École Nationale Supérieure d'Informatique pour l'Industrie et l'Entreprise

Rapport Projet Web : Construction d'un site de rencontre pour l'ENSIIE
FrIIEnds ++

Caroline Cao ; Dorian Joubaud ; Lauriane Schell ; Maxime Brouat

12 Mai 2020

Table des matières

0.1 Présentation du site

0.1.1 Le site : ce qu'il offre

L'objectif de ce site est de permettre à un IIEn de rencontrer des gens ou de trouver les informations nécessaires pour pouvoir communiquer avec une personne s'il le désire.
En étant membre de ce site l'IIEn peut, tout comme un réseau social, ajouter des amis. Une fois une amitié créée il a alors accès au profil de la personne et à ses goûts.
La recherche d'amis peut se faire sur plusieurs critères, le nom, le prénom et le pseudo mais aussi les goûts par exemple, si l'on cherche à partager quelque chose avec cette personne.

0.1.2 Pourquoi ce site

L'idée nous est venue devant la difficulté que l'on peut avoir à fonder une association ou monter un club de handball par exemple. En effet transmettre une idée à travers toute l'école et surtout trouver les personnes intéressées sont parfois compliqués, puisque tout le monde n'est pas présent et disponible à l'école au même moment.

0.1.3 Composition

Le site web est composé de :

Pages visibles	Pages de traitements	Pages CSS
1_Page_1.php		Page_1.css
informations.html		Information.css
2_Inscription.html	2_Inscription.php	Inscription.css
3_Formulaire.php	3_traitement_formulaire	Formulaire.css
4_Page_2.php		Page_2.css
Profil.php	*ajout.php* *suppression.php* *modification.php* *demande_ami.php*	Profil.css
profil_a_visite.php	*ajout.php* *suppression.php* *modification.php* *demande_ami.php*	profil_visite.css
formulaire_de_recherche.php	recherche.php	recherche.css
deconnexion.php		

Le site a été conçu autour de deux pages d'accueil.
la Page 1_*Page*_1 qui est la page sur laquelle on arrive quand on n'est pas encore connecté. Cette page permet de :

- Se connecter au site, si l'on y est déjà inscrit. L'inscription se fait via l'adresse mail et le mot de passe.
- Se créer un compte, si l'on n'est pas encore membre via la page *Formulaire.php*
- Obtenir des informations sur le site et sur l'école via la page *Information.php*

Et la page 4_*Page*_2 qui est la page d'accueil une fois connecté. Sur cette page on peut :

- Consulter et modifier son profil via *Profil.php*
- Rechercher le profil d'une autre personne enregistrée dans la base de donnée via *formulaire_recherche.php*
- Se déconnecter via *deconnexion.php*

0.2 Démarche

0.2.1 Le partage du travail

Pour le travail nous avons décidé de nous répartir les tâches comme suit.
— Caroline a créé la structure de la base de données
— Dorian s'est occupé de toute la partie CSS du site web
— Maxime s'est occupé de la barre de recherche
— Caroline et Lauriane ont réalisé le reste des pages web

0.2.2 La base de donnée

Nous cherchons à stocker des données pour permettre une connexion et des recherches et pour cela, il nous faut obtenir des informations concernant un utilisateur x.On stock le minimum de données nécessaires sur l'utilisateur c'est à dire son nom, son prénom, son pseudo, son genre, son année d'étude, sa ville de résidence, date de naissance, son mot de passe et son adresse e-mail qui nous permettront de l'identifier. Puis l'utilisateur peut fournir des informations telles que son adresse, son numéro de téléphone, sa filière ... Ainsi, nous avons une première table *utilisateur* :

