FOTIE Koffi Dramane
PEZZALI Pierre-Maximilien
REGNIER Fabien
SONG Delphine
ZHANG Alain

GI1C1
Groupe 3

Rapport technique

Projet GL2 - Développement Orienté Objet

Application de gestion de questionnaires

03/06/2017

Table des matières

Introduction

Le projet logiciel de ce second semestre concerne la création d'un logiciel de gestion de questionnaire intuitif et accessible à tous. Ce dernier est compatible avec le logiciel R et est entièrement développé en Java.

Ce projet avait pour but d'approfondir d'une part nos compétences en développement orienté objet mais aussi la gestion de projet à travers le travail en équipe et la planification à travers des problématiques auxquelles nous avons tenté de répondre tout au long du projet.

- Comment prendre avantages des différences de chacun dans un projet ?
- Comment avancer dans un projet quand on a tous un point de vue/interprétations différents du sujet ?

Notre équipe se compose d'un groupe de cinq élèves de GI1C1 tirés au sort, dont deux viennent de CPI, deux de CPGE et un ayant fait auparavant une license informatique.

I. Méthode SIXO

❖ Objectifs

Notre but était de créer un logiciel de gestion de formulaires, le principe étant semblable à celui Google Form, pour la fin du deuxième semestre.

Fonctionnalités critiques :

Inscription et création du profil de l'utilisateur

Créer, modifier et supprimer plusieurs questionnaires

Système de sessions de questionnaires

Transférer les données sur R

Fonctionnalités secondaires :

Traitement des données sur R

❖ Objets

Branche technique :

Développement du logiciel en Java. La persistance des données est faite grâce à la sérialisation de Java.

Branche management :

Nous avons utilisé la méthode SIXO pour le management du projet. Nous avons rendu un rapport détaillé dans la matière Travail en Equipe qui traite la manière dont nous avons géré et perçu le projet tous ensemble.

❖ Opérations et ordre

1. Élaborer la méthode SIXO pour ce projet logiciel et écrire le cahier des charges (5 jours)

2. Analyse UML (diagramme de classe, diagramme de séquence, Use case, diagramme d'état) (20 jours) - *nécessite opération 1*

3. Création des modèles principaux (24 jours) - *nécessite opération 2*

4. Création de la vue (5 jours)

5. Création du contrôleur (3 jours) - *nécessite opérations 3 et 4*

6. Choix du conteneur (implémentation des relations multiples) (10 jours) - *nécessite opération 3*

7. Création des fonctionnalités secondaires (12 jours) - *nécessite opérations 1, 2 et 3*

8. Validation unitaire (2 jours)

9. Réalisation du rapport (10 jours)

❖ Opérateurs

Soient les opérateurs suivants :

- **A** : FOTIE Koffi Dramane
- **B** : PEZZALI Pierre-Maximilien
- **C** : REGNIER Fabien
- **D** : SONG Delphine
- **E** : ZHANG Alain

On a par suite :

- Opération 1 : D
- Opération 2 : B, D, et E
- Opération 3 : A, C et D
- Opération 4 : A, B, C, D et E
- Opération 5 : B et E
- Opération 6 : D
- Opération 7 : B
- Opération 8 : A, B, C, D et E
- Opération 9 : D

