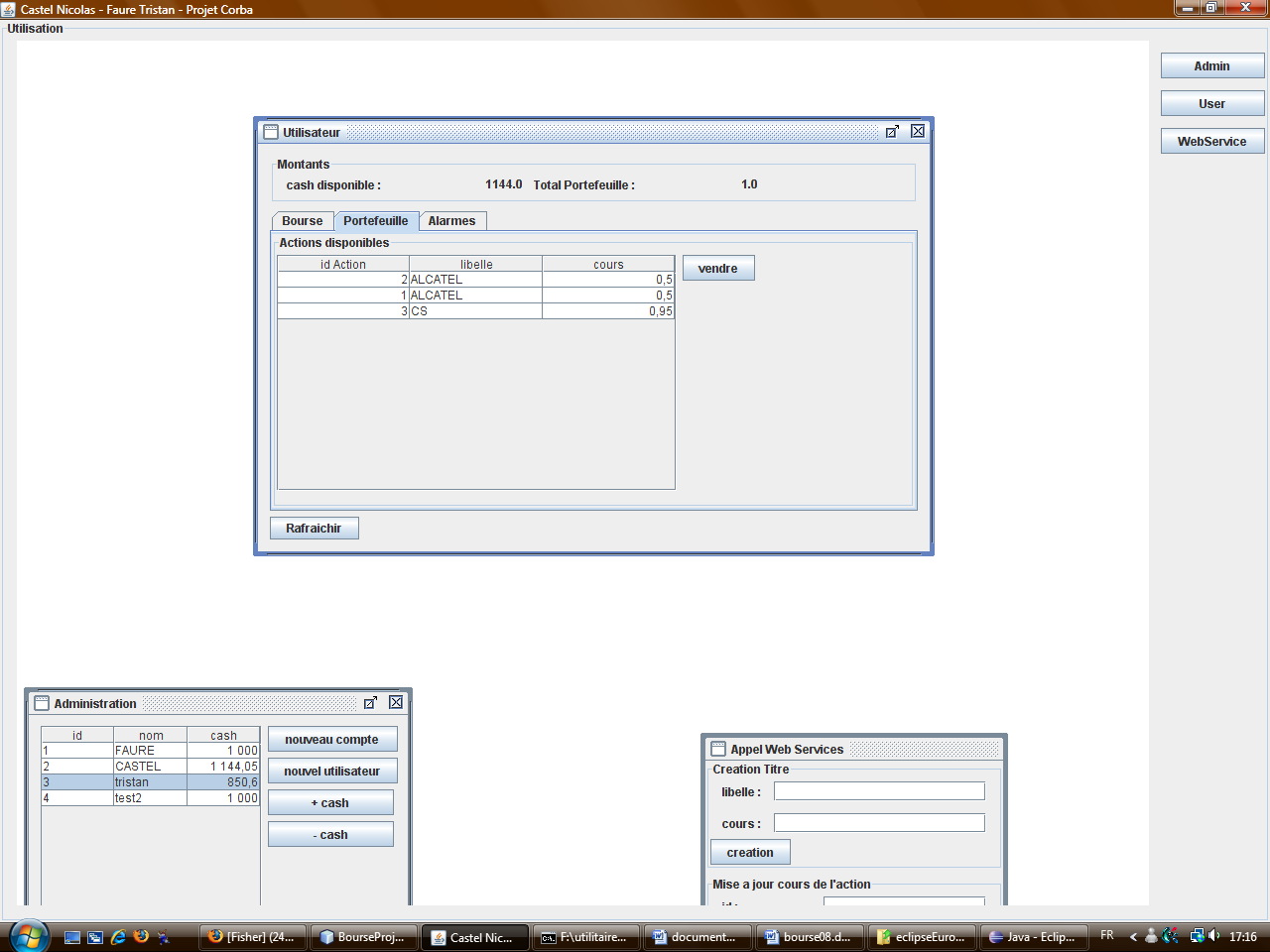


Figure 13 : total du portefeuile et liste des actions avec leurs montants.

### Acheter/vendre des actions

La méthode acheterAction prend en paramètre l’identifiant du titre à acheter. Le service ne lève pas d’erreurs si le compte n’a pas assez de cash c’est à l’application cliente de savoir si le client à le droit d’être à découvert ou non.

