Projet de Développement Web 2 : Édition Collaborative de Documents

1 Introduction

Le projet de Développement Web 2 consiste à programmer en Java et en équipe de 4 étudiant-e-s, une plateforme d'édition collaborative de documents permettant aux utilisateurs d'interagir en direct autour d'un document partagé. Les spécifications de l'application ne sont pas imposées de manière très précise : vous êtes relativement libres dans le choix de l'architecture de votre application et des fonctionnalités à implémenter.

L'objectif du projet est la mise en œuvre des technologies étudiées en Développement Web 2 ce semestre. Les différentes parties de votre projet (réalisation, rapport, soutenance) seront évaluées en fonction de la maîtrise que vous montrerez de chacune des technologies. Il faudra donc prendre soin de bien mettre en avant l'utilisation que vous en faites dans votre application.

2 Édition Collaborative de Documents

2.1 Fonctionnalités

Le projet consiste à programmer en Java une plateforme permettant aux utilisateurs d'*interagir* en direct autour d'un *document partagé* :

- Un document partagé : à vous de choisir le type de document que vous voulez que vos utilisateurs manipulent. L'important est que ce thème ne soit pas limitant pour le développement du projet et se prête donc bien à l'édition collaborative en temps réel. Voici quelques exemple de type de documents, avec des applications existantes (collaboratives ou non) :
  - document texte riche (word, google docs)
  - tableau de formules (excel, google docs, ethercalc[1])
  - document texte (etherpad[2]), code source[3]
  - emploi du temps, planning (ENT, doodle[4])
  - dessins vectoriels (inkscape[5])
  - présentation (powerpoint, google docs)
  - dépenses de groupes pour un événement, comptes entre amis[6]
  - carte mentale (freemind, freeplane[7])
  - carte de concepts, graphe, diagramme UML (bubbl.us[8])
  - niveau d'un jeu (UnrealEd, GTKradiant[9], SuperTux Editor[10])
- Interagir : c'est-à-dire visualiser un document, le modifier, et *communiquer* avec les autres utilisateurs de ce document ;

[1] Ethercalc : https://ethercalc.org/
[2] Etherpad : http://etherpad.org/ et un exemple http://free.primarypad.com/p/fZE3cW5fGu
[3] programmation collaborative, exemple http://sharejs.org/code.html#DREqxvBzKg
[4] Doodle : http://doodle.com/
[5] Inkscape : https://inkscape.org/fr/
[6] http://www.bonscomptesentreamis.com/
[7] Freeplane : http://www.freeplane.org/
[8] bubbl.us https://bubbl.us/
[9] GTKradiant http://icculus.org/gtkradiant/
[10] SuperTux Editor http://supertux.lethargik.org/wiki/Editor_Guide

- Communiquer : discuter via une messagerie, un forum, un chat, etc. ;
- Et bien sûr, pour les documents confidentiels, cela nécessite que les utilisateurs soient authentifiés lors de leur connexion à la plateforme (connexion, pseudo, mot de passe, gestion des sessions, déconnexion).

Outre l'édition du document et la communication, on peut imaginer toutes sortes de fonctionnalités, comme par exemple :

- Ajout de commentaires persistants sur le document partagé ;
- Fonctionnalité de *undo* / défaire / Ctrl+Z ;
- Gestion de l'historique du document partagé (revenir à une ancienne version) ;
- Profils utilisateurs avancés : groupes, liens d'amitiés / de contacts, fil d'actualités / « murs », messagerie, documents favoris, etc ;
- Public / privé : associer à chaque document un degré de visibilité qui permet de restreindre son accès aux amis / famille / collègues (mode privé), à tout le monde (mode public) ou à toute personne qui connaît l'identifiant du document (comme etherpad, etc.) ;
- Protection d'un document « public » par mot de passe ;
- Utilisateurs en mode lecture seule (certains utilisateurs étant seulement « spectateurs ») ;
- Documents composites contenant des sous-documents de différents types ;
- Édition hors-ligne : édition locale sans le serveur, puis synchronisation à la connexion (difficile) ;
- Etc.

Il n'est évidemment pas demandé de développer chacune de ces fonctionnalités, mais seulement quelques-unes. La liste est loin d'être exhaustive : vous pouvez proposer d'autres fonctionnalités qui vous semblent intéressantes. Vous pouvez partir à la pêche aux idées et laisser parler votre imagination !

2.2 Architecture : serveur, client lourd, client léger

Votre projet doit être développé suivant un modèle client-serveur dans lequel le serveur aura pour objectif de :

- permettre la création et la gestion de documents ;
- mettre en relation les utilisateurs de la plateforme ;
- héberger les documents, leurs historiques, etc ;
- gérer la communication entre les utilisateurs.

