Rapport Maya-API

Léandro VEYRENC-CORDERO

Sommaire :

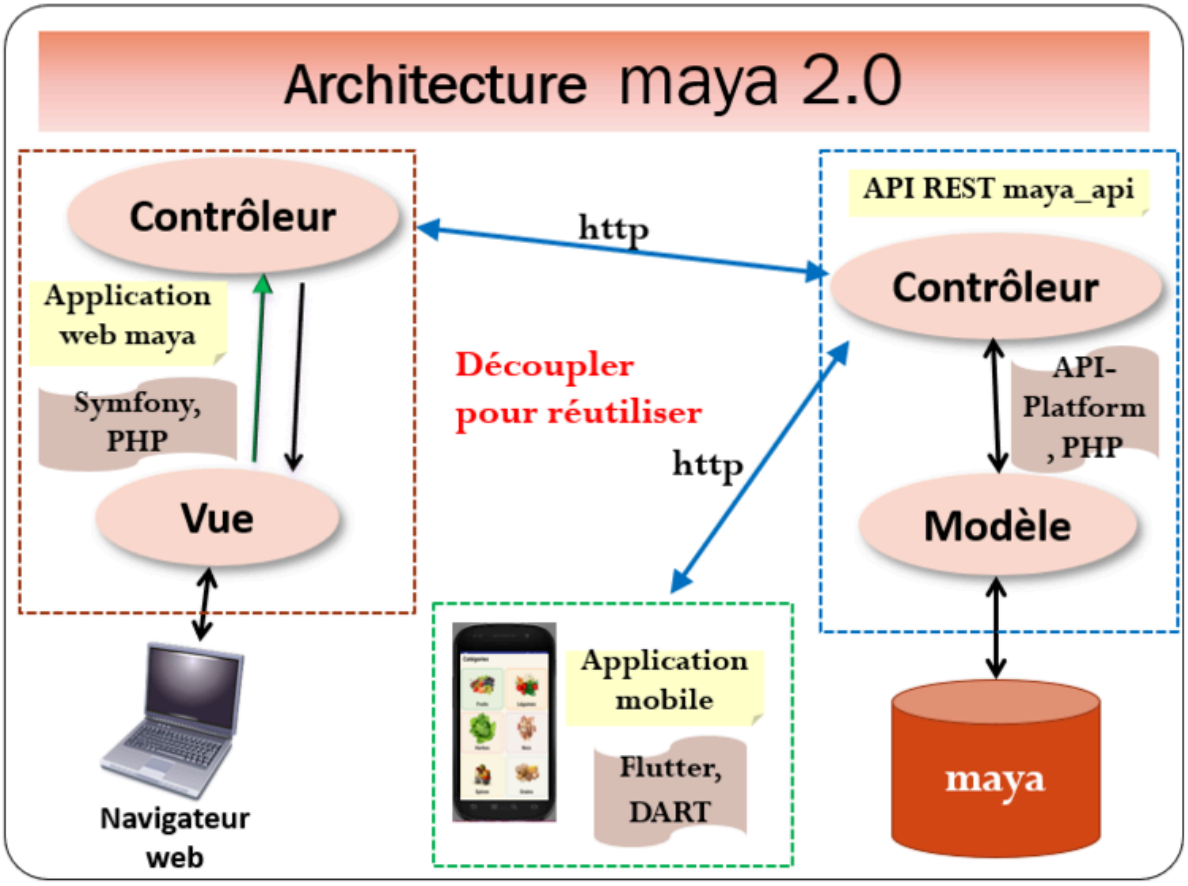
Rapport Maya-API

Léandro VEYRENC-CORDERO

1. Contexte du Projet

Le projet Maya 2.0 s'inscrit dans une transition vers une **Architecture Orientée Services (SOA)**. L'objectif principal est de découpler les couches applicatives pour favoriser la réutilisation des services web. L'écosystème Maya comprend :

- Une application web développée avec **Symfony** et **PHP**.
- Une application mobile utilisant **Flutter** et **DART**.
- Une API REST centralisée agissant comme service web pour distribuer les données depuis une base **MySQL**.



