ENSSAT LANNION

UNIVERSITÉ DE RENNES 1

LSI – 2e année

Année 2014-2015

Projet d'algorithmique distribuée
Enshare : outil de partage et d'écriture collaborative de documents

Gwénolé Lecorvé, Ioana Suciu

1 Présentation du concept

L'objectif de ce projet consiste en la conception, l'implémentation et le test du logiciel *Enshare* qui est un outil de partage et d'écriture collaborative de documents par plusieurs utilisateurs/clients (« à la Google Drive »). Plus particulièrement, vous devez adapter une version déjà conçue et développée selon une **architecture centralisée** vers une **architecture répartie**.

Ce document présente tout d'abord le logiciel existant à travers des scénarios d'utilisation de l'Enshare, selon le modèle centralisé, puis un descriptif de l'implémentation. Ensuite, le travail qui vous est demandé est détaillé, en matière de fonctionnalités ainsi qu'en terme de rendus.

2 Logiciel existant

La solution d'architecture centralisée s'appuie sur un serveur de document qui gère le stockage, l'accès et la modification de ces documents. Chaque client souhaitant effectué des actions doit à chaque fois communiquer avec le serveur. Notamment, pour la modification, le serveur gère un mécanisme d'exclusion mutuelle dont le protocole est tel qu'à l'accoutumé (« demande », « OK », « fin ») enrichi d'une réponse « échec » en complément du « OK » pour ne pas laisser un client dans l'attente.

Cette solution centralisée est illustrée sur plusieurs scénarios présentés, puis une description de l'implémentation existante est fournie.

2.1 Scénario 1 : un client

Nous considérons un client (client 1) et un serveur de documents. Comme illustrés sur la figure 1, les événements possibles (communications et changements d'état) sont les suivants :

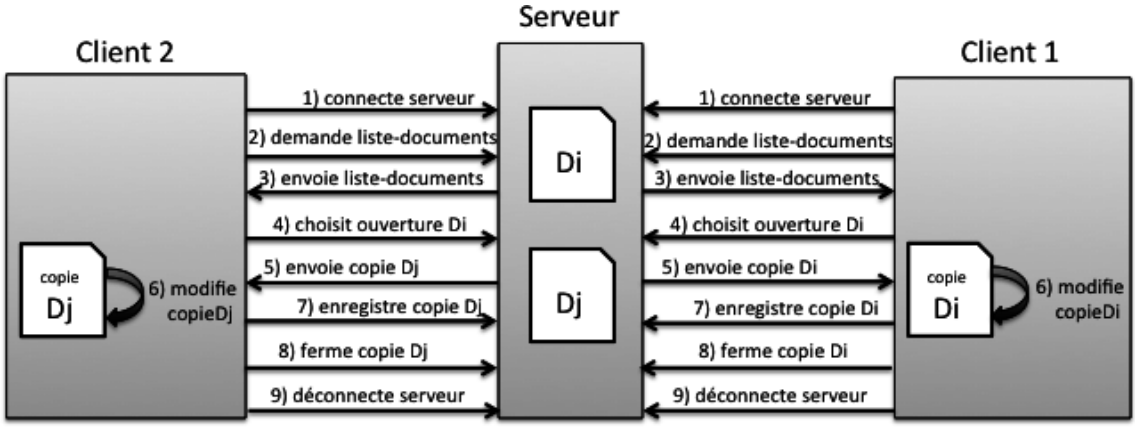

Figure 1 : Exemple de dialogue entre des clients et le serveur sans accès conccurent.

1) client 1 se connecte au serveur ;
2) client 1 demande au serveur la liste des documents partagés ;
3) le serveur lui renvoie la liste des documents ;
4) le client 1 choisit un document parmi la liste envoyée ;
5) le serveur lui envoie une copie du document Di ;
6) le client 1 lit et/ou modifie le contenu de la copie de Di ;
7) quand il a fini, il enregistre (s'il y a eu modification) le document Di sur le serveur qui met à jour son Di ;
8) le client 1 ferme sa copie de Di
9) le client 1 se déconnecte du serveur.