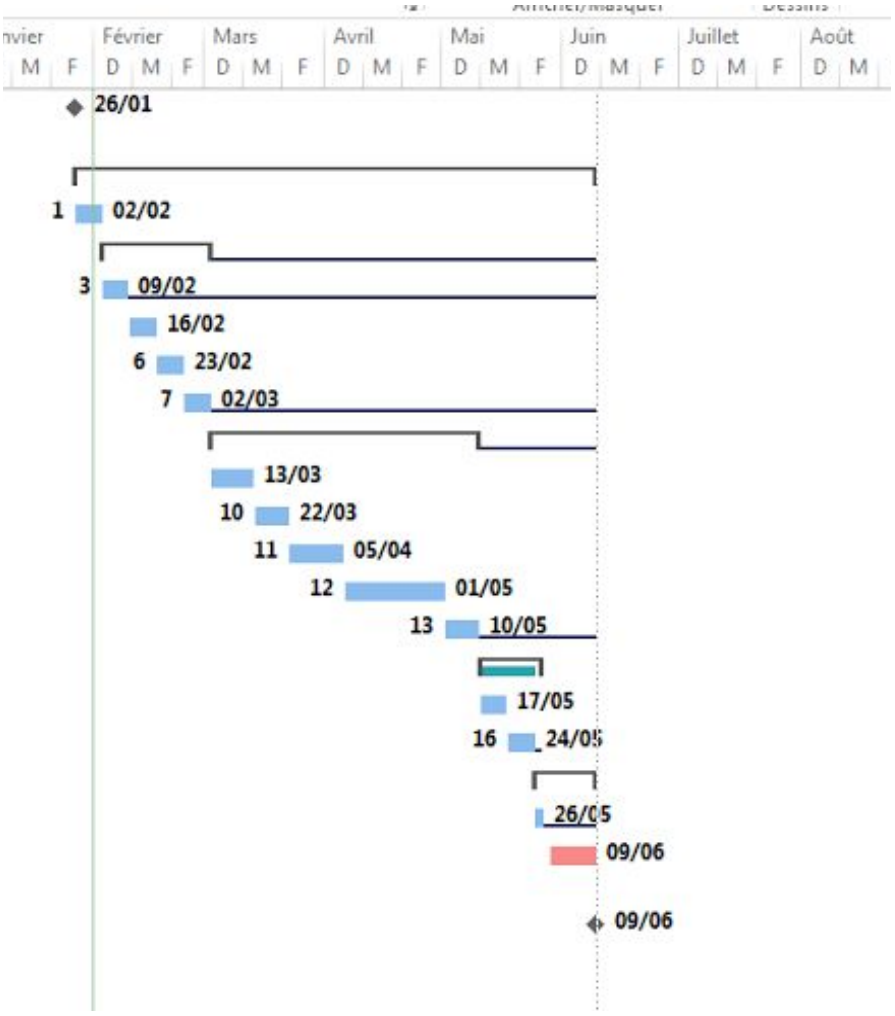
❖ Diagramme de Gantt

Format | Colonnes | Styles des barres

DIAGRAMME DE GANTT

	Mode Tâche	Nom de la tâche	Début	Fin	Durée
1		Début du projet	Jeu 26/01/17	Jeu 26/01/17	0 j
2		**Logiciel**	**Ven 27/01/17**	**Ven 09/06/17**	**96 j?**
3		Méthode SIXO et rédaction du cahier des charges	Ven 27/01/17	Jeu 02/02/17	5 j
4		**Jalon N°1 - Analyse UML**	**Ven 03/02/17**	**Jeu 02/03/17**	**20 j**
5		Diagramme de classe	Ven 03/02/17	Jeu 09/02/17	5 j
6		Diagramme de séquence	Ven 10/02/17	Jeu 16/02/17	5 j
7		Use case	Ven 17/02/17	Jeu 23/02/17	5 j
8		Diagramme d'état	Ven 24/02/17	Jeu 02/03/17	5 j
9		**Jalon N°2 - Fonctionnalités critiques**	**Ven 03/03/17**	**Mer 10/05/17**	**49 j?**
10		Inscription	Ven 03/03/17	Lun 13/03/17	7 j
11		Connection, déconnection	Mar 14/03/17	Mer 22/03/17	7 j
12		Créer, modifier, supprimer un formulaire	Jeu 23/03/17	Mer 05/04/17	10 j
13		Pattern MVC	Jeu 06/04/17	Lun 01/05/17	18 j?
14		Récupérer les données	Mar 02/05/17	Mer 10/05/17	7 j
15		**Jalon N°3 - Fonctionnalités secondaires**	**Jeu 11/05/17**	**Ven 26/05/17**	**12 j**
16		Système de sessions de questionnaires	Jeu 11/05/17	Mer 17/05/17	5 j
17		Transférer les données sur R	Jeu 18/05/17	Mer 24/05/17	5 j
18		**Jalon N°4 - Finalisation**	**Jeu 25/05/17**	**Ven 09/06/17**	**12 j?**
19		Validation unitaire	Jeu 25/05/17	Ven 26/05/17	2 j?
20		Rédiger la documentation technique et le mode d'emploi et les faire imprimer	Lun 29/05/17	Ven 09/06/17	10 j
21		Fin du projet	Ven 09/06/17	Ven 09/06/17	0 j

