Ecole internationale des Sciences du Traitement de l'information

Classe préparatoire intégrée deuxième année
Développement web

EISTI ~ BOOK

Rapport de Projet

Auteurs :
Mlle. Roxane Chatry
M. Mathis Gourc
M. Guillaume Proton
M. Hugo Bernedo

Encadrant :
Pr. Elisabeth Ranisavljevic

8 juin 2019

Table des matières

1 Présentation du projet

Le projet que nous avons choisi est celui d'EISTI-BOOK, il consiste en la réalisation d'un réseau social professionnel permettant d'échanger des informations entre étudiants et anciens étudiants. Il faut ainsi développer une plateforme qui facilite l'échange d'offre d'emploi et de messages à travers un fil d'actualité et de profil.

Ainsi il y aura différents types d'utilisateurs à savoir :

— **Visiteur :** Un simple visiteur, qui se connecte au site n'a pas accès au site internet. Il doit posséder un compte pour accéder aux différentes fonctionnalités.
— **Inscrits :** Un utilisateur inscrit a accès à l'ensemble des fonctionnalités du site internet. Il peut publier des informations, envoyer des messages à ses amis, consulter les profils de ses amis et de demander des inscrits en "ami".
— **Administrateurs :** Un administrateur a accès à toute la base de données du site, et par conséquent, à tous les profils utilisateurs, ainsi qu'à tous leurs messages. Il a la possibilité de modifier les profils, voire de les supprimer en cas d'abus (ainsi que les messages suspects).

2 Utilisateurs

2.1 Inscription/ connexion

La page de base du site est une page qui permet de s'inscrire ou de se connecter si on est déjà inscrit.

La page d'inscription *inscription.php* ...

La page de connexion *login.php* comporte un champ pour entrer l'identifiant, et un pour le mot de passe. La validation du formulaire renvoie sur la même page, et on tente de se connecter avec les identifiants donnés par la fonction *trylogin* dans *util.php*. Cette fonction récupère dans la base de donnée tous les mots de passe et les identifiants des utilisateurs, puis compare avec ceux qui sont donnés. S'ils sont bons, les mot de passe, login et type d'utilisateur sont conservés dans la variable de session. Si les identifiants donnés ne correspondent à rien, une fenêtre d'alerte s'affiche. Sur la page *login.php*, on regarde si la variable de session est définie, et si c'est le cas on redirige vers la page correspondante : *profil.php* si c'est un utilisateur, *admin.php* si c'est un administrateur.

Lorsqu'on est connecté, un lien présent sur toutes les pages permet de se déconnecter. Il renvoie sur la page *deco.php*, qui détruit les variables de session, puis renvoie sur la page *login.php*.

2.2 Profil utilisateur

2.2.1 Les informations du profil

Un utilisateur peut partager un certain nombre d'informations sur son mur (texte ou image). Celles-ci seront présentes dans la base de données, et récupérées par la page *profil.php*. Au début de la page, on regarde qui est connecté, et s'il a le droit d'accéder à la page, ainsi qu'à quel niveau d'informations, selon s'il est sur son propre profil, sur le profil d'un ami ou d'un étranger.
Les informations publiques, accessibles à tous, sont : le nom, prénom, login, la photo de profil, la liste d'amis, l'introduction et la citation, le diplôme, la profession, l'emploi occupé, la ville de résidence et les loisirs.
Les informations privées, accessibles seulement par la personne concernée, ses amis et les administrateurs, sont les descriptions de son caractère, les langues maîtrisées, les outils connus et les compétences acquises.

Le login de la personne connectée est conservé dans la variable de session, et celui de la personne dont on regarde la page de profil est visible en get. On vérifie en premier lieu que la personne dont on cherche le profil existe par la fonction *existe* de la page *util.php*. Cette fonction réalise simplement une requête SQL qui récupère toutes les données de la base de données avec l'identifiant donné. Si cette requête ne retourne aucun résultat, c'est que l'identifiant n'existe pas.
Ensuite, les fonctions *chargerInfos* et *chargerListeAmis* permettent de récupérer dans la base de données, respectivement les informations du profil et sa liste d'amis. La première réalise cinq requêtes SQL similaires pour récupérer les informations de la table EISTI_BOOK_UTILISATEUR, puis des tables de correspondance de caractères, compétences, langues et outils. La seconde est un peu plus complexe. Elle récupère tout d'abord toutes les personnes avec lesquelles l'utilisateur a un lien d'amitié, puis vérifie que l'amitié est réciproque (car notre table amis contient toutes les demandes d'amitiés, même en sens unique). On va ensuite afficher toutes ces informations sur la page de profil.

