Manuel de reprise de code du projet ScanExam

Membres :

BEUREL Luca
CARUANA Romain
COCHET Julien
DANLOS Benjamin
DEGAS Antoine
GHOUTI TERKI Rida
GIRAUDET Théo
GUIBERT Thomas
LALANDE MARCHAND Arthur
LELOUP Alexis
LOCKE Stefan
LUMBROSO Marius
MA Qian
PAYS Matthieu

Supervisé par :

DERRIEN Steven

Table des matières

I - Introduction

Ce manuel doit permettre à toute personne le souhaitant de reprendre le projet ScanExam en l'état. Le projet est accessible sur Github à cette adresse[1].

Il comporte de prime abord les informations nécessaires concernant l'installation de l'environnement de développement du projet, des informations sur les technologies utilisées, ainsi que les conventions utilisées lors du développement. De plus, ce manuel comporte également une vue d'ensemble de l'application, qui explique son découpage métier, et permet de mieux comprendre certains choix réalisés. Pour finir, un tableau récapitulatif de fonctionnalités pas ou partiellement implémentées est proposé comme piste d'évolution.

II - Environnement de développement

1 - Configuration d'Eclipse

ScanExam est un projet Java fonctionnant avec Eclipse IDE[2]. Ce projet utilise les technologies Xtend et XCore. Eclipse IDE étant au moment de l'écriture de ce rapport, le seul IDE à réellement supporter ces deux technologies, son utilisation est donc nécessaire au bon développement du projet si les développeurs ne souhaitent pas gérer toute cette partie par ligne de commandes. L'utilisation de ceux-ci dépend des extensions destinées à Eclipse qui sont disponibles sur son marketplace, pour cela rendez vous dans l'onglet "**Help → Install new software**" puis cherchez "**EMF - Eclipse Modeling Framework Xcore SDK**".

Il est toutefois possible de prendre le paquet de distribution d'Eclipse "Eclipse IDE for Java and DSL Developers" qui inclut les extensions en question[3].

2 - Xtend et Xcore

Xtend est un dialecte de Java. Celui-ci agit donc comme étant une surcouche du langage Java. Ce langage apporte des fonctionnalités supplémentaires telles que la possibilité d'étendre des classes par l'ajout de fonctions, ou encore par le biais d'un meilleur support des expressions lambda. Il est alors transpilé vers du code Java par le biais d'Eclipse IDE. Une documentation complète sur son fonctionnement est disponible sur le site d'Eclipse[4].

Le choix de ce langage provient de la version originale de ScanExam utilisant Xtend comme support. La demande du client était donc de continuer d'utiliser ce dialecte du fait de

[1] https://github.com/ScanExam/ScanExam
[2] https://www.eclipse.org/ide/
[3] https://www.eclipse.org/downloads/packages/release/2021-03/r/eclipse-ide-java-and-dsl-developers
[4] https://www.eclipse.org/xtend/documentation/index.html

ses nombreux avantages. Il a ses racines dans le langage de programmation Java mais se concentre sur une syntaxe plus concise et quelques fonctionnalités supplémentaires. En complément de Xtend, nous utilisons également le plugin Xcore afin de modéliser les données de notre application.

Xcore est également une surcouche de Java qui permet de modéliser les données persistantes, et de sérialiser bien plus simplement les données au format XMI (dérivé du XML). Nous utilisons notamment Xcore pour représenter toutes les méta-données relatives à un examen (le sujet maître, et les corrections d'un sujet). Ainsi, l'ouverture d'un sujet et d'une correction passent par la désérialisation des fichiers XMI sauvegardés préalablement sur le disque. Ces services font donc appel à la classe "**TemplatesIo**" pour manipuler ces fichiers.

3 - JDK Java

C'est la bibliothèque JavaFX qui a été choisie pour modéliser la partie visuelle de l'application. L'entièreté du code Xtend relatif à JavaFX se situe donc dans la partie Vue qui sera présentée plus bas dans ce document. Ce code est totalement dépendant des fichiers FXML se trouvant dans les ressources de l'application. En effet, la structure des composants visuels JavaFX est organisée à l'aide du langage FXML, un dérivé du langage XML adapté à la bibliothèque. Pour avoir une compatibilité totale de JavaFX avec Eclipse IDE, il est nécessaire d'installer le plugin "e(fx)clipse" disponible sur le marketplace.

Pour avoir une version JavaFX présente avec toutes les distributions de Java, la version 11 de Java (LTS actuelle) a été choisie comme version de développement, permettant ainsi l'import de la bibliothèque par Maven.

4 - Maven

Pour faciliter la gestion des dépendances mais aussi le build du projet ScanExam, nous sommes passés par l'outil Maven. Celui-ci nous permet donc par de simples commandes de :

- Compiler : "mvn compile"
- Exécuter le programme : "mvn compile exec:java"
- Packager en jar (jar avec toutes les dépendances incluses à l'intérieur) : "mvn compile assembly:single"

Une liste plus exhaustive des commandes utiles au projet est disponible dans le **README.txt** du dépôt Git.

