Rapport Sprint 5 :

PHP Mastery : From Fundamentals to Frameworks

Réalisé par :

Sara Akharraz

Encadrant :

M.Abdelmajid Bendrif

M.Hamza Bahlaouane

Entreprise :

Void

Année universitaire 2025-2026

Syntaxe, configuration et outils PHP :

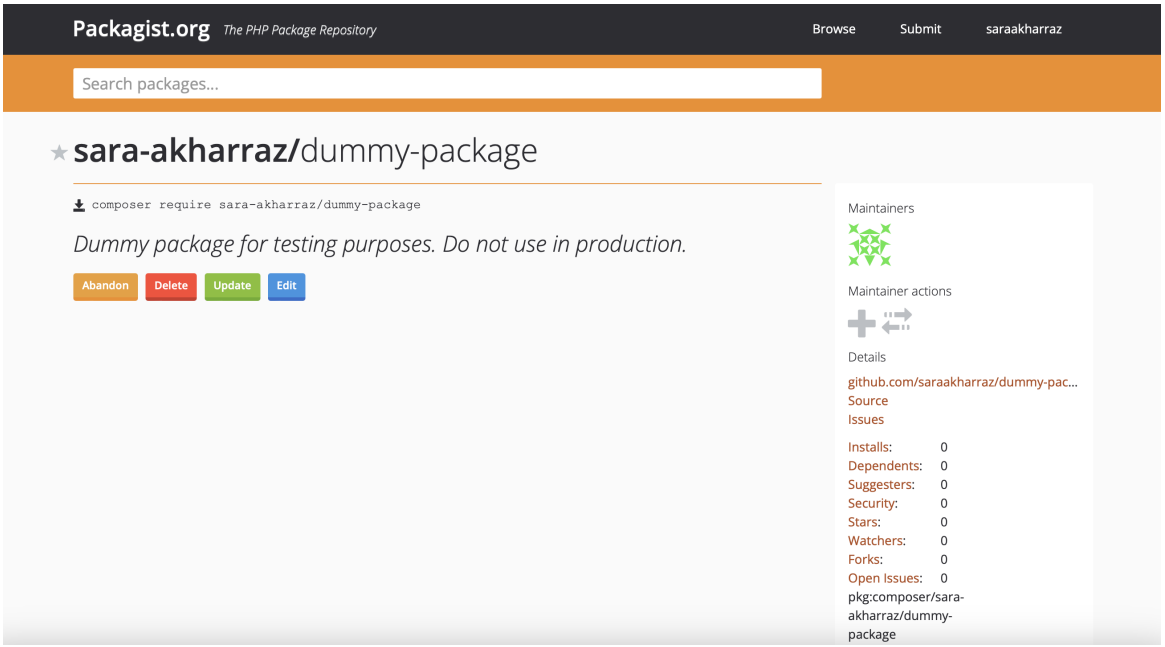
Objectifs :

— Acquérir de la pratique concrète en **PHP** et se familiariser avec sa syntaxe et ses particularités.

— Apprendre à créer, configurer et publier un package PHP factice pour se familiariser avec la gestion de projets et **Packagist**.

Ce module m'a permis d'apprendre les principes de base de PHP, de me familiariser avec sa syntaxe, ses fonctions et la structure d'un projet.

La création d'un dummy-package a été pratiquée en parallèle, et ensuite je l'ai publié sur Packagist.



Le lien vers le dépot : GitHub Repository

Créer votre propre framework :

Objectifs :

— Apprendre à concevoir et structurer un **framework PHP**, en commençant par un modèle MVC simple.

— Se familiariser avec **les composants Symfony** et vérifier la compréhension en installant et utilisant un package publié sur Packagist.

Construire un framework PHP MVC simple :

Pour mettre en pratique les connaissances acquises lors du premier module, j'ai implémenté un framework PHP from scratch en suivant le tutoriel Write Your Own PHP MVC Framework. L'objectif était de créer une application TODO en respectant l'architecture MVC.

