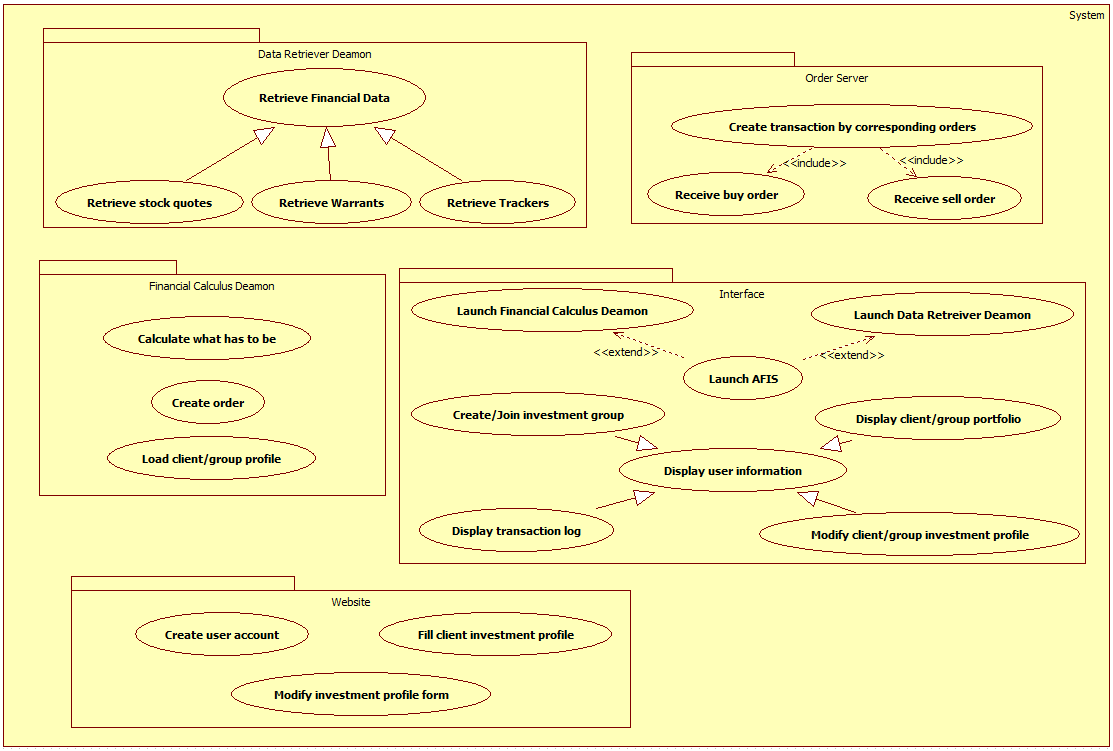
# I Description de l’architecture globale de AFIS

## AFIS: Automated Financial Investment System

Notre système d’investissement est divisé en 5 grandes parties :

* Data Retriever Deamon : C’est la partie chargée de récupérer les cours de bourses des entreprises du côté sur le compartiment A de Eurolist sur le marché de Paris. Le DRD devra également obtenir la liste de tous les trackers et warrants côtés sur le marché de Paris.
* Order server : Ce serveur est chargé d’attendre la réception d’ordre d’achat ou de vente de titres financiers du système AFIS. Il a également pour rôle d’assurer la correspondance entre les différents ordres de vente et d’achat.
* Financial Calculus Deamon : Le FCD est toute la partie mathématique du projet. Il réceptionne les données financières du DRD et à l’aide des données antérieurs sur la base de données, ainsi que le profil client, devra prendre la décision d’acheter ou vendre X titres financiers. Cette partie, comme vous le verrez est assurément la plus complexe que nous gardons donc pour la fin du projet.
* L’interface : Via cette interface l’utilisateur pourra voir le contenu de son portefeuille financier, modifier son profil, créer un groupe d’investissement ou en rejoindre un préexistant, consulter l’historique de toute les transactions le concernant lui ou ses groupes d’investissements. Il lui sera également possible de lancer ou arrêter le DRD ainsi que le FCD via un bouton commun.
* Le site internet : C’est la première étape pour toutes personnes souhaitant utiliser AFIS, il faut créer un compte et remplir son profil d’investissement sur le site, ou sur l’interface.

