Site de rencontre L😍veLink partie4

I. 📎 Contexte

Les équipes de développement du site de rencontre ont acquis de bonnes compétences en programmation objet.

Dans un souci constant d'amélioration et de qualité du code, elles font appel à un architecte technique pour les conseiller.

Celui-ci recommande de travailler avec une architecture en couches afin de moduler davantage l'application, de simplifier la maintenance et la compréhension du code.

Il vous guide pour effectuer cette transformation.

II. ⚙️ Installation du projet

1. Lancer gitBash
2. Se positionner dans votre répertoire web : `cd "c:\wamp64\www\"`
3. Lancer la commande :
   `git clone https://github.com/StephanieANDRES/siteRencontre_part4.git`
4. Lancer Wamp, lancer VScode
5. Dans VSCode, faites `File>>Open Folder>>` et aller chercher le repository `c:\wamp64\www\siteRencontre_part4`
6. Lancer le site : `http://localhost/siteRencontre_part4/`

III. L'architecture logicielle en couche

L'architecture en couche permet de séparer la logique métier (Business Logic), l'accès aux données (Data Access) et les objets de domaine (Domain Objects). Cela rend le code plus lisible et plus facile à maintenir.

Les Beans

Dans cette architecture, les Beans représentent les données d'une entité (table) sous forme d'objets simples avec des propriétés et des méthodes d'accès (getters et setters).

Les DAO

Les DAO (**Data Access Object**) sont une couche supplémentaire qui sert de médiateur entre la logique métier et la base de données. Ils sont utilisés pour organiser l'accès aux données en encapsulant les requêtes SQL dans une classe dédiée.

Toutes les opérations de requêtes SQL sont contenues dans la classe DAO, ce qui rend le code plus propre et la maintenance plus facile.

Les Managers

Les managers coordonnent la logique métier en utilisant le DAO et les Beans. Le Manager contient des méthodes qui appellent les DAO pour effectuer des actions et appliquer la logique spécifique.

IV. Création de l'arborescence

1. Créer l'arborescence suivante dans votre projet pour organiser les différentes couches :

/siteRencontre

```
├── /beans          # Contiendra les classes représentant les entités (Utilisateur...)
├── /dao            # Contiendra les classes DAO pour accéder aux données
├── /managers       # Contiendra les classes de logique métier
├── /views          # Contiendra les fichiers d'affichage HTML/PHP
├── db.php          # Fichier de configuration de la base de données
└── index.php       # Point d'entrée principal de l'application
```

2. Déplacer les fichiers et répertoires HTML/PHP d'affichage dans le dossier `views` (renommer la page `index.php` en `accueil.php`).
3. Déplacer le fichier `db.php` à la racine du site
4. Créer un fichier `index.php` à la racine du site.

V. Création du bean Utilisateur

5. Dans le dossier `/beans`, créer le fichier `Utilisateur.php`

La classe `Utilisateur` ne doit contenir que les propriétés correspondant aux colonnes de la table `utilisateurs`, ainsi que des `getters` et `setters` pour chaque propriété.

6. Les getters et le setters peuvent être générés automatiquement avec une extension. Moi j'ai utilisé celle-ci :

7. Créer la méthode `hydrate(array)` et appeler cette méthode dans le `constructeur`.

Voir le cours :
https://andres-sio.lycee-ozenne.fr/CoursDevWeb/?page=05_Classes.html#hydratation_des_données

VI. Création du DAO Utilisateur

8. Dans le dossier `/dao`, créer le fichier `UtilisateurDAO.php`.

Cette classe doit gérer les interactions avec la **base de données** pour la table `utilisateurs`.

- Faire un require sur le fichier 'bean/UtilisateurBean.php'
- Créer la propriété privée `$pdo`
- Créer le constructeur pour alimenter `$pdo`
- Créer les méthodes du `CRUD` (`Create, Read, Update, Delete`) utiles pour notre projet dans cette classe :

a. `Méthode createUser(array user) : int | null`

Cette méthode est appelée au moment de l'inscription. Elle prend en paramètre un tableau associatif contenant toutes les propriétés de la classe Utilisateur sous forme de clé=>valeur.

- Elle permet d'insérer le nouvel enregistrement utilisateur en BDD.
- Elle bind les champs pour éviter les injections SQL.
- Elle retourne le dernier id généré : `return $this->pdo->lastInsertId();`

b. `Méthode findByEmail(String email) : Utilisateur | null`

Cette méthode permet de récupérer l'utilisateur avec l'email renseigné lors de la connexion de l'utilisateur. Elle prend en paramètre un String contenant l'email saisi.

