Réalisation d'une application de messagerie instantanée

Génie Logiciel
Gestion de Projet

Unité d'enseignement 1 : Ingénierie du logiciel

Unité d'enseignement 2 : Réseaux et Applications Distribuées

Année 2013 - 2014

Dorian Coffinet

Mickaël Fardilha

Raphaël Pillie

Thibault Gauthier

Table des matières

Génie Logiciel

Les outils

Les outils nécessaires pour le développement d'un tel projet sont les suivants :

1. Un système d'exploitation
2. Un langage de développement
3. Des normes de codage
4. Un IDE (*Integrated Development Environment)*
5. Un logiciel de gestion de versions

Le cahier des charges spécifie que le projet doit s'exécuter sur la plate-forme Linux, c'est pourquoi ce système a été retenu pour le développement de l'application. Le langage de développement est également imposé par le cahier des charges. Il s'agit du langage Java. La version standard actuelle de ce langage est la version 7 pour les distributions Linux depuis 2013. Elle apporte de nombreuses améliorations mineures qui permettent dans un premier temps d'améliorer la lisibilité du code (Switch sur les String, Multicatch, Lecture de fichier, etc...). Le projet sera compilé via Ant. L'IDE Eclipse a été choisi car il s'agit d'un IDE libre et facile à prendre en main. De plus de nombreux plugins sont disponibles ce qui permet de produire non seulement du code source mais également des diagrammes de classe UML par exemple. Les plugins utilisés sont WindowsBuilderPro pour le développement de l'interface graphique, JUnit pour les tests unitaires et ObjectAid UML pour la création des diagrammes de classes. Les logiciels de gestion de versions utilisés sont Git et Svn. Le répertoire du projet est, quant à lui, hébergé sur la plate-forme GitHub.

Les normes de codage

Pour que tout le monde puisse s'y retrouver, y compris les personnes qui n'ont pas participé à la rédaction du code source, il est important de définir des règles/normes de codage. Nous avons choisis de suivre les recommandations du langage Java.

Le code source du logiciel est rédigé en anglais et les commentaires sont en français. Les commits sur le répertoire Git sont également en français. Cela permet de différencier rapidement le code source des commentaires.
Les commits sont effectués régulièrement pour garantir une lisibilité du code plus importante. De plus les rollbacks sont plus faciles à réaliser car moins de code est modifié.
Les normes de codage utilisées sont les suivantes :

- Les noms de package commencent par une minuscule.
- Les noms de classe commencent par une majuscule.
- Les noms de variables et de fonctions commencent par une minuscule et suivent la règle du camelcase (chaque terme est collé au précédent et commence par une majuscule).
    - Exemple : ceciEstUnExemple.
- Les noms de variables et de fonctions sont explicites.
- Une Javadoc vient compléter les commentaires.

L'architecture du logiciel

Afin de rendre le code source plus compréhensible et réutilisable, nous l'avons découpé en plusieurs packages. Chaque package contient lui même plusieurs classes.

Les packages sont les suivants :

- userInterface : contient les classes graphique.
- model : contient les classes métiers (Client/Serveur/Thread).
- network : contient les classes protocoles réseau.
- jUnitTest : contient les classes pour les tests unitaires.
- dist : contient le fichier exécutable avec l'extension .jar.
- doc : contient les documents liés au projet.

Le diagramme de classes UML est disponible au format PDF dans le dossier du projet (Uml.pdf).

Gestion de Projet

Présentation de l'équipe

Notre équipe se compose de 4 personnes. Nous venons tous de différents parcours :

- Dorian COFFINET a fait un DUT à Nancy et une L3 Pro à Angers.
- Mickaël FARDILHA a fait un DUT à Tours et une L3 Pro à Angers.
- Raphaël PILLIE a fait un DUT à Caen et une L3 à Caen.
- Thibault GAUTHIER a fait un BTS à Chemillé et une L3 Pro à Angers.

Chacun a apporté diverses briques de connaissances tiré de ses formations pour mener à bien le projet. De ce fait nous avons tous appris quelque chose en dehors du sujet du projet.

Mise en place des environnements

Avant de choisir les outils qui serons utilisés, nous avons fait un tour du groupe pour savoir sous quelle plate-forme de développement chacun voulait travailler. Nous avons choisi unanimement Linux. De plus le cahier des charges spécifie que l'application doit fonctionner pour la distribution Ubuntu 12.10. Nous avons donc travaillé sur des distributions défférentes mais toutes sont basées sur Debian (Ubuntu, Kubuntu, Mint).

La deuxième contrainte était de développer avec le langage Java et plus spécifiquement avec la version standard disponible sous Linux Ubuntu 12.04. La version disponible par défaut sous cet environnement est la version 6 du langage Java. Après concertation, nous avons choisi d'utiliser la version 7 car elle est devenue un standard sur de nombreuses distributions depuis 2013. Une discussion avec les professeurs chargés du projet nous a conforter dans le choix de cette version.