Planning initial réalisé pour le cahier des charges

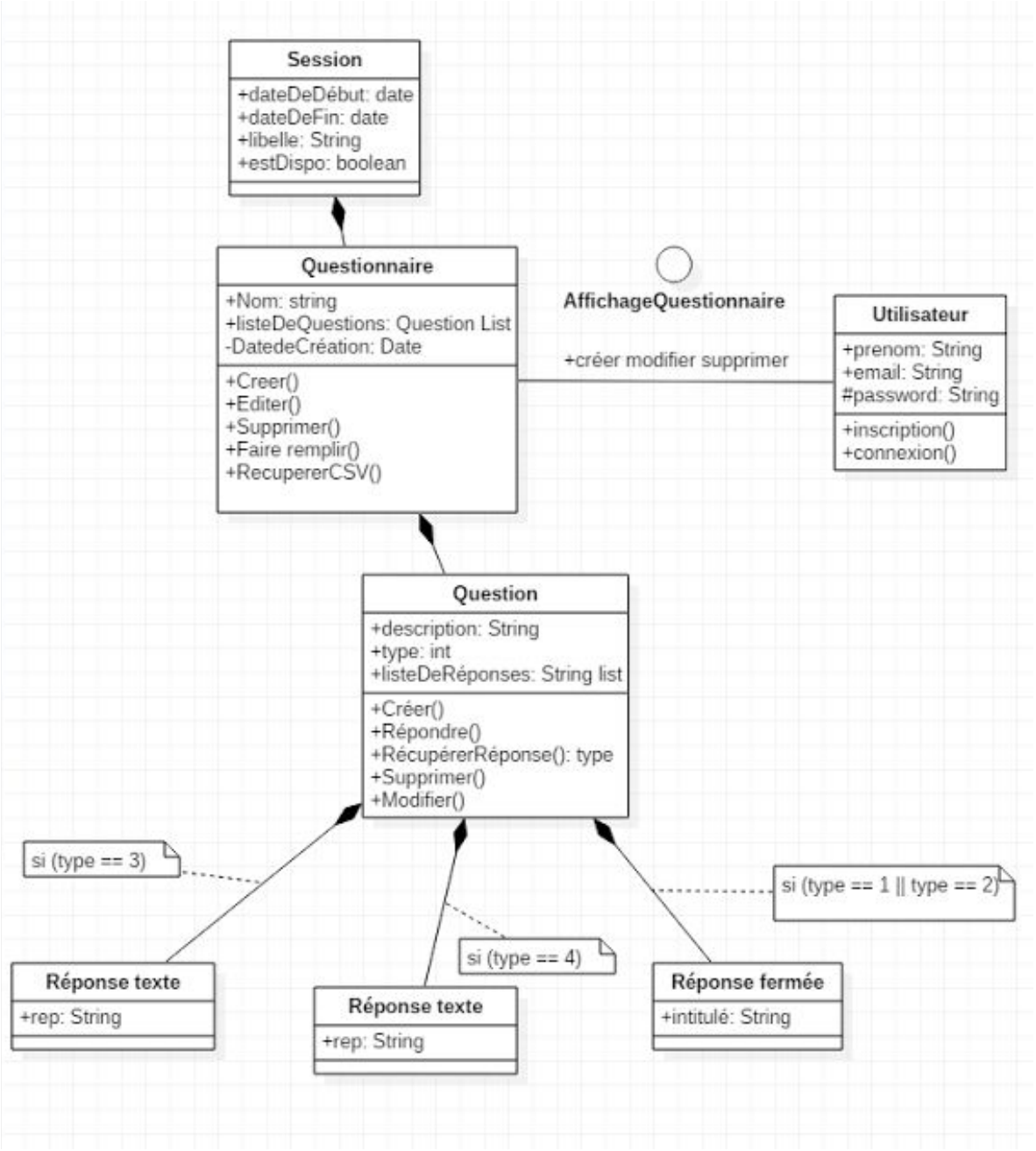


Nous traiterons de la gestion des jalons et des délais dans la partie “Difficultés rencontrées”.

