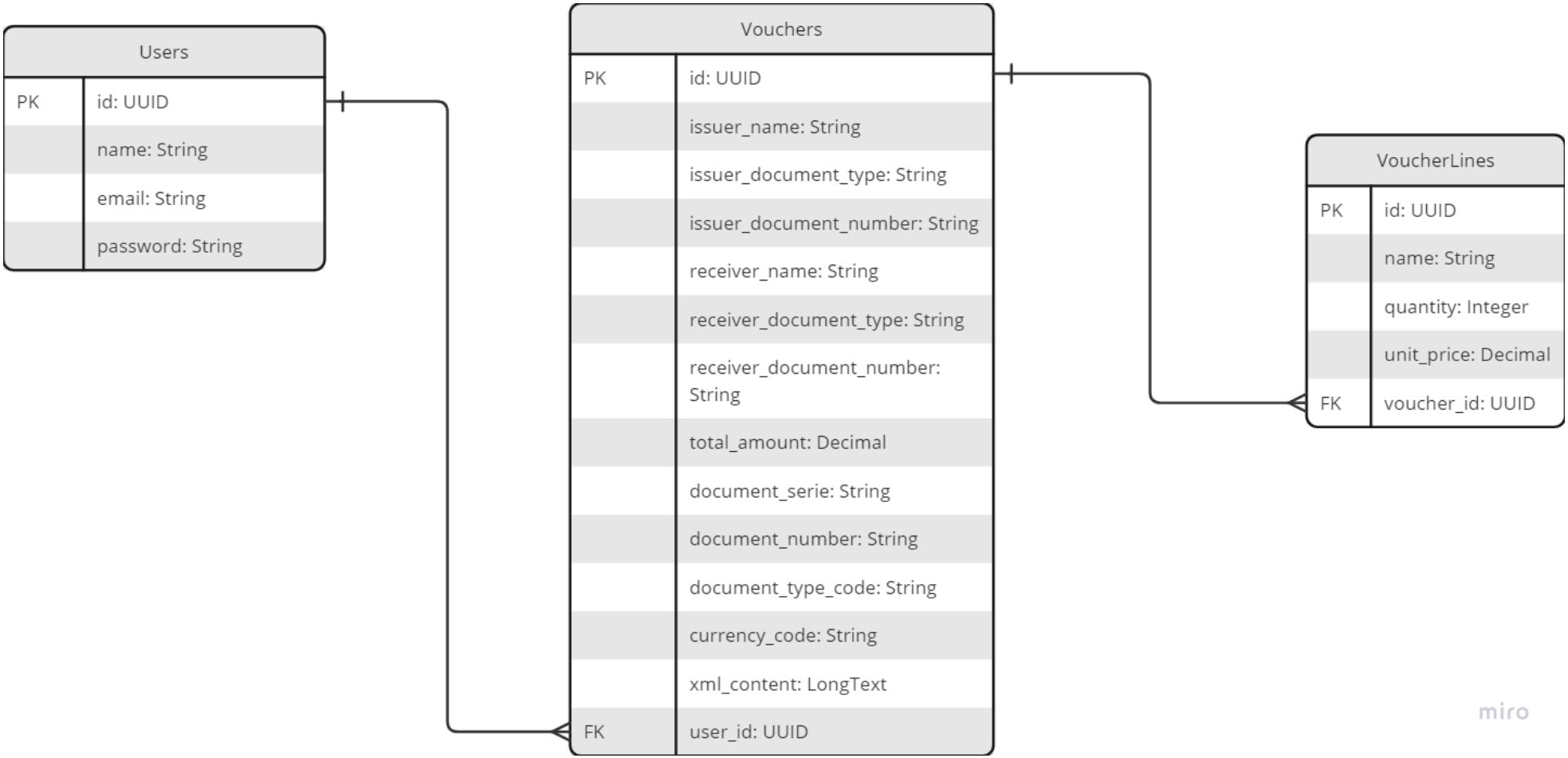
DIAGRAMA ENTIDAD – RELACIÓN



NUEVAS FUNCIONALIDADES

1. Registro de serie, número, tipo del comprobante y moneda.

1.1. Modificación en las Migraciones

Empezaremos En esta sección se describe la implementación de los nuevos campos document_serie, document_number, document_type_code y currency_code en la tabla vouchers, que se utilizarán para almacenar información clave de los comprobantes.

