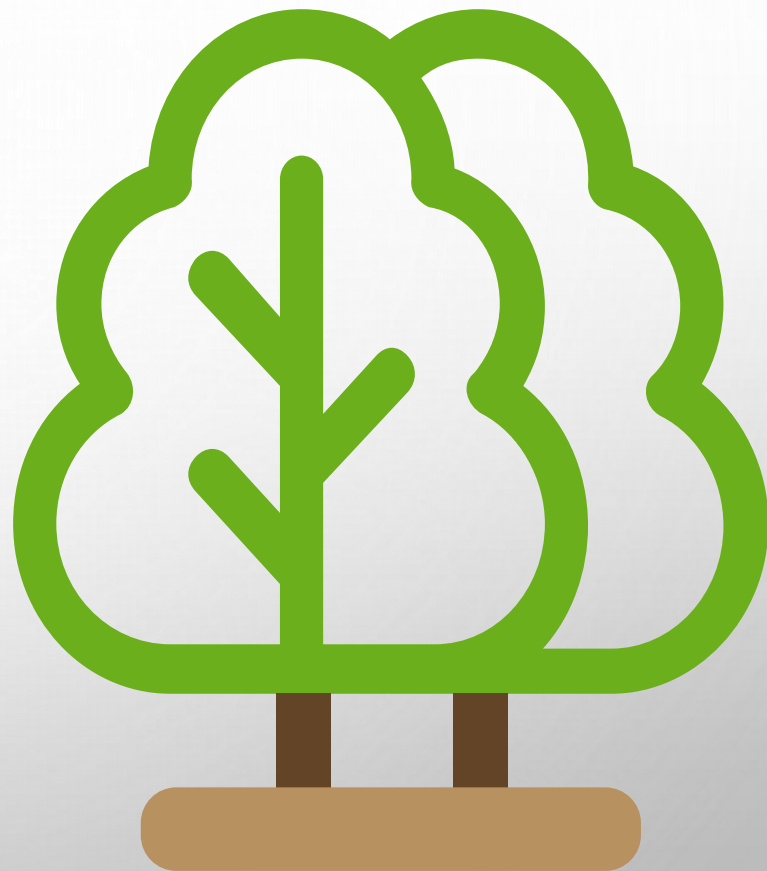


# FAMILY TREE

## GENEPROPP

- MODULE : JEE  
PROFESSEURE : ZAUCHE DJAOUIDA
- CLASSE : ING2 GSIA  
ANNÉE UNIVERSITAIRE : 2023-2024
- DATE DU PROJET : 11/2023 - 12/2023



# Introduction



Ce rapport présente le projet avait pour but de concevoir et implémenter un site web rendant possible la construction, la manipulation et la visualisation d'arbres généalogiques. L'application web devra présenter de nombreuses caractéristiques avancées permettant aux utilisateurs de gérer leurs arbres de manière simplifiée, ergonomique, avec une interface conviviale.

Pour mener à bien ce projet, nous avons constitué deux équipes pour permettre une bonne organisation pendant toute la partie du développement. Une équipe Back-End (Joan et Jordan) utilisant les technologies **Java & SpringBoot**, et une équipe Front-End (Adam et Clément) pour la partie **Angular**. La jointure des deux "projets" s'est réalisé par le biais de l'outil Git hub ([Lien vers notre Github](#)), organisé en plusieurs branches pour avoir une bonne architecture de travail.

La réalisation de ce travail s'inscrit dans le cadre du module JEE réalisé au cours de notre deuxième année du cycle ingénieur, spécialité génie des systèmes d'informations. Vous pouvez trouver un lien vers le sujet en cliquant juste ici.

Ce projet vient avec d'autres documents : [REST API DOCUMENTATION.xlsx](#), [Other Documents](#), [ALL DOCUMENTS FOLDER](#)