Lorsque que la personne qui consulte le profil est ami avec le propriétaire du profil, il faut aussi afficher les publications sur le profil. On va donc utiliser la fonction *charger-Publi_profil* de *util.php*. Celle-ci récupère dans la base de données toutes les publications dont l'auteur est le propriétaire du profil consulté, et les affiche.

2.2.2 La fonction d'édition

Lorsqu'on est connecté à son profil ou en navigation générale sur le site, on peut aller éditer son profil grâce à un lien. La page *edit_ profil.php* comprend un formulaire qui affiche toutes les informations du profil et permet de tout modifier, avec un bouton de validation à la fin. Le formulaire est pré-rempli par les informations actuelles du profil grâce aux mêmes fonctions que dans la page de profil, décrites précédemment. Une fonction supplémentaire, *chargerOptions*, permet de récupérer toutes les propositions de caractères, outils etc., de façon à en faire des boutons radio.
La validation renvoie à la page *valider_ modif*, qui va traiter toutes les informations données et les modifier dans la base de données. On procède en supprimant l'identifiant de la base de données, pour le recréer avec les nouvelles informations.

Une procédure particulière est nécessaire pour modifier la photo de profil et enregistrer une photo chargée par l'utilisateur dans la base de données.

On a aussi créé une page particulière qui permet de modifier le mot de passe.

2.3 Les amis

Un utilisateur pourra ajouter des amis à sa liste d'amis ou en retirer. Il pourra regarder le profil de ses amis et d'étrangers, faire des demandes d'amis, etc. Il peut également envoyer des messages à ses amis.

Une page *amis.php* permet à l'utilisateur de gérer ses amitiés, ses demandes d'amis et de rechercher un utilisateur. Sur cette page, un bouton permet d'afficher sa liste d'amis, qui appelle en Ajax à un traitement par la page *listeAmis.php*, qui réutilise simplement les fonctions *chargerListeAmis* et *affichageAmisProfil* de la page *util.php* qui étaient utilisées dans le profil.

La deuxième partie de cette page permet d'effectuer une recherche parmi les utilisateurs du site. La recherche se fait par mots clefs sur le nom, le prénom et le login de la personne, et retourne une liste de tous les utilisateurs dont le nom, le profil ou le prénom contient un des mots clefs. Le bouton "rechercher" revoie à la page *rechercher.php*, qui effectue la requête correspondante, et affiche la liste des profils résultats. Chaque profil présente un lien vers le profil de cet utilisateur, ainsi qu'un bouton pour ajouter la personne à nos amis si nous ne sommes pas encore amis, ou pour la supprimer si elle en fait partie. Ces boutons effectuent un traitement en Ajax vers les pages *ajouterAmi.php* et *supprimerAmi.php* respectivement, par les fonctions du même nom.

La page *ajouterAmi* permet de faire une demande d'amitié à un personne qui ne nous a pas encore ajouté à ses amis, ou d'accepter une amitié dans d'autres cas. Elle est constituée d'une simple requête qui ajoute l'amitié correspondante dans la table *Amis*.

La dernière partie, à droite de la page, expose les demandes d'amitiés en cours, reçues et envoyées. Pour sélectionner ces demandes, on crée deux listes : une constituées des amitiés dans un sens (tous les gens qui sont amis avec nous) donc les demandes reçus, et l'autre comporte les demandes envoyées (tous les gens avec qui on est ami). Puis on supprime de ces deux tables celles qui sont en double sens, pour ne conserver que les demandes en cours. On affiche ensuite les deux listes, avec, un bouton pour annuler chaque demande envoyée, un bouton pour accepter et un bouton pour refuser chaque demande reçue. Ces boutons réutilisent les traitements d'*ajouterAmi* et de *supprimerAmi*.

