



Projet Tuteuré
Sujet : Le Testament Numérique

RAPPORT TECHNIQUE

Données numériques :
Comment gérer leur vie après notre mort ?

Soutenu par :

Lucas ABITBOL

Jules CLAUDEL

Nelson ROGERS

Baptiste VILLEDIEU

Année universitaire 2020/2021

TABLE DES MATIERES

TABLE DES MATIERES	1
INTRODUCTION.....	2
I. PRÉSENTATION DE L'APPLICATION	3
A. Préambule.....	3
B. Inscription.....	3
C. Connexion utilisateur	5
D. Connexion famille	8
II. GESTION DE PROJET	10
A. Appropriation du projet	10
B. Gestion du temps	10
C. Outils utilisés.....	10
III. STRUCTURE DE L'APPLICATION	12
A. Modèle de données.....	12
B. Back-end	13
1. WebApplication	13
2. Entités	13
3. DAO	13
4. Contrôleurs	14
5. Authentification.....	14
6. SocialLogin.....	14
7. APITwitter	14
8. API Mail	15
9. Code commenté.....	15
C. HTML Templates.....	15
D. Aspect client (JavaScript & CSS).....	16
IV. SUITE DU PROJET	18
A. To Do List	18
B. Prolongements	19
CONCLUSION	20
ANNEXES	21

INTRODUCTION

“Aujourd’hui, les avancées technologiques provoquent une numérisation de plus en plus importante de toutes nos données. Apparaît alors le questionnement autour de ce que vont devenir nos données numériques après la mort lorsque nous ne pourrons plus les contrôler.” C’est sur ces mots que nous avons introduit notre projet il y a 7 mois. Le présent rapport a pour but de compléter ce qui a été expliqué dans notre dossier de veille technologique.

Dès le début de notre projet, nous avons compris qu’il était très important mais aussi très ambitieux. Ainsi, nous avons fait le choix de présenter ici une POC (Proof of Concept) permettant de mettre en place tous les éléments nécessaires à la création de cette application dans quelques années.

Nous allons donc vous présenter l’application telle qu’on l’imagine dans le futur. Nous allons également expliquer la structure de l’application telle qu’elle est aujourd’hui : back-end, front-end et modèle de données. Enfin, nous développerons une liste de problèmes non résolus et de prolongements possibles de notre Proof of Concept.

Vous pouvez accéder à l’application déployée sur : <https://legaceproject.herokuapp.com/>

I. PRÉSENTATION DE L'APPLICATION

A. *Préambule*

L'application Legac.e a pour but de s'occuper de l'avenir des données numériques de nos utilisateurs. Nous allons ici présenter la forme finale que nous voulions donner à notre application i. e. l'intégralité des fonctionnalités que nous souhaitions développer. Pour plus de détails sur notre concept, nous vous invitons à lire notre rapport de veille technologique élaboré au semestre précédent. Nous pouvons cependant résumer les 5 grandes étapes du fonctionnement de notre application :

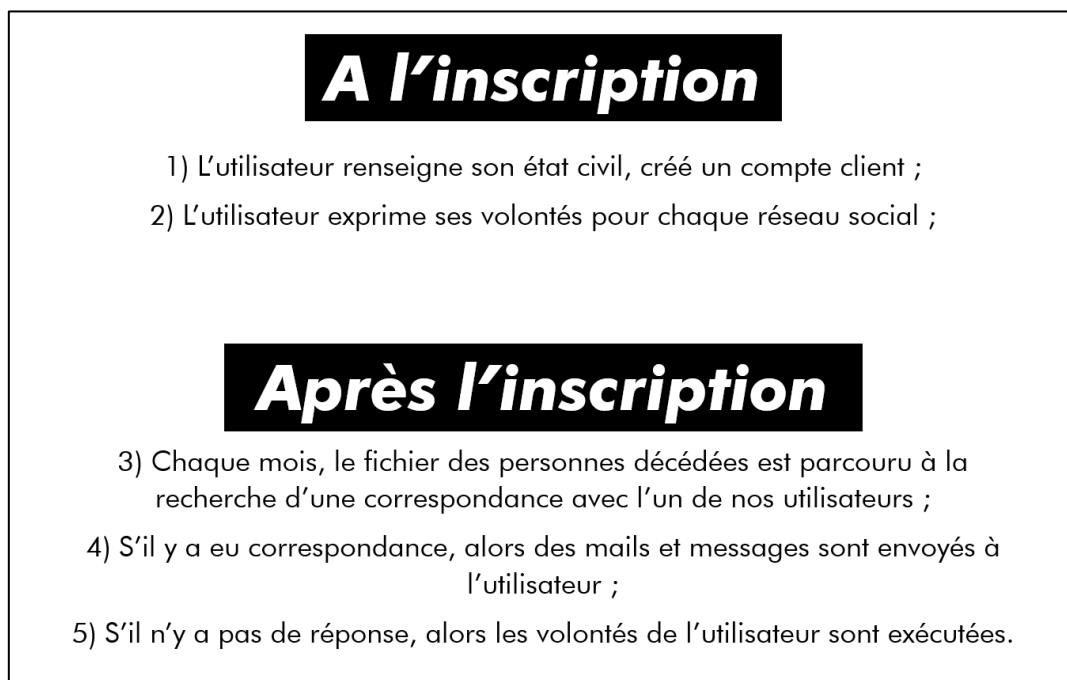


Figure 1 : Protocole en 5 étapes de notre application

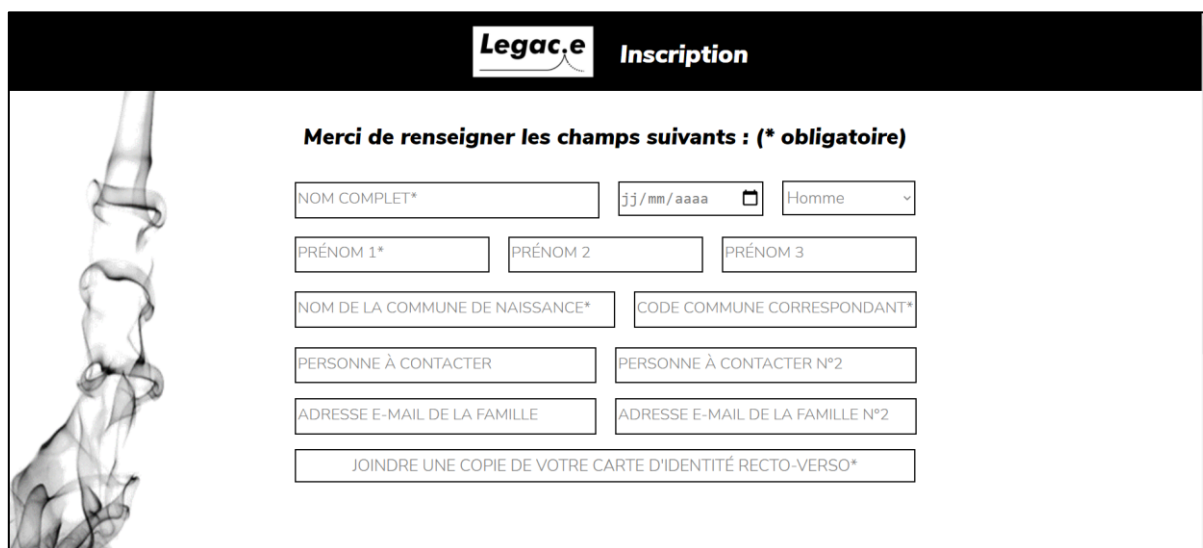
B. *Inscription*

La page d'accueil de notre application offre deux options : Se Connecter & S'inscrire.



Figure 2 : Page d'accueil de notre application

Lors de la création d'un compte, il est demandé de renseigner les informations d'identité : Nom, Prénom(s), Date de naissance, Commune de naissance, etc. L'utilisateur doit aussi désigner une personne à contacter et donner son adresse mail (jusqu'à deux contacts possibles). Il est aussi demandé de fournir une photo recto-verso de votre carte d'identité pour vérifier qu'il n'y ait pas d'usurpation.



