ROUX Raphaël, RICHARD Mathis, SEIGNOBOS Louis

Cahier des charges:

Médiathèque de Châteauroux-Les-Alpes

"Réalisation d’une application Web pour la gestion des emprunts et retours suite à la modernisation de la médiathèque."

Contexte

La médiathèque de Châteauroux-Les-Alpes a été créée en Octobre de l’an 2002. Le village souhaite utiliser les subventions de l'État pour moderniser cette dernière ainsi que faciliter son utilisation pour les clients. Aussi, la médiathèque souhaiterait instaurer un système de notation des médias empruntés par les usagers.

Le contexte est donc celui de la modernisation et la numérisation des registres de visite, d’emprunt, de retour, de location du matériel et de partage d'expérience.

Problématiques

- Amélioration de l'expérience utilisateur : Comment concevoir un site convivial et intuitif qui facilite la recherche, la sélection, et l'emprunt de médias pour les usagers de la médiathèque ?
- Optimisation de la gestion des collections : Comment gérer de manière plus efficace et efficiente les stocks de médias en utilisant une base de données, en suivant les emprunts et en anticipant les besoins des usagers ?
- Mesure de l'impact et de la satisfaction : Comment mesurer l'impact de la modernisation sur la fréquentation de la médiathèque et la satisfaction des usagers, et comment utiliser ces données pour apporter des améliorations continues ?
- Accessibilité et inclusion : Comment s'assurer que le site web et la base de données sont accessibles à tous, y compris les personnes ayant des besoins spécifiques en matière d'accessibilité ?

Besoins

1. **Gestion des Comptes Utilisateurs :**
    - Le besoin de pouvoir créer un compte pour chaque nouvel usager de la médiathèque.
    - L'administration des comptes, y compris la possibilité de les modifier ou de les désactiver si nécessaire.
2. **Gestion des Médias :**
    - La capacité de connaître les médias les mieux notés, impliquant une fonction de classement ou de tri.
    - La possibilité d'ajouter de nouveaux médias à la base de données de la médiathèque.

Besoins fonctionnels pour le Profil Usager :

1. **Gestion des Emprunts :**
    - La possibilité de consulter les dates de retour des médias empruntés.
    - La possibilité de réserver un média s'il est déjà emprunté par quelqu'un d'autre, et de l'emprunter s'il est disponible.
    - La possibilité de consulter la liste des médias et leur statut d'emprunt.
2. **Donations de Médias :**
    - La possibilité d'effectuer des donations de médias personnels à la médiathèque.
3. **Consultation et Classements :**
    - La possibilité de consulter la liste des médias disponibles, y compris leur statut d'emprunt.
    - La possibilité de consulter le classement des médias les plus empruntés.
    -

Contraintes Techniques :

1. **Environnement :**
    - La solution doit être compatible avec l'environnement GNU/LINUX.
2. **Accessibilité :**
    - La solution doit être accessible depuis une interface Web.
3. **Sécurité :**
    - Les mots de passe des utilisateurs doivent être chiffrés en hash BCrypt.

Réponse au besoin

Solutions fonctionnelles

Des quelques problématiques décrites ci-dessus, on peut en déduire les besoins principaux du système qui vont se mettre en place. Ces derniers seront décrits sous 2 catégories : le profil usager et le profil responsable de médiathèque.

Responsable de médiathèque

- Pouvoir créer et administrer un compte à chaque nouvel usager de la médiathèque
- Pouvoir connaître les médias les mieux notés
- Pouvoir rajouter des médias dans la BDD

Usager

- Pouvoir réserver un média si ce dernier a déjà été emprunté par quelqu'un, emprunter sinon.
- Pouvoir effectuer des donations de médias personnels à la médiathèque
- Pouvoir consulter tous les emprunts de tous les utilisateurs avec les dates correspondantes.

Contraintes fonctionnelles

- Les utilisateurs doivent obligatoirement avoir une adresse mail;
- Un média ne doit avoir qu'un seul genre;
- Un média ne doit avoir qu'un seul type;
- La date de retour réelle ne peut pas être renseignée tant que le statut d'emprunt n'est pas défini comme "rendu";
- La date de réservation est la même que la date d'emprunt s'il n'y a pas de délai pour emprunter.
- Le statut d'emprunt peut être : réservé, en cours ou rendu;
- La moyenne des notes pour un média est calculée automatiquement via un Trigger lors de l'insertion sur la table Avis.
- Trigger pour Calculer la Moyenne des Avis (calculMoyenne) :
    - Un déclencheur (Trigger) appelé majMoyenne est créé après l'insertion ou la mise à jour d'un avis (Avis).
    - Le trigger appelle la fonction calculMoyenne() qui met à jour la moyenne des notes du média correspondant.