- Elle permet de récupérer l'objet utilisateur du mail renseigné.
- Elle bind l'email pour éviter les injections SQL.

Si le mail a été trouvé, elle retourne l'objet Utilisateur, sinon elle renvoie `null` :

```
...
$stmt->execute(...);
$data = $stmt->fetch(PDO::FETCH_ASSOC);
return $data ? new Utilisateur($data) : null;
```

c. `Méthode findById(int id) : Utilisateur | null`

Cette méthode permet d'afficher les informations d'un utilisateur particulier. Elle prend en paramètre un entier contenant l'id de l'utilisateur recherché.

- Si l'id a été trouvé, elle renvoie l'objet Utilisateur sinon elle renvoie `null`.

d. `Méthode findAll() : array`

Cette méthode permet de récupérer tous les utilisateurs de la base de données sous forme de tableau d'objets Utilisateur.

- Pour chaque élément renvoyé par la base de données, on crée un utilisateur qu'on place dans un tableau.
- La méthode renvoie un tableau d'Utilisateur.

```
...
 $data = $stmt->fetchAll();
        foreach($data as $user){
            $users[]=new Utilisateur($user);
        }
        return $users;
```

VII. Création du Manager Utilisateur

9. Dans le dossier `/managers`, créer le fichier `UtilisateurManager.php`.

Le Manager contiendra uniquement la logique métier liée aux utilisateurs (connexion, inscription, etc.), en utilisant `UtilisateurDAO` pour accéder aux données.

- Faire un `require` sur le fichier 'dao/UtilisateurDAO.php'
- Créer une propriété privée `UtilisateurDao`
- Créer un `constructeur` permettant d'alimenter notre UtilisateurDao. (Note : Pour créer un objet UtilisateurDao, vous aurez besoin de lui passer le $pdo).

a. `Méthode connexion($email, $password)`

Cette méthode va permettre de gérer la connexion au site. Elle attend en paramètre un email et un mot de passe.

- Elle fait appel au `DAO` pour récupérer `l'Utilisateur` par son email
- Si `l'Utilisateur` a été récupéré, elle vérifie le hash du mot de passe
- Si le hash du mot de passe est identique à celui de `l'Utilistateur`, alors elle crée la variable de session `"user_id"`.
- Elle retourne `true` si l'utilisateur peut être loggué, sinon elle retourne `false`.

b. `Méthode inscription($nom, $prenom, $email, $password…)`

Cette méthode va permettre de gérer l'inscription sur le site. Elle attend en paramètre toutes les informations utiles à la création de l'utilisateur (un tableau associatif, une liste de paramètres : comme vous le souhaitez).

- Elle fait appel au `DAO` pour créer l'utilisateur
- Elle retourne `true` si l'utilisateur a été créé, sinon elle retourne `false`.

c. `Méthode findAll()`

Cette méthode va permettre de récupérer tous les utilisateurs en base de données. Elle n'attend aucun paramètre, mais elle renvoie un tableau d'Utilisateurs.

VIII. La page d'index

La page d'index sera la seule page accessible directement depuis notre site. C'est elle qui inclura telle ou telle page selon la navigation de l'utilisateur.

Cette page doit contenir :

- le seul `session_start()` du site
- un include de `db.php`
- un include du `manager` des utilisateurs.

Selon un paramètre `GET["action"]`, la page incluera :

- la page de connexion
- la page d'inscription,
- la page d'accueil,
- la page profil,
- la page de messagerie,
- la page de déconnexion,
- la page connexion est celle par défaut.

10. Implémenter la page `index.php` en fonction des indications fournies ci-dessus.

IX. Adaptation des pages

11. Modifier les pages `menu.php` et `header.php` afin que les liens soient corrects. *Par exemple :*
    a. `messagerie.php` devient `index.php?action=messagerie`
    b. `logout.php` devient `index.php?action=deconnexion`
    c. On peut aussi écrire le lien avec son chemin relatif : `inscription.php` devient `?action=inscription`

12. Modifier la page `connexion.php` afin qu'elle utilise le manager pour logguer l'utilisateur.
    a. Il ne faut pas inclure le fichier manager, celui-ci est déjà inclus plus haut (dans l'index.php)

13. Modifier la page `accueil.php` afin d'utiliser le manager pour lister tous les utilisateurs.
    a. Si l'utilisateur n'est pas connecté, on le renvoie sur la page index.php
    b. On récupère tous les utilisateurs de la base
    c. On modifie l'affichage car nous ne travaillons plus sur un tableau associatif mais des objets Utilisateurs (les propriétés sont accessibles avec leurs getters).
    d. On n'oublie pas de modifier le chemin du lien "envoyer un message". Nous aurons besoin de passer plusieurs paramètres. Les paramètres sont séparés par un &.