The screenshot shows the 'Legac.e Inscription' page. The header is black with the 'Legac.e' logo and the word 'Inscription' in white. Below the header, there is a decorative image of a hand holding a pen on the left. The main content area is white and contains the following text and form fields:

Merci de renseigner les champs suivants : (* obligatoire)

NOM COMPLET* [input type="text"] jj/mm/aaaa [calendar icon] Homme [dropdown menu]

PRÉNOM 1* [input type="text"] PRÉNOM 2 [input type="text"] PRÉNOM 3 [input type="text"]

NOM DE LA COMMUNE DE NAISSANCE* [input type="text"] CODE COMMUNE CORRESPONDANT* [input type="text"]

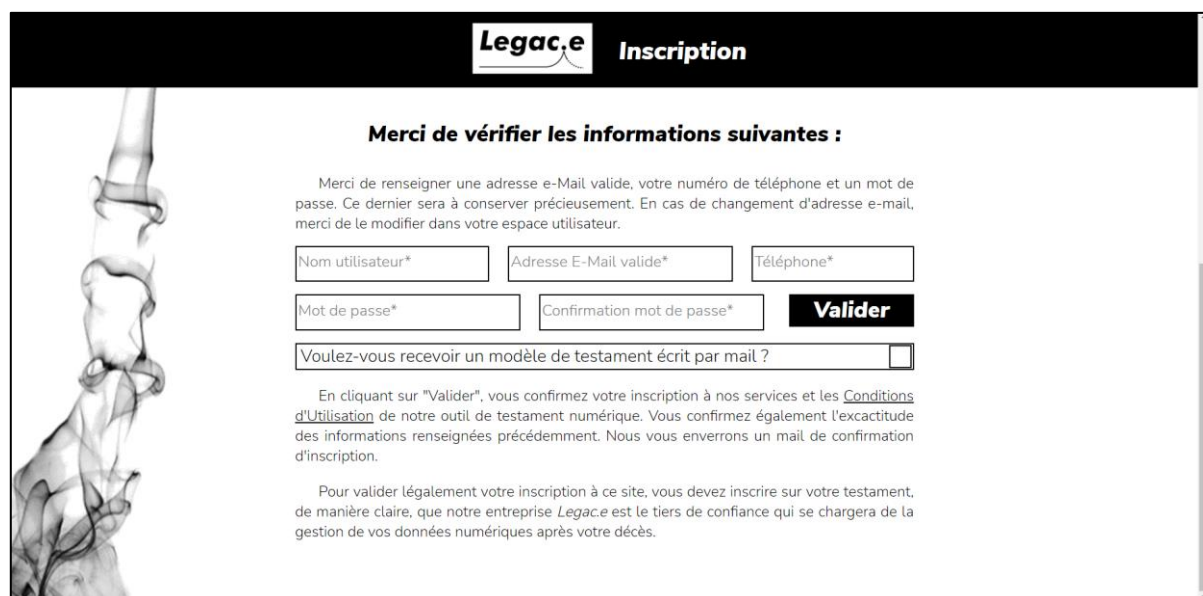
PERSONNE À CONTACTER [input type="text"] PERSONNE À CONTACTER N°2 [input type="text"]

ADRESSE E-MAIL DE LA FAMILLE [input type="text"] ADRESSE E-MAIL DE LA FAMILLE N°2 [input type="text"]

JOINDRE UNE COPIE DE VOTRE CARTE D'IDENTITÉ RECTO-VERSO* [input type="text"]

Figure 3 : Formulaire d'inscription, rubrique « Etat Civil »

De plus, il est demandé de renseigner un nom d'utilisateur, une adresse mail, un numéro de téléphone et un mot de passe (à confirmer) pour terminer la création du compte. L'utilisateur peut également recevoir par mail un modèle de testament écrit s'il le souhaite. Lorsqu'il valide son inscription correctement, il accepte les conditions d'utilisation et devra indiquer sur son testament écrit que Legac.e est le tiers de confiance qui s'occupera de ses données numériques à sa mort.



The screenshot shows the 'Legac.e Inscription' page, specifically the verification section. The header is black with the 'Legac.e' logo and the word 'Inscription' in white. Below the header, there is a decorative image of a hand holding a pen on the left. The main content area is white and contains the following text and form fields:

Merci de vérifier les informations suivantes :

Merci de renseigner une adresse e-Mail valide, votre numéro de téléphone et un mot de passe. Ce dernier sera à conserver précieusement. En cas de changement d'adresse e-mail, merci de le modifier dans votre espace utilisateur.

Nom utilisateur* [input type="text"] Adresse E-Mail valide* [input type="text"] Téléphone* [input type="text"]

Mot de passe* [input type="password"] Confirmation mot de passe* [input type="password"] **Valider** [button]

Voulez-vous recevoir un modèle de testament écrit par mail ? [checkbox]

En cliquant sur "Valider", vous confirmez votre inscription à nos services et les [Conditions d'Utilisation](#) de notre outil de testament numérique. Vous confirmez également l'exactitude des informations renseignées précédemment. Nous vous enverrons un mail de confirmation d'inscription.

Pour valider légalement votre inscription à ce site, vous devez inscrire sur votre testament, de manière claire, que notre entreprise Legac.e est le tiers de confiance qui se chargera de la gestion de vos données numériques après votre décès.

Figure 4 : Formulaire d'inscription, rubrique « Vérification »

Après son inscription, l'utilisateur est automatiquement redirigé vers la page d'accueil des utilisateurs. Dans le même temps, un mail lui est envoyé pour confirmer son inscription.

C. Connexion utilisateur

La connexion d'un utilisateur se fait avec son nom d'utilisateur et son mot de passe. Si l'un de ces champs est incorrect, une erreur s'affiche.

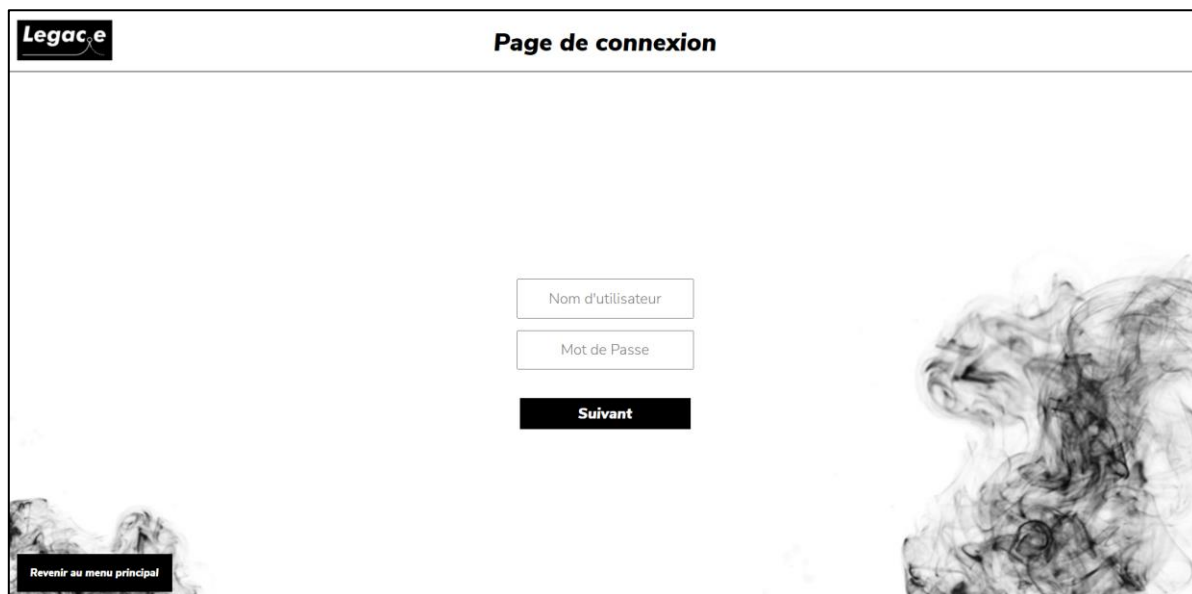
The screenshot shows the 'Page de connexion' (Login Page) of the Legac.e website. At the top left is the 'Legac.e' logo. The title 'Page de connexion' is centered at the top. The main area contains two input fields: 'Nom d'utilisateur' (Username) and 'Mot de Passe' (Password). Below these fields is a black button with the white text 'Suivant' (Next). In the bottom left corner, there is a small black button with the white text 'Revenir au menu principal' (Return to main menu). The background features abstract, swirling smoke-like graphics on the right and bottom left.