Descripción de los Cambios

Para cumplir con los requisitos solicitados, se realizaron las siguientes modificaciones en la base de datos, utilizando como como referencia la "Guía de Elaboración de Documentos XML Factura Electrónica UBL 2.1":

- document_serie: Campo de tipo string con longitud de 4 caracteres, para almacenar la serie alfanumérica del comprobante.
- document_number: Campo de tipo string con longitud de hasta 8 caracteres, para almacenar el número correlativo del comprobante.
- document_type_code: Campo de tipo string con longitud de 3 caracteres, que almacena el código del tipo de comprobante (por ejemplo, "01" para factura).
- currency_code: Campo de tipo string con longitud de 3 caracteres, para el código de moneda (por ejemplo, "PEN").

```php
public function up(): void
    {
        Schema::create('vouchers', function (Blueprint $table) {
            $table->uuid('id')->primary();
            $table->string('issuer_name');
            $table->string('issuer_document_type');
            $table->string('issuer_document_number');
            $table->string('receiver_name');
            $table->string('receiver_document_type');
            $table->string('receiver_document_number');
            $table->decimal('total_amount', 8, 2);

            // DATOS SOLICITADOS
            $table->string('document_serie', 4);
            $table->string('document_number', 8);
            $table->string('document_type_code', 3);
            $table->string('currency_code', 3);

            $table->longText('xml_content');
            $table->uuid('user_id');
            $table->timestamps();
            $table->softDeletes();

            $table->foreign('user_id')
                ->references('id')
                ->on('users')
                ->cascadeOnUpdate()
                ->cascadeOnDelete();
        });
    }
```

Además, se agrego la funcion Dow para asegurar que el sistema sigue prácticas de desarrollo sólidas, manteniendo un código limpio, estándar y capaz de manejar reversiones de manera segura en entornos de producción

```php
public function down(): void
    {
        Schema::table('vouchers', function (Blueprint $table) {
            $table->dropColumn(['document_serie', 'document_number', 'document_type_code', 'currency_code']);

        });
    }
```

1.2. Modificaciones en los Modelos

Para soportar los nuevos campos en la tabla vouchers, se realizaron las siguientes modificaciones en el modelo Voucher.

Descripción de los Cambios

Se añadieron los siguientes campos al array $fillable del modelo Voucher, lo que permite la asignación masiva de estos atributos:

- document_serie: Almacena la serie alfanumérica del comprobante.
- document_number: Almacena el número correlativo del comprobante.
- document_type_code: Almacena el código del tipo de comprobante.
- currency_code: Almacena el código de la moneda.

```php
protected $fillable = [
        'issuer_name',
        'issuer_document_type',
        'issuer_document_number',
        'receiver_name',
        'receiver_document_type',
        'receiver_document_number',
        'total_amount',
        'xml_content',
        'user_id',
        // DATOS SOLICITADOS
        'document_serie',
        'document_number',
        'document_type_code',
        'currency_code',
    ];
```

De esta manera permitimos que los mismos sean manipulados de manera segura a través del framework Laravel.

1.3. Implementación de Extracción de Variables desde Archivos XML

Se agregaron las para cumplir con los requisitos solicitados, extrayendo estos datos desde el contenido XML utilizando la librería SimpleXMLElement. Esto garantiza que la información clave del comprobante, esté correctamente almacenada en la base de datos.

Descripción de los Cambios

Separación de Serie y Número:

- Se usa explode('-', $documentID) para dividir la serie y el número del documento extraído de <cbc:ID>.
- Se verifica que el explode haya producido exactamente dos partes antes de asignarlas a las variables.

