Projet 2

Application de gestion d'appartements

10% de la note finale
Remise le mardi 8 avril

I. Description du projet

`BugCatcher` est une application web permettant aux utilisateurs de signaler des bogues rencontrés dans des logiciels ou applications. Les utilisateurs doivent s'inscrire et se connecter pour soumettre leurs rapports de bogues, et peuvent ensuite gérer leurs soumissions via un espace personnel.

L'application doit permettre:

- Aux visiteurs de créer un compte (inscription)
- Aux utilisateurs enregistrés de se connecter à leur compte
- Aux utilisateurs connectés de soumettre des rapports de bogues
- Aux utilisateurs connectés de consulter, modifier et supprimer leurs propres rapports
- Une validation des formulaires côté serveur (PHP)

A. Récits utilisateurs

- En tant que visiteur, je veux pouvoir créer un compte utilisateur en fournissant mon email et un mot de passe sécurisé, afin d'accéder aux fonctionnalités de l'application.
- En tant qu'utilisateur enregistré, je veux pouvoir me connecter à mon compte avec mon email et mon mot de passe, afin d'accéder à mon espace personnel.

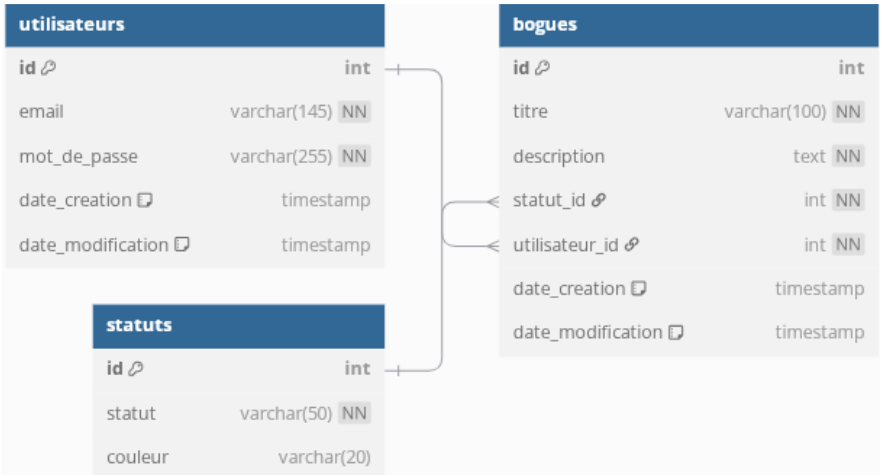
- En tant qu'utilisateur connecté, je veux pouvoir soumettre un rapport de bogue avec une description détaillée, afin d'informer les développeurs des problèmes rencontrés.
- En tant qu'utilisateur connecté, je veux pouvoir consulter la liste de tous mes rapports de bogues soumis précédemment, afin de suivre leur évolution.
- En tant qu'utilisateur connecté, je veux pouvoir modifier mes rapports de bogues existants, afin d'ajouter des informations ou corriger des erreurs.
- En tant qu'utilisateur connecté, je veux pouvoir supprimer mes rapports de bogues, afin de retirer ceux qui ne sont plus pertinents.

B. Technologies utilisées

- Environnement de développement : WAMP (Windows, Apache, MySQL, PHP)
- PHP : Version 7.4 ou supérieure pour le traitement côté serveur
- MySQL : Gestion de la base de données
- phpMyAdmin : Interface d'administration de la base de données
- HTML5/CSS3 : Structure et présentation des pages
- Bootstrap 5 : Framework CSS pour un design responsive et moderne

C. Structure de la base de données

La base de données est structurées de la façon suivante:



II. Mise en place

D. Installation

Téléchargez le projet **`projet2`** sur Lea contenant la structure de base du projet et décompressez-le dans le dossier **`www`** de Wamp sur votre ordinateur. Ce dossier comprend :

- L'arborescence des dossiers
- Les fichiers de configuration de base
- Le script SQL pour initialiser la base de données
- Les templates des pages à compléter

Après avoir démarrer Wamp, pointez le navigateur à . Vous devriez voir la page d'accueil de l'application :

E. Création de la base de données

Vous devez créer la base de données à l'aide de phpMyAdmin (onglet SQL) en exécutant le script **`creer_bd_bugcatcher.sql`**.