Colonnes	Type	Contraintes
id	INTEGER	PRIMARY KEY
prenom	VARCHAR(255)	NOT NULL
nom	VARCHAR(255)	NOT NULL
pseudo	VARCHAR(255)	NOT NULL et UNIQUE
genre	VARCHAR(255)	NOT NULL
annee_etude	VARCHAR(3)	NOT NULL
email	VARCHAR(255)	NOT NULL et UNIQUE
ville	VARCHAR(255)	NOT NULL
filiere	VARCHAR(255)	
adresse	VARCHAR(255)	
numero_tel	VARCHAR(10)	
date_naissance	CHAR(10)	NOT NULL
mot_de_passe	VARCHAR(255)	NOT NULL
statut	VARCHAR(255)	NOT NULL

Dans cette première table, l'adresse email et le mot de passe seront les données qui permettront à un utilisateur de se connecter. Ils sont donc nécessairement non nuls et l'adresse mail unique. De plus pour la confidentialité, le mot de passe est enregistré après avoir été hashé grâce à la fonction php $password\_hash()$. Pour le genre et le statut nous avons créé des nouveaux types enum afin d'avoir des lignes avec des entrées identiques. Pour les autres entrées, la taille n'étant pas constante, on a voulu prévoir large d'ou les $VARCHAR(255)$. En revanche, le numéro de téléphone étant généralement composé de 10 chiffres on à dont définit le type sur $VARCHAR(10)$.
La décision a été faite en sorte que certaines données ne soient jamais vides. En effet cela permettait au moins d'avoir des informations à fournir pour les recherches ultérieures.

Puis pour les informations concernant les préférences et les loisirs, nous avons créé les différentes tables ci-dessous avec comme clé étrangère la clé primaire de la table *utilisateur*.

lecture

Colonnes	Type	Contraintes
id	INTEGER	NOT NULL et FOREIGN KEY
type_oeuvre	VARCHAR(255)	
genre_oeuvre	VARCHAR(255)	NOT NULL
titre	TEXT	
auteurs	TEXT	

gout

Colonnes	Type	Contraintes
id	INTEGER	NOT NULL et FOREIGN KEY
aime_lire	yesno	
aime_voyager	yesno	

film

Colonnes	Type	Contraintes
id	INTEGER	NOT NULL et FOREIGN KEY
type_film	VARCHAR(255)	
nom_film	TEXT	
realisateur_film	VARCHAR (255)	

voyage

Colonnes	Type	Contraintes
id	INTEGER	NOT NULL et FOREIGN KEY
lieu	VARCHAR(255)	
ville	VARCHAR(255)	
pays	VARCHAR(255)	
type_voyage	VARCHAR(255)	

amis

Colonnes	Type	Contraintes
id1	INTEGER	NOT NULL et FOREIGN KEY
id2	INTEGER	NOT NULL
annee_connaissance	INTEGER	

(id1, id2) PRIMARY KEY

demande_ami

Colonnes	Type	Contraintes
id1	INTEGER	NOT NULL
id2	INTEGER	NOT NULL
statut	relation_ami	

(id1, id2) PRIMARY KEY

association

Colonnes	Type	Contraintes
id	INTEGER	NOT NULL et FOREIGN KEY
nom_association	VARCHAR(255)	

sport

Colonnes	Type	Contraintes
id	INTEGER	NOT NULL et FOREIGN KEY
nom_sport	VARCHAR(255)	
club	yesno	
niveau	VARCHAR(255)	

relation

Colonnes	Type	Contraintes
id	INTEGER	PRIMARY KEY et FOREIGN KEY
relation_recherchée	rechercher	
situation_actuelle	situation	
attirance_sexuelle	orientation	

Les différents types que nous avons définis :
- "yesno", un enum ('oui', 'non')
- "relation_ami", un enum ('demande', 'refuser')
- "situation", un enum ('en couple', 'célibataire', 'marié', 'compliqué')
- "rechercher", un enum ('serieuse', 'libre')
- "orientation", un enum ('femme', 'homme', 'bi', 'aucun')

