Cahier des charges - Application de gestion de bibliothèque

Objectif du projet :

Créer une petite application web permettant de gérer une bibliothèque. L’application doit permettre de gérer des livres, leurs auteurs et les emprunts par les utilisateurs.

Entités et relations :

1. Entité `Book` (Livre)

Représente un livre dans la bibliothèque.

- **Attributs :**
  - `id` (int, clé primaire)
  - `title` (string)
  - `genre` (string)
  - `publishedAt` (date, facultatif)
  - `author` (relation vers `Author`)
- **Relations :**
  - Relation **Many-to-One** avec `Author` : Un livre est écrit par un auteur.
  - Relation **One-to-Many** avec `Borrow` : Un livre peut être emprunté plusieurs fois.

---

2. Entité `Author` (Auteur)

Représente un auteur de livres.

- **Attributs :**
  - `id` (int, clé primaire)
  - `firstName` (string)
  - `lastName` (string)
  - `bio` (text)
- **Relations :**
  - Relation **One-to-Many** avec `Book` : Un auteur peut écrire plusieurs livres.

---

3. Entité `User` (Utilisateur)

Représente les utilisateurs qui empruntent des livres.

- **Attributs :**
  - `id` (int, clé primaire)
  - `name` (string)
  - `email` (string, unique)
  - `roles` (array)
  - `password` (string, pour l'authentification)
- **Relations :**
  - Relation **One-to-Many** avec `Borrow` : Un utilisateur peut emprunter plusieurs livres.

---

4. Entité `Borrow` (Emprunt)

Représente un emprunt de livre par un utilisateur.

- **Attributs :**
  - `id` (int, clé primaire)
  - `borrowDate` (date)
  - `status` (string, ex. "en cours", "retourné")
- **Relations :**
  - Relation **Many-to-One** avec `Book` : Un emprunt concerne un seul livre.
  - Relation **Many-to-One** avec `User` : Un emprunt est effectué par un seul utilisateur.

Résumé des relations :

1. `Author` ↔ `Book` :
   - Un auteur écrit plusieurs livres (**One-to-Many**).
   - Un livre est écrit par un seul auteur (**Many-to-One**).
2. `Book` ↔ `Borrow` :
   - Un livre peut être emprunté plusieurs fois (**One-to-Many**).
   - Un emprunt concerne un seul livre (**Many-to-One**).
3. `User` ↔ `Borrow` :
   - Un utilisateur peut effectuer plusieurs emprunts (**One-to-Many**).
   - Un emprunt est effectué par un seul utilisateur (**Many-to-One**).

Fonctionnalités principales :

1. **Gestion des livres :**
    - Ajouter, modifier, et supprimer un livre.
    - Visualiser la liste des livres avec les détails suivants :
        - Titre
        - Auteur
        - Genre
        - Date de publication
    - BONUS : Rechercher un livre par titre ou auteur.
2. **Gestion des auteurs :**
    - Ajouter, modifier, et supprimer un auteur.
    - Afficher la liste des auteurs avec leurs informations :
        - Nom
        - Prénom
        - Biographie.
3. **Gestion des emprunts :**
    - Permettre aux utilisateurs d’emprunter un livre disponible.
    - Visualiser la liste des emprunts en cours avec :
        - Le nom de l’utilisateur.
        - Le titre du livre emprunté.
        - La date de l’emprunt.
4. **Gestion des utilisateurs :**
    - Permettre l’ajout et la gestion d’utilisateurs.
    - Chaque utilisateur doit avoir :
        - Un nom
        - Un email
        - Un rôle (administrateur ou utilisateur standard).

Contraintes techniques :

1. **Front-end :**
    - Utiliser Bootstrap ou Tailwind pour le style afin de simplifier la création d'une interface utilisateur.
2. **Sécurité :**
    - Implémenter un système d’authentification pour différencier les utilisateurs administrateurs et standards.
    - Les actions de création, modification et suppression doivent être réservées aux administrateurs.

Étapes de réalisation :

1. **Installation et configuration de Symfony :**
    - Installation de Symfony via Composer.
    - Configuration de la base de données dans le fichier `.env`.
2. **Création des entités :**
    - Créer les entités Doctrine correspondant aux tables.
3. **Développement des fonctionnalités :**
    - Routes et contrôleurs pour chaque fonctionnalité.
    - Formulaires pour la création/modification des données.
4. **Ajout de la sécurité :**
    - Configuration de l’authentification (utilisation de `Security`).
    - Gestion des rôles et autorisations.
5. **Tests et validation :**
    - Vérification du bon fonctionnement de l’application.
    - Correction des bugs éventuels.

Livrables :

1. Code source du projet (dans un dépôt Git, si possible).
2. Un README utilisateur expliquant comment utiliser l’application.
