DOCUMENTATION TECHNIQUE

L’application M2L a pour but de gérer les compétitions, les équipes et les adhérents de la maison des Ligues de Lorraine.

Cette documentation est produite dans le cadre du développement de la version 1.0 du logiciel (daté du 11/05/2019).

Sommaire

Rappels des fonctionnalités attendus

A) Description fonctionnelle du projet

La gestion des inscriptions aux compétitions est encore aujourd'hui assurée par les ligues avec des tableurs, et le développement d'un logiciel davantage adapté est en cours. L'application en question permet de gérer un ensemble de compétitions, de personnes, et d'affecter des personnes à des compétitions. Il est possible que certaines compétitions soient réservées à des équipes et qu'il soit impossible à une personne seule de s'inscrire. Dans ce cas, tous les membres de l'équipe doivent être enregistrés.

Cette application ne sera pas accessible au grand public, seuls des employés des ligues pourront y accéder pour saisir les inscrits. Les précédents informaticiens se sont penchés sur le problème et ont déjà commencé à développer la couche métier.

Les ressources suivantes étaient à notre disposition :

- Une partie du code (sur Github) : https://bit.ly/2rEHQII
- Une documentation JavaDoc : https://bit.ly/2wBq51W

Le programme devait remplir certaines conditions :

1) Le programme devait pour être exécuté en mode « console ».
2) Le programme devait bénéficier d'une interface graphique utilisateur.
3) L'application devait pouvoir être utilisé en simultané sur plusieurs postes différents.
4) L'envoie d'email à un membre devait être possible directement depuis l'application.

B) Décomposition du projet

Cette application a été conçue par 2 développeurs membre de l'ITIC Paris :

- MLAGHUI Brahim (en tant que chef de projet et développeur principal) : http://www.mlaghuibrahim.com/
- BORGI IHCEN : http://www.borgi.fr/

Le développement de l'application a été découpé en 4 itérations :

- Itération 1 :

- Développement des classes *Compétitions*, *Equipes*, *Candidat* et *Personne* (entités)
- Développement du Controller principal : *Application*
- Ajout des exceptions
- Ajout des tests unitaires
- Modélisation de la base de données (MCD)
- Arborescence des menus.
- Élaboration d'un diagramme UML

- Itération 2 :

- Création de la base de données
- Installation de la base de données sur serveur local et distant
- Création interface console complète

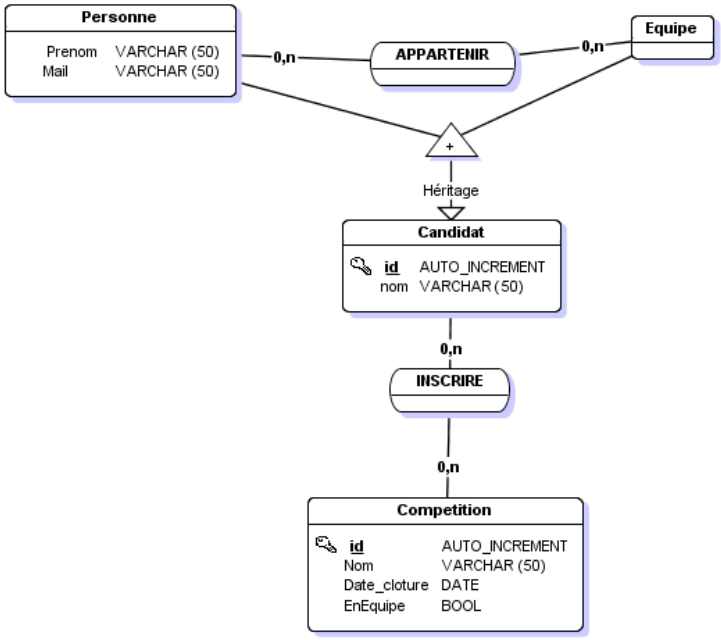
- Itération 3 :

