INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN

Projet Informatique

Projet 4 : Gestion par objet d'une salle d'ordinateurs

Figure 1 – Photo d'une salle informatique

Roxane LEDUC
Ariane DEPONTHIEUX
GM3

Encadré par : M. Abdulrab

Février 2022 — Mars 2022

Table des matières

1 Introduction

L'objectif du projet est de modéliser et de représenter par objets une salle d'ordinateurs, et d'implémenter les opérations de base pour sa gestion, en Java. Nous avons choisi ce sujet dans le but de découvrir par nous même ce nouveau langage de programmation et de mieux appréhender les cours dispensés à l'INSA. Nous sommes persuadées de l'intérêt de ce type de programmation orientée objet dans le cadre de notre formation. Ce projet a été codé avec l'éditeur éclipse.

Ci-dessous notre diagramme UML, qui permet de comprendre la structure générale de notre programme.

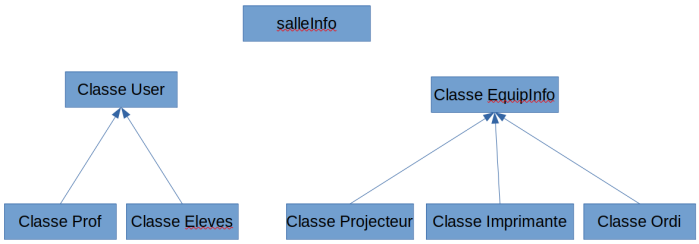

FIGURE 2 – Diagramme UML

2 Les différentes classes

2.1 Classe User

Les classes "Eleve" et "Prof" vont hériter de la classe User. En effet, nous partons du principe que toute personne essayant de se connecter dans la salle informatique doit être l'un ou l'autre.

Voici les attributs et méthodes de la classe "User" :

Classe User
String login String mdp
User(String login, String mdp) public String getLogin() public String getMdp() public boolean mdpValide(String m)

Les méthodes de cette classe sont :
- les getters :
  - **public String getLogin()**
  - **public String getMdp()**
- le constructeur : **User(String login, String mdp)**
- une méthode qui va permettre de vérifier la validité du mot de passe : **public boolean mdpValide(String m)**

2.1.1 Classe Eleve

La classe "Eleve" hérite de tous les attributs de la classe "User" et de toutes ses méthodes. On ajoute l'attribut **String INE** à cette classe.

Voici donc les attributs et méthodes de la classe "User" :

Classe Eleve
String login String mdp private String INE
Eleve(String INE,String login, String mdp) public String getINE() public String toString()

Voici les méthodes de cette classe :
- le getter : **public String getINE()**
- le constructeur : **Eleve(String INE,String login, String mdp)**
- la redéfinition de toString : **public String toString()**

2.1.2 Classe Prof

La classe "Prof" hérite de tous les attributs de la classe "User" et de toutes ses méthodes. On ajoute l'attribut **String NUMEN** à cette classe.

Voici donc les attributs et méthodes de la classe "User" :

Voici les méthodes de cette classe :

Classe Prof
String login String mdp private String NUMEN
Prof(String NUMEN, String login, String mdp) public String getNUMEN() public String toString()

— le getter : **public String getNUMEN()**
— le constructeur : **Prof(String NUMEN, String login, String mdp)**
— la redéfinition de toString : **public String toString()**

2.2 Classe EquipInfo

Voici les attributs et méthodes de la classe "Equipinfo" :

Classe EquipInfo
private String idEquip private Date dateAchat private Date dateHS
EquipInfo () EquipInfo(String id, Date date1, Date date2) public String getIdEquip() public Date getDateAchat() public Date getDateHS() public void setIdEquip(String NouveauId) public String toString()

Voici les méthodes de cette classe :
— le getter : **public String getNUMEN()**
— le constructeur : **Prof(String NUMEN, String login, String mdp)**
— la redéfinition de toString : **public String toString()**

2.2.1 Classe Ordi

La classe "Ordi" hérite de tous les attributs de la classe "EquipInfo" et de toutes ses méthodes. On ajoute aux attributs de cette classe, des caractéristiques importantes qui définissent un ordinateur :
— sa marque
— sa mémoire interne (en GiB)
— la capacité de son disque dur (en GB)
— son système d'exploitation

