UNIVERSITÉ LIBRE DE BRUXELLES

GÉNIE LOGICIEL ET GESTION DE PROJET

Itération 1 - Groupe 4

DESCRIPTION ET JUSTIFICATION DES CHOIX DE PROGRAMMATION

Étudiants :

Mourad AKANDOUCH
Nicolas FERON
Mounir HAFIF
Issam HAJJI
Sacha MEDAER
Benjamin MESTREZ
Allan MURANOVIC
Ibrahim TOURE

Titulaire :
Frédéric PLUQUET
Assistants :
Jacopo DE STEFANI
Luciano PORRETTA
Fabio SCIAMANNINI

13 mars 2017

Table des matières

1 Introduction

Ce document a pour but de décrire l'itération numéro 1. Le premier chapitre explique les fonctionnalités qui ont été implémentées. Le deuxième chapitre, quant à lui, présente les différents choix de programmation utilisés. Enfin, le dernier chapitre décrit les difficultés rencontrées lors de cette première partie et comment celles-ci ont été résolues.

2 Fonctionnalités

Cette itération a implémenté l'histoire numéro 5 pour 35 points. Celle-ci permet de construire les bases pour la future application, notamment créer un utilisateur, login et mot de passe. Voici listé ci-dessous les différentes fonctionnalités implémentées :

— création d'un compte
— vérification des conditions générales d'utilisation
— confirmation du compte par email
— connection à la plateforme
— changememt d'informations personnelles (username et adresse email)
— confirmation de validité par email si changement d'informations personnelles
— captcha pour création d'un nouveau compte
— déconnection de la platforme

3 Choix de programmation

Différents framework ou librairies ont été utilisées lors de cette première partie. Cette section vise a présenter ces différents outils. Il est à noter que l'utilisation de librairies a été réduite au minimum.
Le logiciel est écrit en *Java* et les communications suivent le protocol *RESTFul*. Le code suit également le pattern MVC (Model-View-Controller). La convention d'écriture adopté pour le projet est le *Camel Case*.
L'itération 1 est composée de trois grands éléments ; la base de données, l'interface client serveur et l'interface graphique.

3.1 Base de données

SQLite a été choisie pour la gestion de la base de données. Cet outil est léger, avec une grande portabilité et parfaitement adapté pour les plateformes client-serveur. De plus, un fichier *.db* permet de partager la même base de données entre les différents programmeurs.
La liaison entre le principale langage utilisé pour l'application, *Java*, et la base de données est basé sur le *design pattern* DAO (data access object). Ceci signifie que le serveur ne fait pas directement de requêtes à la base de données. Un niveau d'abstraction supplémentaire est ajouté entre le serveur et celle-ci. À chaque table de la base de données correspond

une classe qui permet d'y accéder depuis le serveur. La connection et les accès à la base de données se fait grâce à la librairie *JDBC*.

3.2 Interface client-serveur

L'interface client-serveur a utilisé l'API fourni par *Java*. Pour suivre le protocol *RESTFul*, le logiciel utilise des *token*. En effet, l'état de l'utilisateur n'est pas sauvegardé. Pour identifier et authentifier les différents utilisateurs, un *accessToken* est attribué à chacun lors de la connection à la platforme. Ce *token* est utilisé pour des requêtes privilégiées effectuées au serveur. Celui-ci a une durée de vie d'une heure. Un *refreshToken* est également attribué dès la connection. Lorsque cette heure est écoulée et que le client désire faire une nouvelle requête, le *refreshToken* est utilisé. Ce dernier va créer un nouveau *accessToken* avec une durée de vie de 1h. Ce système est basé sur le protocol *OAuth 2*.
L'échange de données est assuré par la librairie *Json*. Ce format d'échange de données est léger, facile d'utilisation et très répandu.
Pour l'envoie de mail, l'extension *mail* de la librairie *Javax* a été utilisée.

3.3 Interface graphique

L'interface graphique a été créée grâce au logiciel *Java Fx*. Des CSS (Cascading Style Sheets) ont été utilisées également pour personnaliser le style de la platforme.