

Rapport de projet de développement WEB

Chléo CHEVRIER, Margaux GONTIE,
Alice JANELA CAMEIJO, Téo MARTIN

21 juin 2020

Table des matières

I	Conception du projet	3
	I.1 Choix du thème	3
	I.2 Organisation générale	5
II	Réalisation	17
	II.1 Manuel technique et utilisation	17
	II.2 Identité graphique du site	20
	II.3 Sécurité du site	27
	II.4 Code JavaScript	30
	II.5 Code de la base de données	36
	II.6 Code PHP	41
III	BILAN	49
	III.1 Bilan du projet	49
	III.2 Bilans personnels	52

INTRODUCTION

Ce rapport est issu de deux mois de travail en groupe dans le cadre de la matière **”Développement WEB”**. Il a pour objectif de retracer la façon dont nous avons tout d'abord choisi notre thème de projet, puis celle dont nous l'avons concrètement réalisé. Il comporte enfin un ensemble de bilans englobant une analyse de ce que nous sommes parvenus à produire ou pas ainsi que les avis personnels de chacun des quatre membres de notre groupe. Le caractère aussi théorique que technique de ce rapport a donc été pour nous l'occasion de retracer l'évolution d'un projet informatique dans son intégralité, depuis sa **conception** jusqu'à sa **concréétisation**.



FIGURE 1 – Logo de notre site WEB. Il s'agit d'un **site de rencontre** comme le proposait l'un des trois sujets auxquels nous avions accès sur AREL à partir du 20/04/2020. Bien que devant obéir au **cahier des charges** qui nous fut transmis au même moment, nous avons été libres dans les choix généraux des cibles du site, dans son architecture globale ou encore dans son identité graphique.

I Conception du projet

I.1 Choix du thème

L'objectif du projet final de développement WEB de deuxième année était de créer un site internet où nous avions le choix entre un site de e-commerce, un site de services et un **site de rencontre**. C'est finalement ce dernier que nous avons choisi. Comme notre groupe était bien inspiré par ce thème, nous avons regroupé toutes nos idées au sein d'un document destiné à notre professeur dès la fin de notre première séance de travail en autonomie.

Dans ce document, nous y retrouvions plus précisément des informations relatives aux personnes ciblées par notre site ou encore certaines de ses particularités. Par exemple, afin de rendre le site plus original, nous avons décidé de **le dédier aux personnes présentant des allergies ou intolérances** et de faire en sorte que les premiers rendez-vous amoureux de ces personnes se passent sans plus aucun mal-entendu ! (Une allergie alimentaire pourrait en effet générer de la gêne au moment de choisir un restaurant ou un café si l'autre n'est pas prévenu.) Plus largement, les utilisateurs du site devront donc entrer des données sur leur(s) allergie(s), leurs défauts, leur pire rendez-vous passé et d'autres informations qui peuvent parfois créer de mauvaises surprises lors d'une rencontre. Au final, **notre site permettra à ses utilisateurs de se rencontrer sereinement et de pouvoir se concentrer uniquement sur les sentiments qu'ils commencent à développer, ou pas, pour la personne en face d'eux !**

Téo MARTIN

Margaux GONTIE

Alice CAMEIJO

Chléo CHEVRIERIDÉE PROJET D'INFORMATIQUE 2020**Nature** : Site de rencontre.**Thème général** : Rencontres entre allergiques.**Nom** : ALOVEGIES !**Slogan** : Plus de mauvaises surprises au premier *date* !**bis** : Fini les ambulances, bonjour la romance !**bis bis** : Fini les allergies, finissez enfin avec l'amour de votre vie !**But** : L'utilisateur renseigne ses allergies de sorte à pouvoir aller à un rendez-vous tranquillement, sans se soucier de la nature du restaurant où il ira manger, etc... Il pourra ainsi ne pas paraître ennuyeux auprès de son *date*, et maximiser ses chances d'aboutir à une relation !**Pourquoi ce thème** : C'est un thème assez large, qui permettra de laisser parler notre **créativité**. Nous pensons qu'il est suffisamment spécifique pour s'adresser à une partie restreinte de la population, sans pour autant s'adresser à une minorité trop faible.**Quelques idées concrètes pour le site :**

- Un curseur pour situer la gravité des allergies de chacun (allant de 1 à 5);
- Profil en partie "négatif" : au lieu de rentrer uniquement ce que l'on aime, ses qualités, etc ... les catégories à remplir pourront être par exemple défauts, allergies, choses que l'on détestent, pire date, phobies. **Ainsi, chacun pourra se présenter au naturel, sans avoir à omettre certains détails;**
- Au lieu d'avoir un like en forme de cœur pour récompenser les profils que l'on aime, on pourrait offrir une "ventoline" aux profils allergiques au pollen, etc... (pour les utilisateurs abonnés)
- Toujours pour les abonnés : 3 discussion simultanées pour les niveau 1, 5 pour les niveau 2, mise en avant sur les pages d'accueil et de recherche

FIGURE 2 – Image du document que nous avons fourni au tout début de notre projet. Il contient une première description de sa nature ainsi qu'un ensemble d'idées à son sujet. Nous reviendrons plus tard sur elles et verront si nous sommes effectivement parvenus à toutes les réaliser !

I.2 Organisation générale

1. 2. a) Idée d'architecture du site de rencontre

Première maquette du site Après une lecture attentive du **cahier des charges**, nous nous sommes rendus compte du nombre important de pages et de fonctionnalités qu'il allait falloir créer pour notre site. Afin d'y voir plus clair, nous avons rédigé un document où nous avons listé l'ensemble des **pages** à construire ainsi que les **fonctionnalités** qui leur seront propres. Par ailleurs, comme le cahier des charges indiquait qu'il existerait **plusieurs natures de personnes** pouvant accéder au site (allant du **simple visiteur**, aux **utilisateurs inscrits**, aux **utilisateurs abonnés de niveau 1 ou 2** jusqu'aux **administrateurs**), nous avons également dû y récapituler les droits de chacun d'entre eux pour chacune des fonctionnalités de chacune des pages de notre site ! Par exemple, si l'utilisateur s'est connecté à son compte, il verra apparaître son profil sur la page d'accueil, sinon il n'apparaîtra pas.

Idée d'architecture du site

1) **Personne non inscrite :**

page connexion + page inscription + voir des profils (page recherche) +
page abonnement

2) **Personne inscrite :**

Personne non inscrite + gérer son profil (page Mon profil + page Modifier Mon Profil) + rechercher des profils (page recherche) + page Profil Autre Utilisateur

3) **Personne abonnée :**

Personne inscrite + envoyer des messages (page Messagerie) + bloquée / signaler (page Profil Autre personne)

4) **Administrateur :**

Personne abonnée + accéder à la liste et aux profils/messages de tout le monde (modifier/supprimer) + reçoit les signalements

Page d'Accueil :Titre et sous titre du site et logo + texte explicatif du site,.....

Bandeau en haut avec des onglets pour naviguer sur le site : S'inscrire, Connexion, Voir des profils, Abonnement, Mon profil, Messagerie

Quelques photos des profils inscrits (impossible de cliquer sur le pseudo si non inscrit)

Un coin ou on voit notre profil (et ou on peut le modifier)

Page Connexion (pour entrer son pseudo, mdp, mdp oublié,.....)

Bouton "Tu n'es pas encore inscrit ?" avec lien vers la page inscription

FIGURE 3 – Image du document que nous avons fourni au tout début de notre projet. Il contient une première description de sa nature ainsi qu'un ensemble d'idées à son sujet. Nous reviendrons plus tard sur elles et verront si nous sommes effectivement parvenus à toutes les réaliser !

Notre site se composera donc au minimum des pages suivantes associées à quelques premières idées de fonctionnalités :

- Une **page d'accueil** : avec un petit texte présentant le site ainsi que 3 cartes de profils d'autres utilisateurs inscrits aléatoirement choisies afin de donner envie à un visiteur de se connecter ;
- Une **page de connexion** : avec un formulaire permettant de saisir son pseudonyme et son mot de passe ;
- Une **page inscription** : avec un formulaire qui demande les informations non modifiables et obligatoires du futur utilisateur ;
- Une **page d'abonnement** : qui permet de choisir et de payer un niveau d'abonnement / de résigner un abonnement déjà en cours ;
- Une **page de recherche de profils** : qui propose par défaut 12 cartes de profils d'autres utilisateurs choisis aléatoirement parmi les abonnés de niveau 2 puis de niveau 1 et 0 ;
- Une **page Mon Profil** : qui affiche notre profil tel qu'il apparaît pour les autres utilisateurs ainsi qu'un bouton menant à la page **Modifier Mon Profil** qui permet de changer ses informations et d'accéder à ses données privées et un bouton "SupprimerMonProfil" ;
- Une **page Profil Autre Utilisateur** : similaire à la page MonProfil, elle affiche le profil d'un autre utilisateur du site lorsque l'on a cliqué sur son pseudonyme ;
- Une **page messagerie** : qui permet d'échanger avec les autres utilisateurs lorsque l'on est abonné ;
- Une **page déconnexion** : qui confirme à l'utilisateur qu'il s'est bien déconnecté du site.

Au final, on peut résumer la première idée d'architecture de notre site dans le tableau suivant :

Pages /type d'utilisateur	Visiteur	Inscrit	Abonné
Connexion	X		
Inscription	X		
Accueil	3 profils abonnés aléatoire	3 profils abonnés aléatoires + mon profil	X X
Recherche	12 profils aléatoires +filtres	12 profils aléatoires qui correspondes à mon orientation	
Page autre utilisateur		X	X
MonProfil		X	X
ModifierMonProfil		X	X
Page autre utilisateur		X	X
Déconnexion		X	X
Abonnement		X	X
Messagerie		X	X

Chaque page dispose, en haut, d'un bandeau avec le nom, le logo et le slogan du site ainsi que d'une barre d'onglets qui évolue en fonction de la page sur laquelle on se trouve et de notre niveau d'abonnement.

type d'utilisateur/ Pages	Inscrit	Abonné niv 1	Abonné niv 2
Accueil	3 profils abonnés aléatoires mon profil	X	X
Recherche	12 profils aléatoires selon mon orientation	12 profils aléatoires selon mon orientation + filtres : age, lieu, barre de recherche	12 profils aléatoires selon mon orientation + filtres : age, lieu, barre de recherche + filtre allergie
Page autre utilisateur		+ bouton « like » + bloquer/signaler + message (1 pers)	+ bouton like (en forme de ventoline) + messages (5 pers) + bloquer/signaler + bouton « like »

Première maquette de la base de données Une fois que nous avions une première idée de l'architecture de notre site, il nous fallait réfléchir à **l'organisation de sa base de données**. Comme vous pouvez le voir sur la photo suivante, nous avons tout d'abord commencé par imaginer sur papier comment elle serait globalement organisée ainsi que la répartition des différentes catégories qu'elles contiendrait :

