مكتب التكوين المهني وإنعاش الشغل

Office de la Formation Professionnelle
et de la Promotion du Travail

Direction Régionale Souss Massa

Examen de Fin de Module Régional N°205

« Développement Back-end »

Année 2022/2023

Filière : DDOWFS | **Epreuve : Théorique (variante 2)**

Niveau : TS | **Barème : /40**

Durée : 2h30

Voici les réponses aux différentes questions :

1. Pour créer les modèles "Produit" et "Catégorie", vous pouvez utiliser la commande artisan suivante :

php artisan make:model Produit

php artisan make:model Categorie

2. Pour permettre l'affectation de tous les attributs lors de la création ou de la mise à jour de plusieurs instances du modèle "Produit", vous devez ajouter la propriété `fillable` dans le modèle "Produit" de la manière suivante :

```
protected $fillable = ['titre', 'prix', 'description', 'image', 'categorie_id'];
```

3. Voici le code de la méthode "produits" à ajouter au modèle "Catégorie" :

```
public function produits()
{
    return $this->hasMany(Produit::class, 'categorie_id');
}
```

4. Pour créer le contrôleur "ProduitController", vous pouvez utiliser la commande artisan suivante :

php artisan make:controller ProduitController

Dans la classe "ProduitController", vous pouvez ajouter les méthodes suivantes :

```
use App\Models\Produit;
use App\Models\Categorie;
use Illuminate\Http\Request;
```

```
class ProduitController extends Controller
{
    public function index()
    {
        $produits = Produit::all();
        return view('produits.index', compact('produits'));
    }
    public function create()
    {
        $categories = Categorie::all();
        return view('produits.create', compact('categories'));
    }
    public function store(Request $request)
    {
        $validatedData = $request->validate([
            'titre' => 'required|max:255',
            'pages' => 'required| numeric,
            'description' => 'required',
            'categorie_id' => 'required|exists:categories,id',
        ]);
        Produit::create($validatedData);
        return redirect()->route('produits.index')->with('success', 'Produit ajouté avec
succès');
    }
    public function edit($id)
    {
        $produit = Produit::findOrFail($id);
        $categories = Categorie::all();
        return view('produits.edit', compact('produit', 'categories'));
    }
    public function update(Request $request, $id)
```

```
    {
        $validatedData = $request->validate([
            'titre' => 'required|max:255',
            'prix' => 'required|numeric',
            'description' => 'required',
            'image' => 'nullable|image|mimes:jpeg,png,jpg,gif|max:2048',
            'categorie_id' => 'required|exists:categories,id',
        ]);
        $produit = Produit::findOrFail($id);
        if ($request->hasFile('image')) {
            $imagePath = $request->file('image')->store('produits', 'public');
            $validatedData['image'] = $imagePath;
        }
        $produit->update($validatedData);
        return redirect()->route('produits.index')->with('success', 'Produit mis à jour avec
succès');
    }

    public function destroy($id)
    {
        $produit = Produit::findOrFail($id);
        $produit->delete();
        return redirect()->route('produits.index')->with('success', 'Produit supprimé avec
succès');
    }
}
```

12. Les routes correspondantes aux méthodes du contrôleur "ProduitController" peuvent être définies dans le fichier de routes web.php de votre application Laravel.

13. Voici le code de la vue "index" qui permet d'afficher la liste des produits dans un tableau :

```
@extends('layouts.app')
@section('content')
    <table>
```

```
<!-- Code pour afficher les produits dans le tableau à l'aide de la boucle foreach -->
</table>
@endsection
```

7. Pour la colonne "Actions" contenant les boutons "ajouter", "éditer" et "supprimer", vous pouvez utiliser du code HTML et inclure les liens ou les formulaires correspondants aux routes appropriées.

```
<td>
  <a href="{{ route('produits.create') }}" class="btn btn-success btn-sm">Ajouter</a>
  <a href="{{ route('produits.edit', $produit->id) }}" class="btn btn-primary btn-sm">Éditer</a>
  <form action="{{ route('produits.destroy', $produit->id) }}" method="POST" class="d-inline">
    @csrf
    @method('DELETE')
    <button type="submit" class="btn btn-danger btn-sm" onclick="return confirm('Êtes-vous sûr de vouloir supprimer ce produit ?')">Supprimer</button>
  </form>
</td>
```

8. Pour ajouter des règles de validation au formulaire de création de produit, vous pouvez utiliser les fonctionnalités de validation fournies par Laravel. Par exemple :

```
public function store(Request $request)
{
  $validatedData = $request->validate([
    'titre' => 'required|max:255',
    'prix' => 'required|numeric',
    'description' => 'required',
    'categorie_id' => 'required|exists:categories,id',
  ]);
```

9. Pour créer le middleware "AuthMiddleware", vous pouvez utiliser la commande artisan suivante :

```
php artisan make:middleware AuthMiddleware
```

10. Pour appliquer le middleware "AuthMiddleware" sur les routes du contrôleur "ProduitController", vous pouvez le spécifier dans le constructeur ducontrôleur "ProduitController", vous pouvez le spécifier dans le constructeur du contrôleur ou définir des groupes de routes dans le fichier de routes web.php de votre application Laravel.

```
Route::middleware('auth')->group(function () {
    Route::get('/produits', 'ProduitController@index')->name('produits.index');
    Route::get('/produits/create', 'ProduitController@create')->name('produits.create');
    // ... autres routes pour le contrôleur ProduitController
});
```

Pour ajouter la fonctionnalité de téléchargement d'une image pour chaque produit, voici les réponses aux sous-questions :

18- a. Pour ajouter une migration permettant d'ajouter une colonne "image" à la table des produits, vous pouvez utiliser la commande artisan suivante :

1 pt

```
php artisan make:migration add_image_to_produits --table=produits
```

b. Les fonctions "up" et "down" de la migration peuvent être définies comme suit :

```
public function up()
{
    Schema::table('produits', function (Blueprint $table) {
        $table->string('image')->nullable();
    });
}
public function down()
{
    Schema::table('produits', function (Blueprint $table) {
        $table->dropColumn('image');
    });
}
```

0,5

0,5

c. Pour gérer le téléchargement de l'image dans la méthode "store" du contrôleur "ProduitController", vous pouvez utiliser les fonctionnalités de gestion des fichiers de Laravel. Par exemple :

```
public function store(Request $request)
{
    $validatedData = $request->validate([
        // Les autres règles de validation
        'image' => 'nullable|image|mimes:jpeg,png,jpg,gif|max:2048',
    ]);
    if ($request->hasFile('image')) {
```

1

```
    $imagePath = $request->file('image')->store('produits', 'public');
    $validatedData['image'] = $imagePath;
}
```

d. Pour afficher l'image associée à chaque produit dans la colonne "image" du tableau de la vue "index", vous pouvez utiliser la balise <img> de HTML en spécifiant le chemin de l'image. Par exemple :

```
<td>
    <img src="{{ asset('storage/' . $produit->image) }}" alt="Image du produit" width="100%">
</td>
```

Barème :

Question	Points
1	2
2	2
3	2
4	2
5	2
6	2
7	2
8	2

Question	Points
9	2
10	2
11	2
12	2
13.a	2
13.b	2
14	2
15.a	1

Question	Points
15.b	1
15.c	1
15.d	1
16	1
17	1
18.a	1
18.b	1
18.c	1
18.d	1