2.4 Publications

Un onglet publication permet d'accéder aux publications de l'EISTI-BOOK et d'en créer. Dans le fichier *publi.php*, on définit l'affichage des publications de la plus récente à la plus ancienne. Pour cela nous stockons toutes nos publications dans notre base de donnée. De plus, chaque utilisateur peut uniquement visionner ses propres publications et les publications de ses amis. Malheureusement nous n'avons pas eu le temps d'implémenter les commentaires en dessous des publications et la possibilité d'aimer.

2.5 Messagerie

La messagerie de notre site comporte deux fenêtres principales. Tout d'abord nous avons créé un espace destiné à envoyer un message. Le choix du destinataire se fait au travers d'une liste déroulante parmi tous les amis de l'utilisateur. Pour pouvoir envoyer son message il doit évidemment remplir la zone dédié à l'écriture du message. Il est également possible de saisir un titre au message. Lors de l'envoi, le message sera stocké dans la base de donnée accompagné de l'id du destinataire, l'id de l'utilisateur envoyant le message et le titre du message.

La seconde fenêtre permet d'afficher les messages reçus, ils sont représentés par la date et l'heure d'envoi, le nom de l'émetteur et le titre du message. Ce dernier est un lien redirigeant vers une nouvelle fenêtre (la page *lire.php*). Sur cette page il est possible dans un premier temps de lire le message, d'y répondre, de supprimer le message et en cas d'abus, de le supprimer.
Cliquer sur le bouton supprimer aura pour effet d'envoyer une requête SQL à la base de

donnée afin d'effacer le message. Le bouton signaler lui, permet de prévenir les administrateurs d'un comportement inapproprié. Cela a alors pour effet de remplir la table SQL des messages d'un signalement associé au dit-message, ainsi on permet grâce à une nouvelle requête SQL de récupérer tous les messages comportant un signalement et de les afficher aux administrateurs.
La partie permettant de répondre au message fonctionne exactement de la même manière que la partie permettant d'en envoyer un.

3 Administrateurs

3.1 Gestions des utilisateurs

La gestion des utilisateurs passera par :
— Liste de tous les utilisateurs
— Accès aux informations personnelles de chaque utilisateur
— Modification directe des informations (description personnelle, message d'accueil...)
— Suppression d'un profil et bannissement de l'utilisateur.

Tout d'abord, on vérifie que la personne qui vient de se connecter est bien un administrateur sinon on affiche un message d'erreur. Premièrement, il y a une liste contenant le nom et le prénom de tous les utilisateurs en faisant attention d'y exclure les administrateurs non comptés comme tels générée à l'aide la fonction *getUserList*. Elle retourne un tableau qui contient le nom, le prénom et l'id de chaque utilisateur qui ne sont pas administrateurs. Puis, on peut appuyer sur plusieurs boutons différents qui appellent en Ajax des traitements sur d'autres pages php.

Le premier est la possibilité de consulter les informations personnelles de chaque utilisateur sélectionné, ce bouton est doté de la fonction onclick *execInfo* faisant référence à une fonction dans un fichier JavaScript *admin.js*. Elle envoie l'id de l'utilisateur à la page php *admin_ info.php* qui va générer toutes les informations contenues dans la table EISTI BOOK UTILISATEUR ainsi que les informations considérées comme annexes, c'est-à-dire les informations contenues dans les tables de correspondance de caractères, compétences, langues et outils.

Le deuxième bouton "Modifier informations" fait apparaître une liste des informations personnelles que l'administrateur peut changer (on exclut donc le type d'utilisateur, la photo de profil et la liste d'amis). Après avoir sélectionné l'information voulue, un champ à

compléter selon cette dernière est généré. Par exemple, pour changer le sexe, on utilise un champ de type radio contrairement au prénom où on doit compléter un champ de texte.

Cependant, il y a une particularité pour les informations qui viennent des tables de correspondance faisant référence à notre table principale EISTI BOOK UTILISATEUR. En effet, prenons l'exemple des langues, on va d'abord générer une liste déroulante contenant les langues que l'utilisateur ne parle pas et un bouton "Ajouter" avec un attribut onclick qui appelle en Ajax la page *admin_ annex.php* contenant la fonction *modifInfo* qui permet de changer une information personnelle selon son type, un id utilisateur et la nouvelle valeur. De plus, une liste qui contient les langues que l'utilisateur parle déjà est aussi générée ainsi qu'un bouton "Supprimer" afin d'enlever cette langue des langues parlées.