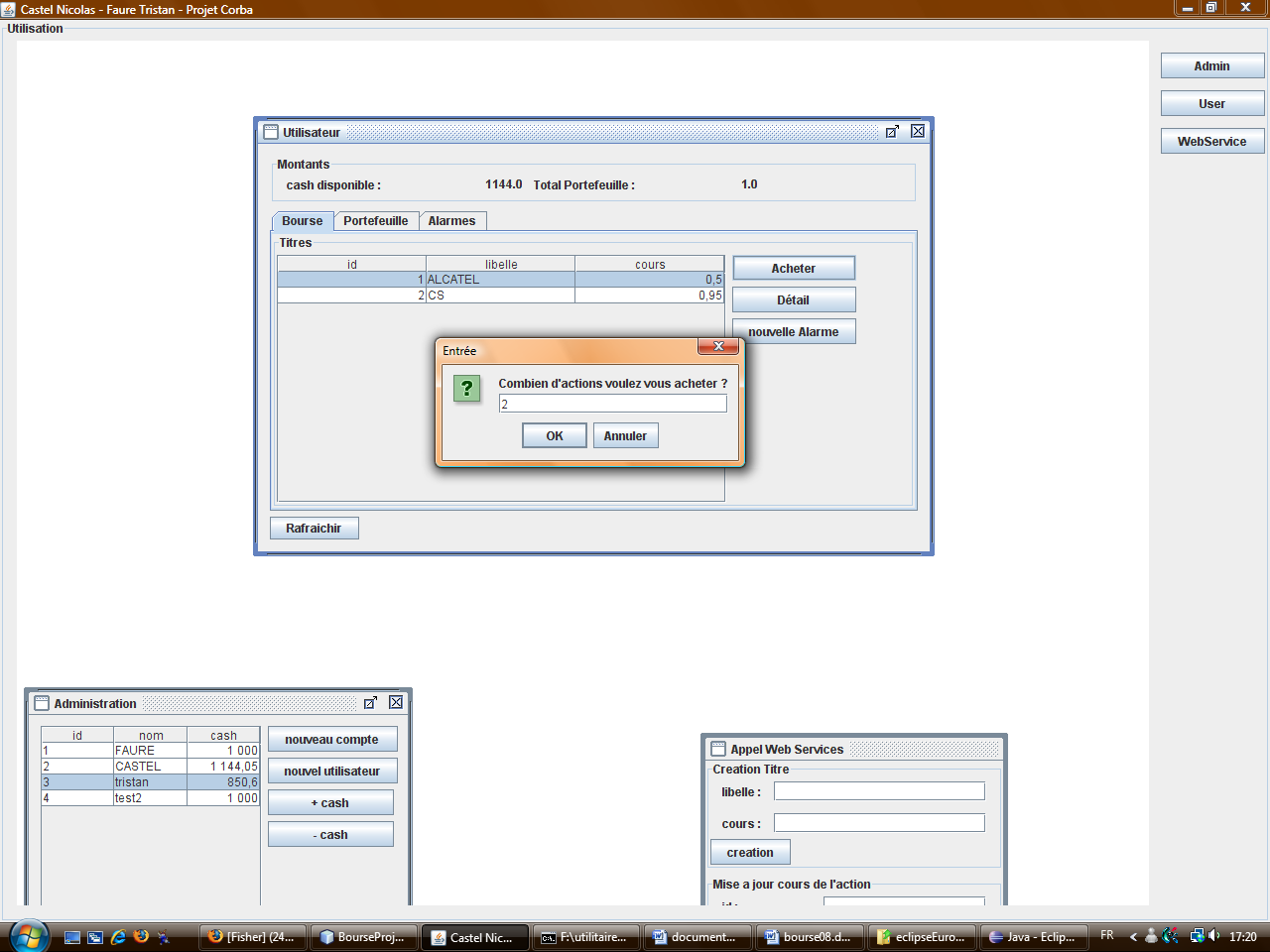


Figure 14 : achat de titres

La méthode vendreAction prend en paramètre l’identifiant de l’action à vendre.