FIGURE 1 – Bulle de Connexion

FIGURE 2 – Topbar

0.2.3 Le CSS

Le CSS permet de donner vie au site en embellissant les différentes pages mais il est aussi question d'ergonomie pour comprendre simplement où est l'utilisateur et quels sont les différents choix qu'offre la page. Ainsi le style dominant du site est sobre et simple, on utilise 2 variantes de bleu pour l'embellir. Les différentes icônes que l'utilisateur peut retrouver sont sous forme de bulles qui ne surchargent pas l'écran :

La navigation étant essentielle dans un réseau social, nous avons implémenté une barre de navigation *topbar*.

```
    .topbar{

    background-color: #3d636f;
    overflow: hidden;
    border-radius: 10px;

}
.topbar a{
    float: left;
    color: #f2f2f2;
    padding: 14px 16px;
    text-decoration: none;
    font-size: 20px;
    font-family: Verdana;
    font-weight: bold;

}
.topbar a:hover, a:focus{
    background-color: #cacbcc ;
    color: white;
    box-shadow: 0 1px 0 rgba(255,255,255,0)
}

.topbar a.actif{
    background-color: #00aabb;
    color: white;
}
```

La top bar indique à l'utilisateur sur quelle rubrique du site il se trouve grâce à la classe **actif**, ainsi cette rubrique sera en bleu clair *(#00aabb)* tandis que le reste de la top bar est bleu foncé *(#3d636f)*
On ajoute un effet de style hover qui grise*#cacbcc)* la case que l'on survole.

Nous avons également utilisé certaines possibilités de la bibliothèque -webkit en suivant un tutoriel sur www3schools.

0.2.4 Connexion

Pour des raisons de confidentialités, nous avons choisi d'ajouter dans la base de données, le mot de passe hashé à l'aide de la commande php $password\_hash(mdp, PASSWORD\_DEFAULT)$. Pour récupérer ce mot de passe lors de la connexion par exemple, on utilise la commande $password\_verify(mdp, mdp\_hash)$. Ainsi, on pourra vérifier si un mot de passe entré correspond bien à celui fournit lors de l'inscription. Si ça l'est alors l'utilisateur est connecté. Sinon un message d'alerte sera afficher et l'utilisateur devrait réessayer.

0.2.5 La session

Pour la session nous avons décidé de l'ouvrir au moment de la connexion ou de l'inscription à l'aide de $session\_start()$;. Nous avons décidé de créer des variables $\$\_SESSION$ dans lesquelles nous avons mis toutes les données relatives à l'utilisateur.
Pour la déconnexion nous avons créé une page php *deconnexion.php* qui redirige vers la page d'accueil numéro 1 1_Page_1 après avoir fermé la session.

0.2.6 La création et modification du profil

Le profil se construit grâce à des formulaires successifs. Le premier formulaire sert à obtenir les informations "clefs" sur l'utilisateur, ce sont toutes les données qui seront par la suite stockées dans la table *utilisateur*.
Le traitement des données est alors plutôt simple puisqu'il s'agit d'enregistrer les données dans les tables.

S'ensuit alors un deuxième formulaire, que l'utilisateur peut choisir d'ignorer. L'objectif de ce formulaire est d'augmenter le nombres de données sur l'utilisateur. Alors qu'au départ nous souhaitions laisser l'utilisateur entrer ses associations ou situations amoureuses nous nous sommes rendus compte que pour pouvoir faire une recherche sur ces critères il fallait que chaque utilisateur aient la même valeur pour un élément *ex : célibataire, célib, celibataire* . C'est pourquoi nous avons ajouté des types à notre base de données et changements dans les formulaires pour des menus déroulants.

