

# Atelier 8 : Ajout d'un Nouveau Produit sur un Site E-commerce

## 1. Objectif de l'Atelier

L'objectif de l'Atelier 8 est de mettre en place une fonctionnalité **Add Product** dans une application Laravel de type e-commerce.

Cette fonctionnalité permet :

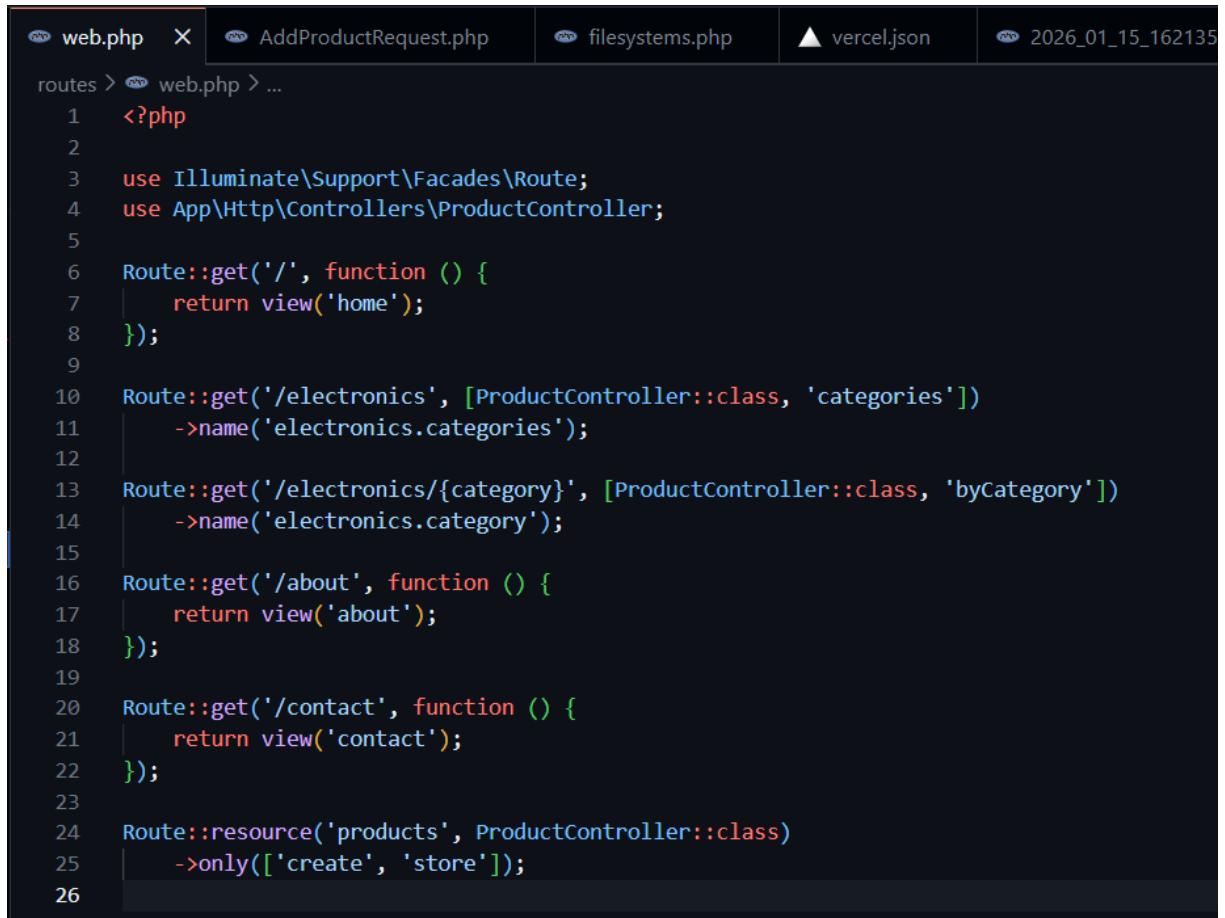
- d'afficher un formulaire d'ajout de produit,
- de valider les données via une **Form Request**,
- d'enregistrer le produit dans la base de données,
- d'uploader une image,
- d'afficher un **message flash** après succès.

