Documentation Technique

Logiciel permettant la communication entre les enseignants et les tuteurs concernant les travaux effectués avec les enseignants ainsi que les travaux à faire et les interrogations à venir.

IUT Département Informatique

Janvier 2019

Documentation technique

Entrée du programme

index.php

C'est le programme qui lance notre site. Il affiche le contenu de la page de connexion à notre site.

- L'utilisateur doit entrer son identifiant et son mot de passe
- Une vérification est effectuée afin de vérifier qu'un utilisateur ne peut pas se connecter sans ses données.
- Après la vérification, le programme affiche la page d'accueil en incluant la classe accueil.inc.php, sinon, redirige vers la page de connexion en incluant la classe login.inc.php.

__inc.php

C'est un élément de programme qui sert à se connecter à la base de données et à créer une template. Il sert aussi à rediriger vers la page de connexion si quelqu'un essaie d'accéder à la page sans s'être connecté. Il est requis en début de chaque programme.

login.inc.php : programme comparant les éléments entrés dans les zones de texte de la page de connexion et les attributs de la base de donnée. Le programme vérifie qu'une requête "post" est envoyée puis vérifie les éléments qi c'est le cas. Plusieurs vérifications sont effectuées sur ceux-ci : que tous les champs sont remplis, que le login appartient à la base de données puis, que le mot de passe affecté est bien correspondant. Si une des conditions n'est pas remplie, un message d'erreur est passé dans une template qui écrira le message en dessous des champs de la page de connexion.

accueil.inc.php : programme permettant d'afficher la page d'accueil. Il est appelé si un utilisateur a réussi à se connecter.

etat.php

Programme permettant l'affichage des séances en fonction des filtres accessibles depuis cette même page. Pour les filtres, toutes les tables sont récupérées à l'exception de celle des affectations.

- Lorsqu'on arrive pour la première fois sur la page, le seul filtre actif est celui de la date de création des séances. Seules les séances du mois sont affichées, les dates

de création minimales et maximales sont initialisées au premier jour du mois courant et au premier jour du mois suivant.

```
$dateCreaMinMois = new DateTime(); $dateCreaMinMois->setDate($anneeActuelle, $moisActuel+1, 1);
```

- Le mois peut être changé grâce aux boutons “Précédent” et “Suivant” qui indiquent par méthode “POST” au fichier etat.php d’augmenter ou de diminuer le numéro du mois courant de 1 et de changer d’année si nécessaire. Ces boutons sont identifiée en tant que “month-” et “month+”, ce qui permet de vérifier s’ils ont été utilisés grâce à la méthode “isset()”.

```
if (isset($_POST['month+'])) {
        $moisActuel = $moisActuel+1;
}
if (isset($_POST['month-'])) {
        $moisActuel = $moisActuel-1;
}
```

- Les filtres peuvent être changés. L’utilisateur peut appuyer sur un bouton lançant une fonction JavaScript afin d’afficher les filtres cachés de façon à ne pas surcharger visuellement la page. Une fois les filtres choisis, le bouton filtrer permet de recharger la page en envoyant les données sélectionnées en méthode POST au script PHP. Les variables contenant les filtres, avant initialisées à vide, sont alors remplies avec ces données grâce à la méthode “**isset** (mixed $var [, mixed $...]) : bool”.

```
$tabModules = array();  //tableau contenant les modules sélectionnés
...
if (isset($_POST['Modules']))
        $tabModules = $_POST['Modules'];
```

- Le filtrage se fait grâce à une boucle parcourant toutes les séances de la base de données. Un if dans cette boucle permet de ne remplir le tableau filtré qu’avec les séances correspondant aux critères renseignés. Si l’un des filtres n’est pas renseigné (la variable est donc à vide), alors cette partie de la condition renvoie “true” de façon à ce que ce filtre ne soit pas pris en compte.

```
if ( ... && (in_array($s->getModule(), $tabModules ) || empty($tabModules )) && ...)

        array_push($seancesFiltrees, $s);   //avec $s la séance traitée
```