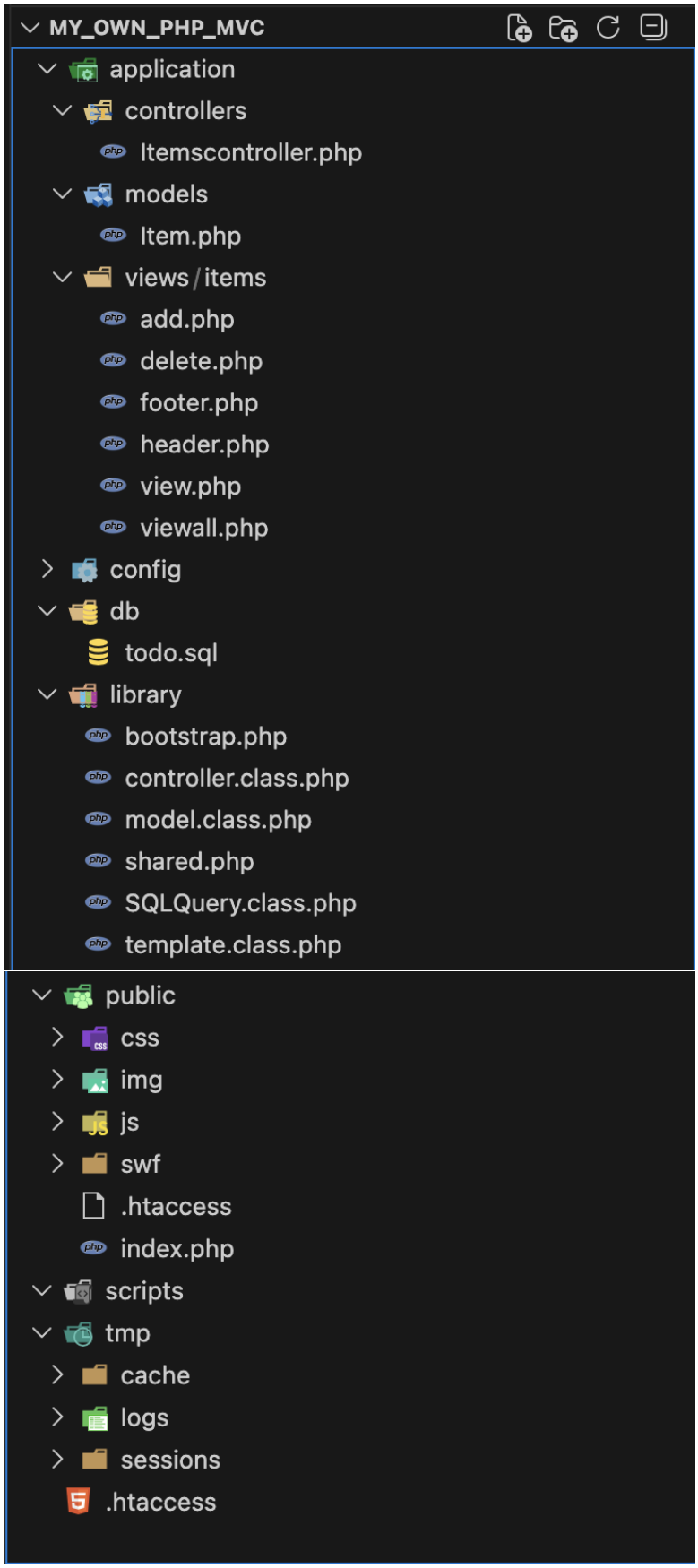


FIGURE 1 – Architecture du projet

Le mode **mod_rewrite** d'Apache permet de réécrire les URL pour les rendre plus lisibles et dirigées vers le bon fichier.
Voici la configuration utilisée dans le fichier `.htaccess` :

```
<IfModule mod_rewrite.c>
```

```
RewriteEngine on
RewriteRule ^$ public/ [L]
RewriteRule (.*) public/$1 [L]
</IfModule>
```

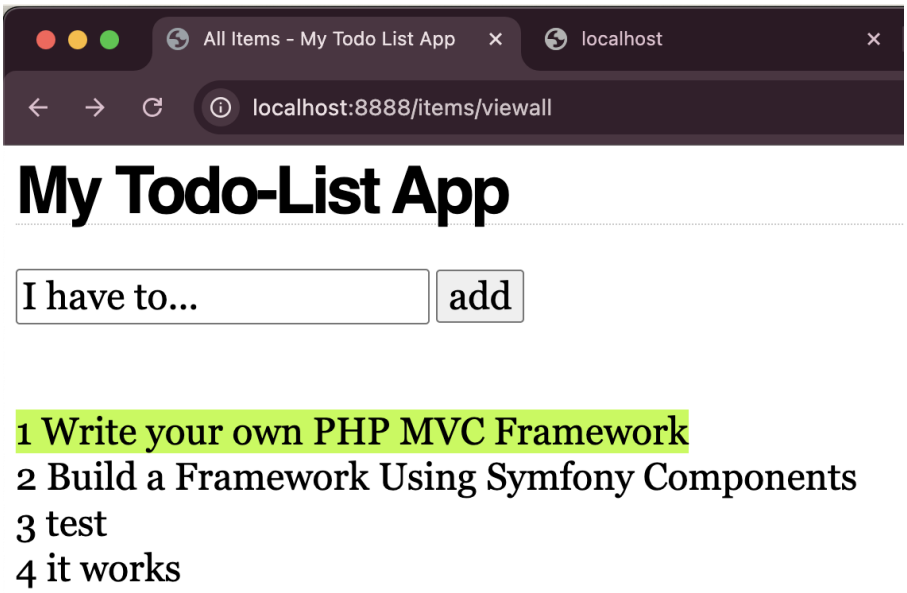
— `<IfModule mod_rewrite.c>` : Vérifie si le module `mod_rewrite` est activé.

— `RewriteEngine on` : Active le moteur de réécriture d'URL.

— `RewriteRule  public/ [L]` : Redirige la racine du site vers le dossier `public/`.

— `RewriteRule (.*) public/$1 [L]` : Redirige toutes les autres requêtes vers le dossier `public/`, en conservant le chemin demandé.

Résultat :



Le lien vers le dépot : GitHub Repository

Construire un framework en utilisant les composants Symfony :

Pour approfondir mes connaissances, j'ai suivi le guide Symfony : Create a PHP Framework Guide. L'objectif était de construire un framework PHP en utilisant les composants Symfony et de comprendre comment organiser un projet complexe tout en réutilisant des bibliothèques existantes.

