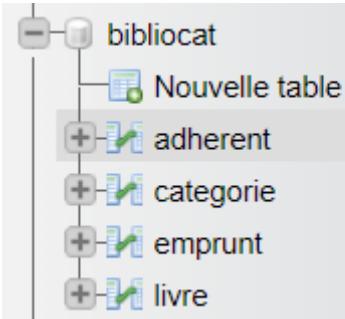


Compte rendu du site BiblioCAT

Dans PHPMyAdmin on a créé une base de données nommé bibliocat, avec 4 tables :



Il faut importer le **fichier bibliocat.sql** dans le **dossier sql du projet** (bibliocat.sql), qui est le script de la base de données puis ses données ou directement le fichier rendu sur moodle bibliocat.sql.

Identifiants importants pour se connecter en tant que compte administrateur :

Adresse mail compte Admin : compteAdmin@email.com

Mot de passe compte Admin : joliverie

Identifiants importants pour la vue adherent (simple utilisateur) :

Adresse mail adherent : loric.worms@email.com

Mot de passe adherent : lworms

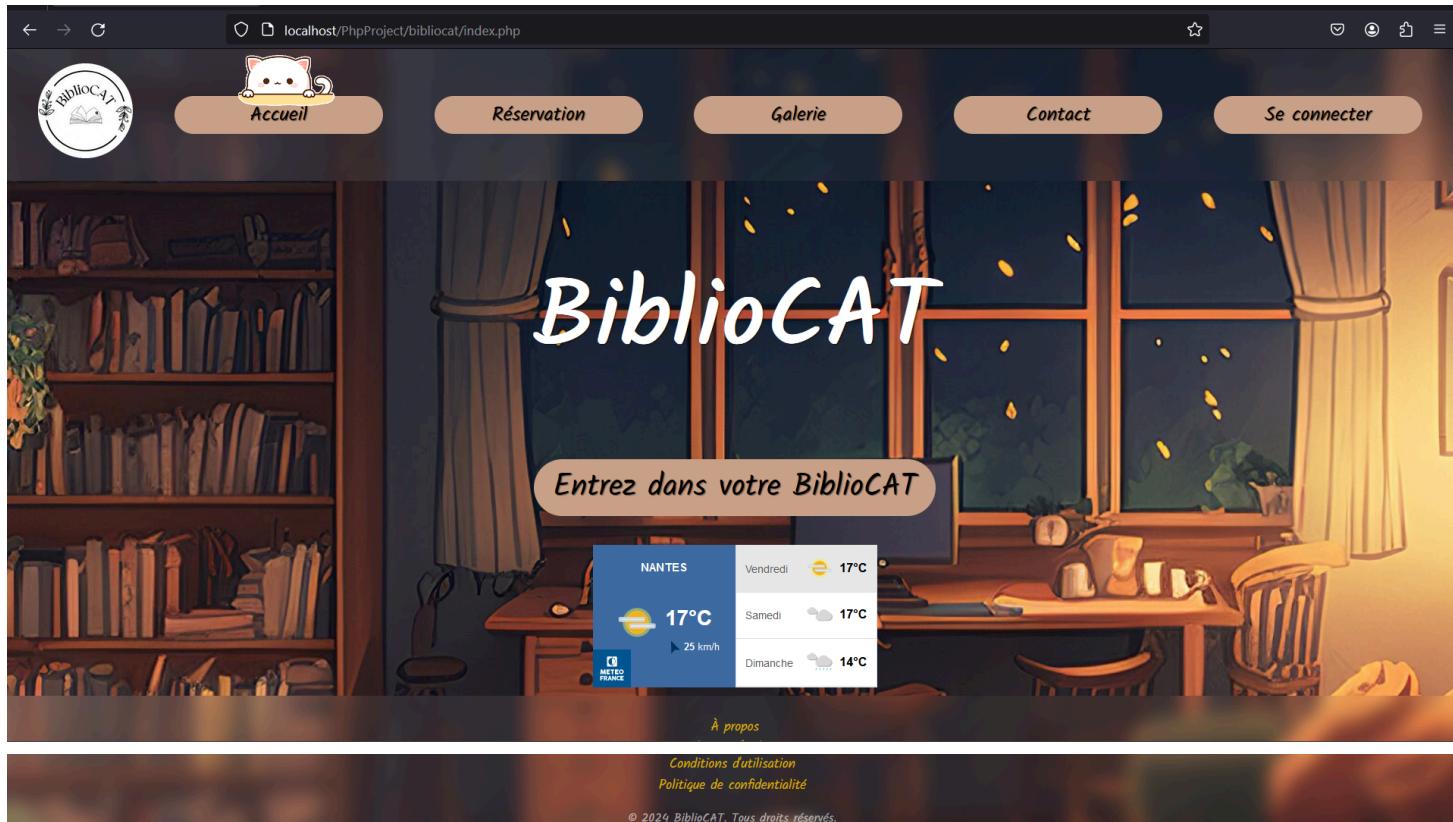
Autresse adresse mail adherent : john.doe@email.com

Mot de passe adherent 2 : jdoe

Explication de toutes les pages de notre site :

Page index.php :

La page index.php ressemble à ceci :



On commence d'abord par créer une barre de navigation (navbar) :

```

<div class="wrapper">
  <header>
    <div class="menu">
      <a href="index.php">
        
      </a>
      <a href="index.php" class="chat-link">
        <button class="menuBloc">Accueil </button>
        
      </a>
      <a href="reservation.php" onclick=<?php echo isset($_SESSION['user_id']) ? '' : 'alert("Vous devez vous connecter avant de réserver"); ?>" class="chat-link">
        <button class="menuBloc">Réservation</button>
      </a>
      <a href="galerie.php">
        <button class="menuBloc">Galerie</button>
      </a>
      <a href="contact.php">
        <button class="menuBloc">Contact</button>
      </a>
      <a href="connexionUtilisateur.php">
        <button class="menuBloc">Se connecter</button>
      </a>
      <!-- Afficher le bouton "Back Office" uniquement si l'utilisateur est connecté en tant que "compteAdmin" -->
      <?php if ($isCompteAdmin) : ?>
        <a href="back-office.php"><button class="menuBloc">Back Office</button></a>
      <?php endif; ?>
    </div>
  </header>

```

Ce code représente la structure d'un menu de navigation de notre site web. Il contient des liens vers différentes pages y compris une vérification pour afficher un lien supplémentaire vers le Back Office, uniquement accessible par l'administrateur. (donc est connecté avec un identifiant compteAdmin)

Cas particulier d'un de nos boutons si on clique sur le bouton réservation on doit être connecté avec un compte pour pouvoir réserver.

Après nous avons ajouté un titre dans la page dans une balise h1, suivi du widget Google Maps qu'on a récupéré sur météo France comme demandé dans les missions.

Puis on a créé un simple footer qui apparaît ensuite sur chaque bas de page, qui apporte un lien vers les réseaux sociaux du site, et diverses informations qu'on peut retrouver sur des sites normaux.

```

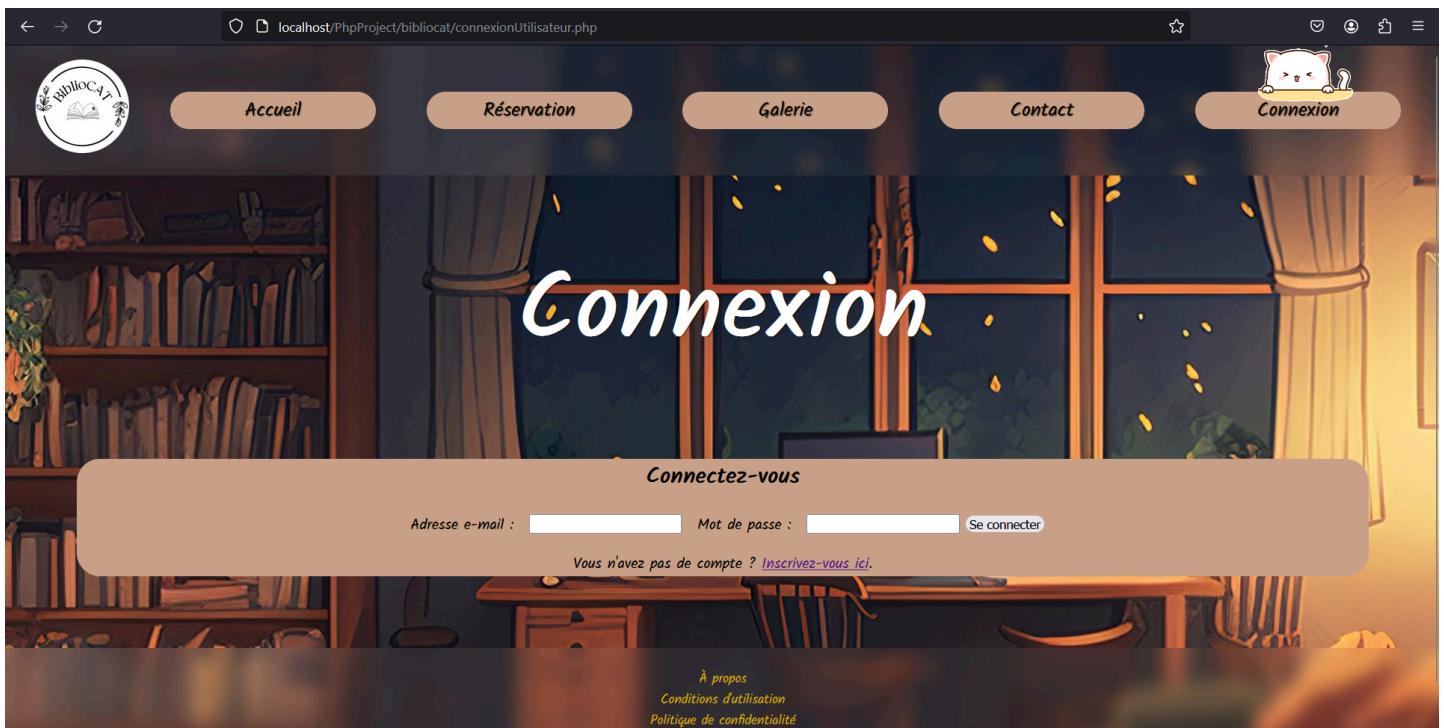
<h1>BiblioCAT</h1>
<div class="buttonAccueil">
  <a href="galerie.php">
    <button id="button-acceuil">Entrez dans votre BiblioCAT</button>
  </a>
</div>

<div id="weather-widget">
  <!-- Ajoutez ici le code du widget Météo France -->
  <iframe id="widget_autocomplete_preview" width="300" height="150" frameborder="0" src="https://meteofrance.com/widget/prevision/441090"> </iframe>
</div>

<!--Footer -->
<div class="footer-clean">
  <footer>
    <div class="container">
      <div class="item social">
        <a href="#" target="_blank"><i class="fab fa-facebook"></i></a>
        <a href="#" target="_blank"><i class="fab fa-twitter"></i></a>
        <!-- Ajoutez d'autres icônes de réseaux sociaux selon vos besoins -->
      </div>
      <ul>
        <li><a href="#">À propos</a></li>
        <li><a href="#">Conditions d'utilisation</a></li>
        <li><a href="#">Politique de confidentialité</a></li>
      </ul>
      <div class="copyright">© 2024 BiblioCAT. Tous droits réservés.</div>
    </div>
  </footer>
</div>