14. Modifier la page `profil.php` afin d'utiliser le manager pour afficher un utilisateur depuis son `id`… Il va probablement vous manquer une méthode dans le `DAO`. A vous de le compléter !
    a. Si l'utilisateur n'est pas connecté, on le renvoie sur la page index.
    b. Si nous n'avons pas l'id du profil à afficher, on renvoie l'utilisateur sur l'accueil.

15. Modifier les pages restantes pour que votre site soit 100% opérationnel suite à ce changement d'architecture.

X. Correction

Voir le résultat : http://andres-sio.lycee-ozenne.fr/siteRencontre_vp4/

Fichier bean/Utilisateur.php

```php
<?php
class Utilisateur
{
    private $id;
    private $nom;
    private $prenom;
    private $pseudo;
    private $email;
    private $mot_de_passe;
    private $date_naissance;
    private $genre;
    private $preference;
    private $bio;
    private $localisation;
    private $dateInscription;

    public function __construct(array $data)
    {
        $this->hydrate($data);
    }

    private function hydrate(array $data)
    {
        foreach ($data as $key => $value) {
            // On récupère le nom du setter correspondant à l'attribut
            $method = 'set' . ucfirst($key);

            // Si le setter correspondant existe.
            if (method_exists($this, $method)) {
                // On appelle le setter
                $this->$method($value);
            }
        }
    }


    public function getId()
    {
        return $this->id;
    }
```

```php
    /**
     * Set the value of id
     */
    public function setId($id): self
    {
        $this->id = $id;

        return $this;
    }

    /**
     * Get the value of nom
     */
    public function getNom()
    {
        return $this->nom;
    }

    /**
     * Set the value of nom
     */
    public function setNom($nom): self
    {
        $this->nom = $nom;

        return $this;
    }

    /**
     * Get the value of prenom
     */
    public function getPrenom()
    {
        return $this->prenom;
    }

    /**
     * Set the value of prenom
     */
    public function setPrenom($prenom): self
    {
        $this->prenom = $prenom;

        return $this;
    }

    /**
     * Get the value of pseudo
```

```php
     */
    public function getPseudo()
    {
        return $this->pseudo;
    }

    /**
     * Set the value of pseudo
     */
    public function setPseudo($pseudo): self
    {
        $this->pseudo = $pseudo;

        return $this;
    }

    /**
     * Get the value of email
     */
    public function getEmail()
    {
        return $this->email;
    }

    /**
     * Set the value of email
     */
    public function setEmail($email): self
    {
        $this->email = $email;

        return $this;
    }

    /**
     * Get the value of motDePasse
     */
    public function getMotDePasse()
    {
        return $this->mot_de_passe;
    }

    /**
     * Set the value of motDePasse
     */
    public function setMot_de_passe($mot_de_passe): self
    {
        $this->mot_de_passe = $mot_de_passe;
```

```
        return $this;
    }

    /**
     * Get the value of date_naissance
     */
    public function getDateNaissance()
    {
        return $this->date_naissance;
    }

    /**
     * Set the value of date_naissance
     */
    public function setDateNaissance($date_naissance): self
    {
        $this->date_naissance = $date_naissance;

        return $this;
    }

    /**
     * Get the value of genre
     */
    public function getGenre()
    {
        return $this->genre;
    }

    /**
     * Set the value of genre
     */
    public function setGenre($genre): self
    {
        $this->genre = $genre;

        return $this;
    }

    /**
     * Get the value of preference
     */
    public function getPreference()
    {
        return $this->preference;
    }
```

```php
    /**
     * Set the value of preference
     */
    public function setPreference($preference): self
    {
        $this->preference = $preference;

        return $this;
    }

    /**
     * Get the value of bio
     */
    public function getBio()
    {
        return $this->bio;
    }

    /**
     * Set the value of bio
     */
    public function setBio($bio): self
    {
        $this->bio = $bio;

        return $this;
    }

    /**
     * Get the value of localisation
     */
    public function getLocalisation()
    {
        return $this->localisation;
    }

    /**
     * Set the value of localisation
     */
    public function setLocalisation($localisation): self
    {
        $this->localisation = $localisation;

        return $this;
    }

    /**
     * Get the value of dateInscription
```

```
     */
    public function getDateInscription()
    {
        return $this->dateInscription;
    }

    /**
     * Set the value of dateInscription
     */
    public function setDateInscription($dateInscription): self
    {
        $this->dateInscription = $dateInscription;

        return $this;
    }
}
```

