Site de rencontre L😍veLink partie3

I. 📎 Contexte

L'équipe de développement de Lovelink continue le développement du site de rencontre.
L'équipe doit coder de nouvelles fonctionnalités décrites dans le document.

II. ⚙️ Installation du projet

Si votre siteRencontre du précédent TP est abouti et sans erreur, repartez de votre TP, sinon récupérez les sources du professeur ici :

1. Lancer gitBash
2. Se positionner dans votre répertoire web : `cd "c:\wamp64\www\"`

`(Remarquez l'utilisation des guillemets dans le chemin, car GitBash fonctionne sous Linux, où les chemins utilisent des slashs / au lieu des anti-slashs \.`
`Nous pouvons donc pouvoir accéder au répertoire en tapant la commande : cd /c/wamp64/www)`

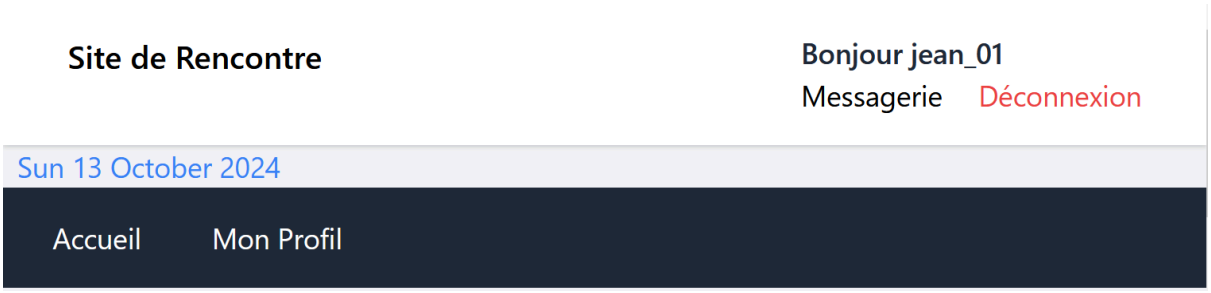
3. Lancer la commande `git clone` qui permet de récupérer les sources en local depuis un repository distant :
   `git clone https://github.com/StephanieANDRES/siteRencontre_part3.git`
4. Lancer VScode
5. Dans VSCode, faites `File>>Open Folder>>` et aller chercher le repository `c:\wamp64\www\siteRencontre_part3`
6. Lancer le site : http://localhost/siteRencontre_part3/

III. 🧰 Fonctions utilitaires

Nous souhaitons ajouter la `date du jour` ainsi que le `pseudo` de l'utilisateur connecté dans l'entête du site.
Pour cela, nous allons créer une page qui contiendra toutes les fonctions "utilitaires" du site de rencontre.

1. Créer un page `utils.php` dans `includes`
2. Créer une `fonction` utilitaire qui renvoie `la date du jour` au format JourEnLettre, N°jour du mois, mois et année.
3. Créer une `fonction` utilitaire qui renvoie le `pseudo` de l'utilisateur connecté.
4. Afficher la `date` et le `pseudo` dans l'entête du site.

Résultat attendu :



IV. 🧮 Classe statique

Étape 1 : La classe Compteur

Nous souhaitons maintenant savoir combien d'utilisateurs sont en ligne.
Pour cela, nous allons créer une classe statique avec un compteur qui sera incrémenté dès qu'un utilisateur se connectera et décrémenté dès qu'un utilisateur se déconnectera.

1. Créer une classe statique `Compteur.php` dans le répertoire `class`
2. Créer une variable statique `nbUser`
3. Créer une méthode statique `incrementer()` pour incrémenter le compteur
4. Créer une méthode statique `decrémenter()` pour décrémenter le compteur
5. Créer une méthode statique `getNbUser()` pour récupérer le compteur
6. Appeler ces méthodes lors de la connexion et de la déconnexion de l'utilisateur
7. Afficher le compteur dans le header, après la date du jour

Testez la solution. Que remarquez-vous ?

Étape 2 : La durée de vie des variables

Une **variable statique** dans une classe PHP n'est conservée que pendant la durée de l'exécution du script PHP actuel. Cela signifie que lorsqu'une page se termine (par exemple, lorsque l'on charge une nouvelle page), la variable est réinitialisée car chaque requête PHP est indépendante des autres.