```

Page connexion.php :



Le header, la connexion à la base de données ne changent pas comme le footer. Voici la partie html :

```

<header>
    <h1>Connexion</h1>
</header>

<div class="login-form">
    <h2>Connectez-vous</h2>
    <!-- Affichage des messages de session -->
    <?php if (isset($_SESSION['message'])) : ?>
        <div class="message">
            <?php echo $_SESSION['message']; ?>
        </div>
        <?php unset($_SESSION['message']); ?>
    <?php endif; ?>
    <form action="connexionUtilisateur.php" method="post">
        <label for="mail">Adresse e-mail :</label>
        <input type="email" name="mail" required>

        <label for="password">Mot de passe :</label>
        <input type="password" name="password" required>

        <button type="submit" name="login">Se connecter</button>
    </form>

    <p>Vous n'avez pas de compte ? <a href="inscription.php">Inscrivez-vous ici</a>.</p>
</div>

```

On commence avec un `<div class="login-form">` : Début de la section du formulaire de connexion. Il utilise une classe CSS pour le style. Puis on utilise une balise h2 : `<h2>Connectez-vous</h2>`: Titre indiquant à l'utilisateur qu'il s'agit du formulaire de connexion.

`<?php if (isset($_SESSION['message'])) : ?>` : Vérifie si un message de session est défini. Ce message peut être utilisé pour afficher des notifications à l'utilisateur après une action, comme une erreur de connexion.

Puis dans `<div class="message">` : qui est le début de la section pour afficher le message de session.

`<?php echo $_SESSION['message']; ?>` : Affiche le message de session, qui peut être une notification, un message d'erreur, etc.

`<?php unset($_SESSION['message']); ?>` : Supprime le message de session après l'avoir affiché une fois, pour qu'il ne soit pas affiché à nouveau lors du rechargement de la page.

`<?php endif; ?>` : Termine la condition PHP pour l'affichage du message de session.

`<form action="connexionUtilisateur.php" method="post">` : Définit le formulaire d'action qui renvoi vers le fichier "connexionUtilisateur.php" lors de la soumission du formulaire, en utilisant la méthode POST.

<label for="mail">Adresse e-mail :</label> : Étiquette pour le champ de saisie de l'adresse e-mail.

<input type="email" name="mail" required> : Champ de saisie de l'adresse e-mail. Le type "email" indique au navigateur de valider automatiquement si l'adresse e-mail saisie est au bon format. L'attribut "required" rend ce champ obligatoire.

<button type="submit" name="login">Se connecter</button> : Bouton de soumission du formulaire. Lorsqu'il est cliqué, les données du formulaire sont envoyées au fichier spécifié dans l'attribut "action".

<p>Vous n'avez pas de compte ? Inscrivez-vous ici.</p> : Message invitant l'utilisateur à s'inscrire s'il n'a pas encore de compte. Le lien "Inscrivez-vous ici" redirige vers la page d'inscription ("inscription.php").

En ce qui concerne le code PHP, la page commence toujours par la connexion à la base de données puis :

```
[1] if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['login'])) {
[2]     $mail = isset($_POST['mail']) ? $_POST['mail'] : '';
[3]     $password = isset($_POST['password']) ? $_POST['password'] : '';

[4]     // Vérification des champs de formulaire
[5]     if (empty($mail) || empty($password)) {
[6]         $_SESSION['message'] = "Veuillez remplir tous les champs.";
[7]         header("Location: connexionUtilisateur.php");
[8]         exit();
[9]     }

[10]    try {
[11]        $query = $connexion->prepare("SELECT num_adherent, user_login, user_mdp FROM adherent WHERE mail = :mail");
[12]        $query->bindParam(':mail', $mail, PDO::PARAM_STR);
[13]        $query->execute();
[14]        $user = $query->fetch(PDO::FETCH_ASSOC);

[15]        if ($user) {
[16]            // Hacher le mot de passe saisi par l'utilisateur pour comparaison
[17]            $hashed_password = hash('sha256', $password); // Utilisation de SHA-256 pour hacher le mot de passe

[18]            // Comparer le mot de passe haché saisi par l'utilisateur avec le mot de passe haché stocké dans la base de données
[19]            if ($hashed_password === $user['user_mdp']) {
[20]                $_SESSION['user_id'] = $user['num_adherent'];
[21]                $_SESSION['user_username'] = $user['user_login'];
[22]                $_SESSION['message'] = "Connexion réussie. Bienvenue, " . $user['user_login'] . "!";
[23]                header("Location: reservation.php"); // Rediriger vers une page appropriée après connexion
[24]                exit();
[25]            } else {
[26]                $_SESSION['message'] = "Identifiants incorrects. Veuillez réessayer.";
[27]                header("Location: connexionUtilisateur.php");
[28]                exit();
[29]            }
[30]        } else {
[31]            $_SESSION['message'] = "Utilisateur non trouvé. Veuillez vous inscrire.";
[32]            header("Location: inscription.php");
[33]            exit();
[34]        }
[35]    } catch (Exception $e) {
[36]        echo "Une erreur s'est produite : " . $e->getMessage();
[37]    }
[38] }
```

Ce code PHP gère la soumission du formulaire de connexion. Voici ce qu'il fait :

1. Vérifie si la requête est de type POST et si le bouton de connexion a été soumis dans le formulaire (`isset(\$_POST['login'])`).
2. Récupère les valeurs des champs de formulaire (`\$mail` et `\$password`) en utilisant l'opérateur ternaire pour éviter les erreurs si les champs ne sont pas définis.
3. Vérifie si les champs de formulaire ne sont pas vides. Si l'un des champs est vide, un message est défini dans la session, puis l'utilisateur est redirigé vers la page de connexion avec `header("Location: connexionUtilisateur.php")`.
4. Effectue une requête SQL pour récupérer les informations de l'utilisateur à partir de la base de données en fonction de l'adresse e-mail saisie dans le formulaire.
5. Vérifie si l'utilisateur existe dans la base de données. S'il existe, il hache le mot de passe saisi par l'utilisateur et le compare avec le mot de passe haché stocké dans la base de données.
6. Si les mots de passe correspondent, l'utilisateur est connecté. Ses informations sont stockées dans la session (`\$_SESSION['user_id']` et `\$_SESSION['user_username']`) et il est redirigé vers une page appropriée.

7. Si les mots de passe ne correspondent pas, un message d'erreur est défini dans la session et l'utilisateur est redirigé vers la page de connexion.
8. Si l'utilisateur n'est pas trouvé dans la base de données, un message invitant l'utilisateur à s'inscrire est défini dans la session, et l'utilisateur est redirigé vers la page d'inscription.
9. En cas d'erreur lors de l'exécution de la requête SQL, un message d'erreur est affiché et le script PHP se termine.

Page inscription.php :

localhost/PhpProject/bibliocat/inscription.php

Inscription

Inscrivez-vous

Nom :

Prénom :

Adresse e-mail :

Nom d'utilisateur :

Mot de passe :

S'inscrire

Vous avez déjà un compte ? [Connectez-vous ici.](#)

```

<header>
    <h1>Inscription</h1>
</header>

<div class="register-form">
    <h2>Inscrivez-vous</h2>
    <form action="inscription.php" method="post">
        <label for="nom">Nom :</label>
        <input type="text" name="nom" required>

        <label for="prenom">Prénom :</label>
        <input type="text" name="prenom" required>

        <label for="email">Adresse e-mail :</label>
        <input type="email" name="email" required>

        <label for="user_login">Nom d'utilisateur :</label>
        <input type="text" name="user_login" required>

        <label for="user_mdp">Mot de passe :</label>
        <input type="password" name="user_mdp" required>

        <button type="submit" name="register">S'inscrire</button>
    </form>

    <p>Vous avez déjà un compte ? <a href="connexionUtilisateur.php">Connectez-vous ici</a>.</p>
</div>

```

On a fait un formulaire simple pour s'inscrire qui est lié à la table adherent qui va permettre à l'utilisateur d'accéder à la page réservation pour réserver un livre.

