



Rapport d'élève ingénieur
Projet de 2^{ème} année
Filière : Génie logiciel et systèmes informatiques

Réalisation d'un logiciel de généalogie

Ancestrel généalogie

Présenté par : **DUMAS Florian**
VILLEDIEU DE TORCY Mathieu

Responsable ISIMA : YON Loïc

Date de la soutenance : **14 mars 2022**

Durée du projet : **60h**

Remerciements

Nous remercions notre tuteur de projet, M. Loïc Yon, pour nous avoir encadrés durant ce projet, avoir pris le temps de répondre à nos questions et d'avoir fait un point avec nous sur l'avancée de notre projet.

Table des figures

| | | |
|----|---|----|
| 1 | Exemple d'arbre : numérotation descendante d'Aboville | 2 |
| 2 | Diagramme de cas d'utilisation | 4 |
| 3 | Représentation de la numérotation Sosa-Stradonitz | 6 |
| 4 | Diagramme de Gantt prévisionnel | 8 |
| 5 | Diagramme de Gantt réel | 9 |
| 6 | Diagramme de classes | 10 |
| 7 | Diagramme de base de données relationnelle | 13 |
| 8 | Contrôle d'utilisateur : carte d'une personne | 15 |
| 9 | Page d'accueil | 16 |
| 10 | Page de création d'une personne ou d'un arbre | 17 |
| 11 | Page de chargement d'un arbre | 17 |
| 12 | Page de visualisation de l'arbre | 18 |
| 13 | Page de détails / Modification d'une personne | 18 |

Résumé

Dans le but de simplifier la création d'un **arbre généalogique**, nous souhaitons développer une application Windows permettant de mettre en forme un arbre généalogique ascendant. Nous nous sommes inspirés de méthodes déjà existantes telles que la **numérotation ascendante Sosa-Stradonitz** permettant de situer chaque personne dans l'arbre. L'objectif de l'application est de renseigner les ascendants biologiques d'une personne en leur associant diverses informations (noms, prénoms, date de naissance/décès, lieu de naissance, images) à travers une interface simple d'utilisation et accessible. Les données sont conservées en local par l'utilisateur et ne sont pas partagées.

Cette application est développée en **.NET/C#, XAML** sous la plateforme de développement **WPF** pour la partie graphique et **Microsoft SQL Server** pour la base de données. Nous avons utilisé l'environnement de développement **Visual Studio** sur Windows et le gestionnaire de version Git.

À ce jour, nous avons terminé les fonctionnalités de base de l'application permettant de structurer un arbre, de modifier les personnes qui le compose avec un système de persistance dans une base de données. Le projet a été conçu de façon à faciliter le maintien de l'application, l'ajout d'extension ou encore le changement du type de sauvegarde des données.

Mot-clés : .NET/C#, Arbre généalogique, Numérotation ascendante (Sosa-Stradonitz), XAML, WPF, Microsoft SQL Server, Visual Studio

Abstract

To simplify the creation of a **family tree**, we wish to develop a Windows application to create an ascending family tree. We were inspired by existing methods such as the **Sosa-Stradonitz ascending number system**. It helps locate each person in the tree. The application's goal is to fill in the biologic ancestors of a person by associating them with various information (last name, first name, date of birth/death, birthplace, images) through a user-friendly interface. Data is kept in the user's computer and is not shared.

This application is developed in **.NET/C#, XAML** under the **WPF** development platform for the graphic part and **Microsoft SQL Server** for the database. We used the **Visual Studio** development environment on Windows and Git as version manager.

To date, we have completed the main functionalities of the application allowing the user to structure the tree and modify the people in it with a persistence system in the database. The project has been designed to facilitate the application's maintenance, the release of future extensions or even data storage modification.

Keywords : .NET/C#, Family tree, Ascending number system (Sosa-Stradonitz), XAML, WPF, Microsoft SQL Server, Visual Studio

Table des matières

| | |
|---|-----------|
| Remerciements | i |
| Table des figures | ii |
| Résumé | iii |
| Abstract | iii |
| Table des matières | iv |
| Introduction | 1 |
| 1 Présentation du sujet | 2 |
| 1.1 Comment faire un arbre généalogique ? | 2 |
| 1.2 Définition des besoins | 3 |
| 1.2.1 L'objectif | 3 |
| 1.2.2 Les fonctionnalités attendues | 4 |
| 1.3 Représentation de l'arbre | 6 |
| 1.3.1 Numérotation ascendante Sosa-Stradonitz | 6 |
| 2 Conception et développement | 8 |
| 2.1 Diagramme de GANTT | 8 |
| 2.1.1 Diagramme de Gantt prévisionnel | 8 |
| 2.1.2 Diagramme de Gantt réel | 9 |
| 2.2 Conception | 10 |
| 2.2.1 Outils utilisés | 10 |
| 2.2.2 Structure | 10 |
| 2.2.3 Modèle | 11 |
| 2.3 Persistance des données | 12 |
| 2.3.1 Interfaces | 12 |
| 2.3.2 Base de données Microsoft SQL Server | 13 |
| 2.4 Partie graphique / Interface utilisateur | 14 |
| 3 Résultats | 16 |
| 3.1 Architecture logicielle | 16 |
| 3.2 Fonctionnement | 16 |
| 3.3 Etat du produit | 19 |
| Conclusion | 20 |
| Bibliographie | v |
| Glossaire | vi |

Introduction

Aujourd'hui le domaine de la [généalogie](#) est dominé par l'église de Jésus-Christ des Saints des Derniers Jours, appelée plus simplement les Mormons. Ceux-ci possèdent le quasi-monopole sur les applications liées à la [généalogie](#). Dans le but de donner une alternative aux possibles usagers de ce genre de logiciel, nous avons décidé de développer une application permettant de créer un arbre généalogique.

Cette application, nommée *Ancestrel*, doit permettre aux utilisateurs de créer un arbre généalogique ascendant en ajoutant des personnes suivant des liens de parentés.

Il existe plusieurs façons de construire et de représenter un arbre généalogique. L'application doit se concentrer sur une représentation ascendante.

Elle doit permettre aux utilisateurs de renseigner différentes informations sur leurs ancêtres (noms, prénoms, date de naissance/décès, lieu de naissance, ...). Ces informations ne doivent pas être partagées automatiquement puisque celles-ci peuvent être considérées comme personnelles par la personne souhaitant créer un arbre. L'utilisateur doit donc être le seul à avoir accès à ces données.

L'interface doit rester sobre et ne pas être surchargée d'informations pour ne pas être rebutante afin que l'application reste facile d'utilisation.

Une carte doit aussi être présentée à l'utilisateur avec la localisation du lieu de naissance de ces ancêtres pour qu'il puisse avoir une idée de ses origines géographiques.

Dans quelle mesure est-il possible de développer une application permettant de gérer un arbre généalogique répondant à des critères d'ergonomie et de simplicité d'utilisation tels qu'énoncés ?

Dans un premier temps, nous introduirons et analyserons le travail à faire, puis nous nous attacherons sur le processus de développement. Enfin, nous discuterons des résultats obtenus.

1 Présentation du sujet

1.1 Comment faire un arbre généalogique ?

La méthode de création d'un arbre généalogique [1] nécessite d'effectuer des recherches dans les archives municipales afin de remonter les informations de chaque ancêtre. En effet, ces archives contiennent divers documents, tels que les actes de naissance et de décès ou de mariage par exemple, qui sont indispensables en [généalogie](#).

De plus, certains fichiers peuvent être conservés dans les archives de différents cultes, comme les actes de baptême ou de mariage religieux.

Enfin les états de service militaire ou autres documents professionnels peuvent être de bonnes sources d'informations.

L'ensemble de ces renseignements permettent d'acquérir les informations nécessaires pour construire un arbre généalogique. Des photos peuvent aussi être associées à des individus dans un arbre afin de le rendre plus complet.