Voici les attributs et méthodes de la classe "Ordi" :

Classe Ordi
private String marque private float memoire //en GiB private float capaciteDisque //en GB private String systExploit
Ordi(String idEquip, Date dateAchat, Date dateHS, String marque, float memoire, float capaciteDisque, String systExploit) public String getMarque () public float getMemoire () public float getCapaciteDisque () public String getSystExploit () public void afficherCaraOrdi() public String toString()

2.2.2 Classe Projecteur

La classe "Projecteur" hérite de tous les attributs de la classe "EquipInfo" et de toutes ses méthodes. On ajoute aux attributs de cette classe, des caractéristiques importantes qui définissent un projecteur :

- sa marque
- le type de résolution (XGA, WXGZ ...)
- l'option de haut parleur intégrée (true si l'option est intégrée / false si elle ne l'est pas)
- l'option WiFi intégrée (true si l'option est intégrée / false si elle ne l'est pas)

Voici les attributs et méthodes de la classe "Projecteur" :

Classe Projecteur
private String marque private String resolution //XGA, WXGZ, ... private boolean hautParleur private boolean wiFi
Projecteur(String idEquip, Date dateAchat, Date dateHS, String marque, String resolution, boolean hautParleur, boolean wiFi) public String getMarque () public String getRes () public boolean getHP () public boolean getWiFi () public String toString()

2.2.3 Classe Imprimante

La classe "Imprimante" hérite de tous les attributs de la classe "EquipInfo" et de toutes ses méthodes. On ajoute aux attributs de cette classe, des caractéristiques importantes qui définissent un projecteur :

- sa marque
- la possibilité d'imprimer en couleur (true si oui/ false si non)
- sa mémoire interne (en Mo)

Voici les attributs et méthodes de la classe "Imprimante" :

Classe Imprimante
private String marque private boolean couleur private int memoireInterne
Imprimante (String idEquip, Date dateAchat, Date dateHS, String marque, boolean couleur, int memoireInterne) public String getMarque () public boolean getCouleur () public int getMemoireInterne () public String toString()

2.3 Classes de Test

Après avoir créé les classes "User", "EquipInfo" et leurs sous-classes, et afin de tester le bon fonctionnement des getters, des setters, des listes et autres méthodes implémentés, nous avons créé deux classes de test : "TestUser" et "TestEquipInfo".

Ceci fonctionnant correctement, nous avons ensuite commencé à mettre en oeuvre le "Main", en veillant à utiliser les méthodes de nos classes.

3 Le Main

Les premières lignes du Main nous servent de base de données. Nous y avons entré toutes les données concernant les utilisateurs et les équipements informatiques dans le dur. Nous créons notamment des tableaux qui nous serviront par la suite. Ensuite vient l'affichage. L'utilisateur a deux choix. Il peut soit se connecter ou quitter l'interface. Une fois connecté, il va pouvoir s'identifier en tant que professeur ou élève. Nous veillons à ce que son login soit présent dans la base de données et ensuite que son mot de passe soit correct. Il a la possibilité de réessayer de se connecter.

Si il parvient à se connecter en tant qu'élève, deux choix vont se présenter à lui :
- consulter la liste des équipements informatiques, et leurs caractéristiques
- créer un nouveau profil utilisateur (et ensuite voir la liste des personnes qui se sont connectées sur ce poste)

Si c'est en tant que professeur il pourra :
- enregistrer un nouvel équipement informatique (et ensuite voir la nouvelle liste des équipements informatiques)
- créer un nouveau profil utilisateur (et ensuite voir la liste des personnes qui se sont connectées sur ce poste)

Nous rendons l'interface agréable à visualiser à l'aide de la fonction **clearScreen()** que nous avons définit à la toute fin du Main. Cependant il est à noter que sous eclipse, cette fonction ne marche pas comme voulu. Il est conseillé de lancer le programme sur le Terminal pour un résultat visuel agréable.