- Les évènements subissent le même traitement dans une boucle parcourant les évènements elle-même dans une boucle parcourant les séances précédemment filtrées. Les évènements parcourus sont ceux correspondant aux séances traitées par la boucle, de façon à ne pas parcourir tous les évènements de la base de données à chaque fois. Les évènements correspondant aux filtres sont placés dans un attribut de l'objet de type séance correspondant de façon à pouvoir être récupérés, envoyés et utilisées dans la template.

```
if (...)

        array_push($seance->evenements, $e);     /*avec $seance->evenements le tableau contenant les évènements filtrés*/
```

- Afin de ne pas surcharger visuellement la page, les évènements ne sont pas affichés de base. L'utilisateur peut appuyer sur un bouton lançant une fonction JavaScript afin de les afficher.
- Un tableau associatif est utilisé afin de regrouper les séances par numéro de semaine pour une meilleure lisibilité. Pour cela, on parcourt toutes les séances filtrées. Ensuite, on vérifie si la séance traitée est dans le tableau associatif. Si elle ne l'est pas, on récupère le numéro de semaine correspondant grâce à la méthode "public **DateTime::format** (string $format) : string". On récupère ensuite tous les évènements qui correspondent à ce numéro de semaine dans un tableau, puis on ajoute au tableau associatif le couple (numéro de semaine ; séances correspondantes). C'est ce tableau qui est ensuite envoyé à la template.

```
foreach ($seancesFiltrees as $seance) {

        if (!array_key_exists(date($seance->getDateCreation()), $tabSeancesFiltrees)) {

                $date = new DateTime(date($seance->getDateCreation()));

                $numSemaine = $date->format("W");

                $seancesSemaine = array();

                foreach ($seancesFiltrees as $s) {

                $dateSeance = new DateTime(date($s->getDateCreation()));

                $numSemaineSeance = $dateSeance->format("W");

                if ($numSemaineSeance == $numSemaine) {
```

```
                array_push($seancesSemaine, $s);
            }
            }
            $tabSeancesFiltrees[$numSemaine] = $seancesSemaine;
        }
    }
```

- Les évènements et séances sont affichés grâce à trois boucles imbriquées dans le fichier contenant la template de la page. La première sert à créer un block contenant le numéro de la semaine et toutes les séances affectées. La deuxième sert à afficher les séances. La troisième sert à afficher les évènements associés à chaque séance.

```
{% for numSemaine,seances in tabSeances %}
    ...
    {% for s in seances %}
        ...
        {% for e in s.evenements %}
        ...
        {% endfor %}
    {% endfor %}
{% endfor %}
```

- Ces boucles permettant l'affichage des séances et évènements sont contenues dans une balise "<form>" permettant à l'appui sur le bouton "enregistrer le marquage" de sauvegarder l'état des sémaphores, qui sont des checkboxes personnalisées. Leur état est envoyé par méthode "POST" au fichier "etat.php" qui s'occupera alors de sauvegarder ces états dans la base de données grâce à la méthode "updateSemaphore($id_seance, $id_utilisateur,$etat)", $etat étant "t" si la checkbox est cochée et "f" si elle ne l'est pas. Pour vérifier cela, on vérifie si le sémaphore traité est dans la liste envoyée par la méthode "POST".

```
if (in_array($semaphore->getSeance(), $_POST['sem'])) {
    $db->updateSemaphore($semaphore->getSeance(), $_SESSION['login'], "t");
}
```

```
else {
        $db->updateSemaphore($semaphore->getSeance(), $_SESSION['login'], "f");
}
```

profil.php

C'est le programme qui lance la page de modification du profil de l'utilisateur. Il envoie les données sur une template appelée "profil.twig" qui comprend deux formulaires portant un identifiant différents (nom et mdp). Au début du programme, l'identifiant est récupéré afin d'éxécuter le script correspondant.