Et dans le code PHP, il y a toujours la connexion à la base de données au début et après on récupère dans des variables les valeurs sur les champs saisies par l'utilisateur :

```

if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['register'])) {
    $nom = isset($_POST['nom']) ? $_POST['nom'] : '';
    $prenom = isset($_POST['prenom']) ? $_POST['prenom'] : '';
    $email = isset($_POST['email']) ? $_POST['email'] : '';
    $user_login = isset($_POST['user_login']) ? $_POST['user_login'] : '';
    $user_mdp = isset($_POST['user_mdp']) ? $_POST['user_mdp'] : '';

    if (empty($nom) || empty($prenom) || empty($email) || empty($user_login) || empty($user_mdp)) {
        echo "Tous les champs sont obligatoires.";
        exit();
    }

    $hashed_password = hash('sha256', $user_mdp); // Utilisation de SHA-256 pour hacher le mot de passe
    $date_inscription = date('Y-m-d');
    $sql = "INSERT INTO adherent (nom, prenom, mail, date_inscription, user_login, user_mdp) VALUES (:nom, :prenom, :email, :date_inscription, :user_login, :hashed_password)";

    try {
        $stmt = $connexion->prepare($sql);
        $stmt->bindParam(':nom', $nom);
        $stmt->bindParam(':prenom', $prenom);
        $stmt->bindParam(':email', $email);
        $stmt->bindParam(':date_inscription', $date_inscription);
        $stmt->bindParam(':user_login', $user_login);
        $stmt->bindParam(':hashed_password', $hashed_password);
        $stmt->execute();
        echo '<p style="color: green; font-weight: bold;">Inscription réussie !</p>';
    } catch (PDOException $e) {
        echo "Erreur lors de l'inscription : " . $e->getMessage();
    }
}
?>

```

On effectue une vérification avec la fonction empty que les champs sont tous remplis.

On stocke le mot de passe dans une variable pour le hacher avec la fonction hash (sha256) pour que le mot de passe ne soit pas en clair dans la base de données. Idem pour la date d'inscription, on récupère la date d'aujourd'hui (la date du jour où il s'est inscrit). Puis on insère les données avec des requêtes préparées pour éviter les injections sql.

Page Back Office :

On a réussi à faire en sorte que seulement celui qui a le compteAdmin puisse accéder à cette page
Voici à quoi elle ressemble :



Vu qu'il y a que le compte Admin qui a accès à cette page on vérifie qu'il est bien connecté sous ce nom :

```
<?php
session_start(); // Démarrer la session

// Vérifier si l'utilisateur est connecté en tant que "compteAdmin"
$isCompteAdmin = isset($_SESSION['user_username']) && $_SESSION['user_username'] === 'compteAdmin';

if (isset($_SESSION['user_username']) && $_SESSION['user_username'] != 'compteAdmin') {
    echo 'Accès refusé que l administrateur est autorisé à accéder à cette page';
}
?>
```

Et si ce n'est pas le cas un message s'affiche accès refusé

```
<h1>Back office</h1>
<!-- Boutons pour différentes fonctionnalités du back office -->
<a href="gestionLivres.php"><button class="menuBloc">Gestion des Livres</button></a>
<br>
<a href="gestionEmprunts.php"><button class="menuBloc">Gestion des Emprunts</button></a>
<br>
<a href="gestionUtilisateurs.php"><button class="menuBloc">Gestion des Utilisateurs</button></a>
<br>
<a href="gestionCatégorie.php"><button class="menuBloc">Gestion des catégories</button></a>
```

Sur la page on a mis des liens pour gérer les différentes bases de données :

Pour faire la mission pour sécuriser le back office pour qu'un utilisateur ne puisse pas accéder à la session avec le lien on a du rajouter :

```
// Vérifier si le formulaire de destruction de session est soumis
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['destroy_session'])) {
    // Détruire la session
    session_destroy();
    // Rediriger l'utilisateur vers une page d'accueil ou une autre page appropriée après la déconnexion
    header("Location: index.php");
    exit; // Terminer le script pour éviter toute exécution supplémentaire
}
```

Pour détruire la session on du utiliser un session destroy non quand l'administrateur va cliquer sur le bouton se déconnecter dans la nav bar cela va détruire la session automatiquement :

Le bouton est dans la nav bar :



Et quand on clique sur le bouton cela nous renvoie vers index.php, et qu'on copie le lien de la page back-office cela nous renvoie bien vers index.php pour montrer que cela arche bien et que si l'utilisateur n'est pas connecté en tant que compte Adminsitarteur cela ne va pas marcher :



Page Back Office pour la gestion des emprunts :

```

<h1>Liste des emprunts</h1>
<?php if (!empty($message)) : ?>
    <p><?php echo $message; ?></p>
<?php endif; ?>
<table>
    <thead>
        <tr>
            <th>Identifiant</th>
            <th>Référence Livre</th>
            <th>Date de début</th>
            <th>Date de fin</th>
            <th>Numéro Adhérent</th>
            <th>Actions</th>
        </tr>
    </thead>
    <tbody>
        <?php foreach ($emprunts as $emprunt) : ?>
            <tr>
                <td><?php echo $emprunt['id']; ?></td>
                <td><?php echo $emprunt['ref']; ?></td>
                <td><input type="date" name="date_deb" value="<?php echo $emprunt['date_deb']; ?>"></td>
                <td><input type="date" name="date_fin" value="<?php echo $emprunt['date_fin']; ?>"></td>
                <td><input type="text" name="num_adherent" value="<?php echo $emprunt['num_adherent']; ?>"></td>
                <td>
                    <button type="submit" name="update">Mettre à jour</button>
                    <button type="submit" name="delete">Supprimer</button>
                </td>
            </tr>
        <?php endforeach; ?>
    </tbody>
</table>

<a href="back-office.php"><button class="menuBloc">Retour au Back Office</button></a>

```

Ce code affiche tous les emprunts des livres récupérés directement dans la base de données.

<?php if (!empty(\$message)) : ?> : Début d'une structure conditionnelle qui vérifie si la variable \$message n'est pas vide. Cette variable est utilisée pour afficher des messages de confirmation ou d'erreur après une opération (comme la mise à jour ou la suppression d'un emprunt).

Cette ligne <p><?php echo \$message; ?></p> : permet l'affichage du message contenu dans la variable \$message, s'il n'est pas vide.

Pour afficher les données on a utilisé : <table> : Début d'un tableau HTML pour afficher les données des emprunts.

<thead> : En tête du tableau.

<tr> : Ligne d'entête du tableau.

<th>Identifiant</th> : Cellule d'en-tête pour l'identifiant de l'emprunt.

<th>Référence Livre</th> : Cellule d'en-tête pour la référence du livre emprunté.

<th>Date de début</th> : Cellule d'en-tête pour la date de début de l'emprunt.

...

</tr> : Fin de la ligne d'entête.

</thead> : Fin de l'entête du tableau.

<tbody> : Corps du tableau contenant les données des emprunts.

<?php foreach (\$emprunts as \$emprunt) : ?> : Début d'une boucle foreach qui parcourt chaque emprunt dans le tableau \$emprunts.

<tr> : Nouvelle ligne pour chaque emprunt.

<form method="POST"> : Formulaire avec une méthode POST pour envoyer les données de mise à jour ou de suppression de l'emprunt.

<input type="hidden" name="id" value="<?php echo \$emprunt['id']; ?>"> : Champ caché contenant l'identifiant de l'emprunt.

<td><?php echo \$emprunt['id']; ?></td> : Cellule affichant l'identifiant de l'emprunt.

<td><input type="text" name="ref" value="<?php echo \$emprunt['ref']; ?>"></td> : Cellule avec un champ texte affichant la référence du livre emprunté.

Et après c'est la même chose pour les autres champs.

<td> : Cellule pour les boutons d'action.

<button type="submit" name="update">Mettre à jour</button> : Bouton de soumission pour mettre à jour les données de l'emprunt.

<button type="submit" name="delete">Supprimer</button> : Bouton de soumission pour supprimer l'emprunt.