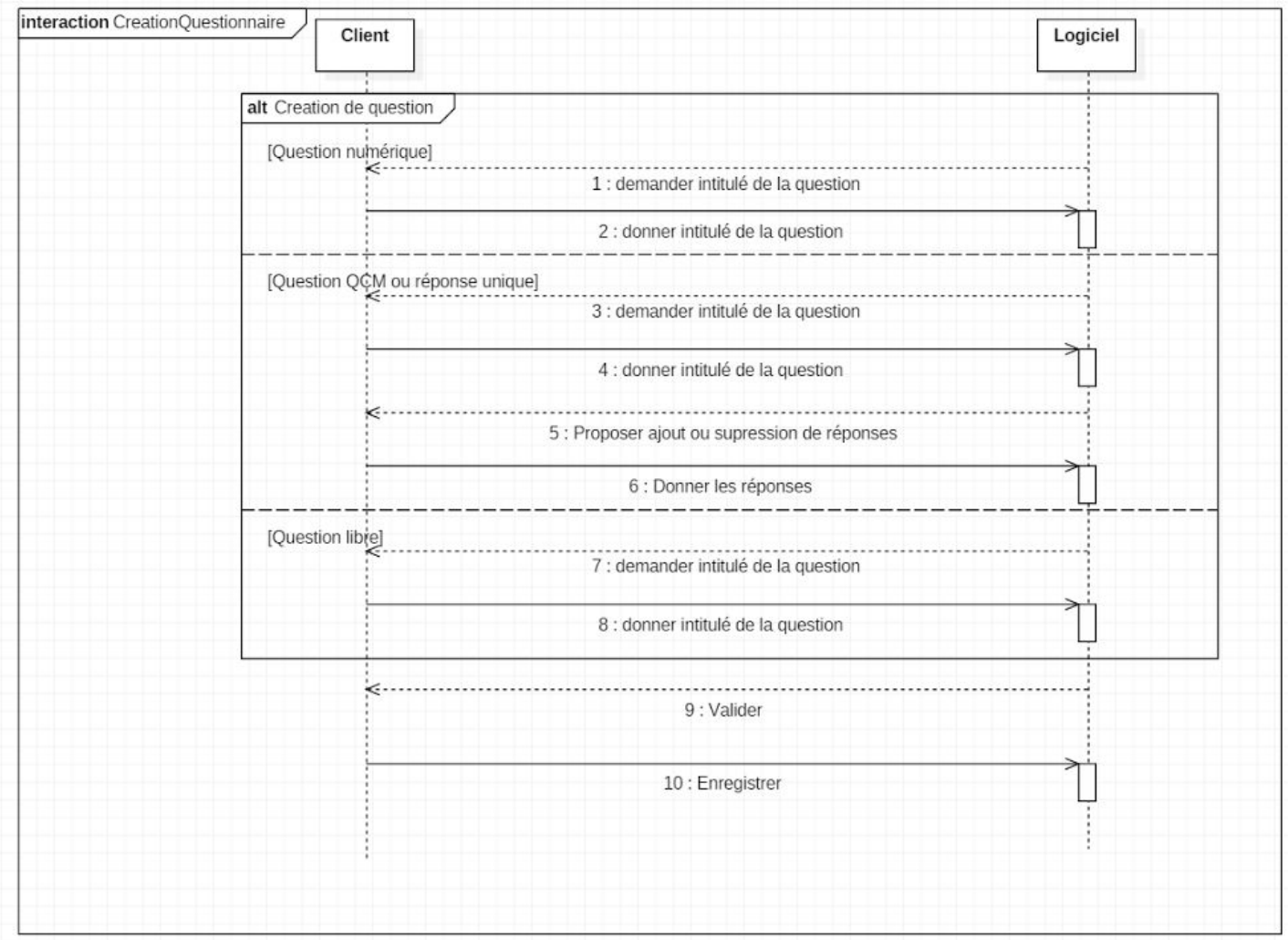
II. Analyse UML

Chaque diagramme a été réalisé sur StarUML.