- Si l'identifiant est "nom", alors l'action effectuée sera une modification des données personnelles de l'utilisateur.
- Si il s'agit de "mdp", alors l'utilisateur doit remplir plusieurs conditions telles que d'indiquer le mot de passe actuel, le nouveau mot de passe et de confirmer celui-ci.
- Le programme empêche de copier/coller le nouveau mot de passe sur les champs de confirmation.
- Une vérification est effectuée pour être sûr que le nouveau mot de passe et la confirmation du mot de passe sont les mêmes avec la fonction strcmp($_POST['nmdp'],$_POST['nmdp2']) === 0.
- Le programme vérifie que le mot de passe actuel est correct avec la fonction password_verify($_POST['omdp'], $user->getMdp()), omdp représentant le mot de passe actuel.
- Puis, le programme vérifie que le mot de passe contient au moins 8 cacarctères(strlen($_POST['nmdp']) >= 8, nmdp représentant le nouveau mot de passe)
- Qu'il contient au moins 2 lettres majuscules (ctype_alpha($str[$i]) && ctype_upper($str[$i])), $str représentant me nouveau mot de passe et $i l'index de la chaine de caractères.
- Qu'il contient au moins 2 lettres minuscules avec la méthode (ctype_alpha($str[$i]) && ctype_lower($str[$i])), $str représentant me nouveau mot de passe et $i l'index de la chaine de caractères.
- Qu'il contient au moins 2 caractères autres que des lettres, donc s'ils ne correspondent pas aux deux vérifications précédentes.
- Si toutes les conditions sont remplies, alors la requête modifiant le mot de passe "UPDATE Utilisateur SET Mdp = ? WHERE Id = ?" est effectuée en passant en paramètres un tableau contenant le nouveau mot de passe et l'identifiant de l'utilisateur.
-

logout.php

Programme qui déconnecte un utilisateur en détruisant la session quoi qu'il arrive avec la méthode session_destroy(); puis qui redirige vers la page de connexion.

Paramètres

Dans un dossier "param" sont répertoriées toutes les classes permettant la gestion qui comprend un fichier __inc.php semblable au précédent sauf que la racine d'éxécution des fichiers est différente et que le programme vérifie que l'utilisateur est bien un administrateur pour pouvoir accéder au paramétrage. Pour chacun de ces programmes, si une requête "post" est passée, alors l'utilisateur pourra effectuer une création, si un identifiant est passé dans l'url, alors ce sera une modification. Si aucun des deux n'est passée, alors le programme affichera la liste des objets de la page.

affectation.php

Programme permettant de lister, d'ajouter et modifier une affectation.

- Pour lister les éléments, le programme réupère la liste depuis la base de données en appelant la méthode $db->getAffectations().
- Pour ajouter une affectation, l'id de l'enseignant ainsi que les groupes sélectionnés par l'administrateur sont récupérés.
    - Pour les sélectionner, le programme récupère tous les utilisateurs et vérifie pour chacun d'entre eux qu'il est un enseignant pour les ajouter au menu de sélection d'enseignant.
    - La requête "INSERT INTO Affectation VALUES (?, 'id du groupe')" est exécutée en passant un tableau en paramètres contenant respectivement l'identifiant de l'utilisateur et l'identifiant du groupe.
- Pour modifier une affectation, les valeurs sont récupérées et sont sélectionnés depuis le menu déroulant.

evenement.php

Programme permettant de lister, d'ajouter et modifier un événement.

- Pour lister les éléments, le programme récupère la liste depuis la base de données en appelant la méthode $db->getEvenements().
- Pour ajouter un événement, la catégorie de l'événement, une description de l'événement, la séance à laquelle il faut ajouter l'événement, les pièces jointes, la durée et date de l'évènement sont récupérées
- Pour les sélectionner, le programme récupère tous les utilisateurs et vérifie pour chacun d'entre eux qu'il est un enseignant pour les ajouter au menu de sélection d'enseignant.