Et le code php pour les modifications et suppressions :

```
// Vérification si le formulaire de mise à jour est soumis
if ($_SERVER['REQUEST_METHOD'] == "POST") {
    // Si le bouton "Mettre à jour" est cliqué
    if (isset($_POST['update'])) {
        // Traitement de la mise à jour des données
        $id = $_POST['id'];
        $date_deb = date('Y-m-d', strtotime($_POST['date_deb']));
        $date_fin = date('Y-m-d', strtotime($_POST['date_fin']));
        $ref = $_POST['ref'];
        $num_adherent = $_POST['num_adherent'];

        $stmt = $connexion->prepare("UPDATE emprunt SET date_deb = :date_deb, date_fin = :date_fin, ref = :ref, num_adherent = :num_adherent WHERE id = :id");
        $stmt->bindParam(':date_deb', $date_deb);
        $stmt->bindParam(':date_fin', $date_fin);
        $stmt->bindParam(':ref', $ref);
        $stmt->bindParam(':num_adherent', $num_adherent);
        $stmt->bindParam(':id', $id);
        $stmt->execute();

        // Vérifier si la mise à jour a été effectuée avec succès
        if ($stmt->rowCount() > 0) {
            $message = "La mise à jour a été effectuée avec succès.";
        } else {
            $message = "La mise à jour a échoué.";
        }
    }

    // Vérifier si le bouton "Supprimer" est cliqué
    if (isset($_POST['delete'])) {
        // Traitement de la suppression de l'emprunt
        $id = $_POST['id'];

        // Vérifier d'abord si le livre est actuellement emprunté
        $stmt_check_emprunt = $connexion->prepare("SELECT * FROM emprunt WHERE ref = :ref AND date_fin >= CURDATE()");
        $stmt_check_emprunt->bindParam(':ref', $_POST['ref']);
        $stmt_check_emprunt->execute();
        $emprunt_existe = $stmt_check_emprunt->fetch();

        if ($emprunt_existe) {
            $message = "Ce livre est actuellement emprunté et ne peut pas être supprimé.";
        } else {
            try {
                // Supprimer l'emprunt de la base de données
                $stmt = $connexion->prepare("DELETE FROM emprunt WHERE id = :id");
                $stmt->bindParam(':id', $id);
                $stmt->execute();

                // Vérifier si la suppression a été effectuée avec succès
                if ($stmt->rowCount() > 0) {
                    $message = "L'emprunt a été supprimé avec succès.";
                } else {
                    $message = "La suppression de l'emprunt a échoué.";
                }
            } catch (PDOException $e) {
                // Afficher les erreurs SQL
                echo "Erreur SQL: " . $e->getMessage();
            }
        }
    }
}

// Récupération des données d'emprunts depuis la base de données
$stmt = $connexion->query("SELECT id, ref, DATE_FORMAT(date_deb, '%Y-%m-%d') as date_deb, DATE_FORMAT(date_fin, '%Y-%m-%d') as date_fin, num_adherent FROM emprunt");
$emprunts = $stmt->fetchAll(PDO::FETCH_ASSOC);

?>
```

Ce code gère la mise à jour et la suppression des emprunts, ainsi que l'affichage des messages de confirmation ou d'erreur. Voici un résumé de son fonctionnement :

La variable \$message est initialisée comme une chaîne vide.

Le code vérifie si le formulaire a été soumis en vérifiant la méthode de la requête (POST).

Si le bouton "Mettre à jour" est cliqué (update est présent dans les données POST), les données de l'emprunt sont récupérées à partir des champs du formulaire. Ensuite, une requête SQL est préparée pour mettre à jour les données de l'emprunt dans la base de données. Si la mise à jour réussit, le message de succès est stocké dans la variable \$message, sinon un message d'échec est stocké.

Si le bouton "Supprimer" est cliqué (delete est présent dans les données POST), l'identifiant de l'emprunt à supprimer est récupéré à partir du champ caché du formulaire. Avant de supprimer l'emprunt, le code vérifie si le livre est actuellement emprunté en utilisant une requête SQL pour rechercher des emprunts actifs pour ce livre. Si l'emprunt existe, un message est stocké dans \$message indiquant qu'il ne peut pas être supprimé. Sinon, l'emprunt est supprimé de la base de données. Si la suppression réussit, un message de succès est stocké dans \$message, sinon un message d'échec est stocké.

Les données d'emprunts sont récupérées depuis la base de données pour être affichées dans le tableau.

Ce processus garantit que les données des emprunts sont mises à jour et supprimées correctement, et que les messages appropriés sont affichés à l'utilisateur en fonction du résultat de ces opérations.

Dans la précédente partie on a expliqué plus clairement comment on utilise le session_destroy pour sécuriser le back_office.

Page Gestion des catégories :

Code Php :

```
// Vérification si le formulaire de gestion de catégorie est soumis
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Si le bouton "Ajouter" est cliqué
    if (isset($_POST['add_category'])) {
        // Récupérer la nouvelle catégorie depuis le formulaire
        $new_category = $_POST['new_category'];

        // Vérifier si la catégorie existe déjà dans la base de données
        $stmt_check = $connexion->prepare("SELECT * FROM categorie WHERE libelle = ?");
        $stmt_check->execute([$new_category]);
        $existing_category = $stmt_check->fetch();

        // Si la catégorie n'existe pas déjà, l'ajouter dans la base de données
        if (!$existing_category) {
            $stmt_insert_cat = $connexion->prepare("INSERT INTO categorie (libelle) VALUES (?)");
            $stmt_insert_cat->execute([$new_category]);

            // Récupérer l'ID de la nouvelle catégorie ajoutée
            $new_category_id = $connexion->lastInsertId();

            $message = "La catégorie a été ajoutée avec succès.";
        } else {
            $message = "La catégorie existe déjà dans la base de données.";
        }
    }

    // Si l'identifiant de catégorie est fourni et valide
    if (isset($_POST['category_id']) && isset($_POST['edited_category'])) {
        $category_id = $_POST['category_id'];
        $edited_category = $_POST['edited_category'];

        // Récupérer la catégorie actuelle depuis la base de données
        $stmt_get_category = $connexion->prepare("SELECT libelle FROM categorie WHERE id = ?");
        $stmt_get_category->execute([$category_id]);
        $current_category = $stmt_get_category->fetchColumn();

        // Vérifier si la catégorie a été modifiée
        if ($edited_category !== $current_category) {
            // Modifier la catégorie dans la base de données
            $stmt_edit_cat = $connexion->prepare("UPDATE categorie SET libelle = ? WHERE id = ?");
            $stmt_edit_cat->execute([$edited_category, $category_id]);

            // Vérifier si la modification a été effectuée avec succès
            if ($stmt_edit_cat->rowCount() > 0) {
                $message = "La catégorie a été modifiée avec succès.";
            } else {
                $message = "La modification de la catégorie a échoué.";
            }
        } else {
            $message = "La catégorie n'a pas été modifiée.";
        }
    }
}
```

```

        $message = "La modification de la catégorie a échoué.";
    }
} else {
    $message = "Aucun changement détecté.";
}

}

// Si le bouton "Supprimer" est cliqué
if (isset($_POST['delete_category'])) {
    // Récupérer l'identifiant de la catégorie à supprimer depuis le formulaire
    $category_id = $_POST['category_id'];

    // Supprimer la catégorie de la base de données
    $stmt_delete_cat = $connexion->prepare("DELETE FROM categorie WHERE id = ?");
    $stmt_delete_cat->execute([$category_id]);

    // Vérifier si la suppression a été effectuée avec succès
    if ($stmt_delete_cat->rowCount() > 0) {
        $message = "La catégorie a été supprimée avec succès.";
    } else {
        $message = "La suppression de la catégorie a échoué.";
    }
}

// Récupération des catégories depuis la base de données
$stmt = $connexion->query("SELECT id, libelle FROM categorie ORDER BY libelle ASC");
$categories = $stmt->fetchAll(PDO::FETCH_ASSOC);
?>

```

Ce segment de code gère l'ajout, la modification et la suppression des catégories dans une base de données. Voici un résumé sans liste :

Vérification de la session administrateur : La variable \$isCompteAdmin est définie en vérifiant si l'utilisateur est connecté en tant que "compteAdmin".

Traitement du formulaire : Le code vérifie si une requête POST est soumise. Si c'est le cas, il vérifie quel bouton a été cliqué :

Ajouter une catégorie : Si le bouton "Ajouter" est cliqué, le code récupère la nouvelle catégorie depuis le formulaire, vérifie si elle existe déjà dans la base de données, puis l'ajoute si elle n'existe pas.

Modifier une catégorie : Si une catégorie existante est modifiée et soumise, le code met à jour la catégorie dans la base de données.

Supprimer une catégorie : Si le bouton "Supprimer" est cliqué, le code supprime la catégorie sélectionnée de la base de données.

Récupération des catégories depuis la base de données : Le code exécute une requête SQL pour récupérer toutes les catégories depuis la table "categorie" et les stocke dans la variable \$categories.

En résumé, ce code PHP gère l'ajout, la modification et la suppression de catégories dans une base de données, ainsi que la récupération des catégories existantes pour les afficher dans un formulaire HTML.

partie html/php :

```
<body>
    <h1>Gestion des Catégories</h1>
    <?php if (!empty($message)) : ?>
        <p><?php echo $message; ?></p>
    <?php endif; ?>
    <form method="POST">
        <label for="new_category">Nouvelle Catégorie:</label>
        <input type="text" id="new_category" name="new_category" required>
        <button type="submit" name="add_category">Ajouter</button>
    </form>
    <h2>Liste des Catégories</h2>
    <ul>
        <?php foreach ($categories as $category) : ?>
            <li>
                <form method="POST">
                    <input type="hidden" name="category_id" value="<?php echo $category['id']; ?>">
                    <input type="text" name="edited_category" value="<?php echo $category['libelle']; ?>">
                    <button type="submit" name="edit_category">Modifier</button>
                </form>
                <form method="POST">
                    <input type="hidden" name="category_id" value="<?php echo $category['id']; ?>">
                    <button type="submit" name="delete_category">Supprimer</button>
                </form>
            </li>
        <?php endforeach; ?>
    </ul>
    <a href="back-office.php"><button class="menuBloc">Retour au Back Office</button></a>
</body>
```

Cette partie du code gère l'affichage des catégories, la possibilité d'ajouter de nouvelles catégories, ainsi que la modification et la suppression des catégories existantes. Voici un résumé sans liste :

Affiche un titre "Gestion des Catégories".

Affiche un message si la variable \$message n'est pas vide.

Affiche un formulaire permettant d'ajouter une nouvelle catégorie. Le formulaire comporte un champ pour entrer le nom de la nouvelle catégorie et un bouton "Ajouter".