Remarques : La connexion du client 1 au serveur peut être fermée avant l'enregistrement du document (ainsi, 7) et 8) sont manquants).

Les opérations possibles sur le contenu d'un document sont des modifications textuelles classiques (modification, suppression et insertion de lignes).

2.2 Scénario 2 : plusieurs clients sans exclusion mutuelle

C'est le cas de l'accès par deux (ou n) utilisateurs à des documents différents (ou au même document, mais pas en même temps). Les événements possibles pour le client 2 sont, eux aussi, illustrés sur la figure 1.

2.3 Scénario 3 : plusieurs clients avec exclusion mutuelle

La solution proposée pour l'architecture centralisée comporte également la gestion des ressources critiques réalisée par un mécanisme d'exclusion mutuelle. Ce cas correspond à l'ouverture du même document par plusieurs utilisateurs en même temps. Plusieurs scénarios sont possibles, en voici un, illustré par la figure 2 :

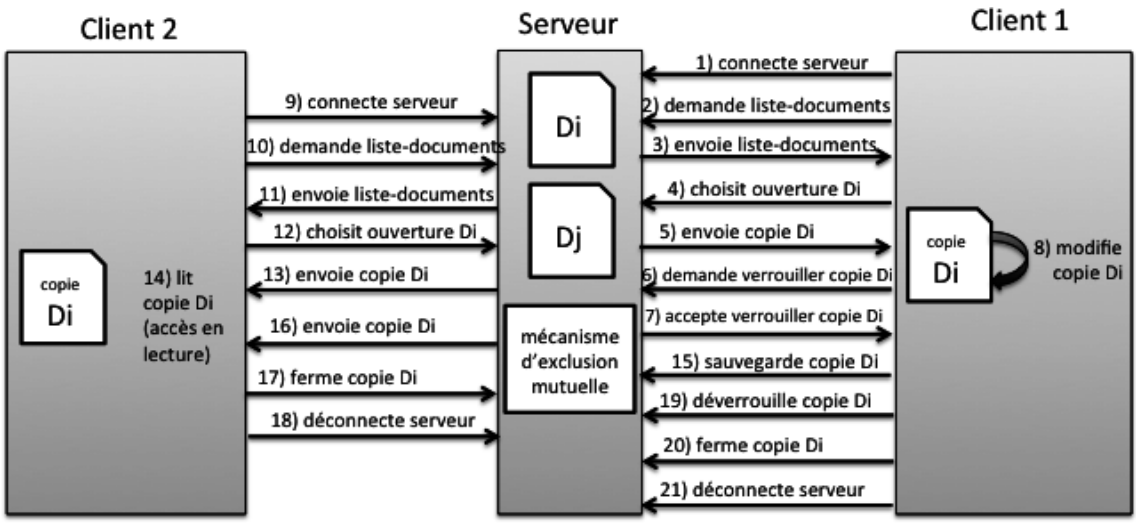

Figure 2 : Exemple de dialogue dans le cas d'un accès concurrent par plusieurs clients.

1) le client 1 se connecte au serveur ;
2) le client 1 demande au serveur la liste des documents partagés ;
3) le serveur lui envoie la liste des documents ;
4) le client 1 choisit un document parmi la liste envoyée ;
5) le serveur lui envoie une copie du document Di ;
6) le client 1 demande au serveur de verrouiller le document Di ;
7) le serveur accepte (ou refuse) de verrouiller le document Di ;
8) le client 1 commence à modifier le contenu de la copie de Di ;
9) le client 2 se connecte au serveur ;
10) le client 2 demande au serveur la liste des documents partagés ;
11) le serveur lui envoie la liste des documents ;
12) le client 2 choisit également d'ouvrir Di ;
13) le serveur lui envoie une copie du document Di ;
14) le client 2 lit le document, mais il ne peut pas modifier le contenu de sa copie de Di ;
15) le client 1 a fini la modification de sa copie de Di, et la sauvegarde sur le serveur qui mettra à jour son Di ;
16) le serveur envoie la nouvelle copie de Di à tous les clients qui sont en mode lecture et/ou écriture sur l'ancien Di (ici, client 2) ;
17) le client 2 ferme sa copie de Di ;
18) le client 2 se déconnecte du serveur ;
19) le client 1 décide de déverrouiller la copie de Di ;
20) le client 1 ferme sa copie de Di ;
21) le client 1 se déconnecte du serveur.