Manejo de Errores:

- Se envuelve todo el proceso en un bloque try-catch para capturar y manejar cualquier excepción que ocurra, registrando los errores en los logs de la aplicación.
- Se usan valores por defecto (?? '' o ?? 0) al extraer información del XML, lo que ayuda a evitar errores si algún nodo no se encuentra.

```php

// DATOS SOLICITADOS
$documentID = (string) $xml->xpath('//cbc:ID')[0];

$parts = explode('-', $documentID);
 if (count($parts) !== 2) {
     throw new Exception("El formato de cbc:ID no es válido.");
 }
[$documentSerie, $documentNumber] = $parts;

$documentTypeCode = (string) $xml->xpath('//cbc:InvoiceTypeCode')[0] ?? '';
$currencyCode = (string) $xml->xpath('//cbc:DocumentCurrencyCode')[0] ?? '';

```

```php
'user_id' => $user->id,
                // DATOS SOLICITADOS
                'document_serie' => $documentSerie,
                'document_number' => $documentNumber,
                'document_type_code' => $documentTypeCode,
                'currency_code' => $currencyCode,
            ]);
```

```php
} catch (Exception $e) {
            Log::error("Error al procesar el XML: " . $e->getMessage());
            throw new Exception("Error al procesar el XML: " . $e->getMessage());
        }
```

1.4. Actualización de VoucherResource para Incluir Nuevas Variables de Comprobante

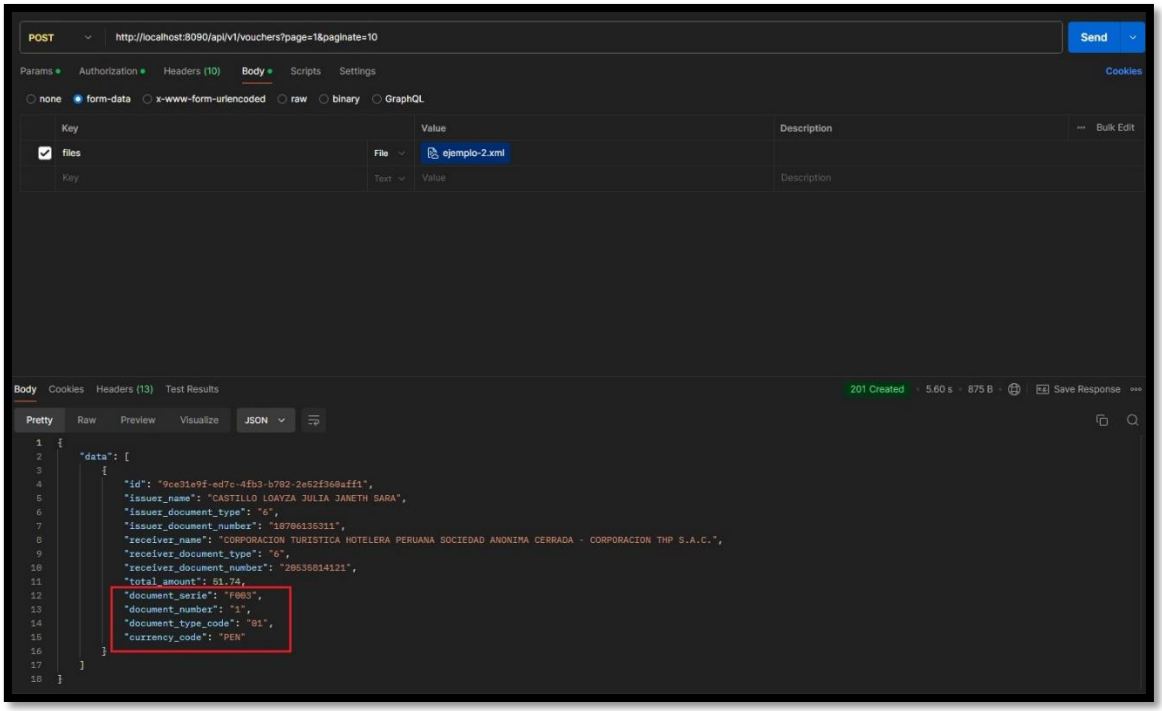
La edición de VoucherResource.php se realizó para permitir que las nuevas variables sean correctamente incluidas en las respuestas JSON cuando se devuelven los datos de los comprobantes.

```php
// DATOS SOLICITADOS
            'document_serie' => $this->resource->document_serie,
            'document_number' => $this->resource->document_number,
            'document_type_code' => $this->resource->document_type_code,
            'currency_code' => $this->resource->currency_code,
```