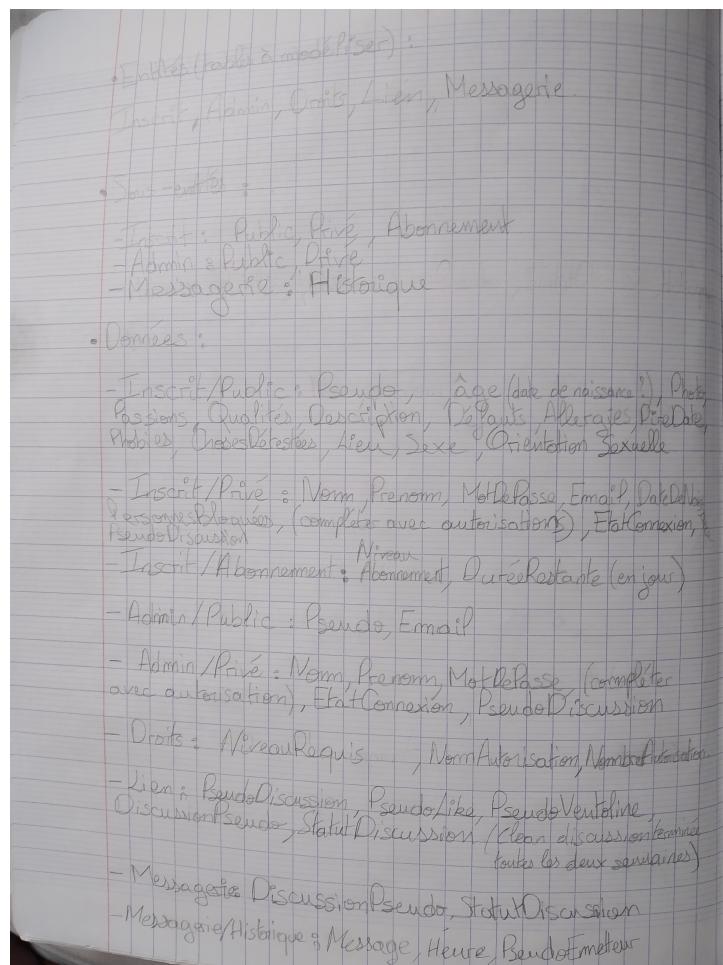


FIGURE 4

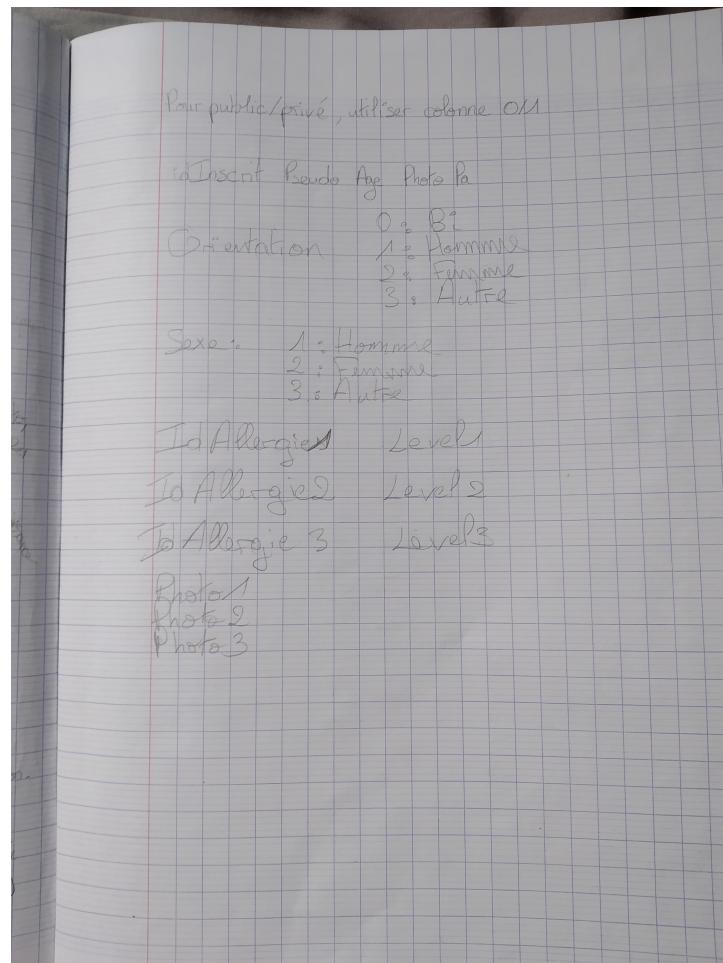


FIGURE 5

Expliqué de façon plus précise, nous avons **débuté par une réflexion sur l'ensemble des données que nous allions devoir stocker**. Au départ, nous avions pensé aux catégories suivantes : Pseudo, Lieu, Sexe, Orientation sexuelle, Nom, Prénom, Mot de passe, Email, Date de naissance, Allergies, 3 photos, Description, Passions, Qualités, Défauts, Pire date, Phobies, Choses détestées, Personnes bloquées, Etat de la Connexion, Pseudo de Discussion en cours.

Tous ces éléments concernent les données en rapport avec **l'utilisateur du site** lui-même. Mais nous avions également pensé que nous aurions besoin de stocker des informations liées à son **fonctionnement**, telles le nom de certaines fonctionnalités (= avoir des discussion avec d'autres membres), le niveau requis et nombre d'autorisations (= abonné de niveau 1 ou 2 et donc 3 ou 5 discussions simultanées autorisées). Autrement dit, nous avions à ce stade là l'idée de donner un nom à chaque manipulation possible et de l'associer à un niveau d'utilisateur, le niveau le plus bas étant celui du simple visiteur qui n'est pas connecté jusqu'à l'administrateur qui a tous les droits. Le niveau requis était le niveau nécessaire pour accéder à la commande, et chaque personne présente sur le site voyait son niveau comparé au niveau requis afin de voir s'il pouvait ou non. Enfin, la dernière ligne du tableau de la base de données aurait servi dans les cas où nous devions limiter l'usage d'une fonction. Par exemple, le nombre de discussions possibles est limité en fonction du niveau que l'on a. Ainsi, si un niveau autorise 3 discussions à la fois, il faut que nous puissions connaître à combien de discussion prend déjà part l'utilisateur dont il est question.

Enfin, après avoir réfléchi à la vue d'ensemble de toutes les données que nous devrions stocker dans notre base de données, nous avons pu réfléchir à la façon dont nous allions pouvoir les regrouper entre elles. Ainsi, nous avons imaginé **l'architecture globale** suivante : une table regroupant simplement les catégories évoquées précédemment, et une autre table par élément, nous permettant de rajouter une ligne pour savoir si l'information qu'il contient était privée ou publique (et donc à qui elle était accessible). Malgré la lourdeur de cette option, cette dernière nous apparaissaient comme la plus efficace, la plus organisée et la moins chaotique. **Cependant, au fur et à mesure que le projet a avancé, nous avons changé notre point de vue et avons trouvé des alternatives, comme vous le verrez dans la section consacrée à la base de données.**

Ainsi, une fois nos premières idées d'architecture du site exposées, nous pouvons passer à la façon dont nous nous sommes concrètement organisés afin de la réaliser. Cette organisation peut se découper en différentes étapes, dont la première a été le partage des tâches entre chaque membre du groupe.

1. 2. b) Répartition des tâches

Pour la répartition des tâches, nous avons d'abord, décidé de nous **répartir le travail en fonction des nombreux langages de programmation** que nous allions devoir utiliser. En effet, nous avons pensé que cette répartition serait optimale dans le sens où elle nous permettrait de travailler en parallèle sur diverses parties du projet et donc d'en avoir une vue d'ensemble. Par ailleurs, il est plus simple pour un élève d'être vraiment performant s'il n'a à se concentrer que sur un ensemble de langages précis. Au niveau du choix de ces langages, il s'est fait en fonction des envies et des compétences de chacun des membres de notre groupes. Ainsi, **le langage SQL a été confié à Téo** qui s'est montré particulièrement intéressé par les bases de données, **le PHP à Margaux**, **le JavaScript et le Ajax à Chléo** et enfin **le HTML et le CSS à Alice**.

Cependant, nous avons constaté qu'au fur et à mesure que le programme avançait, certaines personnes sont venues aider dans d'autres langages. Notamment Alice qui, une fois la première version du CSS finie, est allée aider Margaux avec le PHP puis a aidé Téo à remplir la base de données, avant de revenir travailler sur son langage de prédilection pour réaliser le second design de notre site. On peut encore citer le cas de Téo qui, une fois l'architecture de la BDD terminée et codée, a commencé à réfléchir à notre page Messagerie avant d'être rejoint par le reste du groupe.

Pour finir, nous nous sommes répartis les tâches au niveau de la rédaction de ce présent rapport. Cette répartition a eu lieu lors d'une réunion de notre groupe environ trois semaines avant la date du rendu du projet et avait pour objet la confection du plan de notre rapport et donc la répartition de chacune de ses parties. Voici le document que nous avons produit ce jour là :

INTRODUCTION**II/ CONCEPTION DU PROJET**

- 1) Choix du thème
- 2) Organisation
 - a. Idées de conception / d'architecture du site
 - i. architecture du site
 - ii. base de données
 - b. répartition des tâches
 - c. planning et suivi du projet

II/ RÉALISATION

- 1) Manuel technique et utilisation
- 2) Identité graphique
- 3) Sécurité du site
- 4) Code JavaScript
- 5) Code des bases de données
- 6) Code PHP - SQL
- 7) Code PHP - CSS

III/ BILAN

- 1) Bilan du projet
- 2) Bilans personnels
Chléo
Margaux
Alice
Téo

CONCLUSION**BIBLIOGRAPHIE**

FIGURE 6 – Plan du présent rapport avec la répartition de chacune de ses parties en fonctions des membres du groupe. Le jaune représente Alice, le vert Maragux, le violet Chléo et le bleu Téo.

1. 2. c) Planning et suivi de l'avancement du projet

Pour finir cette partie liée à l'organisation de notre projet, nous avons pensé qu'il pouvait être intéressant de créer un **planning** des tâches que nous prévoyons de réaliser tout le long du projet. Le premier planning que nous avons imaginé s'intitule "**Liste des choses à faire**" et était **séparé selon les différents langages de programmation** que nous utilisions. Il était mis à jour très régulièrement par tous les membres de notre équipe afin de ne pas oublier les nouvelles idées qui nous venaient au fur et à mesure de la confection de notre site. En voici une image :

Liste des choses à faire

HTML :

- **FAIRE LE NOUVEAU CSS !**
- **Créer** les pages [Abonnement.php](#), [Messagerie.php](#)
- **Créer** les **différents affichages** des différentes options /* en commentaires */
- ne pas pouvoir changer le mail dans [MofifierMonProfil.php](#) + rajouter la fonction margaux sur les allergies
- Mettre un bouton retour vers le profil dans [MofifierMonProfil.php](#)

JAVASCRIPT : OK dans [MonProfil](#) et [Recherche](#)

- **fonction** qui affiche le **block de recherche** (si abo) ou centre le **bloc de recherche**
- **function** : faire un bouton qui affiche le mot de passe dans la partie privée de [MofifierMonProfil.php](#)
- gérer le nombre d'allergies dans accueil et mon profil -----> En cour
- refaire le alert de mdp dans [MofifierMonProfil.php](#)
- **Sécurité :**
onload dans chose à faire au chargement de la page -> regarder si la personne est autorisé à voir la page sinon rediriger vers la page d'accueil fait pour :
 - [MonProfil](#)
 - Accueil PAU
 - [ModifierMonProfil](#)
 - [PageAutreUtilisatuer](#)
 - **Abonnement**

FIGURE 7 – Premier planning que nous avons imaginé : la "Liste des choses à faire".

Comme indiqué précédemment, ce document était très pratique car il

nous permettait de bien voir ce qu'il nous restait à faire et de pouvoir s'organiser en conséquence. Ainsi, nous avions pris l'habitude de nous référer à lui à chaque début de séance ainsi que d'ôter les tâches que nous avions pu réaliser à la fin de notre travail.

Néanmoins, plus le projet avançait, et plus l'ensemble des pages que nous avions prévu de créer l'était effectivement, plus il nous est apparu qu'il devenait davantage intéressant pour nous de faire un planning des choses à rajouter par page et non plus par langage de programmation. C'est pourquoi, environ un mois avant le rendu final de notre projet, nous avons créer le nouveau document "**Liste des choses à faire PAGE**" que voici :

Liste des choses à faire PAGE

**PARTOUT : SECURITE + CO/ABO + PAGES ADMIN
GESTION NB CONVERSATIONS**

VÉRIFIER TOUS LES ONGLETS ET BLOCS JUSTE AVANT DE RENDRE

Commentaires dans le code
pb accents

Accueil :

- **JS** : bien placer les carrés
- **CSS** : "connexion établie" **tres moche !**
- (connexion : mot de passe oublié)
- écrire le vrai "A propos"
- **Connexion** : re pop-up après la co ?
- **JS** : **pb** affiche profil alors que pas co !
- Remplacer SESSION Pseudo

MonProfil :

- **JS** : bien placer les carrés
- Supprimer Mon profil

Recherche :

- **CSS** : bon placement de la barre de recherche + "aucun profil ne correspond à votre recherche"
- Mettre le **CSS** de Messagerie car page appelée en AJAX
- **PHP** : décalage des Si (récup mauvaise valeur quand on clique sur pseudo)(verif tous les cas)
- vérifier "Message" avec nv abo dans tous les cas possibles
- (barre de recherche avancée)

FIGURE 8 – Second planning que nous avons imaginé : la "Liste des choses à faire par PAGE".