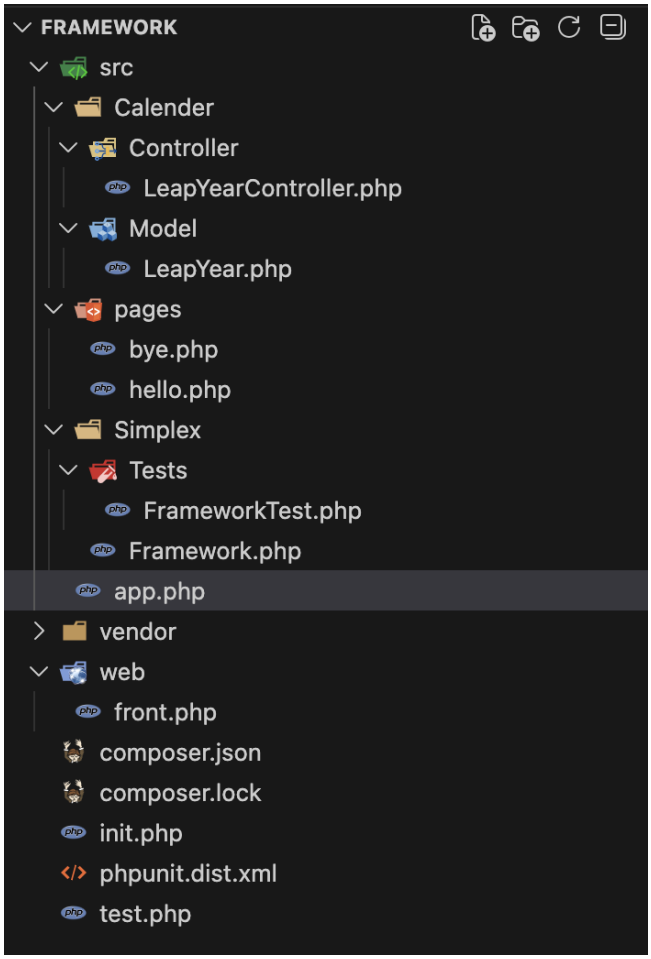


FIGURE 2 – Architecture du projet Symfony

Travailler avec les bases de données et la ligne de commande :

Objectifs :

— Développer une application CLI en PHP pour importer efficacement des fichiers CSV dans une base MySQL.

Pour mettre en pratique mes connaissances en PHP et bases de données, j'ai développé une application CLI pour importer des fichiers CSV dans une base MySQL.
L'application est construite avec le composant Symfony Console et utilise le package `league/csv` pour parser les fichiers.

Elle repose sur PDO et des requêtes SQL brutes, sans ORM, et suit une architecture claire :

— Point d'entrée CLI : `ImportCsvCommand`

— Lecteur de CSV : `CsvReader`

— Validateur de données : `CsvValidator`

— Importeur qui gère les insertions en lots : `CsvImporter`

— **Transactions :** Chaque batch d'insertion s'exécute dans une transaction, garantissant l'atomicité (tout ou rien).
En cas d'erreur, la transaction est annulée avec `rollback`.

— **Insertions en lots :** Les données sont insérées par groupes de lignes (batchs) pour améliorer les performances.

— **Gestion des doublons :** Les doublons sont ignorés avec `INSERT IGNORE`, ce qui évite les interruptions de l'import.

— **Performance :** Optimisation grâce aux colonnes indexées, aux insertions en batch et aux transactions.

— **Export et transfert :** La base peut être exportée avec `mysqldump`, compressée avec `gzip` et transférée via `scp`.

