Développement informatique avancé
Orienté Applications
Enoncé du projet

Virginie Van den Schrieck, Arnaud Dewulf, Louis Van Dormael

28 septembre 2018

1 Choix du sujet

Le thème du projet est laissé au choix des étudiants, pourvu qu'il réponde aux contraintes suivantes :

- Il s'agit soit d'une application utilitaire, soit d'un jeu (autres possibilités à discuter avec l'équipe enseignante)
- L'application doit respecter l'architecture MVC avec deux interfaces utilisateurs (une interface console et une interface graphique)
- L'application doit comporter une communication Socket (les sockets seront vus au cours) OU une interaction JDBC avec une base de données (attention, les étudiants devront alors découvrir JDBC par eux-même).
- L'application doit utiliser au moins une structure de données du framework Java Collection (HashMap, List, ...).

Avant toute implémentation, le sujet doit être soumis aux enseignants pour validation.

2 Modalités pratiques

2.1 Groupes

Les groupes seront composés de **trois** étudiants. Chaque groupe devra créer un **compte sur Github**, sur lequel sera hébergé le code-source de l'application. Le groupe prendra soin de créer une **page Wiki** sur leur site Github, qui reprendra les informations sur le projet ainsi que les délivrables demandés au fur et à mesure (voir plus loin).

2.2 Etapes du projet

La réalisation du projet se fera en respectant les étapes ci-dessous :

1. Formation des groupes, choix du sujet et réalisation du cahier des charges (descriptif),
2. Diagramme UML de la partie *model* de l'application,
3. Création des squelettes des classes du package *model*, spécifications et tests unitaires,
4. Implémentation du package *model* et remise d'une classe complète par **chaque étudiant** du groupe,
5. Interactions utilisateur : Implémentation d'une vue Console et du contrôleur associé et **démo**
6. Interactions utilisateur : Implémentation de la vue GUI et du contrôleur associé
7. Ajout de la couche réseau ou DB

3 Critères d'évaluation

Le projet sera évalué sur base des critères suivants :

3.1 Analyse

— Le diagramme UML a été fait et modélise correctèment l'application (partie modèle uniquement)
— Les classes du modèle ont été correctement spécifiées (javadoc)

3.2 Architecture de l'application

— L'application est structurée selon le pattern MVC
— Il existe deux vues pour l'application : Console et GUI
— L'application utilise des sockets ou une base de données
— L'application utilise des structures de données de l'API Collection, et le choix des structures de données est pertinent et justifié
— Des threads sont utilisés là où c'est nécessaire, et il n'y a pas de deadlocks
— Les interfaces utilisateurs sont ergonomiques et agréables à utiliser

3.3 Implémentation de l'application

— L'application fonctionne sans bug
— Les cas problématiques sont gérés à l'aide d'exceptions ad-hoc
— Les deux interfaces de l'application (console et GUI) fonctionnent comme attendu

— L'application est conforme au cahier des charges
— L'application est conforme au diagramme UML
— La communication réseau/BDD est fonctionnelle
— Vous avez écrit vous-mêmes tout le code source de votre application. L'objectif de ce projet est de montrer que vous êtes apte à créer entièrement une application et non d'utiliser au mieux des outils ou frameworks créés par d'autres. Si vous désirez utiliser ponctuellement du code provenant d'ailleurs pour votre programme, parlez-en au préalable à vos enseignants. Toute utilisation de code d'autrui non référencée est absolument interdite et sera traitée comme du plagiat.

3.4 Gestion du projet / développement en équipe

— Le code a été versionné sur Github et des commits fréquents ont été effectués
— Chaque étudiant a participé de manière équitable au projet, et cela se voit sur le repository Github
— Les étudiants ont suivi une démarche TDD
— La partie modèle de l'application est correctement couverte par des tests unitaires
— Le code est de qualité
— Une convention de codage a été définie et respectée
— Les échéances intermédiaires ont été respectées

3.5 Rapport et Wiki

— Le Wiki Github a été correctement rempli et mis à jour tout au long du projet
— Le rapport final a été rendu conformément à l'échéance
— La forme du rapport est correcte (orthographe, style, fil rouge, ...)
— Le fond du rapport est conforme à ce qui est demandé (cfr description du délivrable ci-dessous)

3.6 Démo/défense

— Tous les étudiants du groupe sont présents à la défense
— La démo a été préparée
— La démo montre que l'application est fonctionnelle
— Les étudiants sont capables de justifier leurs choix d'implémentation à travers les réponses aux questions
— Les étudiants font preuve d'une bonne dynamique de groupe (collaboration, entraide, implication, organisation)
— Chaque étudiant démontre qu'il a contribué de manière significative au code du projet. Le cas échéant, un étudiant ne démontrant pas une implication suffisante sera sanctionné au niveau de sa note de projet.

4 Calendrier et délivrables

4.1 Choix du sujet

Echéance : voir Campus Virtuel

A remettre

Document PDF sur le Campus Virtuel comprenant :
— La composition du groupe
— Une description du cahier des charges du projet (descriptif client)
— L'URL du repository Github avec page Wiki à jour

4.2 Diagramme de classe UML

Echéance : voir Campus Virtuel

A remettre :

Le diagramme UML du modèle de l'application au format PDF, sur le Campus Virtuel ET sur le Wiki Github

4.3 Implémentation du modèle

Echéance : voir Campus Virtuel

A remettre :

Chaque étudiant du groupe soumet une classe complète du package `model`, dûment spécifiée et testée, sur le Campus Virtuel. Le repository Github doit être à jour avec le code correspondant.

4.4 Démo de la vue Console

Echéance 2TL1 : séance TP voir Campus Virtuel

Echéance 2TL2 : séance TP voir Campus Virtuel

A préparer :

Les étudiants font une démo des interactions possibles avec le modèle depuis une interface console (ligne de commande).

4.5 Remise du projet

Echéance : voir Campus Virtuel

A remettre :

— Page Wiki du Github à jour avec :
  — Composition du groupe

    - Cahier des charges/descriptif
    - Version finale du diagramme UML du modèle
    - Mode d'emploi pour installer et utiliser l'application
    - Pointeur vers les délivrables intermédiaires et finaux
- Sur le Campus Virtuel + copie papier à remettre au professeur lors de la démo, un rapport comprenant :
    - Le cahier des charges
    - Le diagramme UML et son explication éventuelle
    - Les choix d'implémentation effectués
    - Les difficultés rencontrées
    - Les pistes d'amélioration éventuelles
    - Une conclusion individuelle de chaque membre du groupe, détaillant ses apports et son vécu personnel lors de la réalisation du projet
- Sur Github : Le code source finalisé. La branche Master ne peut pas avoir de commits ultérieurs à la date de remise.

4.6 Défense finale

La défense du projet aura lieu durant la session de janvier. Elle consiste en une démo de l'application sur machine (pas de projection prévue).

Les étudiants apportent une version imprimée du rapport à cette occasion (noir et blanc, agrafé, pas de reliure, de papier glacé ou de couverture plastique).

Le code source et le rapport doivent être identiques à ceux remis lors de l'échéance de fin de semestre (Campus Virtuel et commits Github faisant foi). Il ne sert donc à rien de continuer le développement du projet pendant le blocus de Noël, qui doit être consacré exclusivement à la révision de vos cours.