Les arbres peuvent prendre différentes formes, ascendante ou descendante. Il est nécessaire d'utiliser une bonne méthode de numérotation afin d'identifier chaque personne. [2, 3]

Avec une numérotation descendante (cf. figure 1), la [généalogie](#) part d'un ancêtre afin de retrouver tous ses descendants, tandis qu'avec la numérotation ascendante (cf. figure 3), la [généalogie](#) part d'un individu afin de retrouver ses ancêtres.

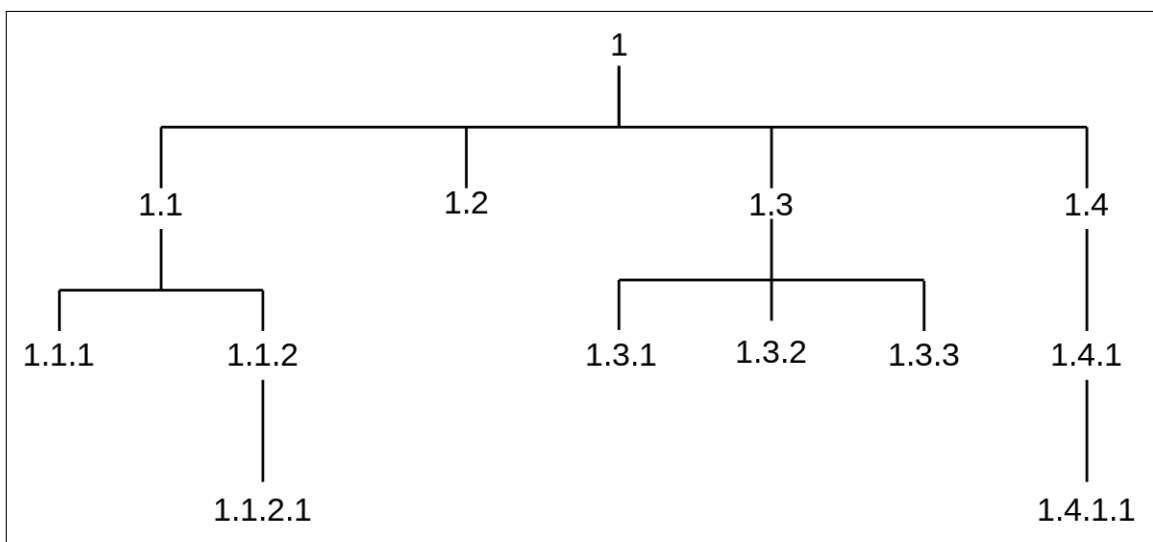


FIGURE 1 – Exemple d'arbre : numérotation descendante d'Aboville

1.2 Définition des besoins

1.2.1 L'objectif

L'objectif du logiciel est de permettre à un usager de créer un arbre généalogique, suivant une numérotation ascendante en ajoutant au fur et à mesure les parents biologiques de chaque personne.

Cet arbre prend comme point de départ le [de cujus](#), la personne à la base de l'arbre. En règle générale il s'agit de l'utilisateur lui-même, mais nous voulons aussi lui permettre de créer un arbre à partir de n'importe quel individu, il lui revient donc de remplir les informations concernant cette personne.

Les différents ascendants seront donc ajoutés en partant du [de cujus](#).

Il sera possible de renseigner des informations sur chaque personne lors de sa création et à posteriori lors de la consultation.

Si l'utilisateur crée une nouvelle personne sans renseigner d'informations sur celle-ci, elle sera ajoutée comme une personne inconnue.

L'utilisateur pourra également supprimer un individu dans l'arbre qui sera alors remplacé par une personne inconnue afin de conserver une certaine continuité dans l'arbre.

Les personnes inconnues dans l'arborescence permettront de garder la connexité de l'arbre .

Les différentes informations sur une personne que le logiciel doit prendre en compte sont :

- Nom d'usage
- Nom de jeune fille (Femme)
- Prénoms
- Date de naissance
- Date de décès
- Lieu de naissance
- Nationalité
- Documents : images, PDF
- Son père
- Sa mère

1.2.2 Les fonctionnalités attendues

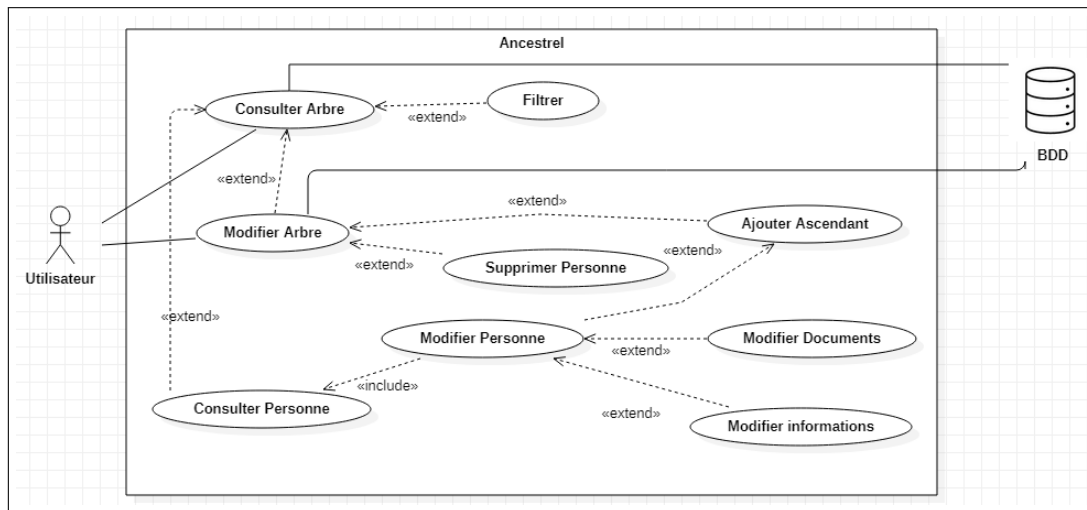


FIGURE 2 – Diagramme de cas d'utilisation

Le logiciel doit proposer un certain nombre de fonctionnalités dites essentielles afin de pouvoir être utilisable. Des extensions possibles pourront être implémentées suivant l'avancement du projet.

Fonctionnalités essentielles

Les fonctionnalités dites essentielles permettent à l'utilisateur de créer simplement un arbre en ajoutant différentes personnes (avec plus ou moins d'informations) suivant des liens de parentés.

Fonction 1 : Consulter un arbre

Résumé : L'utilisateur doit pouvoir consulter l'arbre chargé dans le logiciel. Il doit pouvoir déplacer l'arborescence pour visualiser une partie de l'arbre qui l'intéresse. Il doit aussi pouvoir visualiser l'arbre dans son ensemble.

Acteur : Utilisateur

Fonction 2 : Ajouter un ascendant

Résumé : L'utilisateur doit pouvoir ajouter des informations ou des documents à une personne lors de sa création. Il doit pouvoir également ajouter une personne en ne renseignant aucune information, celle-ci sera considérée comme inconnue mais permettra de garder le lien dans l'arbre.

Acteur : Utilisateur

Fonction 3 : Modifier un arbre

Résumé : L'utilisateur doit pouvoir modifier l'arbre, en ajoutant des ascendants à chaque personne.

Acteur : Utilisateur

Fonction 4 : Supprimer une personne

Résumé : L'utilisateur doit pouvoir supprimer une personne de l'arbre. Cette personne sera alors remplacée par une personne inconnue afin de garder la continuité dans l'affichage l'arbre.

Acteur : Utilisateur

Fonction 5 : Modifier une personne

Résumé : L'utilisateur doit pouvoir modifier les informations ou les images déjà enregistrées d'une personne. Définir une personne comme inconnue.

Acteur : Utilisateur

Extensions possibles

Ces fonctionnalités seront implémentées en fonction de l'avancée du projet et viendront enrichir le logiciel en apportant des actions supplémentaires telles que des affichages personnalisés ou l'exportation de données dans d'autres formats.