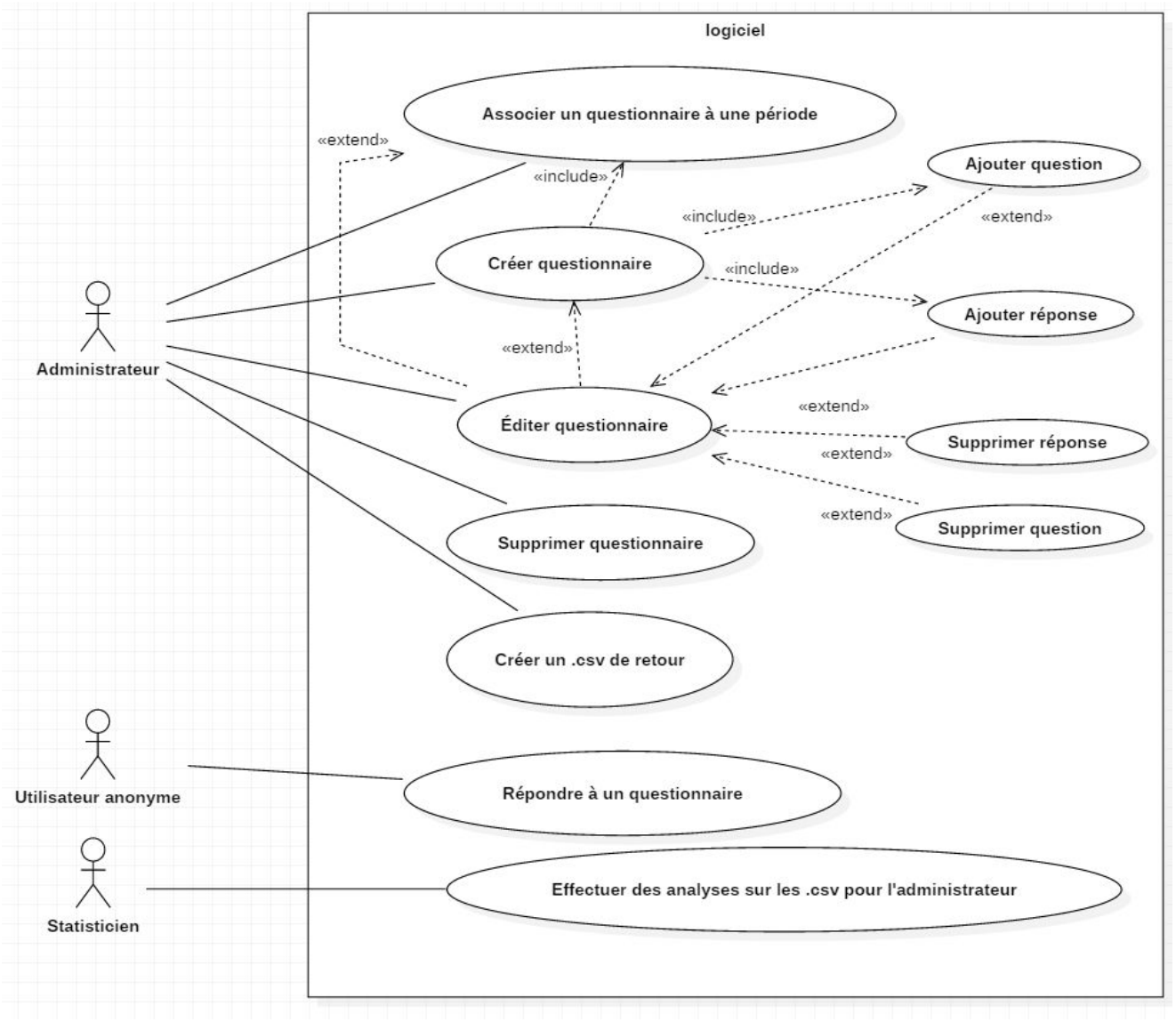
A. Diagramme de classes



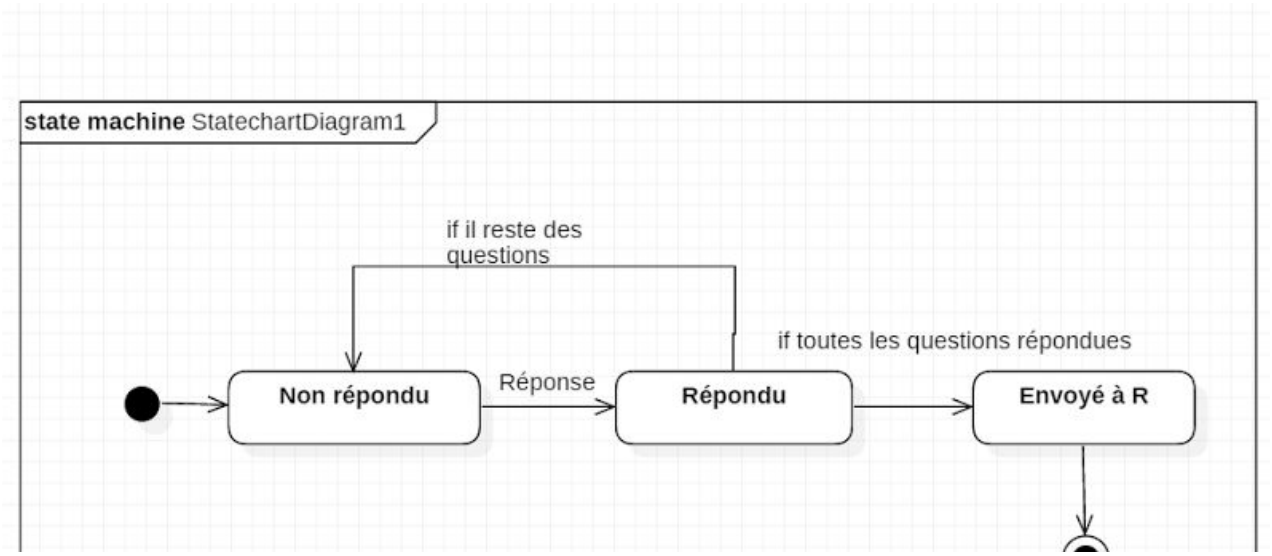
B. Diagramme de séquence



C. Diagramme de cas d'utilisation



D. Diagramme d’états-transitions



III. “Compte” utilisateur (anonyme)

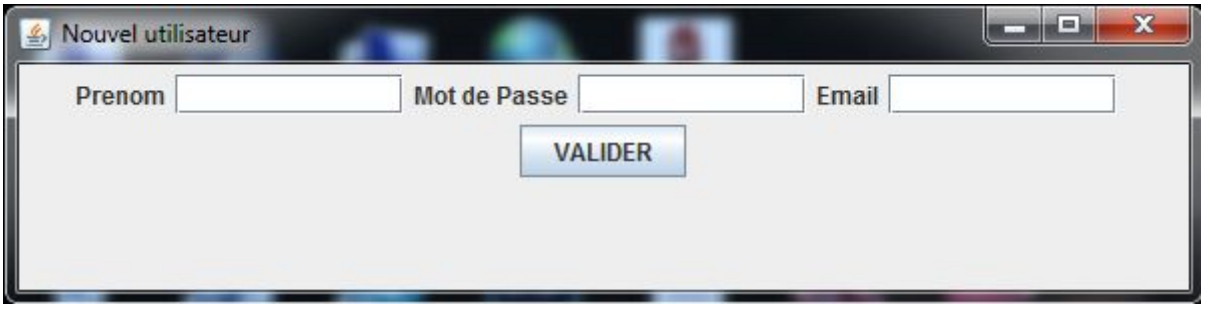
Lorsque l’on lance l’application, l’utilisateur dispose de plusieurs choix, il peut soit :

- s’inscrire
- se connecter s’il possède un compte (l’utilisateur de vient donc admin)
- choisir et répondre aux questionnaires mis à sa disposition

1) Il faut lancer l’application à partir de Inscription.java pour s’inscrire
2) Il faut lancer l’application à partir de Fenetre.java pour accéder à la création de questionnaire

Inscription

Lorsque l’on clique sur “*Inscription*”, une fenêtre s’affiche et il faut renseigner son prénom, un mot de passe ainsi que son email qui servira d’identifiant pour se connecter dans l’application :



Une fois l’inscription validée, on propose tout de suite à l’utilisateur de se connecter.