Affiche un sous-titre "Liste des Catégories".

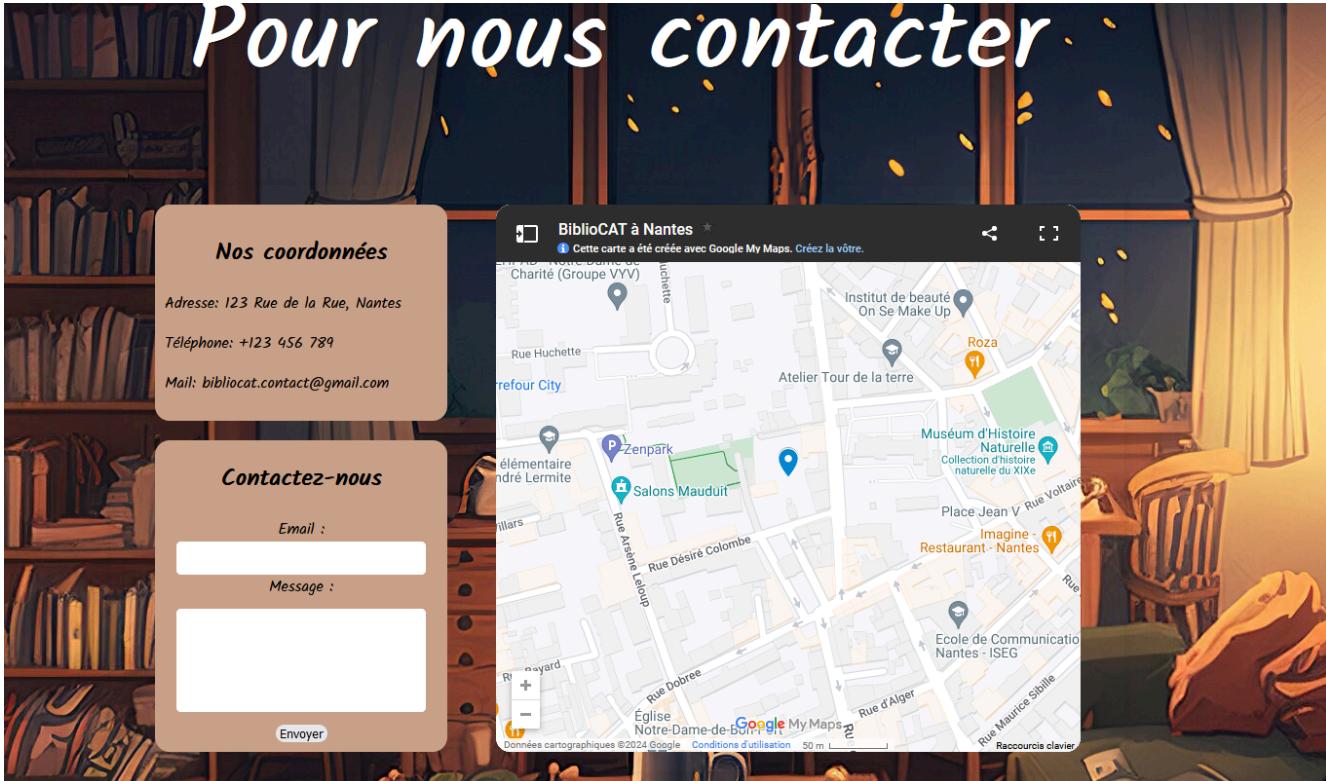
Affiche chaque catégorie existante sous forme de liste. Pour chaque catégorie, affiche un formulaire avec un champ pour modifier le nom de la catégorie et un bouton "Modifier", ainsi qu'un bouton "Supprimer" pour supprimer la catégorie.

Fournit un lien pour retourner au Back Office.

Cette partie traite aussi de la suppression de catégorie(cela vérifie si un livre existe ou pas avec cette catégorie, si elle existe déjà, ...)

Page contact.php :

Pour nous contacter



Code html :

```
<h1>Pour nous contacter</h1>
<!-- Zone Contact formulaire --&gt;
&lt;div class="bloc-contact"&gt;

    &lt;div class="sous-bloc-contact"&gt;
        &lt;div class="bloc-contact-coordonnees"&gt;
            &lt;h2&gt;Nos coordonnées&lt;/h2&gt;
            &lt;p&gt;Adresse: 123 Rue de la Rue, Nantes&lt;/p&gt;
            &lt;p&gt;Téléphone: +123 456 789&lt;/p&gt;
            &lt;p&gt;Mail: bibliocat.contact@gmail.com&lt;/p&gt;
        &lt;/div&gt;
        &lt;div class="bloc-contact-formulaire"&gt;
            &lt;form action="#" method="post" id="contact-form" onsubmit="afficherMessage(); return false;"&gt;
                &lt;h2&gt;Contactez-nous&lt;/h2&gt;

                    &lt;div&gt;
                        &lt;label for="email"&gt;Email :&lt;/label&gt;
                    &lt;/div&gt;
                    &lt;div&gt;
                        &lt;input type="email" id="email" name="email" required&gt;
                    &lt;/div&gt;
                    &lt;div&gt;
                        &lt;label for="message"&gt;Message :&lt;/label&gt;
                    &lt;/div&gt;
                    &lt;div&gt;
                        &lt;textarea id="message" name="message" rows="4" required&gt;&lt;/textarea&gt;
                    &lt;/div&gt;
                    &lt;div&gt;
                        &lt;button type="submit" id="bouton-mail"&gt;Envoyer&lt;/button&gt;
                    &lt;/div&gt;
            &lt;/form&gt;
        &lt;/div&gt;
    &lt;/div&gt;
    &lt;div class="bloc-contact-map"&gt;
        &lt;!-- Ajoutez ici le code nécessaire pour intégrer le widget Google Maps --&gt;
        &lt;!-- (à remplacer par notre code Google Maps) --&gt;
        &lt;iframe src="https://www.google.com/maps/d/embed?mid=1UKRIVhKyi8ghgPsjCmbroY0ildXB4fg&amp;ehbc=2E312F&amp;noprof=1"&gt;&lt;/iframe&gt;
    &lt;/div&gt;
</pre>
```

Ce segment de code concerne la page de contact et comprend les éléments suivants :

Affichage du titre "Pour nous contacter".

Division (div) principale pour le formulaire de contact, avec deux sous-divisions.

La première sous-division affiche les coordonnées de l'entreprise, telles que l'adresse, le numéro de téléphone et l'adresse e-mail.

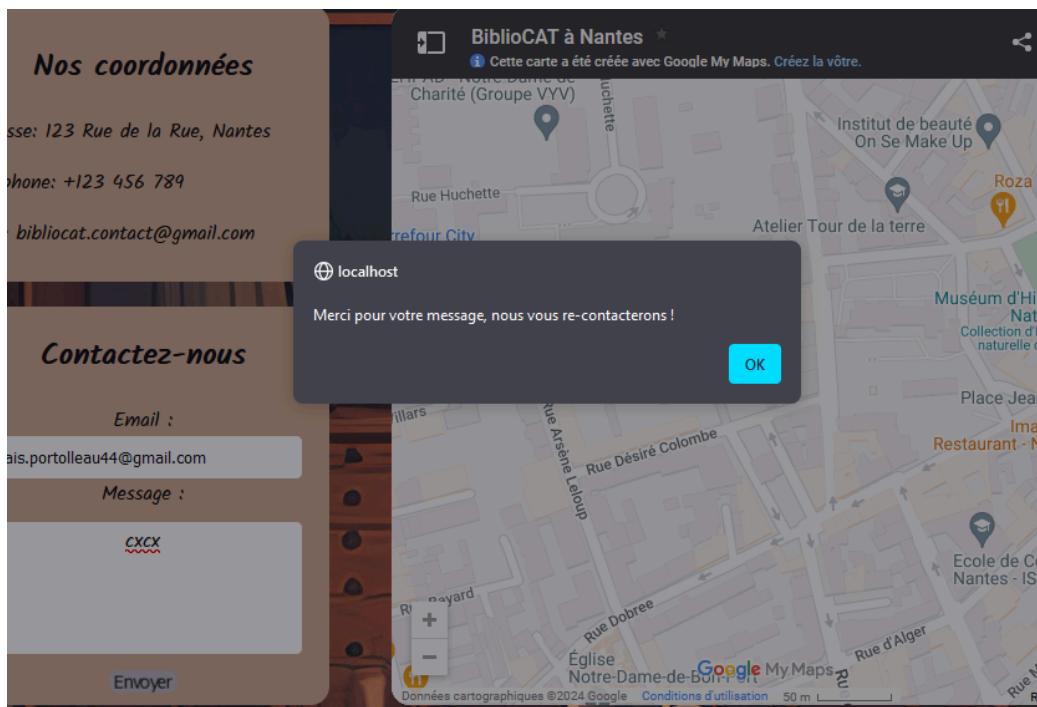
La deuxième sous-division contient le formulaire de contact. Ce formulaire comprend des champs pour l'e-mail de l'utilisateur et son message, ainsi qu'un bouton "Envoyer" pour soumettre le formulaire.

Il existe également une division supplémentaire pour intégrer une carte Google Maps. Dans cet exemple, une iframe est utilisée pour intégrer la carte de la localisation de notre bibliothèque.

En résumé, cette partie de code affiche les informations de contact de l'entreprise, un formulaire de contact pour les utilisateurs et une carte Google Maps pour la localisation.