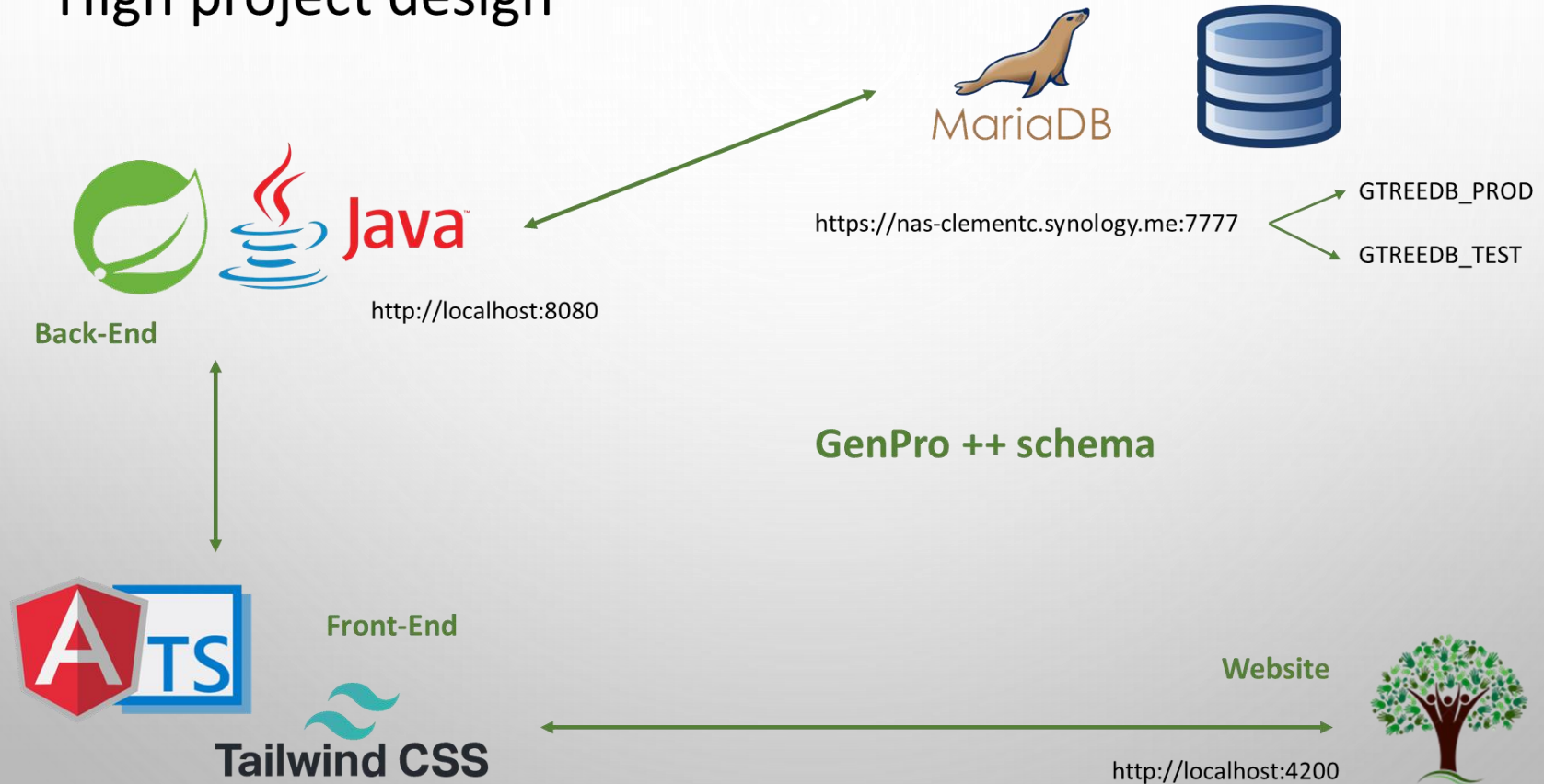
# Sommaire



<b>Cahier des charges du Projet .....</b>	<b>4</b>
<b>Conception.....</b>	<b>5</b>
<b>Architecture Général .....</b>	<b>5</b>
<b>High-Level Design du Projet .....</b>	<b>5</b>
<b>Architecture Back .....</b>	<b>6</b>
<b>Diagramme Database Initial .....</b>	<b>6</b>
<b>Diagramme Database Final .....</b>	<b>7</b>
<b>Diagrammes UML du Back .....</b>	<b>8</b>
<b>Architecture des dossiers .....</b>	<b>10</b>
<b>Architecture Front .....</b>	<b>11</b>
<b>Pages Front-End + Features .....</b>	<b>11</b>
<b>Features Back-End .....</b>	<b>22</b>
<b>Highlights du Back-End .....</b>	<b>24</b>
<b>Highlights du Front-End .....</b>	<b>27</b>
<b>Perspectives .....</b>	<b>28</b>
<b>Faiblesses .....</b>	<b>29</b>
<b>Conclusion .....</b>	<b>30</b>

# Conception - Général

## High project design



# Cahier des charges du projet (tiré du sujet)



Base du site :

- Permettre la construction et la manipulation conviviale d'arbres généalogiques.
- Représentation arborescente des membres d'une famille.
- Membres les plus jeunes en bas de l'arbre.
- Catalogue des créateurs d'arbres généalogiques.
- Liste des utilisateurs inscrits dans la base de données.
- accès ouvert à l'inscription, demande d'adhésion en ligne.
- Fournir informations personnelles, numéro de sécurité sociale, carte d'identité, photo numérique.

- Étude des demandes d'adhésion par les administrateurs.
- Attribution d'une clef de connexion privé après approbation.
- Certaines informations immuables après validation.

Spécificités intra-arbre généalogique:

- Création d'un arbre après inscription.
- Ajout de personnes avec lien de parenté.
- Règles de cohérence: lien de parenté, date de naissance...
- Différenciation nœud personne inscrite, nœud d'une personne non inscrite.
- Consultation de l'arbre avec affichage textuel et graphique.
- Requêtes pour ascendants, tantes, oncles, etc.
- Suppression de nœuds avec approbation du serveur et approbation du créateur du nœud.

(Uniquement le créateur du nœud peut pour le moment)

- Modification de détails de nœuds (certaines infos immuables) avec approbation du serveur et approbation du créateur du nœud. (Uniquement le créateur du nœud peut pour le moment)

- \*Création d'un lien avec une nœud appartenant à un autre arbre, avec validation du créateur du nœud

Manipulation inter-arbres généalogiques:

- Niveaux de visibilité des nœuds de l'arbre: Public, Private, Protected.
- Association de niveaux de visibilité aux nœuds.
- Visualisation et interrogation d'arbres d'autres utilisateurs.

Possibilité de voir les arbres publics,

Refuser l'accès aux arbres privés si on n'en est pas le propriétaire.

Si l'arbre est public (ou s'il s'agit du votre) possibilité de :

- Recherche de membres communs avec son arbre.

\*- Filtrer les nœuds de l'arbres en fonction de n'importe qu'elle champs

- Échange de ressources limité aux membres de la famille.
- Partage de photos, vidéos, nouvelles avec la famille.
- Échange limité aux membres de la famille.

Statistiques des consultations des arbres généalogiques:

- Suivi des consultations par d'autres utilisateurs.
- Calcul des fréquences par mois ou (\*et) par année.
- Affichage des personnes qui ont consulté l'arbre.

Autres

- Documentation technique
- Tests unitaires automatiques.
- Projet entier sauvegardé sur GitHub.