## Diagramme de cas d’utilisation générale de AFIS

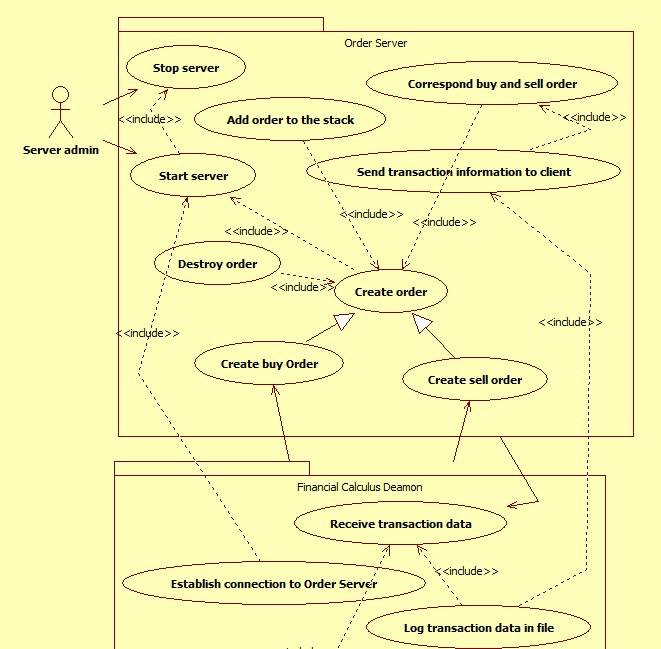


# II Spécifications des composants de AFIS

## Order Server

Afin de fonctionner correctement le serveur d’ordre va fonctionner à l’aide de threads. Il s’agit d’un processus principal qui va attendre les messages des clients AFIS. Lorsqu’un message est détecté, un thread créer pour s’en charger et le serveur d’ordre reprend son attente habituelle.

Ce thread aura pour rôle de déchiffrer le message reçu, vérifier que les propriétés d’un ordre sont respectées et enfin ajouter l’ordre dans la « pile ». Un autre processus aura pour rôle d’analyser la pile d’ordres d’achat/vente afin de rechercher une correspondance entre un achat et une vente. Si une correspondance est détectée, un thread est créé. Ce thread aura pour rôle d’envoyer aux intéressés de cette transaction les données relatives à cette dernière et d’ensuite détruire les deux ordres de la pile.



On voit ici que le FCD dépend du Order Server puisque c’est le FCD qui va se charger de créer un ordre et de suivre le cycle de vie de ce dernier.

## Financial Calculus Deamon

Le FCD se chargera de déterminer si un ordre d’achat ou de vente mérite d’être passé et si c’est le cas, il passera cet ordre.

Le FCD devra lire les données financières écrites dans la mémoire partagé le Data Retriever Deamon.

Lorsque ces données sont récupérées, le FCD rentre dans une série d’algorithmes, pas encore déterminés, et écrit les résultats de calculs dans la mémoire et la base de données. Cette opération se fait infiniment.

Un autre processus se charge de charger le profil du groupe ou de l’utilisateur. Il analysera les résultats des nombreux calculs effectués par l’autre processus et déterminera si un ordre doit être passé ou non. Les critères rentrant en jeu sont la rentabilité future estimée de l’opération, le taux de risque ne dépassant le taux toléré par le client etc.

Si un ordre doit être passé un thread est créé afin de charger de la partie administrative. Il initialise une connexion au serveur d’ordre et gardera cette connexion tant que le serveur d’ordre ne l’aura pas averti de la validité de son ordre ainsi que de son état.

Dans le cas d’un ordre ayant vocation à être traiter dans plusieurs heures/jours la connexion sera fermé et ce sera le rôle du serveur d’ordre d’avertir le FCD des états changeants de son ordre.

Vous pouvez voir sur le diagramme que l’opération « calculate what has to be » ne semble pas très précise. En effet, les algorithmes à employer, les calculs à effectuer ne sont pas encore déterminés.

Nous sommes actuellement en cours de recherche sur ce point, mais il est certain que nous ne pourrons pas créer nous même un modèle financier, de par la complexité de cette action.