## 2. Architecture Générale

L'atelier repose sur **5 éléments principaux** :

1. **Routes** (routes/web.php)
2. **Contrôleur Resource** (app/Http/Controllers/RproductController.php)
3. **Form Request** (app/Http/Requests/AddProductRequest.php)
4. **Vue Blade** (resources/views/addproduit.blade.php)
5. **Message Flash** (resources/views/incs/flash.blade.php)

## 3. Routes (web.php)



```
routes > web.php > ...
1  <?php
2
3  use Illuminate\Support\Facades\Route;
4  use App\Http\Controllers\ProductController;
5
6  Route::get('/', function () {
7      return view('home');
8  });
9
10 Route::get('/electronics', [ProductController::class, 'categories'])
11     ->name('electronics.categories');
12
13 Route::get('/electronics/{category}', [ProductController::class, 'byCategory'])
14     ->name('electronics.category');
15
16 Route::get('/about', function () {
17     return view('about');
18 });
19
20 Route::get('/contact', function () {
21     return view('contact');
22 });
23
24 Route::resource('products', ProductController::class)
25     ->only(['create', 'store']);
```

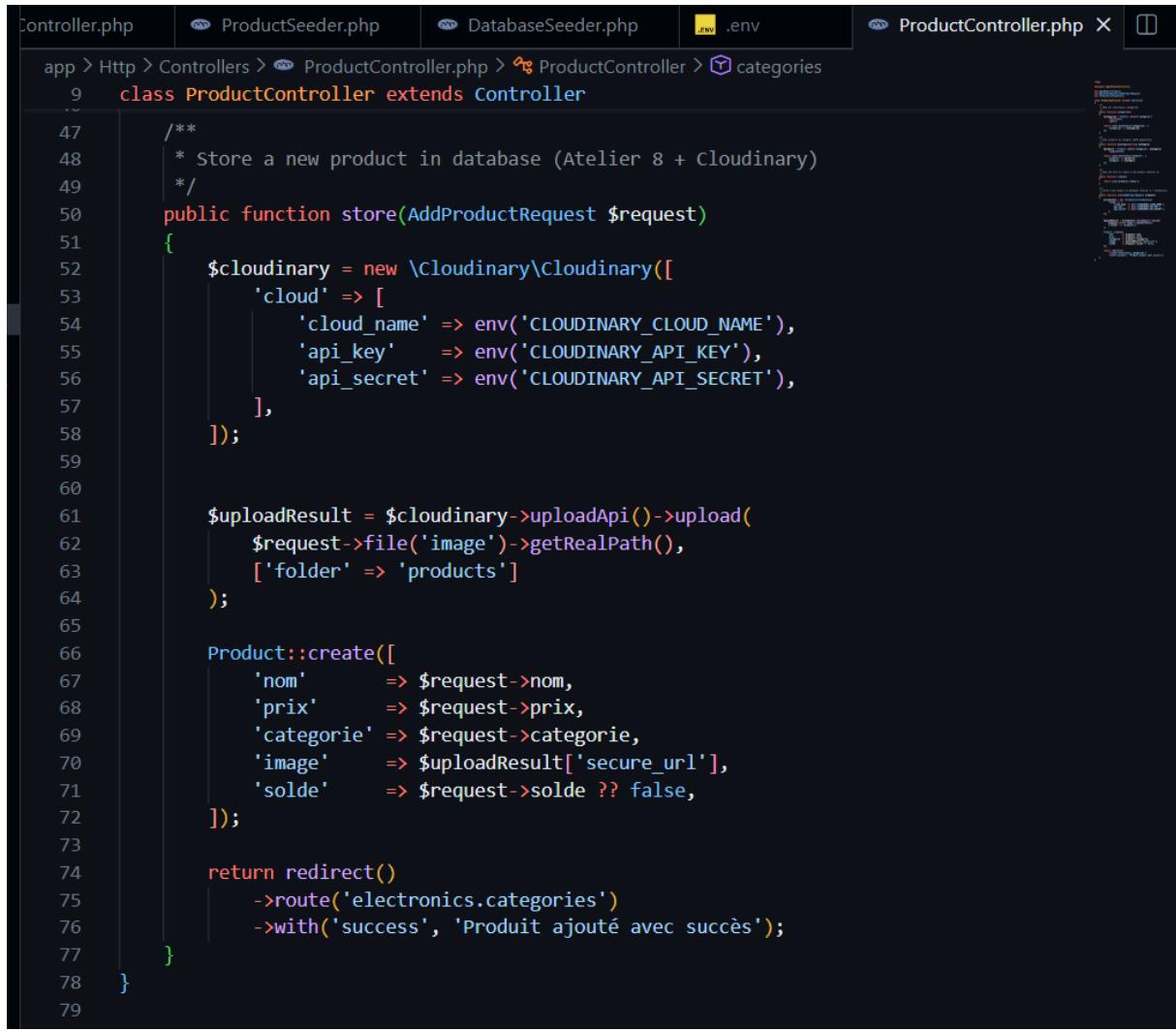
Fonctionnalités assurées :