Par ailleurs, au niveau de nos **horaires de travail**, nous nous sommes rapidement rendu compte que les quatre heures initialement dédiées au Développement WEB ne seraient pas suffisante pour la confection de notre projet. Ainsi, nous avons réorganiser notre semaine de la façon suivante :

- les mardi soirs et mercredi soirs étaient dédiés au codage du projet en lui-même ;
- les jeudi après-midi permettaient à la fois de continuer le codage du site et de faire un point sur l'avancement du projet avec toute l'équipe (c'est souvent à ces moments là que nous mettions en lien les parties de codes PHP, JS et HTML,...) ;
- les weekends étaient laissés intacts mais chacun était libre de travailler de son côté.
- La dernière semaine de travail était particulièrement intense. Nous nous sommes vu tous les matins, et nous avons travailler sur le fignolage du code et sur la rédaction du rapport au moins 3h par jour.

Enfin, en ce qui concerne la **rédaction du rapport final**, nous n'avions pas d'horaires précis qui lui étaient dédiée. Chacun avançait sa partie dans le semaine, la seule obligation était de rendre son travail les samedis à Alice pour qu'elle puisse le relire ainsi que mettre en forme le rapport.

II Réalisation

II.1 Manuel technique et utilisation

Dans cette partie, il s'agira de **rassembler l'ensemble des fonctionnalités** présentes sur chacune des pages de notre site et donc de comprendre en quoi celles-ci sont liées les unes aux autres dans le but de permettre à un utilisateur quelconque du site de pouvoir y naviguer sans se poser de questions. Autrement dit, il s'agira de présenter la **véritable architecture** de notre projet, qui ressemblera évidemment à celle que nous avons imaginée au sein de sa partie de conception bien que certains **détails** initialement non prévus auront pu être rajoutés et inversement.

Notons simplement que nous ne préciseront pas à chaque fois la présence d'**une en-tête, des onglets ainsi que d'un pied de page** au sein de nos pages car ce sont des éléments que l'on retrouvent dans chacune d'entre elles, sans exception !

Page d'inscription : InscriptionW3.php Elle contient :

- **Un formulaire PHP** permettant à un visiteur de se créer un compte et possédant un bouton de validation le menant ensuite à la page [ModifierMonProfilW3.php](#).

Page d'accueil : AccueilW3.php Elle contient :

- **Un pop-up** permettant à un simple visiteur de **se connecter** au site dès son arrivée sur la page d'accueil.
- **Une zone de texte** contenant le témoignage d'un utilisateur du site et destiné à encourager de simples visiteurs à s'y inscrire (voir à s'y abonner).
- **3 cartes de mini-profil**s personnes inscrites choisies au hasard et destinées à donner envie d'en recherche d'autres en cliquant sur l'onglet "Autres profils". A savoir que le bouton "Envoyer un message" situé en bas de chacune de ces cartes ne s'affichera que si l'utilisateur du site est connecté et abonné.

Elles mènent aux pages [ProfilAutreUtilisateurW3.php](#) et [MessengerieW3.php](#)

- **Le profil de l'utilisateur** du site s'il y est connecté.

Page d'abonnement : AbonnementW3.php Elle contient :

- **2 mini-cartes** de présentation des niveaux d'abonnements disponibles sur notre site. Elle sont disponibles à n'importe quel utilisateur du site, même aux visiteurs.
Elles mènent à [PagePayementW3.php](#)
- **1 mini-carte** permettant à un utilisateur abonné du site de résilier son engagement.
Elle mène à [RetourAccueilW3.php](#)

Page de paiement : PagePayementW3.php Elle contient :

- **Un formulaire PHP** permettant à un utilisateur du site de payer et de valider son choix de niveau d'abonnement.
Elle mène à [RetourAccueilW3.php](#)

Page de validation/résiliation d'un abonnement : RetourAccueilW3.php

- **Une zone de texte** adapté à la situation et dont l'un des mots possède un lien menant à la page [AccueilW3.php](#).

Page du profil de l'utilisateur connecté : MonProfilW3.php

- **Un tableau** à trois colonnes et sans bordure **du profil de l'utilisateur** connecté au site. On y retrouve l'ensemble de ses données publiques.
- **Un bouton** "paramètres" menant à la page [ModifierMonProfilW3.php](#).
- **Un bouton de suppression** de son profil
Il mène à la page [DeconnexionW3.php](#)

Page de modification du profil connecté : ModifierMonProfilW3.php

- **Une petite zone de texte privée** contenant les informations personnelles de l'utilisateur connecté (nom, prénom, mail et mot de passe). Elles ne sont visibles que par lui ainsi que par les administrateur du site.
- **Un formulaire PHP** regroupant l'ensemble des informations que l'utilisateur a rentrées au moment de son inscription + d'autres informations qu'il devra remplir avant de pouvoir rejoindre son profil.
Il mène vers [MonProfilW3.php](#)

Page de recherche d'autres utilisateurs : RechercheW3.php

- **12 cartes de mini-profil**s d'autres personnes inscrites choisies au hasard si l'utilisateur du site n'est qu'un visiteur / en fonction du sexe et de l'orientation de l'utilisateur du site s'il y est inscrit et connecté / en fonction du sexe, de l'orientation et des filtres de recherches décidés par l'utilisateur s'il est inscrit, connecté et abonné au site. A savoir que le bouton "Envoyer un message" situé en bas de chacune de ces mini-cartes ne s'affichera que si l'utilisateur du site est connecté. Elles mènent aux pages [ProfilAutreUtilisateurW3.php](#) et [MessagerieW3.php](#)
- **Un bouton permettant d'accéder aux filtres et à la barre de recherche de profils** d'autres personnes inscrites si l'utilisateur est abonné au site.

Page du profil d'autres utilisateurs : ProfilAutreUtilisateurW3.php

- **Un tableau à trois colonnes et sans bordures du profil de l'autre utilisateur** que la personne connectée au site à décider de consulter. On y retrouve l'ensemble de ses données publiques.
- **Un bouton "Like"** permettant à l'utilisateur connecté d'indiquer qu'il aime particulièrement le profil qu'il consulte. A noter qu'il n'est possible de liker plus d'un profil par utilisateur.
- **Un bouton message** permettant à l'utilisateur connecté d'entamer une discussion avec le profil qu'il consulte, à condition qu'ils soient abonnés au site

Il mène à la page [MessagerieW3.php](#)

Page de messagerie du profil connecté : MessagerieW3.php

- **Un tableau de sélection** contenant un aperçu de chaque profil avec lequel l'utilisateur connecté et abonné au site a engagé une discussion.
- **Une zone de d'envoi de messages** contenant elle-même un formulaire PHP et permettant à l'utilisateur du site de discuter avec le profil qu'il aura choisi parmi son tableau de sélection (en cliquant sur son pseudonyme).

Page de déconnexion : DeconnexionW3.php Elle contient :

- **Une zone de texte** dont l'un des mots possède un lien menant à la page [AccueilW3.php](#).

Pages utiles à l'administration du site :

administrateur : PageAdminW3.php Elle contient :

- **Une liste de tous les utilisateurs** avec, sur chaque ligne, un bouton permettant d'accéder à leur profil ([ProfilAutreUtilisateurW3.php](#)), de modifier leur profil ([ModifierProfilGeneral.php](#)), d'accéder à leur messagerie ([MessagerieGenerale.php](#)), de supprimer leur profil et de voir les blocages et les signalements.

Page de modification d'un autre profil : ModifierProfilGeneral.php

Elle contient :

- **Un formulaire** permettant à l'administrateur de modifier toutes les informations d'un autre profil à l'exception du pseudo, de la date de naissance, du mot de passe et du sexe.

Page de messagerie générale : MessagerieGenerale.php Elle contient :

- **La messagerie d'un autre utilisateur.** L'administrateur peut consulter les messages envoyés et reçus par un autre utilisateur et décider de supprimer une conversation. En cliquant sur un pseudo il peut aussi accéder au profil de l'utilisateur : ([ProfilAutreUtilisateurW3.php](#)).

II.2 Identité graphique du site

L'identité graphique de notre site a été définie en deux temps. En effet, comme il nous a tout d'abord semblé nécessaire de créer un premier jet de **code CSS** pour chacune de nos pages HTML avant de pouvoir commencer à coder la suite de notre site, le style graphique de ces premières pages n'était pas extrêmement soigné ; juste suffisamment claire afin de pouvoir être utilisable par la suite.

Premier jet

Comme indiqué précédemment, le code CSS des premières pages de notre site était relativement **simple** et ne faisait appel à aucune librairie de style externe comme celles de W3SCHOOL. Il utilisait uniquement les connaissances que nous avions retenues des cours de développement WEB du début du semestre et **manquait donc de modernité**.

Nous ne détaillerons cependant pas ce code CSS étant donné qu'il n'est pas celui que nous avons retenu pour notre projet final. En voici juste un exemple appliqué à la toute première page que nous avons codée : la page d'accueil.

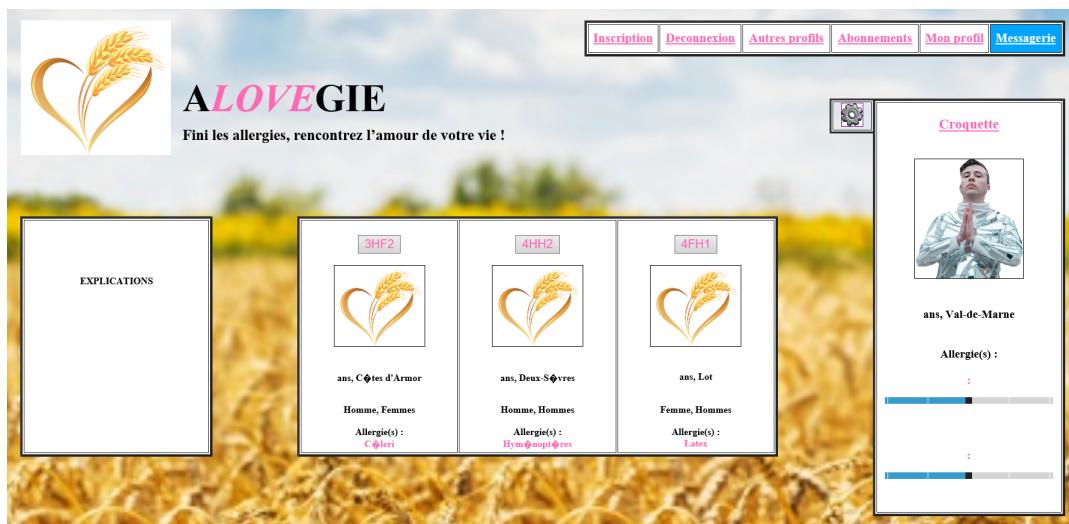


FIGURE 9 – Photo de l'ancienne page d'accueil de notre site.

L'image de fond qui a été choisie est celle d'un champ de blé, en référence à l'un des membres de notre groupe allergique au gluten !

D'un point de vue stylistique, chacune des zones que cette page contient (onglets, profil de l'utilisateur connecté, profils de présentation,...) étaient des **tables HTML** possédant une **bordure double épaisse de 4 pixels**. Les couleurs principales du site étaient le dorée (gold), le rose flash (hotpink) et le bleu (blue).

A noter qu'à cette époque, nous n'avions pas encore écrit tout le code PHP de notre projet, ni confectionné notre base de données finale d'où le manque de finitions de cette page.

Second jet

Arrivé au milieu du temps qui nous était imparti pour réaliser notre projet, nous avons finalement décidé de modifier le style graphique de notre site. **L'objectif était de le rendre plus actuel**, que ce soit au niveau de ses couleurs, de certains de ses détails ou encore de certaines de ses animations CSS.

Les **couleurs** qui ont majoritairement été utilisées dans cette seconde version sont le rose pastel (de code `#FFE6E6`), le gris clair (de code `#f1f1f1`) et le rouge poudré (de code `#E6362E`). **Elles ont été choisie suite au visionnement de plusieurs vidéos parlant de graphisme** expliquant que les couleurs actuellement à la mode étaient les couleurs claires / les couleurs dans les teintes saumon-rose-orangé.