Extension 1 : Récupération de données [GEDCOM](#)

Résumé : Le logiciel doit être capable de lire et d'exploiter des données issues du format [GEDCOM](#).

Acteur : Utilisateur

Extension 2 : Exportation de données en [GEDCOM](#)

Résumé : Le logiciel doit être capable d'exporter les données dans le format [GEDCOM](#).

Acteur : Utilisateur

Extension 3 : Afficher sur une carte

Résumé : Le logiciel doit afficher sur une carte le nombre de personnes nées par villes / régions / pays / continents en fonction du zoom choisi par l'utilisateur.

Acteur : Utilisateur

Extension 4 : Filtrer l'affichage

Résumé : L'utilisateur doit pouvoir choisir, selon certains critères (sexe, parenté), les personnes affichées dans l'arbre.

Acteur : Utilisateur

1.3 Représentation de l'arbre

Il existe plusieurs types d'arbres généalogiques dépendant de numérotations différentes [2, 3] :

- La numérotation descendante :
 - numérotation descendante d'Aboville (*cf. figure 1*)
 - numérotation descendante Pélissier
- La numérotation ascendante :
 - numérotation ascendante Sosa-Stradonitz (*cf. figure 3*)

Nous sommes partis sur l'idée d'un arbre généalogique utilisant la **numérotation ascendante Sosa-Stradonitz** afin de permettre à l'utilisateur de créer son arbre en remontant sa [généalogie](#).

1.3.1 Numérotation ascendante Sosa-Stradonitz

La *numérotation Sosa-Stradonitz* est une numérotation ascendante facilitant la représentation et création d'un arbre généalogique.

Elle est développée en 1590 par Eyzinger, puis reprise en 1676 par Jérôme de Sosa. En 1898, elle est améliorée et utilisée par l'un des précurseurs de la [généalogie](#) moderne, Stephan Kekulé von Stradonitz dans son ouvrage Ahnentafel-Atlas. [2]

Cette numérotation est également connue sous le nom de *système de numérotation Ahnentafel*.

Principe

Cette numérotation attribue un numéro unique et prédéfini à tous les ascendants directs.

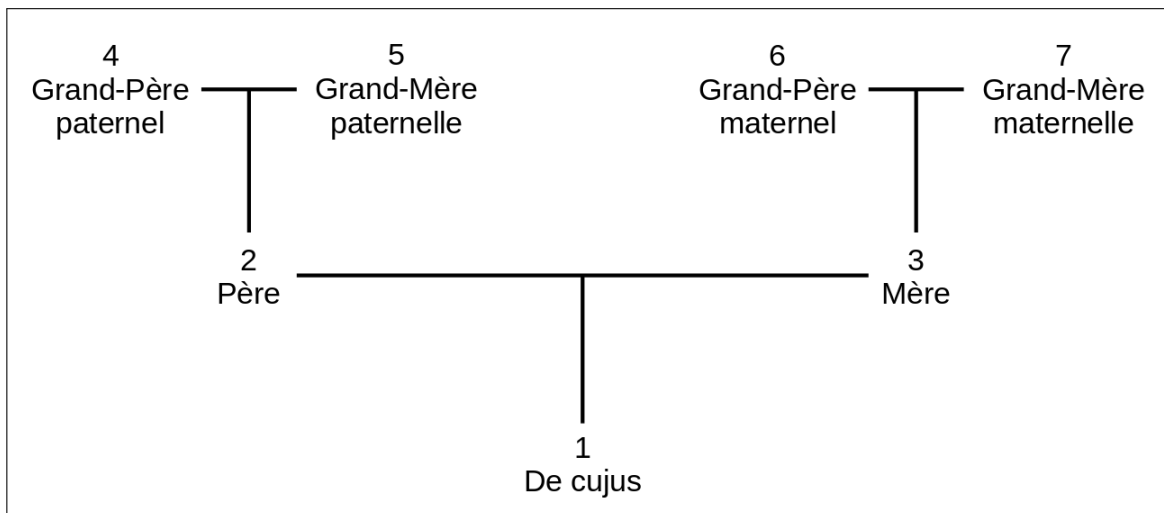


FIGURE 3 – Représentation de la numérotation Sosa-Stradonitz

Le *de cujus* dont on souhaite établir l'ascendance porte le *numéro 1*.

Les hommes portent un numéro pair tandis que les femmes portent un numéro impair.
Les relations de parentés dans l'arborescence sont définies lors de l'ajout des parents biologiques d'une personne en leur attribuant un numéro déterminé suivant les relations définies par la numérotation de Sosa-Stradonitz.

Pour une personne portant le numéro i , son père portera le numéro $2 \times i$ et sa mère le numéro $2 \times i + 1$.

Ainsi le numéro du père, d'une personne portant le numéro 4, sera $2 \times 4 = 8$ et celui de sa mère $2 \times 4 + 1 = 9$.

Le grand-père paternel portera lui le numéro $2 \times 8 = 16$ et la grand-mère paternelle le numéro $2 \times 8 + 1 = 17$.

Le grand-père maternel portera lui le numéro $2 \times 9 = 18$ et la grand-mère maternelle le numéro $2 \times 9 + 1 = 19$.

Avantages :

Cette règle de numérotation permet d'accéder directement aux parents en connaissant le numéro d'un individu.

Elle permet également d'accéder à l'époux / épouse d'une personne.

Elle permet aussi de connaître le sexe d'une personne directement avec la parité du numéro porté par celle-ci.

Chaque ancêtre ayant par avance un numéro fixe et réservé, il est possible de continuer l'arborescence même lorsqu'il manque une personne. En effet, chaque personne possède sa place définie par son numéro dans l'arbre généalogique en fonction de son lien de parenté avec le *de cujus*.

Il est également facile de classer les ancêtres en fonction des générations. Un individu ayant un numéro n appartient à la génération $n / 2$ dans l'arbre. (Le *de cujus* appartenant à la génération *zéro*, ses parents à la génération *un* et ses grands-parents à la génération *deux*)

Inconvénients :

Il est difficile de fusionner deux arbres généalogiques utilisant cette numérotation. En effet, il faut réadapter la numérotation des personnes d'un arbre afin de pouvoir les insérer correctement dans l'autre.

En cas de présence d'un *implexe*, des ancêtres peuvent avoir plusieurs numéros dans l'arbre. [4]
Cela nécessite alors une gestion particulière de ces individus qui ne sera pas prise en compte dans ce projet.

2 Conception et développement

Dans cette partie, nous allons expliquer le processus de conception et de développement de l'application en nous attardant sur les choix qui ont été faits sur la structure et les solutions aux divers problèmes et difficultés rencontrées.

2.1 Diagramme de GANTT

2.1.1 Diagramme de Gantt prévisionnel

Pour ce projet nous avons établi un planning prévisionnel (cf. figure 4) afin de nous organiser et de répartir, au sein du binôme, les différentes tâches à effectuer au cours du projet.