- La requête "INSERT INTO Événement (type, libelle, durée, échéance, séance) VALUES (?,?,?,?,?)" est exécutée en passant un tableau en paramètres contenant respectivement le type, le libelle, la date, la durée et la séance.
- Pour modifier une affectation, les valeurs sont récupérées puis pré-remplissent les champs. Si les champs non obligatoires ne sont pas remplis, ils reçoivent une valeur par défaut. Par exemple, la durée prendra 0:0 et la date, celle du jour. L'id de l'événement est passé en paramètre dans la barre de recherche.

groupe.php

Programme permettant de lister, d'ajouter et modifier un groupe.

- Pour lister les éléments, le programme récupère la liste depuis la base de données en appelant la méthode $db->getGroupes().
- Pour ajouter un groupe, un nouveau nom de groupe est récupéré depuis une zone de texte ainsi que le groupe père sélectionnés parmi ceux du menu déroulant qui les contient tous sans distinction.
- La requête "INSERT INTO Groupe VALUES (?, ?)" est exécutée en passant un tableau en paramètres contenant respectivement le nom du groupe et le groupe père.
- Pour modifier un groupe, les valeurs sont récupérées et sont sélectionnés depuis le menu déroulant. Si le nom du groupe n'est pas vide, il y a une modification en fonction de ces changements.

index.php

Programme permettant d'afficher la page d'accueil du paramétrage.

- Pour ajouter une affectation, l'id de l'enseignant ainsi que les groupes sélectionnés par l'administrateur sont récupérés.

modules.php

Programme permettant de lister, d'ajouter et modifier un module.

- Pour lister les éléments, le programme réupère la liste depuis la base de données en appelant la méthode $db->getModules().
- Pour ajouter un module, l'id du nouveau module saisi dans la zone de texte a une taille maximale de 8 caractères définie par le cahier des charges, le libelle du nouveau module saisi dans la zone de texte, une couleur et les différentes personnes à qui nous pourrons les lier sont récupérés.
    - La requête "INSERT INTO Module (code, libelle, couleur, droits) VALUES (?,?,?,?)" est exécutée en passant un tableau en paramètres contenant respectivement le code du module, son libelle, sa couleur et les droits.

- Pour modifier un groupe, les valeurs sont récupérées et sont présélectionables depuis la page de sélection.

options.php

Programme permettant de modifier les options générales.

- Le programme récupère les valeurs correspondantes et change en effectuant la requête “INSERT INTO Parametres values (?,?) on conflict (Param) do update set Valeur = Excluded.Valeur" en passant en paramètre un tableau comprenant la clé et sa valeur.

seances.php

Programme permettant de lister, d’ajouter et modifier une séance.

- Pour lister les éléments, le programme réupère la liste depuis la base de données en appelant la méthode $db->getSeances().
- Pour ajouter une séance, un module, un type et un groupe sélectionnés parmi ceux des menus déroulants respectifs et une date sont récupérés.
- Une requête "INSERT INTO Seance(module, date, type, groupe, utilisateur) VALUES (?,?,?,?,?);" est exécutée en passant un tableau comprenant respectivement le module, la date, le type, le groupe et l’identifiant de l’utilisateur.
- Pour modifier une séance, les valeurs sont récupérées et sont présélectionables depuis les champs de la page. L’identifiant est récupéré depuis l’url de la page.

type_evenement.php

Programme permettant de lister, d’ajouter et de modifier un type d’événement. Le nom d’un type d’événement et les différents rôles des personnes qui pourront utiliser ce type sont récupérés sur la page. L’administrateur doit forcément pouvoir l’utiliser. Une valeur de l’identifiant est cachée si le type est déjà créé, ce qui permettra de savoir s’il s’agit d’un ajout ou d’une modification.

