

Atelier 10 : Création et consommation d'une API REST avec Laravel & React

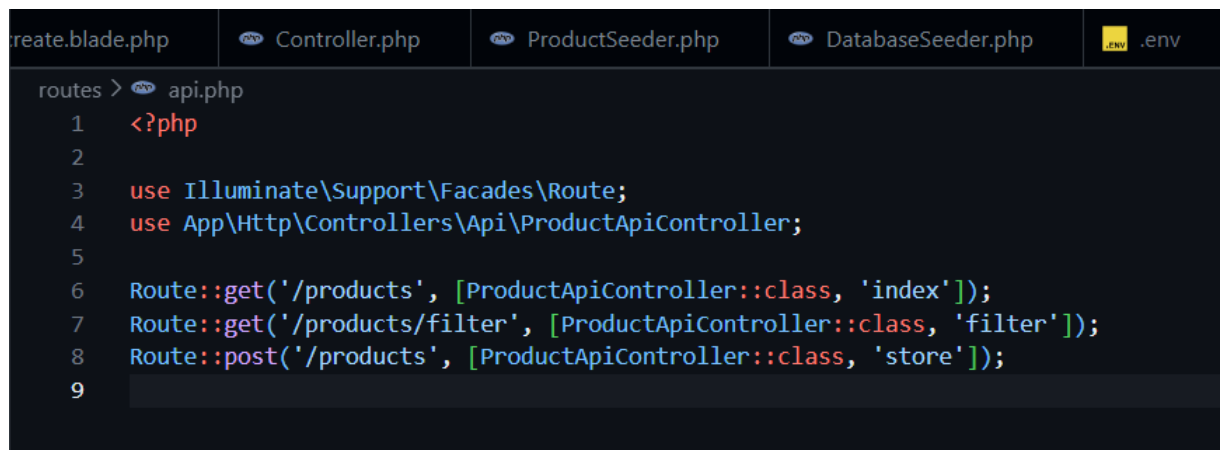
Objectif de l'Atelier

L'objectif de cet atelier est de :

- Créer une **API REST** avec Laravel
- Exposer les données des produits sous forme **JSON**
- Consommer cette API depuis une **application React (Vite)**
- Mettre en place un **filtrage dynamique** des produits
- Héberger l'API Laravel et l'application React sur **Vercel**

Backend Laravel

routes/api.php



```
routes > api.php
1  <?php
2
3  use Illuminate\Support\Facades\Route;
4  use App\Http\Controllers\Api\ProductApiController;
5
6  Route::get('/products', [ProductApiController::class, 'index']);
7  Route::get('/products/filter', [ProductApiController::class, 'filter']);
8  Route::post('/products', [ProductApiController::class, 'store']);
9
```

app/Http/Controllers/API/ProductApiController.php

```
create.blade.php Controller.php ProductSeeder.php DatabaseSeeder.php .env ProductApiController.php X
app > Http > Controllers > Api > ProductApiController.php > ProductApiController > index
1  <?php
2
3  namespace App\Http\Controllers\Api;
4
5  use App\Http\Controllers\Controller;
6  use App\Models\Product;
7  use Illuminate\Http\Request;
8  use Cloudinary\Cloudinary;
9
10 class ProductApiController extends Controller
11 {
12     public function index(Request $request)
13     {
14         $query = Product::query();
15
16         // Filter by category
17         if ($request->filled('categorie')) {
18             $query->where('categorie', $request->categorie);
19         }
20
21         // Filter by name (typing letters)
22         if ($request->filled('search')) {
23             $query->where('nom', 'LIKE', '% ' . $request->search . '%');
24         }
25
26         return response()->json(
27             $query->select('id', 'nom', 'prix', 'categorie', 'image')
28             ->paginate(100)
29         );
30     }
31 }
```

```
31
32     public function store(Request $request)
33     {
34         $request->validate([
35             'nom' => 'required|string|max:255',
36             'prix' => 'required|numeric',
37             'categorie' => 'required|string|max:255',
38             'image' => 'nullable|image|mimes:jpg,jpeg,png,webp|max:2048',
39         ]);
40
41         $imagePath = null;
42
43         try {
44             if ($request->hasFile('image')) {
45                 $cloudinary = new Cloudinary([
46                     'cloud' => [
47                         'cloud_name' => env('CLOUDINARY_CLOUD_NAME'),
48                         'api_key' => env('CLOUDINARY_API_KEY'),
49                         'api_secret' => env('CLOUDINARY_API_SECRET'),
50                     ],
51                 ]);
52
53                 $uploadResult = $cloudinary->uploadApi()->upload(
54                     $request->file('image')->getRealPath(),
55                     ['folder' => 'products']
56                 );
57
58                 $imagePath = $uploadResult['secure_url'];
59             }
60         } catch (\Exception $e) {
61             return response()->json([
62                 'error' => 'Failed to upload image: ' . $e->getMessage(),
63             ], 400);
64         }
65     }
```

```

65
66         try {
67             $product = Product::create([
68                 'nom' => $request->nom,
69                 'prix' => $request->prix,
70                 'categorie' => $request->categorie,
71                 'image' => $imagePath,
72             ]);
73
74             return response()->json([
75                 'message' => 'Product created successfully',
76                 'product' => $product,
77             ], 201);
78         } catch (\Exception $e) {
79             return response()->json([
80                 'error' => 'Failed to create product: ' . $e->getMessage(),
81             ], 400);
82         }
83     }
84 }