Le troisième bouton permet de supprimer le compte de l'utilisateur sélectionné, il consiste en un attribut onclick *supprAccount* qui demande à la page php *admin_ delete.php* une simple requête SQL. De la même façon, on peut bannir le compte d'un utilisateur avec le bouton suivant qui appelle *admin_ ban.php* en Ajax. On peut aussi retirer le bannissement de ce compte avec le bouton "Retirer le bannissement" qui fait référence à la même page php que précédemment.

Enfin, l'administrateur peut consulter les messages reçus par cet utilisateur avec l'id de son expéditeur, le titre du message, son id et la date de reçu du message. Pour cela, on fait appel à la page php *admin_ messagerie.php* qui fait une requête SQL sur la table Messages.

3.2 Gestion des messages

Pour contrôler ce que les utilisateurs s'envoient, les fonctionnalités sont les suivantes :
- Réception des signalements de messages suspects
- Accès à la messagerie complète de chaque utilisateur
- Suppression des messages

L'administrateur peut aussi consulter tous les signalements des messages envoyés par les utilisateurs. Il a accès à l'id du destinataire, celui de l'expéditeur, la date du message, le motif de signalement ainsi que le message de signalement. Pour ce faire, nous avons créé la page *signalement.php* qui, après avoir vérifié que la personne est bien un administrateur, va stocker dans une variable un tableau contenant le résultat de la requête SQL faite à l'aide de la fonction *chargeSignalement*. On affiche ensuite son contenu à l'aide d'une boucle.

Pour finir, l'administrateur peut supprimer les messages reçus par chaque utilisateur. En effet, un bouton "Supprimer message' est généré en même temps que chaque message qui appelle en Ajax la page *admin_ deletemsg.php* qui par le biais d'une requête SQL supprime le message correspondant à l'id passé en paramètre de la fonction *supprMsg*.

4 Base de données

Pour la base de données, nous avons choisi de créer la table principale EISTI BOOK UTILISATEUR qui contient les informations personnelles principales de chaque utilisateur telles que son identifiant, son login, son mot de passe, etc ... Puis, nous avons ajouté quatre tables supplémentaires de correspondance pour les langues parlées, les caractères, les langues parlées et les outils maîtrisés afin d'éviter d'avoir quatre colonnes supplémentaires dans la table principale de chaque utilisateur contenant chacune des listes de chaînes de caractère. Il y a aussi une table Amis qui fait correspondre les identifiants de deux utilisateurs différents et qui indique si le premier a choisi de bloquer le second ou pas. Ensuite, nous avons fait le choix de stocker tous les messages dans une table portant le même nom. Ainsi, nous avons pour chaque message, l'identifiant du destinataire, celui de l'expéditeur, sa date, son titre, son contenu et s'il a été signalé par le destinataire avec le motif. Pour finir, nous avons implémenté la table publication qui stocke toutes les informations publiées sur le fil d'actualité : l'auteur du message, la date et le titre.

5 Difficultés rencontrées

Nous avons rencontré certaines difficultés lors de la réalisation de notre projet. Nous avons trouvé le temps imparti à la réalisation assez court. Notamment pour terminer certaines fonctionnalités comme les blocages, les signalements, les photos de profils... Nous aurions aimé travailler un peu plus l'aspect visuel de notre EISTI-BOOK. Nous avons consacré énormément de temps à la partie administrateur afin quelle soit la plus complète possible. Nous avons essayé de faire une interface à la fois claire et intuitive de façon à optimiser l'utilisation de cette plate-forme.

6 Répartition du travail

Nous n'avons pas rencontré de grande difficulté au sein du groupe. La répartition du travail s'est faite assez naturellement en fonction des aptitudes de chacun afin d'optimiser le temps de travail. Nous avons réussi à faire le projet sur le volume horaire consacré au projet.

7 Conclusion

Ce projet EISTI-BOOK s'est avéré assez complet, nous ne pensions pas qu'il y aurait autant de paramètres à gérer notamment au niveau de la base de données pour recouper les différentes tables. Nous pensons avoir respecté le cahier des charges en grande partie. Nous aurions aimé plus de temps afin de pouvoir rajouter quelques fonctionnalités intéressantes.