TP Laravel : Système de Gestion de Stock

Durée estimée : 2h30 - 3h00
Sujet : Migrations, Modèles Eloquent et CRUD complet.
Enseignant : Y. Mkadem

Contexte

Une boutique locale souhaite moderniser sa gestion d'inventaire. Le gérant utilise actuellement un carnet papier et souhaite passer à une application web. Votre mission est de créer la partie administration permettant de gérer les produits du magasin.
L'application devra permettre de :

1. Lister tous les produits.
2. Ajouter un nouveau produit.
3. Modifier les informations d'un produit existant.
4. Supprimer un produit du stock.

Prérequis

— Environnement de développement PHP/MySQL opérationnel (Laragon, XAMPP, Docker...).
— Composer installé.
— Navigateur web.

---

1 Initialisation du projet

1. Générez un nouveau projet Laravel nommé `stock_app`.
2. Créez une nouvelle base de données MySQL vide nommée `db_stock`.
3. Configurez le fichier `.env` de votre projet pour connecter l'application à cette base de données.

2 Structure de la base de données (Migration)

Vous devez créer une table `products` pour stocker les informations.

1. Créez un Modèle nommé `Product` ainsi que son fichier de migration associé via la commande Artisan.
2. Dans le fichier de migration généré, définissez le schéma suivant pour la table :
   — `id` : Clé primaire (générée par défaut).
   — `name` : Chaîne de caractères (le nom du produit).
   — `description` : Texte long (description du produit), ce champ peut être laissé vide (*nullable*).

- `price` : Décimal (prix du produit), avec une précision de 8 chiffres dont 2 après la virgule.
- `stock` : Entier (quantité disponible).
- `timestamps` : Dates de création et modification (généré par défaut).

3. Lancez la migration pour créer la table en base de données.

3 Configuration du Modèle

Dans le modèle `Product.php`, configurez la propriété `$fillable` pour autoriser l'assignation de masse sur les champs `name`, `description`, `price` et `stock`.

4 Le Contrôleur et les Routes

1. Générez un contrôleur nommé `ProductController` de type **Resource** (il doit contenir les méthodes `index`, `create`, `store`, etc.).
2. Dans le fichier `routes/web.php`, déclarez une route de type ressource liée à ce contrôleur pour l'URL `/products`.
3. Vérifiez vos routes avec la commande :

```
php artisan route:list
```

5 Implémentation du CRUD

Vous devez maintenant implémenter la logique dans le contrôleur et créer les vues Blade correspondantes (créez un dossier `resources/views/products`).

A. Affichage (Read)

- **Méthode :** `index()`
- **Vue :** `index.blade.php`
- **Consigne :**
  - Récupérez la liste de tous les produits via le modèle.
  - Affichez-les dans un tableau HTML (Nom, Prix, Stock).
  - Ajoutez un lien "Créer un produit" au-dessus du tableau.
  - Pour chaque ligne du tableau, ajoutez deux liens/boutons : "Modifier" et "Supprimer".

B. Création (Create)

- **Méthodes :** `create()` (formulaire) et `store()` (enregistrement).
- **Vue :** `create.blade.php`
- **Consigne :**
  - Le formulaire doit pointer vers la route de stockage (méthode POST).
  - N'oubliez pas la protection CSRF.
  - Une fois enregistré, redirigez l'utilisateur vers la liste des produits avec un message de succès.

C. Modification (Update)

— **Méthodes :** `edit($id)` (formulaire pré-rempli) et `update($id)` (enregistrement).

— **Vue :** `edit.blade.php`

— **Consigne :**

  — Récupérez le produit ciblé grâce à son ID.

  — Le formulaire doit contenir les anciennes valeurs du produit dans les attributs `value`.

  — Utilisez la directive Blade adéquate pour simuler la méthode `PUT`.

  — Mettez à jour les données et redirigez vers l'index.

D. Suppression (Delete)

— **Méthode :** `destroy($id)`

— **Vue :** (Bouton dans `index.blade.php`)

— **Consigne :**

  — La suppression doit se faire via un formulaire (pour des raisons de sécurité) simulant la méthode `DELETE`.

  — Supprimez le produit et redirigez vers l'index.

6 Contraintes et Validation (Bonus obligatoire)

Dans les méthodes `store` et `update`, vous ne devez pas enregistrer de données invalides. Implémentez la validation suivante avant l'enregistrement :

— **name** : Requis, Maximum 255 caractères.

— **price** : Requis, Numérique.

— **stock** : Requis, Entier.

Si la validation échoue, Laravel doit rediriger vers le formulaire avec les erreurs affichées sous les champs concernés.

7 Aspect Visuel (Optionnel)

Intégrez un framework CSS (Bootstrap 5 ou Tailwind CSS) via CDN dans un fichier `layout.blade.php` et faites hériter vos vues de ce layout pour avoir une interface propre (Tableaux stylisés, boutons de couleurs, alertes pour les messages de succès).

Livrables attendus

1. Le dossier du projet (zippé, sans le dossier `vendor`).
2. Une capture d'écran de la liste des produits.
3. Une capture d'écran du formulaire de création avec une erreur de validation visible.