```

app/Models/Product.php

```

Product.php X flash.blade.php create.blade.php Controller.php ProductSeeder.php
app > Models > Product.php > ...
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Model;
6
7  class Product extends Model
8  {
9      protected $fillable = [
10         'nom',
11         'prix',
12         'categorie',
13         'image',
14         'solde',
15     ];
16 }
17

```

Frontend React

src/components/AddComp.jsx

```
.env AddComp.jsx X App.jsx MS1-4.png FilComp.jsx index.css m
ms1_api_reader > src > components > AddComp.jsx > AddComp
1 import { useState, useEffect } from "react";
2 import axios from "axios";
3
4 export default function AddComp() {
5   const [form, setForm] = useState({
6     nom: "",
7     prix: "",
8     categorie: "",
9     image: null,
10  });
11  const [error, setError] = useState("");
12  const [loading, setLoading] = useState(false);
13
14  useEffect(() => {
15    const token = document.querySelector('meta[name="csrf-token"]')?.content;
16    if (token) {
17      axios.defaults.headers.common["X-CSRF-TOKEN"] = token;
18    }
19  }, []);
20
21  const submit = async () => {
22    try {
23      setError("");
24      setLoading(true);
25
26      if (!form.nom.trim() || !form.prix || !form.categorie.trim()) {
27        throw new Error("Please fill in all required fields");
28      }
29
30      const data = new FormData();
31      data.append("nom", form.nom.trim());
32      data.append("prix", parseFloat(form.prix));
33      data.append("categorie", form.categorie.trim());
34
35      if (form.image) {
36        data.append("image", form.image);
37      }
38    } catch (error) {
39      setError(error.message);
40      setLoading(false);
41    }
42  };
43}
```

src/components/FilComp.jsx

```

ms1_api_reader > src > components > FilComp.jsx > Products
1  import { useEffect, useState } from "react";
2  import axios from "axios";
3
4  export default function Products() {
5    const [products, setProducts] = useState([]);
6    const [search, setSearch] = useState("");
7    const [category, setCategory] = useState("");
8    const [typingTimeout, setTypingTimeout] = useState(null);
9
10   const fetchProducts = (searchValue, categoryValue) => {
11     // Log values here to debug in your browser console
12     console.log("Filtering by:", { search: searchValue, category: categoryValue });
13
14     axios
15       .get(`${import.meta.env.VITE_API_URL}/products`, {
16         params: {
17           search: searchValue,
18           categorie: categoryValue, // Ensure your backend expects 'categorie'
19         },
20       })
21       .then((res) => {
22         // Accessing data based on standard Laravel/API pagination structures
23         setProducts(res.data.data || res.data);
24       })
25       .catch((err) => console.error("API Error:", err));
26   };
27
28   const handleSearch = (e) => {
29     const value = e.target.value;
30     setSearch(value);
31
32     if (typingTimeout) clearTimeout(typingTimeout);
33
34     setTypingTimeout(
35       setTimeout(() => {
36         fetchProducts(value, category);
37       }, 500)

```