Or, ce n'est pas le comportement que l'on souhaite avoir. Pour y remédier nous devons faire **persister** la variable. Voici avons plusieurs possibilités :

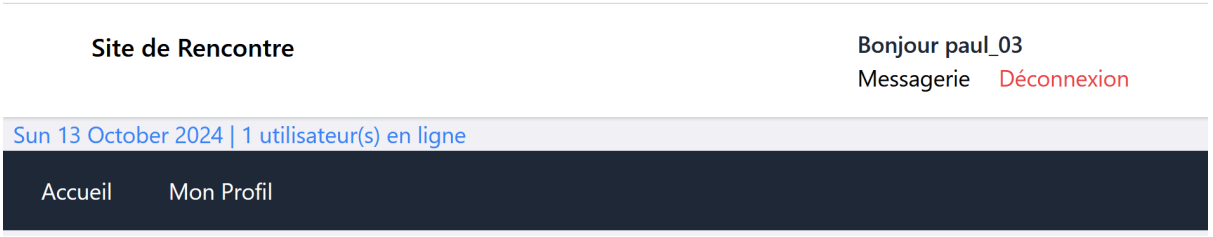
- Utiliser une session PHP
- Utiliser une base de données
- Utiliser un fichier texte

Étape 3 : La durée de vie des variables

Nous avons choisi de stocker notre valeur compteur dans un fichier texte nommé `compteur.txt`.

1. Modifier la classe `Compteur` pour qu'il ait comme propriété une variable statique `$fichier` qui contient le chemin du fichier compteur.txt
2. Modifier la méthode `incrementer()` :
    a. Tester l'existence du fichier `(file_exists(le_chemin_du_fichier))`
    b. Lire le fichier `(file_get_contents(le_chemin_du_fichier))`
    c. Incrémenter la valeur du compteur
    d. Ecrire dans le fichier `file_put_contents(le_chemin_du_fichier, la_nouvelle_valeur)`
3. En vous inspirant de la méthode `incrementer()`, modifier la méthode `decrementer()`
4. Modifier la méthode `getNbUser()` pour qu'elle récupère le compteur du fichier

Résultat attendu :



V. 🔍 Formulaire de recherche

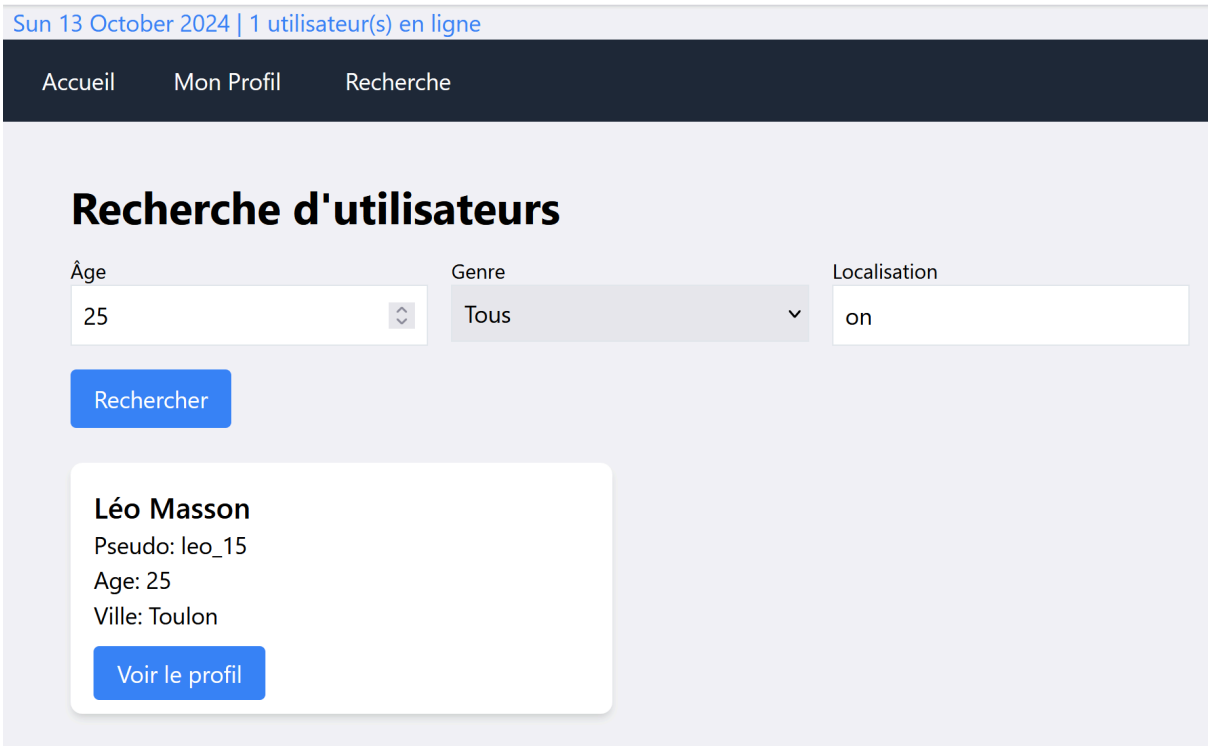
Nous allons maintenant créer un formulaire de recherche. Un utilisateur peut faire des recherches en fonction de l'âge, de la localisation ou du genre recherché.
Lorsque l'on recherche sur la localisation, nous souhaitons utiliser l'opérateur LIKE et ainsi retourner tous les résultats correspondants avec des caractères génériques.