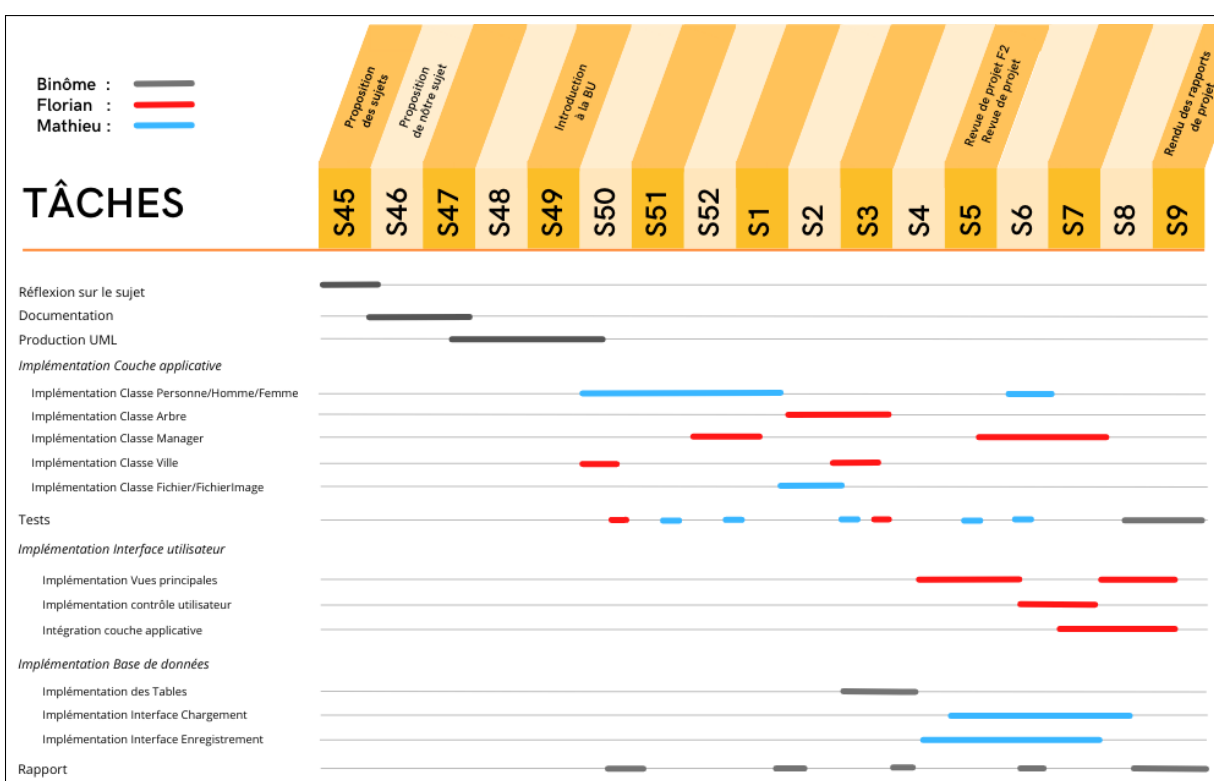


FIGURE 4 – Diagramme de Gantt prévisionnel

La chronologie des grandes étapes du projet a été dans l'ensemble respectée, même si des différences dans les délais et date de début des tâches peuvent être constatées.

2.1.2 Diagramme de Gantt réel

Nous avons pris du retard dès le début du projet en sous estimant le temps nécessaire à sa réalisation.

Certaines tâches nous ont pris plus de temps que prévues. Un changement d'approche sur la structure du code nous a obligé à retravailler certaines tâches déjà complétées. Le manque de temps, dû à notre retard, nous a poussé à nous focaliser uniquement sur les fonctionnalités essentielles afin de produire une version simple mais fonctionnelle du logiciel.

Nous avons cependant continué le développement en gardant une structure qui permet l'ajout des extensions facilement.

Ceci nous a poussé à adopter un rythme de travail plus soutenu sur les dernières semaines du projet.

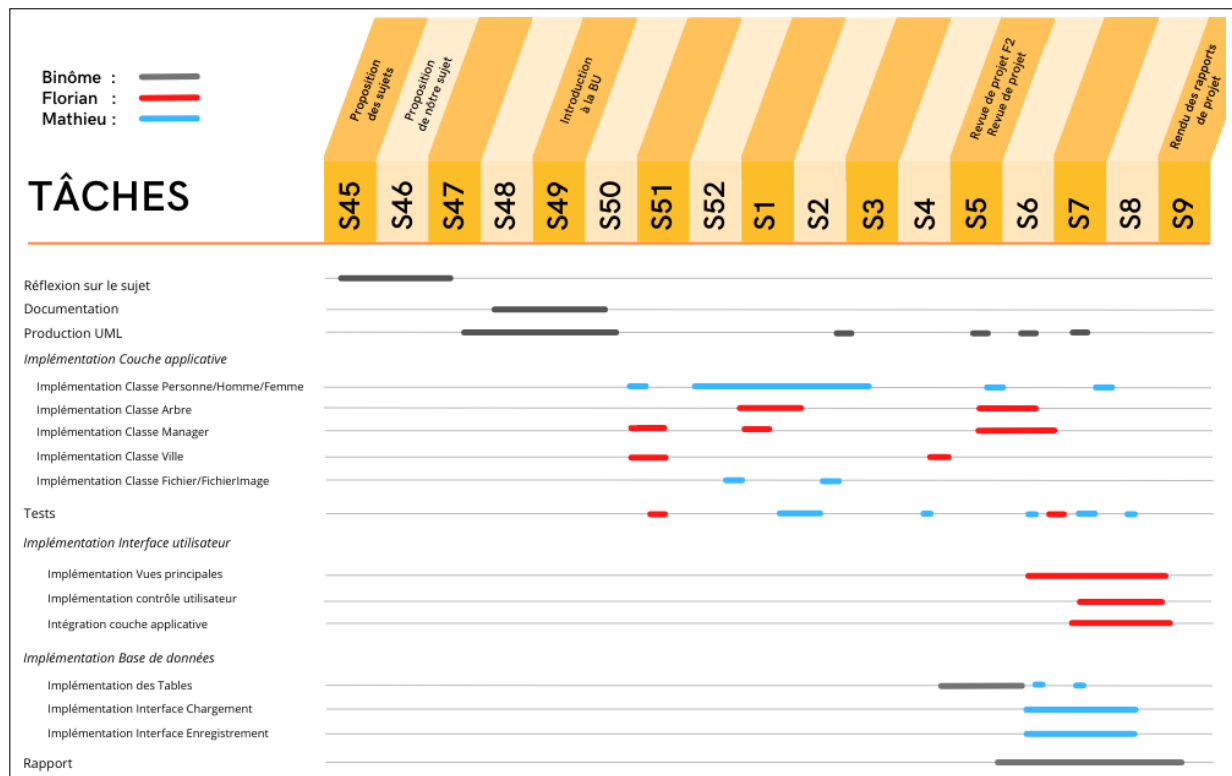


FIGURE 5 – Diagramme de Gantt réel

Notre répartition des tâches au sein du binôme a été correctement effectuée, ce qui nous a permis de travailler de manière indépendante sur les différentes parties du projet.

La communication au sein de l'équipe a été un élément essentiel. Chaque semaine, nous avons pris le temps de partager nos avancées, de s'entraider et d'apporter des solutions aux différentes difficultés rencontrées par chacun.

2.2 Conception

Dans cette partie, nous nous penchons sur les outils utilisés ainsi que la structure du logiciel et le modèle utilisé. Nous expliquerons le fonctionnement et les principales difficultés rencontrées ainsi que les solutions mises en place pour y remédier.

2.2.1 Outils utilisés

Nous nous sommes appuyés sur le logiciel *StarUML* pour la réalisation des différents diagrammes lors de la phase de conception.

Nous sommes partis sur le langage de programmation *.NET/C#* pour le développement et plus spécifiquement sur la version *.NET 6.0*. Ce langage permet un développement simple et rapide dans un environnement de développement. [5]

Nous avons utilisé *Visual Studio* dans sa version 2022 comme *IDE*, cette version étant la seule à prendre en charge la dernière version *.NET 6.0*.

La plateforme *WPF* nous a permis de construire l'interface utilisateur, en utilisant le *XAML* et *C#*, en lien avec la couche applicative. [6]

Pour la persistance de données, nous avons utilisé une base de données relationnelle *Microsoft SQL Server* fournie par l'*IDE Visual Studio*, dans sa version 2019. [7]

Nous avons utilisé le gestionnaire de version Git ainsi que le dépôt Gitlab de l'ISIMA comme service web d'hébergement pour le projet.

