Objectifs :

Cette méthode de contrôleur permet à un administrateur de créer ou modifier une entité exhibition.

En s'assurant du traitement complet des données, textes, images, relations avec d'autres entités dans le respect de l'architecture MVC de Symfony. Je respecte aussi dans cette fonctionnalité l'encapsulation ainsi que la séparation des responsabilités. Nous allons voir en détail la description technique détaillée tout d'abord la méthode **addEdit**.

Exhibit BO est situé dans un contrôleur qui appartient à la couche de présentation du modèle MVC et agit comme orchestrateur entre la couche modèle qui est représentée ici par l'entité Exhibition. La couche vue à travers le rendu du formulaire via le un template twig et la logique de traitement, notamment le plat de fichiers ou le slugging.

Le Controller ne contient pas de logique métier complexe, celle-ci étant délégué à des services dédiés, images services, (respectant ainsi le principe de responsabilité unique du solide SRP.> facultatif)

À votre avis la ligne 37 A 44 ?

La méthode utilise l'injection automatique de dépendance pour recevoir des services nécessaires au fonctionnement. Tout d'abord, l'Entity Manager Interface. Pour la persistance, persistance des données, le file system manipuler le Système de fichiers local le Security pour accéder à l'utilisateur authentifié le service d'images pour le traitement de fichiers.

L'objet Request pour accéder aux données HTTP, un apprenti pour hydrater automatiquement l'objet exhibition à partir du slug passé en paramètres de l’URL via le param Converter de Symphony.

Ce découplage améliore la testabilité, la maintenabilité et le respect du principe d'inversion de dépendance. (Principe DIP que vous verrez en bac +3)

la ligne 46

L'entité exhibition issue de la couche modèle est soit instanciée en mode création soit récupérée via son slug en mode édition selon le principe ou non du paramètre. On va respecter le principe d'encapsulation car les modifications de l'entité passent exclusivement par les assesseurs.

Les accesseurs (getters) et mutateurs (setters) et permettent d’accéder directement à ces propriétés internes. (51 à 53)

Les assesseurs et mutateurs est une logique métier importante et dans ma fonctionnalité c'est la gestion de l'unicité du dossier d'image associé à la date de l'exposition. Cela repose sur la conversion de la date de l’exposition en d et du nom de l’exposition.

La vérification via le composant file system cette logique permet à terme être déportée dans un service métier dédié ou un event Suscriber doctrine pour respecter encore plus encore plus le découplage des responsabilités.

L 74

L'image est récupérée depuis le formulaire.

Son type mime est Controllé, elle est convertie automatiquement au format web P via un service injecté. Ça permet d'améliorer les performances web tout en maintenant un haut niveau de sécurité sur les fichiers utilisateurs.

l 14 : Y a un foreach en bas gestion des ticketPricing.

La fonctionnalité traite dynamiquement la collection de tarifs ticketPricing liée à l'exposition par une relation One to Many. Le traitement est différencié ça permet aussi de supprimer les tarifs obsolètes en comparant l'État d'origine de la collection avec l'État Post formulaire.

ligne 162 163

Toutes les entités, que ce soit exhibition ou ticketPricing, sont persistées via le Entity Manager interface qui gère l'unité de travail et la synchronisation des entités avec la base de données relationnelle doctrine, va assurer la gestion des relations, le cascading, l'optimisation des requêtes via les repository personnalisés si nécessaire.

J'ai pris en compte l'expérience utilisateur et le feedback UX. L'utilisateur va recevoir des messages Flash en cas d'erreur assurant un bon niveau de feedback dans la vue. En cas de succès, une redirection HTTP.

---  
La PO et l'encapsulation. Vous le validez aussi avec la manipulation par getter et setter. L'injection de service OK vitrine ORM, vous le validez ici parce que vous parlez de relation One to Many, vous parlez de persistance, vous parlez de gestion du cycle de vie.

OK, les services personnalisés avec l'externalisation du traitement d'image, la gestion de fichiers avec le la conversion web P le renommage intelligent on valide la sécurité avec un Control mime, une vérification des doublons, des messages Flash, on parle du X et d'accessibilité avec la gestion claire des erreurs.

Des redirections HTTP, des retours utilisateurs.

Et c'est conforme au RGPD parce que vous préparez l'anonymisation d'utilisateur.