Fichier dao/UtilisateurDao.php

```
<?php
require_once  "beans/Utilisateur.php";

class UtilisateurDao
{
    private $pdo;

    public function __construct($pdo) {
        $this->pdo = $pdo;
    }

    public function createUser(array $utilisateur)
    {
       $stmt = $this->pdo->prepare("INSERT INTO utilisateurs
        (nom, prenom, pseudo, email, mot_de_passe, date_naissance, genre,
preference, bio, localisation)
        VALUES (:nom, :prenom, :pseudo, :email, :mot_de_passe,
:date_naissance, :genre,
        :preference, :bio, :localisation)");

        $mdp=password_hash($utilisateur["mot_de_passe"], PASSWORD_DEFAULT);

        $stmt->bindParam(':nom', $utilisateur["nom"], PDO::PARAM_STR);
        $stmt->bindParam(':prenom', $utilisateur["prenom"], PDO::PARAM_STR);
        $stmt->bindParam(':pseudo', $utilisateur["pseudo"], PDO::PARAM_STR);
```

```
        $stmt->bindParam(':email', $utilisateur["email"], PDO::PARAM_STR);
        $stmt->bindParam(':mot_de_passe', $mdp, PDO::PARAM_STR);
        $stmt->bindParam(':date_naissance', $utilisateur["date_naissance"],
PDO::PARAM_STR);
        $stmt->bindParam(':genre', $utilisateur["genre"], PDO::PARAM_STR);
        $stmt->bindParam(':preference', $utilisateur["preference"],
PDO::PARAM_STR);
        $stmt->bindParam(':bio', $utilisateur["bio"], PDO::PARAM_STR);
        $stmt->bindParam(':localisation', $utilisateur["localisation"],
PDO::PARAM_STR);
        $stmt->execute();


        //Renvoie le dernier ID inséré
        return $this->pdo->lastInsertId();
    }

    public function findByEmail($email)
    {
        $stmt = $this->pdo->prepare("SELECT * FROM utilisateurs WHERE email
= :email");
        $stmt->bindParam(':email', $email, PDO::PARAM_STR);
        $stmt->execute();
        $data = $stmt->fetch(PDO::FETCH_ASSOC);

        return $data ? new Utilisateur($data) : null;
    }

    public function findById($id)
    {
        $stmt = $this->pdo->prepare("SELECT * FROM utilisateurs WHERE id =
:id");
        $stmt->bindParam(':id', $id, PDO::PARAM_INT);
        $stmt->execute();
        $data = $stmt->fetch(PDO::FETCH_ASSOC);
        return $data ? new Utilisateur($data) : null;
    }

    public function findAll()
    {
        $stmt = $this->pdo->prepare("SELECT * FROM utilisateurs");
        $stmt->execute();
        $data = $stmt->fetchAll();
        foreach($data as $user){
            $users[]=new Utilisateur($user);
        }
        return $users;
    }
```

```
}
?>
```

Fichier managers/UtilisateurManager.php

```
<?
require_once 'dao/UtilisateurDAO.php';

class UtilisateurManager {

    private $utilisateurDAO;

    public function __construct($pdo) {
        $this->utilisateurDAO = new UtilisateurDAO($pdo);
    }

    public function connexion($email, $password) {
        $utilisateur = $this->utilisateurDAO->findByEmail($email);
        if ($utilisateur && password_verify($password,
$utilisateur->getMotDePasse())) {
            $_SESSION['user_id'] = $utilisateur->getId();
            return true;
        }
        return false;
    }

    public function inscription($array) {
        return $this->utilisateurDAO->createUser($array);
    }

    public function findAll() {
        return $this->utilisateurDAO->findAll();
    }

    public function findById($id) {
        return $this->utilisateurDAO->findById($id);
    }


}
?>
```

