UNIVERSITÉ DE SHERBROOKE
DÉPARTEMENT D'INFORMATIQUE

IFT 585 – Télématique

Travail pratique #3

Travail présenté à
M. Bessam ABDULRAZAK

par

Raphaël Cayouette (15 064 430)
Jean-Philippe Croteau (15 072 957)
Jonathan Martineau (15 059 124)
Félix Vigneault (15 079 576)

Le 28 juillet 2017

Description du contenu du zip

Le ZIP de notre soumission contient notre projet de code et les exécutables *Client.exe* et *Server.exe* qui permettent de démarrer les applications sans ouvrir la solution Visual Studio 2017. Le code est séparé en différents projets. Le projet *Client* contient l'interface client et les méthodes qui permettent d'entrer en communication avec le serveur. Le projet *Server* contient l'interface du serveur, ainsi que les classes permettant le stockage de données, la gestion des clients et des groupes, la programmation de tâches, et les méthodes qui permettent de gérer la communication provenant des clients. Le projet *ShareLibrary* contient une multitude de classes et interfaces utilisées par l'ensemble des autres projets de la solution. Le dossier *Communication* contient tout ce qui fait l'abstraction de la connexion TCP pour le client et le serveur. Celui-ci implémente le concept RPC (*Remote procedure call*). Le dossier *Models* contient plusieurs structures utilisables par tous les autres projets. Le dossier *Summary* contient le fichier *GroupSummary* qui représente la liste des fichiers d'un groupe. Le dernier dossier de ce projet, *Utils*, contient le fichier *DiskAccessUtils*, qui permet la sauvegarde et la lecture de fichiers XML que nous utilisons afin de sauvegarder l'information d'exécution sur le disque entre les sessions. Enfin, les projets *ClientTests*, *ServerTests* et *ShareLibraryTests* contiennent des tests unitaires qui s'assurent de valider l'intégrité du code et la sortie des résultats désirés.

Procédure d'exécution

Installation

Avant de commencer l'exécution, il faut s'assurer que le serveur soit accessible par les clients via le port choisi, puis aller chercher l'adresse IP de celui qui sera le serveur. (Notre programme fonctionne sur une ou plusieurs machines) Ensuite, il faut démarrer Server.exe sur la machine et entrer l'adresse IP du poste dans la case du formulaire. Le dossier de stockage par défaut sera créé au même endroit que l'exécutable. Ensuite, sur les postes clients, il faut démarrer Client.exe et entrer l'adresse IP du serveur dans la case appropriée. Les ports par défaut peuvent être changés, mais doivent correspondre entre le serveur et les clients. Le dossier de stockage par défaut des clients sera également créé à l'endroit de l'exécutable. Il est important de noter que chaque client doit avoir un dossier principal différent. Le bouton *Connexion* initie alors la connexion avec le serveur.

Utilisation

Tout d'abord, les usagers doivent entrer un nom d'utilisateur et un mot de passe, puis cliquer sur *Créer usager*. Si l'usager existe déjà, il faut cliquer sur *Connexion* plutôt que *Créer usager*.

Ensuite, plusieurs options sont disponibles. Il faut commencer par créer un groupe avec un nom et une description. Quand d'autres clients seront créés, ils seront visibles dans la boîte *Usagers non membres du groupe*. Le bouton *Inviter à ce groupe* envoie une notification à l'utilisateur sélectionné. Il pourra ainsi rejoindre le groupe s'il le désire. Une réponse positive l'ajoutera à la liste des membres du groupe et créera le dossier avec les fichiers du groupe sur son répertoire local. Le bouton *Enlever du groupe* permet à l'administrateur de retirer des clients de son groupe. Ceci supprimera aussi le dossier local du groupe sur l'ordinateur du client. Le bouton *Rendre administrateur du groupe* permet, comme son nom l'indique, de donner nos droits d'administrateur au client sélectionné.

Puisque tous les usagers peuvent créer des groupes, le bouton *Rejoindre ce groupe* permet d'envoyer une notification à l'administrateur du groupe sélectionné dans la liste déroulante. Comme pour les invitations, une réponse positive à cette notification ajoutera l'utilisateur au groupe et lui téléchargera le dossier et les fichiers du groupe.