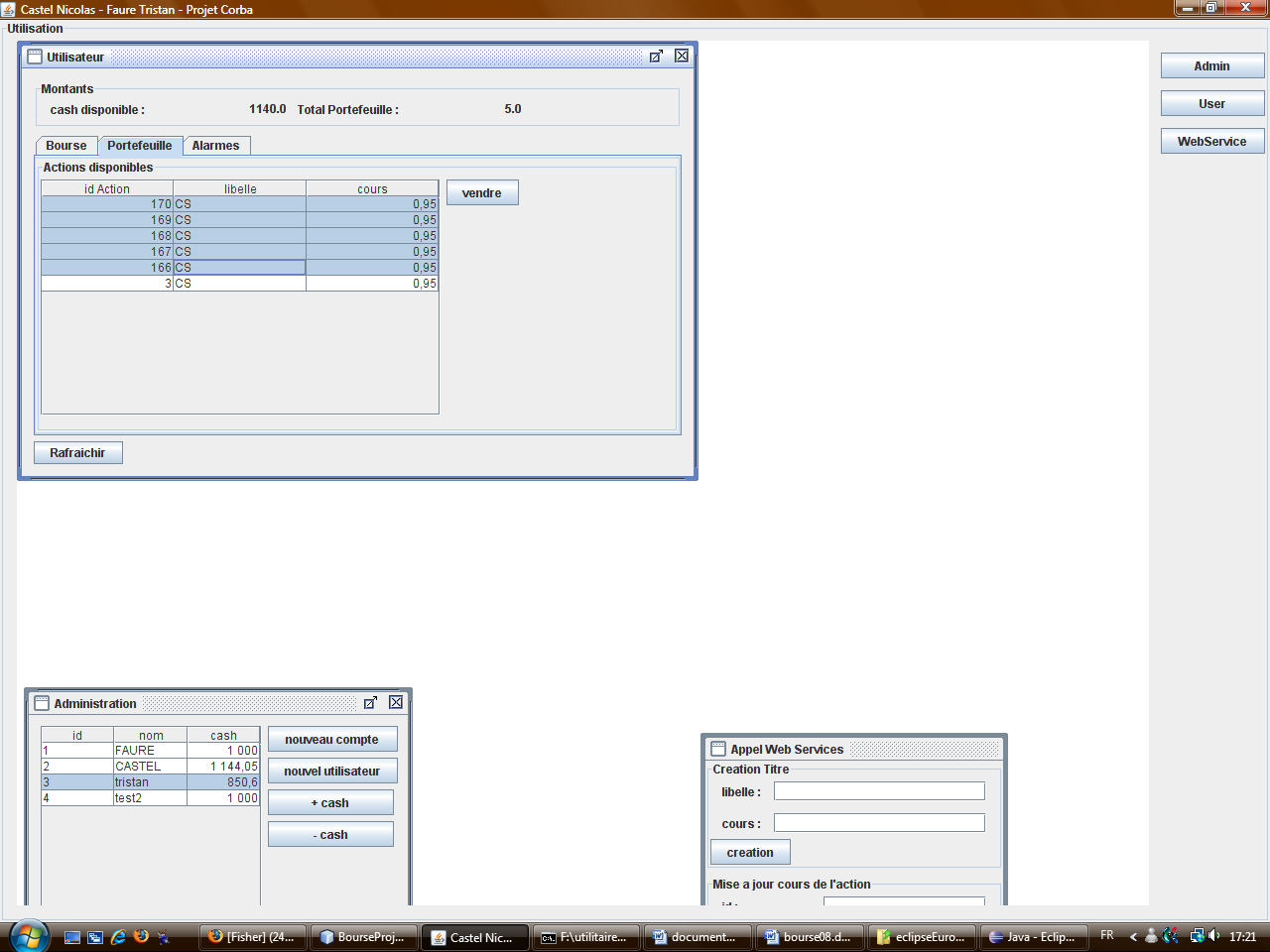


Figure 15 : vente de titre

Dans l’IHM, nous offrons la possibilité d’acheter et vendre plusieurs actions d’un seul clic. Pour ne pas bloquer l’interface par la succession d’appels synchrones au serveur CORBA, nous embarquons ces appels dans un thread.

### Accéder au solde de cash disponible.

La méthode cash retourne le cash disponible.