Figure 5 : Page de connexion

Après avoir renseigné convenablement ces champs, l'utilisateur a accès à une page de présentation de l'espace utilisateur. Il peut facilement naviguer entre ses informations personnelles, ses préférences réseaux et sa page souvenir. De plus, la barre de navigation offre toujours une option de déconnexion.

The screenshot shows the 'Bienvenue dans l'espace utilisateur de Legac.e' (Welcome to the Legac.e user space) page. At the top is a black navigation bar with the 'Legac.e' logo on the left and four links: 'Informations Utilisateur', 'Préférences Réseaux', 'Page souvenir', and 'Déconnexion'. The main heading is 'Bienvenue dans l'espace utilisateur de Legac.e'. Below this, there are three paragraphs of text: 'Retrouvez ici toutes les informations importantes de votre compte dans l'onglet « Informations Utilisateur ».', 'Vous pourrez paramétrer l'avenir de vos données numériques dans l'onglet « Préférences Réseaux ».', and 'Vous pourrez également créer une « Page souvenir » à laquelle vos proches auront accès dans l'onglet du même nom.' The background features abstract, swirling smoke-like graphics on the right.

Figure 6 : Page d'accueil du côté Utilisateur

Sur la page "Informations Personnelles", il peut consulter et/ou modifier ces dernières à l'aide de deux icônes. L'icône loupe permet de télécharger le fichier d'inscription contenant toutes les informations entrées lors de l'inscription tandis que l'icône formulaire renvoie vers le formulaire d'inscription que l'utilisateur peut modifier.

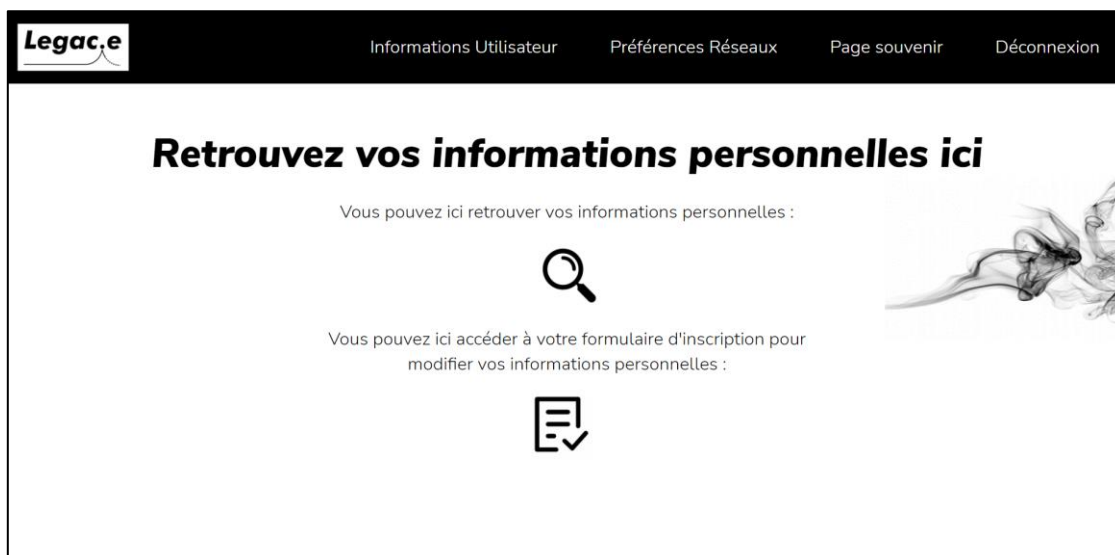


Figure 7 : Page « Informations Utilisateur »

Sur la page “Préférences Réseaux”, l'utilisateur peut indiquer ses volontés quant à l'avenir de ses données pour chaque réseau. Il peut indiquer s'il utilise chaque réseau parmi “PayPal, Pinterest, Facebook, Twitter, Instagram et Google”. En fonction de sa sélection, des options s'affichent pour chaque réseau. Par exemple, pour Facebook, l'utilisateur a le choix entre “Avoir accès à la procédure pour transformer son compte en compte de commémoration” et “Que sa famille reçoive, à sa mort, le formulaire pour fermer son compte Facebook”. Les options disponibles pour chaque réseau sont disponibles [ICI](#) ou en [Annexe 1](#).

C'est également sur cette page qu'il peut se connecter, comme dans le cas de Twitter par exemple.

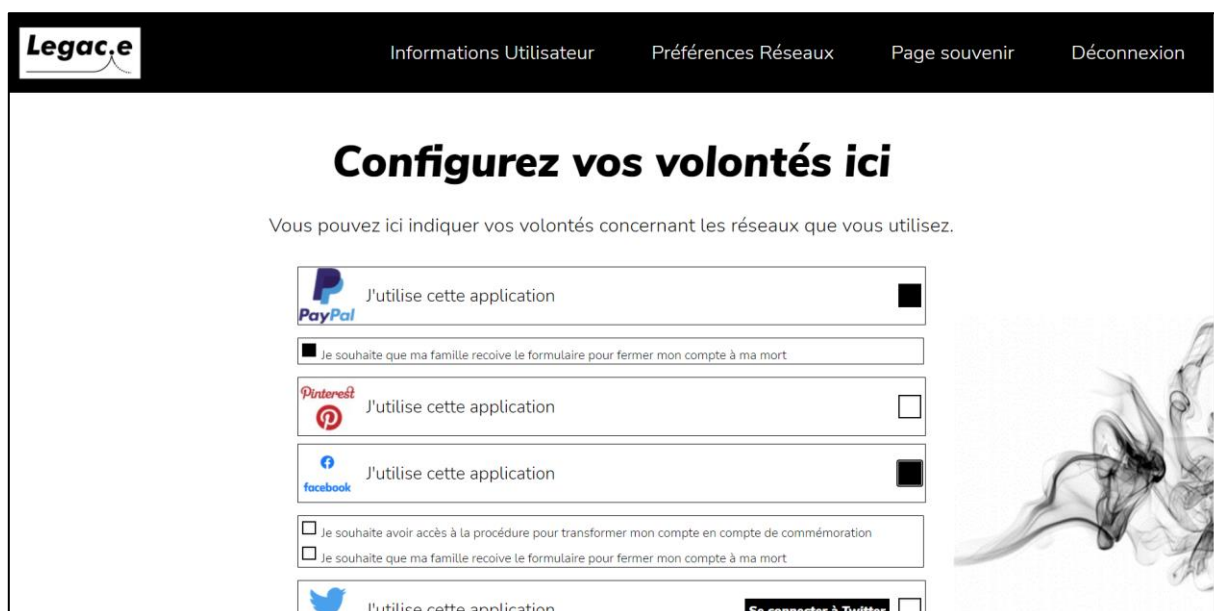


Figure 8 : Page « Préférences Réseaux » côté Utilisateur

Quand il clique sur “Se connecter à Twitter”, l'utilisateur est renvoyé vers une page “Connexion à Twitter”. Sur cette page, l'utilisateur peut se connecter à l'aide d'un bouton ; quand il est connecté, sa photo de profil et son nom d'utilisateur apparaissent :