- Pour lister les éléments, le programme récupère la liste depuis la base de données en appelant la méthode $db->getTypesEvenement().
- Lors d’un ajout, le programme récupère les valeurs saisies par l’administrateur et la valeur de l’identifiant caché. Puisque le type est vide, il exécute la requête "INSERT INTO Type_Evenement (libelle, rôles) VALUES (?,?);" en passant un tableau comprenant respectivement le libellé du type et les droits qui y auront accès.
- Lors d’une modification, le programme récupère les valeurs de la base de données et les rentres dans les champs de la page de modification. Puisque le type n’est pas vide, il exécute la requête "UPDATE Type_Evenement SET Libelle = ?, Roles = ? WHERE

Id = ?", en passant un tableau comprenant respectivement le libellé du type, les droits qui y auront accès et l'identifiant.

utilisateur.php

Programme permettant de lister, d'ajouter et de modifier un utilisateur. Un identifiant, un nom, un prénom et un mot de passe sont récupérés depuis les zones de texte. L'identifiant de l'utilisateur est caché, ce qui permet de savoir, comme pour les types d'événement s'il s'agit d'une création ou d'une modification. Le rôle de l'utilisateur est récupéré depuis un checkbox et le groupe dans une liste.

- Pour lister les éléments, le programme récupère la liste depuis la base de données en appelant la méthode $db->getUtilisateurs().
- Pour créer un utilisateur, le programme récupère les valeurs. Puisque le type est vide, alors il exécute la requête "INSERT INTO Utilisateur (Id, Mdp, Nom, Prénom, Rôles, Groupes) VALUES (?,?,?,?,?,?)", en passant un tableau comprenant respectivement l'identifiant, le mot de passe, le nom, le prénom, le rôle et les groupes.
- Pour une modification, le programme récupère les nouvelles valeurs. Puisque l'identifiant caché n'est pas vide, le programme exécute la requête "UPDATE Utilisateur SET Id = ?, Nom = ?, Prénom = ?, Mdp = ?, Rôles = ?, Groupes = ? WHERE Id = ?", en passant un tableau en paramètres un tableau comprenant respectivement l'identifiant, le nom, le prénom, le mot de passe, le rôle, les groupes et l'identifiant caché puis modifie l'utilisateur.

Saisie

Dans un dossier “api” sont répertoriées toutes les classes permettant la suppression des éléments qui comprend un fichier __inc.php semblable au fichier __inc du dossier param.

deleteAffectation.php

Programme permettant la suppression de la liaison entre la personne et le module.

- Le programme récupère l’identifiant de l’utilisateur et l’identifiant du module de l’affectation. Puis la requête "DELETE FROM Affectation WHERE Utilisateur = ? AND Module = ?" est exécutée en passant en paramètres un tableau contenant respectivement l’identifiant de l’utilisateur et celui du module de l’affectation.

deleteEvenement.php

Programme permettant la suppression d’un événement.

- Le programme récupère l’identifiant de l’événement. Puis les requêtes "DELETE FROM Piece_Jointe WHERE Evenement = ?" et "DELETE FROM Evenement WHERE Id = ?" sont exécutées en passant en paramètres un tableau contenant respectivement l’identifiant de l’événement.

deleteGroupe.php

Programme qui supprime un groupe.

- Le programme récupère l’id du groupe et supprime toutes les données de ce groupe. La requete "DELETE FROM Groupe WHERE Nom = ? " est exécutée en passant en paramètres un tableau contenant l’identifiant du groupe.

deleteModule.php

Programme permettant la suppression d’un module.

- Le programme récupère l'identifiant du module. Puis la requête "DELETE FROM Module WHERE Code = ?" est exécutée en passant en paramètres un tableau contenant respectivement l'identifiant du module.

deleteSeance.php

Programme permettant la suppression d'une séance.

- Le programme récupère l'identifiant de la séance. Puis la requête "DELETE FROM Séance WHERE Id = ?" est exécutée en passant en paramètres un tableau contenant respectivement l'identifiant de la séance.

deleteTypeSeance.php

Programme permettant la suppression d'un type d'événement.