Et le script js quand l'utilisateur clique sur envoyer le message :

```
<!-- Ajoutez du code JavaScript pour afficher une alerte -->
<script>
    function afficherMessage() {
        alert("Merci pour votre message, nous vous re-contacterons !");
    }
</script>
```



Galerie.php :

Code php :

```

<?php
session_start(); // Démarrer la session

// Vérifier si l'utilisateur est connecté en tant que "compteAdmin"
$isCompteAdmin = isset($_SESSION['user_username']) && $_SESSION['user_username'] === 'compteAdmin';

// connexion à la base de données
try {
    $db = "bibliocat";
    $dbhost = "localhost";
    $dbport = 3307;
    $dbuser = "compteAdmin";
    $dbpasswd = "joliverie";
    $connexion = new PDO('mysql:host=' . $dbhost . ':port=' . $dbport . ',dbname=' . $db, $dbuser, $dbpasswd);
    $connexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $connexion->exec("SET CHARACTER SET utf8");
} catch (PDOException $e) {
    echo "Erreur de connexion à la base de données: " . $e->getMessage();
    exit();
}

// Définir la variable $boutonAdmin avec une valeur par défaut
$boutonAdmin = '';

// Vérifie si l'utilisateur est connecté et s'il est administrateur
if(isset($_SESSION['user_username']) && $_SESSION['user_username'] == 'compteAdmin') {
    $boutonAdmin = '<a href="gestionLivres.php"><button class="button-traitement">Gestion des Livres</button></a>';
} else {
    $boutonAdmin = ''; // vérifie si la variable est définie même si l'utilisateur n'est pas administrateur
}
?>

```

Comme d'habitude je me connecte à la base de données, je définis la variable boutonAdmin comme vide.

Et je vérifie si l'utilisateur est connecté en tant que compteAdmin car un bouton apparaît dans la page que si c'est un admin qui va renvoyer vers la page Gestion des Livres.

Et la page Galerie affiche toutes les couvertures des livres de notre base de données et si on clique dessus cela emmène vers les infos des livres, les livres sont ordonnées dans l'ordre croissant des titres :

```

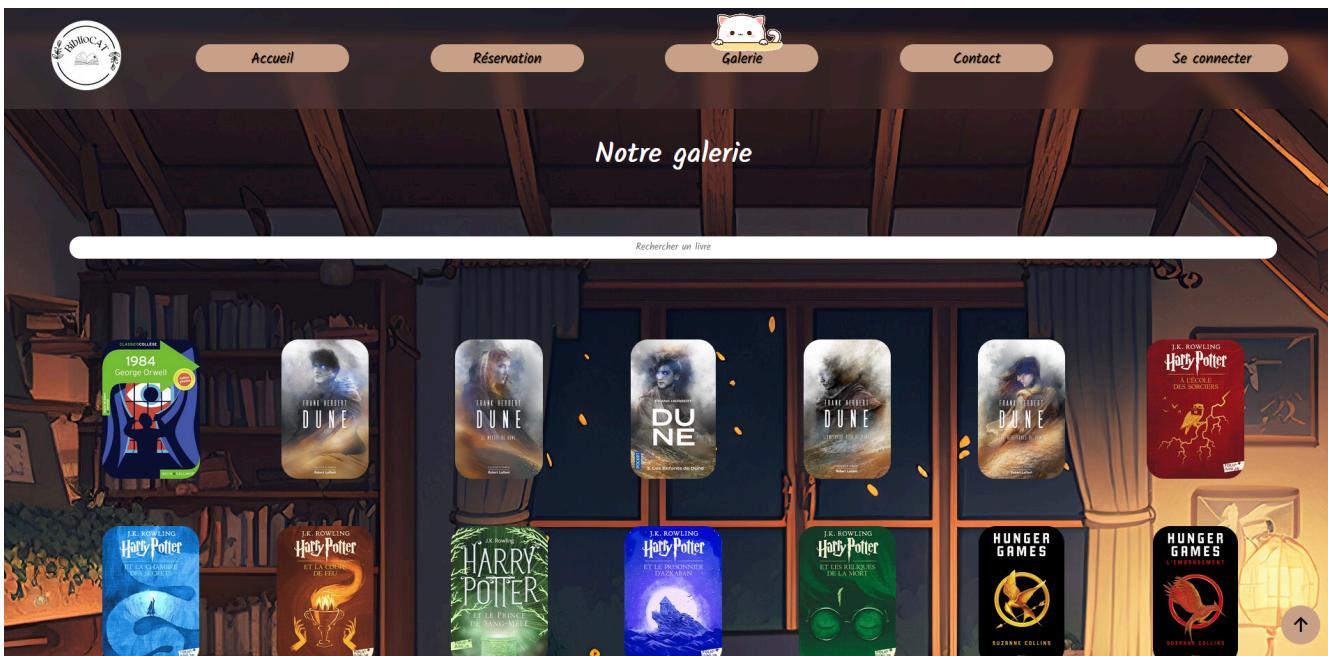
<div class="galerie">Notre galerie
<br>

<!-- Affichage du bouton administrateur --&gt;
&lt;?php echo $boutonAdmin; ?&gt;
&lt;div class="boutique"&gt;
    &lt;?php
        $listeCouverture = $connexion-&gt;query("SELECT * FROM livre ORDER BY titre ASC");

        while ($couverture = $listeCouverture-&gt;fetch()) {
            echo '&lt;div class="grid-item"&gt;
                    &lt;a href="galerie-infos.php?id=' . $couverture["ref"] . '"&gt;
                        &lt;img src="Images/'. $couverture["photo"] . '" height="200px" class="zoom"&gt;
                    &lt;/a&gt;
                &lt;/div&gt;';
        }
    ?&gt;
&lt;/div&gt;
&lt;/div&gt;
&lt;/main&gt;
</pre>

```

Voici à quoi ressemble la page galerie côté utilisateur :



Et côté admin :



Un bouton s'affiche bien que du côté admin.

On a également rajouté une possibilité de chercher directement dans la barre de recherche par nom et cela affiche grâce à un LIKE les noms des livres qui ressemble

Page galerie-infos :

Affiche les infos des livres quand dans la page galerie on clique dessus :



Partie PHP :

```

$compteAdmin = false;
if(isset($_SESSION['email']) && $_SESSION['email'] === "compteAdmin@email.com" && isset($_SESSION['login']) && $_SESSION['login'] === "compteAdmin") {
    $compteAdmin = true;
}

// Vérifier si l'ID du livre est passé dans l'URL
if (isset($_GET['id'])) {
    $id_livre = $_GET['id'];

    // Récupérer les informations du livre depuis la base de données
    $query = "SELECT * FROM livre WHERE ref = :id";
    $stmt = $connexion->prepare($query);
    $stmt->bindParam(':id', $id_livre);
    $stmt->execute();
    $livre = $stmt->fetch();

    // Si le livre existe
    if ($livre) {
        // Afficher l'image du livre
        ?>
        <div class="galerie-infos-livre">
            <?php echo '';?>
        </div>
        <?php
            } else {
                echo "Livre non trouvé.";
            }
    } else {
        echo "ID du livre non spécifié.";
    }
} catch (PDOException $e) {
    echo "Erreur de connexion à la base de données: " . $e->getMessage();
    exit();
}
?>
```

Ce code PHP vérifie d'abord si l'identifiant d'un livre est passé dans l'URL. Si tel est le cas, il récupère les informations du livre correspondant depuis la base de données. Ensuite, il vérifie si le livre existe. Si c'est le cas, il affiche l'image du livre. Sinon, il affiche un message indiquant que le livre n'a pas été trouvé. Si l'identifiant du livre n'est pas spécifié dans l'URL, il affiche un message indiquant que l'identifiant du livre n'est pas spécifié. En cas d'erreur de connexion à la base de données, il affiche un message d'erreur.