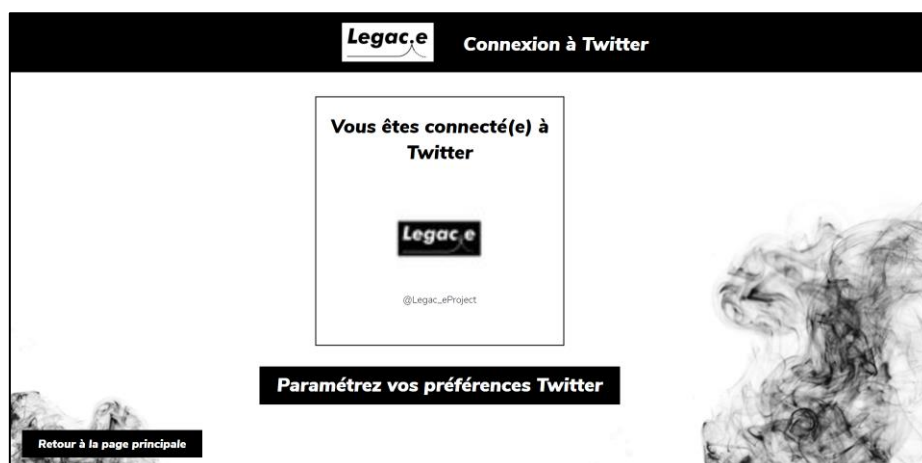


Figure 9 : Page « Connexion à Twitter », dans le cas où un compte Twitter est connecté

En cliquant sur « Paramétrez vos préférences Twitter », l'utilisateur peut entrer des paramètres qui permettront à notre application d'exécuter ses volontés comme l'ID du Tweet s'il veut retweeter à son décès. Il peut aussi indiquer le destinataire et le contenu d'un MP à envoyer après son décès.

Figure 10 : Page « Connexion à Twitter », dans le cas où un compte Twitter est connecté

Sur la page Souvenir, l'utilisateur peut déposer des photos, des vidéos ou encore des documents auxquels sa famille aura accès lorsqu'il sera décédé. Il peut également entrer un message dans la zone de texte prévue à cet effet.

Figure 11 : Page Souvenir, côté Utilisateur

D. Connexion famille

L'application comprend également tout un côté dédié à la famille du défunt sur lequel elle peut se connecter à l'aide des informations contenues dans un mail reçu à la mort de leur proche. La page de connexion est identique que vous soyez utilisateur ou famille. L'application détecte automatiquement la nature de l'identifiant rentré et vous redirige donc en fonction vers l'espace dédié. Nous avons souhaité que le côté famille soit simplifié au possible car nous connaissons la difficulté de traverser une période de deuil. Ainsi, chaque page est optimisée et claire de manière à simplifier la navigation des proches sur notre site.

Après s'être connecté, le proche a accès à une page de présentation de l'espace famille. Il peut naviguer à l'aide d'une barre de navigation entre les différentes rubriques de notre application.



Figure 12 : Page d'accueil du côté Famille

La page "Préférences Réseaux" permet à l'utilisateur de voir les volontés de son proche et, si c'était le souhait de ce dernier, d'accéder aux tutoriels concernés. Par exemple, un renvoi sur la page dédiée permettra à l'utilisateur d'envoyer une demande pour fermer le compte Pinterest de son proche si c'était la volonté de ce dernier.



Figure 13 : Page « Préférences Réseaux » côté Famille

Enfin, la page souvenir permet de télécharger les images, vidéos, documents ou messages que l'utilisateur a renseigné de son vivant.



Figure 14 : Page Souvenir, côté Famille

II. GESTION DE PROJET

A. *Appropriation du projet*

Dans un premier temps, il nous a fallu appréhender et comprendre notre projet. Comme dit précédemment, nous avons tout de suite compris que notre projet était ambitieux et qu'il allait demander du travail et de l'effort.

Cependant, nous avons pu nous aider du TP Galerie d'art pour comprendre de quelle manière implémenter notre modèle de données en Spring. Nous avons donc décidé de reprendre le TP ensemble, et de s'en servir dès le démarrage du projet pour partir sur des bases solides.

Nous avons eu la chance d'avoir des personnalités différentes dans notre groupe et de pouvoir séparer les tâches. Dès le départ, nous savions que Jules et Baptiste préféraient le côté front-end, alors que Nelson et Lucas étaient plus à l'aise avec le back-end.

En partant de ce postulat, les rôles se sont vite attribués et chacun a pu se lancer dans ce qu'il pensait le mieux maîtriser pour faire avancer le projet le plus rapidement possible. Peu à peu, Jules s'est naturellement imposé en tant que "Chef de projet". Il arrivait à motiver l'équipe et à répartir les tâches au mieux. Il avait aussi un peu le rôle de "Référént front-end". Nelson était lui le "Référént back-end" : tout ce qui touchait de près ou de loin au back-end devait être discuté avec lui. Baptiste et Lucas avaient des rôles plus hybrides : ils ont touché à tout, quitte à sortir de leur zone de confort. Ils ont également été responsables des recherches supplémentaires que nous avons dû faire, notamment sur l'envoi de mail et l'API Twitter.

B. *Gestion du temps*

Le temps n'a pas été facile à gérer dans ce projet. Nous pensions au départ ne pas avoir le temps de boucler notre projet, mais nous avons finalement réussi à implémenter notre modèle de données et à naviguer entre les pages de notre application. Le front-end a également rapidement avancé : les différentes pages ayant été codées selon les maquettes déjà effectuées.

Cependant, nous nous sommes retrouvés bloqués à ce niveau-là, nous n'arrivions pas à implémenter les méthodes que nous voulions, et nous étions tous en difficulté. Beaucoup de temps a été perdu à ce moment-là, et nous avons donc dû sélectionner les fonctionnalités qui étaient essentielles à notre application. Le projet, tel qu'il est soutenu, est malgré tout très satisfaisant car il est fonctionnel.

Vous pouvez retrouver ICI ou en **Annexe 2**, notre diagramme de Gantt complet et détaillé de l'avancement de notre projet.

C. *Outils utilisés*

Pour déployer notre application, nous avons utilisé plusieurs outils. Nous avons développé sur nos machines personnelles et sur des IDE différents : IntelliJ et NetBeans. Aucun conflit n'a été généré grâce à l'utilisation d'un répertoire GitHub tout au long du semestre.

Pour le côté front-end, nous avons d'abord utilisé CodeSandbox pour chacune de nos pages, afin d'avoir un visuel dynamique selon nos modifications. Nous avons ensuite intégré les codes HTML/CSS/JS à notre projet.

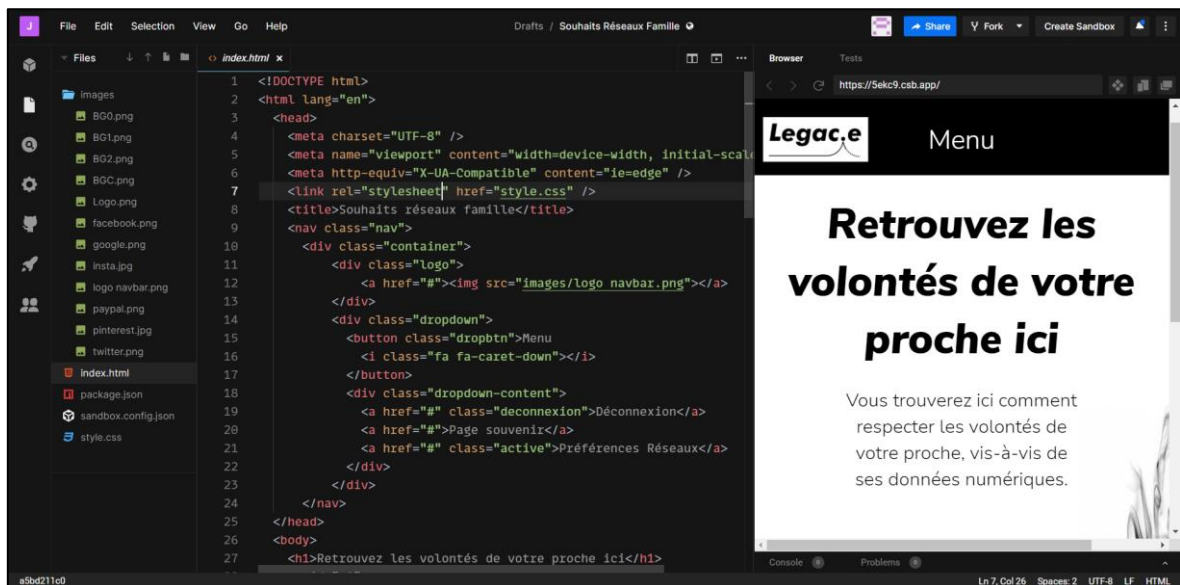


Figure 15 : Capture de l'éditeur de CodeSandBox

Nous avons utilisé des Framework Java pour certaines fonctionnalités, Spring et Heroku afin de déployer notre projet. Nous nous sommes également beaucoup documentés sur Internet afin de pallier tous les problèmes que nous avons rencontrés.

