



RAPPORT DE PROJET

Génie Logiciel

A l'intention de : Mme Zaouche Djaouida et Mme Dounia Awad

SOMMAIRE

Introduction_	3
Présentation Générale du projet	3
Outils et technologies utilisées	3
1 Organisation	4
1.1 Organisation de l'équipe	4
1.2 Planning de réalisation	5
2 Réalisation	6
2.1 Classes, méthodes et difficultés rencontrées	6
2.2 Résultats obtenus	9
2.3 Limites fonctionnelles	13
3 Conclusion	14
Bilan	14

Introduction

Présentation générale du projet

Ce projet consiste à créer une application Java permettant de créer, modifier, visualiser et interagir avec un arbre généalogique interactif. Conçue dans le cadre de notre formation en informatique, cette application modélise les relations familiales entre individus tout en proposant une gestion fine des droits d'accès et des informations personnelles.

Chaque utilisateur peut ajouter des personnes, créer des liens de parenté, interagir avec d'autres utilisateurs par mail, et visualiser l'arbre de façon textuelle ou graphique. L'application gère aussi différents niveaux de visibilité des données selon le profil (utilisateur ou administrateur).

Lien Git du projet : <https://github.com/Azluc/Projet-Arbre-Genealogique->

Outils et technologies choisis

Le projet repose notamment sur les technologies et outils suivants :

- Java (JDK) : langage principal du projet.
- SceneBuilder : pour construire l'interface graphique.
- Staruml : pour la modélisation UML (diagrammes de classes).
- SQL (via fichiers .sql) : pour simuler une base de données relationnelle.
- Jakarta : une bibliothèque utilisée pour gérer l'envoi de mail.
- FakeSMTP : qui est un serveur de messagerie local utilisé pour simuler l'envoi de mails.

Ce choix d'outils nous a permis de travailler dans un environnement structuré, tout en laissant une certaine flexibilité à chacun pour progresser en autonomie sur des composants spécifiques du projet.

1 Organisation

1.1 Organisation de l'équipe

Notre équipe était composée de cinq membres qui ont travaillé de manière complémentaire tout au long du projet. Les tâches ont été réparties selon les affinités et compétences de chacun, tout en veillant à collaborer sur les étapes clés pour garantir la cohérence de l'ensemble :

- La conception de l'entièreté de l'interface graphique avec SceneBuilder a été prise en charge par Lucas, qui s'est également assuré du bon fonctionnement des éléments graphiques.
- La structure de l'arbre généalogique ainsi que toutes les classes du package « com.cytech.classseProjet », « com.cytech.gestionBDD » et « com.cytech.fenetres » ont été réalisée par Lucas.
- Les tests console, la rédaction de la documentation technique (README, rapport, Javadoc) ainsi que le diagramme UML ont été réalisés par Lucas, Charf Eddine, Haitam, Saïd et Evan.
- L'intégration et la gestion des interactions avec la base de données ont été assurées par Lucas.
- La gestion des relations de parenté (parent-enfant, frère-sœur, union) a été développée par Lucas.

Le diagramme UML finale se trouve dans le dépôt du projet sous le nom « UML.pdf ».

1.2 Planning de réalisation

Le projet s'est déroulé sur trois semaines, avec une progression logique de la conception à la finalisation. Voici les grandes étapes réalisées chaque semaine :

Semaine 1 : Démarrage et fondations

- Réunion de lancement du projet.
- Mise en place de l'uml

Semaine 2 : Développement des fonctionnalités avancées

- Implémentation des classes de l'arbres généalogique
- Création de l'ihm

Semaine 3 : Intégration, tests et finalisation

- Réalisation de l'interface graphique par Lucas (utilisation de SceneBuilder)
- Intégration complète de l'ihm avec les classes entre les classes métiers pour pouvoir générer l'arbre généalogique en console et graphique.
- Implémentation des classes de gestion de bases de données
- Ajout de tests unitaires
- Ajout de requêtes

2 Réalisation

Nous allons partager les différentes étapes de réflexion et de réalisation de chaque partie.