- Trigger pour Vérifier la Disponibilité d'un Média (verifDisponibiliteMedia) :
    - Un trigger verifDisponibiliteMedia est créé pour vérifier la disponibilité d'un média avant d'effectuer un emprunt.
    - La fonction associée retourne un booléen indiquant si l'exemplaire du média est disponible.

- Trigger pour Vérifier les Emprunts avant la Suppression d'un Exemplaire (checkEmpruntBeforeDelete) :
    - Un trigger exemplaire_before_delete est mis en place avant la suppression d'un exemplaire.
    - La fonction associée checkEmpruntBeforeDelete() s'assure qu'aucun emprunt en cours n'est associé à l'exemplaire.

- Procédures pour Gérer les Opérations d'Emprunt et de Retour (empruntExemplaire, retourExemplaire) :
    - Deux procédures, empruntExemplaire et retourExemplaire, sont créées pour gérer respectivement l'emprunt et le retour d'un exemplaire.
    - La procédure empruntExemplaire vérifie la disponibilité de l'exemplaire et crée un emprunt en cours ou une réservation.
    - La procédure retourExemplaire met à jour le statut de l'emprunt en cours, la date de retour réelle et gère les réservations en attente.
    -
- Procédure pour Annuler une Réservation (annulerReservation) :
    - Une procédure annulerReservation est créée pour annuler une réservation en attente.

- Procédures pour Ajouter des Types, Genres et Auteurs (ajouterTypes, ajouterGenres, ajouterAuteursAMedias) :
    - Trois procédures (ajouterTypes, ajouterGenres, ajouterAuteursAMedias) sont créées pour ajouter de nouveaux types, genres et associer des auteurs à des médias.
    -
- Vues pour l'Analyse des Données (MediasSouventEmpruntes, ClassementMediasNoteMoyenne, UtilisateursAdministrateurs, UtilisateursNormaux) :
    - Quatre vues (MediasSouventEmpruntes, ClassementMediasNoteMoyenne, UtilisateursAdministrateurs, UtilisateursNormaux) sont créées pour faciliter l'analyse des données.

    - Ces vues fournissent des informations sur les médias souvent empruntés, le classement des médias par note moyenne, et la séparation des utilisateurs administrateurs et normaux.
    -
- Fonction pour vérifier si un exemplaire peut être supprimé (checkEmpruntBeforeDelete) :
    - Une fonction checkEmpruntBeforeDelete est créée pour être utilisée dans le trigger exemplaire_before_delete. Elle vérifie si un exemplaire peut être supprimé en fonction de son statut d'emprunt.

Contraintes techniques

- La solution devra pouvoir fonctionner dans un environnement GNU/LINUX.
- La solution devra être accessible depuis une interface Web.
- Les mots de passe des utilisateurs sont chiffrés en hash BCrypt.

Livrables

2 livrables sont attendus :

- La solution *packagée* dans des conteneurs type Docker
- La documentation contenant les précédures d'installation et de déploiement, le guide utilisateur et le guide administrateur.

Solutions techniques

D'un point de vue technique, nous proposons d'intégrer des technologies éprouvées, performantes et peu onéreuses :

- Une base de données PostgreSQL pour la gestion et la manipulation des données. PostgreSQL est une base de données libre, sans coût de licence, et elle existe depuis plus de 20 ans
- "pgAdmin, interface de gestion de la BD Postgres permettant de requêter en SQL au besoin
- Middleware Java, framework Spring, pour le dialogue avec la base de données. Servi en API REST permettant de communiquer avec le front ou de rendre accessible à l'utilisateur la BD sans requêter via SQL
- Stack HTML/CSS/JS pour la génération des pages web
- Docker pour le packaging de l'application dans un environnement maîtrisée et cloisonné

Comment interagir avec la base de données

- En connexion directe à postgres: localhost:5432 depuis l'extérieur du cluster ou postgres:5432 dans pgadmin (servi sur localhost:8090, vous trouverez un fichier servers.json à importer dans le dossier docker/pgadmin/ à la racine du projet)
- Via l'API servie par Spring: Spring sert l'API qui dialogue avec la base de données. Vous pouvez faire les appels directement avec Postman (exemple: localhost:9191/medias) ou via l'interface Swagger (localhost:9191/swagger-ui/index.html)
- Via le front: localhost:80, dans n'importe quel navigateur

Amélioration pour la prochaine version

- Implémenter un système de connexion avec identifiant et mot de passe: des vestiges d'une tentative de le faire dans les temps subsistent (SecurityConfig dans le middleware…) mais faire dialoguer un front sans framework particulier et une mécanique de sessions via token JWT ou cookie par exemple, ne nous semblait pas réalisable à temps.
- Un routeur maison pour Apache, qui permettrait par exemple d'accéder à une ressource niveau front-end par le path (par exemple, /media/1 pour accéder au média 1) plutôt que par le query string (par exemple, /media.html?id=1 pour accéder à ce même média)
- Faire un envoi de mail lors de l'approche de la date de rendu du média.