Enfin, nous avons utilisé plusieurs outils de gestion de projet comme Zoom ou Discord pour la tenue des réunions, Messenger pour communiquer, Google Drive pour la rédaction du rapport ou Google Keep pour la liste des pages à coder en front par exemple :

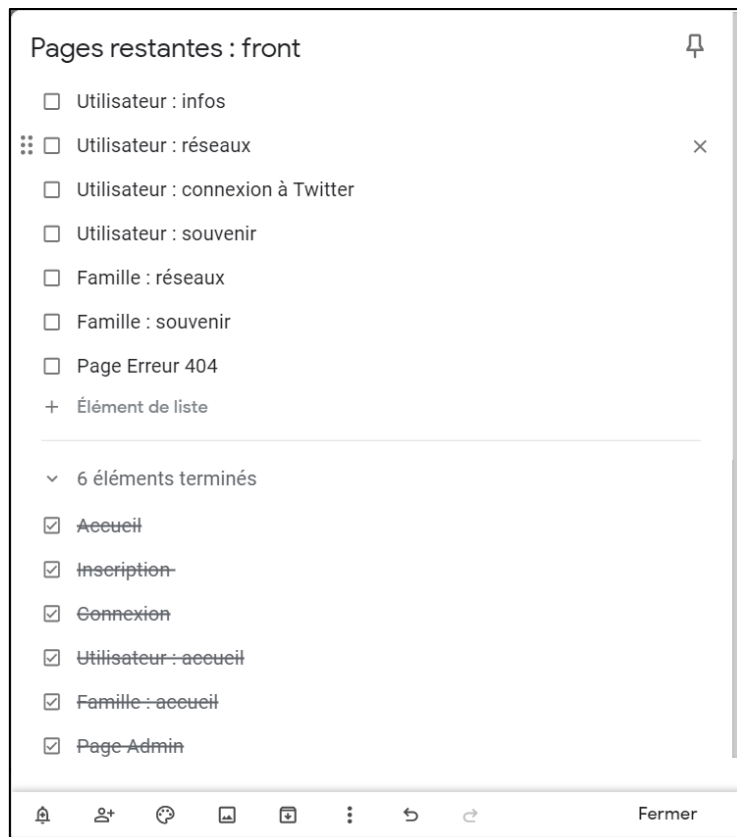


Figure 16 : Liste Google Keep des pages restant à coder en front

III. STRUCTURE DE L'APPLICATION

A. **Modèle de données**

Dans le cadre de ce projet, il a été nécessaire de créer un modèle de données. Ce dernier a d'ailleurs subi de nombreuses mises à jour tout au long de notre projet. Vous pouvez le retrouver [ICI](#) ou en **Annexe 3**.

Il est construit autour de la classe **Utilisateur**. Cette classe contient toutes les informations d'un utilisateur : Nom, Prénom(s), Sexe, Date de naissance... Un utilisateur peut avoir plusieurs rôles, ce qui explique la multiplicité entre les classes **Rôle** et **Utilisateur** ; il peut être administrateur ou simple utilisateur.

Ensuite, les classes **Utilisateur** et **Volonté** sont liées par une relation oneToMany, ce qui signifie qu'un utilisateur peut avoir plusieurs volontés. De plus, un utilisateur peut réaliser des souhaits concernant plusieurs réseaux sociaux, ce qui explique la multiplicité entre la classe **Utilisateur** et la classe **SocialConnection**. L'utilisateur doit pouvoir choisir les actions qu'il veut donc la classe **SocialConnection** est connectée avec la classe **Action**. Ainsi, dans la classe **Volonté** apparaît un attribut de type **Action** pour retransmettre ce que l'utilisateur veut faire de ses données numériques après sa mort. Les classes des réseaux héritent de la classe **Action** comme celles qui sont sur le modèle de données.

Enfin, les classes d'actions comme **Retweet** ou **DirectMessage** héritent des classes des réseaux (**TwitterAction**). En effet, ces classes représentent les actions que notre application peut réaliser pour chaque réseau social. Par exemple, **DirectMessage** permet d'envoyer un message privé à un proche par Twitter ; il faut donc pour cela entrer l'identifiant Twitter du proche ainsi que le contenu du message à envoyer. De l'autre côté, pour retweeter un tweet, il suffit de renseigner l'identifiant du Tweet en question.

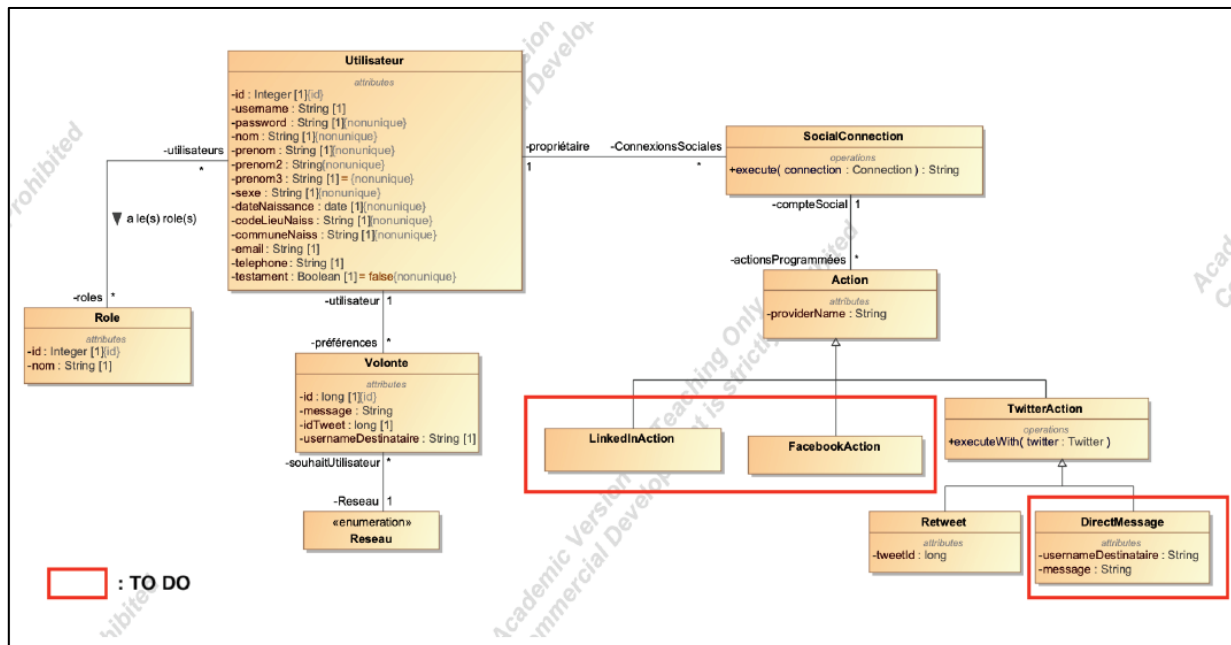


Figure 17 : Modèle de données correspondant à notre projet

B. Back-end

Cette partie étant très importante mais aussi très complexe, nous avons décidé de la séparer en plusieurs sous-parties :

1. WebApplication

Cette classe est centrale dans notre projet car c'est elle qui permet de lancer notre application. C'est également elle qui nous a permis de tester nos modifications tout au long du semestre grâce au localhost.

