Ajouts et modifications par rapport à l'exemple développé en cours pour ce TP3

1- Contrôleurs

Une classe contrôleur pour chaque entité à gérer (AdminFilm, AdminGenre,… , AdminUtilisateur) qui hérite d'une classe Admin.

Dans cette classe Admin:

- La méthode 'gererEntite', instancie la classe de l'entité associée au paramètre 'entite' de l'uri ou force le login si l'utilisateur n'est pas connecté.
- La méthode ''gererAction', exécute la méthode associée au paramètre 'action' de l'uri, après avoir contrôlé si l'utilisateur connecté est autorisé.
- L'objet 'oUtilConn' (utilisateur connecté dans la variable $_SESSION), l'entité et l'action de l'uri sont stockés dans des variables statiques de la classe Admin pour pouvoir être utilisées par les objets des classes enfants.

2- Modèles

Les classes entités créées, Film et Utilisateur, héritent d'un classe Entite abstraite qui contient les méthodes génériques __construct, __get et __set.

La méthode 'getFilms' de la classe RequetesSQL est complétée pour adapter la SELECT aux informations nécessaires à la page de liste de l'interface d'administration, prise en compte du paramètre $critere = 'admin' pour cela.

3- Vues

3.1 Remplacement du constructeur par la méthode 'generer' pour permettre de retourner une page html à envoyer par courriel, ou bien d'envoyer directement la page html au navigateur par la commande echo.

3-2 Le dossier 'templates' est réorganisé pour répartir les composants dans 5 sous-dossiers:

- 'gabarits': gabarits selon les types de pages (erreur, frontend, admin et courriel),
- 'erreurs': pages d'erreurs (http 404 et 500),
- 'frontend': pages du site web public,
- 'admin': pages de l'interface d'administration,
- 'courriels: génération des messages courriels au format html.

La méthode 'generer' de la classe Vue est modifiée pour prendre en compte les sous-dossiers ci-dessus.

3-3 Codage de 'vAdminFilms.twig' et de la méthode 'listerFilms' dans la classe AdminFilm, pour afficher la page de liste par défaut de l'interface d'administration.
Codage également de toutes les vues pour gérer toutes les entités, hormis l'entité "seance" dont seule la vue de liste est codée.

4- JavaScript

Dans le gabarit 'admin', chargement du script '**admin.js**' qui permet le retrait/réaffichage du menu latéral et laisse ainsi plus de largeur à l'affichage des templates spécifiques.

Dans les templates de listes (vAdminFilms.twig, vAdminUtilisateurs.twig, etc.) chargement du script générique '**filtrer.js**' qui permet de:

- trier les lignes sur le critère le la colonne choisie en cliquant sur la flèche vers le haut pour un tri ascendant ou sur la flèche vers le bas pour un tri descendant.
- filtrer les lignes qui contiennent la chaîne saisie dans le champ input en haut de la page.

Dans la fenêtre modale modaleJonction.twig, chargement du script générique '**modaleJonction.js**' qui permet de gérer par drag&drop (glisser/déposer) les tables de jonctions film_acteur, film_pays et film_realisateur pour un film donné, à partir de la page de liste des films.

5- Gestion des utilisateurs

Un utilisateur peut être géré dans l'interface admin par un administrateur, à savoir être créé ou modifié en lui attribuant un profil et en lui envoyant un mot de passe aléatoire par courriel qu'il doit changer à la première utilisation (champ utilisateur_renouveler_mdp à oui).

Un utilisateur de profil client peut s'inscrire directement sur le site frontend via une fenêtre modale accessible dans le menu commun à toute les pages, et saisir directement son mot de passe en respectant les contraintes de complexité imposées par le site.