1.5. Incorporación de Nuevas Variables en la Vista de Correo Electrónico de Comprobantes

1.6. Prueba de API REST para la Creación de Comprobantes desde Archivos XML mediante Postman

Para validar la funcionalidad del API REST que permite la creación de comprobantes desde archivos XML, se utilizó Postman como herramienta de prueba. Se configuró una petición POST a la URL http://localhost:8090/api/v1/vouchers?page=1&paginate=10, en la cual se subió un archivo XML llamado ejemplo-2.xml. Este proceso permitió verificar que la API extrae correctamente los datos del archivo XML y crea los comprobantes con las nuevas variables implementadas, asegurando así la funcionalidad y el manejo adecuado de los recursos.



2. Carga de comprobantes en segundo plano

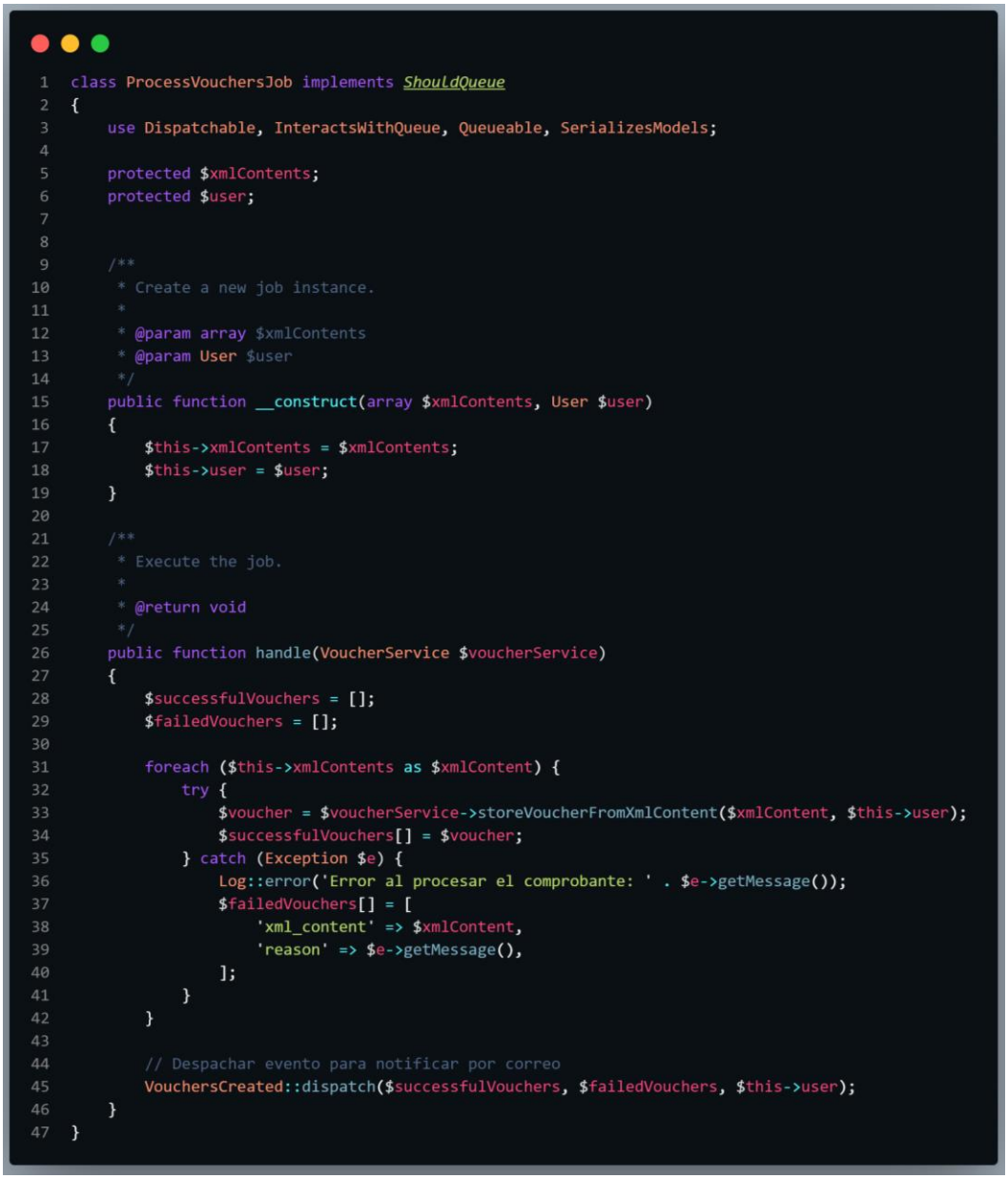
2.1. Creación e implementación de Jobs en Laravel