2. Entités

Nous avons créé 4 classes JAVA et 1 Enumération selon le modèle de données. On y a renseigné leurs attributs ainsi que leurs liaisons (OneToMany, ManyToMany...). Par exemple, l'entité Rôle est définie par 2 attributs : l'identifiant "ID" généré automatiquement et le nom "name", non nul.

On remarque aussi la présence d'une liste d'utilisateurs "users", essentielle pour connecter l'entité Rôle à l'entité Utilisateur. Avec la fonction ManyToMany utilisant la valeur "roles" initialisée dans la classe Utilisateur, on est capable d'associer chaque utilisateur à plusieurs rôles et inversement.

```
@Entity
// Lombok
@Getter @Setter @NoArgsConstructor @RequiredArgsConstructor @ToString
public class Role {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Setter(AccessLevel.NONE)
    private Integer id;

    @NonNull
    @Column(unique = true)
    private String name;

    @ManyToMany(mappedBy = "roles")
    @ToString.Exclude
    @Setter(AccessLevel.NONE)
    private List<Utilisateur> users = new LinkedList<>();
}
```

Figure 18 : Capture de la classe entité « Role »

L'Enum "Réseau" permet, elle, de répertorier les différents réseaux dont les services sont proposés par notre application. On y retrouve actuellement Twitter, Facebook & LinkedIn.

3. DAO

Nous avons créé des répertoires dans le package "testament.dao" afin d'enregistrer dans notre base de données les utilisateurs et volontés qui y sont associées. Nous pouvons ainsi les manipuler dans notre application. Il existe également un package "testament.dao.social" qui contient les répertoires relatifs à la connexion aux réseaux des utilisateurs.

4. Contrôleurs

Les contrôleurs nous permettent de connecter les pages les unes aux autres et d'enregistrer des informations dans notre base de données. Nous en avons créé 6 :

- Admin : concerne les éléments présents sur la page administrateur. On y accède en étant administrateur.
- LoginAndRegistration : ce contrôleur gère l'inscription et la connexion étant donné que ces deux éléments sont reliés à la page d'accueil. Il permet aussi d'envoyer un mail à l'utilisateur lorsqu'il est inscrit.
- Scanner : comme son nom l'indique, il s'occupe de scanner le registre des personnes décédées et permet d'établir une correspondance avec notre base de données.
- User : il permet à l'administrateur ou à l'utilisateur de naviguer entre toutes les pages de l'espace utilisateur.
- Volonte : ce contrôleur permet d'accéder au formulaire des préférences réseaux. Il permet également à un utilisateur d'enregistrer ses volontés par rapport Twitter.
- PostTweet : ce contrôleur permet de gérer les actions relatives au compte Twitter du défunt. C'est dans ce contrôleur qu'il faudrait implémenter les fonctions de PostTweet, sendDirectMessage, getTweet... Il ne réalise, à ce stade du projet, que des retweets.

5. Authentification

L'authentification à notre site est gérée par la classe WebSecurityConfig. Elle permet notamment de crypter les mots de passe, par exemple. Le package "testament.service" gère l'authentification : l'inscription, la connexion, la répartition des rôles, la sécurité de notre application...

Enfin, la classe UserValidator nous permet de confirmer le format des mots de passe rentrés. Elle gère également la confirmation du mot de passe.

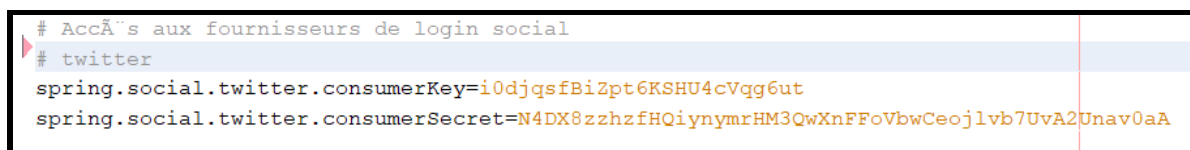
6. SocialLogin

La connexion à Twitter depuis notre application est gérée par les classes SocialConfig, les packages "testament.config.restemplate", "testament.util" et "testament.security".

7. APITwitter

Pour utiliser l'API proposée par Twitter, il nous a fallu demander une autorisation. Pour cela, nous avons dû nous inscrire sur Twitter Developer et motiver notre projet auprès du portail. Vous pouvez retrouver notre explication en anglais [ICI](#) ou en [Annexe 4](#).

Après avoir été accepté, nous avons obtenu des clés et des tokens d'accès pour nous permettre de mettre en place les volontés des utilisateurs comme poster un tweet ou retweeter par exemple... Nous avons dû insérer ces clés dans le fichier application.properties :



```
# Accès aux fournisseurs de login social
# twitter
spring.social.twitter.consumerKey=i0djqsfbizpt6KSHU4cVqg6ut
spring.social.twitter.consumerSecret=N4DX8zzhzfHQiynymrHM3QwXnFFoVbwCeojlvb7UvA2Unav0aA
```

Figure 19 : Capture du fichier application.properties

L'API Twitter permet à l'utilisateur de s'authentifier depuis notre application. On remarque que lorsqu'il s'inscrit, le logo de notre application apparaît à la droite de notre écran, preuve que nos clés d'accès sont utilisées dans le processus.

Les informations nous permettant d'implémenter les actions Twitter voulues ont été trouvées sur le Spring Social Twitter Reference Manual à l'adresse suivante :

<https://docs.spring.io/spring-social-twitter/docs/1.0.5.RELEASE/reference/htmlsingle/#twitter-directMessages>

Les méthodes proposées par l'API Twitter sont principalement utilisées dans le PostTweetController.

8. API Mail

Nous avons la volonté, dans notre projet, de pouvoir envoyer des mails dans deux cas de figure. Premièrement, lorsqu'un utilisateur s'inscrit sur notre site, il doit recevoir automatiquement un mail de confirmation d'inscription.

Pour cela nous avons implémenté une méthode `envoiMail(String email)` dans la classe `loginAndRegistrationController`. Cette méthode est assez simple d'utilisation et se sert du framework `mail.javamail.Javamailsender`. Il nous a alors suffi d'appeler la méthode `envoiMail` dans le `@PostMapping("/registration")` et le mail défini peut s'envoyer automatiquement lorsque l'utilisateur valide le formulaire d'inscription.

Nous voulions également que l'utilisateur ait la possibilité de recevoir, s'il le souhaite, un modèle de testament écrit. Pour cela, nous avons implémenté une méthode permettant d'envoyer en pièce jointe un fichier "Testament.pdf" comprenant notre modèle de testament numérique écrit.

Cette nouvelle méthode, appelée `envoiTestament(email)`, reprenait comme la première l'email de l'utilisateur qui s'inscrit grâce au même getter. Nous avons cependant dû modifier notre entité `Utilisateur` afin d'y ajouter le paramètre "Testament". En effet, nous avons besoin de ce booléen pour savoir si l'utilisateur avait, lors de son inscription, coché ou non la case "Je souhaite recevoir un modèle de testament numérique". S'il la coche, le booléen devient `True` et la méthode est donc appelée, et s'il ne la coche pas, le booléen reste `False` et la méthode n'est pas appelée.

9. Code commenté

En parcourant le code, vous pourrez trouver des morceaux de code commentés. Ce code correspond à des pistes, des fonctionnalités non-abouties cette année. En effet, nous sommes toujours en train de travailler sur les actions pour poster un tweet ou envoyer un Direct Message mais nous rencontrons quelques bugs. Nous avons également travaillé sur une fonctionnalité permettant d'exécuter automatiquement les volontés des utilisateurs lorsqu'ils sont détectés sur le fichier de personnes décédées.

C. HTML Templates

Comme énoncé précédemment, nous avons initié la création de toutes nos vues HTML/JS/CSS sur CodeSandbox. Nous les avons créées indépendamment, en s'inspirant des maquettes faites au préalable et de manière à simplifier leur intégration (avec le CSS et le reste du code).

Une fois terminées, nous avons relié certaines de nos vues à notre projet en modifiant les HTML et CSS pour qu'ils soient compatibles avec Spring. En effet, il a été nécessaire, afin de relier nos vues à notre application web selon le modèle MVC, d'utiliser les fonctionnalités de Thymeleaf permettant d'utiliser les objets (text, action...) en tant qu'attributs dans les classes Controller.