Maven ne permet néanmoins pas dans cette version de ScanExam de gérer la transpilation de Xtend vers Java. En effet, le plugin Maven supportant Xtend et Xcore semble ignorer le paramètre spécifiant le version cible de la transpilation vers Java 11[5] et ne transpile qu'en Java 5. Malgré l'aspect rétroactif du code Java, le code obtenu de la

[5] https://stackoverflow.com/questions/65963841/xtend-with-maven-does-not-preserved-semantic

transpilation des sources Xtend n'est sémantiquement pas similaire au code original, résultant en des erreurs de compilation du code Java. Il est possible de reproduire l'erreur en compilant le projet minimal disponible ici[6]. Une liste exhaustive des dépendances java est spécifiée dans le fichier **pom.xml** à la racine du projet.

III - Arborescence du projet

src/main/java : Contient des fichiers annexes. Certaines fonctionnalités Java n'étant pas disponibles en Xtend, ce dossier est réservé au code source Java spécifique.

src/main/xtend : Contient les sources effectives du projet.

src/main/xtend-gen : Contient les classes Java générées après transpilation des classes Xtend. Ces fichiers ne doivent pas être source de modifications. Ils sont générés automatiquement lors du build par Eclipse.

src/main/ressources : Contient des métadonnées et des fichiers relatifs à la configuration de l'application. (Fichiers de langages, Polices, Fichiers FXML).

src/test/resources : Fichiers ressources pour réaliser des tests.

model : Les données de ScanExam en Xcore (ce sont les données qui seront sérialisées sur le disque au format XMI).

src/gen : Contient la transpilation des fichiers Xcore en Java.

[6] https://github.com/theogiraudet/XtendMinimalProject

IV - Architecture

ScanExam a été conçu suivant le pattern MVC[7]. Le principe de ce pattern design est de décomposer l'application en trois blocs métier :

Le Modèle (M), qui représente les données de l'application. Dans ce projet, les données sont modélisées au sein de fichier Ccore, dans le dossier "**model**" à la racine.

La Vue (V), qui représente la partie visuelle de l'application. Il s'agit donc de ce qui est visible par l'utilisateur (l'interface graphique). La bibliothèque graphique utilisée par ScanExam et JavaFX, qui supporte nativement le pattern MVC. Ainsi, la vue est séparée dans des fichiers FXML (dérivé du XML) se trouvant dans le dossier "**src/main/resources/viewsRessources**".

Les Contrôleurs (C), ceux-ci servent d'intermédiaires entre la couche Modèle et la couche Vue, ils s'occupent de récupérer les entrées utilisateurs depuis les composants de la Vue, mais aussi de modifier les données (le Modèle) en fonction de l'état de la Vue ou de les lui passer pour affichage. Les contrôleurs se trouvent dans le répertoire "**src/main/xtend/view/fx**".

Ce pattern représente donc la structure de ScanExam. À cela s'ajoutent les Services. Il s'agit de deux interfaces servant de façades aux données, facilitant leur accès et leur modification par les Contrôleurs. Chaque service est spécialisé dans l'un des services que propose ScanExam, c'est-à-dire l'édition du sujet maître (EditionService), et la correction des copies (GraduationService).

Pour résumer, l'architecture globale peut-être représentée par la figure 1.

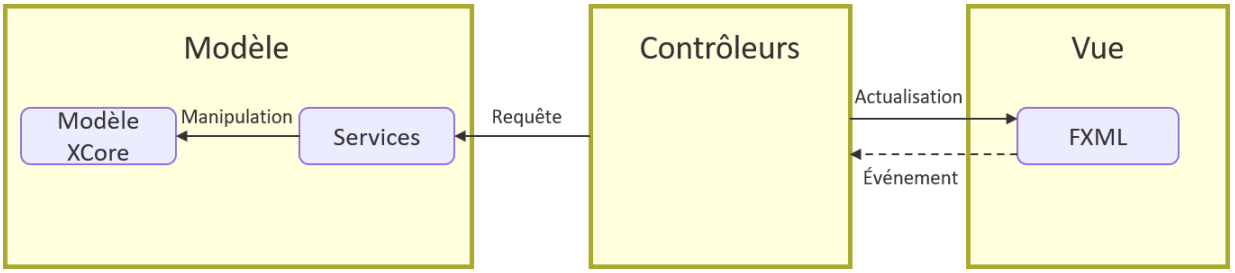

***Figure 1**. Schéma récapitulatif de l'architecture de ScanExam*

De manière générale, il est important de passer par le service lorsqu'une fonctionnalité ajoutée au programme touche de près ou de loin aux données persistantes (Xcore), et ne concerne pas uniquement la vue. Cela permet d'une part de centraliser les accès à celle-ci mais aussi d'éviter les problèmes de synchronisation lors de l'obtention des données à travers les services par les contrôleurs.

[7] https://fr.wikipedia.org/wiki/Mod%C3%A8le-vue-contr%C3%B4leur

V - Pistes d’amélioration

Par manque de temps, certaines fonctionnalités plus secondaires n’ont pas pu voir le jour lors du développement. En voici une liste non exhaustive :

	Partiellement implémentée	Non implémentée
Récapitulatif des notes		✔
Menu et raccourcis claviers		✔
Voir corrigé en parallèle		✔
Ajouter des annotations	✔	
Filtres	✔	
Annotations manuscrites	✔	
Prévoir/prédire les items de la grille de notation à partir des résultats précédents		✔
Gestion de pages/copies mal scannées		✔
Statistiques des questions en temps réel		✔
Aperçu du pourcentage de copies corrigées		✔
Gestion de bonus/malus	✔	

***Figure 2**. Tableau de propositions de fonctionnalités*