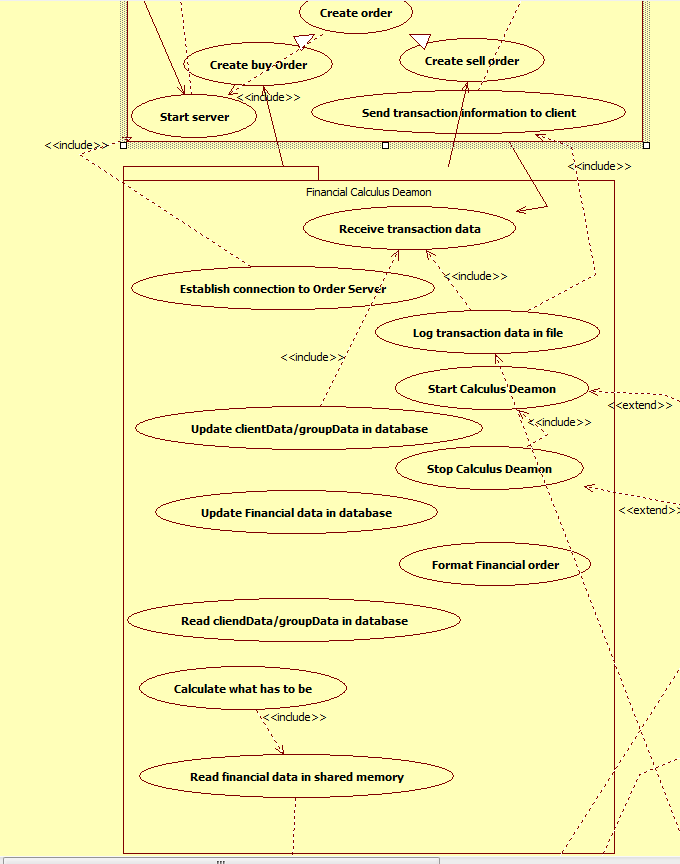
La stratégie financière sera vraisemblablement une suite d’utilisation de modèle financier et la décision de création d’ordre sera prise en fonction de cet ensemble de calculs puis filtrer en fonction du profil utilisateur/groupe.

Dans le cadre du FCD, le modèle Black-Scholes est déjà implémenté, c’est un modèle d’évaluation de valeur d’une option financière Européenne qui sera certainement appliqué aux trackers et warrants du marché de Paris puisque aucune option financière n’est côté sur ce marché.

Les greeks ont également en grande partie été implémentées, ce sont des critères de qualification de derivatives permettant d’évaluer certaines propriétés intéressantes de ces derniers.

La suite des implémentations sera faite à l’aide des livres : *Finance des marchés, techniques quantitatives et applications pratiques* et de : *Advanced Quantitative Finance with C++*.

Nous avons également téléchargés un grand nombre de PDF spécialisés sur les mathématiques financières et la finance classique dans le cas où ces deux livres ne suffiraient pas.

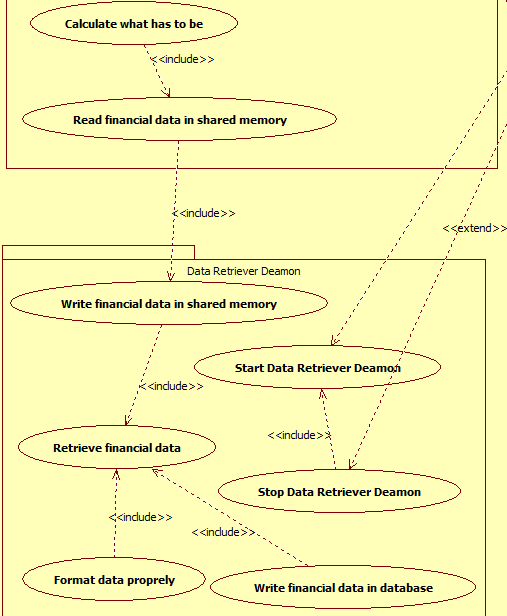


## Data Retriever Deamon

Cette partie du système est à ce jour très avancé et sur le pas de l’achèvement. Nous utilisons le site internet l’echo.be afin d’obtenir les données financières qui nous intéressent. Les flux de données sont interceptés, formatés, stockés dans la mémoire partagé, entre DRD et FCD, puis stockés dans la base de données.