Le troisième formulaire est le profil en lui même de l'utilisateur. Il faut afficher dans un premier temps les données de l'utilisateur, en traitant différemment les cas où la donnée est vide. Puis permettre à l'utilisateur de modifier, ajouter et supprimer.
Cette partie à été plus délicate à traiter pour deux raisons, il fallait transmettre des données issues de la base de données afin de savoir quelles données modifier et la manière de modifier les données changeait, bien évidement, en fonction de la catégorie.
–> Pour l'affichage des données nous avons eu recours à $fetch\_num\_rows()$ pour se déplacer de tuple en tuple dans le résultat de notre requête et à $pg\_fetch\_result()$ afin d'accéder aux différentes valeurs des différentes colonnes.
–> Pour la transmissions des données nous avons créé des balises input caché dans lequel on met les données tirées des requêtes SQL à l'aide de $print\_r()$.
–> Enfin pour le traitement des données nous avons crées trois pages :

- un page *suppressions.php* qui s'occupe de supprimer ou de mettre à NULL certaines valeurs
- une page *ajout.php* qui elle s'occupe d'ajouter des données dans les tables ou de les modifier
- une page $modification.php$ qui s'occupe exclusivement de la base relation.

Afin de savoir quelle modification appliquer à quelle formulaire nous avons crée une balise $name = categorie$ dont la valeur indique la requête à effectuer dans la page *php*. Un exemple est :

Dans $Profil.php$

```php
<?php
$gout = pg_query($bdd, 'SELECT aime_voyager FROM gout WHERE id =  \''.$id.'\'');
if ($gout == false) {
    print_r(pg_last_error());
}
$nb_lignes = pg_num_rows($gout);

for ($i = 0; $i < $nb_lignes; $i ++) {
    if (pg_fetch_result($gout, $i, 0) == 'oui') {
        ?>Jaime voyager <br/><?php
    }
    if (pg_fetch_result($gout, $i, 0) == 'non'){
        ?>Je naime pas voyager <br/> <?php
    }
}
?>
<input type="button" name="ajouter" value="Modifier mes gouts" id="gout" onclick = "cacher('modifier_gout_v')"/> <br/>

<form method="post" action = "./pages_traitements/ajout.php" id = "modifier_gout_v" hidden>
    Aimez-vous voyager ?
    <select name = "aimer_voyager"  id = "aimer_voyager" >
        <option value="oui" id="oui"> Oui </option>
        <option value="non" id="non"> Non </option>
    </select>
    <input type="hidden" name= "categorie" value="aimer_voyager">
    <input type="submit" name="modifier" value="Valider" id="modifie_gout"/> <br/>
    <br/>
</form>
```

Dans *ajout.php*

```php
if ($_POST["categorie"] == 'aimer_voyager') {
    if (!empty(htmlspecialchars($_POST["aimer_voyager"])))
    {
        $query = 'UPDATE gout SET aime_voyager = \''.htmlspecialchars($_POST["aimer_voyager"]).'\' WHERE id = \''.$_SESSION["id"].'\'';
        $data = pg_query($bdd, $query); /*execution requete*/

        if ($data == false) {
            print_r(pg_last_error());
        }
        echo "<script type = 'text/javascript'> alert('Goûts de voyages modifiés !');
                                                document.location.href = '../Profil.php';</script>";
    }
    else {
        echo "<script type = 'text/javascript'> alert('Vous enregistrez des données vides ...');
                                                document.location.href = '../Profil.php';</script>";
    }
}
```

0.2.7 L'affichage d'un profil en tant que visiteur

Suite à la recherche effectuée le profil demandé s'affiche. À ce niveau là, nous avons effectué une série de test afin de savoir si, le profil demandé existe réellement et si les données en entrée sont correctes.

0.2.8 La demande en amis

La demande en amis s'est révélée compliqué à mettre en place au début. Il fallait faire la demande, puis afficher cette demande sur la page de la personne concernée et enfin lui laisser la possibilité d'accepter ou de refuser. S'il accepte il faut ensuite ajouter dans la bdd *amis*.
Nous avons donc décidé de créer une base de donnée *demande_ami* qui stock les demandes en amis et leur statuts {demande, refuser}. L'*id*1 correspond à la personne qui a été demandé en ami et l'*id*2 à la personne qui a demandé en ami.

