UD 2

2.3 CSV

¿Qué veremos hoy?

- Introducción a CSV
- Comparativa
- CSV y Laravel 11.x

Introducción

Un **CSV (Comma-Separated Values)** es un formato de archivo sencillo que almacena datos tabulares en texto plano.

Cada línea del archivo representa un registro, y los valores de cada columna están separados por un delimitador, típicamente una coma (`,`), aunque a veces se usan otros delimitadores como el punto y coma (`;`) o tabulaciones.

Ventajas

- **Simplicidad**:
  - Es fácil de leer y escribir manualmente.
- **Hojas de cálculo:**
  - Es compatible con las aplicaciones de hoja de cálculo más populares, como Microsoft Excel o Google Sheets.
- **Portabilidad**:
  - Es un formato ampliamente adoptado y soportado en diferentes plataformas y lenguajes.

Desventajas

- **Tipos de datos**:
  - No es adecuado para datos jerárquicos (como objetos o listas dentro de listas).
  - No describe los tipos de datos ni incluye información adicional (como claves, estructura o validación).
- **Problemas con caracteres especiales**:
  - El uso de comas, saltos de línea o caracteres especiales en los datos puede causar problemas si no se escapan adecuadamente.
- **Ambigüedad del delimitador**:
  - No existe un estándar universal para el delimitador; algunas aplicaciones usan comas, otras punto y coma, lo que puede generar incompatibilidades.
- **Dificultad para representar relaciones complejas**:
  - No puede modelar fácilmente datos con relaciones entre entidades o estructuras jerárquicas.

Comparativa

Texto plano vs JSON vs CSV

Aspecto	Texto plano	JSON	CSV
Estructura	Sin estructura predefinida, solo texto.	Jerárquica (objetos y arrays).	Tabular (filas y columnas).
Facilidad de lectura	Fácil si los datos son simples.	Fácil para datos complejos y etiquetados.	Fácil para datos tabulares.
Portabilidad	Portabilidad universal pero sin organización.	Ideal para APIs y datos estructurados.	Amplia compatibilidad con hojas de cálculo.
Tamaño del archivo	Generalmente pequeño si no hay redundancia.	Más grande debido a las etiquetas.	Más pequeño que JSON.
Tipos de datos soportados	Solo texto, sin definición de tipo.	Texto, números, booleanos, arrays, objetos.	Texto y números básicos.
Flexibilidad	Muy limitado, solo texto lineal.	Muy flexible para datos anidados o jerárquicos.	Limitado a tablas planas.
Usabilidad	Útil para notas o datos no estructurados.	Ideal para datos estructurados y complejos.	Ideal para datos tabulares y hojas de cálculo.
Complejidad de parsing	Nula, pero sin análisis estructurado posible.	Moderada (requiere librerías para procesar).	Fácil.

Laravel y CSV

1. Instalación y configuración de las librerias necesarias.
2. Procesar peticiones en Laravel.

Instalación de las librerias necesarias

- Añadimos el paquete con Composer:

```
composer require maatwebsite/excel
```

- Publica el archivo de configuración:

```
php artisan vendor:publish --
provider="Maatwebsite\Excel\ExcelServiceProvider"
```

Ejemplos

Ejemplo de CSV

```
name,email,phone
John Doe,johndoe@example.com,123-456-7890
Jane Smith,janesmith@example.com,098-765-4321
Alice Johnson,alicejohnson@example.com,555-123-4567
Bob Brown,bobbrown@example.com,444-555-6666
```

Cargar CSV

```
public function cargarCSV(Request $request) {
        $request->validate([
            'file' => 'required'
        ]);
        $file = $request->file('file');
        $data = Excel::toArray(null, $file);
        foreach ($data[0] as $row) {
            $name = $row[0];
            $email = $row[1];
            $phone = $row[2];
            $user = new User(); // En este caso, estoy guardando el User en BD
            $user->name = $name;
            $user->email = $email;
            $user->phone = $phone;
            $user->save();
        }
        return response()->json(['mensaje' => 'Usuarios creados con éxito.'], 201);
    }
```

CSV to JSON

```
public function convertCsvToJson()
{
    $request->validate([
        'filename' => 'required'
    ]);

    if (!Storage::exists($request->filename)) {
        return response()->json(['error' => 'El archivo no existe'], 404);
    }

    $collection = Excel::toCollection(null, Storage::path($filePath));

    $data = $collection->first();
    $headers = $data->shift();
    $jsonData = $data->map(function ($row) use ($headers) {
        return array_combine($headers->toArray(), $row->toArray());
    });

    return response()->json($jsonData);
}
```