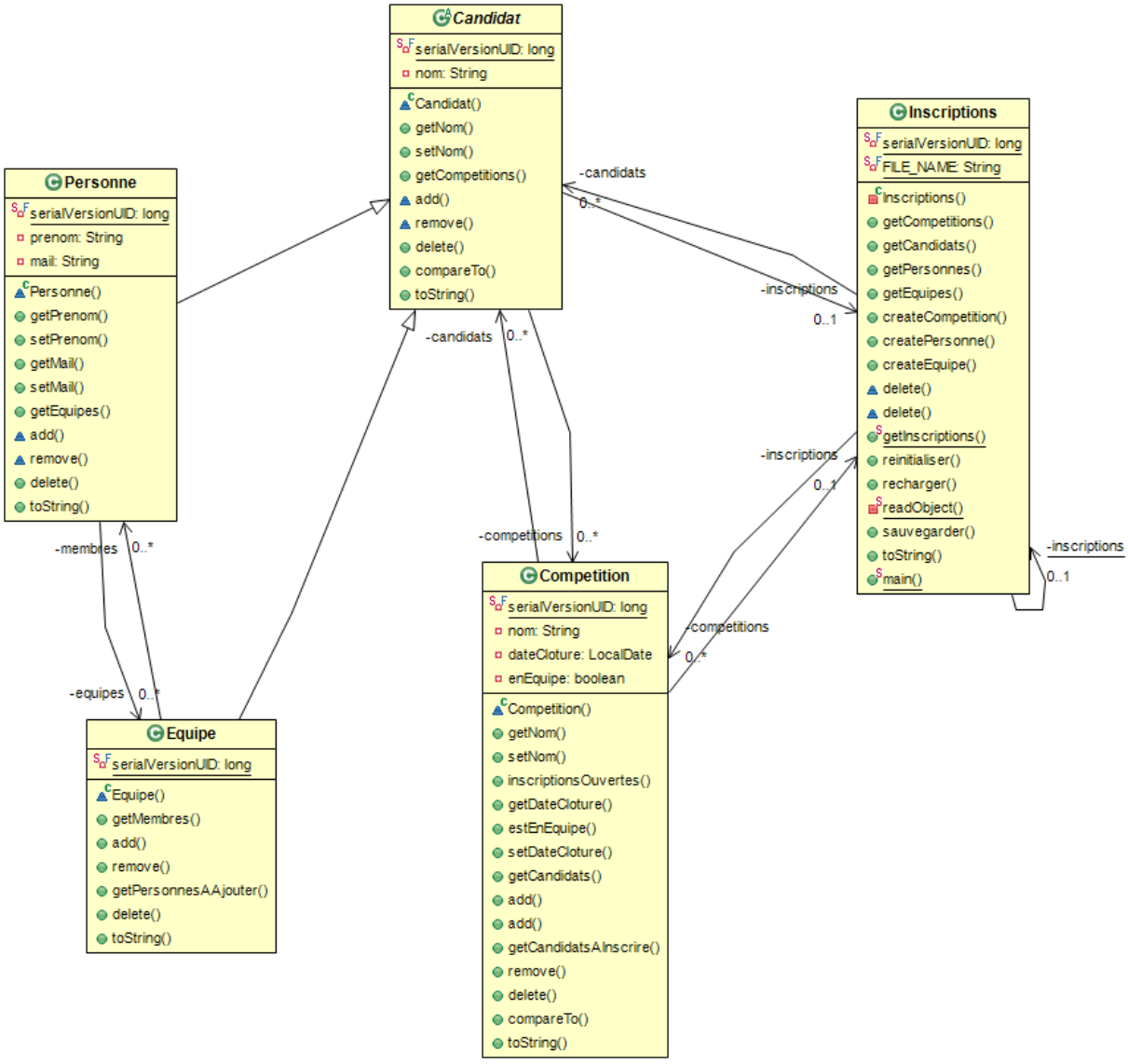
- Ajout de la connexion à la BDD avec l'ORM Hibernate
- Intégration d'Hibernante dans le code source de l'application
- Maquettages de l'interface graphique utilisateur.

- Itération 4 :

- Développement de l'IHM avec Swing
- Création Documentation Utilisateur et Technique

C) Principe de fonctionnement


Personne
Prenom VARCHAR (50)
Mail VARCHAR (50)
0,n
APPARTENIR
0,n
Equipe
+
Héritage
Candidat
id AUTO_INCREMENT
nom VARCHAR (50)
0,n
INSCRIRE
0,n
Competition
id AUTO_INCREMENT
Nom VARCHAR (50)
Date_cloture DATE
EnEquipe BOOL



D) Fichier de configuration

Lors de l'installation de l'application, plusieurs fichiers sont à configurer.

Tout d'abord renommer le fichier « hibernate.sample.cfg.xml » en « hibernate.cfg.xml ». Pour vous connecter à une base de données « custom » remplacer :

- ***localhost*** par l'url de votre base de données.
- ***3306*** par le port de votre base de données.
- ***m2L*** par le nom de votre base de données.
- ***root*** par le nom d'utilisateur de votre base de données.
- ***''*** par le mot de passe de votre base de données.

Technologies utilisées

A) Java (MVC, Héritage, PD, Polymorphisme …)

L'ensemble du programme a été développé avec la langage Objet Java. Les différentes classes ont été découpé en plusieurs packages :

- Le package ***controllers*** qui contient l'ensemble des classes logique de l'application.

- Le package ***entity***, lui contient l'ensemble des classes du logiciel représentants les candidats, équipes, personnes et compétitions (ces classes contiennent notamment les getters et les setters des différentes propriétés des classes).

- Le package ***models*** contient les classes permettant de sauvegarder / persister les éléments de l'application dans une base de données.

- Le package ***views*** contient l'ensemble des classes générant l'interface graphique (IHM) de l'application. Ce package contient également l'ensemble des classes permettant d'utiliser le programme en mode console.

