Test Technique Arengi : Gestion des Voitures

Durée estimée : 2 à 5 heures
Version Symfony requise : 6.4 ou supérieure

Objectif :

Réaliser une application Symfony permettant de gérer une collection de voitures avec deux niveaux d'accès distincts :

1. **Un rôle "Admin"** : Pour gérer les voitures (CRUD).
2. **Un rôle "Utilisateur anonyme"** : Pour consulter et filtrer les voitures sans authentification.

Spécifications

1. Environnement technique

- Symfony 6.4 minimum.
- Base de données relationnelle (MySQL, PostgreSQL, etc.).
- Utilisation des outils modernes de développement Symfony (autowiring, services, FormBuilder, validation, etc.).
- Respect des standards PSR et des bonnes pratiques Symfony.

2. Fonctionnalités demandées

Authentification et gestion des rôles

- Implémenter un système d'authentification sécurisé avec FOSUserBundle
- Deux rôles sont requis :
    - **ROLE_ADMIN** : accès complet aux fonctionnalités de gestion.
    - Aucun rôle (anonyme) : accès en lecture uniquement.

Gestion des voitures (ROLE_ADMIN uniquement)

1. **Ajout d'une voiture** via un formulaire.
    - Champs requis :
        - **Marque** (ex : Citroën, Mercedes...).
        - **Type** (berline, citadine, utilitaire).
        - **Nombre de passagers** (entier).
        - **Couleur** (texte).

Si le type de véhicule sélectionné est utilitaire, faire apparaître un nouveau champ **PTRA** (Poids Total Roulant Autorisé) qui sera **requis** également.

2. **Modification des informations d'une voiture** via un formulaire.
3. **Suppression d'une voiture** après confirmation.
4. **Liste des voitures (dashboard admin)** avec :
    - Pagination.
    - Possibilité de trier par marque, modèle, ou nombre de passagers.

Consultation et filtrage des voitures (accessible à tous)

1. Afficher une liste publique des voitures (aucune connexion requise).
    - La liste doit inclure :
        - La marque.
        - Le modèle.
        - Le nombre de passagers.
        - La couleur.
    - Possibilité de trier les résultats
2. **Filtrage** des voitures :
    - Par marque (ex : afficher uniquement les voitures de marque Citroën).
    - Par nombre de passagers (ex : afficher uniquement les voitures ayant 5 places ou plus).

---

3. Contraintes techniques

- **Séparation des responsabilités** : respecter le principe MVC.
- **Validation et sécurité** :
    - Validation des données dans les formulaires Symfony.
    - Accès sécurisé via les rôles définis.
- **Frontend** :
    - Pas de contrainte spécifique sur le style (Le visuel n'est pas évalué)
    - Une présentation simple mais fonctionnelle est suffisante.

Livrables

- Le projet Symfony hébergé sur un dépôt Git (GitHub, GitLab ou autre). Inclure des instructions d'installation claires dans un fichier `README.md` :
    - Commandes pour installer les dépendances.
    - Configuration de l'environnement (fichier `.env`).
    - Commandes pour exécuter la migration de la base de données.

Évaluation

L'évaluation sera basée sur :

1. La qualité du code (lisibilité, respect des bonnes pratiques).
2. Le respect des spécifications fonctionnelles.
3. La clarté des instructions pour installer et exécuter le projet.

Bon courage !