En terme de **modernité**, nous avons fait en sorte de **ne plus afficher les bordures de certains tableaux**, comme par exemple ceux d'affichage des profils au sein des pages **MonProfilW3.php** ou **ProfilAutreUtilisateurW3.php**. Cela enlève en effet le côté "trop brut" que l'on pouvait reprocher aux anciens tableaux de nos premières pages du site. Par ailleurs, nous avons aussi décidé d'**utiliser des bibliothèques de style CSS externes** comme celles de W3SCHOOL. Celles-ci nous ont ainsi permis de créer les petites cartes d'affichage des mini-profil sur les pages **AccueilW3.php** et **RechercheW3.php**. Ces cartes sont plus épurées et plus "professionnelles" que celles de la version précédente, dans le sens où elles sont, à l'origine, des templates de cartes standard de visites d'administrateurs de site internet. Voici donc un exemple de l'une de ces cartes de mini-profil, munie de son code CSS :

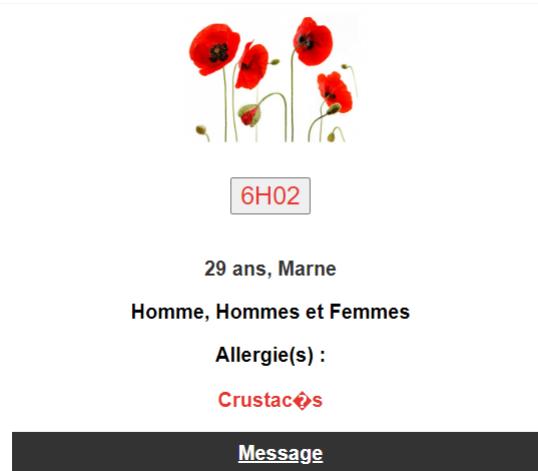


FIGURE 10 – On trouve ces petites cartes présentatrices des profils des autres utilisateurs du site au sein des pages **AccueilW3.php** et **RechercheW3.php**.

```

1 <style> /* zone profils autres utilisateurs */
2 .column {
3   float: left;
4   width: 30.3%;
5   margin-bottom: 16px;
6   padding: 0 8px;
7 }
8 .card {
9   box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);
10  margin: 8px;
11 }
12 .about-section {
13   padding: 50px;
14   text-align: center;
15   background-color: #474e5d;
16   color: white;
17 }
18 .container {
19   padding: 0 16px;
20 }
```

```
21 | .title {  
22 |   color: #333;  
23 | }  
24 | .PseudonymesAutresUtilisateurs{  
25 |   text-align: center;  
26 |   color : #E6362E;  
27 |   background-color : #f1f1f1 ;  
28 |   font-size:20px;  
29 | }  
30 | .button {  
31 |   border: none;  
32 |   outline: 0;  
33 |   display: inline-block ;  
34 |   padding: 8px;  
35 |   color: white ;  
36 |   background-color: #333;  
37 |   text-align: center ;  
38 |   cursor: pointer ;  
39 |   width: 100%;  
40 | }  
41 | </style>
```

Code trouvable au sein de la page **AccueilW3.php**

Une autre mise en page intéressante que nous a permis de réaliser les bibliothèques de style CSS et les exemples de templates de code W3SCHOOL a été une **structure en "trois colonnes juxtaposées"** (un peu à la manière d'un tableau mais sans aucune bordure et adapté à la largeur de notre page,...). On peut notamment observer cela au sein de la page **Message-rieW3.php** où il a suffit d'écrire les lignes suivante au sein du *<style>* de la page :

```
1 /* Creation de colonnes positionnees
2 l 'une a cote de l autre : */
3 .column {
4     float: left ;
5     width: 50% ;
6     margin-top: 6px ;
7     padding: 20px ;
8 }
9 .row: after {
10    content: " " ;
11    display: table ;
12    clear: both ;
13 }
```

Code CSS trouvable au sein de la page **MessagerieW3.php**

Au niveau des **animations** apportées à notre site, nous pouvons observer la plus intéressante d'entre elles au moment de la connexion d'un utilisateur à son compte. En effet, nous avons fait en sorte que, contrairement à tous les autres onglets de notre site, lorsqu'un visiteur clique sur le bouton "Connexion", il n'est pas directement redirigé vers une nouvelle page HTML mais a plutôt la possibilité de se connecter au sein d'un **pop-up JavaScript** apparaissant sur son écran. Or cette animation possède de nombreuses qualités. Tout d'abord, elle **apporte de l'originalité** au site, qui semble tout de suite **plus vivant**. Par ailleurs, elle est très **fluide** dans le sens où elle ne nécessite pas de temps de chargement. Enfin, elle apporte beaucoup de **modernité** au style graphique de la page de par son cadre d'opacité très faible autour d'elle (ce qui donne un effet de "fondu" avec l'arrière plan) et de part sa capacité à recouvrir la page dont elle est issue sans pour autant totalement la cacher. En voici une image :

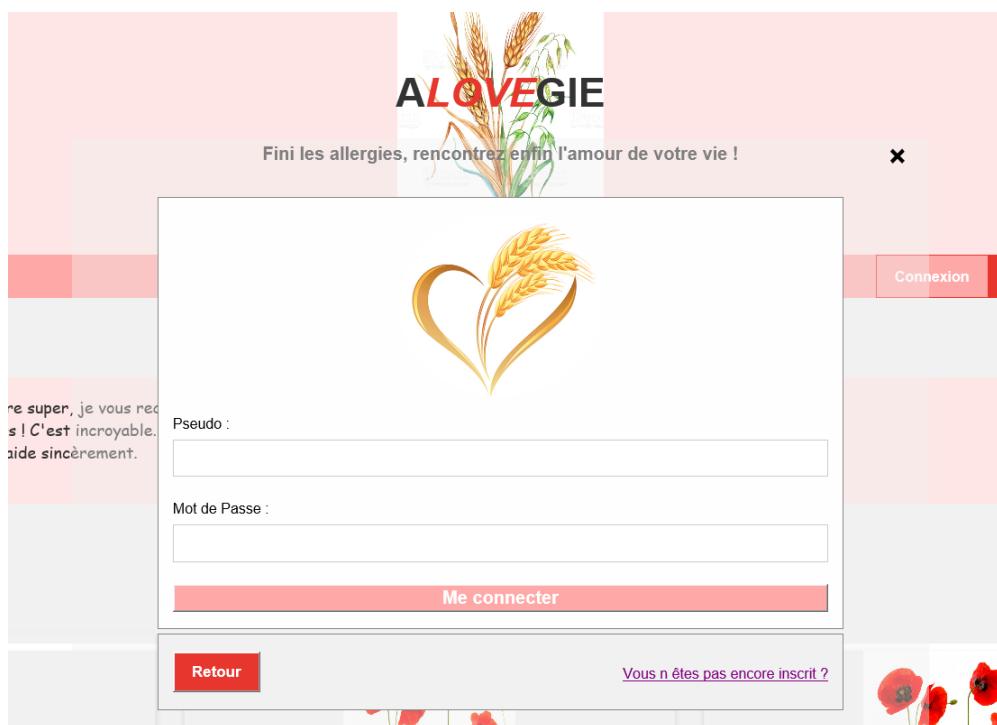


FIGURE 11 – Pop-up de connexion au site de rencontre. Il apparaît lorsqu'un visiteur clique sur l'onglet "Connexion" ou encore la première fois qu'il arrive sur la page d'Accueil.

Il se fond sur l'arrière plan grâce à un contour gris clair translucide (=attribut CSS : "background-color : rgba(241, 241, 241, 0.4);") et il est **rétractable** grâce à la petite croix située en haut à droite de son cadre.

Nous avons estimé que le formulaire PHP qu'il contenait serait plus moderne encore si ses zones de saisies représentaient la totalité de la largeur du pop-up (= attribut CSS "width : 100 ;").

Une seconde animation que nous pouvons rapidement évoquer est celle que l'on retrouve dans la page **RechercheW3.php**. Elle consiste en un bouton sur lequel seul un utilisateur abonné au site peut cliquer et qui permet de faire **glisser un panneau vers la gauche**. Ce panneau contient alors le formulaire PHP qui constitue les filtres ainsi que la barre de recherche de profils du site. Là encore, nous pouvons dire que ce procédé permet de **dynamiser la page** en question tout **en ne la surchargeant pas** d'informations

(étant donné que si l'abonné désire ne pas utiliser les filtres de recherche dont il dispose pourtant, alors ceux-ci ne seront pas visibles sur la page).

Enfin, nous avons essayé de créer un code CSS permettant de donner de la **cohérence / crédibilité** à notre projet en tant que site internet. Ainsi, nous avons ajouté à chacune de ces pages un **pied de page** contenant les noms, la classe et le groupe des créateurs / administrateurs du site (en l'occurrence, nous!). Ci-dessous un dernier exemple de code CSS permettant de comprendre comment nous avons pu coder ces pieds de page :

```
1 /* Footer */
2 .footer {
3     position: fixed;
4     left: 0;
5     bottom: 0;
6     width: 100%;
7     background-color: #E6362E;
8     color: white;
9     text-align: center;
10}
```

Code CSS trouvable au sein toutes les pages du site.

Au final, nous avons essayé de donner à notre site un code CSS lui permettant d'acquérir une **véritable identité graphique** - à travers ses couleurs - **du dynamisme** - à travers ses animations - ou encore une **de la cohérence en tant que site internet** - avec ses en-têtes et ses pieds de page -. Les bibliothèques de style W3SCHOOL lui ont enfin apporté de la **modernité**.

II.3 Sécurité du site

La sécurité de notre site a été réalisée en **JavaScript**. Son objectif majeur est de s'assurer qu'un utilisateur non inscrit (ou non abonné) au site ne puisse pas, par quelques moyens que ce soit, accéder à une page pour laquelle il n'a pas l'autorisation de se rendre. De même, il faut s'assurer que les priviléges réservés aux personnes abonnées ne soient accessibles qu'à elles ect...

Le principe du code de sécurité des pages du site en lui-même est très simple : au chargement d'une page donnée (= **onload** en JS), une fonction est appelée. Dans celle-ci, nous vérifions si l'utilisateur a le droit d'avoir accès à la page grâce à la valeur de ses paramètres d'entrées que sont les fonctions **VerifCo()** et **VerifAbo()** capable de prendre respectivement les valeurs 0 (= non connecté) ou 1 (= connecté) et 0 (=non abonné), 1 (= abonné niveau 1) ou 2 (= abonné niveau 2).

Dans le cas où l'utilisateur a le droit de se rendre sur la page en question, alors rien ne se passe et la page se charge comme prévu. dans le cas contraire, alors une **alerte JS** s'affiche avec un message indiquant à l'utilisateur que les droit d'accès lui sont refusé et que ce dernier sera donc redirigé vers la page d'Accueil du site.

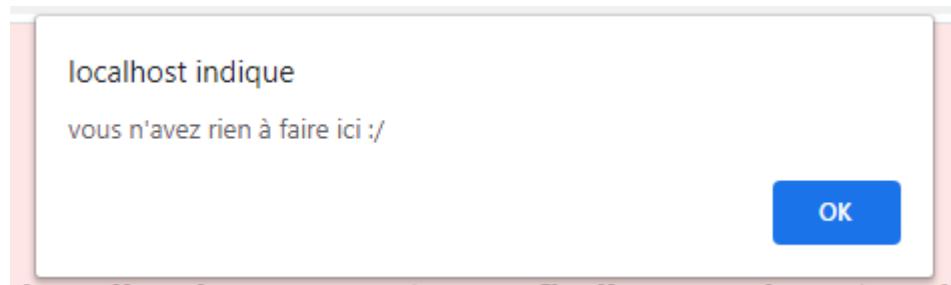


FIGURE 12 – Pop-up de sécurité

```

1 function choseAFaireAuChangementDeLaPage (connection ,
2 abonnement , nbAllergie , administration ){
3
4 if ( connection==0){
5     alert ("vous n'avez rien à faire ici :/");
6     window.location.href="AccueilW3.php";
7 }
8 [...]
9 }
```

Ci-dessus on retrouve une partie du **script** de la page **MonProfil.php**. Ce script est le même pour toutes les autres pages du site avec, bien évidemment, les conditions précédemment évoquée en plus.