Ci-dessous quelques affichage de notre programme depuis le terminal :

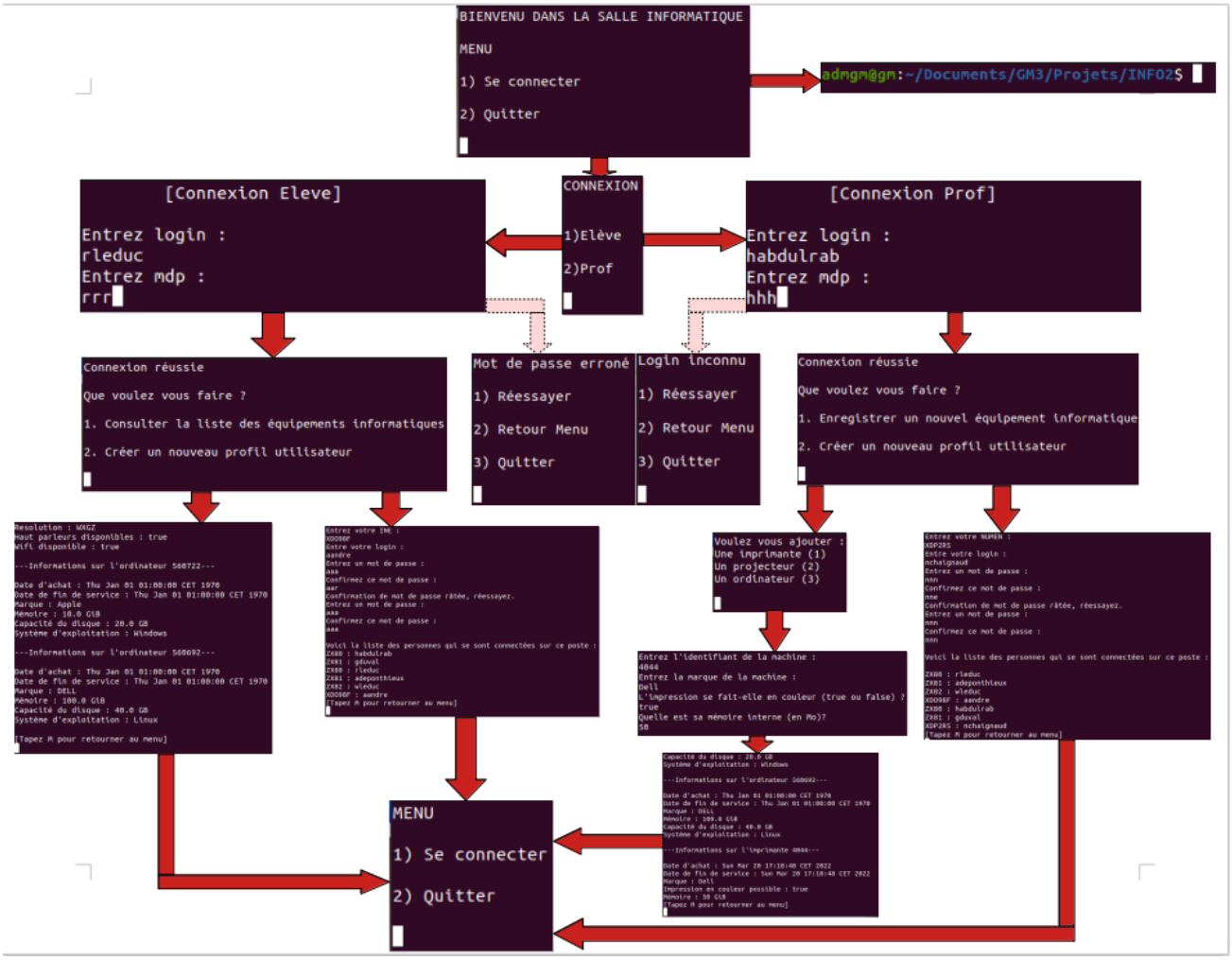

FIGURE 3 – Captures et liens des possibilités offertes dans la salle informatique au global

Remarque : A chaque nouveau affichage, l'utilisateur peut soit choisir l'action qui va suivre, soit il est redirigé (vers le menu par exemple). Ses enchaînements de possibilités sont représentés par les flèches rouges. On voit notamment que la plupart des possibilités mènent au menu. En effet, une fois que l'utilisateur a effectué l'action qu'il souhaitait (exemple : un élève que cré un nouveau profil utilisateur d'élève), il est ramené au menu et a la possibilité d'effectuer une nouvelle action dans la salle informatique, puis recommencer, et ainsi de suite.