- /produits/create → affiche le formulaire
- POST /produits → enregistre le produit

Laravel relie automatiquement :

- create() → affichage du formulaire
- store() → traitement du formulaire

## 4. Contrôleur Resource



The screenshot shows a code editor with several tabs at the top: Controller.php, ProductSeeder.php, DatabaseSeeder.php, .env, and ProductController.php. The ProductController.php tab is active, displaying the following PHP code:

```
Controller.php | ProductSeeder.php | DatabaseSeeder.php | .env | ProductController.php X
app > Http > Controllers > ProductController.php > ProductController > categories
9  class ProductController extends Controller
10
11  /**
12   * Store a new product in database (Atelier 8 + Cloudinary)
13   */
14  public function store(AddProductRequest $request)
15  {
16      $cloudinary = new \Cloudinary\Cloudinary([
17          'cloud' => [
18              'cloud_name' => env('CLOUDINARY_CLOUD_NAME'),
19              'api_key' => env('CLOUDINARY_API_KEY'),
20              'api_secret' => env('CLOUDINARY_API_SECRET'),
21          ],
22      ]);
23
24      $uploadResult = $cloudinary->uploadApi()->upload(
25          $request->file('image')->getRealPath(),
26          ['folder' => 'products']
27      );
28
29      Product::create([
30          'nom' => $request->nom,
31          'prix' => $request->prix,
32          'categorie' => $request->categorie,
33          'image' => $uploadResult['secure_url'],
34          'solde' => $request->solde ?? false,
35      ]);
36
37      return redirect()
38          ->route('electronics.categories')
39          ->with('success', 'Produit ajouté avec succès');
40  }
41
42 }
```

Méthodes utilisées :

create()

- Retourne la vue du formulaire d'ajout de produit.

store(AddProductRequest \$request)

- Valide les données automatiquement via la Form Request
- Récupère les champs :

- nom
- prix
- catégorie
- image

- Enregistre le produit dans la base de données
  - Déplace l'image vers public/imgs
  - Retourne un message flash de succès

## 5. Validation – Form Request

```
app > Http > Requests > AddProductRequest.php > AddProductRequest > rules
1  <?php
2
3  namespace App\Http\Requests;
4
5  use Illuminate\Foundation\Http\FormRequest;
6
7  class AddProductRequest extends FormRequest
8  {
9      public function authorize(): bool
10     {
11         return true;
12     }
13
14     public function rules(): array
15     {
16         return [
17             'nom'      => 'required|min:5',
18             'prix'     => 'required|numeric',
19             'categorie' => 'required|min:3',
20             'image'    => 'required|image|mimes:jpg,jpeg,png,webp|max:2048',
21             'solde'    => 'nullable|boolean',
22         ];
23     }
24
25     public function messages(): array
26     {
27         return [
28             'nom.required'      => 'Product name is required',
29             'nom.min'          => 'Name must be at least 5 characters',
30             'prix.required'    => 'Price is required',
31             'prix.numeric'     => 'Price must be a number',
32             'categorie.required' => 'Category is required',
33             'categorie.min'    => 'Category must be at least 3 characters',
34             'image.required'   => 'Image file is required',
35             'image.image'       => 'Image must be a valid image',
36             'image.mimes'       => 'Image must be jpg, jpeg, png, or webp',
37             'image.max'         => 'Image must not exceed 2MB',
38         ];
39     }
40 }
```

La validation est externalisée pour :

- un code plus propre,
  - une meilleure organisation,
  - une meilleure réutilisabilité.

