

Working with Databases & CLI CSV Importer and Laravel API

Hatim Salhi

February 21, 2026

Contents

1	Introduction	2
2	Part 1 – Symfony CLI CSV Importer	2
2.1	Project Setup	2
2.2	Database Creation	3
2.3	Table Design	4
2.4	CSV Import Logic	4
2.5	Execution Time Measurement	4
2.6	CLI Execution	5
3	Export and Transfer of Data	5
3.1	Export using mysqldump	5
3.2	Compression	5
4	Part 2 – Laravel REST API	5
4.1	Laravel Setup	5
4.2	Customer Model	5
4.3	Custom Header Middleware	5
4.4	Basic Authentication	6
4.5	Filtering, Sorting and Pagination	6
5	Testing Tools	6
6	Conclusion	7

1 Introduction

The objective of this project is to practice backend development concepts including:

- Building a Command Line Interface (CLI) application using Symfony Console
- Importing data from a CSV file into a MySQL database
- Using PDO and RAW SQL with transactions
- Exporting and transferring database data
- Creating a REST API using Laravel
- Implementing filtering, sorting, pagination and basic authentication

Two main parts were implemented:

1. Symfony CLI application for CSV import
2. Laravel API connected to the imported data

2 Part 1 – Symfony CLI CSV Importer

2.1 Project Setup

The project was initialized using Composer and the following packages were installed:

- symfony/console
- league/csv
- symfony/dotenv

```
•
├── README.md
├── bin
│   └── console.php
├── composer.json
├── composer.lock
├── csv_demo.sql.gz
├── customers.csv
└── src
    └── Command
        └── vendor
            ├── autoload.php
            ├── composer
            ├── league
            ├── psr
            └── symfony

9 directories, 7 files
```

Figure 1: Project structure

2.2 Database Creation

A MySQL database named `csv_demo` was created. A user `csv_user` was created and granted privileges on the database.

```
1 CREATE DATABASE csv_demo;
2
3 CREATE USER 'csv_user'@'localhost' IDENTIFIED BY 'password';
```

```
4 GRANT ALL PRIVILEGES ON csv_demo.* TO 'csv_user'@'localhost';
5 FLUSH PRIVILEGES;
```

2.3 Table Design

A `customers` table was created to store the CSV data.

```
1 CREATE TABLE customers (
2     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
3     csv_index INT,
4     customer_id INT NULL,
5     first_name VARCHAR(100),
6     last_name VARCHAR(100),
7     company VARCHAR(150),
8     city VARCHAR(100),
9     country VARCHAR(100),
10    phone1 VARCHAR(50),
11    phone2 VARCHAR(50),
12    email VARCHAR(190),
13    website VARCHAR(190),
14    subscription_date DATE,
15    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
16);
```

2.4 CSV Import Logic

The League CSV Reader was used to read the CSV file. Data was inserted using PDO prepared statements.

All inserts were wrapped inside a database transaction:

```
1 $pdo->beginTransaction();
2
3 try {
4     foreach ($records as $record) {
5         $stmt->execute([...]);
6     }
7     $pdo->commit();
8 } catch (Exception $e) {
9     $pdo->rollBack();
10}
```

This guarantees that either all rows are inserted or none.

2.5 Execution Time Measurement

Execution time was measured using:

```
$start = hrtime(true); // nanoseconds
```

Figure 2: Execution Start

```
$end = hrtime(true);  
$seconds = ($end - $start) / 1_000_000_000;
```

Figure 3: Execution End

2.6 CLI Execution

```
● → csv-importer git:(main) php bin/console.php app:import-csv customers.csv  
Inserted rows: 10000  
Execution time: 1.3151 seconds
```

Figure 4: CSV Execution Result

3 Export and Transfer of Data

3.1 Export using mysqldump

```
mysqldump -u csv_user -p csv_demo > csv_demo.sql
```

Figure 5: Database export

3.2 Compression

```
csv-importer gzip -9 csv_demo.sql
```

Figure 6: Compressed SQL file

4 Part 2 – Laravel REST API

4.1 Laravel Setup

A new Laravel application was generated and connected to the existing database.

4.2 Customer Model

```
1 class Customer extends Model {  
2     protected $table = 'customers';  
3     public $timestamps = false;  
4 }
```

4.3 Custom Header Middleware

A middleware adds the header:

```
x-api-version: v1
```

```

public function handle(Request $request, Closure $next): Response
{
    $response = $next($request);
    return $response->header('x-api-version', env('API_VERSION', 'v1'));
}

```

Figure 7: API version header

4.4 Basic Authentication

API routes are protected using Basic Auth with credentials stored in `.env`.