Les deux flèches roses pâles en pointillés représentent des possibilités dans le cas où l'utilisateur se trompe de login ou de mot de passe lorsqu'il tente de se connecter. A noter que nous avons pris soin d'éviter au maximum les plantages dans les cas où l'utilisateur entre des mauvaises commandes. Par exemple, si on lui demande de choisir entre 2 options ((1) ou (2)) et qu'il rentre "a", il pourra réessayer, et ce jusqu'à ce que la commande soit valide. De même, si il se trompe de login ou de mot de passe, il peut réessayer, comme on peut le voir sur la figure.

Afin d'y voir mieux les différentes options, voici les captures des possibilités restreintes à la **[Connexion Eleve]**, puis à la **[Connexion Prof]**.

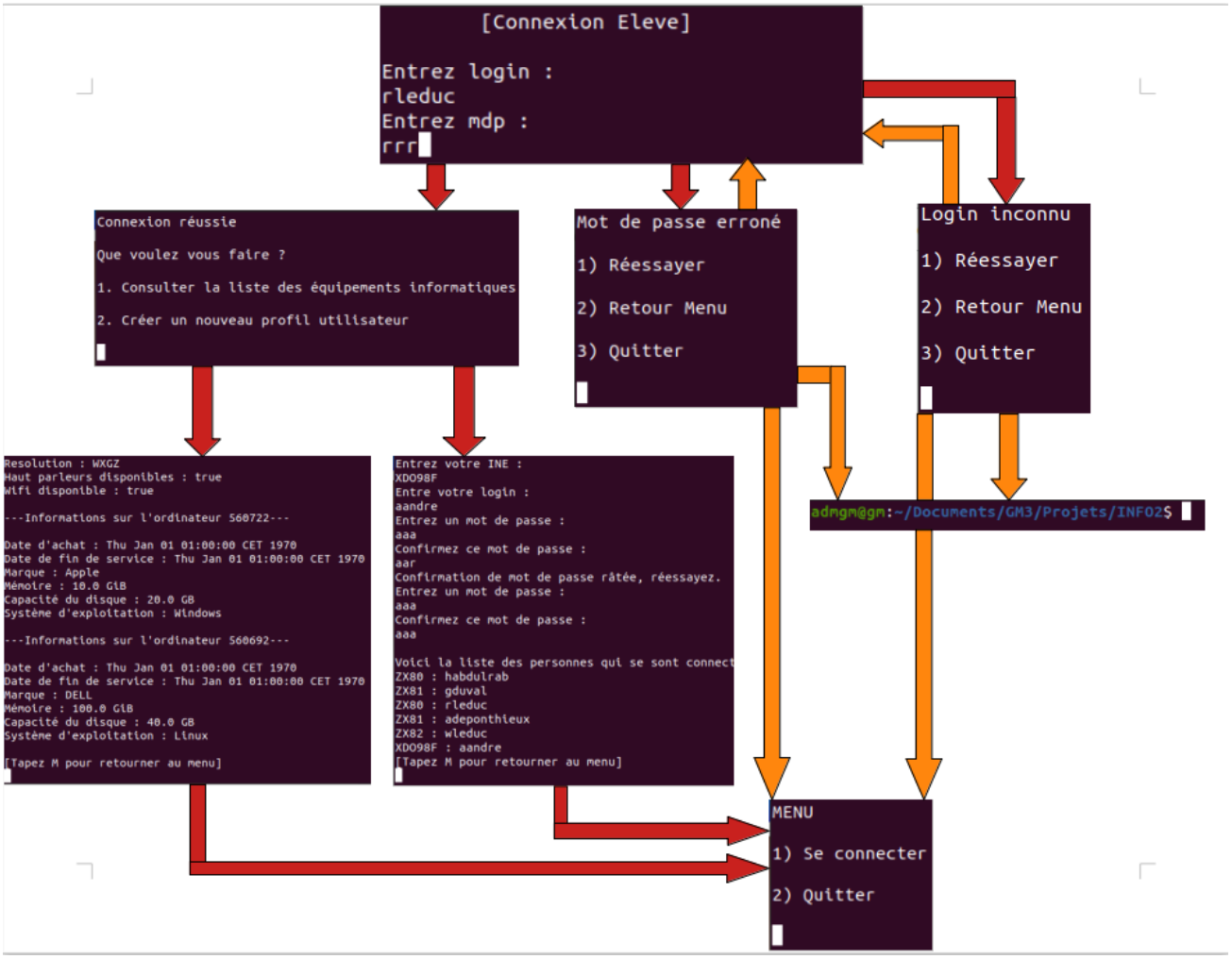

FIGURE 4 – Captures et liens des possibilités de connexion pour un élève

Les flèches oranges correspondent ici à des actions effectuées après s'être trompé de login ou de mot de passe.