Remarque : Pour faire une modification ou sauvegarder, il faut être en mode « verrouillé ». Sauvegarder et déverrouiller impliquent une mise à jour sur le serveur du document en question, avec une diffusion de la part du serveur de la nouvelle copie aux clients qui ont demandé l'accès au document en question (la différence entre les deux étant que sauvegarder fera, en plus, une mise à jour du document sur le disque de stockage).

2.4 Diagramme de communication

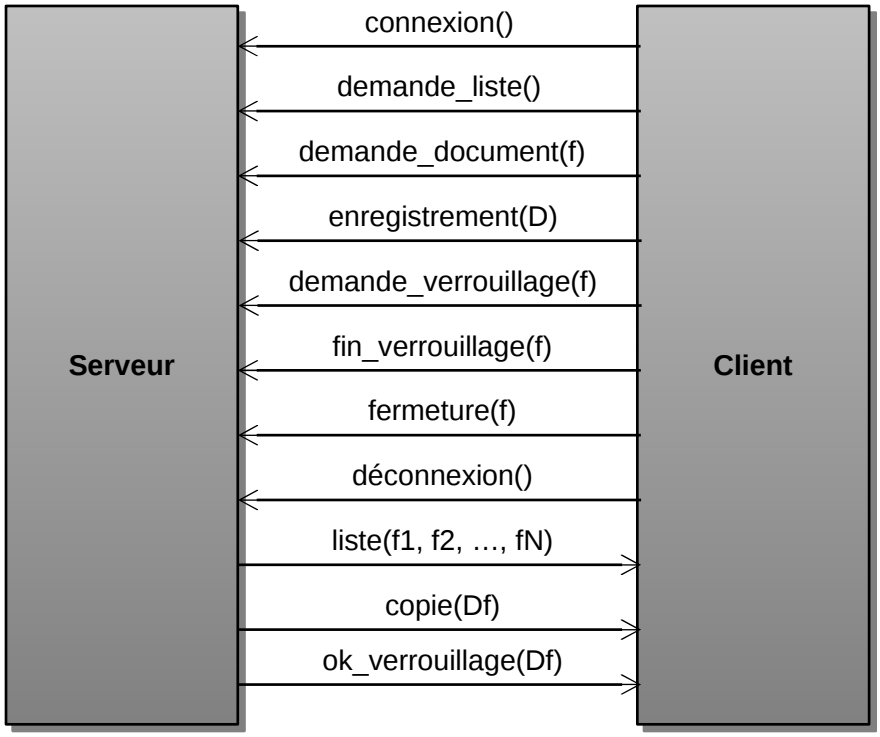

Figure 3 : Diagramme de communication de l'architecture centralisée.

2.5 Code fourni

La version centralisée du logiciel Enshare est codée en Java en utilisant RMI pour les communications inter-processus. Vous trouverez sur l'ENT une archive de ce code de départ. Le code est documentée et accompagné d'un fichier `README` permettant de s'en servir. Le logiciel peut être testé en mode console mais il dispose également d'une interface graphique s'appuyant sur Swing. L'architecture actuelle se compose de 2 packages : `document` qui contient les classes « métier » et `enshare` qui contient les contrôleurs et les interfaces graphiques. Le package `enshare` est lui-même divisé en 2 sous-packages : l'un pour la partie serveur (package `enshare.server`), l'autre pour la partie client (package `enshare.client`). Les diagrammes de classes de ces packages sont donnés en figure 4, 5 et 6.