1. Ouvrir phpMyAdmin et cliquer **`Nouvelle base de données`** dans le panneau de gauche.
2. Cliquer l'onglet **`SQL.`**
3. Copier-coller le script contenu dans **`creer_bd_bugcatcher.sql.`**
4. Cliquer **`Exécuter.`**

Suite à l'exécution de la requête, vérifier à l'aide de phpMyAdmin que toutes les tables de la base de données **`bug_catcher`** ont été créées.

III. Développement des fonctionalités

F. Fonctionnalité - Page d'accueil

La page d'accueil constitue le point d'entrée principal de l'application **BugCatcher**. Elle offre un accès rapide aux principales fonctionnalités selon que l'utilisateur soit connecté ou non.

- Page: **`index.php`**

Exigences fonctionnelles

1. Pour tous les visiteurs

La page d'accueil affiche :

- Un titre de bienvenue présentant clairement le nom et l'objectif de l'application BugCatcher
- Une description concise expliquant l'utilité de l'application pour signaler des bogues

2. Pour les visiteurs non connectés

La page doit présente :

- Un message invitant à créer un compte ou à se connecter pour accéder aux fonctionnalités
- Des boutons/liens clairement visibles pour :
    - Se connecter à un compte existant
    - Créer un nouveau compte

3. Pour les utilisateurs connectés

La page présente :

- Un message de bienvenue personnalisé incluant l'email de l'utilisateur
- Des boutons/liens clairement visibles pour :
    - Accéder à la liste des bogues déjà signalés
    - Signaler un nouveau bogue

4. Contraintes techniques

- La page doit détecter automatiquement si l'utilisateur est connecté ou non pour afficher le contenu approprié
- Les messages flash de confirmation ou d'erreur provenant d'autres actions (comme une connexion/déconnexion récente) doivent être correctement affichés

5. Conseils de développement

- Utilisez la fonction `est_connecte()` pour détecter l'état de connexion de l'utilisateur

G. Fonctionnalité - Page de connexion

La page de connexion permet aux utilisateurs déjà inscrits d'accéder à leur compte sur l'application BugCatcher afin de consulter et gérer leurs bogues signalés.

- Page: `views\enregistrement\connexion.php`

Exigences fonctionnelles

6. Formulaire de connexion

Le formulaire doit contenir les champs suivants :

- **Email** (obligatoire)
- **Mot de passe** (obligatoire)
- Bouton de soumission clairement identifié

7. Validation des données

La validation doit être effectuée côté serveur en PHP et doit vérifier :

- Que tous les champs obligatoires sont remplis
- Que les identifiants correspondent à un compte existant dans la base de données

- Que le mot de passe fourni correspond au hachage stocké en base de données

8. Traitement de la connexion

Après validation réussie :

- Une session utilisateur doit être créée avec au minimum :
  - L'identifiant de l'utilisateur (id_utilisateur)
  - L'email de l'utilisateur
- L'identifiant de session doit être régénéré pour des raisons de sécurité
- L'utilisateur doit être redirigé vers la page d'accueil avec un message de bienvenue

9. Gestion des erreurs

En cas d'erreur de validation ou d'authentification :

- Le formulaire doit être réaffiché
- L'email précédemment saisi doit être conservé (mais pas le mot de passe)
- Des messages d'erreur appropriés doivent être affichés :
  - Message général pour authentification échouée
  - Messages spécifiques pour champs manquants

10. Sécurité

- Les mots de passe ne doivent jamais être stockés en clair en base de données
- Les identifiants incorrects ne doivent pas révéler lequel des deux (email ou mot de passe) est erroné

11. Contraintes techniques

- Si l'utilisateur est déjà connecté, il doit être automatiquement redirigé vers la page d'accueil
- Créer des fonctions pour minimiser le code php dans la page de connexion.