- Le programme récupère l'identifiant du type d'événement. Puis la requête "UPDATE Type_Evenement SET Actif = false WHERE Id = ?" est exécutée en passant en paramètres un tableau contenant respectivement l'identifiant du type d'événement.

deleteTypeSeance.php

Programme permettant la suppression d'un type de séance.

- Le programme récupère l'identifiant du type de séance. Puis la requête "UPDATE Type_Seance SET Actif = false WHERE Id = ?" est exécutée en passant en paramètres un tableau contenant respectivement l'identifiant du type de séance.

deleteUtilisateur.php

Programme permettant la suppression d'un utilisateur.

- Le programme récupère l'identifiant du type de séance. Puis la requête "DELETE FROM Utilisateur WHERE Id = ?" est exécutée en passant en paramètres un tableau contenant respectivement l'identifiant de l'utilisateur.

Visualisation

Dans un dossier “php” sont répertoriés les différents objets de la base de données.

affectation.php

Classe regroupant les méthodes utilisées par le programme pour agir sur l’objet.

- Possède comme attributs provenant de la base de données l’identifiant de l’utilisateur et du module.
- Possède une fonction qui construit un objet Affectation.
- Possède une fonction qui retourne l’identifiant de l’utilisateur.
- Possède une fonction qui retourne l’identifiant du module.

Evenement.php

Classe regroupant les méthodes utilisées par le programme pour agir sur l’objet.

- Possède comme attributs provenant de la base de données l’identifiant de l’événement, le type de l’événement, le libellé, la durée, l’échéance.
- Possède comme attributs extérieurs la ou les pièces jointes et le nom du type d’événement.
- Possède une fonction qui construit un objet Evenement.
- Possède une fonction qui retourne l’identifiant de l’événement.
- Possède une fonction qui retourne le type de l’événement.
- Possède une fonction qui retourne le libellé de l’événement.
- Possède une fonction qui retourne la durée de l’événement.
- Possède une fonction qui retourne l’échéance de l’événement.
- Possède une fonction qui retourne l’identifiant de la séance.
- Possède une fonction qui retourne la durée formatée pour s’afficher en heures et en minutes.

Groupe.php

Classe regroupant les méthodes utilisées par le programme pour agir sur l’objet.

- Possède comme attributs provenant de la base de données le nom du groupe et son groupe père.
- Possède une fonction qui construit un objet Groupe.

- Possède une fonction qui retourne le nom du groupe.
- Possède une fonction qui retourne le groupe père.

Module.php

Classe regroupant les méthodes utilisées par le programme pour agir sur l'objet.

- Possède comme attributs provenant de la base de données le code du module, le libellé, la couleur, les droits, la date de création et la date de modification.
- Possède une fonction qui construit un objet Module.
- Possède une fonction qui retourne le code du module.
- Possède une fonction qui retourne le libellé du module.
- Possède une fonction qui retourne la couleur du module.
- Possède une fonction qui retourne les droits qui auront accès au module.
- Possède une fonction qui retourne la date de création du module.
- Possède une fonction qui retourne la date de modification du module.

PieceJointe.php

Classe regroupant les méthodes utilisées par le programme pour agir sur l'objet.

- Possède comme attributs provenant de la base de données l'identifiant de la pièce jointe, le nom du fichier, le chemin depuis la racine du site et l'identifiant de l'événement.
- Possède une fonction qui construit un objet PieceJointe.
- Possède une fonction qui retourne l'identifiant de la pièce jointe.
- Possède une fonction qui retourne le nom du fichier
- Possède une fonction qui retourne le chemin depuis la racine du site.
- Possède une fonction qui retourne l'identifiant de l'événement.

Seance.php

Classe regroupant les méthodes utilisées par le programme pour agir sur l'objet.