Légendes : États de chaque point

Complètement Implémenté

Partiellement implémenté et dont la finalisation serait sans difficulté

Prévu / Partiellement Implémenté / Implémentable facilement en l'état actuelle

Pas faisable sans grosses modifications

Choix de non implémentation

\*Point interprété / amélioré

[Lien vers l'image](#)



## Architecture de la Database Final





## Architecture Back-End



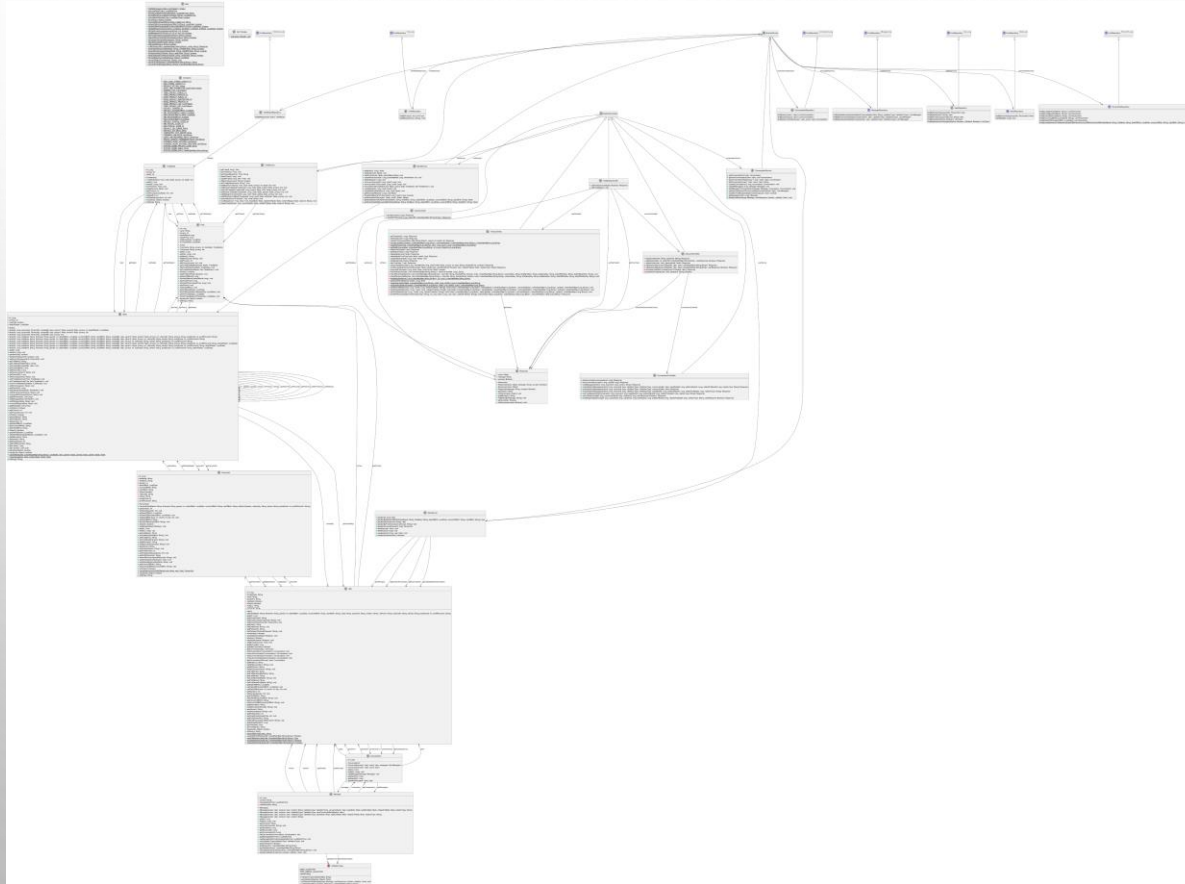


# Conception - Back

## Architecture Back-End



[Lien vers l'image](#)  
[Lien vers la source](#)



# Architecture Back-End – architectures des dossiers



- Src/main
  - Java
    - Com.acfjj.app
      - controller (Contains controllers)
      - model (Contains models)
      - repository (Contains repositories)
      - service (Contains services)
      - utils (Contains utility files)
      - GenTreeApp.java (Main class to launch the back-end)
  - ressources
    - Application.properties
    - Test.properties
- Src/test
  - Java
    - model
    - service

# Architecture Front-End



```
▼ GENEALOGIC-TREE-FRONT
  > node_modules
  ▼ src
    ▼ app
      > components
      > pages
      > services
      TS app-routing.module.ts
      # app.component.css
      <> app.component.html
      TS app.component.spec.ts
      TS app.component.ts
      TS app.module.ts
    > assets\media
    <> custom-theme.scss
    ★ favicon.ico
    <> index.html
    TS main.ts
    # styles.css
```

```
▼ components
  > contact-folder
  > directory-folder
  > footer
  > home-content
  > identification-folder
  > nav-folder
  > PopUps
  > profil-folder
  ▼ tree-folder
  > linked-hashmap
  ▼ tree-content
    # tree-content.component.css
    <> tree-content.component.html
    TS tree-content.component.spec.ts
    TS tree-content.component.ts
```

```
▼ services
  > conversation
  > cookies
  > identificaton
  > tree
  ▼ user
    TS user.service.spec.ts
    TS user.service.ts
```

```
▼ pages
  > contact-page
  > directory-page
  > home-page
  > login-page
  > profil-page
  ▼ registration-page
    # registration-page.component.css
    <> registration-page.component.html
    TS registration-page.component.spec.ts
    TS registration-page.component.ts
```

# Architecture Front-End



L'architecture du projet sur la partie front-end a été conceptualisée et réalisée permettant un accès rapide et ordonné aux fichiers, la réutilisation de code, ainsi qu'une évolutivité simplifiée.

Les principaux dossiers composant cette architecture sont :

- Components : Stockages de tous les composants Angular nécessaires sur la partie visuelle et fonctionnelle des pages du site web. Chaque sous dossier est composé de tous les composants nécessaire pour une page spécifique du site (voir slide précédent pour illustration).
- Pages : Stockage des composants Angular représentant les pages complètes
- Services : Stockages des composants Angular permettant l'accès et l'envoi de données à la partie back-end
- Assets : Stockage des images propre au site web (logo sous différents format)

# Pages Front-End + Features



Registration page : Lien vers la page d'inscription [ici](#)

Login page : Lien vers la page de connexion [ici](#)

Home page : Lien vers la page Home [ici](#)

My Tree page : Lien vers la page de l'arbre [ici](#)

Contact page : Lien vers la page contact [ici](#)

Directory page : Lien vers la page d'annuaire [ici](#)

Profil page : Lien vers la page profil [ici](#)