Les seules pages à faire exception de ce système sont les **pages chargées en AJAX** depuis d'autres pages du site, comme par exemple **ProfilAutreUtilisateurW3.php** ou encore **PagePayementW3.php**. En effet, la particularité de ces pages est qu'elles ne possèdent pas de `<head>` ou de `<body>` à proprement parlé mais que leur code interne est plutôt "copié" tel quel au sein de la page HTML depuis laquelle elle a été appelé. Ainsi, il n'est pas possible de gérer leur sécurité au moment de leur chargement si l'on cherche à se rendre sur ces pages en cliquant sur un onglet. La solution qui a été trouvée à ce problème a donc été de **déplacer le `<script>`** contenant tout de même des fonctions de sécurité tout à la fin du code interne de ces pages. Et comme, rappelons-le, les onglets de notre site ne s'affichent que si son utilisateur a les droits de s'y rendre, alors la sécurité de la page est quand-même assurée.

```

1 <script type="text/javascript">
2   alert("vous n'avez rien à faire ici :/");
3   window.location.href="AccueilW3.php";
4 </script>

```

Voici le code mis à la fin de chaque page chargée en AJAX.

Notons enfin que nous avons tout de même fait en sorte que la page en question ne puisse pas être chargée si une personne cherche directement à s'y rendre sans passer par les **onglets sécurisés** (= en passant par la barre de recherche du navigateur). En effet, si tel est le cas, alors une alerte s'affichera et l'utilisateur sera automatiquement redirigé vers la page d'Accueil.

En résumé, à l'exception des pages chargée en AJAX, le fait d'avoir placé nos conditions d'accessibilité à une page au niveau de son **body** au sein d'une **fonction JavaScript onload** nous permet de sécuriser son accès, que ce soit à partir d'un onglet cliqué ou même à partir la barre de recherche internet du site !

Quant-à la **sécurité de certaines fonctionnalités** du site, elle est également assurée par une fonction JS en onload : elle vérifie tout d'abord si l'utilisateur est connecté ou pas / abonné ou pas puis elle afficher ou non certains éléments d'une page donnée. Par exemple, si un simple visiteur arrive sur la page d'Accueil du site, notre fonction fera en sorte de détecter

qu'il n'est pas connecté et donc de ne pas afficher son profil en bas de la page en question ! Ci-dessous un extrait du code de cette fonction de la page d'Accueil :

```

1 if (connection==0){
2     document.getElementById('id01').style.display='block';
3     document.getElementsByClassName('card2')[0].remove();
4 }
```

Au final, la seule contrainte qu'impose cette fonction de sécurité est de bien placer les éléments dont l'affichage est variable au sein de block HTML possédant soit un **identifiant** soit un **nom de classe**. Sans quoi, notre fonction JS ne pourrait pas savoir de quel élément il s'agit !

II.4 Code JavaScript

Le code JavaScript de notre site se situe principalement dans le *<head>* de chacune de nos pages, ou à la fin de leur *<body>* si elles sont chargées en AJAX depuis d'autres pages.

L'intérêt principal de notre code JavaScript est de **pouvoir modifier certaine partie d'une page / donner l'accès à certaines fonctionnalités du site** en fonction du statut de son utilisateur (s'il est connecté, s'il est abonné et à quel niveau,...).

L'exemple le plus marquant est celui des **onglets** de chacune des pages du site. En effet, la mise en place de ces onglets va se faire au **chargement de la page** (= en *onload*) et va tenir compte du niveau d'abonnement et de la connexion éventuelle d'un utilisateur afin de placer, ou non, certains onglets. L'onglet messagerie sera ainsi masqué si l'utilisateur en question n'est pas connecté ou n'est pas abonné au site. De même, le bouton "Connexion" sera remplacé par le bouton "Déconnexion" si l'utilisateur du site est déjà connecté à son compte.

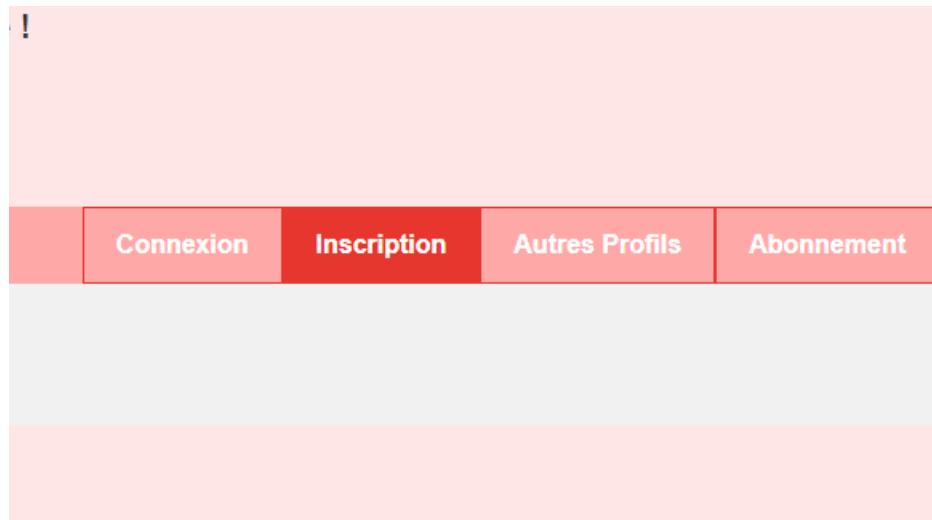


FIGURE 13 – Liste des onglets si on est pas connecté



FIGURE 14 – Liste des onglets si on est connecté de niveau 2

```

1 //function qui ajoute les onglet en
2 fonction de si tu es connecte et si tu es abonne
3 function Onglet (connection , abonnement){
4
5     var liste = document .getElementById ( ' listeOnglet ' );
6     if ( connection==1){
7         switch ( abonnement){
8             case 1 :
9             case 2 :
10
11             var newlien=document .createElement ( 'a' );
12             newlien .setAttribute ( " href " , " MessagerieW3 .php " );
13             newlien .setAttribute ( " class " , " right " );
14             newlien .innerHTML= "<b> MESSAGERIE </b> ";
15
16             liste .appendChild ( newlien );
17             break ;
18             default : alert ( " pas d 'abonnement " );
19                     }
20
21 }

```

Ci-dessus le code de la fonction affichant les onglets de la page **MonProfilW3.php**. Sachant que la base commune à tous les onglets de cette page sont les onglets "Accueil", "Déconnexion", "Abonnement" et "Recherche", on remarque que l'onglet "Messagerie" s'ajoutera si l'utilisateur est connecté (cf. ligne 6) et qu'il est abonné, de niveau 1 ou de niveau 2 (cf. lignes 8 et 9). Notons ici que l'onglet "Connexion" n'apparaît jamais car pour accéder à cette page, il faut nécessairement que l'utilisateur du site y soit déjà connecté! Ce cas là n'est donc pas à prévoir et l'on peut automatiquement se contenter du bouton "Déconnexion".

Mais les onglets ne sont pas les seules modifications possibles à apporter au chargement d'une page. En effet, il y a aussi la fonction qui place les boutons "Envoyer un message" sous les pseudos des cartes de profils / qui affiche les filtre de recherches dans la page **RechercheW3.php** si l'utilisateur du site est abonné, celle qui affiche les allergies de chaque membre du site

sur leur profil,...

Autrement dit, chaque page aura son lot de fonctions à exécuter au moment de son chargement. Cependant, **dans le onload placé dans le body, nous ne pouvons exécuter qu'une seule fonction à la fois!** D'où la création de l'unique fonction choseAFaireAuChangementDeLaPage(...) présente dans la grande majorité des pages et qui permettra d'appeler toutes les autres. En voici un exemple :

```

1 //fonction appelee au changement de la page recherche :
2
3 function
4 choseAFaireAuChangementDeLaPage ( connexion , abonnement ){
5   if ( connexion==0){
6     // alert ("pas connecte ");
7     document . getElementById ( ' BoutonFiltreR ' ) . remove ();
8     document . getElementById ( ' afficheFiltres ' ) . remove ();
9   }
10
11   if ( abonnement==0){
12     document . getElementById ( ' afficheFiltres ' ) . remove ();
13   }
14   Onglet ( connexion , abonnement );
15   messagerie ( abonnement );
16
17 }
```

Cependant, toutes les fonctions en JavaScript ne sont pas appelées dès le chargement de la page. Il en existe aussi appelées dans certaines balises du body, à l'aide de **onclick** et qui gère certains détails. On peut ainsi citer la fonction ChangementPhotoCarre() présente dans les pages **AccueilW3.php**, **MonProfilW3.php** ou encore **ProfilAutreUtilisateurW3.php** et qui permet de positionner les trois boutons de défilement des photos d'un profil ainsi que de changer leur couleur lorsque l'on clique dessus.



FIGURE 15 – screen de la photo de profil et des carrées qui change de couleur et la photo dès que l'on clique dessus

Pour finir, nous avons utilisé le JavaScript afin de **charger certaines pages en AJAX**. La fonction enregistrerPseudoDansFichier() présente dans les pages d'Accueil et de recherche ou la fonction souscrireAbonnement() présente dans la page d'abonnement permettent par exemple de charger du AJAX. Plus précisément, la fonction de la page d'Accueil et de recherche charge le PHP, le HTML et le JavaScript de la page ProfilAutreUtilisateurW3.php dans le body de la page Accueil/Recherche ce qui permettra de charger la fonction choseAFaireAuChangementDeLaPage(...), qui ne sera donc pas appelée à l'aide du onload car la page n'est pas chargée à proprement parler. En effet, le body de la page Accueil/Recherche va juste être remplacé par celui de l'autre page! Quant-à la fonction dans la page Abonnement, elle est un peu différent car on ne charge pas le JavaScript. Mais son fonctionnement reste le même.

L'utilisation de AJAX permet donc de passer des variables d'une page à une autre sans utiliser de formulaire. Dès que l'on clique sur le pseudo d'un utilisateur (resp. un abonnement), nous allons récupérer ces informations et nous allons charger la page AJAX les contenant. Voici ci-dessous un exemple de code si l'on clique sur un niveau d'abonnement à souscrire :

```

1 //fonction appelee dans la page abonnement :
2
3 function souscrieAbonnement(bouton){
4     var Niveau=bouton.getAttribute('id');
5     var body=document.getElementById('BodyAbonnementW3');
6     var xhr= new XMLHttpRequest();
7
8     xhr.onreadystatechange = function() {
9         if ((xhr.readyState==4) && (xhr.status==200)){
10             var reponse=xhr.responseText;
11             body.innerHTML = reponse;
12         }
13     }
14 }
15
16 xhr.open("POST", "PagePayementW3.php", true);
17 xhr.setRequestHeader("content-type",
18 "application/x-www-form-urlencoded");
19 xhr.send("Niveau="+Niveau);
20 }
21
22
23 //verification du niveau d abonnement :
24
25 function AboVerif(bouton , Niveau){
26     if(Niveau!=0){
27         if( confirm("Vous avez deja un abonnement !
28 Etes vous sur de vouloir continuer ?")){
29             souscrieAbonnement(bouton);
30         }
31     } else{ souscrieAbonnement(bouton); }
32 }
```

Les fonctions ci dessus seront appelées si nous cliquons sur un niveau d'abonnement.

Et ci dessous voici l'endroit où on appelle la fonction

```

1 <input type="button" class="Niveau"
2 id="Niveau1" name="Niveau1"
3 value="Je souscris à un abonnement de niveau 1"
4 onclick="AboVerif(this,
5 <?php echo ChercheBDD("abonne", $_SESSION["pseudo"]);?>
6 );"/>
```

II.5 Code de la base de données

Dans cette partie, nous verrons comment a été **créée la base de données** de notre projet ainsi que le cheminement de pensées et les choix qui nous ont guidés tout au long de ce travail. Nous y verrons également la **méthodologie** que nous avons suivie afin de la **remplir** tout en étant sûr de pouvoir tester la majorités des cas proposés par notre site de rencontre.

Du point de vue de sa **structure générale**, notre base de données comportera **trois types de tables** différentes : une table dite "**principale**" et qui comprendra l'ensemble des profils des utilisateurs inscrit au site, une table réservée à la "**messagerie**" où l'ensemble des discussions des utilisateurs abonnés au site sera rassemblé et enfin des tables de type "**autres**", souvent liées à des fonctionnalités particulières du site (blocage d'un membre, envoi de like,...).

Par ailleurs, notre base de donnée s'appelle **bddfinale**.

Table de données principale

Pour créer la base de données principale contenant toutes les informations, nous avons commencé par réfléchir à combien de lignes devait contenir notre table, et quelles catégories devaient y figurer.