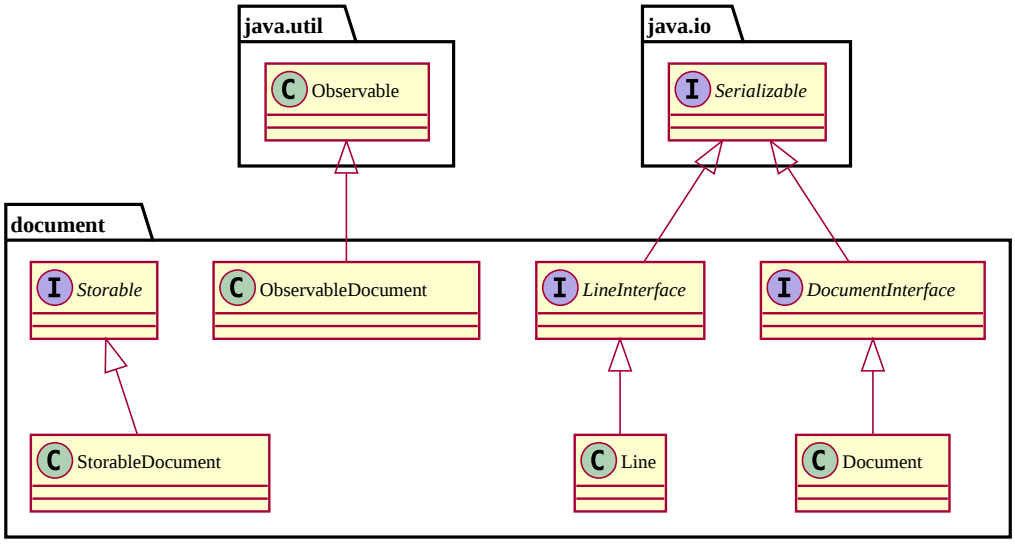

Figure 4 : Diagramme de classes du package `document`.

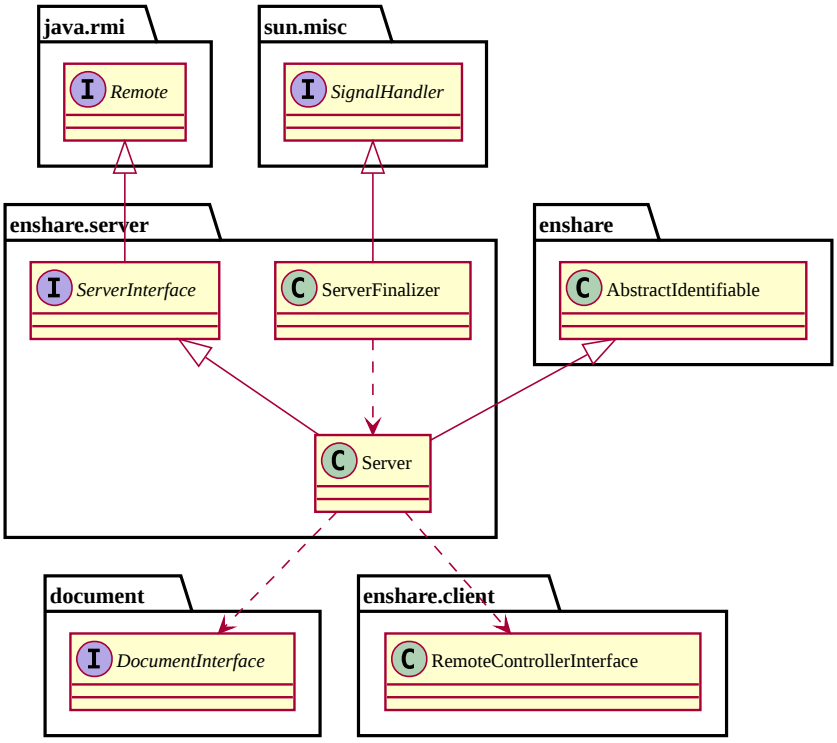

Figure 5 : Diagramme de classes du package `enshare.server`.