---

2. Développement de l'API REST

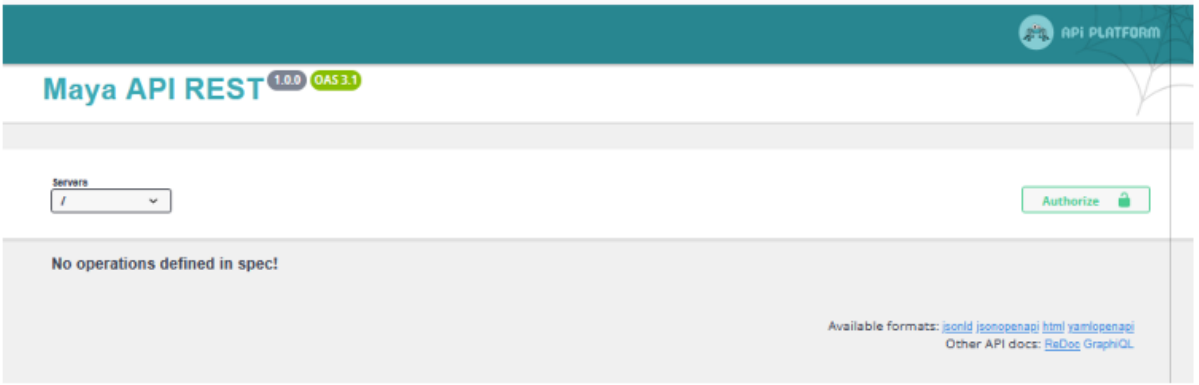
Le développement repose sur le framework **Symfony 7** couplé au bundle **API-Platform**.

2.1 Configuration de l'Environnement

- **Initialisation :** Création du squelette de l'application via la commande symfony new maya-api & composer require api

- **Installation d'API Platform :** Utilisation de Composer pour intégrer la plateforme, configurée pour exposer les endpoints via OpenAPI et Swagger UI.

```yaml
api_platform:
    title: Maya API REST
    version: 1.0.0
    defaults:
        stateless: true
        cache_headers:
            vary: ['Content-Type', 'Authorization', 'Origin']
```



- **Base de Données :** Connexion à MySQL via le fichier .env, utilisant une URL de base de données de type mysql://root:@127.0.0.1:3306/maya.

2.2 Gestion des Médias et Images

Pour permettre le téléchargement d'images de produits et de catégories, des configurations spécifiques ont été appliquées :

- **Formats autorisés :** Activation des formats json et multipart/form-data dans le fichier api_platform.yaml.

Fichier maya-api/config/packages/api_platform.yaml

```yaml
api_platform:
    title: Maya API REST
    version: 1.0.0
    defaults:
        stateless: true
        cache_headers:
            vary: ['Content-Type', 'Authorization', 'Origin']
    formats:
        json: ['application/json']
        multipart: ['multipart/form-data']
    error_formats:
        jsonproblem: ['application/problem+json']
        json: ['application/json']
```

D

- **VichUploaderBundle :** Installation de ce bundle pour lier physiquement les fichiers téléchargés aux entités Doctrine. composer require vich/uploader-bundle:^2.8

- **Mapping :** Configuration de dossiers de destination spécifiques (ex: /public/images/categories) avec renommage automatique via SmartUniqueNamer.

```yaml
mappings:
    categories:
        uri_prefix: /images/categories
        upload_destination: '%kernel.project_dir%/public/images/categories'
        namer: Vich\UploaderBundle\Naming\SmartUniqueNamer
        inject_on_load: false
        delete_on_update: true
        delete_on_remove: true
    produits:
        uri_prefix: /images/produits
        upload_destination: '%kernel.project_dir%/public/images/produits'
        namer: Vich\UploaderBundle\Naming\SmartUniqueNamer
        inject_on_load: false
        delete_on_update: true
        delete_on_remove: true
```

- **Composants personnalisés :** Création d'un MultipartDecoder et d'un UploadedFileDenormalizer pour permettre à Symfony de décoder correctement les fichiers reçus par l'API.

---

3. Implémentation de la Ressource "Categorie"

La ressource **Categorie** sert de modèle de référence pour les endpoints de l'API.

Rapport Maya-API

Léandro VEYRENC-CORDERO

- **Attributs PHP :** L'entité utilise l'attribut #[ApiResource] pour s'exposer automatiquement.

