Atelier 10:

Création et Consommation d'une API REST avec Laravel et React

1- Creation de api controller :

```
houdaifa@HoudaifaPC:~/everything/laravel/Ecomerce$ php artisan make:controller Api/ProduitControllerApi --api

  INFO  Controller [app/Http/Controllers/Api/ProduitControllerApi.php] created successfully.

houdaifa@HoudaifaPC:~/everything/laravel/Ecomerce$
```

2- Recuperer tous les produits et les renvois sous la format json:

```php
public function index()
{
    $produits=Produit::all();
    return response()->json($produits);
}
```

3- Validation, enregistrement de l'image sur cloudinary et des informations dans la base de donnees :

```php
public function store(Request $request)
{
    // Validation
    $validated = $request->validate([
        'nom' => 'required|string|max:255',
        'prix' => 'required|numeric|min:0',
        'categorie' => 'required|string|max:255',
        'image' => 'required|image|mimes:jpeg,png,jpg,gif|max:5120',
        'description' => 'nullable|string',
    ]);

    // image exist?
    if (!$request->hasFile('image')) {
        return response()->json(['error' => 'Image file is required.'], 400);
    }

    try {
        $cloudinary = new Cloudinary([
            'cloud' => [
                'cloud_name' => env('CLOUDINARY_CLOUD_NAME'),
                'api_key'    => env('CLOUDINARY_API_KEY'),
                'api_secret' => env('CLOUDINARY_API_SECRET'),
            ],
        ]);
        @var \Cloudinary\Api\ApiResponse $result
        $result = $cloudinary->uploadApi()->upload(
            $request->file('image')->getRealPath(),
            ['folder' => 'produits']
        );

        $validated['image'] = $result['secure_url'];

        $Produit = Produit::create($validated);
        return response()->json(['message' => 'produit ajouté avec succès!', 'product' => $Produit], 201);
    } catch (\Exception $e) {
        return response()->json(['error' => 'Error uploading image: ' . $e->getMessage()], 500);
    }
}
```

4- La methode filter pour filtrer les produits par nom et les renvois sous la format json :

```php
public function filtrer(Request $request)
{
    // Récupération de la chaîne de caractères de recherche
    $query = $request->input('p');

    // Recherche des articles correspondants
    $produits = Produit::where('nom', 'like', "%$query%")->get();

    // Renvoi des articles au format JSON
    return response()->json($produits);


}
```

5- Dans bootstrap/app.php (api.php) :

```php
return Application::configure(basePath: dirname(__DIR__))
    ->withRouting(
        web: __DIR__.'/../routes/web.php',
        api: __DIR__.'/../routes/api.php',
        commands: __DIR__.'/../routes/console.php',
        health: '
    )
```

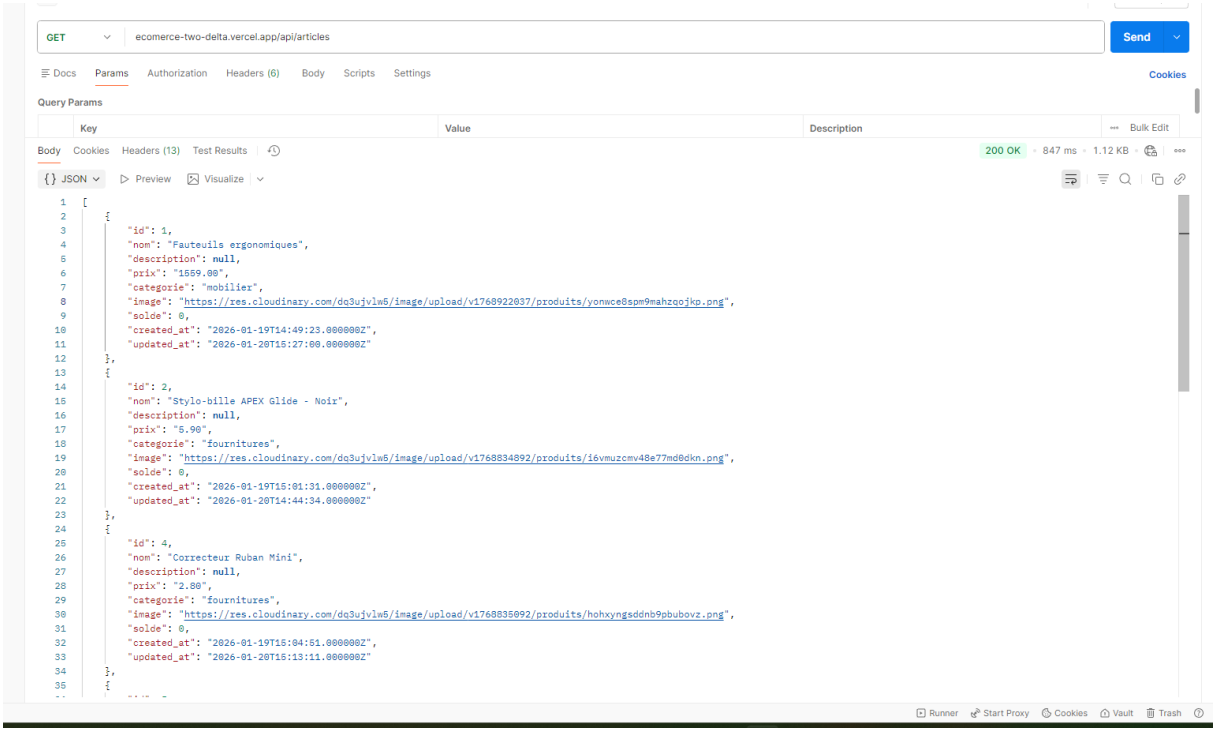
Symfony\Component\HttpFoundation\Request::HEADER_X_FORW