2.1 Classes, méthodes et difficultés rencontrées

Tout d'abord, nous nous sommes appuyés sur le diagramme de classes UML pour débiter la programmation sur Eclipse. La première étape a été la création du package « `com.cytech.classeProjet` », dans lequel nous avons intégré toutes les classes représentées dans notre diagramme UML.

Nous avons ensuite défini, pour chaque classe, ses attributs, ses constructeurs, ainsi que les méthodes d'accès (getters) et de modification (setters). Afin de mieux organiser notre code, nous avons structuré notre projet en quatre packages distincts :

- « `com.cytech.classeProjet` » : contient les classes principales (modèle métier) de notre projet.
- « `com.cytech.classeTestsUnitaires` » : regroupe les classes dédiées aux tests unitaires.
- « `com.cytech.gestionBDD` » : contient les classes de gestion de la base de données.
- « `com.cytech.fenetres` » : regroupe toutes les fenêtres de l'IHM ainsi que les classes associées aux fonctionnalités graphiques.

Mise en place de l'arbre généalogique

L'implémentation de l'arbre généalogique a représenté un véritable défi. Il fallait concevoir une structure permettant l'ajout cohérent de personnes, tout en assurant l'intégrité de l'arbre à chaque modification. Par exemple, il est impossible d'ajouter un enfant à un parent si la date de naissance de l'enfant précède celle du parent. De même, nous devons éviter la création de cycles, ce qui aurait compromis la validité de l'arbre.

Pour cela, nous avons établi plusieurs règles de cohérence :

- **Unicité des personnes** : une personne est unique dans l'arbre ; on ne peut donc pas retrouver deux personnes avec le même prénom et nom.

- **Ajout par relation directe** : l'ajout d'une nouvelle personne se fait uniquement par une relation directe à une personne existante — soit horizontalement (frère/sœur, union), soit verticalement (parent/enfant). Par exemple, pour ajouter un oncle à la racine, il faut d'abord créer son parent, puis ajouter un frère à ce parent.

- **Cohérence d'âge** : lors de l'ajout d'un enfant, une différence minimale de 18 ans est exigée avec le parent pour garantir la plausibilité des données.

Implémentation de la classe `ArbreGenealogique`

L'implémentation principale, réalisée par Lucas, se trouve dans la classe `ArbreGenealogique`. Elle contient quatre attributs essentiels :

- **`personneRacine`** : l'utilisateur, point de départ de l'arbre.
- **`idArbre`** : identifiant unique correspondant au code privé de l'utilisateur.
- **`liensParente`** : une liste des liens de parenté pour chaque personne.
- **`personnes`** : un ensemble (`Set<Personne>`) contenant toutes les personnes rattachées à l'arbre.

Cette architecture nous permet de parcourir l'arbre, de gérer dynamiquement les relations familiales et d'assurer la cohérence des données à chaque ajout ou suppression.

Gestion de l'envoi de mails

Un autre défi important a été la mise en place de l'envoi de mails. Dans un premier temps, nous avons utilisé la bibliothèque `Jakarta Mail`. Cependant, cette solution repose sur un service Google payant (`Google Cloud`), nous avons donc abandonné cette idée.

Pour contourner cette limitation, nous avons opté pour `FakeSMTP`, une solution locale de messagerie qui permet de simuler l'envoi d'emails de manière simple et efficace.

Classe de test

La partie test est organisée dans le package `com.cytech.classeTestsUnitaires`. Ce dernier contient l'ensemble des classes de test liées aux modules de données et de gestion.

Nous avons importé les classes nécessaires depuis les autres packages (com.cytech.classeProjet) pour faciliter l'accès aux éléments à tester. Les tests incluent :

- Tests des constructeurs : vérifient la bonne initialisation des objets lors de leur création.
- Tests des getters : s'assurent que les valeurs retournées correspondent aux attributs attendus.
- Tests des setters : garantissent que les modifications apportées aux attributs sont bien prises en compte.
- Tests de la base de données : valident l'enregistrement, la mise à jour et la lecture des données depuis notre base MySQL.