La partie manquante à cette partie du projet est l’interception d’ordres financiers réels afin de pouvoir faire correspondre nos faux ordres d’achat/vente avec de vrai ordres et légitimiser notre application.

Le futur semble sombre sur cette partie du projet, il sera peut être nécessaire de continuer sans temporairement et, lorsque l’échéance sera proche, se repencher sur la question.



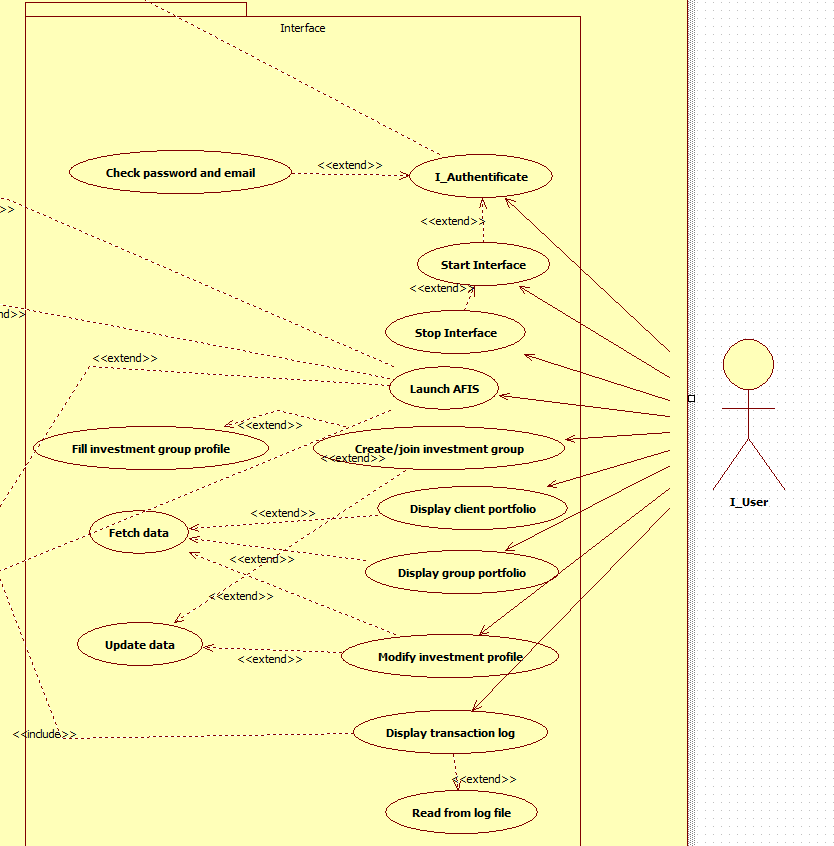
## Interface utilisateur

L’étape graphique de la connexion et du menu a été réalisé dans son ensemble, il reste les écrans de portefeuille client/groupe, l’historique de transaction et la modification du profil utilisateur.

Nous estimons que l’interface sera finit avant la fin du mois de janvier. Le rôle de l’interface est de simplement montrer à l’utilisateur ce qui compose son portefeuille, ainsi que gérer ses groupes d’investissements.