# Registration Page



First Name	Last Name
<input type="text"/>	<input type="text"/>
Email	Social Security Number
<input type="text"/>	<input type="text"/>
Sexe	Phone Number
<input type="radio"/> Man <input type="radio"/> Woman <input type="radio"/> Other	<input type="text"/>
Date of Birth	City of Birth
<input type="text" value="mm/dd/yyyy"/>	<input type="text"/>
Country of Birth	Nationality
<input type="text"/>	<input type="text"/>
Address	Postal Code
<input type="text"/>	<input type="text"/>
Password	Confirm password
<input type="text"/>	<input type="text"/>

[Make Your Membership request](#)

Already a member? [Sign in to Genepp++](#)

Remplissage des champs necessaires pour l'inscription sur le site GenPro ++, tel que : Prénom, nom, adresse email, numéro de sécurité social, sexe, numéro de téléphone, date de naissance, ville de naissance, pays de naissance, nationalité, adresse postal, code postal, mot de passe

Bouton permettant l'envoi du formulaire d'inscription pour validation d'un administrateur du site web

Possibilité de retourner sur la page de connexion si l'utilisateur possède déjà un compte

# Login Page



## Sign in to GenePro++

Private code

Password

[Forgot password?](#)

Sign in

Not a member? [Make your membership request](#)

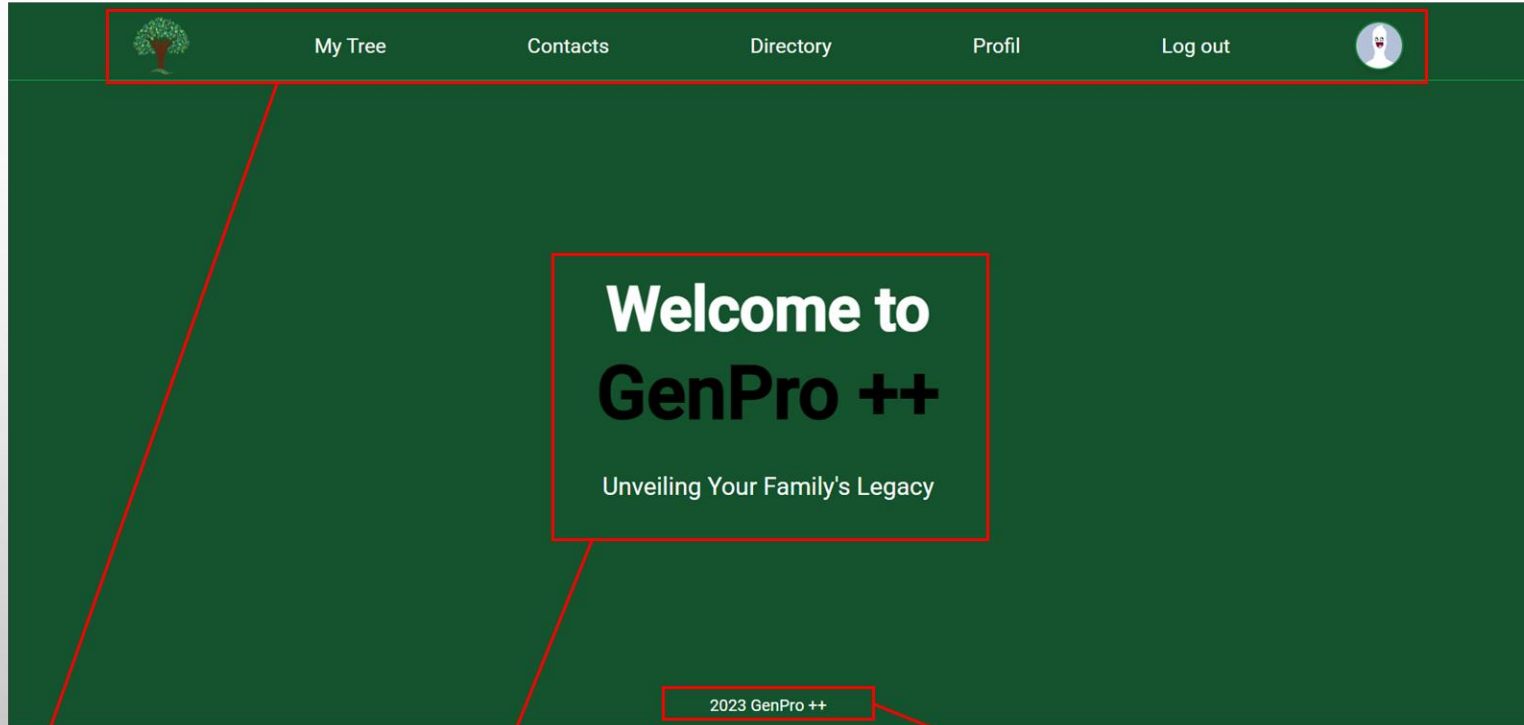
Remplissage des champs nécessaires pour la connexion sur le site GenPro ++. Cela correspond au code privé de l'utilisateur (reçu à l'inscription), ainsi que son mot de passe

Bouton permettant la réinitialisation du mot de passe de l'utilisateur si celui-ci l'a égaré

Bouton permettant l'envoi du formulaire de connexion

Possibilité d'aller sur la page d'inscription si l'utilisateur ne possède pas encore de compte sur le site