Interface graphique

Le package com.cytech.fenetres regroupe toutes les classes dédiées à l'interface graphique de notre application. Il comprend les différentes fenêtres de navigation ainsi que les classes associées aux fonctionnalités visuelles telles que la gestion des clics, l'affichage dynamique de l'arbre généalogique ou encore les formulaires d'ajout de personnes.

Pour concevoir l'interface, nous avons utilisé l'outil SceneBuilder, qui permet de créer visuellement les fichiers FXML associés aux différentes fenêtres de l'application. Cela nous a permis de gagner en rapidité et en clarté lors de la conception de l'IHM. L'interface permet ainsi à l'utilisateur de :

- Visualiser dynamiquement l'arbre généalogique,
- Ajouter ou modifier des personnes via des formulaires,
- Créer des relations de parenté cohérentes,
- Être notifié par mail après certaines actions (via FakeSMTP).

Dans l'interfac graphique, nous avons modéliser les relations frères-sœurs par une droite en pointillé. Les autres relations sont représentées par des droites sans pointillées

2.2 résultats obtenus

Voici un aperçu du travail fourni :

Page d'accueil :

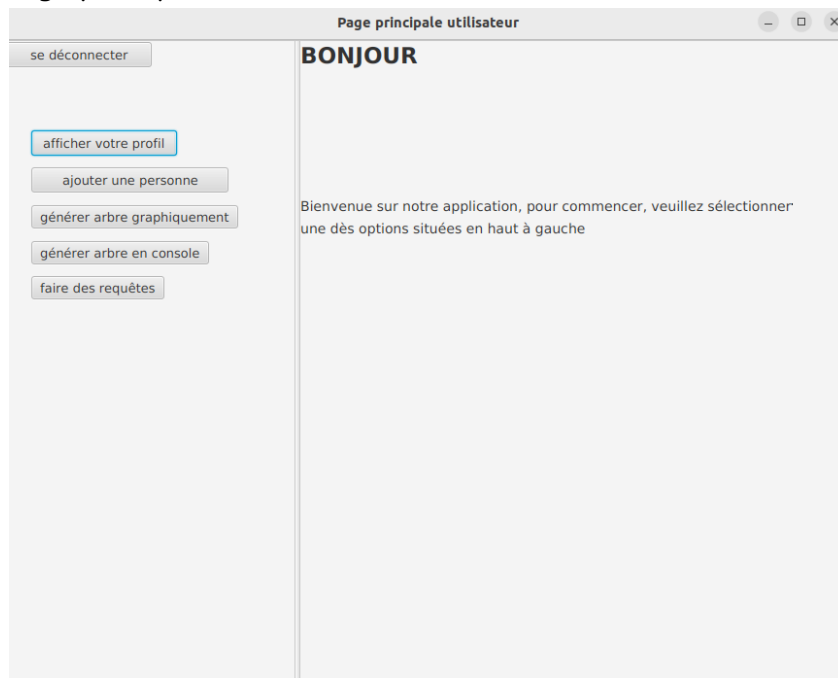
The screenshot shows a web browser window titled 'Accueil'. The page is divided into two main sections. On the left, the title 'Site Arbre Généalogique' is displayed, followed by a welcome message: 'Bienvenue sur notre application d'arbre généalogique'. Below this, it lists the features of the application: 'Dans cette application, vous pourrez : -Créer et ajouter des membres de votre famille dans l'arbre -Partager et échanger vos photos et vidéos avec votre famille.' On the right, under the heading 'Vous êtes ?', there are three buttons: 'Utilisateur non inscrit' (highlighted in blue), 'Utilisateur', and 'Administrateur'.