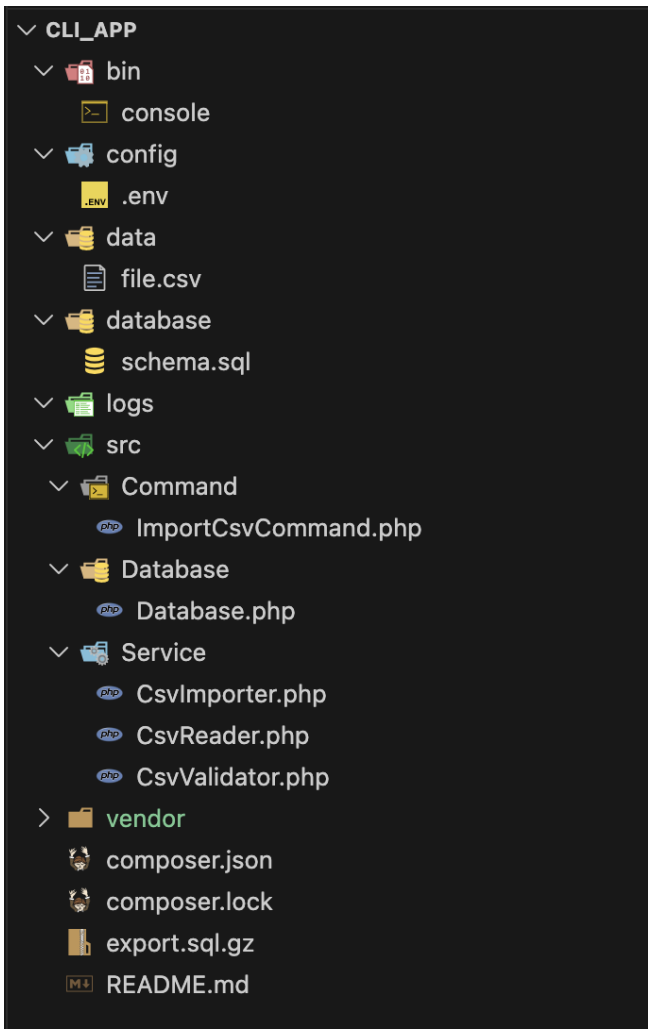
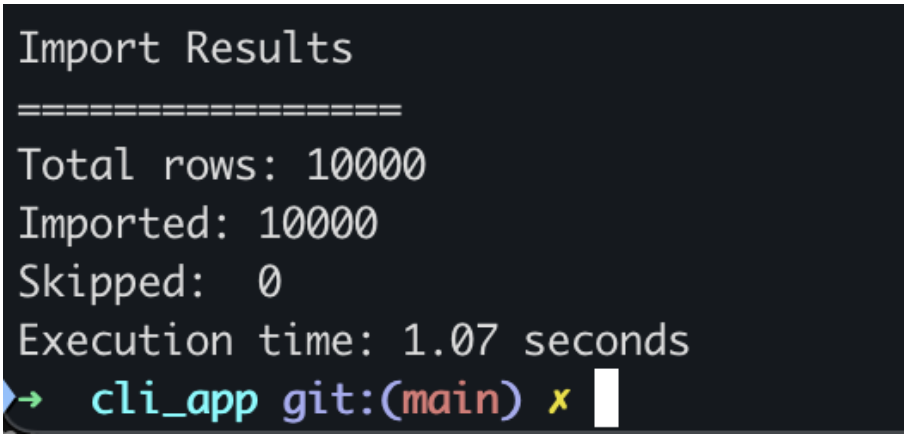


FIGURE 3 – Architecture de l'application CLI d'import CSV

On run via la commande :

```
./bin/console csv:import data/file.csv customers
```



Le lien vers le dépot : GitHub Repository

API REST avec Laravel :

Objectifs :

— Mettre en pratique l'utilisation de Laravel pour créer une API REST sécurisée et fonctionnelle, capable de filtrer, trier et paginer des données.

Pour approfondir l'utilisation de PHP et des frameworks modernes, une application Laravel a été mise en place pour exposer une API REST sur la base de données `customers`.
Un schéma JSON d'exemple (*Example Schema*) a été adapté afin de faciliter l'implémentation des filtres.
Plusieurs fonctionnalités ont été mises en œuvre pour l'API :

— Ajout d'un en-tête personnalisé dans la réponse (`x-api-version: v1`) pour indiquer la version de l'API.
— Mise en place d'une authentification basique pour sécuriser l'accès.
— Interaction avec l'API via Postman et Curl.
— Possibilité de filtrer, trier et paginer les données.