<?php

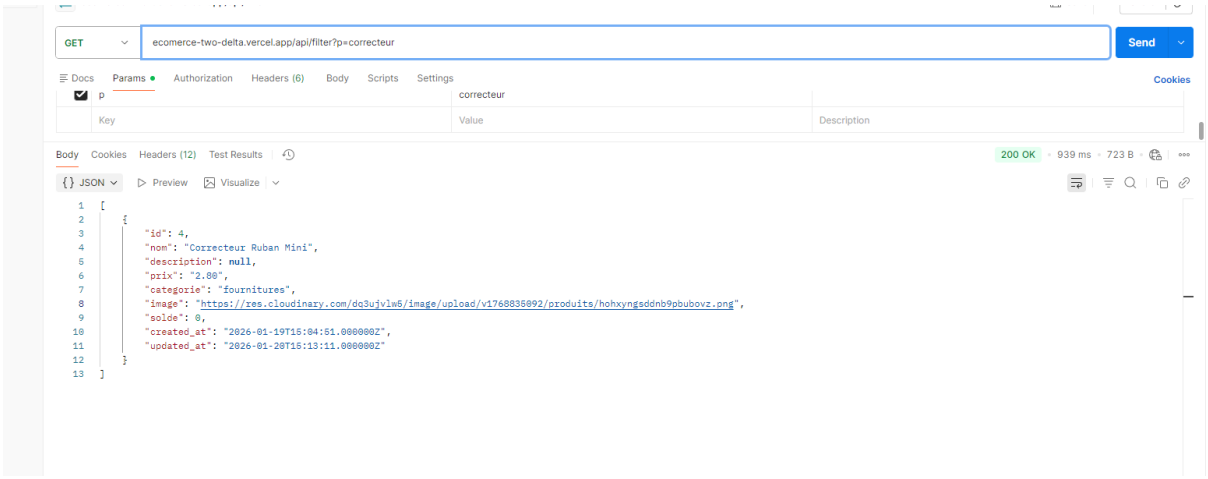
6- Des routes dans api.php :

```php
Route::apiResource('articles', ProduitControllerApi::class);
Route::get('/filter', [ProduitControllerApi::class, 'filtrer']);
```

7- Tester la methode index (get) :



8- Tester la methode filter (get) :



9- Recuperation des donnes dans react (useEffect() et axios) :

```
useEffect(() => {
  if (searchTerm.trim()) {
    searchProducts()
  } else {
    fetchProducts()
  }
}, [searchTerm])

const fetchProducts = async () => {
  try {
    setLoading(true)
    setError(null)
    const response = await axios.get(`${API_BASE}/articles`)
    const productList = Array.isArray(response.data) ? response.data : response.data.data || []
    setFilteredProducts(productList)
  } catch (err) {
    setError(err.message || 'Failed to fetch products')
    console.error('Fetch error:', err)
  } finally {
    setLoading(false)
  }
}
```

10- Axios.post() pour ajouter un produit :

```
const fetchProducts = async () => {
  try {
    setLoading(true)
    setError(null)
    const response = await axios.get(`${API_BASE}/articles`)
    const productList = Array.isArray(response.data) ? response.data : response.data.data || []
    setProducts(productList)
  } catch (err) {
    setError(err.message || 'Failed to fetch products')
    console.error('Fetch error:', err)
  } finally {
    setLoading(false)
  }
}
```