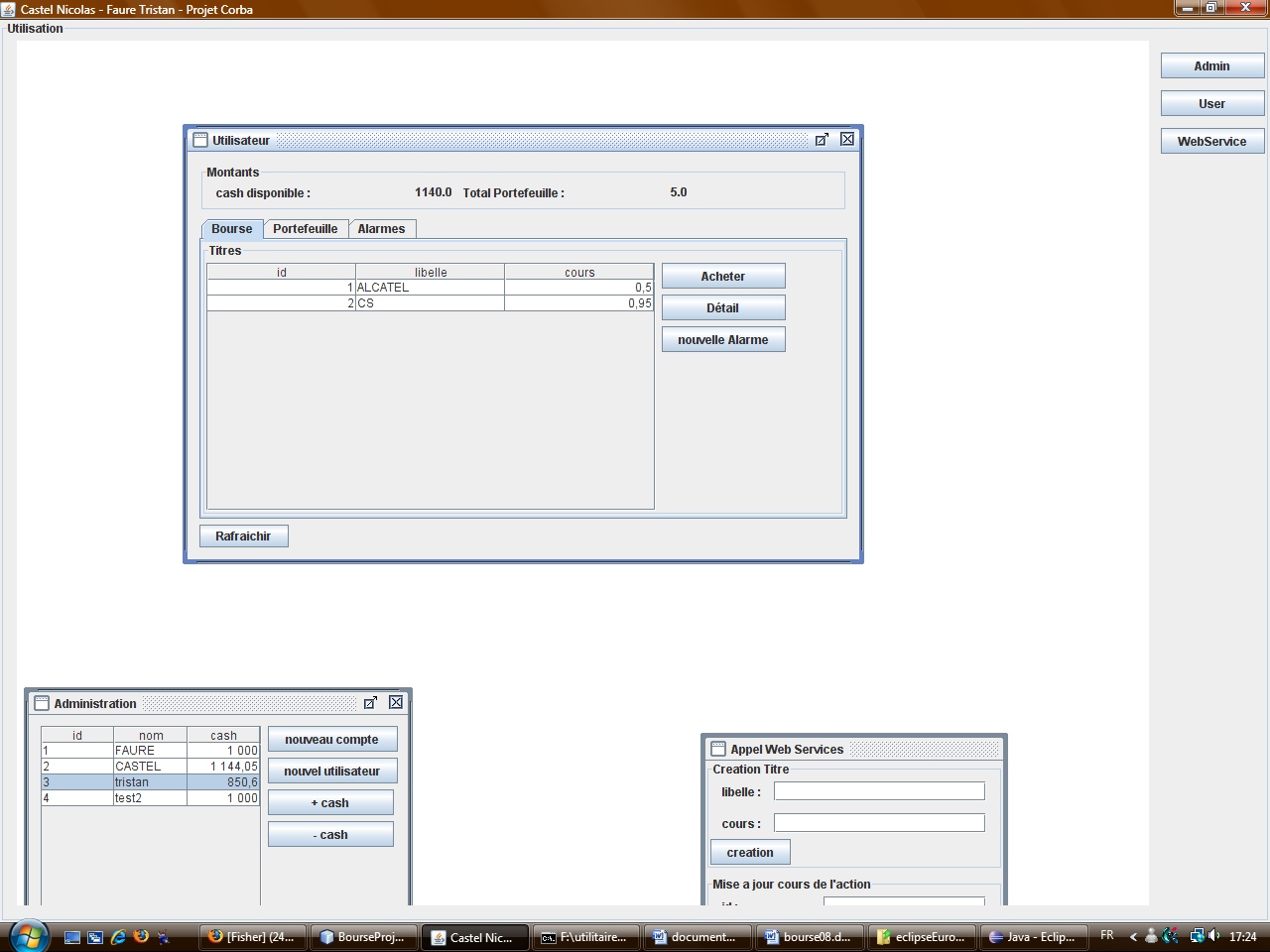


Figure 16 : cash disponible

### Pouvoir positionner des alarmes et être averti quand une alarme se déclenche. Une alarme est du type « avertis moi si le cours devient supérieur à un seuil S » ou « avertis moi si le cours devient inférieur à un seuil S ». Lorsqu’un seuil est atteint, le serveur envoie une notification à l’abonné lui donnant : l’action concernée, le rappel du type de l’alarme est du seuil, le cours réel.

L’interface clientAlarme permet de disposer d’un objet qui peut être notifié par le serveur lorsqu’il le faut. Dans notre cas lors d’une mise à jour d’un cours d’action. Une méthode du serveur va vérifier tous les clientAlarmes enregistrés via la méthode enregistrerClientsAlarmes si une alarme va être déclenchée via cette modification de cours.

Si le serveur essaie de notifier un clientAlarme et qu’il n’est plus joignable (déconnexion …) alors le serveur va effacer ce client alarme de son registre pour ne pas le rappeler au prochain déclenchement d’alarmes.

Il faut noter que toutes les alarmes sont enregistrées en base de données donc lorsque les clients se connecte toutes ses alarmes sont automatiquement chargées sur le serveur.