Livrables

Le livrable est un environnement *Docker* contenant l'ensemble des services, base de données et IHMs. Le livrable pourra s'installer sur n'importe quelle machine GNU/Linux avec *Docker* et *docker-compose* installés. Le fichier README décrit la procédure d'installation de l'application.

Annexe

Modèle entité association de la base de donnée :

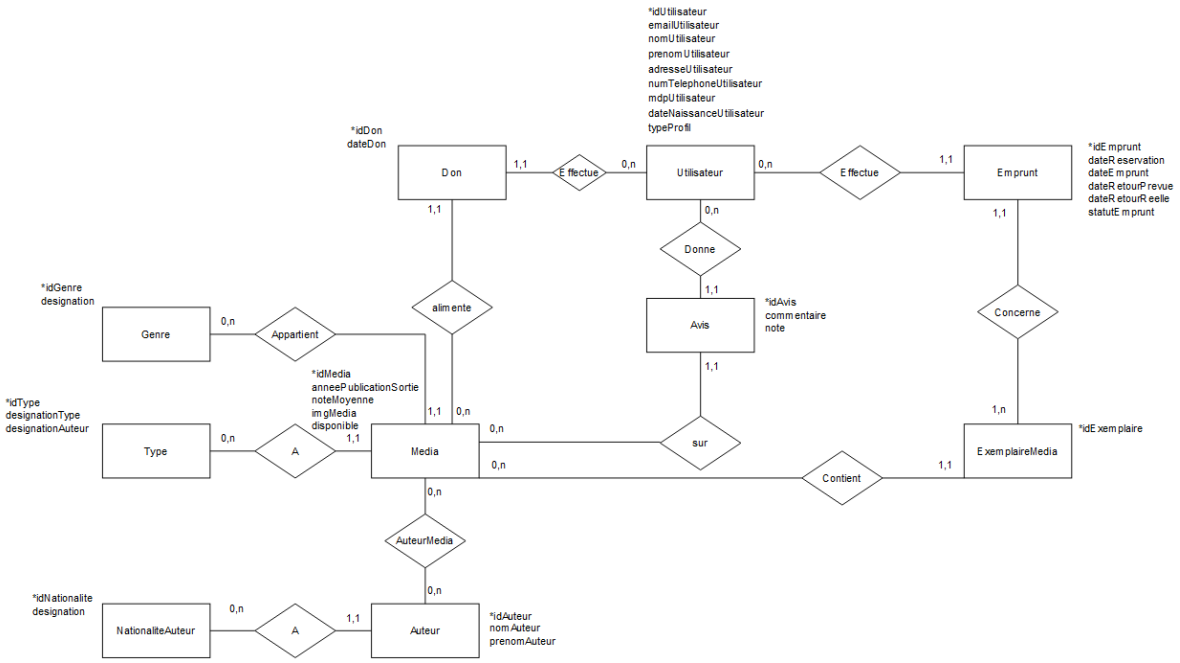

Schéma relationnelle de la base de donnée :

TypeMedia(**idType*, designationType, designationAuteur)

NationaliteAuteur(**idNationalite*, designation)

GenreMedia(**idGenre*, designation)

Auteur(**idAuteur*, nomAuteur, prenomAuteur, #nationaliteAuteur)

Media(**idMedia*, titre, #idType, #idGenre, anneePublicationSortie, noteMoyenne, imgMedia, disponible)

ExemplaireMedia(**idExemplaire*, #idMedia)

AuteurMedia(*#idAuteur,*#idMedia)

Utilisateur(**idUtilisateur*, emailUtilisateur, nomUtilisateur, prenomUtilisateur,

adresseUtilisateur, numTelephoneUtilisateur, mdpUtilisateur, dateNaissanceUtilisateur, typeProfil)

Don(*idDon*, #idUtilisateur, #idMedia, dateDon)

Avis(*idAvis*, #idUtilisateur, #idMedia, commentaire, note)

Emprunt(*idEmprunt*, #idUtilisateur, dateReservation, dateEmprunt, dateRetourPrevue, dateRetourReelle, statutEmprunt, #idExemplaire)

Et le diagramme généré par IntelliJ IDEA, montrant la base ainsi que les vues à droite.