# Home Page



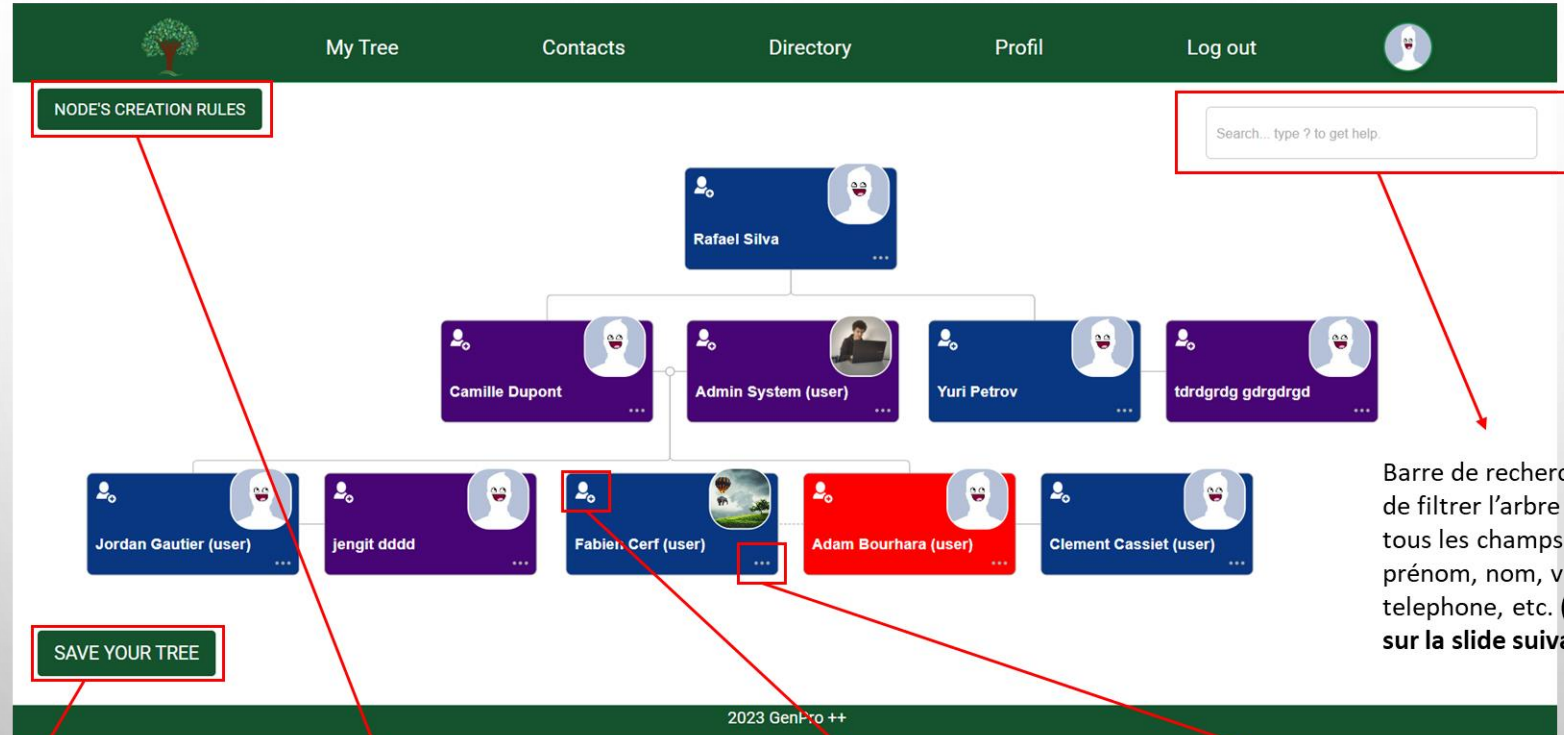
Barre de navigation permettant de parcourir les différentes pages du site

Contenu de la page home avec phrase d'accueil du site

Bas de page composé de l'année et du nom du site



# Tree Page



Barre de recherche permettant de filtrer l'arbre en fonction de tous les champs possibles, prénom, nom, ville, pays, telephone, etc. (voir image 3 sur la slide suivante)

Bouton permettant de sauvegarder l'arbre modifié par l'utilisateur en base de donnée

Bouton permettant d'afficher une popup présentant les règles quant à la création de personnes dans son arbre (voir image 1 sur la slide suivante)

Bouton permettant l'ajout d'une personne à son arbre en relation avec cette personne (voir image 2 sur la slide suivante)

Bouton permettant d'afficher et de modifier les informations d'une personne de son arbre (voir image 3 sur la slide suivante)

**Image 1 :** règles pour la création de personne dans l'arbre

Before creating your family tree, please provide the following required information:

- First Name \*
- Last Name \*
- Date of Birth \*
- City of Birth \*
- Country of Birth \*
- Gender: Choose either "Male," "Female," or "Other."
- Privacy: Choose either "Public," "Private," or "Restricted."

The tree owner's node will be highlighted in red, male individuals in blue, and female individuals in purple.

Nodes with '(user)' after their first and last names represent users associated with the site.

Fields marked with \* are mandatory. Additionally, please select an option for both the Gender and Privacy settings.

Close Window

Admin System (user) →



FirstName  
Admin

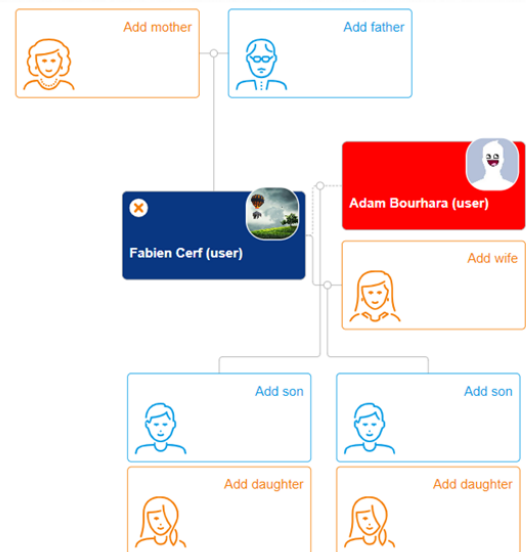
LastName  
System

gender  
other

privacy  
public

Photo Url  
<https://media.lidn.com/dms/image/C4E03AQEV>

**Image 2 :** ajout d'une relation dans son arbre



**Image 3 :** affichage des informations d'une personne dans l'arbre, possibilité de modification et de suppression de la personne.