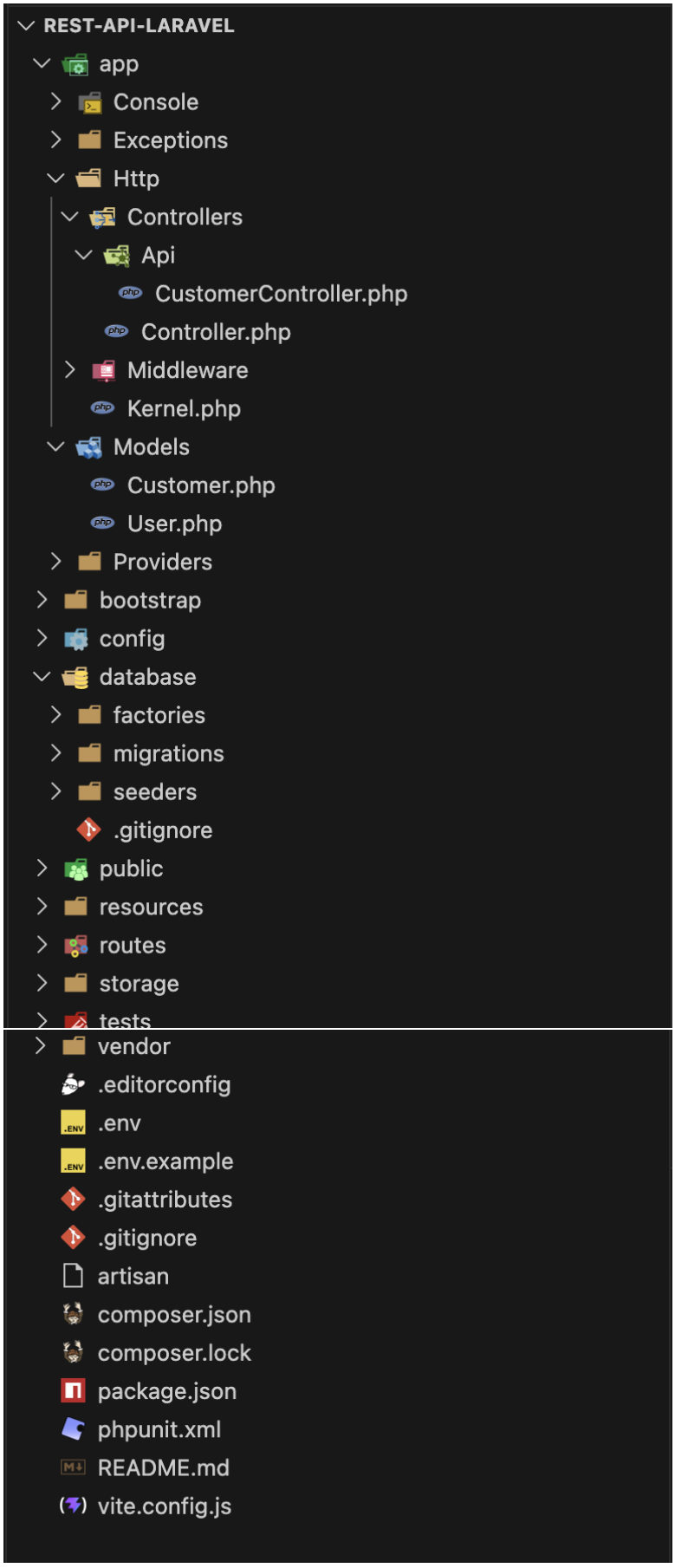
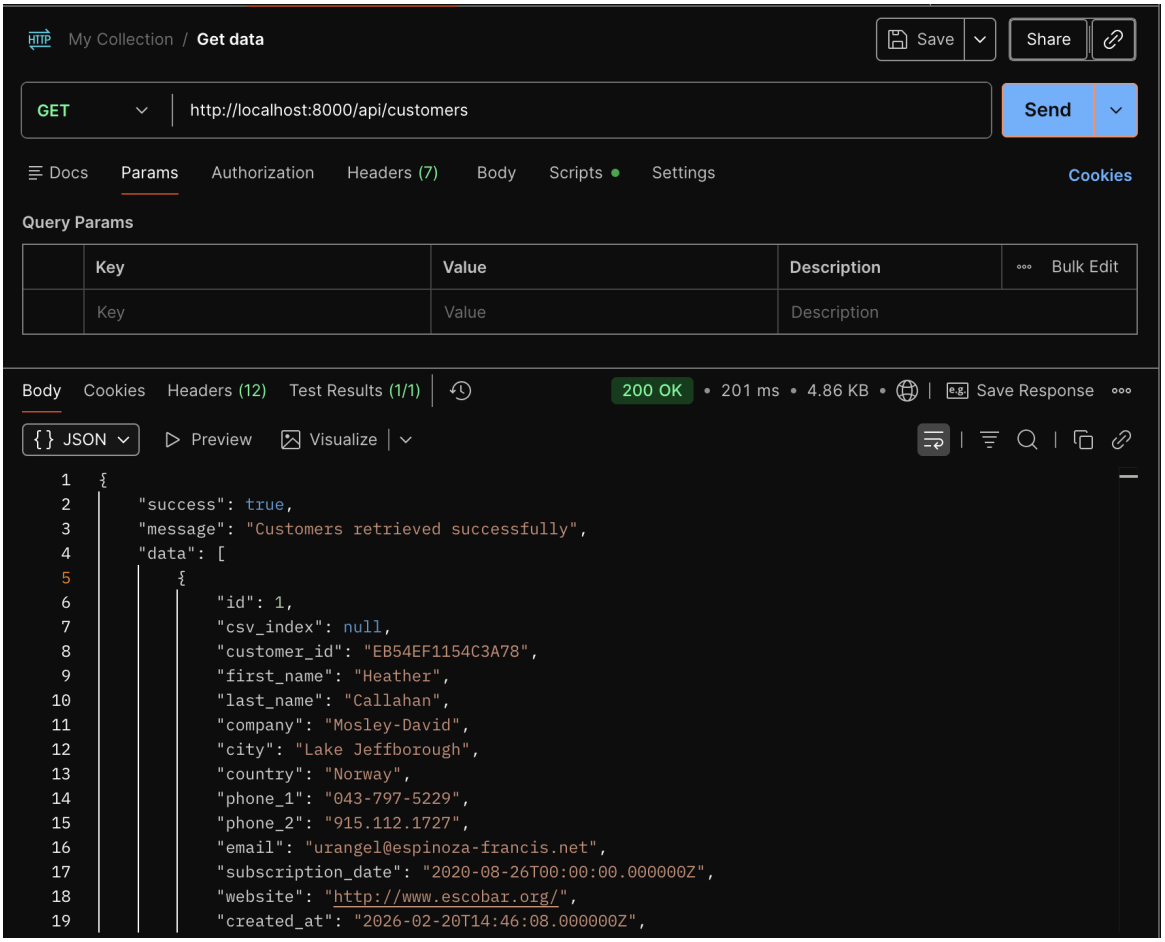
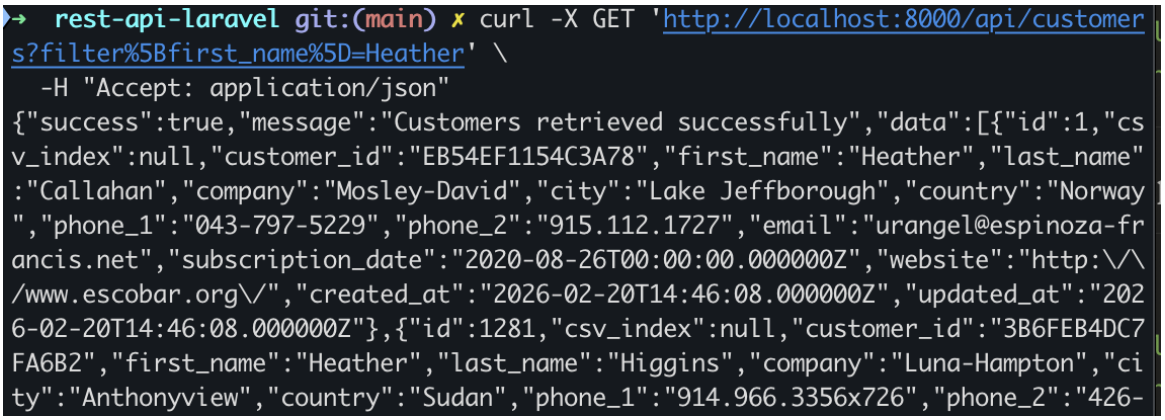