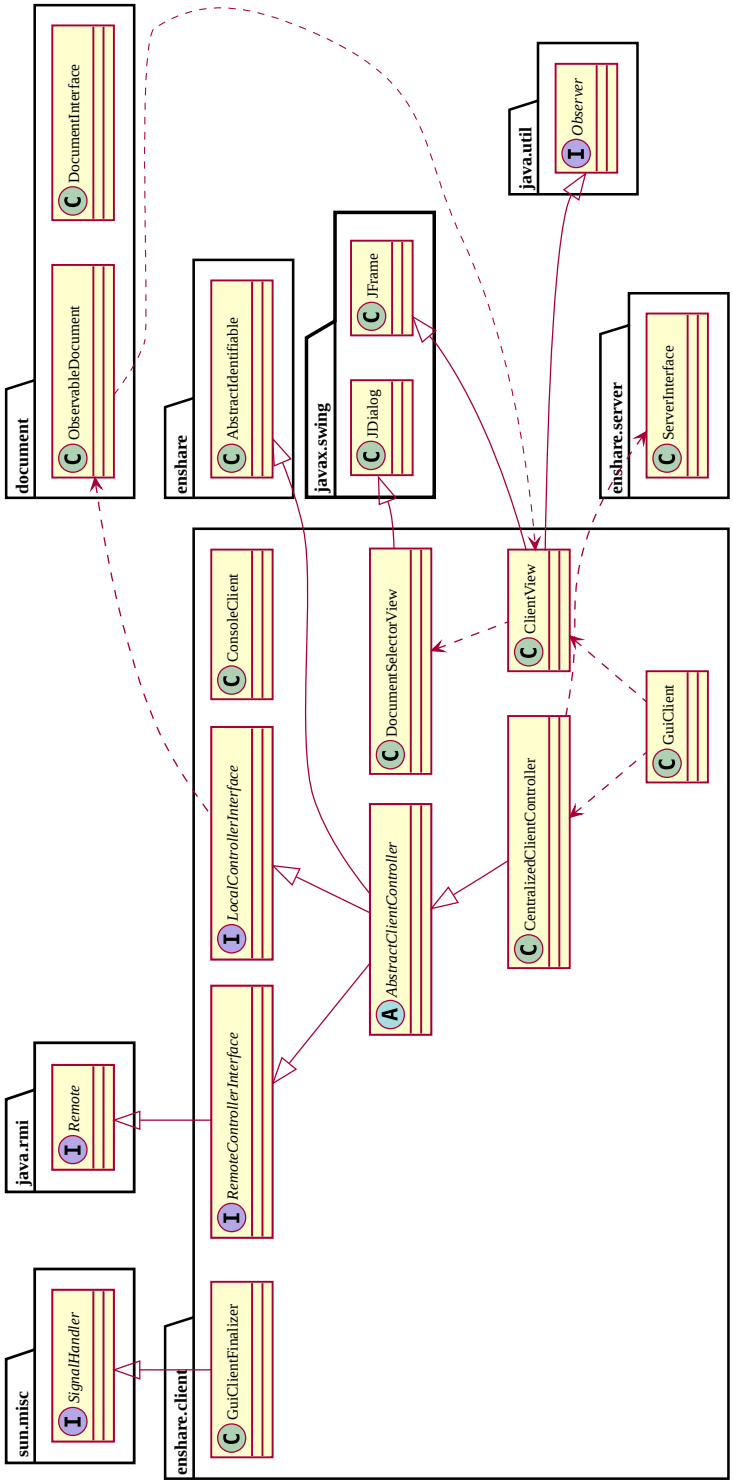

Figure 6 : Diagramme de classe du package `enshare.client`.

Il est principalement important que les fonctionnalité des diverses classes est largement spécifié à travers des interfaces. Notamment, la classe générique de contrôleur de client `AbstractClientController` implémente les méthodes qui peuvent être appelées à distance telles que les définit l'interface `RemoteControllerInterface` et celles qui peuvent l'être en local, notamment par une interface graphique, telles que les définit l'interface `LocalControllerInterface`. La version totalement centralisée du contrôleur du client est donnée par la classe `CentralizedClientController`. Cette dernière classe ne fait principalement que transmettre les appels qu'elle reçoit en local vers le serveur.