Nous avons ensuite du choisir un IDE (*Integrated Development Environment)* pour le développement du projet. L'IDE Eclipse a été retenu à l'unanimité car chaque personne du groupe connaissait cet environnement avant le début du projet. La dernière version disponible sur le site d'Eclipse est la suivante : Kepler 4.3.1. Nous avons utilisé divers plugins pour la réalisation du projet. Ils nous ont permis de réalisé notamment le diagramme de classes UML, l'interface graphique, etc.

Un autre point important lorsque l'on travaille en groupe sur un projet consiste à utiliser un outil de gestion de versions. Cela nous a paru évident pour deux raisons. La première étant que pour un projet en groupe de cette envergure, il est indispensable d'utiliser un outil qui gère le code de tout le monde. Il n'est pas envisageable de devoir s'échanger les fichiers par mail ou par support physique (clef USB par exemple). La seconde étant qu'il est beaucoup plus facile de suivre un projet avec un outil de gestion de versions. Chaque commit (envoie) permet de voir l'avancement du projet et savoir si une fonctionnalité est livrée ou non. Nous avons choisi d'utiliser un service web, en complément du logiciel de gestion, qui propose des services d'hébergement. Notre choix s'est tourné naturellement vers Github. Il accepte plusieurs logiciels de gestion de versions et, de ce fait,

chacun à ainsi eu le choix du logiciel de gestion de versions qu'il allait utiliser. Deux logiciels sont ressortis : Git et Svn.

Pour finir nous avons utilisé GoogleDrive pour le partage de documents autres que le code source et aussi pour la rédaction de documents. Cet outil est complet et permet de réaliser très facilement des partages ainsi que de stocker des documents.

Après avoir choisi et installé ces outils nous avons dû former ceux qui ne les connaissaient pas, pour commencer sereinement le projet. Les personnes qui connaissaient bien un outil ont réalisé des tuteurs sur GoogleDrive qui servaient de références en cas de besoin de la part d'un autre utilisateur. Une fois ces étapes passées nous avons pris du temps pour organiser le planning du projet.

Mise en place du planning

Nous avons réalisé un diagramme de GANTT pour estimer le temps necéssaire à la réalisation de chaque tâches.

Diagramme de Gantt

TPÉ	Nom	Travail	Semaine 47, 2013							Semaine 48, 2013							Semaine 49, 2013							Semaine 50, 2013							Semaine 51, 2013										
			18	19	20	21	22	23	24	25	26	27	28	29	30	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
1	Création : Client/Serveur	15d																																							
2	Interface User	10d																																							
3	Ant	2d																																							
4	Test Unit	5d																																							
5	Test Application 1	5d																																							
6	Redaction de documents	10d																																							
7	Livraison V1	1d																																							
8	Amelioration V1	5d																																							

Nous avons organisé le projet en suivant un approche Kanban / Scrum. Les tâches utilisées sont celles du diagramme de GANTT.

Nous avons fait une première réunion pour définir les limites du projets, les tâches à réaliser et le temps approximatif pour les réaliser. A partir de là nous avons commencé la tâche principale qui est la réalisation du t'chat décentralisé, qui est découpé en plusieurs sous tâches, ainsi que de nombreuses tâches annexes telle que la réalisation de documents par exemple. La conclusion de cette réunion s'est traduite par la réalisation d'un planning Gantt prévisionnel. Ensuite, nous avons effectué des réunions chaque lundi pour vérifier l'avancement du projet et voir si certaines tâches avaient besoin de renfort pour être terminées dans le temps imparti.

Le projet étant étalé sur plusieurs mois, en parallèle des cours et non pas une optique de projet à temps complet, il est difficile de mettre en place une véritable méthode Kanban ou Scrum. Néanmoins la découpe du projet en tâches bien spécifiques et l'ensemble des réunions régulières ont permis de rendre le projet solide.

Outils adoptés pour le suivi et le pilotage

Le but principal était de livrer une version 1.0 stable le plus rapidement possible pour pouvoir ajouter différentes améliorations par la suite. Cette version comprend les spécifications demandées par le sujet. C'est à dire le t'chat décentralisé avec les contraintes données (LogIn/MdP, communication Serveur/ Client et Client/Client...) et les tests unitaires.

Les versions 1.x avaient pour but d'améliorer la version 1.0 stable, notamment en corrigeant les bugs, en améliorant l'interface graphique, en remaniant le code et en ajoutant la conversation de groupe. Le fil rouge de ce projet était de réaliser des documents relatifs au génie logiciel, à la gestion de projet, à la partie réseau (diagrammes de séquence, automates), à la javadoc ainsi que le manuel d'utilisation.

Pour le suivi et le pilotage, GitHub nous offre de nombreux outils comme par exemple la possibilité de commenter les commits. Cela nous a été très utile pour comprendre le code de chacun. De plus les réunions hebdomadaires ont permises de voir l'avancement des tâches avec une vision globale.