FIGURE 4 – Architecture de l'api rest Laravel

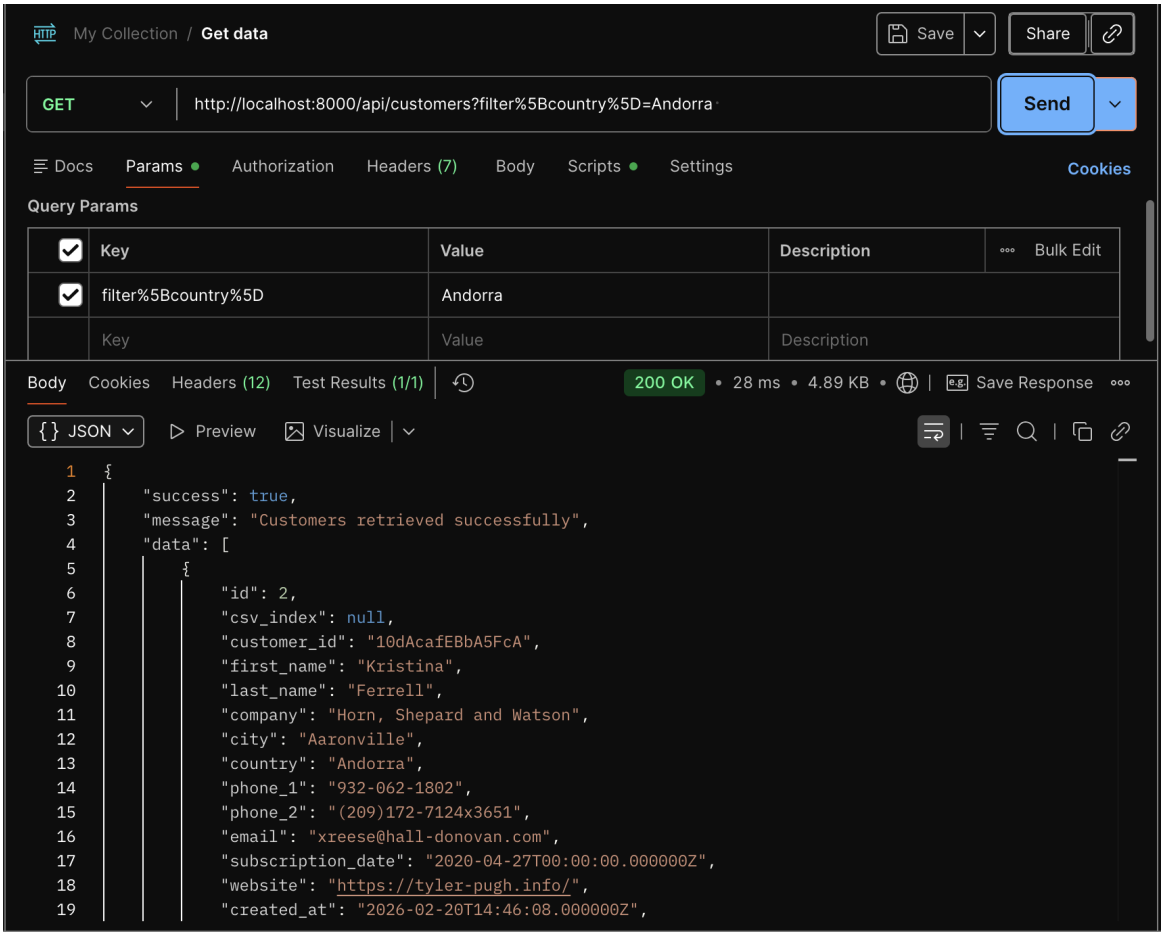
Les Tests : Get All Customers (No Auth) :