Tel que visible dans le haut de l'interface, le texte *Temps avant sync* affiche un minuteur du temps restant avant la prochaine synchronisation des fichiers avec le serveur. Si le chiffre devient "NOW!", l'interface client devient figée et le client synchronise ses fichiers avec le serveur. À noter que si les fichiers sont plutôt gros, cela peut durer un petit moment.

L'ajout, la modification ou la suppression de fichiers dans le dossier d'un groupe reproduira automatiquement l'action correspondante sur le serveur lors de la synchronisation, puis tous les usagers du groupe recevront ces changements lors de leur propre synchronisation avec le serveur.

Enfin, la liste *Noms des usagers en ligne* affiche tous les usagers qui étaient en ligne dans les cinq dernières minutes. Si le nom de l'usager est jaune, cela indique que ça fait plus de deux minutes qu'il n'a pas effectué une action auprès du serveur. Si l'usager était actif dans les deux dernières minutes, son nom sera vert.

Si le client ferme son application, lors de la réouverture et la reconnexion avec le serveur, le client se synchronisera avec celui-ci et tout reprendra tel que laissé à la fermeture. De plus, les listes de groupes et de clients du serveur sont sauvegardées sur le disque afin de pouvoir les recharger à sa réouverture.

La description sommaire du système solution

Le serveur est responsable de contrôler l'accès et la modification aux fichiers. Il utilisera des connexions TCP sur des sockets afin de recevoir les commandes des clients. Les clients feront eux-mêmes des demandes de synchronisation et seront responsables de fournir les informations nécessaires au serveur pour qu'il retourne toutes les nouvelles modifications.

La description des modules principaux

Serveur

Afin de préserver la cohérence des données, le serveur devra utiliser un processus de mise en attente d'opérations afin de pouvoir recevoir plusieurs commandes de clients différents, en même temps, sur des connexions TCP différentes. Il devra aussi gérer le fait que les clients ne seront pas nécessairement à jour dans leur version locale des fichiers lors de leurs demandes d'opérations. Des procédures seront mises en place pour gérer les cas de multiples opérations sur un même fichier par plusieurs clients différents.

Certaines opérations auront une importance plus élevée et effaceront les opérations d'importance inférieure lorsqu'elles arrivent de façon simultanée. Par exemple, si un client demande la suppression d'un fichier et qu'un autre demande la modification de ce même fichier en même temps, une des deux opérations sera traitée en premier par le serveur, mais

peu importe l'ordre des opérations, c'est la suppression qui va l'emporter puisqu'elle a une importance plus élevée que la modification.

Ordre d'importance des opérations :

1. Suppression
2. Modification avec la date de modification la plus récente
3. Ajout avec la date de modification la plus récente

Le serveur aura aussi comme rôle de gérer les droits d'accès aux informations concernant les utilisateurs, les groupes et les dossiers. Il devra donc garder les informations concernant les clients, les groupes, la composition des groupes, la structure des dossiers (fichiers présents) et le contenu des fichiers. Cette information est persistante afin de ne pas être perdue lors d'un redémarrage ou une panne du serveur.

Le contrôle d'accès se basera sur un système d'authentification où les clients fournissent un nom d'utilisateur et un mot de passe afin de s'identifier. Le serveur gardera une trace de qui est connecté grâce à un horodatage de la dernière opération. Les demandes des clients devront être validées lors de chaque appel afin de s'assurer que leur demande est valide selon leurs appartenances aux groupes et leur statut dans le groupe (administrateur ou membre).

Client

Le client aura une version locale des dossiers auxquels il a accès qui sera placée dans un dossier racine de son choix. Il sera responsable de lui-même analyser les modifications qu'il fait sur les fichiers. Pour ce faire, il sera en mesure de calculer un résumé de l'état courant de ses fichiers en itérant sur tous les fichiers de ses dossiers pour construire une liste contenant toutes les informations sur les fichiers. En conservant le résumé de l'état de la dernière capture, le client pourra refaire une capture et comparer ces deux captures pour déterminer les fichiers ajoutés, modifiés et supprimés.