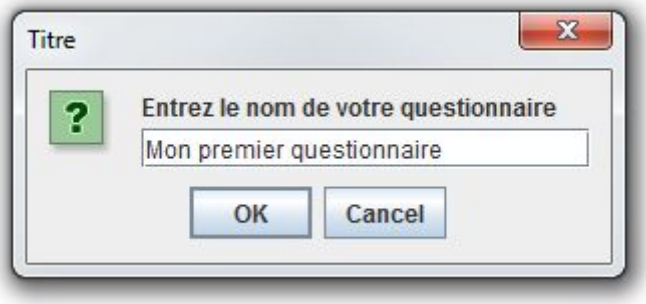
Connexion

Lorsque l’utilisateur clique sur “*Connexion*”, une autre fenêtre apparaît où il faut renseigner son identifiant (email) et son mot de passe :

IV. Compte admin

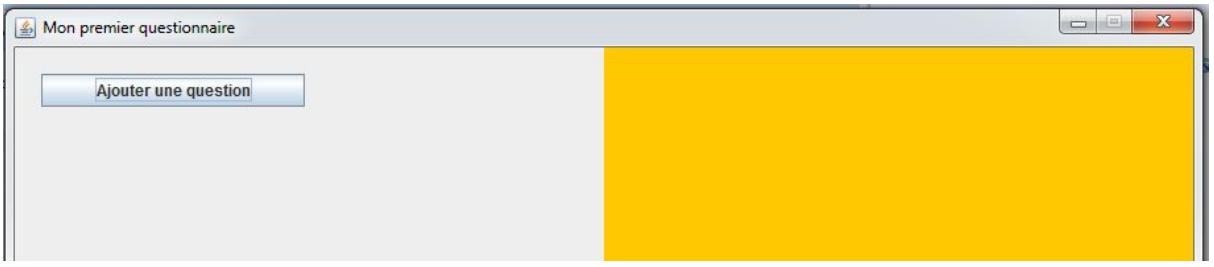
Une fois connecté, l'utilisateur (que l'on va désormais appelé admin) peut créer un questionnaire s'il le souhaite en cliquant sur le bouton *"Créer un questionnaire"*.

On lui demande alors grâce à un *JOptionPane.showInputDialog* le nom qu'il souhaite donner au questionnaire (il peut alors confirmer la création en cliquant sur *OK*, ou l'annuler en appuyant sur *cancel*).



Création de questionnaire

En validant l'admin arrive donc sur la fenêtre ci-dessous.



En cliquant sur "*Ajouer une question*", un autre *JOptionPane.showInputDialog* apparaît (*Figure 1*) et demander l'intitulé de la question. On demande ensuite à l'admin quel type de question il souhaite (*Figure 2*)

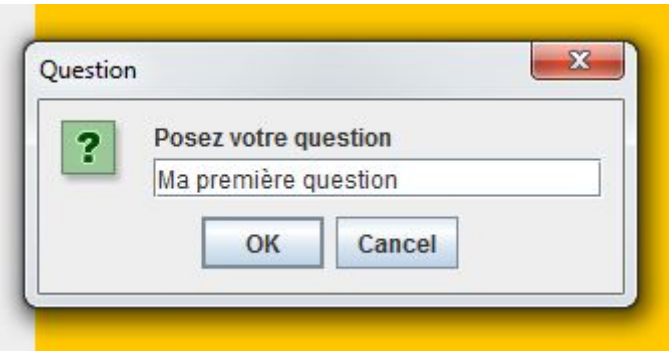


Figure 1

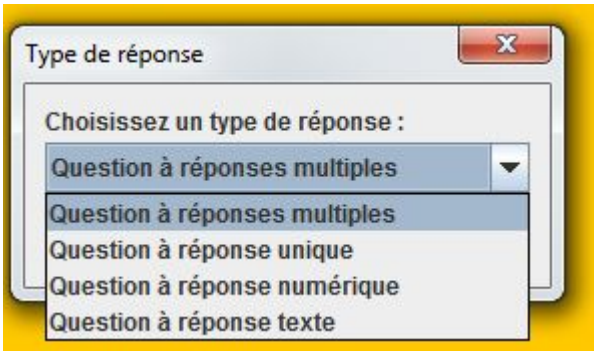
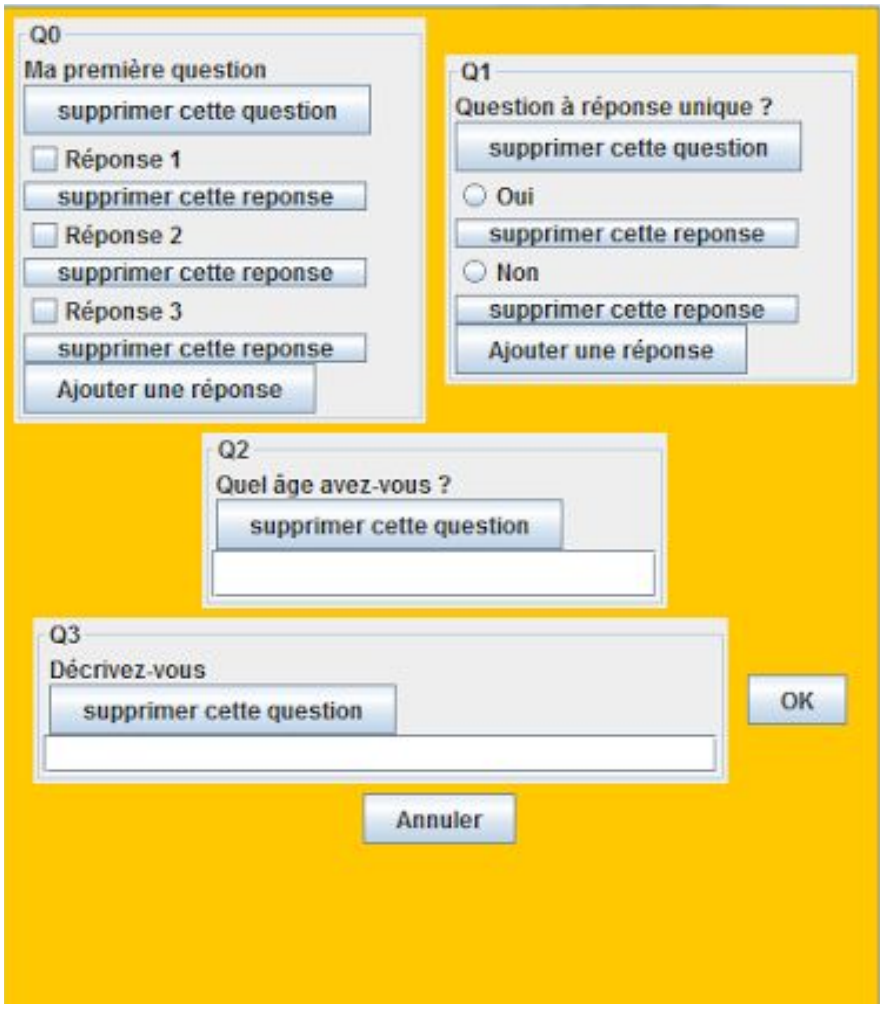