Pour la documentation du code nous nous sommes appuyés sur l'outil *Doxygen* permettant de produire simplement une documentation au format *HTML* ou *Latex*. A travers des commentaires balisés, l'outil extrait les prototypes des fonctions dans le code et les commentaires associés pour générer la documentation. [8]

2.2.2 Structure

Durant la phase de conception, nous avons établi différents diagrammes *UML*, dont celui de classes ci-dessous. Il permet de visualiser la structure de notre projet.

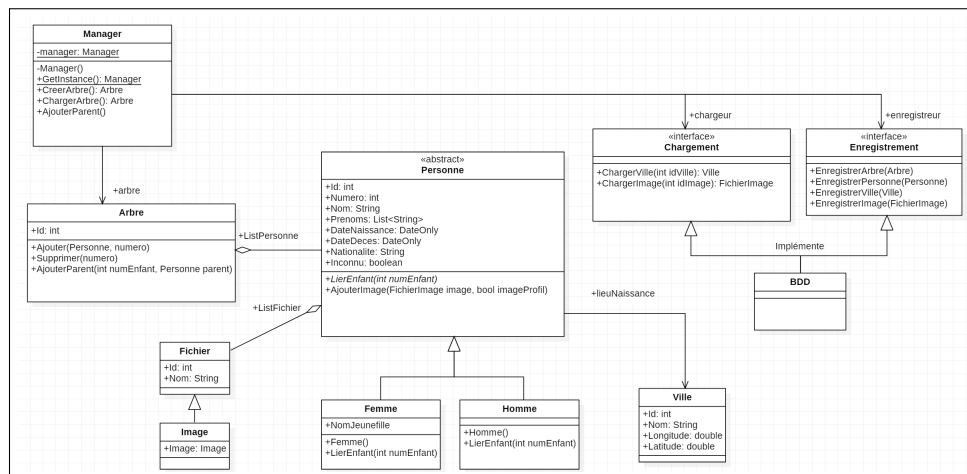


FIGURE 6 – Diagramme de classes

L'application Ancestrel est conçue selon une architecture **MVC**. Celle-ci permet de séparer la couche applicative et l'interface graphique.

Nous avons choisi de mettre en place une classe qui servira d'interface, de point d'entrée avec la couche applicative, le contrôleur. Cette classe **façade** est le Manager et doit être unique, c'est pourquoi nous utilisons le patron **singleton** pour nous assurer l'unicité de cet objet.

Afin de permettre différentes méthodes de persistance des données, nous avons choisi d'implémenter cette partie sous la forme d'une **stratégie**. Ce patron nous permet d'avoir différentes implémentations possibles pour la persistance. Il faut donc passer par les interfaces d'*Enregistrement* pour les méthodes de sauvegarde et de *Chargement* pour les méthodes de chargement. L'utilisation de ces méthodes ne doit pas dépendre de leur implémentation.

2.2.3 Modèle

Personne, Homme, Femme

Nous avons implémenté la classe abstraite *Personne* dont dérivent les classes *Homme* et *Femme* qui contiendront les informations des individus.

La spécialisation de la classe abstraite en sous-classe nous a permis d'ajouter des informations supplémentaires pour chaque personne en fonction du sexe.

- Rajouter le *nom de jeune fille* pour une femme.
- Attribuer aux personnes leur numéro correspondant dans l'arbre suivant le système de numérotation Sosa-Stradonitz. À savoir :
 - Numéro = $2 \times$ Numéro de l'enfant, pour un Homme (Relation de paternité)
 - Numéro = $2 \times$ Numéro de l'enfant + 1, pour une Femme (Relation de maternité)

Pour plus de cohérence et de lisibilité dans le code, nous avons déplacé l'initialisation de la propriété *Numéro* (contenant le numéro d'une personne dans l'arbre) dans une méthode *LierEnfant(int numEnfant)* virtuelle dans la classe *Personne*. Cette méthode est implémentée respectivement dans *Homme* et dans *Femme* suivant les relations de paternité et de maternité.

Arbre

La classe *arbre* décrit la structure de l'arbre ascendant. Elle contient la liste des personnes appartenant à l'arbre et gère l'ajout des ascendants.

Manager

Le *manager* permet de faire le lien entre les différentes couches de l'application. Cette classe est chargée de lier la couche applicative avec la couche de persistance et de remonter les informations à la *Vue*. De plus, les actions utilisateurs impliquant des modifications sur les données sont transmises par l'intermédiaire du *manager* au système de persistance de données.

2.3 Persistance des données

Nous souhaitons pouvoir enregistrer et lire les données d'un arbre généalogique sous différents formats, notamment pour pouvoir importer un arbre d'un autre type de données supporté par le logiciel. *(En particulier le format [GEDCOM](#) [9])*

C'est pourquoi, nous avons décidé de communiquer avec le système de persistance de données par l'intermédiaire de deux interfaces, une interface pour enregistrer les données et une pour les charger.

Cette abstraction nous permet de changer de système de persistance de données en maintenant la compatibilité avec le logiciel. Celui-ci utilisera les méthodes fournies par les interfaces pour accéder aux différentes données de l'arbre en faisant abstraction de la solution retenue pour l'implémentation du stockage.

2.3.1 Interfaces

Le système de stockage choisi devra implémenter deux interfaces pour pouvoir être utilisé par le logiciel.

Interface d'enregistrement des données

- Insérer un arbre généalogique
- Insérer une personne
- Insérer une image
- Insérer une ville
- Insérer un lien enfant-parent

Interface de chargement des données

- Charger un arbre généalogique
- Charger une personne
- Charger une image
- Charger une ville

2.3.2 Base de données Microsoft SQL Server

Dans un premier temps, nous avons décidé d'implémenter une base de donnée relationnelle pour le stockage de données. Nous nous sommes appuyés sur les outils fournis par l'IDE Visual Studio : Microsoft SQL Server 2019. [7]

Modèle de base de données

Nous avons créé une base de données suivant ce schéma :

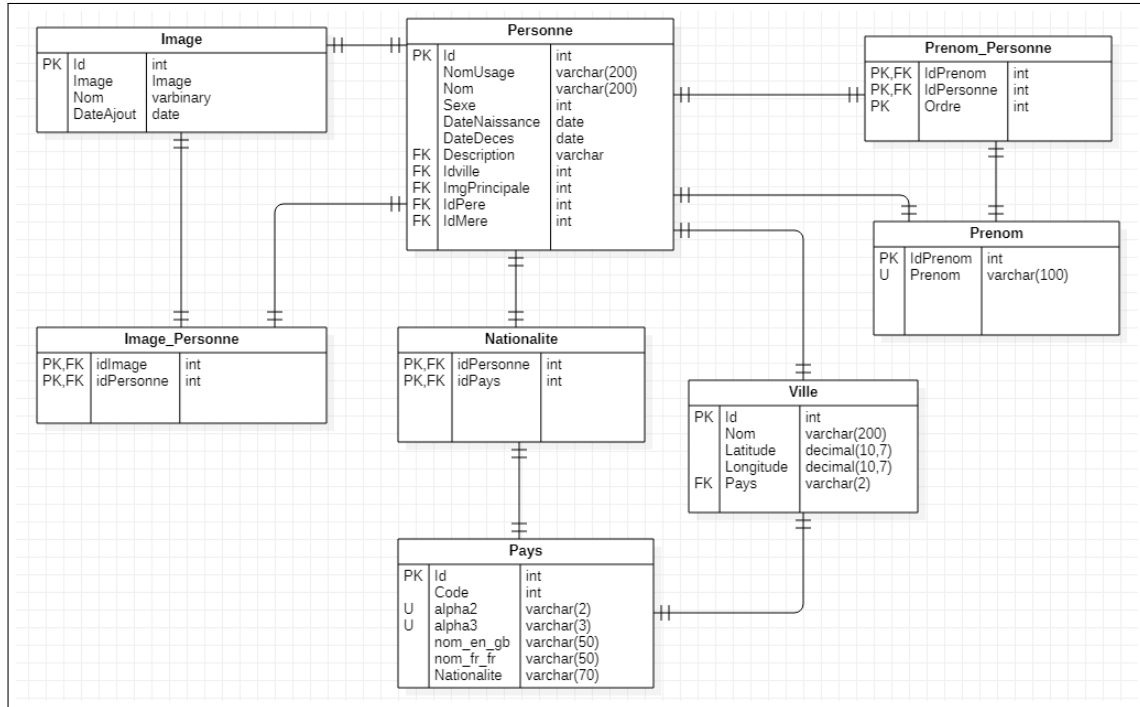


FIGURE 7 – Diagramme de base de données relationnelle

Nous utilisons une table *Personne* contenant les principales informations sur les individus.