Page d'inscription :

The screenshot shows a web browser window titled 'Accueil' displaying the 'INSCRIPTION' page. The page contains a registration form with the following fields and buttons: 'Votre prénom :', 'Prénom'; 'Votre nom :', 'Nom'; 'Votre email :', 'email'; 'Votre nationalité :', 'Nationalité'; 'Votre date de naissance :', 'Date de Naissance' (with a calendar icon); 'Votre adresse :', 'adresse'; 'Votre numéro de téléphone :', 'Numéro téléphone'; 'Numéro de sécurité sociale :', 'Numéro sécurité sociale'; 'Choisir Photo', 'Pièce numérique'; 'Choisir pièce d'identité', 'Pièce d'identité'; 'S'inscrire' (highlighted in blue); and 'Retour'.

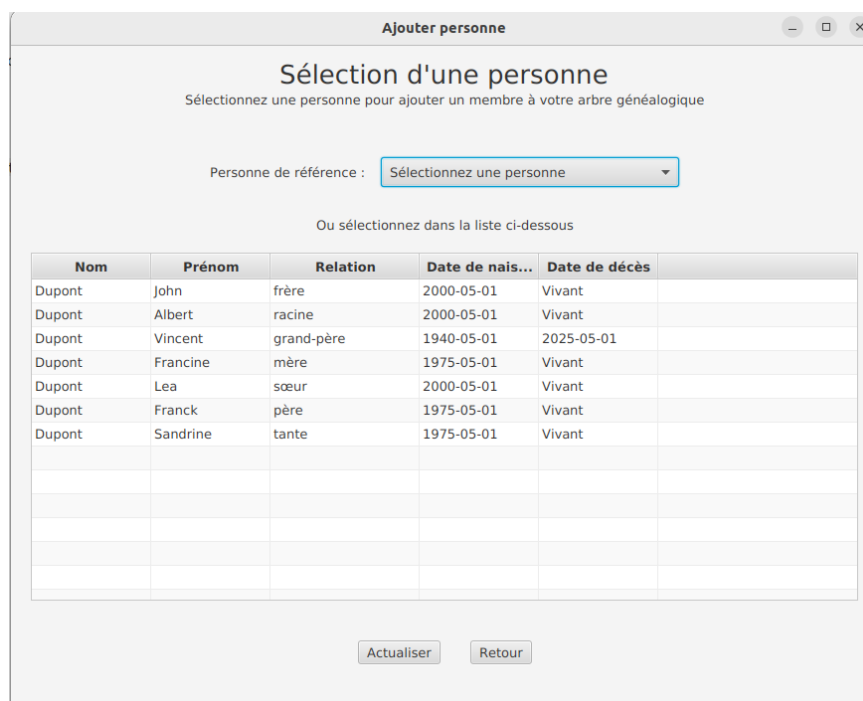
Page Administrateur :

Page principale utilisateur :



The screenshot shows a web application window titled "Page principale utilisateur". On the left side, there is a vertical menu with five buttons: "se déconnecter", "afficher votre profil", "ajouter une personne", "générer arbre graphiquement", and "générer arbre en console". Below these is a button labeled "faire des requêtes". The main content area on the right has a large heading "BONJOUR" and a paragraph of text: "Bienvenue sur notre application, pour commencer, veuillez sélectionner une dès options situées en haut à gauche".