Le client est responsable de communiquer correctement ses modifications et son état local au serveur afin que le serveur le mette à jour correctement.

Communication et connexion TCP

Pour synchroniser les opérations locales, le client calculera ses modifications locales ainsi que le résumé de l'état courant pour les envoyer au serveur, qui choisira quelles modifications peuvent être appliquées. Le client transférera ses fichiers en même temps que ses demandes d'ajout et de modification pour que le serveur puisse appliquer directement les modifications à sa version interne si elles sont valides. Dans la réponse du serveur, le client pourra retrouver les dernières modifications faites par les autres clients et pourra alors mettre à jour sa version locale des fichiers. De cette façon, le client sera complètement synchronisé lors de chaque vérification récurrente (à toutes les 15 secondes).

Pour que le serveur puisse communiquer avec un client, le client devra créer des connexions TCP avec le serveur pour envoyer ses commandes. Chaque commande est

associée à une réponse. Étant donné que les commandes du client peuvent être très espacées entre elles (dépendent de l'utilisateur), nous avons choisi de créer une nouvelle connexion TCP à chaque commande. Cela permet de réduire le nombre de Sockets ouverts sur le serveur et simplifie grandement la gestion des connexions TCP ouvertes et fermées.

Nous avons implanté le concept de RPC (Remote Procedure Call) pour les communications client/serveur. Pour ce faire, nous avons une interface commune qui représente l'API serveur et sur laquelle le client peut faire un appel normal sans que cela apparaisse comme un appel réseau. Pour ce faire, nous avons utilisé la réflexion en .Net, ce qui nous permet de passer une référence vers la méthode exacte que nous tentons d'appeler. Les appels RPC sont sujets à des erreurs réseau et le client doit donc valider la réussite de chaque appel afin d'être certain qu'il n'y a pas eu d'erreur.

Pour que le serveur puisse analyser correctement les commandes, il doit connaître la longueur des messages envoyés par le client (et vice-versa). Pour ce faire, nous avons préfixé chaque message avec la longueur du message à recevoir. Donc, chaque message comprend 32 bits au début qui indiquent la taille de ce qu'il reste à recevoir, ce qui permet théoriquement d'envoyer jusqu'à 4 Gb dans un seul message.

Nous avons une limitation importante imposée par nos choix de conception en matière de communication. En effet, lors du transfert d'un fichier de grande taille, le client termine son envoi bien avant que le serveur ait fini de recevoir le fichier. Le client doit donc attendre la réponse du serveur pendant une quantité de temps suffisante pour permettre le transfert. Toutefois, étant donné que les connexions sont de vitesses variables et qu'il est difficile d'estimer le temps de transfert d'un fichier, nous ne pouvons pas donner un délai réaliste à tout coup pour l'attente de la réponse. Cela entraîne une Timeout et la connexion est présumée perdue par le client. Notre gestion de la synchronisation client/serveur permet toutefois de rattraper cette erreur et la mise à jour des fichiers a tout de même lieu. Malgré la grande fiabilité de TCP, il faut quand même implémenter de la validation au niveau de la couche application puisque la couche session n'est pas présente.

Chaque message aura un type sur 32 bits qui permettra de déchiffrer le contenu du message. Les types Request et Response sont ceux utilisés actuellement. Les messages de type Request contiennent les informations pour identifier la méthode appelée par le client ainsi que les valeurs des paramètres passés. Les messages de type Response contiennent uniquement l'objet retourné par la méthode serveur. Les tailles variables des champs de nos messages sont gérées automatiquement par la sérialisation de .Net. Il est donc important que les clients utilisent .Net, sinon ils ne pourront pas facilement déchiffrer les messages.