Colonnes	Type	Contraintes
id1	INTEGER	NOT NULL
id2	INTEGER	NOT NULL
statut	relation_ami	NOT NULL

Ainsi quand un utilisateur demande en ami un autre utilisateur, sa demande est stockée dans la base de donnée avec pour statut 'demande'. Cette demande s'affiche sur le profil de l'autre utilisateur qui peut choisir d'accepter ou de refuser.

- S'il accepte alors on supprime l'entrée dans *demande_ami* et on enregistre dans la table *amis* la nouvelle amitié. Comme l'amitié est dans les deux sens on l'enregistre deux fois, mais en changeant *id*1 avec *id*2 pour la deuxième fois. Le fait de l'enregistrer dans les deux sens permet ensuite d'afficher plus facilement sur le profil de la personne ses amis.
- S'il refuse, on change le statut dans *demande_ami* pour le passer à refuser. L'utilisateur qui a été demandé ne verra alors plus la demande apparaître sur son profil.

0.2.9 Le rôle d'administrateur

Nous avons décidé de créer un rôle d'administrateur. L'administrateur de $FrIIEnds++$ à le droit de regard sur tous les profils, et, peut supprimer un profil quand il juge celui-ci inadéquat pour le site. Pour faire cela on a créé l'entrée *statut* dans la table *utilisateur* dont la valeur par défaut est 'utilisateur' mais peut être 'admin' pour l'utilisateur. Ainsi en fonction du statut du visiteur la page de *profil_a_visite* va s'afficher différemment. La différence se fait avec des boucles "if" sur le statut de l'utilisateur au sein du code.

0.2.10 La barre de recherche

Afin d'offrir à l'utilisateur un choix multiple de critère de recherche, nous avons en faite créer plusieurs barre de recherches et menus déroulants. Pour traiter ces informations du formulaire, nous nous sommes servi une fois encore de la méthode *POST* pour récolter les données, que n'avons communiquer à notre base de données dans une requête SQL en nous servant de l'extension PHP DATA OBJECTS (PDO).

Pour éviter des intrusions malveillantes dans le code source de cette page de recherche, nous avons utilisé la fonction *htmlentities* qui protège la partie html et pour se protéger des injections SQL, nous avons préparé nos variables associé à la requête grâce à la fonction *bindValue* .

```
1 $req = $pdo->prepare($sql);
2 $req->bindValue(":prenom", "%".strtolower($firstname)."%");
3 $req->bindValue(":nom", "%".strtolower($surname)."%");
4 $req->bindValue(":pseudo", "%".strtolower($pseudo)."%");
5 $req->bindValue(":asso", "%".strtolower($asso)."%");
```

(Les % servent à ce que si l'on entre dor, orian, ou bien or, on trouve tout de même Dorian. Elles donnent également à l'utilisateur la liberté de ne pas remplir tous les formulaires sans restreindre la recherche, ce que qui devrait normalement être le cas du fait de l'utilisation des AND dans notre requête SQL.)

La requête SQL fait correspondre trois tables différentes répondant ainsi, avec toutes nos autres requêtes, aux attentes du projet concernant les bases de données.

```
1  $sql = "SELECT * FROM utilisateur JOIN association ON utilisateur.id=association.id JOIN
       gout ON utilisateur.id=gout.id WHERE lower(prenom) like :prenom AND lower(nom) like :nom
       AND lower(pseudo) like :pseudo AND lower(nom_association) like :asso ";
```

0.2.11 Démarche(s) non aboutie(s)

Nous voulions dans un premier temps créer une zone de tchat pour les utilisateurs afin, qu'une fois amis ils puissent discuter. Nous n'avons cependant pas eu ce temps. L'idée était de créer une base de donnée ou les messages seraient enregistrés pour pouvoir ensuite les ressortir lorsque la conversation reprendrait. Le système fonctionnerait comme dans un formulaire avec une zone de texte. Cependant, l'affichage nous paraissait plus compliqué.