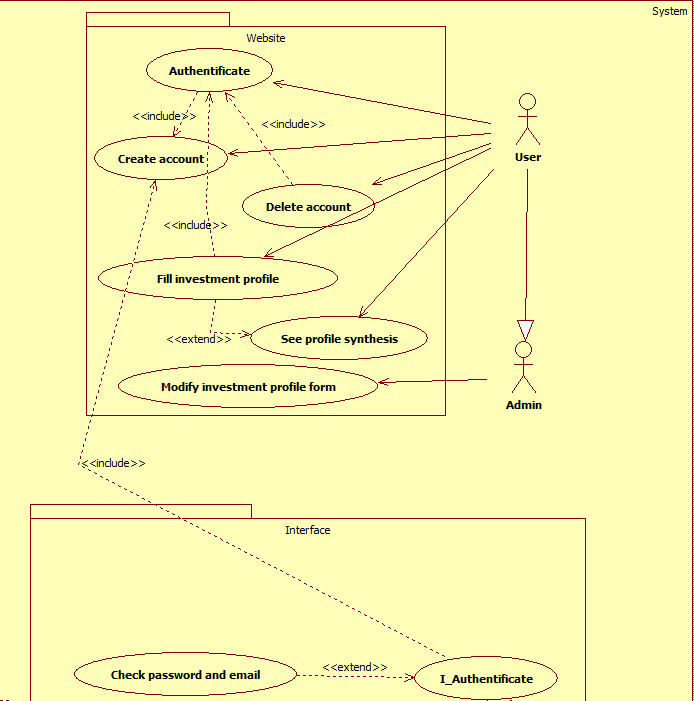
Il faut prouver à l’utilisateur que son argent n’a pas été volé, d’où l’écran d’historique de transaction.

Il serait possible de montrer à l’utilisateur des graphs, des séries de calculs, de prévision, etc, mais puisque le principe même est de gérer les fonds d’une personne sans qu’elle besoin d’y consacré du temps ou du quotient intellectuel, il est nécessaire de rester aussi simple que possible.Les seules informations affichées sont celles dont on ne peut priver l’utilisateur pour avoir sa confiance.



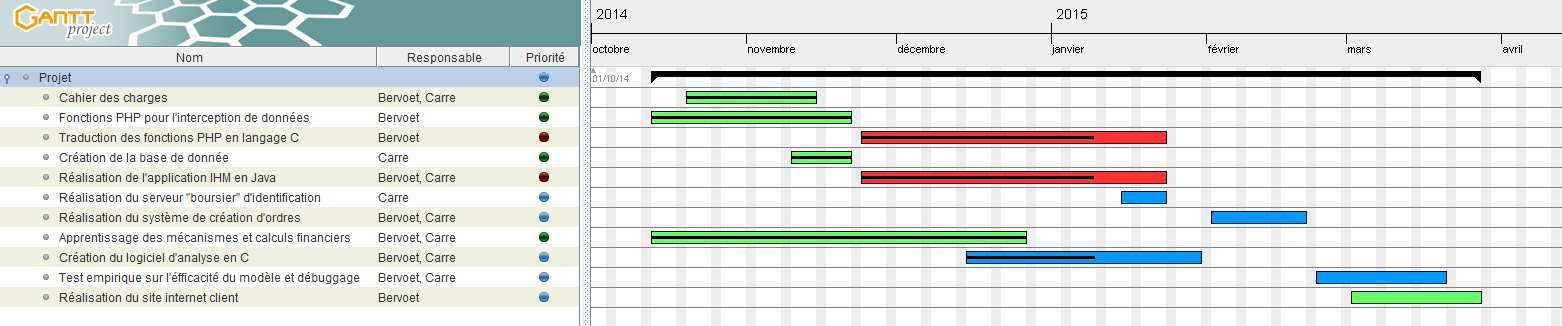
## Site internet

Il n’y a pas vraiment d’information à rajouter sur le site internet, son rôle est seulement de créer un compte pour l’utilisateur et de lui permettre de modifier, remplir son profil d’investisseur particulier en ligne.



# III Update du diagramme de Gant

Dans l’ensemble, les prévisions ne sont pas correctes et je pense que nous sommes dans l’ensemble en retard. Le projet étant assez ambitieux, il est je pense normal que nous soyons en retard, il va falloir travailler plus pendant le semestre 2. Les heures totales réservées au projet dans son ensemble sont-elles depuis longtemps dépassées. Là encore, c’était prévu.



# IIII Conclusion

Les diagrammes montrés ici proviennent de general\_system\_use\_case.uml (le premier) et de System\_use\_case.uml (les autres). Les diagrammes d’activités des composants de AFIS sont joint dans le fichier system\_activity. Un diagramme de classe représente la base de données (pas encore terminée puisqu’il manque les tables d’analyse financières et des éléments dans les tables group, clientData et groupData). Un diagramme de classe représentant l’étape de connexion et le menu de l’interface, en java, est également joint : Interface\_uml\_notDone.uml. Toutes les images dans le rapport sont joint dans le dossier.