Format d’un message générique :

Taille du message (32 bits)	Type du message (32 bits)
Contenu du message	

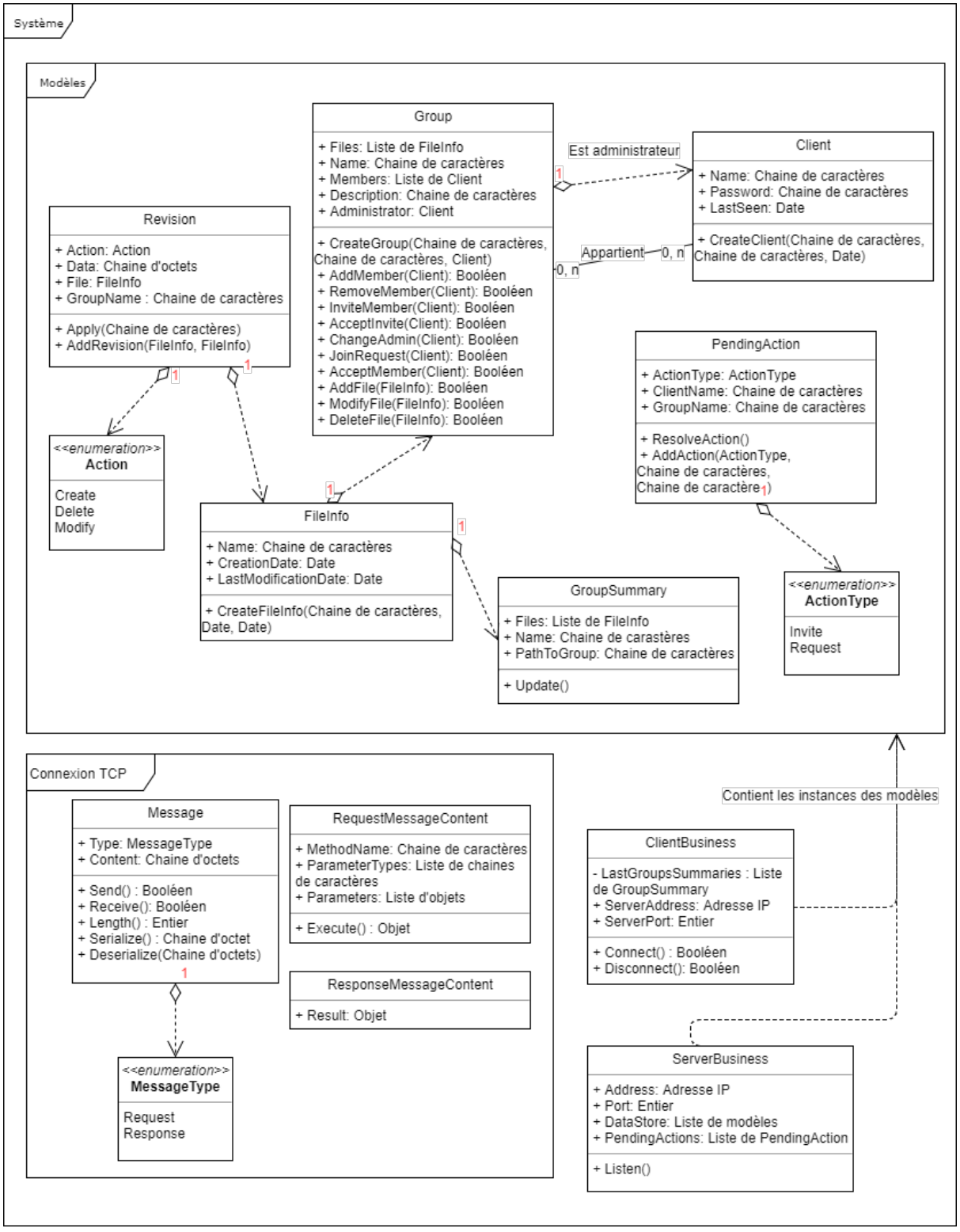
Format d’une Request :

Taille du message (32 bits)	Type du message (32 bits) = Request
Nom de la méthode (variable)	
Liste des types de paramètres (variable)	
Liste des valeurs des paramètres (variable)	