Pour rappel, **LIKE** est un opérateur SQL utilisé dans une clause WHERE pour rechercher un motif spécifique dans une colonne.
Les caractères génériques associés à l'opérateur `LIKE` sont :

- `%` : Représente zéro, un ou plusieurs caractères. Par exemple, `LIKE 'Paris%'` trouvera toutes les entrées commençant par "Paris".
- `_` : Représente un seul caractère. Par exemple, `LIKE 'P__is'` trouvera "Paris" et "Pamis".

1. Ajouter "Recherche" dans le menu et le faire pointer vers la page `recherche.php`
2. Compléter la requête qui permet de lancer la recherche.
3. Compléter l'affichage du résultat pour faire apparaître l'âge et la ville du profil.
4. En utilisant l'opérateur Null coalescent, récupérer les valeurs soumises dans le formulaire pour les réafficher lors de l'affichage du résultat.

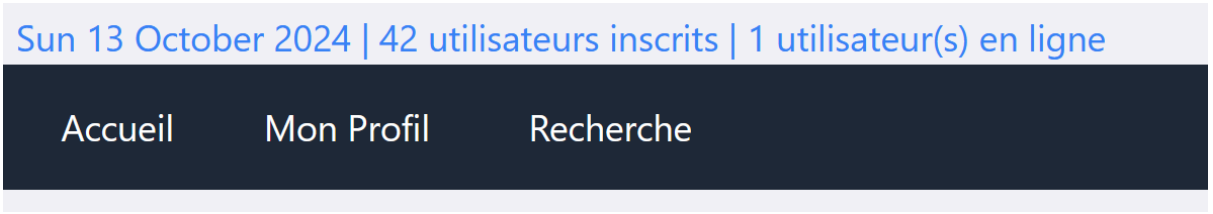
Résultat attendu :



VI. 🙄 Nombre d'inscrits / Correction sur mon profil

1. On souhaite connaître le nombre d'inscrits sur le site.
   Cette information devra être visible entre la date du jour et le nombre d'utilisateurs connectés. Réfléchir à l'endroit le plus judicieux pour coder cette fonctionnalité et la coder.
2. Modifier la page profil.php afin que le bouton "Envoyer un message" soit supprimé lorsqu'il s'agit du profil de l'utilisateur connecté.

Résultat attendu :



VII. Analysons notre code…

Nous avons désormais un code avec des classes, comme Utilisateur et Message et encore du code procédural.

Observer les méthodes de chacune des classes.

Par exemple, dans la classe Utilisateur, nous avons des propriétés qui concernent 1 seul utilisateur et des méthodes qui font des requêtes pour récupérer plusieurs utilisateurs.

Les classes ne respectent pas pleinement le principe de responsabilité unique, car elles gèrent des opérations qui ne devraient pas être de leur ressort.… De plus cela renvoie des tableaux associatifs et non des objets Utilisateur…

1. Faire des recherches et réfléchir à une nouvelle conception de notre application qui permettrait de corriger cela.
2. Faire une proposition d'amélioration et tester la faisabilité de votre solution.

VIII. 🏆 Pour aller plus loin… les "matches"

Implémenter la fonctionnalité suivante :

1. Créer la classe `Correspondance` qui correspond aux matches entre 2 utilisateurs.
2. Créer la page "`matches.php`".
3. Créer un lien "Match" dans le menu qui pointe sur `matches.php`
4. La page matches.php ressemble à la page `profil.php`, seuls s'affichent les profils qui matchent avec celui de l'utilisateur.
    a. Pour matcher, il faut que les âges correspondent entre les 2 utilisateurs à +/- 3 ans.
    b. Pour matcher, il faut que les genres correspondent au genre recherché
    c. Pour matcher, il faut qu'il y ait au moins 1 mot de plus de 5 caractères qui corresponde dans les biographies des 2 utilisateurs.
5. Une fois un match trouvé, la correspondance est enregistrée en base de données.