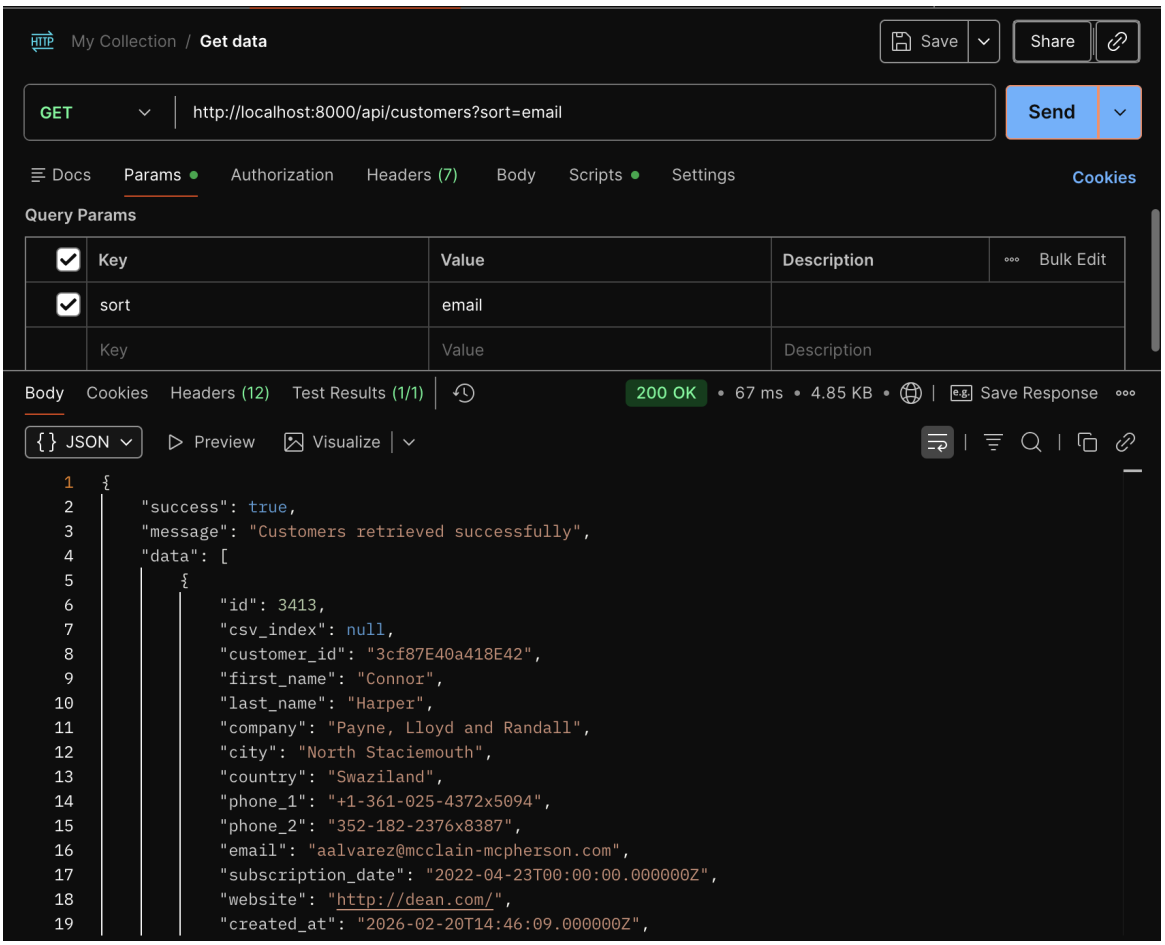
Filter by First name (Heather) :



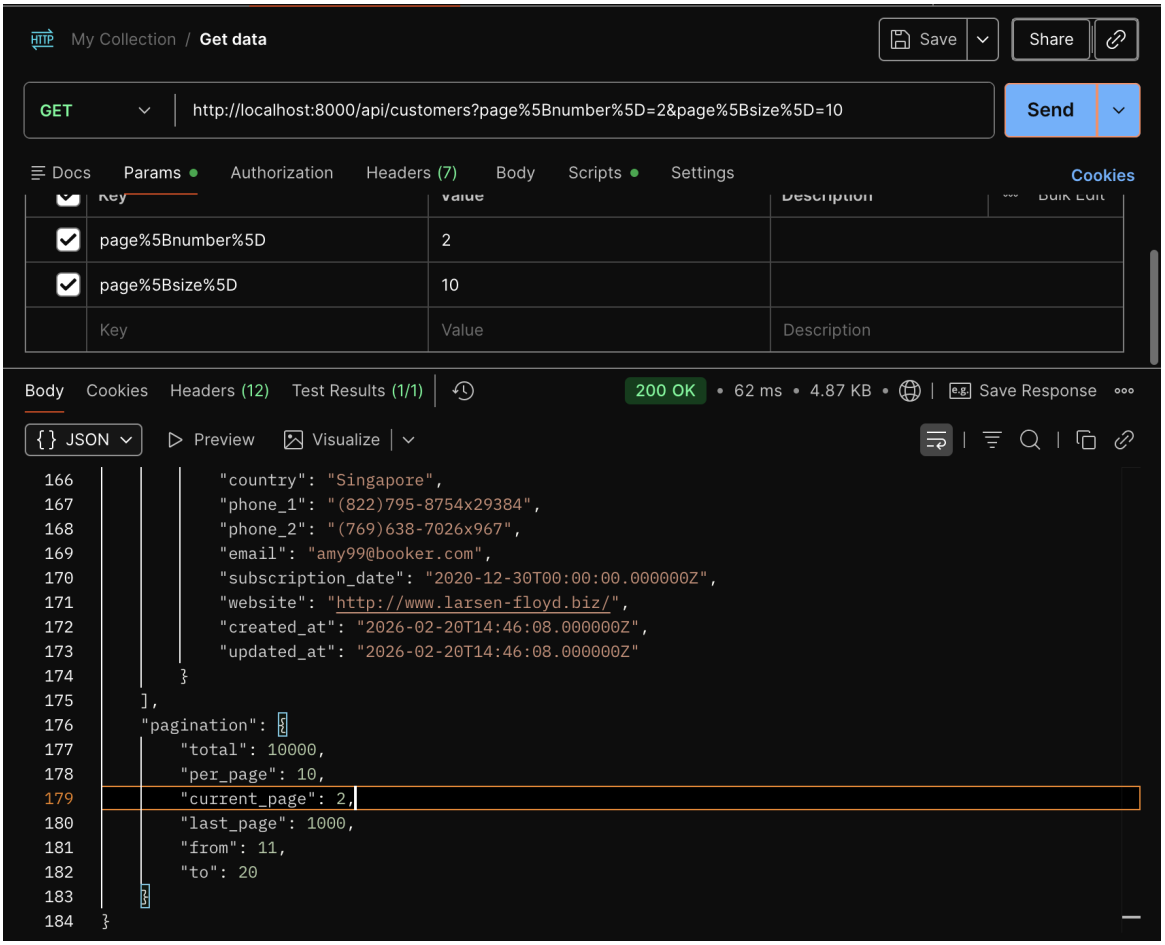
Filter by Country (Andorra) :