Les tables *Ville* et *Pays* servent à stocker le lieu de naissance d'une personne. Les données concernant ces tables ont été récupérées en ligne [10] permettant d'avoir une base de villes et pays à proposer à l'utilisateur lors du choix d'un lieu de naissance pour un individu, ou du choix de la nationalité. La table *Ville* pourra être complétée par l'utilisateur, s'il souhaite ajouter un lieu non présent dans la base de données (notamment pour l'ajout de lieu ayant disparu ou changé de nom au cours du temps).

Nous avons décidé d'externaliser dans une table différente les prénoms d'une personne afin d'éviter la redondance du stockage des prénoms. En effet, un prénom peut être porté par plusieurs personnes, notamment pour des prénoms classiques.

Requêtes sur la base

Nous manipulons la base de données directement par des requêtes [Microsoft SQL Server](#).

La requête étant construite sous forme de chaîne de caractères, il a fallu convertir les paramètres, que l'utilisateur souhaite stocker dans la base, en chaîne de caractères afin d'être correctement interprétés par le système de base de données.

Les propriétés des personnes, telles que le nom ou la date de naissance, pouvant être du type *NULL* dans l'application, il a fallu lors de l'insertion remplacer ce champ par la chaîne de caractères "*NULL*" pour l'insertion de la valeur *NULL* dans la base de données. Le problème étant la conversion implicite de la valeur *NULL* qui retourne une chaîne vide et non une chaîne de caractères "*NULL*" comme souhaitée.

Lors de la récupération des données, il a fallu convertir le type *System.DBNull* de la base de données en *NULL* de [C#](#).

Les dates respectant un certain format, il a fallu expliciter ce format afin que la base de données puisse les lire et les interpréter correctement.

2.4 Partie graphique / Interface utilisateur

L'interface utilisateur se décompose en différentes [vues](#) dont les principales sont :

- La page d'accueil (cf. [figure 9](#))
- La page de chargement d'un arbre (cf. [figure 11](#))
- La page de création d'une personne ou d'un arbre (cf. [figure 10](#))
- La page de visualisation de l'arbre (cf. [figure 12](#))
- La page de détail / modification d'une personne (cf. [figure 13](#))

Ces différentes [vues](#) sont codées en [XAML](#) permettant ainsi de structurer les pages et leurs différents composants.

Le [code-behind](#), en [C#](#), permet de les compléter avec les données provenant du modèle et d'implémenter les événements liés aux actions de l'utilisateur. Le *manager* permet la manipulation du modèle par l'intermédiaire des éléments de la [Vue](#).

Les principales interactions de l'utilisateur avec l'interface graphique se font à travers des boutons pour naviguer et valider ses choix et par l'intermédiaire de *TextBox* pour rentrer des informations.

Ainsi sur les différentes [vues](#), l'utilisateur sera invité à fournir des informations et à les valider. Lorsque l'utilisateur valide les données rentrées, celles-ci doivent être ajoutées, ou bien modifiées, dans le modèle puis dans la base de données. Cette dernière opération étant réalisée par l'intermédiaire du *manager* à travers l'objet implémentant les interfaces d'enregistrement et de chargement.

Pour construire certaines pages, des *contrôles utilisateur* sont utilisés afin de créer des éléments réutilisables. Par exemple les cartes représentant les personnes (cf. figure 8). Ces cartes sont utilisées pour l’affichage de l’arbre et représentent une personne dans cet arbre. Ainsi, une instance d’une de ces cartes est liée à une unique personne.

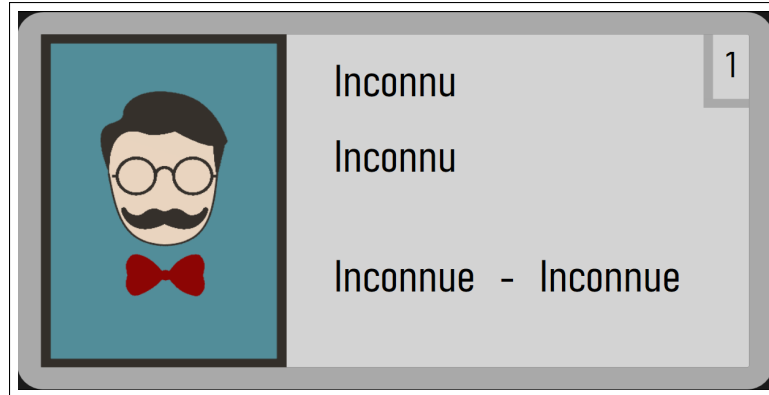


FIGURE 8 – Contrôle d'utilisateur : carte d'une personne

Chacune des pages a une responsabilité particulière.

La *page d'accueil* permet d'aiguiller l'utilisateur dans son utilisation de l'application (charger un arbre déjà existant ou en créer un nouveau).

La *page de chargement* permet à l'utilisateur de choisir le [de cujus](#). Sur cette page est présentée l'ensemble des personnes enregistrées dans la base. L'utilisateur doit donc choisir une de ces personnes et l'application se charge de recréer l'arbre à partir de cette personne grâce aux informations issues de la base.

La *page de création d'une personne* permet l'ajout d'un individu et l'intègre dans l'arbre. Par l'intermédiaire d'un formulaire, l'utilisateur renseigne les diverses informations nécessaire à la création de la personne.

La *page de visualisation*, quant à elle, est chargée de représenter visuellement la structure de l'arbre. Chaque personne est représentée par une carte qui résume ses informations. Ces cartes sont placées et liées de telle sorte que les liens de parentés entre les individus soient clairement identifiables. C'est également sur cette page que l'utilisateur est invité à ajouter les ascendants mais aussi à choisir les personnes qu'il souhaite visualiser en détail ou modifier.

La *page de détail / modification* joue un double rôle. Elle permet à l'utilisateur de visualiser une personne en détail, mais également de modifier ses informations ou d'en ajouter. Elle se présente sous une forme assez similaire à la *page de création d'une personne* mais en ayant un rôle différent. En effet, elle est attachée à une personne déjà existante et intégrée dans l'arbre. Elle ne doit donc pas gérer les liens entre cette personne et le reste de l'arbre contrairement à la page de création.

3 Résultats

3.1 Architecture logicielle

Le logiciel, implémenté en [.NET/C#](#), s'articule autour d'un Manager composé d'un arbre contenant différentes personnes liées entre elles par un système de numérotation ascendant. Celui-ci gère l'ajout d'un individu dans l'arbre avec la création d'une personne à l'aide des différentes informations fournies. Il crée la relation de parenté entre la personne nouvellement ajoutée et son enfant à travers le calcul du numéro cette personne, indiquant sa position dans l'arbre, en fonction de celui de l'enfant.

À chaque validation d'ajout d'une Ville / Image / Personne, celle-ci est ajoutée au système de persistance choisie, dans notre cas sur une base de données [Microsoft SQL Server](#).

Les multiples [vues](#), construites en [XAML](#), permettent les différentes interactions avec l'utilisateur, que ce soit pour l'affichage des données (de l'arbre ou des informations sur les personnes) ou bien pour la demande de saisie des informations sur les personnes.