Le client aura pour fonctionnalité de permettre aux utilisateurs de se connecter au serveur et d'accéder à des documents partagés. Pour faciliter la vie des utilisateurs, vous développerez un client lourd et un client léger :

- Client lourd : une application Java que l'utilisateur doit télécharger et exécuter sur sa machine. L'interface graphique « lourde », bénéficiant de la puissance de *Swing*, permet d'obtenir la meilleure ergonomie ;
- Client léger : votre application doit également permettre à un utilisateur de se connecter via un simple navigateur Web, éventuellement sur un appareil mobile (par exemple un smartphone). Dans ce cas-là, les technologies mises en œuvre sont : HTML, CSS, Servlets, JSP.

Attention : si le design et l'ergonomie peuvent différer entre le client lourd et le client léger, les fonctionnalités accessibles doivent être identiques.

3 Technologies à mettre en œuvre

L'objectif du projet est la mise en œuvre des technologies étudiées en Développement Web 2 ce semestre. Vous devez donc montrer votre maîtrise de ***chacune d'entre elles***. Les spécifications du projet ne détaillent pas la manière dont vous devez les utiliser : nous vous donnons quelques pistes / indications dans cette section.

3.1 Modélisation UML

Dans un premier temps, il va falloir réaliser une analyse exhaustive des éléments de votre application : identifier les différents acteurs qui interviennent dans votre application, identifier les différents rôles que ces acteurs peuvent endosser et les fonctionnalités auxquelles ils peuvent avoir accès.

Une fois cette tâche réalisée, il va vous falloir définir les *cas d'utilisation* de votre application. Pour cela, appuyez-vous sur un diagramme de cas d'utilisation. Ce diagramme aura pour principal objectif de représenter les liens entre les acteurs de votre système et les fonctionnalités offertes par l'application. Vous compléterez par la suite ce diagramme avec une description textuelle de deux ou trois scénarios concrets.

Aussi, afin de compléter votre modélisation, il vous est demandé de réaliser le *diagramme de classes* afin de décrire les objets-clés de votre application. Vous veillerez à spécifier pour chaque classe ses attributs, ses méthodes et le type de relations que peut avoir la classe avec d'autres classes.

Enfin, afin d'illustrer les scénarios que vous avez préalablement définis, il vous est demandé de réaliser les *diagrammes de séquence* correspondant à chaque scénario.

3.2 Réseau

Pour la partie réseau, il vous est demandé de mettre en place une architecture 3-tiers (ou n-tiers) à travers laquelle vous procéderez à un découpage logique de votre application. Vous mettrez donc en œuvre le schéma suivant :

> Client <-> Serveur d'application <-> Serveur de données.

Le serveur doit donc centraliser la majorité des échanges d'informations entre les utilisateurs. Toutefois, un certain type d'informations (par exemple les messages privés) circuleront d'utilisateur à utilisateur, sans passer par le serveur. Il s'agit donc d'une architecture pair-à-pair :

> Client <-> Client

3.2.1 Le client

Dans un premier temps, faites une version basique de votre plateforme, qui ne comporte que deux utilisateurs connectés en utilisant des sockets TCP. Faites en sorte que le premier se lance en écoute (avec un numéro de port connu) via une socket d'écoute comme vu en TD et sans essayer de se connecter à l'application d'un autre utilisateur. Ensuite, lancez l'application du deuxième utilisateur qui aura pour mission de se connecter à celle du premier utilisateur. On obtient ainsi une situation simple, avec laquelle vous pouvez commencer à faire vivre votre plateforme (très réduite pour le moment). En traitement des commandes envoyées, faites déjà en sorte que les deux applications puissent s'envoyer des informations (par exemple des messages privés) et que chacune affiche l'information envoyée par l'interlocuteur. Vous devriez ainsi obtenir une version simpliste, partielle mais fonctionnelle avec deux applications Java.

3.2.2 Le serveur d'application

Une fois l'implémentation de la communication pair-à-pair terminée, on développera une application Java pour jouer le rôle du serveur, qui aura pour tâche de centraliser les informations envoyées par les utilisateurs (exceptées celles envoyées en pair-à-pair !). Cela permet, entre autres, à un utilisateur de récupérer les informations que les autres membres lui ont envoyé alors qu'il était hors-ligne. Il suffit d'ajouter au serveur une socket d'écoute à laquelle se connecte le client. Le serveur envoie alors au client toutes les informations reçues en son absence. Ensuite, tant que le client est connecté, le serveur lui fournit directement les nouvelles informations.

3.2.3 Le serveur de données

Voir la section suivante.

3.3 Bases de données

Les documents gérés par votre plateforme devront être stockés dans une base de données MySQL. La structure des données devra être formalisée préalablement à l'aide d'un modèle Entité-Association.

3.4 XML

Lors d'échanges de données entre deux utilisateurs ou d'un utilisateur au serveur, vous devez utiliser le format XML. Le client et le serveur doivent donc disposer de la / des DTD et ils doivent pouvoir analyser (lecture) et générer (écriture) les fichiers XML respectant cette/ces DTD, pour importer/exporter des informations.

3.5 Swing

L'interface graphique du client « lourd » sera réalisée à l'aide de *Swing*.