Figure 2

On peut alors obtenir cet affichage selon le type de question chois :.



L'admin peut supprimer des questions ou des réponses en cliquant simplement sur le bouton correspondant. Ici les questions 2 et 3 nécessitant l'entré de l'utilisateur, il n'est pas possible de supprimer ou d'ajouter des réponses.

L’admin peut également accéder à sa liste de questionnaires déjà créés. Il peut alors les modifier ou les supprimer en cliquant sur les boutons correspondants. S’il clique sur le bouton “*Modifier*”, il se retrouve sur la fenêtre de création de questionnaire avec les questions et réponses du questionnaire déjà créés. Il pourra donc supprimer/ajouter des question et/ou des réponses.

V. Partie statistique

VI. Explication du code

A. Inscription et connexion

Nous avons eu besoin de 4 classes concernant l'inscription et la connexion :

- **NewUser.java**

```
public class NewUser {
    String prenom;
    String motDePasse;
    String email;
```

Cette classe sert à modéliser un admin. Ce dernier aura comme identifiant son email ainsi qu'un mot de passe qu'il aura lui-même choisi.

- **Inscription.java**

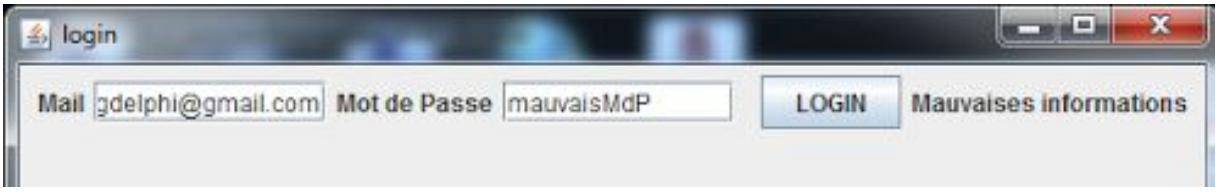
On demande à l'utilisateur de renseigner les informations cités ci-dessus afin de les stocker dans les variables de classes de NewUser. Le NewUser créé sera ajouté à une liste de NewUser (dans ***connection1.java***)

```
public  class connection1 {

    public static   List<NewUser> liste=new ArrayList<NewUser>();
```

- **Login.java**

On vérifie que ce que rentre l'utilisateur concorde avec l'email ET le motDePasse d'un NewUser en parcourant la liste de NewUser et en comparant grâce à equals(). Si l'utilisateur rentre les mauvaises informations pour se connecter, alors on le prévient :



B. Création de questionnaire

Pour créer un questionnaire nous avons créé 4 classes :

- **Reponse.java**

```
public class Reponse {

    private String reponse;
```

Un objet Reponse ne comporte qu'une variable de classe : le *String reponse* correspond à l'intitulé de la réponse (pour les questions de type QCM ou de type réponse unique).

La méthode ci-dessous sert à demander à l'admin quel sera l'intitulé de la réponse (et donc de créer la réponse).. On ajoute la réponse dans une liste de réponses associé à un objet Question.