3 Travail demandé

Dans cette section, les fonctionnalités attendues sont tout d'abord décrites, avant d'aborder les délivrables et de la séance de démonstration de fin de projet.

3.1 Fonctionnalités

En partant de la solution d'architecture centralisée décrite auparavant, il vous est demandé de proposer une solution répartie pour le partage et l'édition des documents multi-utilisateurs. Plusieurs cas peuvent être envisagées et sont à implémenter graduellement :

Cas 0 : présence d'un serveur qui stocke les documents et gère l'exclusion mutuelle, document par document (fourni) ;

Cas 1 : présence d'un serveur qui stocke les documents mais exclusion mutuelle repartie, toujours document par document ;

Cas 2 : présence d'un serveur qui stocke les documents mais exclusion mutuelle repartie, ligne par ligne pour chaque document ;

Cas 3 : pas de serveur, documents stockés chez les clients et exclusion mutuelle repartie, ligne par ligne pour chaque document.

La résolution du cas 1 est exigée pour la validation du projet. Les projets ayant (correctement) traité les cas 2 et 3 seront considérés respectivement comme bons et excellents. Le code de la solution pour chaque cas doit être conservé et doit pouvoir être exécuté.

Remarque : Privilégiez tout d'abord des tests en console avant d'utiliser l'interface graphique.

Le choix de l'algorithme d'exclusion mutuelle à implémenter vous revient. Il conviendra notamment une solution qui limite, si possible, la bande passante (taille et nombre des messages).

3.2 Livrables attendus

Plusieurs livrables sont attendus :

- **(DL1) Vos programmes commentés et documentés.** Vous êtes encouragés à créer des packages vous permettant de structurer intelligemment votre code. Votre code devra être documenté (javadoc) et commenté. Le code source sera fourni sous la forme d'une archive intitulée `code_projet_algo_dist_LSI2_N1_N2_N3.tgz` (ou `.zip`) où les *Ni* sont à remplacer par les noms des membres du groupe. Votre archive devra contenir un fichier `README` indiquant au minimum comment utiliser les fichiers fournis dans l'archive (pré-requis système, procédures d'installation et de lancement). Le correcteur se mettra dans la peau d'un client et n'ira pas explorer le code pour faire marcher le programme.

  → À déposer sur l'ENT avant le **lundi 19 janvier 2015 à 09h00**.

- **(DL2) Un rapport** expliquant l'organisation mise en œuvre et vos choix d'implémentation. Il s'agira de montrer :
  - comment vous vous êtes organisés pour le développement (diagramme de Gantt, répartition des tâches) ;
  - comment vous avez mis en œuvre le système (diagramme de communication, code CSP, diagramme de classes...) ;
  - comment vous avez testé votre application répartie.

  Le rapport comportera **au maximum quinze pages** (annexes incluses). Si vous désirez inclure des annexes, celles-ci ne devront pas comporter d'impression des fichiers de code. Le rapport aura une forme similaire à celui du projet Java pour lequel un document d'aide à la rédaction du rapport vous a été fourni sur l'ENT. Le rapport prendra la forme d'un fichier PDF nommé `rapport_projet_algo_dist_LSI2_N1_N2_N3.pdf` où les *Ni* sont à remplacer par les noms des membres du groupe.

  → À déposer sur l'ENT avant le **lundi 26 janvier 2015 à 12h00**.

3.3 Démonstration

La dernière séance de projet (lundi 19 janvier 2015, 14h00-16h00) sera dédiée aux démonstrations. **La démonstration devra durer 7 minutes.** Elle débutera par un descriptif de votre implémentation et continuera avec la présentation d'un ou plusieurs scenarios que vous aurez judicieusement élaborés de façon à (i) montrer que votre application respecte le cahier des charges et (ii) mettre l'accent sur les points forts de votre application. Tous les membres du groupe devront participer à la démonstration. Vous organiserez la démonstration de façon à prendre la parole à tour de rôle, avec une répartition équitable du temps de parole. Suivront ensuite **3-4 minutes de questions**.