|  |
| --- |
| Documentation Utilisateur  De l’application JAVA pour la gestion des classes |

|  |
| --- |
| Par Amine NAKHIL, Loïc GUO et Quentin LIGNANI |



1. Connexion
2. Page D’accueil
3. Page « Evénements »
4. Page « Offres d’emplois »
5. Profil

1.Connexion



**Dès le lancement du programme, on accède à une page de Connexion pour entrer nos identifiants, le programme va également évaluer notre niveau d’évaluation en vue de nous rediriger de manière appropriée.**

2.Page D’accueil

2.1 Vue Administrateur



**Après la Connexion et l’éventuel constat d’une habilitation administrateur, l’utilisateur tombe sur cette page. L’administrateur voit s’afficher la liste des noms des élèves. S’il clique sur le nom d’un élève en particulier, il s’affiche les nom et prénom de l’élève en gros. En plus, cela permet de sélectionner un élève pour lui assigner une absence sur le bouton en haut à droite.**

2.2 Vue Utilisateur



**Si l’utilisateur qui se connecte n’est pas habilité administrateur, il tombe sur cette page. Une salutation lui est adressé en haut et à gauche y figure la liste des classes du professeur. Au centre, c’est l’emploi du temps hebdomadaire du professeur qui précise les créneaux pendant lesquels il exerce et avec laquelle de ses classes. En bas à gauche, figure un bouton pour que le professeur modifie les informations de son profil.**

4.Modifier ses informations



**Quand sur sa page d’accueil, le professeur clique sur le bouton en bas à gauche « Modifier son profil », il tombe sur la page ci-dessus. Il y a quatre formulaires pour renseigner les informations de son profil que l’on peut modifier à savoir le Nom, le Prénom, L’adresse électronique et la matière. Puis un formulaire pour la validation qui sers à applique ces changements. Les informations d’origines sont pré-renseignées dans les formulaires. Une fois le profil modifié, il s’affiche en vert « Modifications enregistrées ». Si un formulaire est NUL, il s’affiche en rouge « Veuillez renseigner tous les champs ».**

5.Page Manager\_User.php



**Cette page étant un Manager dans le cadre de la programmation orientée objet, on importe le model en indiquant l’emplacement du fichier. Comme l’on a à utiliser Php Mailer, on appelle ses extensions.**



**Comme cette page sert à faire le pont en notre page internet et notre base de données, nous mettons des variables avec le même nom que ce que l’on doit avoir à traiter comme informations. On commence ce code en ouvrant la classe et en la nommant « Manager\_User », n’utiliser qu’une seule classe par fichier, et en nommant le fichier et la classe avec le même nom. Tout ceci dans le plus strict respect des conventions. On fonde quatre attributs à savoir « nom », « prénom », « email » et « mdp ». Ces attributs sont en privé, ils ne sont pas accessibles en dehors de la présente classe.**



**On ouvre une méthode en public, on la nomme « inscription », elle sert donc à l’inscription. On y met une variable interne nommée « inscrit ». Dans cette méthode, on appelle la classe « User », qui représente le model. Dans cette méthode, on utilise l’interface PHP Data Objects (PDO). Dans la première ligne de l’utilisation de PDO, on instancie un nouvel objet : PDO, on informe que l’on travaille en local, le nom de la base de données, en l’occurrence « projet\_lycee », puis on communique les identifiants avec lesquels on interagit avec la base de données, ici avec le nom d’utilisateur « root » et avec un mot de passe blanc. Au sein de la ligne suivante, on prépare la requête qui va interroger la base de données dans laquelle on communique la requête en langage SQL. Notre commande permet d’afficher dans la table « utilisateur » toutes les lignes parmi lesquelles la variable « email » est semblable à la colonne « email » de la base. L’avant dernière étape de cette utilisation de PDO est d’exécuter la commande dans un tableau, on demande d’obtenir ce que contient la ligne dont « email » est semblable, on obtient donc l’adresse email. Pour que cette commande préalablement peaufinée soit appliquée, il faut utiliser la commande « fetch » à la dernière ligne. Ainsi, la variable « donnee » doit contenir le contenu de la colonne « email » à la ligne ou l’adresse mail saisie est similaire à elle dans le cas où elle existe.**



**Ensuite, on met une condition. Comme dans la dernière ligne, la variable « donne » est censée contenir l’adresse mail saisie par l’utilisateur si et seulement si elle existe. Ainsi, cette condition permet de vérifier si la variable « donnee » est remplie donc à fortiori si l’adresse mail entrée est déjà associée à un compte. Si cette condition est vérifiée, on affiche sur la page html « L’email est déjà utilisée ». Après cela, on redirige vers la page « inscription.php » pour retenter une connexion.**



**Ce code est une condition inverse, ainsi, il ne s’applique uniquement dans le cas ou la condition précédente n’est pas vérifiée. En clair, cette condition s’applique si l’adresse mail entrée par l’utilisateur n’existe pas déjà. Cette condition sert à l’envoi de mail, on y utilise l’extension PHPMailer. On importe donc les modules qui sont utilisés dans ce cadre à travers des balises « require » qui pointent vers des fichiers par défaut de PhpMailer**.