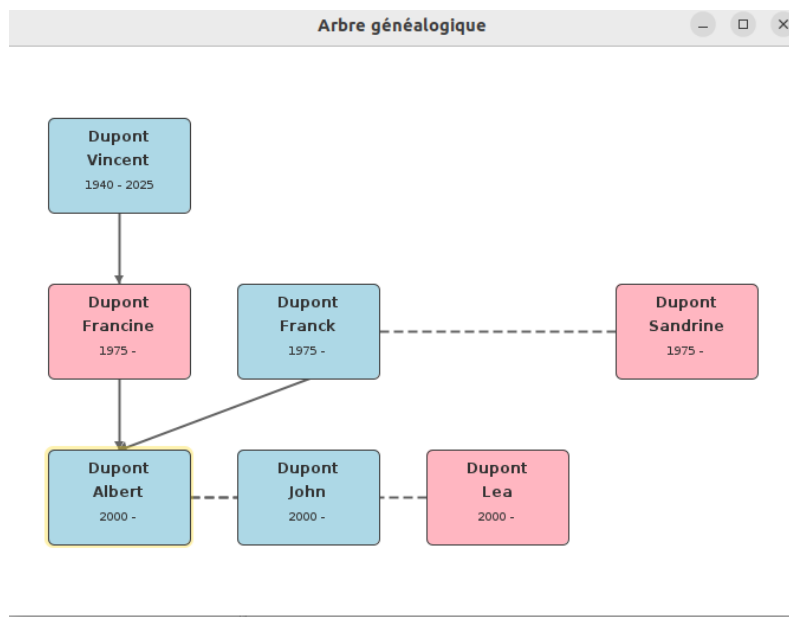
Exemple d'arbre généalogique généré représentant la famille Dupont:



The screenshot shows a web application window titled "Ajouter personne". The main heading is "Sélection d'une personne", followed by the instruction "Sélectionnez une personne pour ajouter un membre à votre arbre généalogique". Below this, there is a label "Personne de référence :" and a dropdown menu with the text "Sélectionnez une personne". Underneath, it says "Ou sélectionnez dans la liste ci-dessous". A table with five columns is displayed: "Nom", "Prénom", "Relation", "Date de nais...", and "Date de décès". The table contains six rows of data for the Dupont family, followed by four empty rows. At the bottom of the page, there are two buttons: "Actualiser" and "Retour".

Nom	Prénom	Relation	Date de nais...	Date de décès
Dupont	John	frère	2000-05-01	Vivant
Dupont	Albert	racine	2000-05-01	Vivant
Dupont	Vincent	grand-père	1940-05-01	2025-05-01
Dupont	Francine	mère	1975-05-01	Vivant
Dupont	Lea	sœur	2000-05-01	Vivant
Dupont	Franck	père	1975-05-01	Vivant
Dupont	Sandrine	tante	1975-05-01	Vivant

Génération d'arbre graphique :



Génération d'arbre en console :

```
History Git Staging Git Reflog Console x Install Java 24 Support Eclipse IDE for Java Developers 2025-06 M2
Main (9) [Java Application] /usr/lib/jvm/java-21-openjdk-amd64/bin/java (May 23, 2025, 11:11:04 PM elapsed: 0:07:42) [pid: 5652]

Liste des liens de parenté dans l'arbre :
Relations dans l'arbre depuis la racine :
Dupont John est frère de la racine.
Dupont Albert est la racine.
Dupont Vincent est grand-père de la racine.
Dupont Francine est mère de la racine.
Dupont Lea est sœur de la racine.
Dupont Franck est père de la racine.
Dupont Sandrine est tante de la racine.
```

2.3 Limites fonctionnelles

- Pas de persistance des données après avoir généré l'arbre généalogique graphiquement : l'arbre est temporaire (pas de sauvegarde via base de données).
- Fonctionnalités issues du cahier des charges non réalisées comme (La fonction permettant de consulter l'arbre d'un autre utilisateur, le nombre de requêtes restent limitées)

Conclusion

Bilan

Au cours de ce projet, nous avons conçu et développé une application Java permettant la gestion dynamique d'un arbre généalogique, intégrant à la fois une structure de données orientée objet, une interface graphique, et des règles de gestion des relations familiales. Nous avons mis en place une application fonctionnelle avec un système d'utilisateurs et d'administrateurs, des fonctionnalités avancées comme la création de liens de parenté (frère-soeur, parent-enfant), la vérification de la cohérence lors de l'ajout de personnes ainsi qu'une interface graphique réalisée avec SceneBuilder permettant une utilisation simple et intuitive pour les utilisateur. Bien que nous ayons réussi à générer un affichage graphique et en console et fait quelques requêtes, le projet reste limité, nous ne sommes pas parvenus au résultat escompté.