Ça a été le cas pour les pages d'accueil et le formulaire d'inscription. Ce dernier nous a demandé beaucoup de réflexion étant donné qu'il est très complexe.

Cependant, nous avons continuellement modifié nos pages en fonction de nos besoins. Nous avons par exemple eu besoin de créer la vue "Connexion à Twitter" à la fin du projet car nous nous sommes rendus compte qu'il était plus pertinent, selon notre modèle de données, d'en faire une page à part.

Enfin, contrairement au back-end, nous avons été capable de créer l'intégralité des vues, y compris celles qui correspondent au côté famille.

Vous pouvez retrouver [ICI](#) ou en [Annexe 5](#) le schéma de navigation de notre projet.

D. Aspect client (JavaScript & CSS)

L'aspect client a été fortement inspiré des maquettes effectuées au premier semestre. Vous pourrez retrouver ces dernières [ICI](#) ou en [Annexe 6](#). Nous voulions un aspect sobre et sans fioritures. Nous avons donc choisi un style simple caractérisé par :

- un fond blanc avec une texture de fumée, différente selon le "côté" de l'application dans lequel on se trouve ;
- une police, déclinée en deux graisses pour les titres et les corps de texte ;
- une navbar uniforme sur le site proposant une navigation claire parmi toutes les pages pertinentes.

Nous avons réussi à coder le site de manière que la taille de la fenêtre n'influe pas sur l'organisation de la page : l'application s'adapte à la taille de la fenêtre. La navbar est telle que, lorsque la taille de la fenêtre ne permet plus d'afficher toutes les pages, un menu déroulant s'affiche et permet de naviguer sur toute l'application.



Figure 20 : Menu déroulant qui s'affiche lorsque la fenêtre est réduite

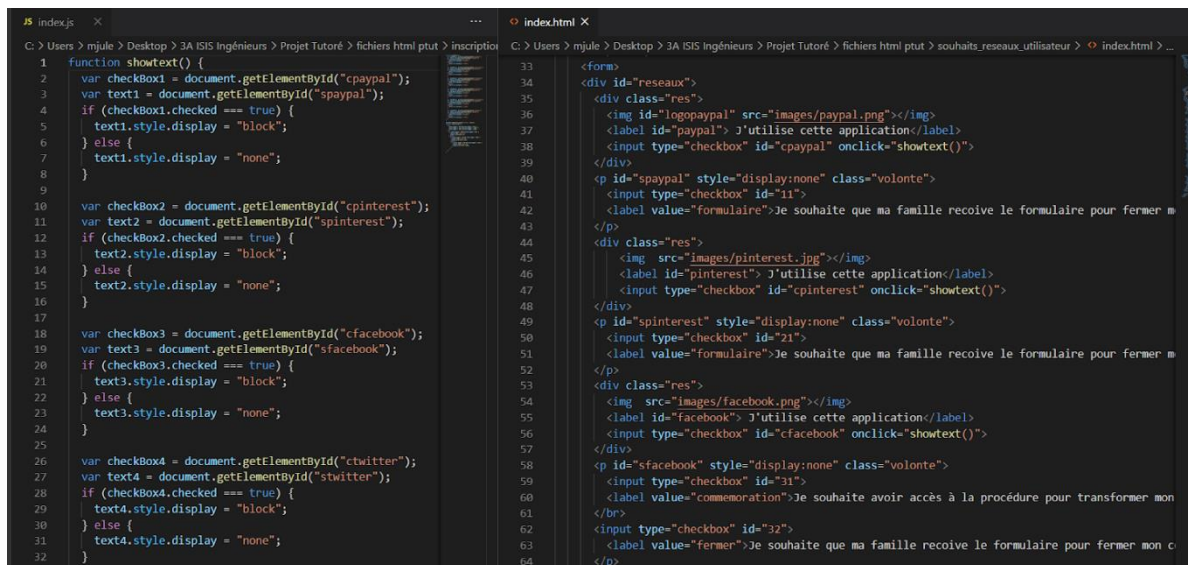
Nous avons codé cela en CSS, en utilisant la règle @media qui permet d'associer un ensemble d'instructions imbriquées avec une condition définie : ici, lorsque la largeur de la fenêtre est inférieure à 1300px, le menu déroulant s'affiche.

```
49  /* Media query section */
50
51  @media screen and (max-width: 1300px) {
52      /* The dropdown container */
53      .dropdown {
54          z-index: 999;
55          color: white;
56          overflow: hidden;
57      }
58  }
```

Figure 21 : Utilisation de la règle @media pour la navbar

Nous n'avons pas beaucoup développé en JavaScript tout au long de ce projet. Nous avons essayé d'utiliser une API gouvernementale pour trouver automatiquement le code de la commune quand on rentrait son nom mais, pour une raison totalement inconnue, l'API cessait de fonctionner plusieurs fois par jour alors même qu'il n'y avait pas de token d'utilisation limitée. La seule fonction que nous avons conservée en JavaScript est liée à la page "Préférences Réseaux" des utilisateurs. Elle permet d'afficher les possibilités liées à chaque réseau quand on coche la case "J'utilise cette application".

Concrètement, un "onclick" appelle une fonction showtext() qui identifie l'état de la case à cocher et en conséquence, le texte à afficher grâce à une fonction display ; et ce, pour chaque réseau :



```
1  function showtext() {
2      var checkBox1 = document.getElementById("cpaypal");
3      var text1 = document.getElementById("spaypal");
4      if (checkBox1.checked === true) {
5          text1.style.display = "block";
6      } else {
7          text1.style.display = "none";
8      }
9
10     var checkBox2 = document.getElementById("cpinterest");
11     var text2 = document.getElementById("spinterest");
12     if (checkBox2.checked === true) {
13         text2.style.display = "block";
14     } else {
15         text2.style.display = "none";
16     }
17
18     var checkBox3 = document.getElementById("cfacbook");
19     var text3 = document.getElementById("sfacbook");
20     if (checkBox3.checked === true) {
21         text3.style.display = "block";
22     } else {
23         text3.style.display = "none";
24     }
25
26     var checkBox4 = document.getElementById("ctwitter");
27     var text4 = document.getElementById("stwitter");
28     if (checkBox4.checked === true) {
29         text4.style.display = "block";
30     } else {
31         text4.style.display = "none";
32     }
33 }
34
35 <form>
36 <div id="reseaux">
37     <div class="res">
38         </img>
39         <label id="paypal"> J'utilise cette application</label>
40         <input type="checkbox" id="cpaypal" onclick="showtext()">
41     </div>
42     <p id="spaypal" style="display:none" class="volonte">
43         <input type="checkbox" id="11">
44         <label value="formulaire">Je souhaite que ma famille recoive le formulaire pour fermer m
45     </p>
46     <div class="res">
47         </img>
48         <label id="pinterest"> J'utilise cette application</label>
49         <input type="checkbox" id="cpinterest" onclick="showtext()">
50     </div>
51     <p id="spinterest" style="display:none" class="volonte">
52         <input type="checkbox" id="21">
53         <label value="formulaire">Je souhaite que ma famille recoive le formulaire pour fermer m
54     </p>
55     <div class="res">
56         </img>
57         <label id="Facebook"> J'utilise cette application</label>
58         <input type="checkbox" id="cfacbook" onclick="showtext()">
59     </div>
60     <p id="sfacbook" style="display:none" class="volonte">
61         <input type="checkbox" id="31">
62         <label value="commemoration">Je souhaite avoir accès à la procédure pour transformer mon
63     </p>
64     <input type="checkbox" id="32">
65     <label value="fermer">Je souhaite que ma famille recoive le formulaire pour fermer mon c
66 </p>
```

Figure 22 : Extraits du JS et du code HTML relatifs à la fonction showtext()

Enfin, nous avons mis notre logo en favicon et avons configuré le titre de chaque page afin que la navigation soit la plus claire possible.