- Le package ***images*** contient l'ensemble des assets nécessaire au fonctionnement de l'application (telle que les images, les sons …).

- Le package ***config*** contient l'ensemble des fichiers de configuration nécessaire à l'application. Comme le fichier de configuration pour l'envoi d'email et le fichier de configuration pour la connexion à la base de données.

L'application est conçue suivant différents Pattern Design, telle que le PD Fluent (pour chainer les méthodes), le PD Singleton (pour l'accès à l'application) et le PD Adapter.

Le polymorphisme est également utilisé au sein du programme. Notamment pour les candidats (qui peuvent être à la fois des Personnes et des Équipes.

B) Hibernate (ORM)

Afin de persister / sauvegarder les données de l'application en base de données, nous nous somme tourner vers l'utilisation d'un ORM. En effet cela permet de sauvegarder automatiquement les objets en base de donnée sans à devoir générer les requêtes SQL associés. De plus la récupération des relations entre entitées est simplifiée, l'ORM faisant toutes les requêtes nécessaires et les traitements automatiquement. Ainsi en quelques lignes de code, nous pouvons récupérer une masse de donnée sans difficulté.

Notre choix c'est instinctivement tourné vers l'ORM Hibernate. En effet c'est le plus utilisé aujourd'hui, il est simple d'utilisation et rapide à l'exécution. De plus une grande communauté l'utilisant, l'aide et la recherche de solutions dans son utilisation est grandement facilité et simplifié.

Lien Hibernate : http://hibernate.org/orm/

Pour configurer correctement Hibernate, merci de vous référer au chapitre de configuration ci-dessus.

D) Junit

Une des grandes problématiques lors du développement d'une application / d'un programme est l'apparition de nouveau Bug lors de la résolution d'autres Bug. Afin de garder un code toujours fonctionnel et opérationnel, l'utilisation des tests unitaires nous a été d'une grande aide. En effet ces tests permettant de tester le programme à chaque démarrage et de se stopper immédiatement si un test échoue.

Cela permet de ne pas à avoir à re-tester 50 fonctionnalités à chaque modification d'un bout de code. Junit le faisant pour nous directement.

E) CommandLine

La création de menu en version console est une tâche longue en java. En effet il faut contrôler les entrées dans le programme et cela peut parfois mener à certains bugs. Pour rendre cette tâche plus aisée, nous avons utilisé une librairie permettant de simplifier la création de menu en Java. L'on peut ainsi imbriquer des menus, faire des sélections dans des listes et réaliser des actions beaucoup plus rapidement, plus proprement et ce avec un contrôle des saisies automatiques.

Lien CommandLine : https://github.com/alexandreMesle/CommandLine/

Outils

A) Eclipse

Le développement d'un programme en Java nécessite un environnement complet. Le moyen le plus simple de disposer de cet environnement est d'installer le logiciel Eclipse (https://www.eclipse.org/) édité par la Fondation Éclipse. Ce logiciel est un véritable IDE, en effet il dispose d'une véritable mitraillette d'outils. Il intègre Git (voir dans le chapitre suivant), il est cross-plateforme (identique sur Windows, Mac et Linux) et optimisé pour Java.

Il permet également de développer des programmes Web et des applications mobiles androids. Il permet l'utilisation et l'édition d'une multitude de fichiers et d'extensions. Il est extensible (on peut y rajouter des plugins) et un de ces grands points fort, c'est qu'il est énormément utilisé, par des particuliers comme des professionnels. Il est adapté aux débutants comme aux expert Java.

B) Git

Projet à plusieurs est souvent synonyme de conflits et donc d'erreur de compilation. Pour pallier ce problème l'utilisation d'un logiciel de versionning est plus que recommandé (voire obligatoire). Le plus utilisé étant Git (et donc GitHub qui se base dessus), il est facile de travailler à plusieurs sur un projet. Les modifications de chacun ne rentre pas en conflit avec les modifications des autres et cela permet de garder un historique des versions du projet en cas de découverte plus tard d'un bug majeur.

L'utilisation de git permet à tous les développeurs d'un même projet de bénéficier automatiquement (ou presque) du code le plus à jour possible, et ce sans devoir s'envoyer le code par mail ou via un drive. Git offre également la possibilité de pouvoir « cloner » un projet sur sa machine, il offre la possibilité de créer un autre projet se basant sur un projet déjà existant, il permet d'envoyer des mises à jour de code à un développeur. Les possibilités avec Git son nombreuse et permettent de résoudre bien des problèmes lors de travaux en groupe (mais aussi lors des travaux perso !!).

C) JMérise

Afin de modéliser notre base de données, nous avons utilisé le logiciel JMérise. En effet celui-ci permet de schématiser entièrement la structure d'une BDD, les relations entre les tables, les cardinalités (types de relation) et offre un tas d'autres fonctions supplémentaire. Comme l'exportation d'un script SQL, ou encore la modification en temps réel d'une base de données.