12. Conseils de développement

- Créer une fonction de validation pour les champs de formulaires
- Créer une fonctions dans **`lib/requetes`** pour trouver l'utilisateur par email dans la base de données

- Créer une fonction dans `lib/fonction` pour gérer l'authentification de l'utilisateur.
- Utilisez la fonction `password_verify()` pour vérifier la correspondance du mot de passe
- Utilisez `session_regenerate_id(true)` après connexion réussie pour prévenir les attaques de fixation de session
- Pour les messages d'erreur d'authentification, utilisez un message générique comme "Email ou mot de passe incorrect" pour ne pas donner d'indices aux attaquants
- Pour afficher un message d'erreur de champs en rouge, utilisez la classe *Bootstrap* `class="text-danger"`

H. Fonctionnalité - Page d'inscription

La page d'inscription permet aux nouveaux utilisateurs de créer un compte sur l'application BugCatcher afin de pouvoir signaler et suivre des bogues. Cette étape est obligatoire avant de pouvoir utiliser les fonctionnalités principales de l'application.

- Page: `views\enregistrement\inscription.php`

Exigences fonctionnelles

13. Formulaire d'inscription

Le formulaire doit contenir les champs suivants :

- **Email** (obligatoire)
- **Mot de passe** (obligatoire)
- **Confirmation du mot de passe** (obligatoire)
- Bouton d'inscription clairement identifié

14. Validation des données

La validation doit être effectuée côté serveur en PHP et doit vérifier :

- Que tous les champs obligatoires sont remplis
- Que l'email est dans un format valide (utiliser les fonctions de validation d'email de PHP)
- Que l'email n'est pas déjà utilisé par un autre compte dans la base de données
- Que le mot de passe respecte une longueur minimale de 8 caractères
- Que la confirmation du mot de passe correspond exactement au mot de passe saisi

15. Traitement de l'inscription

Après validation réussie :

- Le mot de passe doit être haché de manière sécurisée avant d'être stocké
- Les informations de l'utilisateur doivent être enregistrées dans la base de données

- L'utilisateur doit être redirigé vers la page de connexion avec un message de confirmation

16. Gestion des erreurs

En cas d'erreur de validation :

- Le formulaire doit être réaffiché
- Les valeurs précédemment saisies doivent être conservées (sauf les mots de passe)
- Des messages d'erreur spécifiques doivent être affichés à côté de chaque champ problématique

17. Contraintes techniques

- Si l'utilisateur est déjà connecté, il doit être automatiquement redirigé vers la page d'accueil
- La page doit inclure un lien vers la page de connexion pour les utilisateurs déjà inscrits
- L'interface doit être responsive et s'adapter aux différentes tailles d'écran

18. Conseils de développement

- Utilisez la fonction `password_hash()` avec l'algorithme PASSWORD_DEFAULT pour hacher les mots de passe
- Vérifiez l'unicité de l'email avant de créer un nouveau compte
- Utiliser la fonction php **`filter_var`** pour valider le format du email

I. Fonctionnalité - Ajout d'un nouveau bogue

Cette fonctionnalité permet à un utilisateur connecté de signaler un nouveau bogue dans le système. L'utilisateur doit compléter un formulaire détaillant le problème rencontré.

- Page: **`views/bogues/nouveau.php`**

Contraintes

- Seuls les utilisateurs authentifiés peuvent accéder à cette fonctionnalité.
- Les utilisateurs non connectés doivent être redirigés vers la page de connexion.
- La validation des données doit être effectuée côté serveur avec PHP (pas de validation JavaScript).

Exigences fonctionnelles

19. Formulaire de création

Le formulaire doit contenir les champs suivants :

- **Titre du bogue** (obligatoire)
  - Minimum 5 caractères
  - Maximum 100 caractères
  - Doit être descriptif et résumer clairement le problème
- **Description détaillée** (obligatoire)
  - Minimum 10 caractères
  - Doit permettre de comprendre le problème et les étapes pour le reproduire

20. Validation

La validation doit être effectuée en PHP et doit vérifier :

- Que tous les champs obligatoires sont remplis
- Que les contraintes de longueur sont respectées
- Que l'utilisateur est bien connecté

21. Traitement des données

Après une validation réussie :

- Le nouveau bogue doit être enregistré dans la base de données
- Le statut initial du bogue doit être défini comme "Nouveau" (statut_id = 1)
- L'identifiant de l'utilisateur connecté doit être associé au bogue
- La date de création doit être automatiquement enregistrée

22. Messages et redirection

- En cas d'erreur de validation, le formulaire doit être réaffiché avec :
  - Les valeurs précédemment saisies
  - Des messages d'erreur spécifiques à côté de chaque champ problématique

- En cas de succès :
  - L'utilisateur doit être redirigé vers la liste de ses bogues.
  - Un message de confirmation doit indiquer que le bogue a été créé avec succès (utiliser `$_SESSION["flash"]` pour stocker le message avant le redirection).