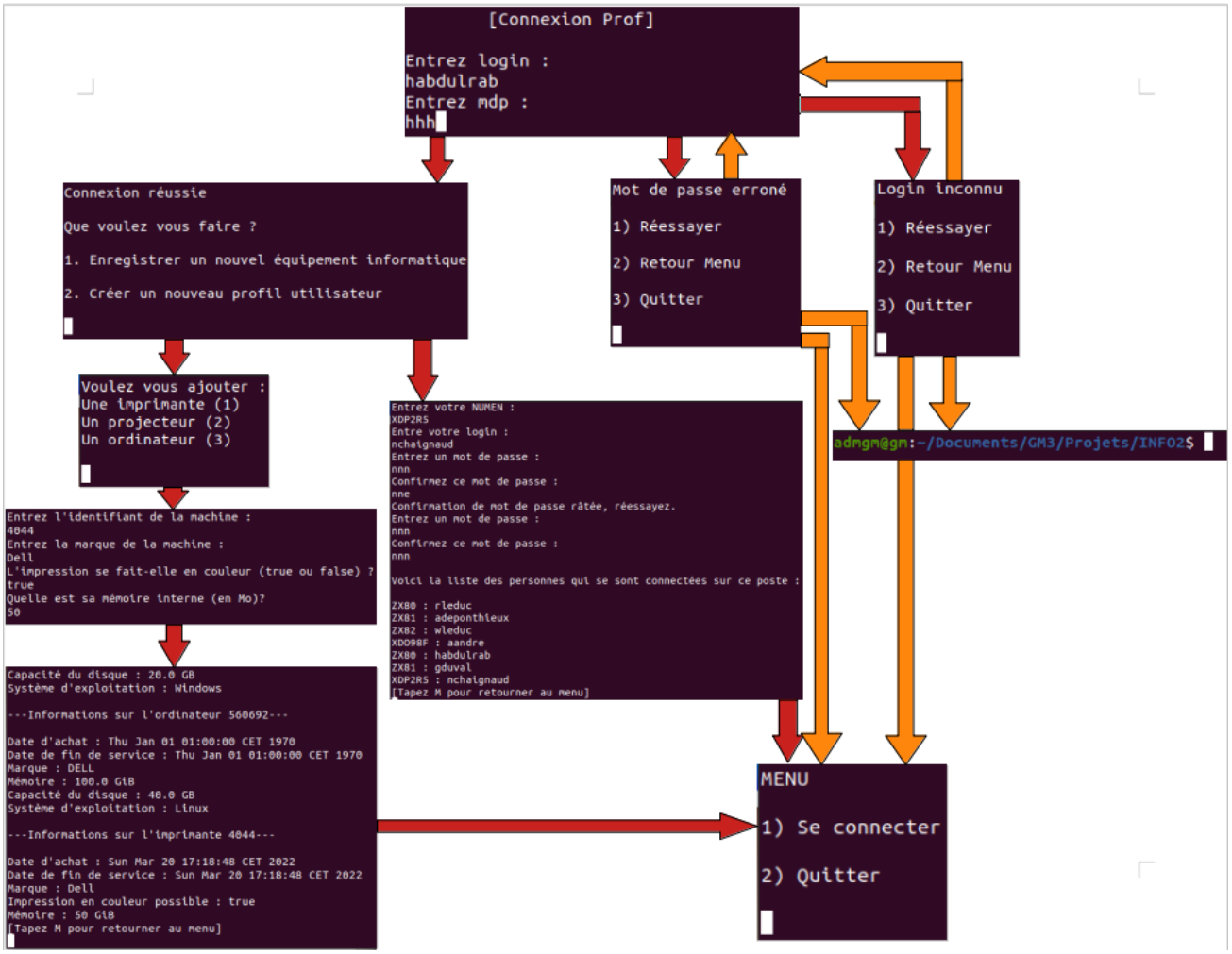

FIGURE 5 – Captures et liens des possibilités de connexion pour un prof

Ces trois figures permettent d'avoir une vue d'ensemble des possibilités offertes dans notre salle informatique. Nous vous invitons à aller parcourir cette salle par vous-même et à tester les différentes options.

Veuillez à suivre le guide d'utilisation pour vous assurer de réussir à lancer le programme et à pouvoir vous connecter à un compte.

4 Guide d'utilisation

Il s'agit ici de compléments au aareadme.

N'ayant pas encore appris à manipuler les fichiers en java, nous avons codé "dans le dur" les données de chaque User et de chaque EquipInfo (visibles directement depuis le début Main). Afin de ne pas trop alourdir le code, vous n'aurez la possibilité de vous identifier que sous 5 comptes différents. En tant que :

- **[Connexion Eleve]**
  - **Eleve1 :**
    - login : rleduc
    - mdp : rrr
  - **Eleve2 :**
    - login : adeponthieux
    - mdp : aaa
  - **Eleve3 :**
    - login : wleduc
    - mdp : www

- **[Connexion Prof]**
  - **Prof1 :**
    - login : habdulrab
    - mdp : hhh
  - **Prof2 :**
    - login : gduval
    - mdp : ggg

Afin de réussir à vous identifier et à accéder au reste du programme, il faudra que votre mot de passe soit correct. Une fois chose faite, vous aurez la possibilté de consulter ou de modifier certaines données (vous n'aurez pas les mêmes droits d'accès si vous êtes prof ou élève).

Nous avons codés sous éclipse et nous avons pu observer que le clearSreen() était défaillant avec cet éditeur. C'est pourquoi, pour que le programme fonctionne correctement, il est préférable de le faire fonctionner depuis le terminal.

