CHEBBAH Nassim

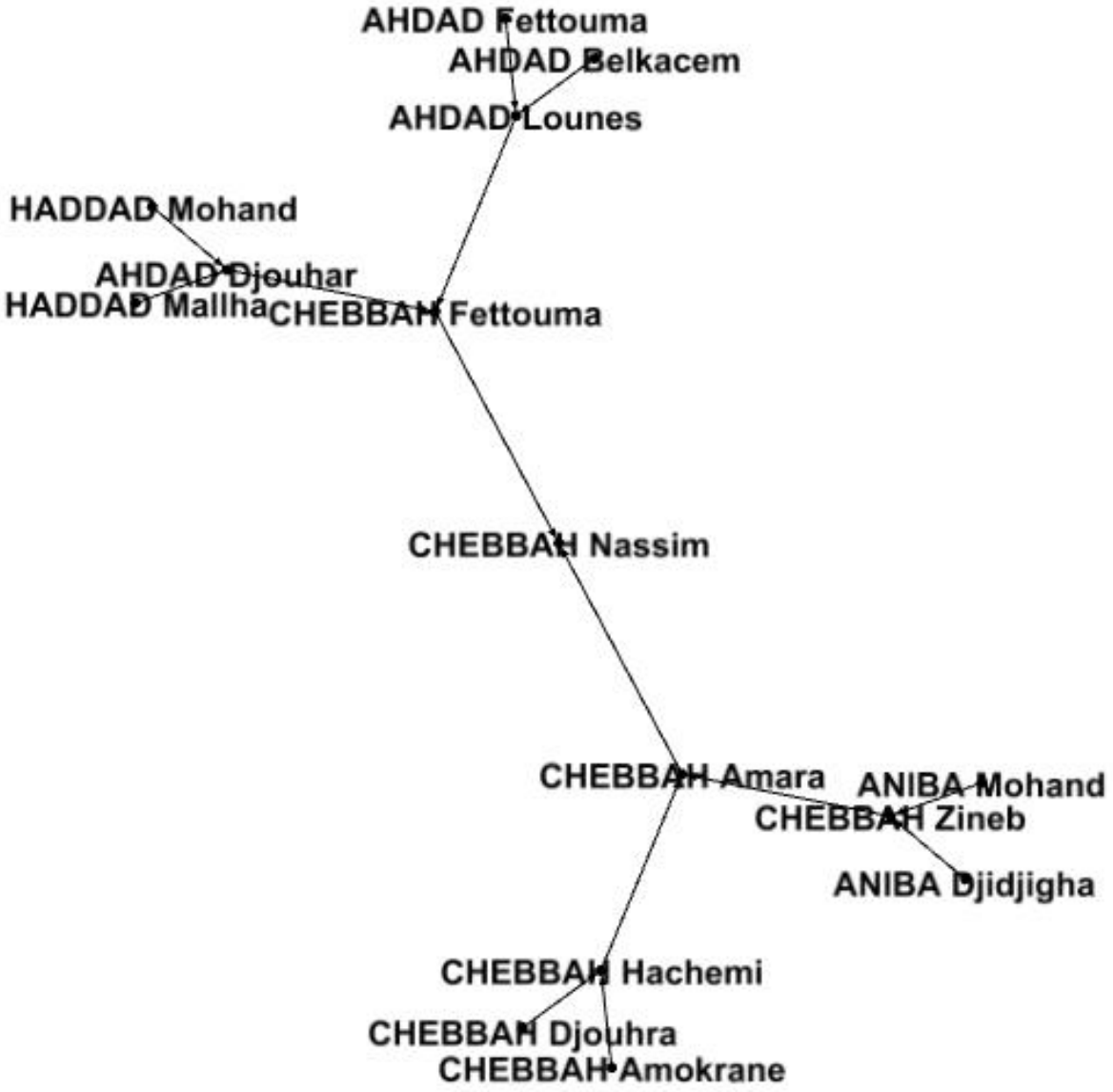
LENHOF Edgar

Application de gestion d'arbre généalogique :

gIGNealogie

Rapport de Synthèse

Notre application permet la gestion d'une base de données généalogiques. Il est possible d'enregistrer, modifier, contrôler et consulter des informations. Il est bien sûr possible de générer l'arbre généalogique d'une personne. L'application est livrée avec une base de données SQLite afin d'assurer la portabilité. Nous avons fait en sorte de rendre l'application la plus intuitive possible.