23. Conseils de développement

- Assurez-vous que vous récupérer toutes les entrées utilisateur de façon sécuritaire
- Vérifiez toujours que l'utilisateur est connecté au début de la page avec `est_connecte()`
- Ajoutez une valeur par défaut pour le statut dans la requête d'insertion
- Lors du traitement du formulaire, vérifiez la longueur du titre et de la description
- Stockez l'ID de l'utilisateur connecté comme propriétaire du bogue
- N'oubliez pas de définir correctement les codes d'erreur HTTP en cas de problème
- Utilisez les messages flash pour confirmer la création réussie après redirection

J. Fonctionnalité - Liste des bogues

Cette fonctionnalité permet à un utilisateur connecté d'afficher la liste de tous les bogues qu'il a signalés. L'utilisateur peut visualiser rapidement le statut de ses bogues et accéder aux détails de chacun d'entre eux.

- Page: `views/bogues/liste.php`

Contraintes

1. Seuls les utilisateurs authentifiés peuvent accéder à cette fonctionnalité
2. Les utilisateurs non connectés doivent être redirigés vers la page de connexion
3. L'utilisateur ne peut voir que ses propres bogues

Exigences fonctionnelles

24. Affichage de la liste

La page doit afficher :

- Un titre clair indiquant qu'il s'agit de la liste des bogues de l'utilisateur
- Un bouton permettant de créer un nouveau bogue
- Une liste de tous les bogues soumis par l'utilisateur, triés par date de création (du plus récent au plus ancien)

25. Informations à afficher pour chaque bogue

Pour chaque bogue dans la liste, les informations suivantes doivent être visibles :

- Le titre du bogue
- La date de création (format : jour/mois/année et heure)
- Un aperçu de la description (limité à environ 100 caractères)
- Le statut actuel du bogue (avec un indicateur visuel selon le statut)
- Un lien ou bouton pour accéder aux détails complets du bogue

26. Gestion des cas particuliers

- Si l'utilisateur n'a pas encore signalé de bogues, un message informatif doit être affiché
- Si une action vient d'être effectuée (création, modification ou suppression d'un bogue), un message de confirmation approprié doit apparaître

27. Interaction

- Le clic sur un bogue ou sur le bouton "**Voir détails**" doit rediriger l'utilisateur vers la page de détails du bogue correspondant
- Le clic sur le bouton "**Nouveau bogue**" doit rediriger vers le formulaire de création d'un bogue

28. Conseils de développement pour la liste des bogues

- Utilisez la fonction `connexion_requise()` au début de la page pour vérifier l'authentification
- Récupérez les bogues avec une jointure entre les tables `bogues` et `statuts` pour obtenir les informations sur le statut
- Pour l'affichage condensé de la description, si nécessaire utilisez la fonction `substr()` pour limiter le nombre de caractères
- Formatez les dates avec la fonction `date()` pour une meilleure lisibilité
- Utilisez des badges colorés pour rendre les statuts visuellement distincts
- Contrôlez l'accès pour que les utilisateurs ne puissent voir que leurs propres bogues
- Exploitez les messages flash pour confirmer les actions (création, modification, suppression)
- Testez votre code avec différents scénarios : aucun bogue, un seul bogue, plusieurs bogues

K. Fonctionnalité - Afficher et modifier un bogue

Cette fonctionnalité permet à un utilisateur connecté de consulter les détails d'un bogue spécifique, de modifier ses informations et éventuellement de le supprimer. C'est une page centrale qui combine l'affichage, l'édition et la suppression en une seule interface.