- Possède comme attributs provenant de la base de données l'identifiant de la séance, le module, la date de création, la date de modification, le type de séance, le groupe et l'identifiant du créateur de la séance.
- Possède comme attributs extérieurs un objet "Module" qui possède comme identifiant celui de l'attribut de la séance, un tableau comprenant tous les événements de la séance, un tableau comprenant tous les événements qui serviront pour le filtrage, le nom du type d'événement et un objet "Utilisateur" qui possède comme identifiant celui de l'attribut de la séance.
- Possède une fonction qui construit un objet Séance.
- Possède une fonction qui retourne l'identifiant de la séance.

- Possède une fonction qui retourne l'identifiant du module.
- Possède une fonction qui retourne la date de création.
- Possède une fonction qui retourne la date de modification.
- Possède une fonction qui retourne le type de la séance.
- Possède une fonction qui retourne le groupe associé à la séance.
- Possède une fonction qui retourne l'identifiant du créateur de la séance.
- Possède une fonction qui retourne l'objet "Module".
- Possède une fonction qui retourne la date formatée.

Semaphore.php

Classe regroupant les méthodes utilisées par le programme pour agir sur l'objet.

- Possède comme attributs provenant de la base de données l'identifiant de l'utilisateur, l'identifiant de la séance et l'indice de marquage du sémaphore.
- Possède une fonction qui construit un objet Sémaphore.
- Possède une fonction qui retourne l'identifiant de l'utilisateur.
- Possède une fonction qui retourne l'identifiant de la séance.
- Possède une fonction qui retourne le marquage du sémaphore.

TypeEvenement.php

Classe regroupant les méthodes utilisées par le programme pour agir sur l'objet.

- Possède comme attributs provenant de la base de données l'identifiant du type d'événement, le libelle du type d'événement, les rôles affectés et indicateur de possibilité d'utilisation par les utilisateurs.
- Possède une fonction qui construit un objet TypeEvenement.
- Possède une fonction qui retourne l'identifiant du type d'événement.
- Possède une fonction qui retourne les rôles associés.
- Possède une fonction qui retourne l'indicateur de possibilité d'utilisation par les utilisateur.

TypeSeance.php

Classe regroupant les méthodes utilisées par le programme pour agir sur l'objet.

- Possède comme attributs provenant de la base de données l'identifiant du type de séance, le libelle du type de séance, les rôles affectés et indicateur de possibilité d'utilisation par les utilisateurs.
- Possède une fonction qui construit un objet TypeSeance.
- Possède une fonction qui retourne l'identifiant du type de séance.
- Possède une fonction qui retourne les rôles associés.

- Possède une fonction qui retourne l'indicateur de possibilité d'utilisation par les utilisateur.

Utilisateur.php

Classe regroupant les méthodes utilisées par le programme pour agir sur l'objet.

- Possède comme attributs provenant de la base de données l'identifiant de l'utilisateur, son mot de passe, son nom, son prénom, ses rôles, ses groupes, la date de création, la date de modification, affirmation en cas de changement récent de mot de passe.
- Possède une fonction qui construit un objet Utilisateur.
- Possède une fonction qui retourne l'identifiant de l'utilisateur.
- Possède une fonction qui retourne le mot de passe de l'utilisateur.
- Possède une fonction qui retourne le nom de l'utilisateur.
- Possède une fonction qui retourne le prénom de l'utilisateur.
- Possède une fonction qui retourne les rôles de l'utilisateur sous forme de caractères.
- Possède une fonction qui retourne ls groupes de l'utilisateur.
- Possède une fonction qui retourne la date de création de l'utilisateur.
- Possède une fonction qui retourne la date de modification de l'utilisateur.
- Possède une fonction qui retourne "vrai" si le mot de passe vient d'être modifier.
- Possède une fonction qui retourne 'vrai si le mot de passe passé en paramètres correspond au mot de passe dans la basse de données.
- Possède une fonction qui retourne les rôles de l'utilisateur sous forme de mot complet.

DB.php

Programme regroupant plusieurs toutes les fonctions permettant d'interagir avec les objets.