3.2 Fonctionnement

L'utilisateur doit être en mesure de **consulter un arbre**. Pour cela, un arbre doit être créé ou chargé, et ce depuis la page d'accueil. (cf. figure 9) Sur cette page, l'utilisateur est aiguillé vers ces deux choix. Un bouton permettant de se diriger vers la création d'un arbre et l'autre vers le chargement d'un arbre depuis une personne déjà existante.

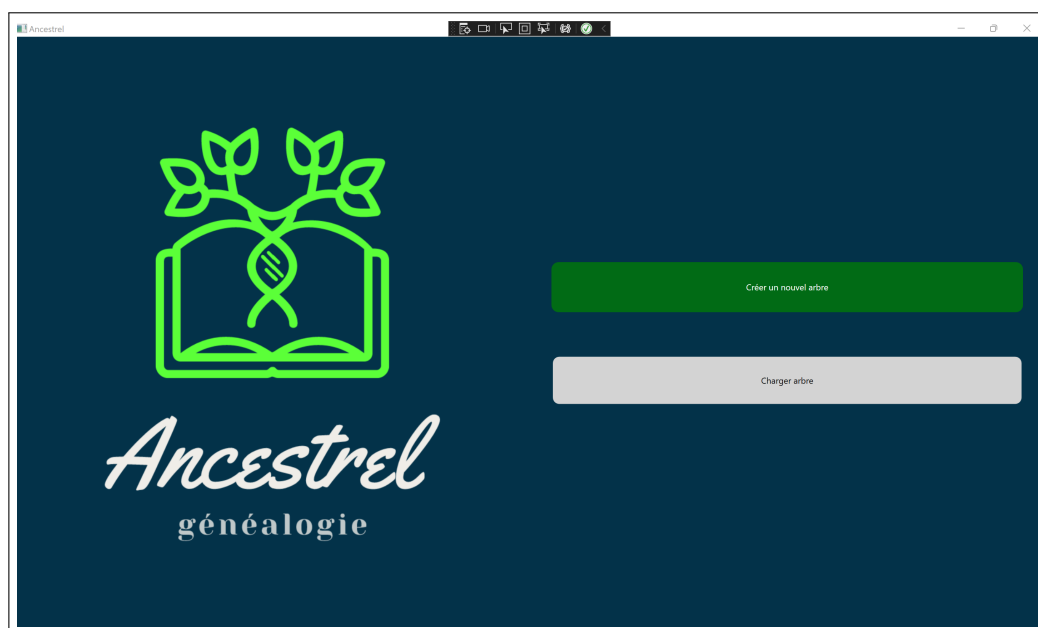



FIGURE 9 – Page d'accueil


Pour la création d'un arbre, il faut créer une personne sur la page de création de personne, (cf. figure 10) l'utilisateur est invité à rentrer les informations de base de la personne qui sera le *de cujus* du nouvel arbre. L'arbre est alors vide, il ne contient que la personne qui vient d'être créée.



The screenshot shows a web browser window titled 'Ancestrel'. The main content area has a dark blue background. At the top, there are three input fields: 'Nom d'Usage : Dumas', 'Prenoms : Florian', and 'Sexe : Masculin'. Below these, there is a button labeled 'Confirmer'.

FIGURE 10 – Page de création d'une personne ou d'un arbre

Le chargement permet de choisir une personne déjà existante dans la base de données. Ainsi, l'application charge depuis la base les différentes personnes ayant au moins un nom et un prénom connus. Ces personnes sont affichées sur la page de chargement. (cf. figure 11) L'utilisateur choisit alors la personne qu'il souhaite à la base de l'arbre, le *de cujus*, cette personne ainsi que tous ses ascendants sont alors chargés depuis la base de données.



The screenshot shows a web browser window titled 'Ancestrel'. The main content area has a dark blue background. At the top, there is a text prompt: 'Veuillez choisir la personne enregistrée comme de cujus'. Below this, there are two horizontal bars representing a list of names: 'Paul Dupont' and 'Pierre Rigaud'.

FIGURE 11 – Page de chargement d'un arbre

Après la création ou le chargement d'un arbre, l'utilisateur arrive sur la page de visualisation (cf. figure 12) de l'arbre dans sa globalité. Il peut ainsi voir chaque personne présente dans l'arbre de manière structurée. Il peut **modifier l'arbre** en choisissant d'ajouter une personne, ou bien **consulter une personne** déjà présente en cliquant dessus.

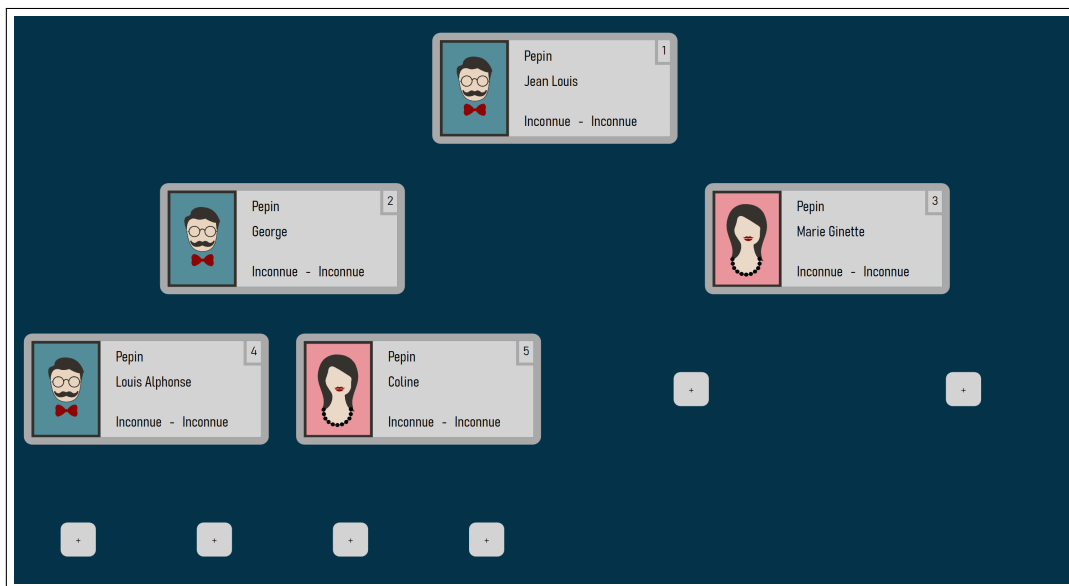


FIGURE 12 – Page de visualisation de l'arbre

Pour **ajouter un ascendant**, l'utilisateur doit fournir les informations de cette personne. Dans le cas où aucune information ne serait saisie, celle-ci serait considérée comme inconnue.

Pour **modifier une personne**, l'utilisateur doit accéder à la page de détails (cf. figure 13) en cliquant sur la carte de la personne à modifier. Il peut simplement avoir accès aux détails de la personne ou alors modifier et ajouter des informations ainsi que des **images**.