IV. SUITE DU PROJET

A. *To Do List*

Comme évoqué précédemment, le deadline de ce projet ne nous a pas permis de développer toutes les fonctions que nous souhaitions initialement.

En effet, la quasi-totalité du back-end correspondant à la partie famille n'est pas codée. Il faudra tout d'abord envoyer un mail aux proches des utilisateurs décédés avec les identifiants leur permettant de se connecter à notre site. Ensuite, il restera à relier la page "Espace Famille" avec la page de connexion selon le type d'identifiant utilisé.

Il faudra également lier la "Page Souvenir" et la page "Préférences Réseaux" selon les informations de l'utilisateur et ses souhaits. En faisant cela, la famille du défunt aura donc accès aux photos, vidéos ou encore messages déposés sur notre site. Il faudra ainsi se renseigner sur l'enregistrement des pièces jointes laissées. Il faudra en faire de même pour la page concernant les préférences pour chaque réseau ainsi que pour la carte d'identité à l'inscription.

Il faudrait également ajouter une fonction de déconnexion ou une option de comptes multiples sur la page « Connexion à un réseau social ». En effet, pour supprimer un utilisateur actuellement, il faut que l'administrateur le fasse depuis sa base de données ou que l'utilisateur révoque l'autorisation depuis ses préférences sur son compte Twitter.

Du côté de l'espace utilisateur, notre site ne permet pas encore de consulter et modifier les informations indiquées à l'inscription. Il faudrait donc aussi développer cette fonction pour que les utilisateurs puissent, à tout moment, changer leurs informations personnelles. Il serait aussi intéressant sur cette même page d'ajouter un bouton permettant à un utilisateur de supprimer son compte.

Pour vérifier la véracité des données rentrées à l'inscription mais aussi à la suppression, il pourrait être intéressant de demander une vérification par mail.

Tel que décrit dans notre procédure en 5 étapes, nous voulions mettre en place un système d'envoi de mail et/ou de SMS permettant de vérifier qu'il n'y a pas de mauvaise correspondance entre notre base de données et le registre des décès du gouvernement. L'idée était donc d'envoyer trois SMS/Mails, à trois jours d'intervalle à chaque fois, à l'utilisateur pour lequel il y a eu correspondance. En cas d'absence de connexion au bout de 10 jours, la procédure pouvait s'enclencher. Cette fonction serait donc nécessaire dans le cadre d'un déploiement pour le grand public.

De plus, une des tâches qu'il serait essentiel de mettre en place pour un déploiement et une utilisation du grand public serait la page "Conditions d'utilisation et Mentions légales". Il existe un renvoi dans le dernier paragraphe de la page "Inscription" vers cette page mais elle n'existe pas encore. Il faudrait se renseigner plus précisément sur l'aspect légal afin de rédiger cette page convenablement mais ce n'était donc pas urgent de le faire dans le cadre de notre soutenance.

Dans les derniers moments de notre projet, nous avons cependant pu nous pencher sur des tâches qui étaient initialement des //TODO, mais que nous avons pu commencer à implémenter. Nous avons alors pu, sur nos machines locales, récupérer les volontés d'un utilisateur indiquées dans l'onglet « Préférences réseaux », en rajoutant des attributs de type boolean dans la classe Volonté. Nous avons également réussi à ressortir ces données dans l'onglet « Informations utilisateurs » sous forme d'une liste par réseau et résumant les volontés de la personne.

Nous avons cependant rencontré un problème lors du déploiement sur Heroku de cette fonctionnalité, et nous avons donc dû les abandonner dans le projet final. Dans cette lignée, il resterait

à proposer à l'utilisateur d'effacer ses volontés grâce à un bouton pour qu'il puisse les modifier à sa guise. Il faudrait aussi pouvoir utiliser ces fonctionnalités sur Heroku sans avoir d'erreurs.

Nous aurions également aimé réussir à poster un message directement sur Twitter, et à envoyer un message privé. Cependant, nous n'avons pas réussi à utiliser ces fonctions convenablement car une autorisation spéciale était requise. Le travail supplémentaire n'aurait pas été titanesque mais aurait demandé de se pencher davantage sur l'utilisation de l'API Twitter.

B. Prolongements

Notre site a été développé pour des utilisateurs nés en France. Changer cela en accédant par exemple à d'autres registres qu'à celui du gouvernement français pourrait nous permettre de développer notre application dans le monde entier.

Pour l'instant, il est possible pour l'utilisateur de désigner 2 proches (via deux adresses e-mail) mais les deux proches ont accès exactement aux mêmes images, vidéos ou messages. Il pourrait être intéressant de développer une fonction permettant à l'utilisateur de choisir le nombre de proches qu'il veut désigner ainsi que les souvenirs qu'il veut transmettre à un proche. Il pourra par exemple envoyer une certaine photo à son fils et transmettre un message à sa femme sans que l'un voie le souvenir que l'autre a reçu.

Lors de l'inscription, il est demandé à l'utilisateur de joindre sa carte d'identité pour vérifier qu'il n'y a pas d'usurpation. Cependant, rien n'est prévu à l'heure actuelle pour comparer les informations sur la carte d'identité et les informations rentrées dans le formulaire.

Une façon d'optimiser notre application pourrait aussi être de mettre en place un téléchargement mensuel automatique du registre des décès du gouvernement car pour l'instant ce dossier est joint à notre application manuellement. Même si changer le lien de référence du scanner une fois par mois n'est pas une tâche très longue, faire cela automatiquement pourrait être très intéressant dans le cadre de notre apprentissage.

En plus des réseaux sociaux que notre application traite actuellement, il faudrait étendre la liste de réseaux proposés et développer les fonctionnalités de chacun afin de permettre à plus d'utilisateurs d'utiliser notre application. Ce prolongement est le genre de tâche qu'il serait intéressant de développer au fur et à mesure, via de multiples mises à jour.

Nous avons également pensé qu'il serait intéressant d'étendre notre travail et de donner la possibilité à l'utilisateur de rendre sa page Legac.e publique à sa mort. Dans ce cas de figure, nous pourrions imaginer que les proches, ou toute personne ayant connu le défunt pourrait à sa guise publier sur le profil des photos, des messages ou des vidéos. Ces fonctionnalités viendraient en plus de la page souvenir, accessible uniquement à la famille.

Enfin, un prolongement intéressant pourrait être de créer une fonction « Mot de Passe Oublié » lors de la connexion.

CONCLUSION

Ce rapport technique représente l'aboutissement de 8 semaines de travail, de recherches et de questionnement. Nous avons beaucoup pivoté depuis notre dossier de veille technologique. La faisabilité, le temps restant et les moyens nécessaires nous ont limités en termes de fonctions mais cela nous a permis d'y voir plus clair dans l'avancement de notre projet.

Ce projet nous a poussé dans nos retranchements, nous sommes sortis de notre zone de confort afin de nous confronter à des situations stressantes, à l'image de ce que notre vie professionnelle nous amènera. Nous avons réussi un vrai travail d'équipe, en conservant une ambiance chaleureuse mais studieuse.

Malgré les difficultés, nous avons toujours pu rebondir et grâce à l'aide de Monsieur Bastide que nous remercions, nous sommes aujourd'hui fier de présenter une application fonctionnelle, dans sa version 1.0.

Les objectifs quant aux fonctions d'inscription, de connexion, de correspondance et de navigation sont remplis. Nous avons également prouvé qu'il était possible d'agir sur le compte Twitter d'un utilisateur avec des actions basiques.

L'idée d'un Testament Numérique est très importante dans le monde dans lequel on vit actuellement et élaborer un projet dans cette thématique était donc très intéressant. Ce projet n'en est qu'à ses balbutiements mais nous espérons qu'un jour, il pourra voir le jour dans une version parfaitement fonctionnelle et optimisée.

ANNEXES

Annexe 1 : Options disponibles pour chaque réseau

Annexe 2 : Diagramme de Gantt de notre projet

Annexe 3 : Modèles de données complet

Annexe 4 : Lettre de demande envoyée concernant l'API Twitter

Annexe 5 : Schéma de navigation entre toutes les pages de notre application

Annexe 6 : Maquettes originelles de notre application