Contraintes

1. Seuls les utilisateurs authentifiés peuvent accéder à cette fonctionnalité
2. Un utilisateur ne peut voir et modifier que ses propres bogues
3. L'accès à un bogue qui n'existe pas ou qui appartient à un autre utilisateur doit être refusé
4. La validation des données doit être effectuée côté serveur avec PHP

Exigences fonctionnelles

29. Affichage des détails

La page doit afficher :

- Le titre du bogue
- La description complète
- Le statut actuel (avec un indicateur visuel)
- La date de création
- La date de dernière modification
- Des options pour modifier ou supprimer le bogue

30. Modification du bogue

L'utilisateur doit pouvoir modifier :

- Le titre du bogue
- La description détaillée
- Le statut du bogue (via un menu déroulant)

La validation doit vérifier :

- Que tous les champs obligatoires sont remplis
- Que les contraintes de longueur sont respectées (mêmes règles que lors de la création)

31. Suppression du bogue

La page doit permettre de supprimer le bogue :

- Une confirmation doit être demandée avant la suppression définitive
- Après suppression, l'utilisateur doit être redirigé vers la liste des bogues

32. Messages et redirection

- En cas d'erreur de validation, le formulaire doit être réaffiché avec :
  - Les valeurs précédemment saisies
  - Des messages d'erreur spécifiques à côté de chaque champ problématique
- En cas de succès :
  - Pour une modification : afficher un message de confirmation
  - Pour une suppression : rediriger vers la liste des bogues avec un message de confirmation

33. Conseils de développement

- Utilisez l'ID du bogue passé dans l'URL (paramètre GET) pour récupérer ses informations
- Vérifiez toujours que l'utilisateur connecté est bien le propriétaire du bogue
- Pour la suppression, utilisez une boîte de dialogue modale Bootstrap pour la confirmation
- Assurez-vous que tous les champs du formulaire sont pré-remplis avec les valeurs actuelles du bogue

IV. Pondération et échéance

- Ce projet compte pour 10 % de la note finale, mais plus important encore, il vous prépare à l'examen 2.

- Date de remise: le mardi 8 avril en fin de journée.

    - Une pénalité de 10 % par jour de retard, incluant les jours de congé, sera imposée, et ce, jusqu'à concurrence de 5 jours. (Après ce délai, la note attribuée est zéro.)
    - Fichier à remettre (pénalité jusqu'à 40 % si consigne non respectée)

      Remettre le fichier compressé `projet2_nom_famille.zip` via LÉA. Vous devrez vous assurer que tous les fichiers nécessaires sont présents sans quoi vous serez pénalisés.

V. Barème de correction

- Code (fiabilité, validité, conformité aux spécifications) 80 %

- Style (commentaires, normes de programmation, clarté et disposition du code) 20%

VI. Consigne

L. Style et conventions

- Vous êtes libres d'utiliser la convention de nommage ***`snake case`*** ou ***`camel case`***, à condition que son utilisation soit cohérente à travers l'application. Vous ne devez pas alterner entre les deux.

M. Modularité

Votres application doit être modulaire et respecter les principes suivants.

- Chaque fichier doit avoir une responsabilité.
- Structurez votre projet en plusieurs dossiers clairement structurés.
- L'objectif est de mimimiser les code php à insérer dans le html.
- Utilisez des fonctions php dans un fichier commun

N. Documentation

1. Créer un fichier **`README.md`** avec le contenu suivant:

    a. Nom du projet : Expliquer brièvement son objectif.
    b. Auteurs : Indiquer les étudiants responsables du projet

c. Technologies utilisées : Expliquer les outils et langages employés.
d. Installation et exécution : Instructions claires pour exécuter le projet.
e. Fonctionnalités principales : Explication des modules clés.

2. Commentaires dans le Code
   a. Chaque fichier doit inclure :
      - Un en-tête expliquant le fichier et son rôle.

      Ex fictif.:

```
/**

gestion.php - Gestion des locataires

Ce fichier permet d'afficher, ajouter et modifier des locataires.

@author Dom
@date 2024-02-10

*/
```

   b. Des commentaires de fonction décrivant les paramètres, les valeurs retournées et le fonctionnement.

      Ex. :

```
/**
Ajoute un locataire dans la base de données.
@param string $nom Nom du locataire
@param string $email Email du locataire
@return bool Retourne true si l'ajout est réussi, false sinon

*/

function ajouterLocataire($nom, $email) {...}
```