Table de données "Messagerie"

Pour créer la table de données de la Messagerie, nous avons tout d'abord réfléchi à ce que nous devions y incorporer. Après cette première réflexion, nous avons conclu que nous avions besoin 5 lignes. La première ligne : **IdMessage**, qui correspond à l'identifiant du message de sorte à le rendre unique

et à tout de même pouvoir différencier deux messages contenant exactement le même texte. La deuxième ligne est celle de **l'Emetteur**, c'est-à-dire le pseudonyme de la personne qui envoie le Message tandis que la troisième est celle du **Recepteur**, qui contient cette fois-ci le pseudonyme de la personne qui reçoit le message. La quatrième ligne, **Texte**, correspond au contenu du message et la cinquième, enfin, devait contenir la date et l'heure d'envoi du texte.

Or, à l'issue d'un second temps de réflexion, nous avons finalement fait le choix de séparer la dernière ligne de notre table en deux lignes distinctes afin de simplifier la gestion du moment d'envoi des messages. Ainsi, nous avons choisi de mettre la **date**, au format **AAAA/MM/JJ** au sein de la 5ème ligne et de mettre l'**heure** dans la 6ème. En effet, ce choix permet de rendre certaines manipulations plus simples, comme par exemple au niveau de la fonction qui permet **d'effacer** les messages. Cela rend également **l'affichage** des messages plus claire.

Relativement au remplissage de cette table, nous avons suivi **la même méthodologie** qu'utilisé précédemment pour la table principale de la base de données. Autrement dit, nous avons fait en sorte que tous les cas - tous les niveaux abonnements, tous les sexes et toutes les orientations - soient dans le tableau, afin d'ensuite pouvoir vérifier qu'aucun bug n'apparaît pour certaines catégories.

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
1	IdMessage	int(11)			Non	Aucun(e)			
2	Emetteur	text	latin1_swedish_ci		Non				
3	Recepteur	text	latin1_swedish_ci		Non				
4	Texte	text	latin1_swedish_ci		Non				
5	Date	date			Non	Aucun(e)			
6	Heure	time			Non	Aucun(e)			

FIGURE 16 – Capture d'écran de la table de la Messagerie. On y retrouve bien les **type** de chacune des données qu'elle contient.

Autres tables de données

Pour finir, nous avons eu besoin de plusieurs autres tables de données, plus petites, et afin de gérer certaines **fonctionnalités** proposées par notre site. Parmi elles, on retrouve **le blocage et le signalement** d'autres utilisateurs ainsi que **l'envoi de Jaime ou de SuperJaime** pour les abonnés.

La table de signalement est construite de la manière suivante : on a l'**Id-Signalement**, afin de rendre chaque signalement unique. Ensuite nous avons le **Signaleur**, et le **Signalé**. La dernière ligne du tableau correspond au **motif** du signalement.

A noter que **signaler quelqu'un permet d'informer (via un message) les administrateurs du site qu'une personne vous a envoyé un message incorrect et/ou inapproprié (= irrespectueux, propos haineux,...)**. Le bouton permettant d'effectuer cette action se trouve au sein de la page **Messagerie.W3.php**

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
<input type="checkbox"/>	1	Id_Signalement	int(11)		Non	Aucun(e)		 Modifier  Supprimer ▾	Plus
<input type="checkbox"/>	2	Signaleur	text	latin1_swedish_ci	Non			 Modifier  Supprimer ▾	Plus
<input type="checkbox"/>	3	Signale	text	latin1_swedish_ci	Non			 Modifier  Supprimer ▾	Plus
<input type="checkbox"/>	4	Motif	text	latin1_swedish_ci	Non			 Modifier  Supprimer ▾	Plus

FIGURE 17 – Capture d'écran de la table de données de Signalement d'autres utilisateurs

La table de blocage est construite de la manière suivante : **Id-Blocage** qui permet de rendre unique chaque blocage. Et ensuite, une ligne pour le **Bloqueur**, et une pour le **Bloqué**.

A savoir que **bloquer quelqu'un permet de ne plus avoir d'interactions avec cette personne, que ce soit à travers des messages, des réception de Like,...** Le bouton permettant d'utiliser cette fonctionnalité se trouve directement sur la page **ProfilAutreUtilisateurW3.php** du membre à bloquer.

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
<input type="checkbox"/>	1	Id_blocage	int(11)		Non	Aucun(e)		 Modifier  Supprimer ▾	Plus
<input type="checkbox"/>	2	Bloqueur	text	latin1_swedish_ci	Non			 Modifier  Supprimer ▾	Plus
<input type="checkbox"/>	3	Bloqué	text	latin1_swedish_ci	Non			 Modifier  Supprimer ▾	Plus

FIGURE 18 – Capture d'écran de la table de données du Blocage

Enfin, pour les tables Jaime et SuperJaime (pour les personnes abonnées), nous avons le même principe : l'**Id-Jaime** ou l'**Id-Superjaime** afin d'identifier et de rendre unique le Jaime ou le SuperJaime sur la première ligne des tableau respectifs. Pour la deuxième ligne, il s'agit de l'**Aimeur**, ou du **SuperAimeur** et enfin, pour la troisième ligne, il s'agit de l'**Aimé** ou du **SuperAimé**.

Les boutons d'envoie de Like ou de SuperLike si la personne est abonnée se trouvent au sein de la page **ProfilAutreUtilisateur** du membre en question.

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
<input type="checkbox"/>	1	Id_Jaime	int(11)		Non	Aucun(e)		Modifier Supprimer ▾ Plus	
<input type="checkbox"/>	2	Aimeur	text	latin1_swedish_ci	Non			Modifier Supprimer ▾ Plus	
<input type="checkbox"/>	3	Aime	text	latin1_swedish_ci	Non			Modifier Supprimer ▾ Plus	

FIGURE 19 – Capture d'écran de la table de données des Jaime

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
<input type="checkbox"/>	1	Id_Superjaime	int(11)		Non	Aucun(e)		Modifier Supprimer ▾ Plus	
<input type="checkbox"/>	2	Superaimeur	text	latin1_swedish_ci	Non			Modifier Supprimer ▾ Plus	
<input type="checkbox"/>	3	Superaime	text	latin1_swedish_ci	Non			Modifier Supprimer ▾ Plus	

FIGURE 20 – Capture d'écran de la table de données des SuperJaime

Au niveau du **remplissage** de l'ensemble de ces tables, il a été réalisé de la même manière : pour la première phase de test, nous avons fait en sorte de pouvoir réaliser **au moins 6 exemples** avec, à chaque fois, deux exemples d'homme aimant les hommes, deux exemples de femme aimant les femmes, et deux exemples d'une femme et d'un homme aimant respectivement les hommes et les femmes. Ainsi, nous étions sûr que la majorité des **cas possibles de sexe et d'orientation** étaient représentés. Nous avons, enfin, **fait varier le niveau d'abonnement** afin de vérifier qu'il n'existe pas de bug lié à cette caractéristique.

Ci-dessous quelques exemple de tables de notre base de données qui ont été ainsi remplies :

Id_Signealement	Signaleur	Signale	Motif
1	EloElo	CathWoman	Insultes
2	Chateline	Sirine37	Propos racistes
3	Lolo	Seb	Menaces
4	AlexDu38	Alex6	Harcelment
5	Clem	Logan	Propos homophobes
6	Viv	Jodielol	Remarques déplacées

FIGURE 21 – Capture d'écran de la table de données du Signalement

<u>Id_blocage</u>	<u>Bloqueur</u>	<u>Bloqué</u>
1	Emilliiie	JulieJulie
2	Lolalol	Clarus
3	YannYann	MarcMarc
4	DanDan	Tomyyy
5	Clairouuu	TheoMik
6	Hugo02	LouisDkkr

FIGURE 22 – Capture d'écran de la table de données du Blocage

<u>Id_Jaime</u>	<u>Aimeur</u>	<u>Aime</u>
1	Romie	Celia42
2	Valentina	Blanche
3	LiamDu41	Lyamy
4	HarryPotter	RoyRoy
5	HelenaW	CharlyChocolaterie
6	Baz	Vic

FIGURE 23 – Capture d'écran de la table de données du Jaime

<u>Id_Superjaime</u>	<u>Superaimeur</u>	<u>Superaime</u>
1	Dorothie	Doriie
2	LucLuc	Justin102
3	Bebe	SachaPoke
4	Maelll	Alicia18
5	HaileyKyoko	Suzon
6	Rome	Jam

FIGURE 24 – Capture d'écran de la table de données de SuperJaime

Au final, nous pouvons dire que nous avons utilisé une **méthodologie précise et construite** pour créer chacune des tables de notre base de données. Nous devions penser à **chaque cas possible**, et être en mesure de **tester** la majorité de ces combinaisons et associations possibles. Enfin, les choix de remplissage que nous avons fait nous semblent pertinents, tant pour une question de **pratичité** que pour une question d'**efficacité**.

II.6 Code PHP

Lien PHP-SQL

Afin d'alléger le code de chaque page nous avons regroupé une grande partie des fonctions php dans le fichier **bddInfoRecherche.php**. Nous allons à présent expliquer les méthodes et commandes utilisées afin de réaliser ces fonctions.

- Le premier groupe de fonctions correspond aux fonctions basiques qui font appel à des fonctions **mysqli**. Elles permettent de se connecter à une base de données (fonction **Connect()** avec `mysqli-connect("localhost", "root", "", "bddfinale")` ;), d'exécuter une requête(`mysqli-query(conn, requete)`) et de se déconnecter (fonction **Disconnect(conn)** avec `mysqli-close(conn);`). Elles seront toutes appelées par la suite au sein d'autres fonctions plus complexes.
- Le deuxième groupe de fonctions permet de rechercher des informations concernant un profil en particulier. Celles-ci sont très souvent appelées au sein des autres pages du site.

La fonction **ChercheBDD(champ,pseudonyme)** permet de rechercher une information précise d'un utilisateur à partir de son pseudo grâce à la requête :

```
1 SELECT $champ FROM autresutilisateurs
2 WHERE Pseudo='$pseudonyme' ;
```

Il est important de noter que nous nous assurons dans notre base de données que chaque pseudo est unique. Comme cet identifiant correspond à un seul utilisateur nous sommes sûr que cette fonction ressort la bonne information.

Les quatre fonctions suivantes vont nous permettre de rechercher un profil selon plusieurs critères. Tout d'abord la fonction **TrouveLigne(lvlAbo)** permet de trouver un profil aléatoirement en tenant simplement compte de son niveau d'abonnement. La fonction **TrouveProfilBarreR(maRecherche)** permet quant à elle de trouver le profil qui correspond à la recherche entrée en paramètre.

De la même manière, mais en ajoutant des critères, nous pouvons, avec la fonction **TrouveProfilFiltres(lvlabo,lieu,allergie)**, retourner un tableau avec tous les pseudos des utilisateurs qui correspondent aux critères de lieu et

d'allergies. Pour ce faire nous utilisons une requête SELECT avec plusieurs conditions :

```

1 WHERE Abonnement='lvlabo' AND
2 (Allergie1='$allergie' OR Allergie2='$allergie'
3 OR Allergie3='$allergie')
4 AND Lieu='$lieu' ";

```

En effet, les conditions WHERE des requêtes peuvent être construites avec des AND (ou `&&`) et des OR (ou `||`). Une structure conditionnelle IF nous permet de modifier la requête exécutée en fonction des paramètres qui ont été rentrés. Par exemple, si aucun critère de lieu à été entré la requête n'en tiendra pas compte pour sa recherche dans la BDD.

Enfin, la fonction **ValideProfilFiltres(tabGens,AgeMin,AgeMax)** complète **TrouveProfilFiltres** en vérifiant d'autres critères comme l'âge et l'orientation sexuelle avec des structures conditionnelles. Si le profil est effectivement validé il est aussi affiché dans cette fonction.

Toutes les fonctions décrites ci-dessus font directement appel à notre base de données pour rechercher des informations. Lorsque la requête donne plusieurs résultats nous les avons récupérés dans un tableau à l'aide du code suivant :

```

1 $n = mysqli_num_rows($ligne);
2 $Tab = array();
3 if ($n > 0) {
4     while ($TabLigne = mysqli_fetch_assoc($ligne)){
5         $Tab[] = $TabLigne["Pseudo"];
6     }
7 }

```

La commande **mysqli-fetch-assoc** permet de récupérer les données trouvées dans la BDD. Si une seule valeur est recherchée on peut utiliser directement la commande **mysqli-fetch-array** pour accéder au tableau de résultats.