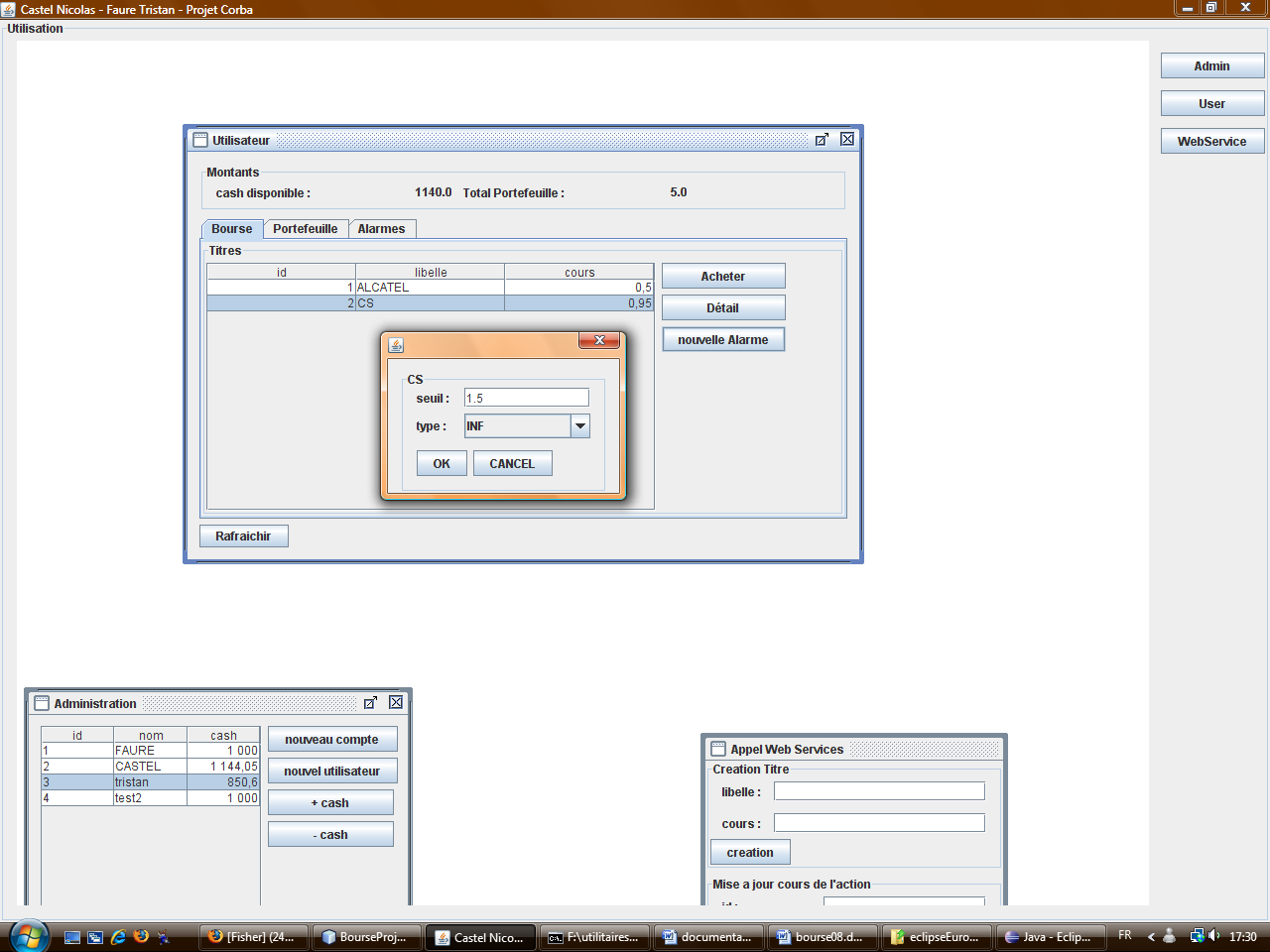


Figure : definition d’une alarme

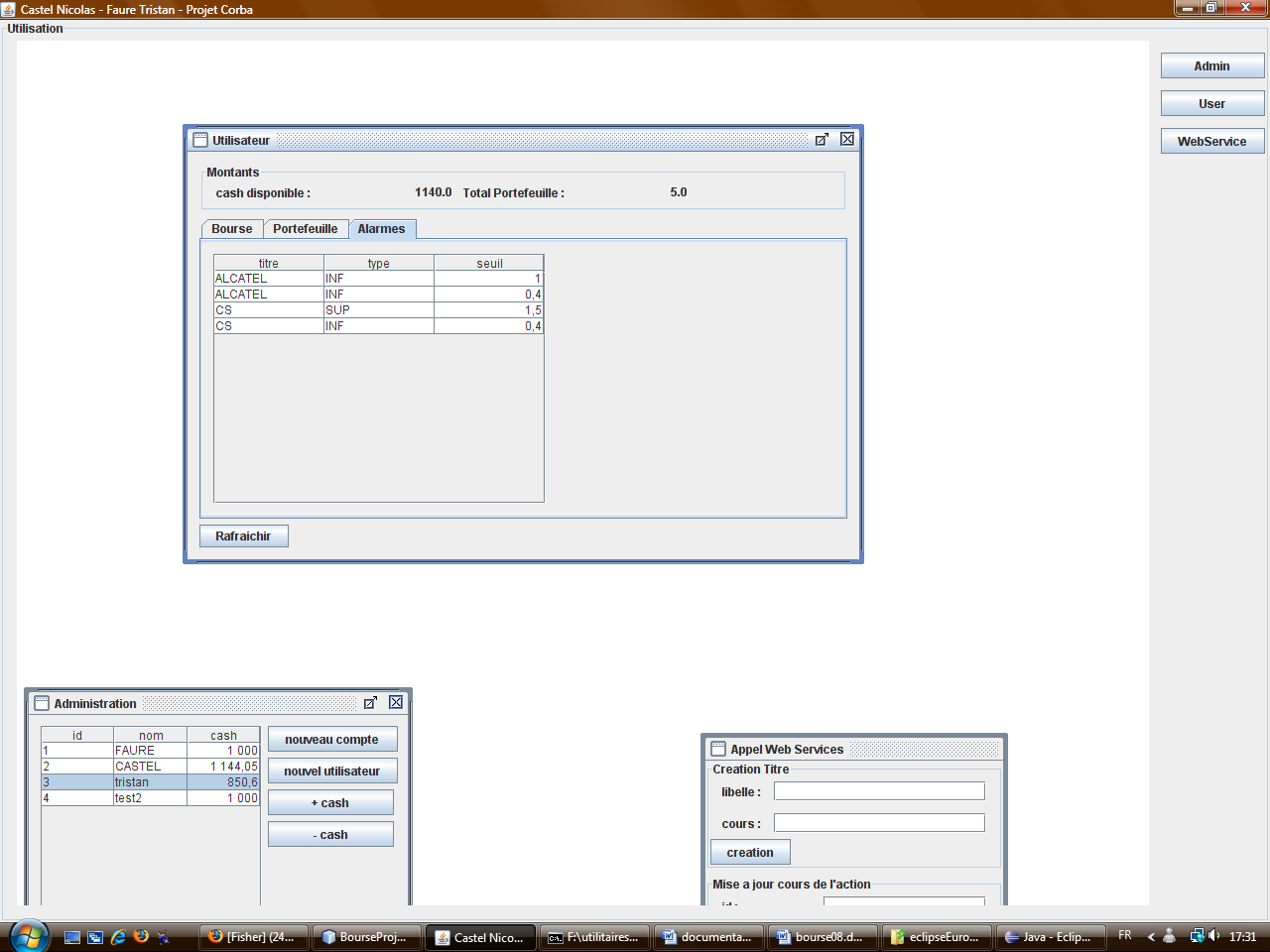


Figure 18 : liste des alarmes d’un compte

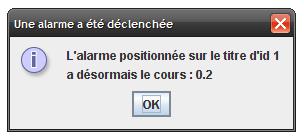


Figure 19 : notification d’une alarme

# CONCLUSION

Le projet CORBA fut intéressant sur plusieurs aspects.

Le premier est évidement la découverte d’une nouvelle technologie. Ce qui nous permettra de nous préparer si nous devons à l’avenir travailler sur une architecture utilisant CORBA. Cette norme est rapidement utilisable en JAVA en tout cas et à deux trois contraintes qu’il faut connaître (éviter d’utiliser de null par exemple) nous n’avons pas vu de réels défauts à la plateforme.

Le second point intéressant et qu’il nous a permit de retravailler sur des problématiques d’outil client serveur. Ca nous permet de voir que même si les technologies sont différentes les problématiques restent les mêmes. En effet se poser des questions sur le stockage en base de données ou la gestion des threads pour éviter l’encombrement du serveur sont autant de points qu’il est important d’assimiler avant de partir sur des projets plus « opérationnels ».