Fichier index.php

```
<?php
```

```php
session_start();
require_once 'db.php';
require_once 'managers/UtilisateurManager.php';

$action = $_GET['action'] ?? 'connexion';

switch ($action) {
    case 'accueil':
        include 'views/accueil.php';
        break;
    case 'connexion':
        include 'views/connexion.php';
        break;

    case 'inscription':
        include 'views/inscription.php';
        break;

    case 'profil':
        include 'views/profil.php';
        break;

    case 'messagerie':
        include 'views/messagerie.php';
        break;

    case 'deconnexion':
        include 'views/logout.php';
        break;

    default:
        include 'views/connexion.php';
        break;
}
?>
```

Fichier views/connexion.php

```php
<?php

$utilisateurManager = new UtilisateurManager($pdo);

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $erreur = "";
    $email = filter_input(INPUT_POST, 'email', FILTER_VALIDATE_EMAIL);
    $mot_de_passe = $_POST['mot_de_passe'];
```

```
    if ($email === false) {
        $erreur = "Adresse e-mail invalide!";
    } elseif (empty($mot_de_passe)) {
        $erreur = "Le mot de passe ne peut pas être vide!";
    } else {

        $user = $utilisateurManager->connexion($email, $mot_de_passe);
        if($user){
            header('Location: index.php?action=accueil');
            exit();
        }else{
            $erreur = "Identifiants incorrects";
        }
    }
}
?>

<?php include 'includes/header.php'; ?>

<div class="container mx-auto mt-10 max-w-lg">
    <h2 class="text-2xl font-bold text-center">Connexion</h2>
    <form action="index.php?action=connexion" method="post" class="mt-4
bg-white p-6 rounded-lg shadow-md">
        <input type="email" name="email" placeholder="Email" class="border
p-2 w-full mb-4" required>
        <input type="password" name="mot_de_passe" placeholder="Mot de
passe" class="border p-2 w-full mb-4" required>
        <button type="submit" class="bg-blue-500 text-white py-2 px-4
rounded hover:bg-blue-700 w-full">Se connecter</button>
        <?php if (isset($erreur)): ?>
            <p class="text-red-500 mt-2"><?= htmlspecialchars($erreur)
?></p> <!-- Utilisez htmlspecialchars pour éviter les injections XSS -->
        <?php endif; ?>
    </form>
</div>

<?php include 'includes/footer.php'; ?>
```

Fichier views/accueil.php

```
<?php
// Vérifie si l'utilisateur est connecté
if (!isset($_SESSION['user_id'])) {
    header("Location: index.php");
```

```
    exit();
}

// Récupère tous les utilisateurs
$utilisateurManager = new UtilisateurManager($pdo);
$utilisateurs = $utilisateurManager->findAll();
?>

<?php include 'includes/header.php'; ?>

<div class="container mx-auto mt-10">
    <h1 class="text-3xl font-bold">Utilisateurs</h1>

    <div class="mt-6 grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">
        <?php foreach ($utilisateurs as $user): ?>
            <div class="bg-white p-4 rounded-lg shadow-md flex flex-col">
                <h2 class="text-xl font-semibold"><?=
htmlspecialchars($user->getPrenom() . ' ' . $user->getNom()) ?></h2>
                <p class="mb-4">Pseudo: <?=
htmlspecialchars($user->getPseudo()) ?></p>
                <div class="mt-auto">
                <a href="index.php?action=profil&id=<?= $user->getId() ?>"
class="bg-blue-500 text-white py-2 px-4 rounded hover:bg-blue-700 w-full
text-center">Voir le profil</a>
                </div>
            </div>
        <?php endforeach; ?>
    </div>
</div>

<?php include 'includes/footer.php'; ?>
```