- Le groupe de fonction suivant agit sur le profil de l'utilisateur connecté. En effet, la base de donnée peut être modifiée directement à partir des actions

de l'utilisateur.

La fonction **NouveauProfil(prenom,nom,pseudonyme,mdp,date,sexe,orientation)** permet comme son nom l'indique, d'enregistrer un nouveau profil dans la base de données grâce à la requête "INSERT VALUES ()".

La fonction **ModifierProfil(pseudonyme,champ,nvValeur)** fait appel à la requête SQL :

```
1 UPDATE bdd SET $champ='$nvValeur'
2 WHERE Pseudo='$pseudonyme';
```

et permet de modifier la valeur d'un champ dans le profil.

Sur le même principe la fonction **SupprimerProfil(pseudonyme)** appelle la requête SQL "DELETE" et permet de supprimer le profil d'un utilisateur.

- Passons maintenant aux fonctions d'affichage et d'initialisation :

La fonction **InitSession(pseudonyme)** initialise toutes les variables de session à l'aide de notre fonction **ChercheBDD**. Cela permet d'alléger le code des autres pages et d'avoir facilement accès aux informations de l'utilisateur dans le reste du programme. Nous avons également créé des fonctions d'affichages : **AfficheSexe**, **AfficheOrientation**, **AfficheDepartement**, **AfficheAllergie**... Chacune de ses informations est représentée dans la base de données par un entier, d'où l'existence de ses fonctions. En ce qui concerne les départements et les allergies nous avons créé deux fichiers textes comportant leurs noms respectifs. Le code suivant nous permet de rechercher la ligne (et donc le texte) qui correspond au numéro :

```
1 $ouverture = fopen ("departements.txt","r");
2         for ($i=0;$i<$var ; $i++){
3             fgets ($ouverture);
4         }
5         echo fgets ($ouverture)."  
>";
6         fclose ($ouverture);
```

- Les quelques fonctions suivantes sont assez simples et permettent d'accéder rapidement à des informations sur l'utilisateur comme l'état de la connexion (**VerifCo()**), son niveau d'abonnement (**VerifAbo()**), son nombre d'allergies (**CalculNbAllergies**) etc ...

- Viennent ensuite les fonctions utiles au fonctionnement de notre messagerie. En effet, il est possible d'envoyer un message avec la fonction **Nouveau-**

Message. De la même manière que pour enregistrer un nouvel utilisateur, nous allons insérer une nouvelle ligne dans la table "messages" à l'aide de la requête :

```
1 "INSERT INTO messages
2 VALUES (NULL, 'de', 'a', 'msg', 'date', 'heure')";
```

On peut remarquer que la valeur de l'id ici est défini comme NULL. Or comme lors de création de la table il est préciser que cette valeur est "NOT NULL" et "AUTO-INCREMENT", l'id va automatiquement prendre une valeur logiquement croissante dans la table.

La fonction **TrouveAfficheMessages** permet d'afficher tous les messages relatifs à une conversation entre deux personnes. La fonction **TrouveNb-Conversations** détermine l'ensemble des personnes avec qui l'utilisateur connecté a des conversations. La fonction **TrouveDernierMessage** trouve le texte et l'heure du dernier messsage échangé entre deux personnes. Nous avons bien entendu une fonction permettant de supprimer une conversation (**SupprimeMessages**) afin qu'un utilisateur puisse libérer son espace messagerie.

De plus, afin d'inciter les utilisateurs à être actif sur notre site, nous sauvegardons la date de l'envoi du message pour que celui-ci soit supprimé au bout de 7 jours. Cette action se fait grâce à la fonction **SupprimerMessages7Jours** qui est appelée à chaque fois qu'un utilisateur se connecte au site. La requête utilisée dans cette fonction est

```
1 "SELECT mp_id FROM Messages
2 WHERE mp_date + INTERVAL 7 DAY <= NOW()"
```

L'outil de comparaison \leq permet de vérifier que la date d'envoi du messsage + 7 (jours) est inférieure à la date actuelle. Si ce n'est pas le cas le message est stocké dans un tableau puis supprimé.

Nous allons maintenant expliquer les fonctions qui permettent de bloquer un autre utilisateur. Pour la fonctionnalité de blocage nous avons la fonction **Bloquer** qui ajoute une nouvelle ligne dans la table blocage avec un id-blocage, un bloqueur et un bloqué. La fonction **TrouveBlocage** permet de trouver toutes les personnes ayant bloqué un certain utilisateur tandis que **TrouveBloquesPar** permet de trouver tous les utilisateurs qui sont bloqués par une certaine personne. Enfin, la fonction **Debloquer** permet de débloquer un utilisateur.

Les fonctionnalités de signalement, like et "Superlike" sont gérées par des fonctions similaires à celles de blocage en appelant leurs tables respectives (signalement,jaime,superjaime).

Le code PHP dans le programme principal

Nous allons maintenant voir comment toutes ces fonctions sont appelées dans notre programme. Dans la majorité de nos pages nous utilisons les variables de SESSION précédemment initialisées et les fonctions d'affichages ainsi que la fonction **ChercheBDD**. Nous allons revenir sur quelques éléments php dans chaque page.

- Accueil : Les 3 profils affichés sur la page d'accueil sont choisi aléatoirement par la fonction **TrouveLigne**, la seule condition étant que si l'utilisateur est connecté son propre profil ne s'affiche pas. Pour la partie Connexion un formulaire php nous permet de vérifier si les identifiants entrés sont valides avec la ligne :

```
1 if (( $_POST[ 'mdp' ] ==  
2           ChercheBDD( "MotDePasse" , $_POST[ 'pseudo' ] ) ));
```

- MonProfil : Les fonctions **TrouveLike** et **TrouveSuperLike** sont appelées en dessous du profil pour afficher les noms (s'ils existent) des utilisateurs ayant aimés mon profil.

- ModifierMonProfil : cette page est constituée d'un formulaire php

```
1 <form action='ModifierMonProfilW3 .php ' method='POST'>
```

qui recharge la page et envoie les données récoltés par la méthode **POST**. Si le formulaire a été rempli nous enregistrons les nouvelles valeurs grâce au code suivant :

```
1 if ( $_POST[ "description" ] != $_SESSION[ "description" ] ){  
2   $_SESSION[ "description" ]=  
3   ModifierProfil( $_SESSION[ "pseudo" ] ,  
4     "Description" ,$_POST[ "description" ] );  
5 }
```

De plus, comme il est possible de changer sa (ses) photo(s) de profil nous utilisons un

```
1 <input type='file' name='photo []' multiple
2 accept='image/*'/>
```

qui autorise l'utilisateur à importer uniquement des fichiers de type image (jpg,jpeg,png...). Celles-ci sont stockées dans le tableau

```
1 $_FILES [ ' photo ' ] [ ' name ' ] [ ]
```

- Recherche : La page Recherche est assez complexe car plusieurs méthodes sont utilisées pour afficher les mini-profil en fonction des critères de recherche de l'utilisateur. Tout d'abord, nous affichons par défaut 5 profils d'abonnés de niveau 2, 4 profils de niveau 1 et 3 de niveau 0 correspondant tous aux critères d'orientation de l'utilisateur. S'il s'agit d'un visiteur les profils affichés respectent juste les niveaux d'abonnement. Pour cela nous utilisons la fonction **TrouveLigne** (comme dans Accueil) puis des structures if pour s'assurer de proposer des profils correspondants aux critères d'orientation :

```
1 if (((ChercheBDD("orientation",$_SESSION["Profil"])!=
2 $_SESSION["sexe"]])&&(ChercheBDD("orientation",
3 $_SESSION["Profil"])!=0))
4 || ((ChercheBDD("sexe",$_SESSION["Profil"])!=
5 $_SESSION["orientation"])
6 &&($_SESSION["orientation"]!=0)) {
7
8 // conditions pour ne pas afficher ce profil
```

Ensuite, l'utilisateur peut rechercher un profil par rapport à son pseudo. Dans ce cas c'est la fonction **TrouveProfilBarreR** qui est appelée. Enfin, si les autres filtres sont activés ce sont les fonctions **TrouveProfilFiltres** et **ValideProfilFiltres** qui sont appelées. Notons aussi que c'est à cet endroit que nous vérifions si l'utilisateur n'a pas inversé le critère d'âge minimum et d'âge maximum. Si c'est le cas nous inversons les valeurs et lui rappelons son erreur :

```

1 if ($_POST[ " ageMin "]>$_POST[ " ageMax "]){
2     $tmp = $_POST[ " ageMin "];
3     $_POST[ " ageMin "]= $_POST[ " ageMax "];
4     $_POST[ " ageMax "]= $tmp;
5 }
```

- ProfilAutreUtilisateur : Il est possible sur cette page d'attribuer un Like (ou SuperLike) au profil d'un autre utilisateur. Pour cela nous vérifions si l'action 'j'attribue un like à cette personne' a déjà été effectuée ou pas avec le code suivant :

```

1 $aime =0; // profil non aimé par moi
2 $Tab = TrouveSuperLike($_POST[ " Pseudo "]);
3 if (isset($Tab)){
4     for ($i=0;$i<sizeof($Tab); $i++){
5         if ($_SESSION[ 'pseudo ']==$Tab[ $i ]){
6             $aime=1; // j'aime déjà ce profil
7             break ;
8         }
9     }
10 }
```

Si je n'ai pas déjà donné un like à ce profil un formulaire permet de recharger la page du profil grâce au input de type **hidden**. La fonction Liker n'est appelée que si l'utilisateur a cliqué sur le bouton "Liker". Ensuite, ce bouton disparaîtra car il n'est possible d'en donner qu'un seul like à chaque personne. Le même principe est utilisé pour le bouton 'Bloquer' mis à part le fait que celui-ci se transforme en 'débloquer' après clic.

- Messagerie : Dans la partie Messagerie nous avons sur la partie gauche la liste des conversations de l'utilisateur qui s'affiche grâce aux fonctions TrouveNbConversations et TrouverDernierMessage. Au centre, la fonction TrouveAfficheMessages permet d'afficher l'ensemble de la conversation entre 2 personnes, du message le plus récent au plus vieux, grâce aux id qui représentent la clé primaire de la table "messages" et possèdent l'option "Auto Increment". Sur la partie droite, il est possible de signaler un autre utilisateur en entrant son pseudo et un motif de signalement dans le formulaire. Au chargement de la page, nous procédons à plusieurs vérification si celle-ci

est appelée avec ajax (dans ce cas la variable S-POST['Pseudo'] est initialisée). Par exemple le code suivant permet de vérifier si l'utilisateur a déjà une conversation avec l'autre personne.

```
1 $TabConv = TrouveNbConversations($_SESSION["pseudo"]);
2     for ( $i=0;$i<sizeof($TabConv); $i++){
3         if ( $_POST[ 'Pseudo']==$TabConv[ $i] ){
4             $autrePseudo=$_POST[ 'Pseudo' ];
5             break;
6         }
7     }
```

III BILAN

III.1 Bilan du projet

Réalisations et points forts du projet

Notre groupe est globalement satisfait de notre projet ; du **thème original** qu'il traite, à la **forme** que nous avons su lui donner et aux réalisations que nous su mener jusqu'à leur terme. De manière générale, nous pensons que la cohérence de notre projet est l'un de ses plus gros points forts. En effet, la réflexion que nous avons effectuée en amont de la phase de codage nous a permis de suivre une **ligne directrice claire** et précise, le rendant cohérent dans son entièreté. Nos différents **planning** ainsi que la **répartition des tâches** évoquée au début de ce rapport ont donc été des outils capitaux dans notre travail qui nous ont permis de réaliser à temps la majorité de nos objectifs fixés avec des membres aptes à modifier leur partie de code sans avoir à poser de questions techniques à d'autres membres. Nous sommes donc satisfait de l'**organisation** que nous avons su nous imposer au cours de ces deux derniers mois.

Relativement à l'**architecture de notre site** en elle-même, nous trouvons qu'elle est plutôt simple à exploiter ainsi que fidèle à celle que nous avions imaginée tout au début de notre projet. Sa **simplicité de prise en main** vient des fonctions JavaScript permettant d'afficher ou non les différents onglets présents sur chaque page e selon le niveau (de connexion et/ou d'abonnement) de l'utilisateur du site. Ainsi, un simple visiteur ne pourra jamais cliquer sur l'onglet d'une page à laquelle il n'a pas accès ce qui est à la fois **pratique et sécuritaire** ! En plus de cela, l'ajout de pages comme **Deconnexion.php** ou **RetourAccueil.php** - qui n'utilisent que les langages HTML et CSS et qui ne servent qu'à indiquer le résultat d'une action avant de rediriger l'utilisateur vers une autre page - aident grandement, selon nous, à clarifier les actions qui été réalisée sur le site au fur et à mesure de son utilisation.