Arbre généalogique généré par gIGNealogie

I. Fonctionnement

Structure du code

Les classes du code sont divisées selon 3 *packages* : Personne, Relation et Source. A cela s'ajoute un *package* Utile.

Package **Personne** : contient les classes PersonneCivile, PersonneCatholique, Metier, Temoin. Ces classes instancient et enregistrent dans la base de données les informations relatives à une personne.

Package **Relation** : contient les classes Fratrie, Mariage, Parente, Parrainage. Ces classes instancient et enregistrent dans la base de données les informations relatives à une relation entre deux personnes.

Package **Source** : contient les classes SourcePersonne, SourceRelation ainsi que les classes qui héritent de celles-ci. Ces classes instancient et enregistrent dans la base de données les informations relatives à un document source à propos d'une personne (*ex : Acte de naissance*) ou d'une relation (*ex : Livret de famille*).

Package **Utile** : contient le Main, des classes purement utilitaires (Scan, MyConnection, …), les classes d'affichage graphique des document sources, la classe Arbre qui permet de générer l'arbre généalogiques ainsi que la classe Requete qui contient des requêtes toutes prêtes pour l'utilisateur.

Exécution

L'utilisateur n'a qu'à lancer le fichier *gIGNealogie.bat.* S'il souhaite passer par Eclipse il suffit d'exécuter le script de la classe Main. Une interface console viendra ensuite le guider pendant tout le processus.

II. Organisation des développements

Analyse

Il a d'abord été nécessaire dans un premier temps de cerner tous les enjeux de notre sujet et de réfléchir à l'architecture de notre application. Rapidement nous avons implémenté l'architecture globale du code afin de concilier une vision théorique du sujet avec des aspects de programmation plus pratiques. Puis nous nous sommes attelés à la rédaction du rapport d'analyse et à la modélisation UML. Prendre un temps suffisant sur la partie d'analyse nous aura permis de suivre un cap précis sans avoir à en dévier. Nos schémas UML du début restent donc en adéquation avec notre code final.

Mise en relation avec la base de données

Nous avons ensuite gérer la mise en relation du code java avec la BDD aux travers des méthodes *save*( ) et *create*( ). Toutefois, nous avons rencontré un problème avec la connexion à la BDD. Nous avons dû poursuivre notre travail « à l'aveugle » pendant une semaine sans pouvoir tester notre code afin de ne pas prendre trop de retard sur le planning. Nous avons décidé de nous répartir chacun sur une tache différente pour la suite.

Implémentation des fonctions de requête et de contrôle

Edgar s'est attelé à l'implémentation des méthodes de requêtes toutes faites et en interaction console avec l'utilisateur. Il a aussi fallu implémenter les différents tests de cohérence sur les informations en entrée.

Génération de l'arbre généalogique

Pendant ce temps, Nassim s'est occupé de la génération de l'arbre généalogique d'une personne à partir des données enregistrées dans la table *relation* de la BDD. Il a aussi géré sa visualisation sous *gephi.*

Mise en commun et débogage

Lors des dernières séances nous avons préféré nous concentrer sur la mise en commun de nos travaux, des tests de notre code afin de soulever les bugs encore présents et sur l'améliorations de l'ergonomie de notre application.

Derniers réglages et rendu final

Comme il nous restait un peu de temps, nous avons ajouter une fonction d'affichage graphique des documents sources. Enfin nous avons exporter notre projet sous forme d'exécutable et y avons ajouter un logo.