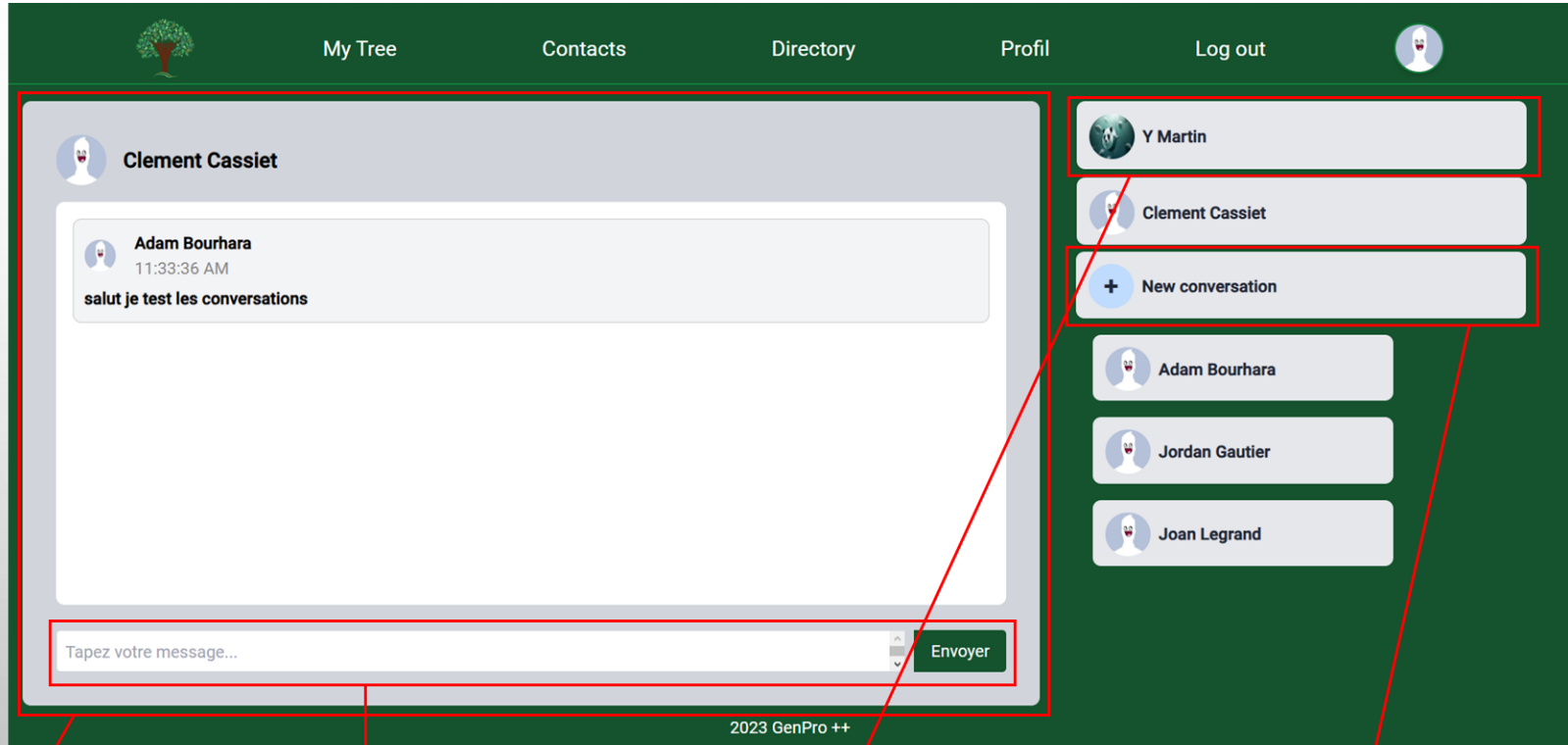
**Image 4 :** barre de recherche pour filtrer l'arbre, le champ par défaut est le nom et prénom

Search... type ? to get help. X

ad

- Admin System (user)
- Adam Bourhara (user)

# Contact Page



Partie d'affichage des conversations avec la personne selectionnée

Champ permettant l'écriture du message avec bouton pour envoi

Cadre permettant de naviguer entre les différentes conversations actives de l'utilisateur

Cadre permettant d'afficher les utilisateurs du site avec lesquels on n'a pas de conversation, et d'en créer une si l'on appuie sur le cadre correspondant

# Directory Page

Search by First name and Last name

**Adam Bourhara**  
Adam Bourhara's Tree | Size: 10  
[View Tree](#)

**Clement Cassiet**  
Clement Cassiet's Tree | Size: 1  
[View Tree](#)

**Jordan Gautier**  
Jordan Gautier's Tree | Size: 1  
[View Tree](#)

**Joan Legrand**  
Joan Legrand's Tree | Size: 2  
[View Tree](#)

**Y Martin**  
Y Martin's Tree | Size: 23

**Discover the GenPro ++ Directory**



2023 GenPro ++

Affichage de l'ensemble des utilisateurs du site web, avec nom, prénom, nom de l'arbre, taille de l'arbre ainsi qu'un lien pour s'y rendre


Lien pour accéder à l'arbre de l'utilisateur, fonctionnel si celui-ci est en public

Barre de recherche permettant de filtrer les utilisateurs du site en fonction du début de leur nom ou prénom

# Profile Page

My TreeContactsDirectoryProfilLog out

### Profil :



Change Profile Picture

Month views  
0

Annual views  
0

Tree length  
1

**Tree visibility :** ☐ Public ☐ Private  
**Go to the Tree**  

Submit modifications

### My information :

Firstname Adam	Lastname Bourhara
Secu number 1234567890123	Phone number +0000000000
Birthday 04/02/2002	Nationality nationality
Email adam@mail	Address adress
Sexe <input type="radio"/> Man <input checked="" type="radio"/> Woman <input type="radio"/> Other	
<div>Submit modification</div>	

Formulaire pour changer la visibilité de son arbre, bouton pour valider la modification (visibilité de l'arbre et photo de profil), et lien vers son arbre

Affichage de sa photo de profil et bouton pour la changer

Affichage de statistiques sur son arbre, comme sa taille (nombre de personnes), le nombre de vues sur le mois et sur l'année

Formulaire pour changer des informations sur son profil hors le nom et le prénom qui sont immuables