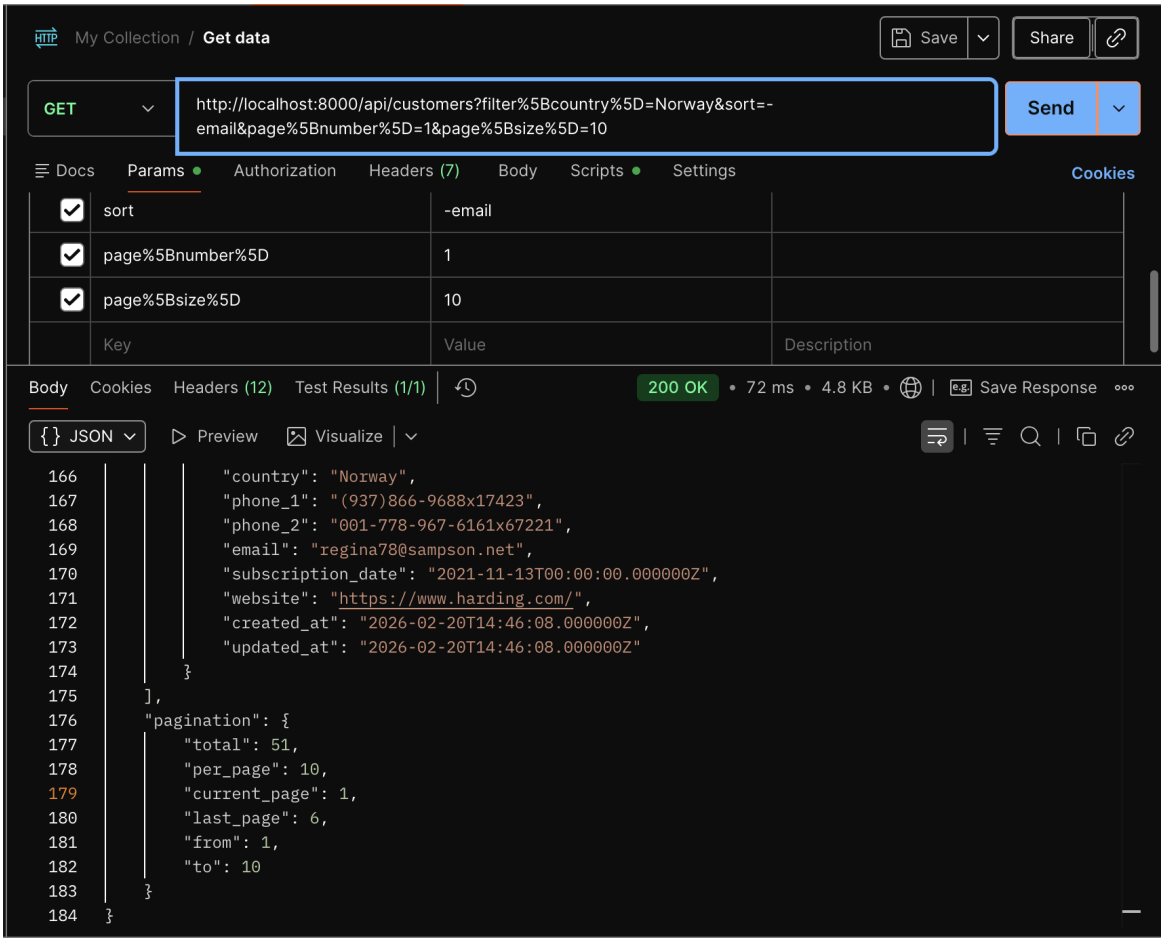
SORT BY EMAIL (ascending A→Z)



PAGE 2 with 10 items per page :



Filter by country + Sort by email + Pagination :



Le lien vers le dépot : GitHub Repository