Compte Rendu : Développement d'une Application de Blog Symfony

1. Introduction

Ce projet consiste en la réalisation d'un blog complet permettant la gestion d'articles et de commentaires. L'objectif était de mettre en œuvre les fondamentaux du framework Symfony tout en respectant des contraintes de sécurité, de validation et d'ergonomie.

2. Architecture de Données (Modélisation Doctrine)

Le projet repose sur trois entités principales interconnectées :

- Article : L'entité centrale contenant le titre, le contenu, l'auteur et la date de création (`createdAt`).
- Comment : Lié à l'Article par une relation ManyToOne. Chaque commentaire possède un auteur, un contenu et un statut d'approbation (`isApproved`) pour la modération.
- User : Gère l'authentification avec les champs email, rôles, mot de passe, prénom et nom.

3. Fonctionnalités Implémentées

3.1 Gestion des Articles (CRUD)

J'ai implémenté un système complet de gestion dans `ArticleController.php` :

- Liste et Recherche : La page d'accueil affiche les articles triés par date décroissante. Un moteur de recherche permet de filtrer les articles par titre, contenu ou auteur via une requête personnalisée dans `ArticleRepository`.
- Création et Édition : Utilisation de formulaires sécurisés (`ArticleType`) avec protection CSRF intégrée.

3.2 Système de Commentaires et Modération

- Ajout : Les utilisateurs (connectés ou non) peuvent poster des commentaires sous chaque article.
- Modération : Un système de validation a été mis en place. Les commentaires ne sont visibles qu'après approbation par un administrateur (`isApproved = true`).

- Interface Admin : Un espace dédié (`/admin/comments`) permet aux administrateurs de valider ou supprimer les commentaires.

3.3 Sécurité et Authentification

- Authentification Personnalisée : Utilisation d'un `CustomAuthenticator` pour gérer la connexion via email et mot de passe.
- Contrôle d'Accès :
    - Les actions sensibles (créer/éditer un article) sont protégées par le rôle `ROLE_USER`.
    - La modération des commentaires est strictement réservée au rôle `ROLE_ADMIN` via l'attribut `#[IsGranted('ROLE_ADMIN')]`.

3.4 Validation des Données

Toutes les entrées sont validées côté serveur via des contraintes (`Assert`) dans les entités pour garantir l'intégrité de la base de données :

- Articles : Titre obligatoire (min 3 car.) et contenu consistant (min 10 car.).
- Commentaires : Contenu obligatoire et limité à 1000 caractères.
- Utilisateurs : Email unique et format valide.

4. Expérience Utilisateur (UX/UI)

L'interface a été soignée grâce à l'intégration de Bootstrap 5 :

- Design Responsive : Le blog est consultable sur mobile et tablette.
- Navigation Intuitive : Une barre de navigation dynamique affiche les options selon les droits de l'utilisateur (Connexion/Inscription ou Profil/Déconnexion/Admin).
- Messages Flash : Notification systématique de l'utilisateur après chaque action (succès de création, modification, erreur d'authentification).

5. Qualité Technique

- Fixtures : Création d'un jeu de données de test (administrateur `admin@blog.com` et utilisateurs types) pour faciliter le développement.
- Respect des conventions : Le code suit les standards de nommage PSR et l'architecture MVC propre à Symfony.
- Base de données : Utilisation de SQLite pour une portabilité maximale lors de l'évaluation.

6. Conclusion

Ce projet démontre une maîtrise des cycles CRUD, de la gestion des relations entre entités et de la sécurisation d'une application web. Le système de modération des commentaires et le moteur de recherche constituent les valeurs ajoutées techniques de cette réalisation.