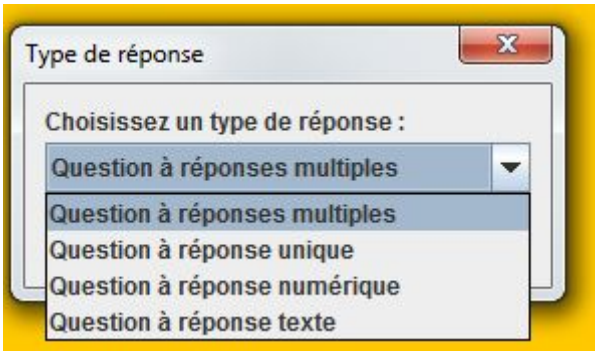
```
public void creerReponse(Question q){
```

- **Question.java**

```
public class Question {
    private String description;
    private int type;

    List<String> repTab; //tableau où on stocke toutes les réponses
```

Grâce à **public int** typeReponse(), l'admin peut choisir le type de question grâce à un *JOptionPane.showInputDialog* et donc la valeur de *type* qui peut être égale à 1, 2, 3 ou 4.



```
public void afficherReponse(Fenetre fenetre, JPanel panel_QR, JPanel panel_R){
```

Cette méthode affiche les réponses différemment selon le type de la question.

Si l'admin choisit :

- Question à réponses multiples -> *type* = 1 -> JCheckBox
- Question à réponse unique -> *type* = 2 -> JRadioButton
- Question à réponse numérique -> *type* = 3 -> JFormattedTextField
- Question à réponse texte -> *type* = 4 -> JTextField

Pour les questions de type 1 et 2, les réponses étant stockées respectivement dans leur liste *repTab*, on utilise une boucle *for* qui parcoure la liste et qui affiche les réponses dans un JPanel.

Etant une liste, il est plus facile pour nous de supprimer une réponse de la liste en faisant *.remove()*

- **Questionnaire.java**

```
public class Questionnaire {
    private String nom;
    private Date dateCreation;
    List<Question> questionTab; //tableau où on stocke toutes les questions
    List<JButton> tabJButtonRep;
```

Un questionnaire possède également une liste de Question (et chaque Question de la liste possède donc une liste de Réponse).

```
public void affichageQuestions(Fenetre fenetre, JPanel pan)
```

Le principe va être le même, nous utilisons une boucle *for* pour l'affichage. Pour chaque question on crée un JPanel pour pouvoir afficher les questions les unes après les autres.

❖ **Fenetre.java**

Cette classe crée la fenêtre qui sert à créer/modifier un questionnaire. Il suffit juste de créer un seul objet *Fenetre*.

```
public class Fenetre extends JFrame implements ActionListener{

    private JPanel pan = new JPanel();
    private JPanel panGauche = new JPanel();

    private Questionnaire questionnaire = new Questionnaire();

    private JButton boutonAddQuestion = new JButton ("Ajouter une question");
```

On souhaite diviser la fenêtre en deux pour avoir d'un côté un bouton "*Ajouter question*" (JButton *boutonAddQuestion*) sur le JPanel *panGauche* et de l'autre côté l'affichage du questionnaire sur le JPanel *pan*.

Lorsque l'on clique sur *Ajouter question* on crée un nouvel objet Question qui demande systématiquement l'intitulé de la question.

❖ **Boutons et Listeners**

À chaque fois que l'on clique sur un bouton on remet à zéro l'affichage et on ré-affiche pour tout actualiser avec les boucles for (affichage des réponses et des questions).

```
private class ButtonReponseListener implements ActionListener
```

➔ crée une réponse, clear le panel de réponses, réaffiche toutes les réponses pour la question associée

```
private class ButtonQuestionListener implements ActionListener
```

➔ crée une question, demande le type de question, ajoute la question à la liste de Question du questionnaire, clear le panel et réaffiche toutes les questions

```
private class ButtonDeleteListener implements ActionListener
```

➔ dans Questionnaire.java : supprime une question de la liste, clear le panel et réaffiche toutes les questions

➔ dans Questionnaire.java : supprime une réponse de la réponse, clear le panel et réaffiche toutes les réponses associées à chaque question

VII. Difficultés rencontrées

La difficulté majeure de ce projet a été les cours. En effet, nous avons commencé à avoir des cours d'IHM seulement le 4 Mai dernier. Nous, ainsi que d'autres groupes, trouvons que nous les avons eu beaucoup trop tard par rapport à la date de rendu du projet (1 mois d'écart). Nous avons donc pris beaucoup de retard en attendant ces cours. Nous nous sommes bien sur documenter par nous même, mais il était compliqué de bien assimiler ces compétences puisque jusqu'au 4 Mai, nous continuions d'apprendre le Java en cours. Nous ne voulions pas trop aller trop loin dans le code tant que nous n'avions pas terminé tous les cours de Java par peur de perdre du temps en tentant de réaliser certaines choses qui peuvent être faites rapidement une fois les compétences acquises en cours de Java et en TD. Nous nous sommes donc retrouvés à passer beaucoup de temps à faire énormément de recherches et à manquer de temps vers la fin du projet. Il n'était donc pas possible pour nous de respecter les jalons que nous nous étions imposés au tout début du projet.

Bibliographie

https://openclassrooms.com

https://docs.oracle.com

http://www.youtube.com

www.javatpoint.com

https://developpez.net

Enormément de forums sur :

- https://stackoverflow.com
- https://developpez.net
- https://commentcamarche.net