Remarque : Nous avons compilé les **.java* avec la commande *javac* depuis le dossier salleInfo mais pour exécuter le Main nous avons été obligé de nous remettre dans le ficher src et d'entrer *java salleInfo/Main*.

Les instructions pour lancer le programme sont précisés en détail dans le aareadme.

5 Conclusion

Ce projet nous a permis de faire nos premiers pas dans le monde de la programmation orientée objet. Nous avons pu nous avancer un peu sur les TDs et apprendre par nous mêmes à nous servir de tableaux en Java. Nous n'en sommes encore qu'au début de notre apprentissage de ce langage, mais ce projet nous a permis d'être un peu plus à l'aise. Apprendre à utiliser éclipse est aussi un plus, car cet éditeur présente de nombreux avantages et nous permet d'être plus rapide (auto-complétion).

Nous savons que la base de données entrée en début de programme à chaque compilation n'est pas le moyen le plus optimal pour véritablement permettre une gestion de salle informatique. En effet, le meilleur moyen pour garder les informations pour les utilisateurs et les équipements informatiques serait l'utilisation de fichiers. Voilà un nouveau domaine qui nous intéresserait en Java.

Au niveau de la gestion de travail au sein du groupe, nous pensions que cela serait pénalisant d'avoir peu de moments pour nous retrouver car nous sommes dans les deux TD différents. Mais finalement, cela a été avantageux car cela nous a permis d'avancer chacune de notre côté et d'apprendre à utiliser l'outil github afin d'avoir accès aux modifications apportées sur le code par chacune. Roxane Leduc s'est cependant plus concentrée sur la partie $User$ et Ariane Deponthieux sur la partie $EquipInfo$ mais la répartition a été assez globale.