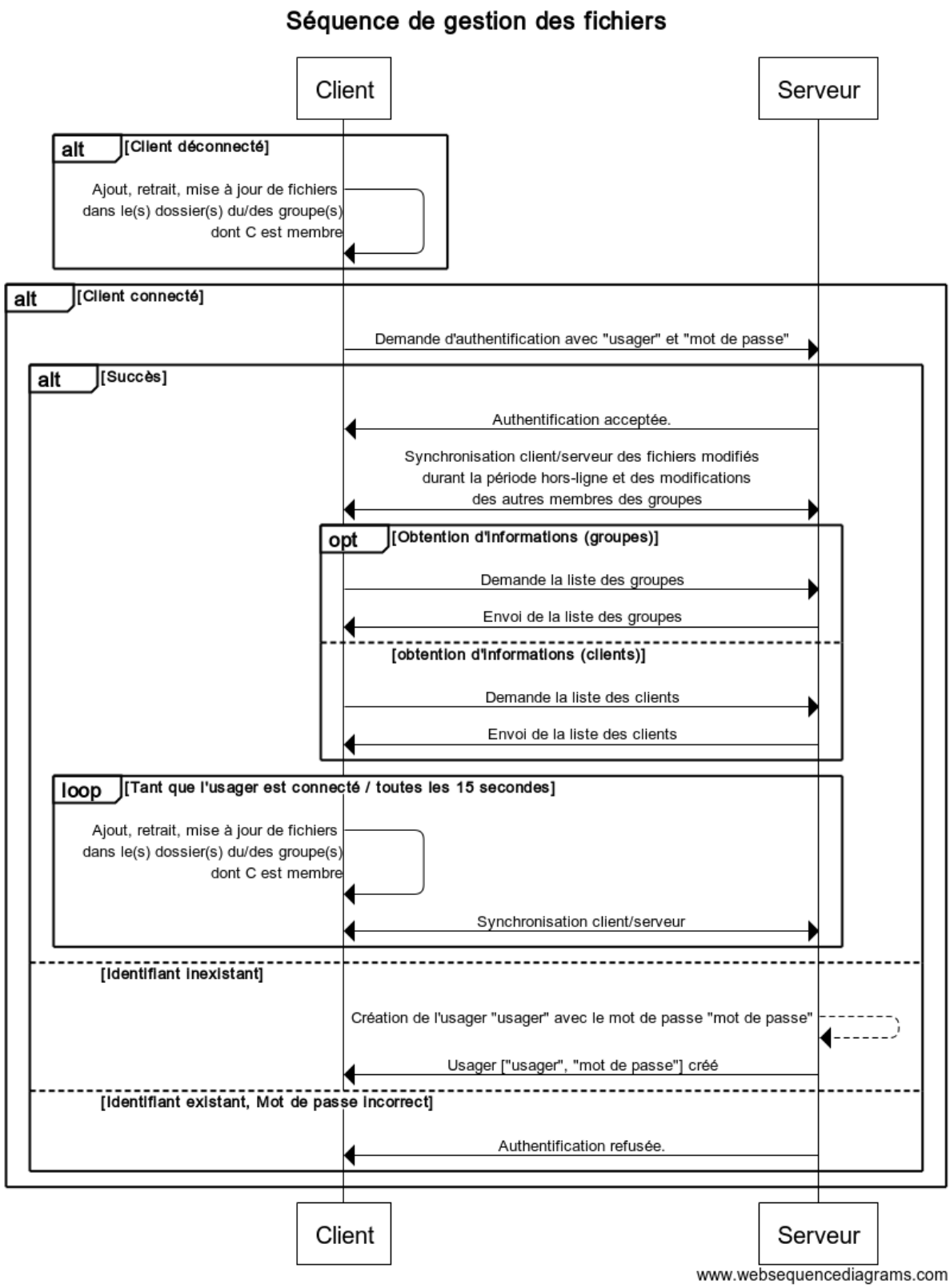
Format d’une Response :

Taille du message (32 bits)	Type du message (32 bits) = Response
Valeur du retour (variable)	