Code HTML :

```

<div class="bloc-galerie-infos">
    <?php
        // Afficher les autres informations du livre
        if ($livre) {
            echo '<h2><strong>' . " " . $livre['titre'] . '</strong></h2>';
            echo '<p><strong>ISBN:</strong>' . " " . $livre['iban'] . '</p>';
            echo '<p><strong>Auteur:</strong>' . " " . $livre['auteur'] . '</p>';
            echo '<p><strong>Langue:</strong>' . " " . $livre['langue'] . '</p>';
            echo '<p><strong>Année:</strong>' . " " . $livre['annee'] . '</p>';

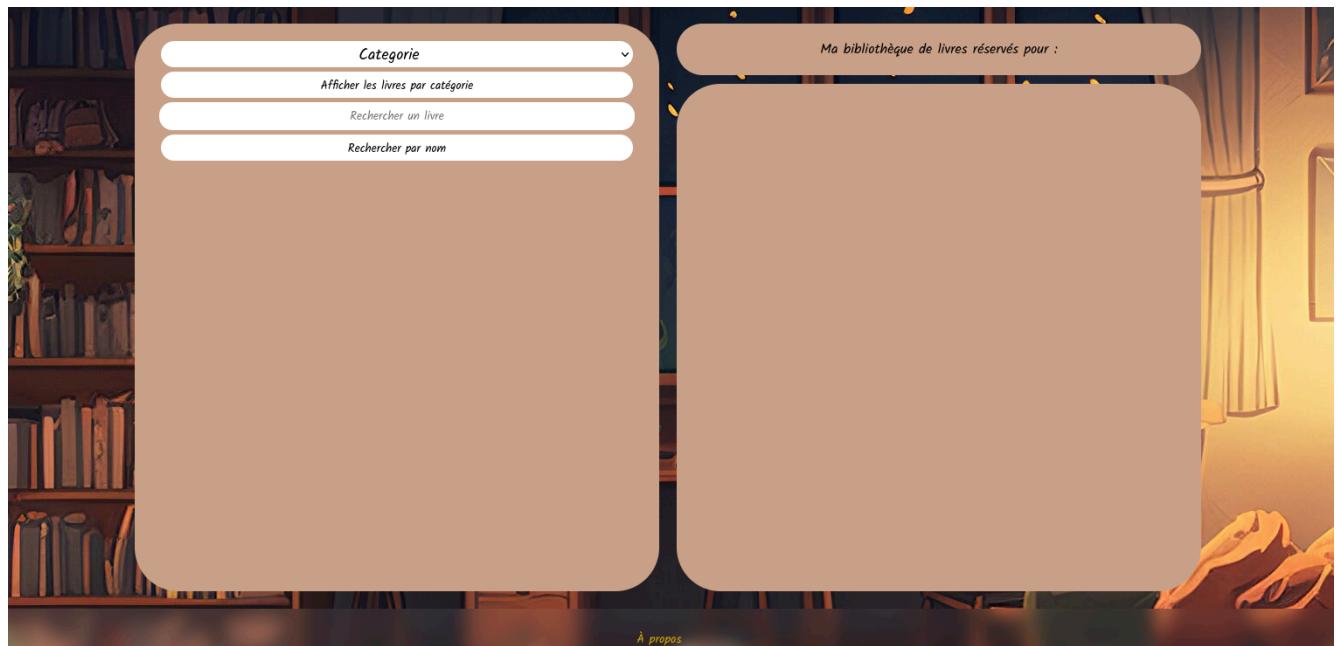
            // Récupérer le libellé de la catégorie du livre depuis la table catégorie
            $query_cat = "SELECT libelle FROM categorie WHERE id = :id_cat";
            $stmt_cat = $connexion->prepare($query_cat);
            $stmt_cat->bindParam(':id_cat', $livre['id_cat']);
            $stmt_cat->execute();
            $categorie = $stmt_cat->fetch();

            echo '<p><strong>Catégorie: </strong>' . $categorie['libelle'] . '</p>';
            echo '<p><strong>Description :</strong>' . " " . $livre['descrip'] . '</p>';
        }
    ?>
</div>
</div>
</div>

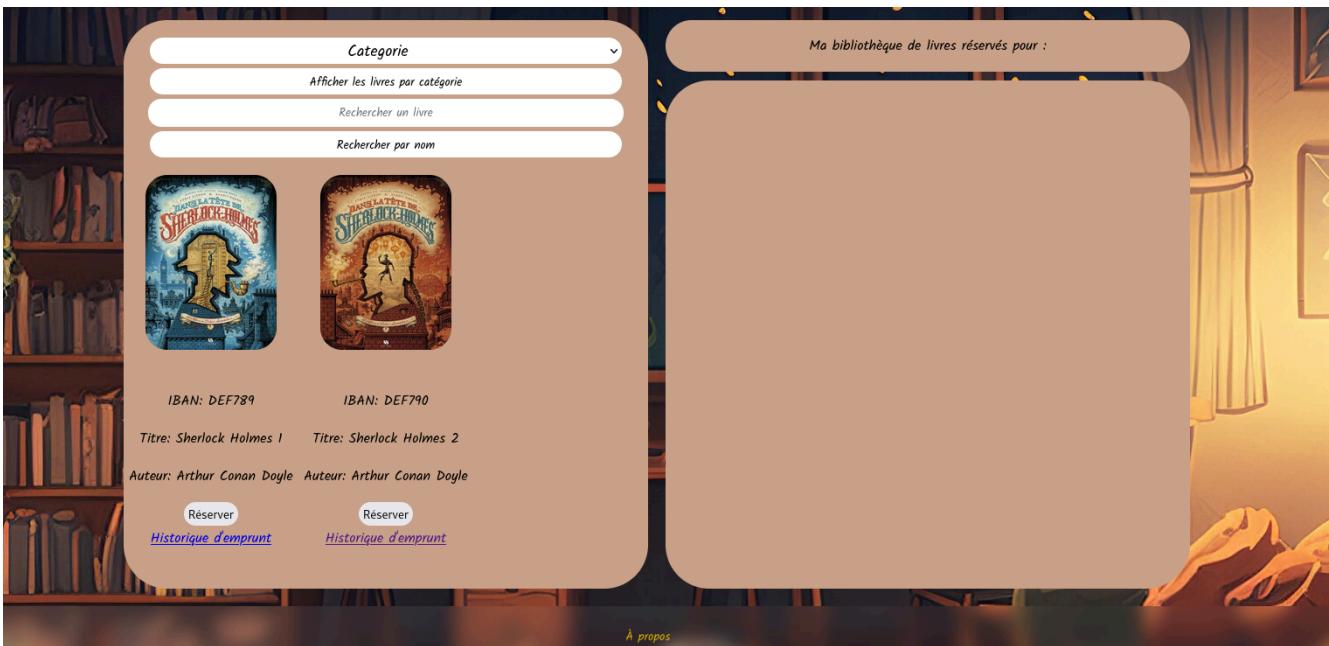
```

Ici on affiche diverses informations sur un livre, telles que le titre, l'ISBN, l'auteur, la langue, l'année, la catégorie et la description. Il vérifie d'abord si les informations sur le livre existent. Si c'est le cas, il récupère et affiche ces informations à partir du tableau associatif \$livre. Il récupère également le libellé de la catégorie du livre depuis la table des catégories et l'affiche.

Page reservation.php :



Sur cette page, toutes les fonctionnalités liées à notre système de réservation sont présentes. Nous avons en premier deux filtres possibles. Un filtre par catégorie et un filtre par nom du livre.

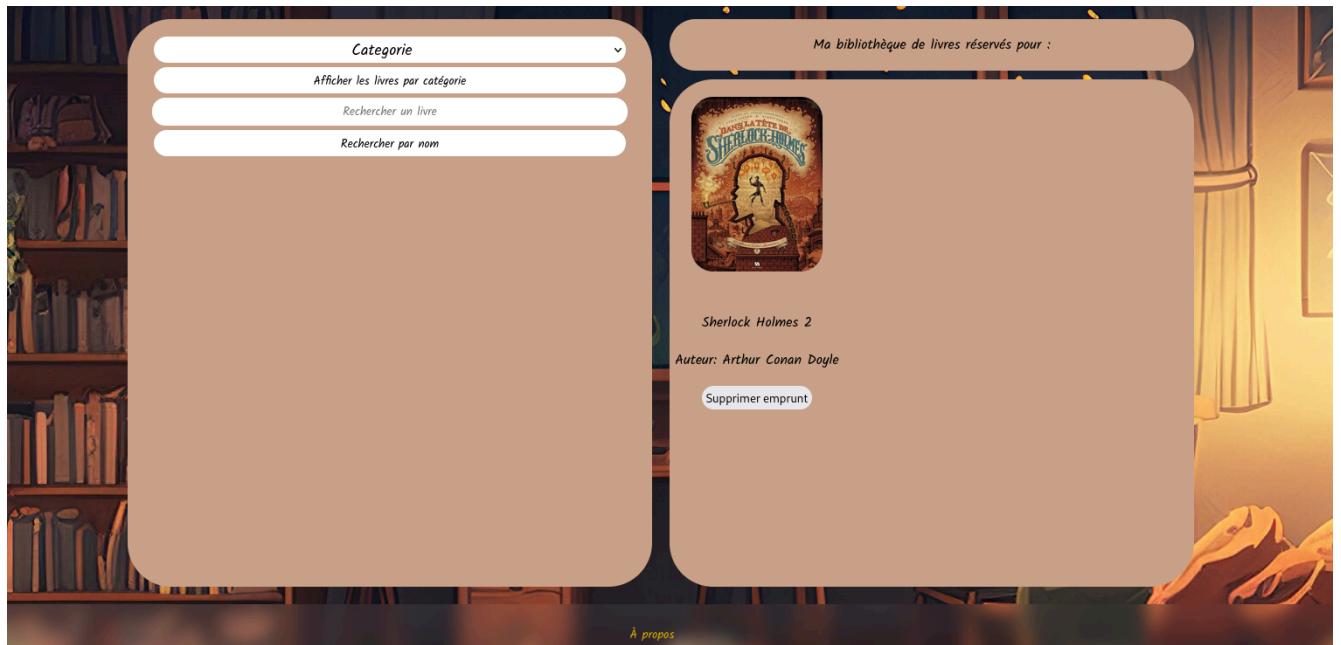


Par exemple, je choisis la catégorie Mystère et uniquement les livres de la catégorie Mystère s'affichent.

Il est indiqué à ce moment en dessous des images, quelques informations sur les livres concernés, un lien vers l'historique de ses emprunts et pour finir un bouton pour le réserver.

Si on clique sur le bouton réserver, le livre va dans la partie des livres réservés à droite. Unique par utilisateur.

Si on re-réserve un livre qui est actuellement emprunté :



On peut ensuite si on veut supprimer l'emprunt.