# Features du Back-End (non-exhaustive)



- **User registration** : Enregistre un utilisateur (user) en vérifiant que les champs sont conformes, avec cas particulier pour si une node ou un user lui "ressemblant" existe déjà en base de données. Quand un nouveau user est créé, son arbre ainsi que sa node sont également créés en base de données. La validation d'un administrateur est nécessaire avant que l'utilisateur puisse agir sur son compte.
- **Tree update** : Met à jour l'arbre dans la base de données. Un utilisateur est autorisé à sauvegarder des modifications dans l'arbre seulement s'il en est le propriétaire. L'utilisateur ne peut apporter des modifications qu'aux nodes qu'il a créés. De nouveaux nœuds peuvent être ajoutés s'ils sont liés à un nœud déjà présent dans l'arbre. Il est également possible de supprimer et de mettre à jour des nodes depuis l'arbre.
- **Node relations** : Sauf à la création d'un arbre, chaque node doit être connecté à un autre node déjà en base de données en respectant les relations suivantes : Parent (1 ou 2) et Partner. Les relations exPartners et Siblings existent également, mais il n'est pas encore possible d'établir ces liens directement depuis l'interface utilisateur. Il est possible d'ajouter un enfant à une node, mais la relation enfant n'existe pas. Un lien Parent est créé entre la nouvelle node et celle qui existe déjà.

# Features du Back-End



- **Conversation** : Il est possible d'avoir une conversation avec tous les utilisateurs présents sur le site. Seuls des messages textuels peuvent être échangés.
- **User Validation** : Lors de la création d'un compte utilisateur, un message est envoyé à un administrateur. Ce dernier a alors la possibilité de valider ou non la création du compte en fonction des informations renseignées à l'inscription.
- **Tree Merge Validation** : Lors de l'ajout d'une node similaire à une node en base de données, une demande peut alors être envoyée au créateur originel de la node. Un message est alors envoyé à cet utilisateur afin qu'il valide ou non l'opération. En cas de réponse positive, la node est alors ajoutée à l'arbre de la personne qui a fait la demande. Pour l'autre utilisateur, la node reliée est ajoutée à son arbre.
- **Validations** : Il est très simple d'ajouter un nouveau type de validation, par le biais de certains champs et fonction dans la classe Message.java et grâce au Design Pattern Strategy encapsulé dans la classe énumérée ValidationType.java. Grâce à cela aucune modification dans le front n'est nécessaire.
- **Visibilité des arbres** : Chaque arbre possède un niveau de visibilité : privé, protégé ou public. Un arbre public est visible par tous les utilisateurs, un arbre protégé est visible par les utilisateurs qui sont présents dans l'arbre. Un arbre privé n'est visible que par le créateur de l'arbre.
- **Nombre de visite** : Chaque fois qu'un utilisateur visite un arbre, une vue est ajoutée. Il est ainsi possible de connaître le nombre de visite par mois et par an.



# Highlights du Back-End

## “Singularité” des PersonInfo



Comme vous avez pu le constater dans la section architecture du back-end, les informations d'un utilisateur (user) ou d'un nœud (node) sont stockées dans le même type d'entité, à savoir le PersonInfo. L'entièreté du back-end, ainsi que sa conception, repose sur la singularité de chaque PersonInfo. Ainsi, le PersonInfo d'un utilisateur sera le même que celui de son nœud ; il en est de même si un nœud apparaît dans deux arbres distincts (deux instances de TreeNodes) : ils seront tous liés au même nœud qui possède son unique PersonInfo (une seule instance de Node et de PersonInfo).

Cette fonctionnalité garantit un maintien d'une base de données sans doublons de données et théoriquement donc plus rapide et optimisé. Malgré tout, cela implique une rigueur et une réflexion poussées pour la création des processus, car cela les rend plus complexes, vu qu'il est nécessaire de gérer plus de problèmes (on ne doit pas simplement créer une nouvelle instance, il faut garantir leur singularité notamment).

Ainsi, nous avons défini cinq champs qui définissent un PersonInfo (et donc ces cinq champs forment une clé primaire, en quelque sorte), régissant en partie le back-end, à savoir : LastName, FirstName, DateOfBirth, CountryOfBirth et CityOfBirth.



# Highlights du Back-End

## Algorithmie des Arbres



Un arbre possède un niveau de visibilité, d'un nom et de TreeNodes. Les TreeNodes permettent de relier une node à un arbre. Cette liste permet donc de savoir pour chaque arbre, les nodes qui sont dedans, et pour chaque node, à quel arbre elle appartient.

À la création d'une node, des vérifications sont faites afin de garantir son unicité. Si une node similaire est trouvée, distingue deux cas : La node est dans le même arbre ou dans un autre.

Si la node est dans le même arbre, l'ajout est bloqué. En effet, un node ne peut pas être à plusieurs endroit dans un même arbre.

Si la node est dans un autre arbre, on demande alors à l'utilisateur de vérifier ses informations. Si les informations sont correctes, il a alors la possibilité de demander au créateur originel de la node s'il accepte la liaison. En cas de réponse positive, la node et sa nouvelle relation sont alors ajoutées dans chaque arbre (seule une node est ajoutée de chaque côté). Tant que l'utilisateur n'a pas accepté ou refusé la fusion, aucuns de deux (envoyeur et receveur) ne peut modifier son arbre.

Si les informations sont correctes et qu'aucune node similaire n'est trouvé, alors de vérifications sont faites sur la cohérence de l'arbre. En effet, les dates de naissances et de morts de chaque node doivent être logique par rapport aux enfants et aux partenaires. De plus, la profondeur de chaque node doit être cohérente pour assurer que les plus jeunes soient en bas.

# Highlights du Back-End

## Sécurité des Champs



Tout champs reçus par le front dans les formulaires notamment (registration, ajout/update de node, profils ...) passe par un panel de vérifications.

La plupart des fonctions de vérification sont situés dans le Misc.java et appelés dans différents controllers.

Les données des gros formulaires sont notamment récupérées en LinkedHashMap. Les clefs doivent être autorisées pour passer (grâce à des constantes updatable dans Constants.java), et une fois autorisées chaque clef a des vérifications propres qui peuvent être communes aux autres.