4.5 Filtering, Sorting and Pagination

The endpoint supports:

- filter[name]
- filter[email]
- sort[name]
- sort[email]
- page[number]
- page[size]

Example request:

```

1 curl -u admin:admin123 \
2 "http://127.0.0.1:8000/api/users?filter[name]=Eladio&page[number]=1&
  page[size]=10"

```

```

+ customers-api curl -i --globoff -u admin:admin123 \
"http://127.0.0.1:8000/api/users?filter[name]=Eladio&page[number]=1&page[size]=10"
HTTP/1.1 200 OK
Date: Tue, 12 Feb 2024 14:21:41 GMT
Content-Type: application/json
Connection: close
X-Powered-By: PHP/8.5.2
Cache-Control: no-cache, private
DNT: 1
User-Agent: curl/8.2.1
Accept: */*
X-API-Version: v1
Access-Control-Allow-Origin: *

{"data": [{"id": 20007, "csv_index": 1, "customer_id": null, "first_name": "Heather", "last_name": "Callahan", "company": "Mosley-David", "city": "Lake Jeffborough", "country": "Norway", "phone": "+43-797-5229", "email": "urangel@espinosa-francis.net", "website": "http://www.escoabar.org/", "subscription_date": "2020-08-26", "created_at": "2024-02-20 12:36:50"}, {"id": 10007, "csv_index": 1, "customer_id": null, "first_name": "Heather", "last_name": "Callahan", "company": "Mosley-David", "city": "Lake Jeffborough", "country": "Norway", "phone": "+43-797-5229", "email": "urangel@espinosa-francis.net", "website": "http://www.escoabar.org/", "subscription_date": "2020-08-26", "created_at": "2024-02-20 12:36:50"}, {"id": 24544, "csv_index": 1, "customer_id": null, "first_name": "Heather", "last_name": "Callahan", "company": "Mosley-David", "city": "Lake Jeffborough", "country": "Norway", "phone": "+43-797-5229", "email": "urangel@espinosa-francis.net", "website": "http://www.escoabar.org/", "subscription_date": "2020-08-26", "created_at": "2024-02-20 10:46:37"}, {"id": 4538, "csv_index": 1, "customer_id": null, "first_name": "Heather", "last_name": "Mercado", "company": "Le-Osborne", "city": "Luisville", "country": "Guatemala", "phone": "+502-1233876411", "email": "theresamitchell@bradford.org", "website": "http://v.popo.com/", "subscription_date": "2022-04-27", "created_at": "2024-02-20 12:36:50"}, {"id": 10454, "csv_index": 1, "customer_id": null, "first_name": "Heather", "last_name": "Mercado", "company": "Le-Osborne", "city": "Luisville", "country": "Guatemala", "phone": "+502-234-685-9673", "email": "theresamitchell@bradford.org", "website": "http://v.popo.com/", "subscription_date": "2022-04-27", "created_at": "2024-02-20 10:47:01"}, {"id": 1, "csv_index": 1, "customer_id": null, "first_name": "Heather", "last_name": "Meredith", "company": "Le-Osborne", "city": "Luisville", "country": "Guatemala", "phone": "+502-234-685-9673", "email": "theresamitchell@bradford.org", "website": "http://v.popo.com/", "subscription_date": "2022-04-27", "created_at": "2024-02-20 10:47:01"}], "links": [{"self": "http://127.0.0.1:8000/api/users?filter%5Bname%5D=Heather&filter%5Bemail%5D=&sort%5Bemail%5D=desc&page=1", "prev": null}], "meta": {"number": 1, "size": 24, "pages": 15}}

```

Figure 8: API JSON response (filtered)

5 Testing Tools

API was tested using:

- curl
- Postman

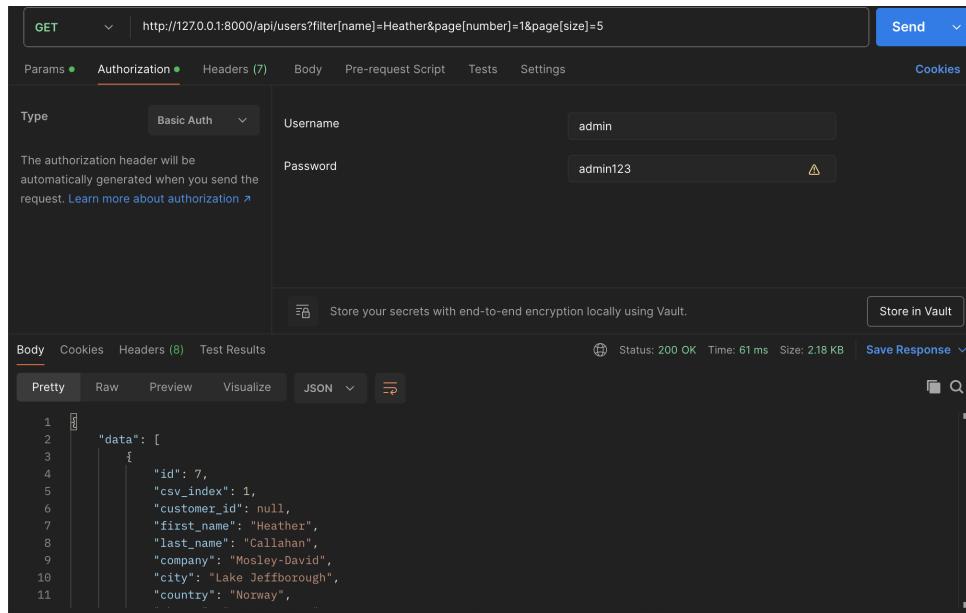


Figure 9: Postman testing

6 Conclusion

This project demonstrated how to:

- Build a CLI tool with Symfony
- Import large CSV files safely
- Use database transactions
- Export and transfer databases
- Create a secure Laravel REST API
- Implement filtering, sorting and pagination

These skills reflect real-world backend development practices.