3.6 Servlets, JSP et Applets

Le client « léger » sera basé sur les technologies des Servlets, des JSP et éventuellement des Applets. Il vous faudra proposer une architecture pertinente combinant des servlets et des JSP. Vous pouvez aussi imaginer une fonctionnalité pour laquelle l'utilisation d'une applet peut s'avérer nécessaire et judicieuse.

3.7 Patrons de conception (Design Patterns)

Au cours de la conception et de l'implémentation de votre application, une attention particulière devrait être accordée au respect des patrons de conceptions vus en cours. Il s'agit également de rendre compte dans le rapport final des problèmes de conception rencontrés et des patrons utilisés pour y remédier. Par ailleurs, il faut veiller à ce que votre architecture respecte rigoureusement le patron de conception modèle-vue-contrôleur (MVC) étudié dans le cours. Tout projet ne se conformant pas à ce patron sera pénalisé car considéré « hors sujet ».

3.8 Cadre d'Application (Framework)

La mise en œuvre d'une « bonne » architecture MVC demeure une tâche complexe. C'est pourquoi, de nombreux cadres d'application ont été proposés pour faciliter le développement d'applications Web Java respectant le modèle MVC. Par conséquence, il est impératif de développer votre application en vous appuyant sur un des cadres vus en cours (Struts, Spring ou JSF). Il faut également justifier votre choix quant au cadre d'application choisi et si nécessaire rendre compte des limites de son utilisation.

4 Organisation, rendu et soutenance

Le projet doit être conçu et développé en groupe de 4 étudiants (dont la composition doit être envoyée par mail au plus tard le 21 mars). Néanmoins, la notation pourra être individualisée. Le projet doit être réalisé avec l'aide d'un outil de gestion de version (SVN, Git, etc.). Si vous utilisez Git, adressez-vous à vos enseignants pour la création d'un groupe Github privé dédié à ce projet.

Vous devez déposer un document de conception intermédiaire sur le cours en ligne « Développement Web 2 » (espace « Travaux ») au plus tard le 14 avril 2017 à 23h59 (sous peine d'un point de pénalité par jour de retard). Ce document synthétique (quelques pages : allez à l'essentiel), au format PDF, contiendra :

- Une description générale de l'application (type de documents considéré, fonctionnalités envisagées, etc.) ;
- Une première version (simplifiée) des diagrammes de modélisation UML, du plan du site, du/des template(s) de présentation, et autres schémas entité-association ;
- Une indication de l'utilisation prévue des différentes technologies (de ***chacune d'entre elles***) ;

- Le rôle de chaque membre de l'équipe ;
- La désignation d'un chef d'équipe.

Vous devez déposer votre projet final sur le cours en ligne « Développement Web 2 » (espace « Travaux ») au plus tard le 2 mai 2017 à 23h59 (sous peine d'un point de pénalité par jour de retard), sous la forme de 3 fichiers :

- Le rapport au format PDF.
- L'application complète sous la forme de fichiers *.jar* (client) et *.war* (serveur), incluant le code source complet, un fichier de sauvegarde des ressources et des données, et les éventuelles ressources graphiques utilisées.
- La documentation *Javadoc* sous la forme de fichiers HTML compressés dans un fichier zip.

Le rapport doit mettre en valeur votre travail. Il contiendra notamment :

- Une introduction qui décrit succinctement le but du projet et le plan du rapport ;
- Un rapport de conception du projet comprenant diagrammes de modélisation UML, plan du site, template(s), et autres schémas entité-association ;
- Une description des fonctionnalités de l'application ;
- Les technologies de Développement Web 2 mises en œuvre (**important !** Il s'agit de bien mettre en avant la manière dont elles sont exploitées dans votre application) ;
- Une description précise de l'origine de chaque partie du code : code 100% personnel, librairies de la plateforme Java, librairies trouvées sur le Web, fragments de code trouvés sur Internet ou réalisés en collaboration avec un autre groupe, etc. ;
- Une estimation du temps consacré au projet, ainsi que de l'organisation de votre travail (répartition des tâches au sein de l'équipe, etc.) ;
- Une liste complète des ressources utilisées (outils, livres, tutoriels, sites Web),
- Une conclusion qui pourra comporter un bilan critique de votre travail (résultats, points forts / faibles, etc.), ainsi que vos impressions sur les aspects de Développement Web 2 qui vous ont particulièrement intéressé (ou non), que vous avez trouvé plus difficiles, etc.

Votre code doit :

- utiliser à bon escient les technologies vues en Développement Web 2 ;
- être compact, lisible, commenté ;
- avoir été soigneusement testé ;
- et enfin, pouvoir être compilé en ligne de commande sur l'environnement utilisé en TP.

Chaque équipe doit réaliser son propre projet. Vous pouvez bien évidemment utiliser des librairies externes (vous le devez !), mais il est impératif de préciser soigneusement les parties de code que vous avez développé entièrement et celles que vous réutilisez.

La présentation finale du projet se fera le 5 mai après-midi (à partir de 13h) en mode soutenance (à raison de 30 minutes par groupe).