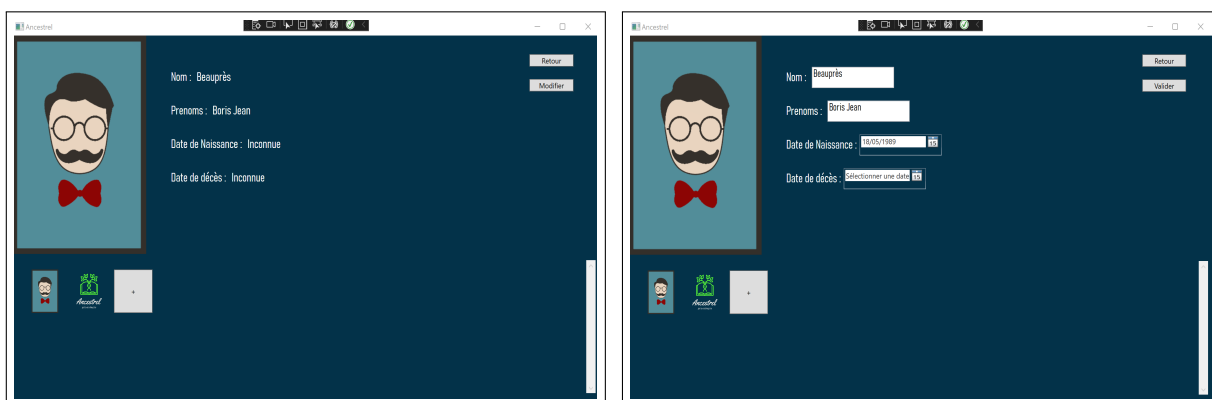


FIGURE 13 – Page de détails / Modification d'une personne

3.3 Etat du produit

Nous avons pu implémenter les fonctionnalités essentielles hormis la suppression d'un individu et la gestion des nationalités ou l'ajout d'une ville de naissance pour une personne. L'interface utilisateur n'est pas terminée, une première version, très simpliste est implémentée afin de pouvoir interagir avec l'application. On peut voir par exemple que les cartes de personnes ne sont pas liées entre elles. (cf. figure 12)

Les extensions n'ont pas pu être implémentées par manque de temps, mais leur intégration devrait être possible sans modification majeure dans le code déjà présent.

La première extension que nous voudrions implémenter serait l'importation d'arbre depuis des données sous format **GEDCOM** qui est la norme dans les logiciels de **généalogie** déjà existants. Ceci permettrait à l'utilisateur de remplir rapidement la base de données à partir d'un fichier déjà existant contenant des informations sur des personnes pour la réalisation d'un arbre.

Une documentation est disponible afin de faciliter la reprise du code de l'application pour sa maintenance ou son amélioration par l'ajout de nouvelles fonctionnalités.

Le dépôt du projet contenant le code et la documentation se trouve à l'adresse suivante :
<https://gitlab.isima.fr/fldumas5/genealogie.git>

Conclusion

Dans le cadre de recherche généalogique, il peut être intéressant de conserver des données et de les mettre en forme numériquement à l'aide d'un arbre généalogique.

Ainsi notre objectif était de produire un logiciel permettant de créer simplement un arbre généalogique ascendant à partir d'un individu, le [de cujus](#). Le logiciel devait permettre d'ajouter simplement des personnes avec diverses informations et de les lier entre elles dans l'arbre. Il devait aussi permettre de sauvegarder et de charger un arbre.

Le retard prit sur notre planning, nous a contraint à réduire le nombre de fonctionnalités et à écarter les extensions prévues du planning.

Cependant, nous avons réussi à atteindre l'objectif principal, c'est-à-dire la possibilité de créer un arbre généalogique, même si nous voulions arriver à une application plus complète au terme de ce projet.

Bien que l'application soit fonctionnelle, amélioration au niveau de l'interface graphique reste à faire, pour rendre l'utilisation plus agréable. La prise en compte de toutes les informations sur les individus dans l'arbre reste à implémenter. En effet, certains renseignements sont déjà existants dans la couche applicative mais les [Vues](#) ne permettent pas encore de les manipuler (ville de naissance, nationalité, description). La prise en charge de PDF comme document associé à une personne pourrait permettre une plus grande flexibilité pour l'utilisateur.

Il serait également intéressant d'implémenter une prise en charge du format [GEDCOM](#) afin de se conformer aux standards de généalogie et pouvoir ainsi utiliser des données provenant d'autres applications.

Références

- [1] GENEAWIKI. *Guide guide genealogie*. URL : <https://fr.geneawiki.com/index.php?title=M%C3%A9thodologie> (visité le 06/12/2021).
- [2] Christian C. EMIG. *Quelques réflexions sur la Généalogie et sur son usage*. 2014. URL : http://paleopolis.rediris.es/NeCs/NeCs_03-2014/index.html (visité le 13/12/2021).
- [3] GENEAWIKI. *Numérotation*. 12 nov. 2016. URL : https://fr.geneawiki.com/index.php?title=Num%C3%A9rotation#Num.C3.A9rotation_d.27ascendants (visité le 06/12/2021).
- [4] GENEAWIKI. *Implexe*. 4 juin 2012. URL : <https://fr.geneawiki.com/index.php?title=Implexe> (visité le 31/01/2022).
- [5] MICROSOFT. *Documentation C#*. URL : <https://docs.microsoft.com/fr-fr/dotnet/csharp/> (visité le 13/12/2021).
- [6] MICROSOFT. *Vue d'ensemble du langage XAML*. URL : <https://docs.microsoft.com/fr-fr/visualstudio/xaml-tools/xaml-overview?view=vs-2022> (visité le 31/01/2022).
- [7] MICROSOFT. *Créer une base de données et ajouter des tables dans Visual Studio*. URL : <https://docs.microsoft.com/fr-fr/visualstudio/data-tools/create-a-sql-database-by-using-a-designer?view=vs-2022> (visité le 31/01/2022).
- [8] DOXYGEN. *Special Commands*. URL : <https://www.doxygen.nl/manual/commands.html> (visité le 20/12/2021).
- [9] FAMILYSEARCH. *GEDCOM Specifications*. URL : <https://gedcom.io/specs/> (visité le 06/12/2021).
- [10] GEONAME. *Data Sources*. URL : <http://download.geonames.org/> (visité le 31/01/2022).

Glossaire

.NET Plateforme de développement, gratuite, multiplateforme et open source permettant de créer de nombreux types d'applications différents. [iii](#), [10](#), [16](#)

C# Language de programmation orienté objet qui permet aux développeur de créer une variété d'applications sécurisées et robustes qui s'exécutent sur .NET. [iii](#), [10](#), [14](#), [16](#)

code-behind Terme utilisé pour décrire le code joint aux objets définis par le balisage, quand une page XAML est compilée. Il permet de décrire les méthodes liées aux différents éléments graphiques définis dans le fichier XAML. [14](#)

de cujus Personne à l'origine d'un arbre généalogique, qu'il soit ascendant ou descendant. [3](#), [7](#), [15](#), [17](#), [20](#)

eXtensible Application Markup Language (XAML) Language balisé permettant de décrire un environnement graphique. [iii](#), [10](#), [14](#), [16](#)

Genealogical Data COMmunication (GEDCOM) Norme élaborée par les Mormons, qui permet les échanges de données informatisées entre les différents logiciels de généalogie, quels que soient les systèmes d'exploitation ou le matériel. [5](#), [12](#), [19](#), [20](#)

généalogie Étude de l'origine et de l'histoire des familles, pour laquelle les généalogistes compilent des listes d'ancêtres. [1](#), [2](#), [6](#), [19](#)

HyperText Markup Language (HTML) Langage balisé conçu pour représenter les pages web. [10](#)

implexe Ancêtre apparaissant à plusieurs endroits dans un arbre généalogique. [7](#)

Integrated Development Environment (IDE) Un environnement de développement est un ensemble d'outils qui permet d'augmenter la productivité des programmeurs qui développent des logiciels. [10](#), [13](#)

latex Langage et système de composition de documents. [10](#)

Microsoft SQL Server Système de gestion de base de données en langage SQL développé par Microsoft. [iii](#), [10](#), [13](#), [14](#), [16](#)

Modèle-Vue-Contrôleur (MVC) Patron d'architecture permettant de séparer l'implémentation de la logique applicative (le contrôleur), les données à manipuler (le modèle) et l'interface graphique (la vue). [11](#)

Unified Modeling Language (UML) Langage de modélisation graphique conçu comme une méthode normalisée de visualisation dans les domaines du développement logiciel et en conception orientée objet. [10](#)

Visual Studio Environnement de développement complet pour les développeurs .NET et C++ sur Windows. [iii](#), [10](#), [13](#)

vue Terme utilisé pour décrire l'environnement graphique et le code qui lui est associé. [11](#), [14](#), [16](#), [20](#)

Windows Presentation Foundation (WPF) Infrastructure d'interface utilisateur qui permet de créer des applications clientes de bureau. [iii](#), [10](#)