7.3 Développement de la fonctionnalité de validation des fiches de frais.

Vous devez travailler en binôme (l'un des groupes sera éventuellement composé de 3 personnes si vous êtes un nombre impair en SLAM). Demandez des précisions à votre chef de projet.
Le module AFC est en cours de réalisation, vous allez poursuivre son développement.

Commencez par récupérer le projet. Demandez à votre chef de projet la marche à suivre.

Attention, le projet doit être impérativement développé en utilisant NetBeans.

Une fois le projet récupéré, et avant même de vous lancer dans le développement, il vous faut comprendre le fonctionnement de l'existant. Le module AFC est réalisé à partir du code du module AFV (Application Frais Visiteurs) destiné à la saisie des fiches de frais par les visiteurs. Il s'agit d'une application qui repose sur l'architecture MVC.

7.3.1 Etape 1 - Compréhension du fonctionnement de l'existant

Répondez aux questions suivantes en mobilisant vos connaissances en développement et en vous aidant du débogueur de votre IDE et de la documentation des logiciels et langages qui vous semblent nécessaires.

1. Quel est le programme de démarrage de l'application ?
2. Quel est le rôle d'un tel programme dans l'architecture MVC ?
3. A quoi servent les deux fichiers incluent dans ce fichier de démarrage via l'instruction `require_once()` ?
4. Qu'indique les extensions `.inc.php` dans le nom de ces deux fichiers ?
5. A quoi servent les fichiers **`v_entete.php`** et **`v_pied.php`** inclus dans la suite du fichier de démarrage (ont-ils un rapport avec l'architecture MVC) ?
6. Quel est le rôle de l'instruction `$pdo = PdoGsb::getPdoGsb();` et pourquoi n'a-t'on pas écrit plutôt `$pdo = PdoGsb->getPdoGsb();` (vous trouverez la solution à ce problème en consultant le site `https://www.php.net/manual/fr/language.oop5.php`) ?
7. Comment l'application détermine-t'elle qu'un utilisateur est déjà connecté ?
8. A quoi sert le fichier **c_connexion.php** inclus dans le bloc `switch` (a-t'il un rapport avec l'architecture MVC) ?

9. Comment le programme de démarrage de l'application sait-il à quel contrôleur il faut faire appel ?

10. Comment le programme **c_connexion.php** sait-il quelle action il faut exécuter ?

11. Quelle(s) action(s) le programme **c_connexion.php** prend-il en compte ?

12. Comment le programme **c_connexion.php** traite-t'il une demande de connexion ?

13. Comment le programme **c_connexion.php** obtient'il les informations sur l'utilisateur qui essaye de se connecter afin de vérifier que son login et son mot de passe sont corrects ?

14. Quel rôle joue ici la classe `PdoGsb` contenue dans le fichier `class.pdogsb.inc.php` par rapport à l'architecture MVC ?

15. Si les informations de connexion sont invalides, comment le programme **c_connexion.php** affiche-t'il un message d'erreur et de nouveau le formulaire de connexion ?

16. Dans la fonction ajouterErreur() que fait l'instruction `$_REQUEST['erreurs'][]=$msg;` ?

17. Comment le programme **c_connexion.php** affiche-t'il le sommaire de l'application ?

18. Dans le fichier `c_connexion.php` il est écrit `include("vues/v_sommaire.php");`. Pourtant, le fichier `c_connexion.php` est stocké dans le dossier "`Controleurs`". Donc pour inclure le fichier `v_sommaire.php` du dossier "Vues" dans le fichier `c_connexion.php` du dossier "`Controleurs`" il aurait fallu écrire "`include(../vues/v_sommaire.php);`". Hors, ici l'instruction sans les "`..`" fonctionne parfaitement. Pourquoi ?

19. Quelle(s) affirmation(s) est(sont) vraie(s) :
    a - Le fichier index.php est exécuté une seule fois au début de l'application.
    b - Le fichier index.php est exécuté systématiquement à chaque demande de l'utilisateur.
    c - Le fichier index.php n'est exécuté que lors de l'ouverture et de la fermeture d'une session.

20. Dans l'architecture MVC, comment appelle-t'on un fichier tel que `index.php` ?