**Ce code est issu de l’extension « PhpMailer », il sert à renseigner les informations que l’extension exige pour fonctionner. On encode l’affichage en UTF-8 pour faire supporter notamment les caractères spéciaux, on active le protocole SMTP qu’utilise PHPMailer pour envoyer des mails, on active l’authentification , on utilise le transfert sécurisé avec SSL, on renseigne le domaine de mails qui est supporté par notre fournisseur de mails, on renseigne le port que l’on veut utiliser pour envoyer des mails, on renseigne l’adresse mail d’expédition, on renseigne le mot de passe lié à cette adresse, on met dans l’objet du mail que l’on envoie « Création de votre compte réussie », on met le contenu du mail à travers une mise en forme.**



**Ceci est une condition. Cette condition ne s’applique que si le mail n’a pas été envoyé. Comme nos paramètres sont inchangés, cela ne peut que provenir d’une adresse mail invalide. Ainsi, on utilise un message d’erreur pour prévenir l’utilisateur de sa saisie invalide de l’adresse mail. On affiche donc « Erreur dans l’email ». On renvoie vers la page d’inscription pour une nouvelle tentative.**



**Ce code est une condition, il ne s’applique que si la condition précédente n’est pas vérifiée. En clair, ce code ne s’applique que l’adresse mail saisie est valide. On commence par informer l’utilisateur que l’envoi est réussi en affichant « message has been sent ». Ensuite, on utilise PHP Data Object pour interagir avec la base de données. On commence par utiliser la fonction « prepare » comme la syntaxe de PDO l’exige, on y affiche notre commande SQL à savoir « INSERT into utilisateur (nom, prenom, email, mdp) value (?,?,?,?) », cette commande permet d’aller dans la table « utilisateur », et de remplir les colonnes « nom », « prenom », « email », « mdp », avec les valeurs respectives que l’on renseignera après. A la ligne de code suivante, ont émet un tableau associatif dans lequel on relie le getter de chaque attribut renseigné dans le model, avec les valeurs fournies par l’utilisateur. Il est a noté que l’on hache préalablement « mdp » avec l’algorithme SHA1 avant de l’inscrire dans la base de données. On redirige ensuite l’utilisateur vers la page « confirm\_inscription.html ». On clôture la méthode.**



**On ouvre une méthode qui sers à la connexion. On nomme cette méthode « connexion », on déclare l’attribut interne « connexion ». On utilise PDO, on veut interagir avec une base de sonnées. On fonde bien sur un nouvel objet PDO, comme le veut la convention d’utilisation de PDO. On informe que l’on travaille en local, que le nom de la base de données est « projet\_lycee », que le nom d’utilisateur est « root » et que le mot de passe est blanc. Dans la fonction « prepare », on met la requête SQL, à savoir « SELECT \* from utilisateur where email = ? AND mdp = ? ». Cette requête signifie que l’on veut connaitre dans toute les lignes que comporte la table « utilisateur » dans lequel le mot de passe saisi et l’adresse mail saisie correspondent. On utilise la fonction « execute », et on applique avec la fonction « fetch ».**



**On met une condition pour déterminer si la variable « donnée » existe, ce qui revient à ce que l’utilisateur entre une adresse mail ainsi qu’un mot de passe correct. On insère ainsi l’adresse mail et le nom de l’utilisateur dans la session.**



**On imbrique une nouvelle condition qui vérifie si le rôle de l’utilisateur renseigné dans la colonne « role » de la table « utilisateur » de la base de données correspond bien à « admin ». Si tel est le cas, on insère également dans la session le rôle qu’a l’utilisateur. Indépendamment de cette condition, on redirigera l’utilisateur vers la page « index.php ».**



**Si la première condition n’est pas vérifiée, concrètement, si l’utilisateur n’entre pas d’identifiants corrects, on insère dans la session « erreur\_co » le booléen « true ». Ensuite, on le redirige vers la page sign-in.php.**