Du point de vue du **contenu de notre site**, nous sommes particulièrement fière de notre **page recherche** qui contient non seulement de nombreux filtres de recherche mais aussi une barre de sélection de pseudo ! Toujours en lien avec cette page, nous sommes également satisfait d'avoir pu concevoir

une **base de données fournie** (360 profils) et capable d'offrir un large choix d'utilisateurs aux personnes inscrites au site. Cela constitue pour nous l'un des grands point fort de notre projet.

Au sujet de la page d'Accueil, nous trouvons original l'idée d'avoir créer une sorte de "pop-up" lorsque l'on clique sur l'**onglet "Connexion"**. En plus d'avoir été intéressant à coder en CSS, nous avons aussi pensé que cela changeait de tous les autres onglets qui mènent systématiquement vers de nouvelles pages à part entière.

Enfin, nous sommes fière de notre page **d'abonnement** dans le sens où nous trouvons qu'elle est similaire **visuellement** à de véritable page de payement. De même, nous trouvons que les avantages que nous avons réservées aux différents niveaux d'abonnés au site sont cohérents et intéressants (augmentation du nombre de discussions,...) !

Limites, difficultés et points faibles du projet

Cette nouvelle sous-partie est pour nous l'occasion de reconnaître que nous avons conscience que notre projet est loin d'être parfait que notre site possède des faiblesses. Parmi elles, on peut nommer la messagerie qui reste assez basique, et qui est notamment assez limitée par l'emploi de notre **base de données de messagerie peu remplie** pour des questions de timing.

Par ailleurs, et en lien avec notre **base de données principale** cette fois-ci, nous avons rencontré une difficulté importante lorsqu'il nous a fallu afficher un nombre suffisemment important de profils différents au sein de la page recherche. Ainsi, nous avons calculé qu'il nous faudrait au minimum 360 profils si nous souhaitions pouvoir en afficher au moins un lorsqu'un utilisateur inscrit combine plusieurs filtres de recherche en même temps (avec une fourchette d'âges précise, une localisation donnée,...), ce qui a été long à coder.

Pour en finir avec les limites liées à notre base de données principale, nous pouvons aussi mentionner le fait que les descriptions, passions, qualités/défauts, pire date,... de chaque profil autre que ceux des administrateurs n'ont pas été rentrés. En effet, nous avons estimé que cela n'était pas une priorité par rapport à beaucoup d'autres fonctionnalités du site, véritablement liées à son bon fonctionnement. Nous avons donc préféré nous concentrer sur elles.

Enfin, et c'est sans doute la limite la plus importante de notre projet, sa **sécurité** reste assez légère et peu développée. Cela s'explique par le fait que, lors de la création du site, nous sommes parti du principe que nous serions les seuls à avoir accès à la base de données, et ainsi, que les dangers ne viendraient que de connexions frauduleuses. Nous n'avons donc pas forcément pris en compte l'entièreté des risques qu'un site de rencontre hébergé sur Internet pourrait rencontrer. De plus, étant donné que nous n'avons vu que peu d'exemples pour protéger une page web, et au vu de l'immensité du projet que nous souhaitions réaliser, il nous a été compliqué de nous renseigner sur cette partie, et par conséquent mettre en place un système de sécurité optimum.

Perspectives

Malgré ces quelques points faibles, nous sommes fiers de notre site WEB, et nous avons enfin pensé à différentes idées qui pourraient permettre de l'améliorer dans une optique où il serait mis en ligne et utilisé par le grand public. Par exemple, nous pourrions **améliorer son design** pour lui donner encore un peu plus de vie ainsi que le rendre encore plus moderne. En effet, nous savons que les sites de nos jours utilisent souvent des zones de textes / d'images,... dont l'opacité varie ; procédé que nous n'avons utilisé qu'une seule fois dans notre propre site au niveau du "pop-up" de connexion sur la page d'Accueil.

Nous avons également pensé à rajouter davantages onglets, comme un onglet "**Rencontres en groupe**" qui permettrait de créer des évènements en choisissant une allergie particulière pour un regroupement, ou encore l'onglet "**Commentaires**" qui permettrait aux utilisateurs de faire remonter des remarques ou des idées concernant le site. Enfin, nous avons aussi imaginé l'onglet "**Vidéos**" qui permettrait de poster une vidéo de soi pour se présenter.

Au final, nous sommes parvenus à créer un site internet de rencontre avec une marque avec une identité propre, pourvu de nombreux points forts malgré plusieurs difficultés rencontrée et avec de possibilités d'évolution pour améliorer et ajouter des options !

III.2 Bilans personnels

Chléo Ce projet a été très intéressant et très intense, chaque membres du groupe étaient très investis et sérieux. De plus, je pense qu'il y a eu une très bonne répartition des tâches basée à la fois sur les aptitudes personnelles et sur les envies de chacun. Personne n'était délaissé, nous avions à chaque fois quelque chose à faire et chacun était prêt à aider si quelqu'un avait des difficultés, ce qui nous a permis d'avancer vite et de ne pas rester trop longtemps bloqué sur un problème.

De plus, ce travail m'a permis de m'améliorer dans de nombreux langages. Il a été très instructif. Il m'a permis de bien voir les liens, les différences, les ressemblances entre tous les langages utilisés pour réaliser un site web. Cependant les conditions de travail n'étaient pas adéquates pour le travail en groupe malgré les appels vidéo sur teams, de nombreux problèmes informatiques et de communication ont légèrement ralenti l'avancement du projet. Toutefois, cela a aussi permis d'améliorer notre gestion du travail à distance et notre organisation (à l'aide de teams, du drive, messenger...).

Margaux J'ai particulièrement apprécié ce projet. Je me suis occupée de la partie PHP et j'ai trouvé ce travail très intéressant. J'ai aussi eu l'occasion de travailler sur d'autres parties comme le CSS avec Alice, Chléo avec le Javascript et Téo pour la communication avec la base de données. La communication au sein du groupe était très bonne, nous nous sommes bien organisés. Chaque membre était à l'écoute des autres et nous avons régulièrement échangé nos idées sur les améliorations à apporter.

Je regrette seulement ne pas avoir eu plus de temps pour perfectionner le site car nous avons encore beaucoup d'idées sur des éventuelles améliorations à apporter.

Alice J'ai apprécié travailler ce projet de Développement WEB durant cette fin d'année, notamment en raison du thème de **site de rencontre** qui m'a beaucoup amusé. J'ai aimé la liberté que nous avions au niveau du choix des cibles de notre site, de son identité graphique ou encore de certaines de ses fonctionnalités. Évidemment, cette liberté présentait également des risques, et il nous a fallu apprendre à ne pas être trop ambitieux afin de pouvoir tout terminer dans le temps qui nous était imparti.

Dès le départ, j'ai senti que mon groupe était très investi et nous sommes rapidement parvenu à nous organiser sur le long terme en dépit de la distance qu'imposait le confinement. Nous avons en effet produit les nombreux documents précédemment évoqués dans la première partie de ce rapport avant de réellement commencer à coder notre site, et je pense que c'est grâce à cela que nous avons pu avoir un bon suivit de celui-ci tout le long de sa réalisation. Ainsi, alors que je craignais qu'un groupe de quatre personnes soit compliqué à gérer pour travailler sur un même site, sans même pouvoir se réunir physiquement, il s'est avéré que nous n'avons en fait jamais rencontré de problème à ce niveau là.

Concernant ma participation concrète au codage du site, j'ai principalement eu à charge les langages HTML et CSS. Cela m'a permis d'avoir une vision d'ensemble du site à tout instant, étant donné que je devais toujours être celle qui produisais en première les pages sur lesquelles les autres membres du groupe allaient ensuite pouvoir travailler. J'avoue cependant avoir commencé à me lasser petit à petit de ne pouvoir gérer que l'identité graphique du site. C'est pourquoi, à partir de la fin du premier mois de travail, j'ai commencé à aider Margaux à s'occuper de la partie PHP, et j'ai ainsi pu manipuler diverses variables, structures de contrôles,... comme j'avais toujours eu l'habitude de le faire au cours des trois autres projets d'informatiques rencontrés à l'EISTI.

Au final, j'ai réellement apprécié la richesse de ce projet et je suis fière d'avoir pu le mener à bout. C'est pour moi le projet le plus abouti de tous ceux que j'ai déjà été amenée à faire, dans le sens où je trouve son interface graphique vraiment soignée et où il s'apparente beaucoup à de véritables sites internet tel que j'en ai toujours vu et utilisé. J'aime me dire qu'avec davantage de temps, nous aurions presque pu l'uploader réellement !

Je regrette seulement d'avoir eu à gérer ce projet d'informatique en même temps que trois autres dans trois autres matières...

Téo J'ai beaucoup apprécié ce projet car c'est le premier projet où l'objectif était un objectif assez proche de ce qui pourrait être demandé par une entreprise. J'ai beaucoup aimé le sujet, et surtout la liberté de création, qu'il s'agisse des options ou de l'organisation du site. De plus, le groupe a été très investi, et chacun a apporté des idées et des méthodes, ce qui nous a permis de confronter différents points de vue. Malgré les conditions particulières en cette période, l'organisation a été claire, ce qui a permis une avancée com-

mune et cohérente, avec des réunions très régulières, afin de mettre à jour l'avancement, valider les éléments terminés et ajouter de nouvelles idées. Pour ma part, je me suis occupé de la gestion des bases de données. J'ai beaucoup apprécié travailler dessus, car cela m'a montré l'importance de planifier, et de faire des schémas afin de visualiser l'objectif, et non pas de se lancer directement dans le codage. Le fait que l'utilisation de base de données soit nouveau pour moi m'a beaucoup plu, et m'a donné envie d'en apprendre plus, et pourquoi pas de choisir cette option dans le cycle ingénieur informatique. Dans sa globalité, le projet a été très intéressant autant pour le code, que pour la méthode que nous avons utilisé pour nous organiser. Je pense que l'un de nos gros points forts est notre communication, qui nous a permis de rendre un projet à la fois sérieux, cohérent et complet. C'est donc une très bonne expérience, qui me servira très clairement pour les futurs projets sur lesquels je travailleraï.

CONCLUSION

Ce projet est l'avènement d'un semestre de travail qui nous aura permis d'aborder un grand nombre de questions autour du domaine informatique, de la création d'un site web, ainsi que des méthodes de travail.

La création de ce site web a été très enrichissante, tant d'un point de vue technologique et technique que d'un point de vue humain. Apprendre à s'organiser, à répartir les tâches, à prendre des décisions et faire des choix argumentés, justifiés ne sont que quelques une des nombreuses compétences que nous avons développées avec ce projet. Le fait de pouvoir mettre à profit les langages appris en cours a été vraiment intéressant, d'autant plus que le projet emploi énormément d'entre eux. Parmi eux, on trouve les langages HTML, CSS, JavaScript, ou encore PHP. Nous avons également pu faire appel à des bases de données, ce qui est une partie très intéressante tant d'un point de vue technique que pratique.

L'un des points forts de notre projet à notre sens est la cohérence que nous lui avons donné, en prenant des décisions logiques permettant de rendre la page simple d'utilisation, et complexe de procédés. De plus, la très grande liberté laissée par le sujet, tant pour la conception du site d'un point de vue structurel que d'un point de vue identitaire, nous a permis de développer aussi bien nos capacités algorithmiques que notre réflexion artistique, technique ou encore que notre habileté à décrire nos avancées et nos choix.

Ce projet est donc le reflet d'un semestre de travaux qui nous aura permis de balayer d'un bout à l'autre le spectre de la réalisation d'un projet, allant de sa conception purement théorique jusqu'à la mise en fonctionnement d'un site, en passant par des choix sur la conception et la structure, la création d'une identité d'une marque ou encore la réflexion à d'éventuelles améliorations.

Bibliographie

- [1] <https://www.w3schools.com>
- [2] <https://openclassrooms.com>
- [3] <https://www.journaldunet.fr>
- [4] <https://www.developpez.net>