Fichier views/messagerie.php

```
<?php
if (!isset($_SESSION['user_id'])) {
    header('Location: index.php');
    exit;
}

$user_id = $_SESSION['user_id'];
$destinataire_id = null;

if(isset($_GET["destinataire_id"])){
```

```php
    $destinataire_id = $_GET["destinataire_id"];
}

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $destinataire_id = $_POST['destinataire_id'];
    $message = $_POST['message'];
}

//ListeDéroulante
$utilisateurManager = new UtilisateurManager($pdo);
$utilisateurs = $utilisateurManager->findAll();

?>

<?php include 'includes/header.php'; ?>

<div class="container mx-auto mt-10">
    <h2 class="text-2xl font-bold">Messagerie</h2>

    <form action="?action=messagerie" method="post" class="mt-4">
        <select name="destinataire_id" class="border p-2 w-full mb-4">
            <!-- Liste des utilisateurs avec lesquels échanger des messages
-->
            <?php
            foreach($utilisateurs as $user){
                if($destinataire_id==$user->getId()){
                    echo "<option selected=\"selected\"
value='{$user->getId()}'>{$user->getPseudo()}</option>";
                }else{
                    echo "<option
value='{$user->getId()}'>{$user->getPseudo()}</option>";
                }
            }
            ?>
        </select>
        <textarea name="message" placeholder="Votre message" class="border
p-2 w-full mb-4"></textarea>
        <button type="submit" class="bg-blue-500 text-white py-2 px-4
rounded hover:bg-blue-700 w-full">Envoyer</button>
    </form>

    <div class="mt-6 bg-white p-6 rounded-lg shadow-md">
        <h3 class="text-xl font-semibold mb-4">Vos messages</h3>

    </div>
</div>

<?php include 'includes/footer.php'; ?>
```

Fichier views/inscription.php

```php
<?php

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $nom = $_POST['nom'];
    $prenom = $_POST['prenom'];
    $pseudo = $_POST['pseudo'];
    $email = $_POST['email'];
    $mot_de_passe = password_hash($_POST['mot_de_passe'], PASSWORD_DEFAULT);
    $date_naissance = $_POST['date_naissance'];
    $genre = $_POST['genre'];
    $preference = $_POST['preference'];
    $bio = $_POST['bio'];
    $localisation = $_POST['localisation'];

    $utilisateurManager = new UtilisateurManager($pdo);
    $utilisateur = [
        "nom"=>$nom,
        "prenom"=>$prenom,
        "pseudo"=>$pseudo,
        "email"=>$email,
        "mot_de_passe"=>$mot_de_passe,
        "date_naissance"=>$date_naissance,
        "genre"=>$genre,
        "preference"=>$preference,
        "bio"=>$bio,
        "localisation"=>$localisation
    ];

    if ($utilisateurManager->inscription($utilisateur)) {
        header('Location: index.php');
    } else {
        $erreur = "Une erreur est survenue lors de l'inscription";
    }
}
?>

<?php include 'includes/header.php'; ?>

<div class="container mx-auto mt-10 max-w-lg">
    <?php if (isset($erreur)){ ?>
        <p class="text-red-500 mt-2"><?= htmlspecialchars($erreur) ?></p>
```

```
<!-- Utilisez htmlspecialchars pour éviter les injections XSS -->
    <?php } ?>
    <h2 class="text-2xl font-bold text-center">Inscription</h2>
    <form action="?action=inscription" method="post" class="mt-4 bg-white
p-6 rounded-lg shadow-md">
        <input type="text" name="nom" placeholder="Nom" class="border p-2
w-full mb-4" required>
        <input type="text" name="prenom" placeholder="Prénom" class="border
p-2 w-full mb-4" required>
        <input type="text" name="pseudo" placeholder="Pseudo" class="border
p-2 w-full mb-4" required>
        <input type="email" name="email" placeholder="Email" class="border
p-2 w-full mb-4" required>
        <input type="password" name="mot_de_passe" placeholder="Mot de
passe" class="border p-2 w-full mb-4" required>
        Date de naissance : <input type="date" name="date_naissance"
class="border p-2 w-full mb-4" required>
        Vous êtes : <select name="genre" class="border p-2 w-full mb-4">
            <option value="Homme">Homme</option>
            <option value="Femme">Femme</option>
            <option value="Autre">Autre</option>
        </select>
        Vous recherchez  : <select name="preference" class="border p-2
w-full mb-4">
            <option value="Homme">Homme</option>
            <option value="Femme">Femme</option>
            <option value="Tous">Tous</option>
        </select>
        <textarea name="bio" placeholder="Votre bio" class="border p-2
w-full mb-4"></textarea>
        <input type="text" name="localisation" placeholder="Localisation"
class="border p-2 w-full mb-4">
        <button type="submit" class="bg-blue-500 text-white py-2 px-4
rounded hover:bg-blue-700 w-full">S'inscrire</button>
    </form>
</div>

<?php include 'includes/footer.php'; ?>
```