Fichier maya-api/src/Entity/Categorie.php

```php
<?php

namespace App\Entity;

use ApiPlatform\Metadata\ApiResource;
use App\Repository\CategorieRepository;
use Doctrine\ORM\Mapping as ORM;
use Symfony\Component\Validator\Constraints as Assert;

use Symfony\Component\HttpFoundation\File\File;
use Vich\UploaderBundle\Mapping\Annotation as Vich;

#[ORM\Entity(repositoryClass: CategorieRepository::class)]
#[Vich\Uploadable]
#[ApiResource]
class Categorie
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    private ?int $id = null;

    #[ORM\Column(length: 50)]
    #[Assert\NotBlank(message: 'Le libellé est obligatoire')]
    #[Assert\Length(
        min: 3,
        max: 50,
        minMessage: 'Le libellé doit comporter au moins {{ limit }} caractères',
        maxMessage: 'Le libellé ne peut pas dépasser {{ limit }} caractères',
    )]
    private ?string $libelle = null;
. . .
```

- **Groupes de Sérialisation :** Mise en place d'un contexte de normalisation (categorie:read) pour la lecture et de dénormalisation (categorie:write) pour l'écriture, permettant de contrôler finement les données exposées.

Rapport Maya-API

Léandro VEYRENC-CORDERO

```
#[ApiResource(
    normalizationContext: ['groups' => ['categorie:read']],
    denormalizationContext: ['groups' => ['categorie:write']],
```

```
#[ApiResource(
    normalizationContext: ['groups' => ['categorie:read']],
    denormalizationContext: ['groups' => ['categorie:write']],
    operations: [
        new Get(),
        new GetCollection(),
        new Post(
            outputFormats: ['json' => ['application/json']],
            inputFormats: ['multipart' => ['multipart/form-data']]
        ),
        new Delete(
            outputFormats: ['json' => ['application/json']],
            inputFormats: ['json' => ['application/json']]
        )
    ]
)]
```

```
class Categorie
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    #[Groups(['categorie:read', 'categorie:write'])]
    private ?int $id = null;
```

- **Filtrage :** Intégration de l'attribut OrderFilter pour permettre le tri automatique de la liste des catégories par libellé (ordre croissant).

```
)]
#[ApiFilter(OrderFilter::class, properties: ['libelle' => 'ASC'])]
class Categorie
```

D

---

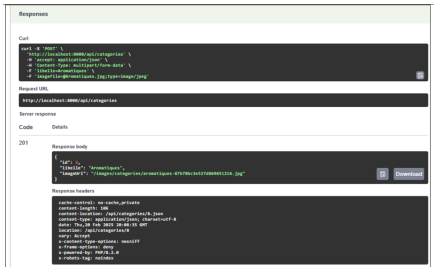
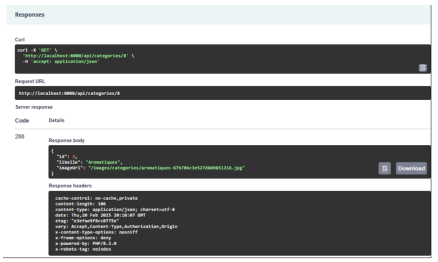
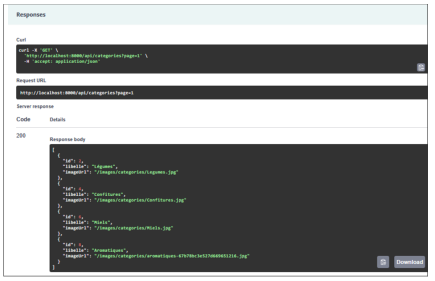
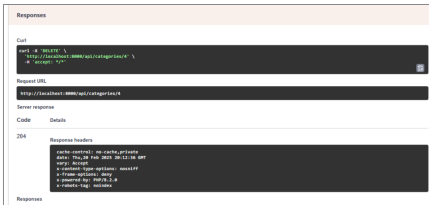
4. Tests et Opérations API

Les tests ont validé les points de terminaison (endpoints) suivants via Swagger UI:

Opération	Endpoint	Description	Résulat

Rapport Maya-API

Léandro VEYRENC-CORDERO

POST	/api/categories	Création d'une catégorie avec envoi d'image (multipart).	
GET	/api/categories/{id}	Récupération des détails d'une catégorie spécifique.	
GET	/api/categories	Récupération de la collection complète des catégories.	
DELETE	/api/categories/{id}	Suppression d'une ressource catégorie.	

Résultat technique : Lors d'une requête POST réussie, l'API renvoie un code **201** et un corps de réponse JSON incluant l'ID généré, le libellé et l'URL publique de l'image stockée.