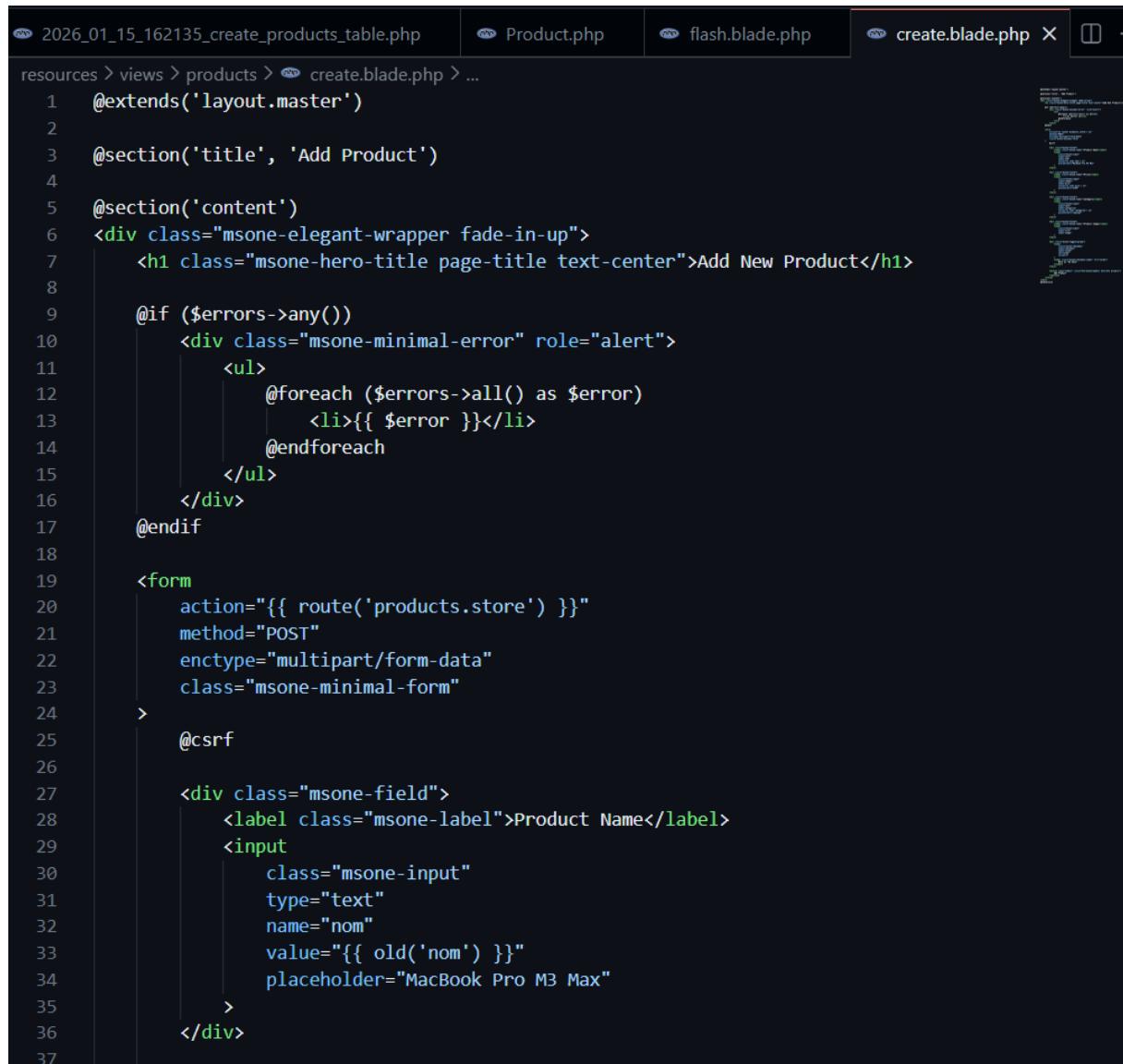
Règles appliquées :