- Pour chaque objet, la classe possède des méthodes permettant de récupérer l'ensemble des objets de la classe, de récupérer un objet précis en passant un indicateur en paramètres, d'ajouter un nouvel objet en passant les mêmes paramètres que ceux de la méthode de construction de l'objet, de modifier un objet en passant les paramètres à modifier et un ou plusieurs indicateurs qui permettent de reconnaître l'objet à changer et de supprimer un objet précis en passant un identifiant correspondant à l'identifiant de l'objet.
- La méthode "query()" sert à exécuter les requêtes SQL de type "select". Elle prend en paramètre la requête, les paramètres et le type d'objet que renverra la requête si elle doit renvoyer quelque chose. Les paramètres sont insérés dans l'ordre aux endroits où des "?" sont présents dans la requête.
- La méthode "update()" permet d'effectuer des modifications dans la base de données. Elle prend en paramètre la requête et les paramètres. L'insertion des paramètres dans la requête se fait de la même façon que pour la méthode "query()".

func.php

Programme regroupant plusieurs fonctions utiles au programme.

- Contient une méthode retournant les options du menu de navigation (partie basse).
- Contient une méthode retournant les sections du menu de navigation (partie haute).
- Contient une méthode retournant la période scolaire courante.
- Contient une méthode retournant la période comprise entre le 1er septembre de l'année passée en paramètre et la fin de cette période.

Templates

Dans un dossier "tpl" sont répertoriées toutes les templates permettant l'affichage de la page internet.

base.twig

Il s'agit de la template de base de toutes les pages.

- Elle contient la mise en forme globale des pages du site ainsi que le menu de navigation. Ce menu est caché de base et peut être affiché grâce à un bouton activant une fonction javaScript.

  ```
  <div class="shade" onclick="toggleMenu(false)"></div>

  function toggleMenu(force = null) {...}
  ```

- Le menu déroulant contient des zones cliquables renvoyant vers les différentes sections du site. Ces sections sont renseignées par le script php qui appelle la template.

  ```
  {% for f in sections %}

      <a href="{{ f.url }}">{{ f.nom }}</a>

  {% endfor %}
  ```

- Toutes les autres templates sauf base-no-menu.twig et login.twig héritent de base.twig grâce à la ligne suivante
  `{% extends "base.twig" %}`

base-no-menu.twig

Il s'agit de la même template que base.twig sans le menu. Elle est utilisée par la template login.twig, elle utilisée pour générer la page de connexion.

accueil.twig

Cette template permet d'afficher l'accueil. Elle contient 2 boutons renvoyant vers saisie.php et etat.php.

```
<a class="button" href="saisie.php">Saisie</a>

<a class="button" href="etat.php">Visualisation</a>
```

profil.twig

Cette page permet à l'utilisateur de consulter et modifier ses informations personnelles.

- Le nom et le prénom de l'utilisateur sont chargés et envoyés par profil.php. Ils sont ensuite affichés de base dans les textfields correspondants.

  ```
  <input type="text" id="f_nom" name="nom" value="{{ user.nom }}">
  <input type="text" id="f_prenom" name="prenom" value="{{ user.prenom }}">
  ```
- Le bouton "Enregistrer" sous ces deux textfields envoie les informations à profil.php par la méthode "POST" grâce à la balise form qui les englobe.
  ```
  <form action="" method="POST">
      ... /* textfields */
      <input type="submit" value="Enregistrer" class="button">
  </form>
  ```
- Un champ permettant de changer les mots de passe est également disponible. Des textfields sont contenus dans une balise form afin de pouvoir changer de mot de passe. Lorsque le bouton "Modifier le mot de passe" est utilisé, les informations contenues dans ces textfields sont envoyées à profil.php où elles seront traitées.
- Un message d'erreur peut également être affiché si les informations entrées sont invalides

  ```
  <span class="erreur">{{ erreurMdp }}</span>
  ```