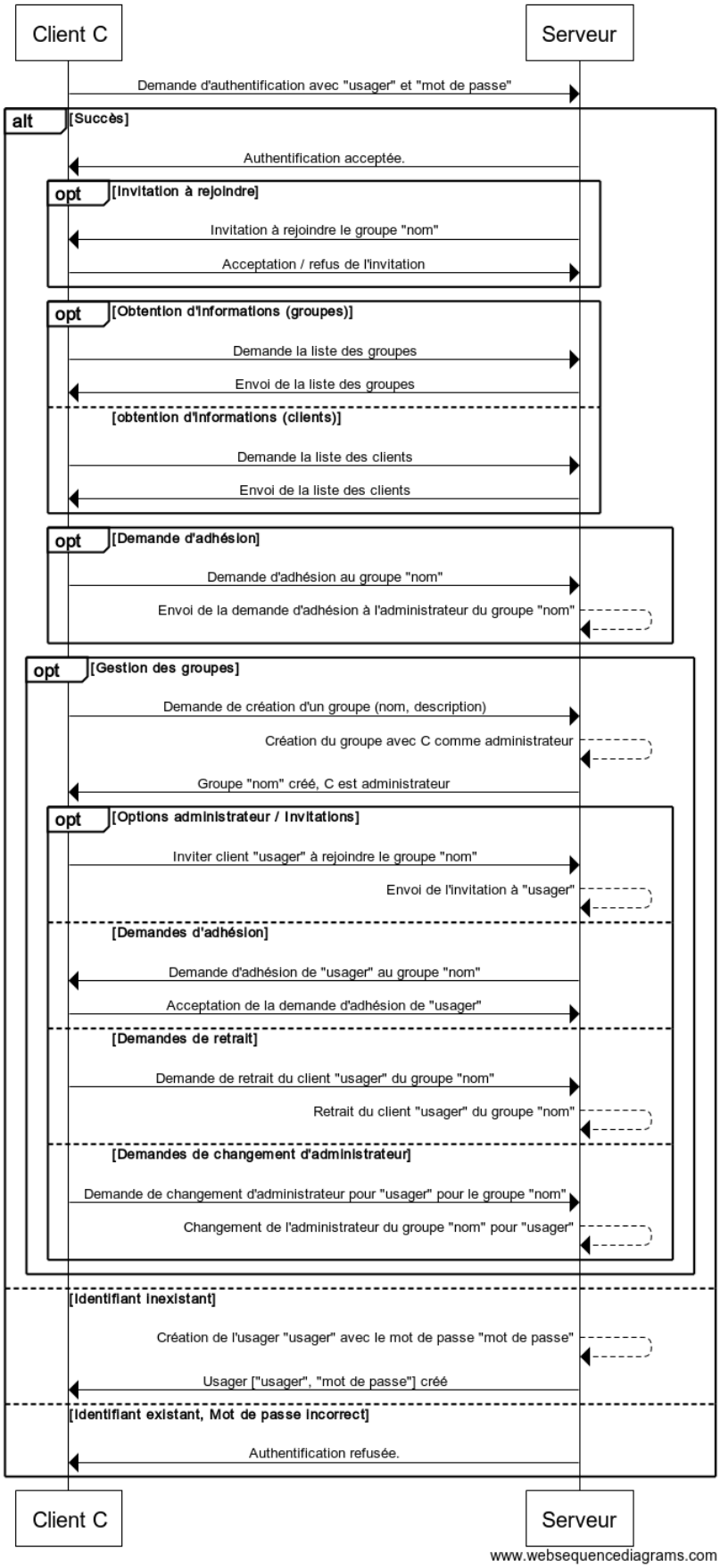
Le diagramme de classes



Les diagrammes de séquence

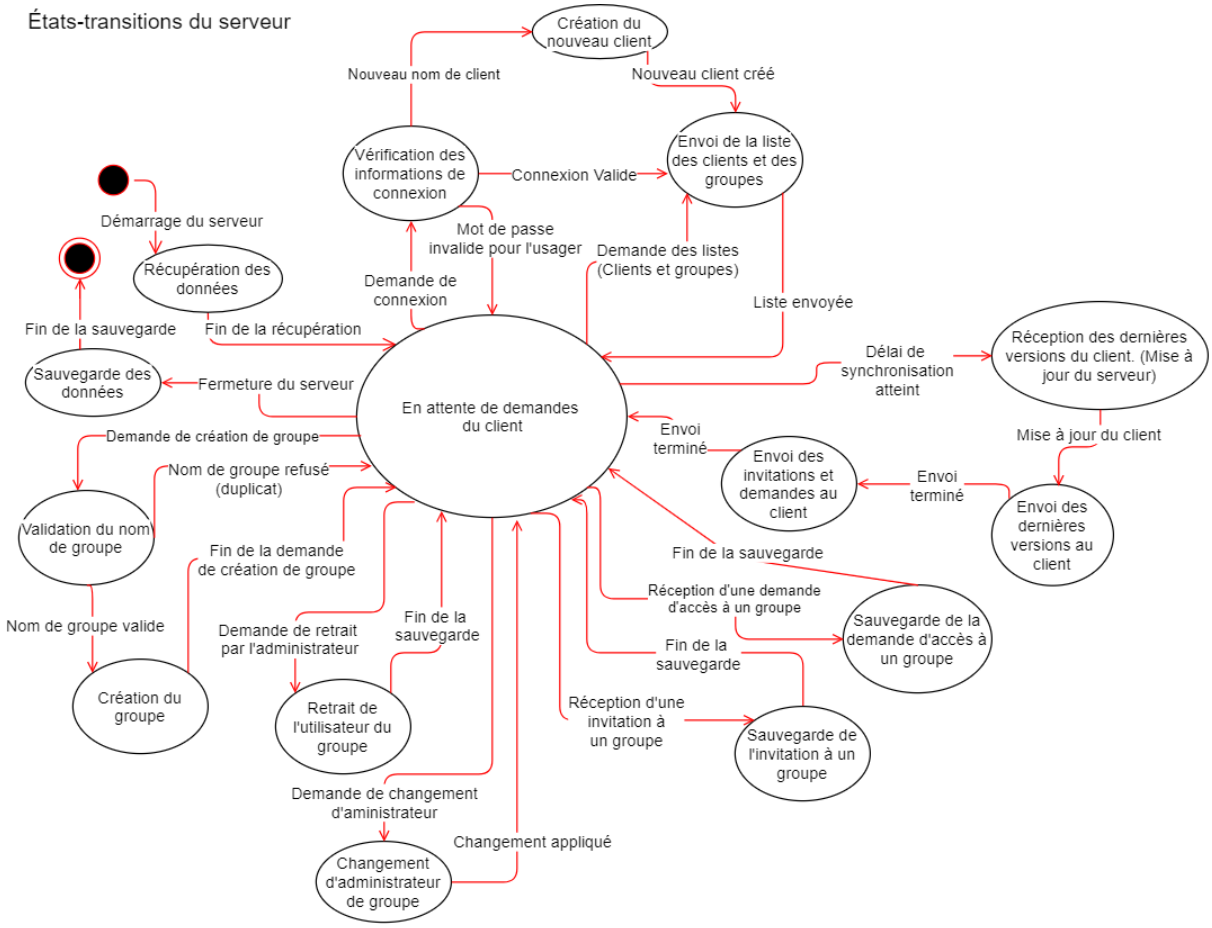


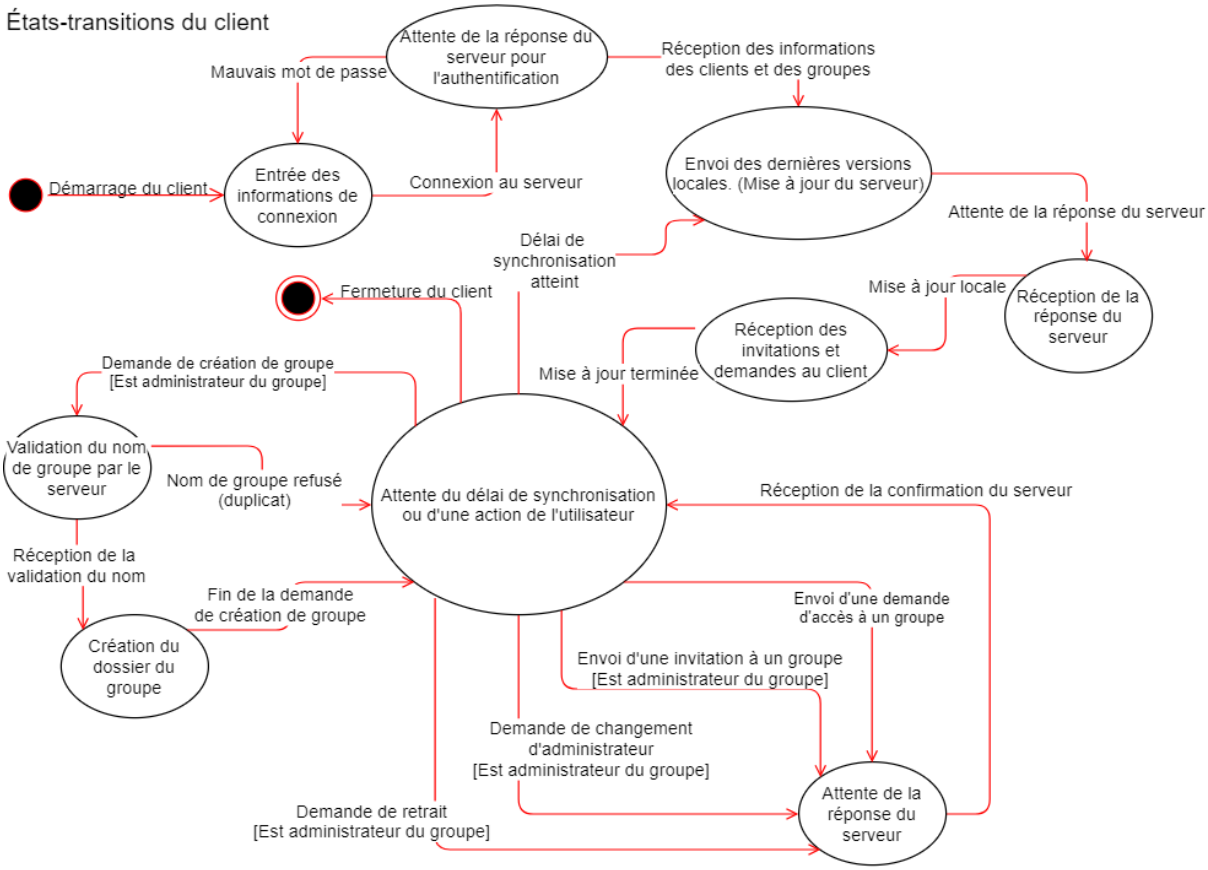
Séquence de gestion des groupes



www.websequencediagrams.com

Les diagrammes d'états transitions




États-transitions du client
Attente de la réponse du serveur pour l'authentification
Mauvais mot de passe
Réception des informations des clients et des groupes
Démarrage du client
Entrée des informations de connexion
Connexion au serveur
Envoi des dernières versions locales. (Mise à jour du serveur)
Attente de la réponse du serveur
Délai de synchronisation atteint
Fermeture du client
Mise à jour locale
Réception de la réponse du serveur
Réception des invitations et demandes au client
Mise à jour terminée
Demande de création de groupe [Est administrateur du groupe]
Validation du nom de groupe par le serveur
Nom de groupe refusé (duplicat)
Attente du délai de synchronisation ou d'une action de l'utilisateur
Réception de la confirmation du serveur
Réception de la validation du nom
Fin de la demande de création de groupe
Création du dossier du groupe
Envoi d'une demande d'accès à un groupe
Envoi d'une invitation à un groupe [Est administrateur du groupe]
Demande de changement d'administrateur [Est administrateur du groupe]
Demande de retrait [Est administrateur du groupe]
Attente de la réponse du serveur