- nom : obligatoire, minimum 5 caractères
- prix : obligatoire
- categorie : obligatoire, minimum 5 caractères

Messages personnalisés :

- messages clairs pour chaque champ en cas d'erreur

## 6. Vue – Formulaire d'ajout



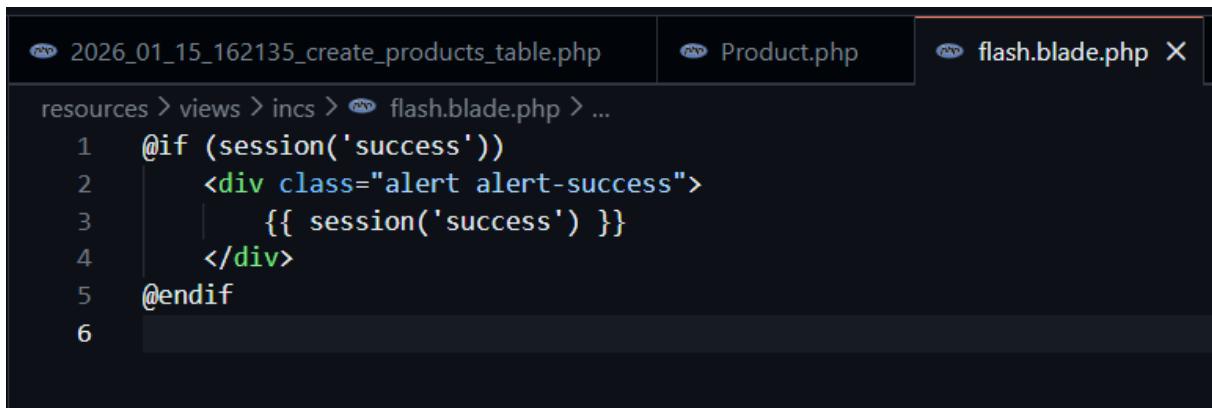
The screenshot shows a code editor with the 'create.blade.php' file open. The file contains Blade template code for a product creation form. It includes sections for errors, a title, and a form with a product name input field. The code is numbered from 1 to 37.

```
resources > views > products > create.blade.php > ...
1 @extends('layout.master')
2
3 @section('title', 'Add Product')
4
5 @section('content')
6 <div class="msone-elegant-wrapper fade-in-up">
7     <h1 class="msone-hero-title page-title text-center">Add New Product</h1>
8
9     @if ($errors->any())
10         <div class="msone-minimal-error" role="alert">
11             <ul>
12                 @foreach ($errors->all() as $error)
13                     <li>{{ $error }}</li>
14                 @endforeach
15             </ul>
16         </div>
17     @endif
18
19     <form
20         action="{{ route('products.store') }}"
21         method="POST"
22         enctype="multipart/form-data"
23         class="msone-minimal-form"
24     >
25         @csrf
26
27         <div class="msone-field">
28             <label class="msone-label">Product Name</label>
29             <input
30                 class="msone-input"
31                 type="text"
32                 name="nom"
33                 value="{{ old('nom') }}"
34                 placeholder="MacBook Pro M3 Max"
35             >
36         </div>
37     
```

Le formulaire contient :

- @csrf pour la protection CSRF
- enctype="multipart/form-data" pour l'upload d'image
- Champs :
  - nom
  - prix
  - catégorie
  - image
- Affichage des erreurs avec @error
- Inclusion du message flash

## 7. Message Flash



```
resources > views > incs > flash.blade.php > ...
1  @if (session('success'))
2      <div class="alert alert-success">
3          {{ session('success') }}
4      </div>
5  @endif
6
```

Rôle :

- afficher un message de succès après l'ajout du produit
- utiliser la session Laravel (Session::get('success'))

Ce fichier est inclus dans les vues avec :

```
@include('incs.flash')
```