Au niveau du code que vous pouvez voir ci-dessous, il permet de faire la recherche d'un livre par nom. Il récupère grâce à une requête SQL les livres qui correspondent partiellement ou complètement au texte écrit grâce au "LIKE" de la requête.

```
// Effectuez la requête SQL en fonction des données du formulaire
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (isset($_POST['search'])) {
        $search = $_POST['search'];

        // Requête SQL avec utilisation de paramètres préparés pour éviter les injections SQL
        $requete = "SELECT * FROM livre l WHERE titre LIKE :search";

        // Préparez la requête SQL
        $stmt = $connexion->prepare($requete);

        // Bind des valeurs des paramètres
        $stmt->bindValue(':search', '%' . $search . '%', PDO::PARAM_STR);

        // Exécutez la requête SQL
        $stmt->execute();
    }
}
```

Par la suite, le code suivant permet d'afficher les catégories de la liste déroulante. De même grâce à une requête SQL. On précise également qu'on a utilisé des requêtes préparées pour augmenter la sécurité du site.

```
} elseif (isset($_POST['categorie'])) {
    $categorie = $_POST['categorie'];

    // Requête SQL avec utilisation de paramètres préparés pour éviter les injections SQL
    $requete = "SELECT * FROM livre l WHERE l.id_cat = :categorie";

    // Préparez la requête SQL
    $stmt = $connexion->prepare($requete);

    // Bind des valeurs des paramètres
    $stmt->bindValue(':categorie', $categorie, PDO::PARAM_INT);

    // Exécutez la requête SQL
    $stmt->execute();

}
```

Maintenant, ce code est le code du clique sur le bouton réserver. Il insère les valeurs du livre dans la table emprunt de notre base de données. Il vérifie que le livre n'est pas déjà réservé.

```

} else {
    // Une erreur s'est produite lors de la réservation
    $_SESSION['error_message'] = "Une erreur s'est produite lors de la réservation du livre.";
    header("Location: reservation.php");
    exit();
}

} else {
    // L'utilisateur n'est pas connecté, redirigez-le vers la page de connexion
    $_SESSION['error_message'] = "Vous devez vous connecter avant de réserver un livre.";
    header("Location: connexionUtilisateur.php");
    exit();
}

// Traitement pour la réservation des livres
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['reserver'])) {
    // Vérifiez si l'utilisateur est connecté
    if (isset($_SESSION['user_id'])) {
        // Récupérez les données du formulaire
        $ref_livre = $_POST['reserver'];

        // Requête SQL pour insérer une nouvelle réservation dans la table appropriée
        $requete_reservation = "INSERT INTO emprunt (num_adherent, ref) VALUES (:user_id, :ref_livre)";
        $stmt_reservation = $connexion->prepare($requete_reservation);
        $stmt_reservation->bindValue(':user_id', $user_id, PDO::PARAM_INT);
        $stmt_reservation->bindValue(':ref_livre', $ref_livre, PDO::PARAM_STR);

        // Vérifier si le livre est déjà réservé par l'utilisateur
        $requete_verif_reservation = "SELECT COUNT(*) FROM emprunt WHERE num_adherent = :user_id AND ref = :ref_livre";
        $stmt_verif_reservation = $connexion->prepare($requete_verif_reservation);
        $stmt_verif_reservation->bindValue(':user_id', $user_id, PDO::PARAM_INT);
        $stmt_verif_reservation->bindValue(':ref_livre', $ref_livre, PDO::PARAM_STR);
        $stmt_verif_reservation->execute();
        $reservation_count = $stmt_verif_reservation->fetchColumn();

        if ($reservation_count > 0) {
            // Le livre est déjà réservé par l'utilisateur
            $_SESSION['error_message'] = "Vous avez déjà réservé ce livre.";
            header("Location: reservation.php");
            exit();
        }
    }

    // Exécutez la requête SQL pour effectuer la réservation
    if ($stmt_reservation->execute()) {
        // La réservation a été effectuée avec succès
        $_SESSION['message'] = "Le livre a été réservé avec succès.";
        header("Location: reservation.php");
        exit();
    }
}

```

Maintenant, le code ci-dessous est le code du bouton supprimer ce qui permet de réaliser un delete depuis la table emprunt.

```
// Traitement pour la suppression d'un livre réservé par l'utilisateur
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['supprimer_emprunt'])) {
    // Vérifier si l'utilisateur est connecté
    if (isset($_SESSION['user_id'])) {
        // Récupérer la référence du livre à supprimer
        $ref_livre = $_POST['supprimer_emprunt'];

        // Requête SQL pour supprimer l'emprunt de la base de données
        $requete_suppression = "DELETE FROM emprunt WHERE ref = :ref_livre AND num_adherent = :user_id";
        $stmt_suppression = $connexion->prepare($requete_suppression);
        $stmt_suppression->bindValue(':ref_livre', $ref_livre, PDO::PARAM_STR);
        $stmt_suppression->bindValue(':user_id', $user_id, PDO::PARAM_INT);

        // Exécuter la requête SQL pour supprimer l'emprunt
        if ($stmt_suppression->execute()) {
            // Emprunt supprimé avec succès
            $_SESSION['message'] = "L'emprunt a été supprimé avec succès.";
        } else {
            // Erreur lors de la suppression de l'emprunt
            $_SESSION['error_message'] = "Une erreur s'est produite lors de la suppression de l'emprunt.";
        }

        // Rediriger l'utilisateur vers la page de réservation
        header("Location: reservation.php");
        exit(); // Assurez-vous de terminer le script après la redirection
    } else {
        // L'utilisateur n'est pas connecté, redirigez-le vers la page de connexion
        $_SESSION['error_message'] = "Vous devez vous connecter avant de supprimer un livre réservé.";
        header("Location: connexionUtilisateur.php");
        exit();
    }
}
```

Affichage des livres par ordre alphabétique dans la liste que la bibliothèque propose. Ainsi que tous les boutons de filtres applicables et de case de textes à rentrer.

```
<div class="bloc-reservation-recherche">
    <form method="post" action="reservation.php">
        <div>
            <select id="bouton-filtre" name="categorie">
                <option value="test" disabled selected>Catégorie</option>
                <?php
                    $listeCate = $connexion->query("SELECT * FROM categorie ORDER BY libelle");

                    while ($cate = $listeCate->fetch()) {
                        echo '<option value="' . $cate["id"] . '">' . $cate["libelle"] . '</option>';
                    }
                ?>
            </select>
            <button type="submit" id="afficherLivre">Afficher les livres par catégorie</button>
        </div>
    </form>
    <form method="post" action="reservation.php">
        <div class="search-container">
            <input class="text-recherche" type="text" placeholder="Rechercher un livre" name="search">
            <br>
            <button type="submit" id="afficherLivre">Rechercher par nom</button>
        </div>
    </form>
```

Partie du code qui va afficher les informations des livres, ainsi que les boutons qui renvoient vers emprunts réalisés sur le livre, et aussi le bouton qui permet la réservation du livre.

```

<div class="bloc-reservation-recherche-grid">
    <?php if ($_SERVER['REQUEST_METHOD'] == "POST") : ?>
        <?php if ($stmt !== null && $stmt->rowCount() > 0) : ?>
            <?php while ($livre = $stmt->fetch()) : ?>
                <div class="grid-item">
                    " height="200px" alt="<?php echo $livre["titre"]; ?>">
                    <div class="livre-details">
                        <p>IBAN: <?php echo $livre["iban"]; ?></p>
                        <p>Titre: <?php echo $livre["titre"]; ?></p>
                        <p>Auteur: <?php echo $livre["auteur"]; ?></p>
                        <form method="post" action="">
                            <input type="hidden" name="ref_livre" value="<?php echo $livre["ref"]; ?>">
                            <button class="button-traitement" name="reserver" value="<?php echo $livre["ref"]; ?>">Réserver</button>
                        </form>
                        <a href="infos-livres.php?ref=<?php echo $livre["ref"]; ?>">Historique d'emprunt</a>
                    </div>
                </div>
            <?php endwhile; ?>
        <?php elseif ($stmt !== null && $stmt->rowCount() == 0) : ?>
            <p>Aucun livre trouvé.</p>
        <?php elseif (!isset($_POST['categorie'])) : ?>
            <p>Aucune catégorie n'a été sélectionnée.</p>
        <?php endif; ?>
    <?php endif; ?>
</div>

</div>

```

Enfin, partie du code dans la bibliothèque des livres réservés par l'utilisateur. Toujours sécurisé par les requêtes préparées.

```

<div class="bloc-reservation-biblio">
    <div id="text-ma-biblio">
        <p>Ma bibliothèque de livres réservés pour <?php echo isset($user_username) ? $user_username : '' ; ?></p>
    </div>

    <div class="bloc-reservation-biblio-grid">
        <?php
        // Vérifier si la variable de session user_id est définie
        if (isset($_SESSION['user_id'])) {
            $user_id = $_SESSION['user_id'];

            // Requête SQL pour récupérer les livres réservés par l'utilisateur
            $requete_reservations = "SELECT l.* FROM livre l INNER JOIN emprunt e ON l.ref = e.ref WHERE e.num_adherent = :user_id";
            $stmt_reservations = $connexion->prepare($requete_reservations);
            $stmt_reservations->bindValue('user_id', $user_id, PDO::PARAM_INT);
            $stmt_reservations->execute();

            // Afficher les livres réservés
            while ($livre_reserve = $stmt_reservations->fetch()) {
                echo '<div class="grid-item">
                    
                    <div class="livre-details">
                        <p>' . $livre_reserve["titre"] . '</p>
                        <p>Auteur: ' . $livre_reserve["auteur"] . '</p>
                        <form method="post" action="">
                            <input type="hidden" name="ref_livre" value="' . $livre_reserve["ref"] . '">
                            <button class="button-traitement" name="supprimer_emprunt" value="' . $livre_reserve["ref"] . '">'Supprimer emprunt</button>
                        </form>
                    </div>
                </div>';
            }
        }
    </div>;

```