III. Bilan

Travail réalisé – Possibilités de l'application

L'application est rendue intuitive grâce à l'interface console qui demande à l'utilisateur de rentrer les informations dont elle a besoin. En cas d'incohérence elle le signalera à l'utilisateur et, dans certains cas, lui demandera confirmation.

Notre application permet d'enregistrer dans une base de données des informations (civiles, religieuses, métier) relatives à des personnes ainsi que les relations qui les unissent entre elles (mariage, parenté, fratrie, …). Il est ensuite possible de modifier ou d'afficher ces informations.

Lors de la saisie des informations, des contrôles sont effectués. Le programme vérifiera notamment si les dates des évènements vécus par une personne ont bien lieu de son vivant, si l'âge auquel elle a des enfants n'est pas improbable (*ex : avoir un enfant avant 15 ans*) ou si tout simplement l'on parle bien de la même personne (l'application gère le fait d'avoir plusieurs personnes ayant le même nom-prénom).

Ces informations peuvent être attestées par des documents sources dont les caractéristiques (emplacement, nature du document, ce qu'elle renseigne) sont aussi enregistrées dans la BDD. Toutefois, il est nécessaire si l'on souhaite enregistrer un document source de venir le copier dans le dossier /gIGNealogie/sources. Une fois enregistré, on peut demander à consulter le document qui va s'afficher dans une fenêtre.

Enfin, l'application permet de créer l'arbre généalogique complet (jusqu'au dernier ancêtre connu) d'une personne donnée en utilisant les attributs de la table relation. Il est sous forme de graphe où sont retracées les relations parents-enfants ascendantes (seulement les ancêtres directs). Il est possible de visualiser et d'organiser proprement le graphe avec *gephi.*

Améliorations à apporter

Il y a encore de nombreux points qu'il est possible d'améliorer sur notre application.

Tout d'abord au niveau des méthodes de contrôle, on pourrait ajouter des tests sur l'âge maximal d'une personne, sur les relations incestueuses (un mariage avec un parent ou au sein d'une fratrie), sur les mariages multiples, … Ces fonctions n'ont rien de compliqué par rapport aux tests déjà implémentés mais nécessitent juste du temps en plus.

Nous avons également quelques problèmes au niveau de l'affichage avec les accents et autres caractères spéciaux.

Ensuite on pourrait penser à ajouter des relations supplémentaires soit en créant de nouvelles classes soit en utilisant les relations déjà existantes et en les mettant en commun (*ex : un oncle n'est jamais que le frère d'un parent*).

On pourrait aussi penser à ajouter les photos de gens dans la BDD et les afficher par la suite dans l'arbre généalogique.

Il pourrait aussi être intéressant de développer le code de création de l'arbre généalogique afin d'avoir aussi les relations fraternelles et les relations de parenté descendantes. C'est probablement la partie la plus complexe à développer si l'on veut gérer la cohérence et la redondance d'informations.

Enfin la réalisation d'une véritable interface graphique serait tout à fait envisageable dans l'optique de rendre l'application encore plus ergonomique.

Ce que nous avons appris

Durant ce projet nous avons appris à mener un projet à bien du début jusqu'à la fin. De ce fait il nous a été nécessaire d'assurer la livraison d'un code dont les fonctions de base tournent sans encombre, même si nous l'estimons incomplet sur certains points.

Ce projet nous aura aussi permis de vraiment intégrer les concepts d'orienté objet dans un cas pratique et global. Edgar a appris à ne plus faire d'erreurs dans ses requêtes SQL. Nous avons pu découvrir de nouveaux attraits de la programmation java (Date, affichage graphique, …). Nous avons également appris à créer une exécutable java et aussi à utiliser d'autres outils comme *gephi.* Nous avons enfin appris à générer de la javadoc au format HTML.