Parmi les vérifications il y a, vérification de tentative de XSS qui est commune à toute, ou des vérifications plus simples, quant à l'acceptabilité des champs : date Of Birth et OfDeath < date du jour et d'autres vérifications de format avec des regex par exemple.

En fonction de leur destination les champs sont également vérifiés en accord avec ce qui est déjà présent en database. Une node et un user seront vérifiés pour empêcher les doublons en accord avec la "singularité" des PersonInfo.

Chaque tree a sa cohérence à vérifier à chaque ajout/update de node.

Chaque fois que le user update son profil les informations sont vérifiées également pour que leur conformité soit garantie.

# Highlights du Front-End

## Affichage et Maniabilité de l'arbre



Sur la partie Front-End, nous voulons mettre en avant l'affichage et la manipulation de l'arbre via l'API Family Tree JS que nous avons intégré pour le site Genepro ++. Cette implémentation permet une compréhension claire de la gestion de l'arbre par les utilisateurs, notamment sur l'ajout de personnes à son arbre en fonction du type de relation voulue.

De plus, nous voulons aussi souligner et rappeler la possibilité d'effectuer une recherche poussée sur l'arbre se basant sur tous les champs possibles. Ainsi que la visualisation précises des informations des personnes via un onglet situé à droite de la fenêtre du site.

## Système de communication

Enfin, notre système de chat nous satisfait pleinement. Ce dernier permet de discuter avec les autres membres. De plus, ce système permet de valider des actions, que ce soit pour les admins ou pour les utilisateurs. Et il est fait de tel sorte dans le Front, qu'en association avec une optimisation du back, n'importe qu'elle ajout de Validation n'engendre aucune modification à faire dans la page de conversation.

# Perspectives



Affichage des nœuds d'un arbre uniquement si l'utilisateur en a la permission (privacy de la node déjà disponible, mais restriction d'affichage pas encore fait)

Private code pour autorisation et permission : pour chaque requête, intégrer du `privateCode` de l'utilisateur connecté pour faire des vérifications d'autorisation/permissions (Ce paramètre est déjà implémenté dans les cookies dans cette perspective).

Finaliser les Test unitaires dans le Back : notre objectif était de faire des classes de tests pour toutes les classes qui sont présentes dans les packages `Model` et `Service`. Nos objectifs étaient d'avoir 90% de coverage sur ces 2 packages.

Actuellement les classes de tests existantes sont : `PersonInfoTest`, `UserTest`, `TreeTest`, `NodeTest`, `UserServiceTest`, `TreeServiceTest`, `NodeServiceTest`. Et elles n'ont malheureusement pas été updates depuis longtemps donc les dizaines de test unitaires faits fail pour la majorité d'entre eux.

Possibilité d'ajouter un nœud avec l'interface du front en utilisant les relations frères/sœurs/ex partenaire existantes dans le Back. Ces relations sont déjà implémentées dans le Back, mais pas encore au niveau du front.

La possibilité d'exporter son arbre en format imprimable (pdf) a été envisagée pour permettre un partage hors ligne de son arbre généalogique. Ainsi que l'opportunité pour les utilisateurs d'effectuer des actions en parallèle d'arbres d'autres membres du site, tel que la possibilité de "liker" des arbres ou de les commenter.

Faire un affichage textuel des arbres. Tous les `"toString"` sont créés côté back, donc serait facilement intégrable dans le front.

Enfin, nous souhaitons ajouter la possibilité de voir le profil des autres utilisateurs.

# Faiblesses du projet



Lourdeur des processus du back dû aux tests qui impliquent beaucoup de vérification et de validation.

Il reste probablement des cas particuliers qui provoquent des bugs dans la base de données => Peut être résolu en faisant plus de tests qui permettent de repérer et de corriger ces bugs.

Le sujet était très long, le temps de développement a donc été très long avant de pouvoir relier le Front et le Back, ce qui fait que le projet n'est pas abouti tel qu'on l'aurait souhaité.

Autorisation et sécurité, il reste encore des problèmes de sécurité, notamment au niveau de la vérification de qui envoie la requête. Ce Point serait largement résolu par l'ajout du `privateCode` de l'utilisateur connecté dans les Requetes, et d'une fonction de vérification d'autorisation. C'est l'un des points dont nous regrettons le plus ne pas avoir eu le temps de finalisé.



# Conclusion



En conclusion, ce projet a constitué une expérience extrêmement enrichissante tant sur le plan technique que sur celui de la méthodologie de travail en groupe. Sur le plan technique, nous avons acquis des compétences essentielles, notamment dans l'utilisation d'Angular, Tailwind CSS pour le développement frontend, ainsi que Spring Boot et Postman pour le backend. La diversité des technologies employées a permis d'approfondir notre compréhension des frameworks et outils modernes, renforçant ainsi notre bagage technique.

Parallèlement, le déroulement de ce projet sur plusieurs mois, caractérisé par sa densité et la nécessité d'intégrer de nombreuses fonctionnalités sur le site, a constitué un défi stimulant. Notre apprentissage s'est étendu au-delà des compétences techniques, englobant également la gestion efficace du temps et des ressources. La mise en place d'une organisation rigoureuse pour le partage des fichiers avec l'outil Git Hub, l'utilisation d'une base de données commune hébergé sur un NAS et la spécification claire des étapes ont été des éléments cruciaux pour la réussite du projet.

L'un des points forts majeurs de notre équipe a résidé dans son homogénéité, tous les membres du groupe ont été engagés et ont axé leur effort sur le travail collaboratif. Cette cohésion a grandement facilité la communication et la résolution des défis rencontrés tout au long du projet. Nous sommes convaincus que cette expérience en tant que reflète favorablement les dynamiques de travail que nous serons amenés à rencontrer en entreprise, et que nous rencontrons déjà actuellement en tant qu'alternants. En définitive, ce projet a été une expérience riche, stimulante, fastidieuse, mais surtout enrichissante pour tous.