Se implementó una funcionalidad para que la carga de comprobantes se ejecute en segundo plano, utilizando un trabajo (Job) en Laravel. Además, en lugar de enviar una única notificación por correo tras la subida de los comprobantes, se estableció el envío de dos listados: uno con los comprobantes que se subieron correctamente y otro con aquellos que no pudieron registrarse, especificando la razón del fallo.

Descripción del Código:

- Propiedades: El Job almacena el contenido de los archivos XML y la información del usuario que subió los comprobantes.
- Método handle: Se encarga de procesar cada comprobante:
- Almacenamiento de Comprobantes: Se utiliza el servicio VoucherService para intentar almacenar cada comprobante. Si el almacenamiento es exitoso, el comprobante se añade a la lista de successfulVouchers.
- Manejo de Errores: Si ocurre un error durante el almacenamiento, el comprobante junto con la razón del fallo se añade a la lista de failedVouchers. Los errores se registran en el log del sistema para su revisión.
- Notificación: Una vez procesados todos los comprobantes, se dispara un evento (VouchersCreated) que se encarga de enviar una notificación por correo al

usuario, incluyendo ambos listados: los comprobantes exitosos y los fallidos con sus respectivas razones.



```php
class ProcessVouchersJob implements ShouldQueue
{
    use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;

    protected $xmlContents;
    protected $user;


    /**
     * Create a new job instance.
     *
     * @param array $xmlContents
     * @param User $user
     */
    public function __construct(array $xmlContents, User $user)
    {
        $this->xmlContents = $xmlContents;
        $this->user = $user;
    }

    /**
     * Execute the job.
     *
     * @return void
     */
    public function handle(VoucherService $voucherService)
    {
        $successfulVouchers = [];
        $failedVouchers = [];

        foreach ($this->xmlContents as $xmlContent) {
            try {
                $voucher = $voucherService->storeVoucherFromXmlContent($xmlContent, $this->user);
                $successfulVouchers[] = $voucher;
            } catch (Exception $e) {
                Log::error('Error al procesar el comprobante: ' . $e->getMessage());
                $failedVouchers[] = [
                    'xml_content' => $xmlContent,
                    'reason' => $e->getMessage(),
                ];
            }
        }

        // Despachar evento para notificar por correo
        VouchersCreated::dispatch($successfulVouchers, $failedVouchers, $this->user);
    }
}
```

2.2. Modificación del Controlador StoreVouchersHandler para Despachar Jobs en Segundo Plano

Se realizaron modificaciones en el controlador StoreVouchersHandler para que, en lugar de procesar los comprobantes directamente, se despache un Job que maneje la carga de comprobantes en segundo plano. Esta modificación mejora la eficiencia y la experiencia del usuario al no requerir que esperen a que se complete el procesamiento de los comprobantes, permitiendo que este se ejecute de manera asíncrona.

Descripción del Código:

- Recepción de Archivos: El controlador recibe los archivos XML subidos por el usuario. Si solo se sube un archivo, se convierte en un array para un tratamiento uniforme.
- Lectura del Contenido XML: Se extrae el contenido de cada archivo XML y se almacena en un array $xmlContents.
- Autenticación del Usuario: Se obtiene al usuario autenticado que está realizando la carga de los comprobantes.
- Procesamiento de Comprobantes: En lugar de procesar los comprobantes directamente en el controlador, se despacha el Job ProcessVouchersJob, que ahora maneja el almacenamiento y notificación en segundo plano.
- Respuesta Exitosa: Si el Job se despacha correctamente, se devuelve una respuesta con el listado de comprobantes almacenados.
- Manejo de Excepciones: Si ocurre un error en el proceso, se captura la excepción y se devuelve un mensaje de error con un código de respuesta 400.

```php
class VouchersCreatedMail extends Mailable
{
    use Queueable, SerializesModels;

    public array $successfulVouchers;
    public array $failedVouchers;
    public User $user;

    public function __construct(array $successfulVouchers, array $failedVouchers, User $user)
    {
        $this->successfulVouchers = $successfulVouchers;
        $this->failedVouchers = $failedVouchers;
        $this->user = $user;
    }

    public function build(): self
    {
        return $this->view('emails.comprobante')
            ->with([
                'successfulVouchers' => $this->successfulVouchers,
                'failedVouchers' => $this->failedVouchers,
                'user' => $this->user,
            ]);
    }
}
```

2.3. Modificación de la Clase VouchersCreatedMail para Incluir Resultados de Comprobantes Exitosos y Fallidos

Se realizaron actualizaciones en la clase VouchersCreatedMail para permitir que maneje tanto los comprobantes que se subieron exitosamente como aquellos que no pudieron

registrarse. Esta modificación asegura que el usuario reciba una notificación detallada con la lista de comprobantes procesados y cualquier error ocurrido durante el proceso, mejorando así la transparencia y la comunicación con el usuario.

Descripción del Código:

Declaración de Propiedades:

- Se agregaron las propiedades successfulVouchers y failedVouchers para almacenar los comprobantes que se procesaron correctamente y aquellos que fallaron, respectivamente.

Constructor:

- El constructor recibe tres parámetros: un array de comprobantes exitosos, un array de comprobantes fallidos y el usuario que realizó la operación. Estos se asignan a las propiedades de la clase.

Método build:

- El método build configura la vista del correo electrónico utilizando la plantilla emails.comprobante.
- Se pasan los arrays de comprobantes exitosos y fallidos, así como la información del usuario, para que se incluyan en el contenido del correo electrónico.

```php
class VouchersCreatedMail extends Mailable
{
    use Queueable, SerializesModels;

    public array $successfulVouchers;
    public array $failedVouchers;
    public User $user;

    public function __construct(array $successfulVouchers, array $failedVouchers, User $user)
    {
        $this->successfulVouchers = $successfulVouchers;
        $this->failedVouchers = $failedVouchers;
        $this->user = $user;
    }

    public function build(): self
    {
        return $this->view('emails.comprobante')
            ->with([
                'successfulVouchers' => $this->successfulVouchers,
                'failedVouchers' => $this->failedVouchers,
                'user' => $this->user,
            ]);
    }
}
```

2.4. Modificación de la Vista del Correo para Mostrar Comprobantes Exitosos y Fallidos

Se ajustó la vista emails.comprobante para incluir una lista detallada de los comprobantes procesados, separando aquellos que se subieron correctamente de los que fallaron

Explicación del Código:

Sección de Comprobantes Subidos Correctamente:

Si hay comprobantes exitosos, se despliegan en una lista. Cada comprobante incluye información clave, como:

- Nombre y tipo de documento del emisor.
- Número de documento del emisor.
- Nombre y tipo de documento del receptor.
- Número de documento del receptor.
- Monto total y moneda.
- Identificador de la factura (serie y número).
- Tipo de comprobante.

Si no hubo comprobantes exitosos, se muestra un mensaje indicando que no se subieron comprobantes correctamente.

Sección de Comprobantes Fallidos:

- Si hay comprobantes fallidos, se listan con detalles del contenido XML y el motivo del fallo.
- Si no hubo fallos, se muestra un mensaje indicando que no hubo comprobantes fallidos.

```
<body>
    <h1>Estimado {{ $user->name }},</h1>
    <p>A continuación, se detallan los comprobantes procesados:</p>

    <h2>Comprobantes Subidos Correctamente</h2>
    @if(count($successfulVouchers) > 0)
        <ul>
            @foreach ($successfulVouchers as $comprobante)
            <li>
                <ul>
                    <li>Nombre del Emisor: {{ $comprobante->issuer_name }}</li>
                    <li>Tipo de Documento del Emisor: {{ $comprobante->issuer_document_type }}</li>
                    <li>Número de Documento del Emisor: {{ $comprobante->issuer_document_number }}</li>
                    <li>Nombre del Receptor: {{ $comprobante->receiver_name }}</li>
                    <li>Tipo de Documento del Receptor: {{ $comprobante->receiver_document_type }}</li>
                    <li>Número de Documento del Receptor: {{ $comprobante->receiver_document_number }}</li>
                    <li>Monto Total: {{ $comprobante->total_amount }} {{ $comprobante->currency_code }}</li>
                    <li>Identificador de Factura: {{ $comprobante->document_serie }}-{{ $comprobante->document_number }}</li>
                    <li>Tipo de Comprobante: {{ $comprobante->document_type_code }}</li>
                    <li>Moneda: {{ $comprobante->currency_code }}</li>
                </ul>
            </li>
            @endforeach
```

```
</ul>
    @else
        <p>No se subieron comprobantes correctamente.</p>
    @endif

    <h2>Comprobantes Fallidos</h2>
    @if(count($failedVouchers) > 0)
        <ul>
            @foreach ($failedVouchers as $comprobante)
            <li>
                Contenido: {{ $comprobante['xml_content'] }}<br>
                Motivo: {{ $comprobante['reason'] }}
            </li>
            @endforeach
        </ul>
    @else
        <p>No hubo comprobantes fallidos.</p>
    @endif

    <p>Gracias por usar nuestro servicio.</p>
</body>
```

2.5. Modificación del Servicio VoucherService para Manejar Comprobantes Exitosos y Fallidos

Se actualizó la función storeVouchersFromXmlContents en la clase VoucherService para manejar tanto los comprobantes que se procesan con éxito como aquellos que fallan. Adicionalmente, se despacha el evento VouchersCreated con los resultados de estos procesos, incluyendo al usuario que realizó la operación.

```
public function getVouchers(int $page, int $paginate): LengthAwarePaginator
    {
        return Voucher::with(['lines', 'user'])->paginate(perPage: $paginate, page: $page);
    }

    /**
     * @param string[] $xmlContents
     * @param User $user
     * @return array
     */
    public function storeVouchersFromXmlContents(array $xmlContents, User $user): array
    {
        $successfulVouchers = [];
        $failedVouchers = [];

        foreach ($xmlContents as $xmlContent) {
            try {
                $voucher = $this->storeVoucherFromXmlContent($xmlContent, $user);
                $successfulVouchers[] = $voucher;
            } catch (Exception $e) {
                Log::error("Error al procesar el XML: " . $e->getMessage());
                $failedVouchers[] = [
                    'xml_content' => $xmlContent,
                    'reason' => $e->getMessage(),
                ];
            }
        }

        // Despachar evento con comprobantes exitosos y fallidos
        VouchersCreated::dispatch($successfulVouchers, $failedVouchers, $user);

        return $successfulVouchers;
    }
```

2.6. Visualización de Correos de Notificación de Comprobantes Procesados

En esta sección se muestran dos capturas de pantalla de correos electrónicos de notificación enviados tras el procesamiento de comprobantes en el sistema. La primera captura evidencia un correo en el que los comprobantes se han subido correctamente, mientras que la segunda muestra un correo en el que algunos comprobantes no pudieron registrarse, indicando la razón de los fallos.

3. Endpoint de montos totales

3.1. Implementación de Endpoint para Montos Totales

Se ha añadido un nuevo endpoint en la ruta de vouchers para proporcionar la información total acumulada en soles y dólares. La nueva ruta GET /vouchers/totals está gestionada por el controlador VoucherTotalsController, específicamente en el método getTotals. Este endpoint permite a los usuarios obtener un resumen consolidado de los montos totales en las dos monedas especificadas, facilitando la visualización de datos financieros acumulados.

```php
Route::prefix('vouchers')->group(
    function () {
        Route::get('/', GetVouchersHandler::class);
        Route::post('/', StoreVouchersHandler::class);
        Route::get('/totals', [VoucherTotalsController::class, 'getTotals']);
    }
);
```

3.2. Creación del Controlador para el Endpoint de Montos Totales

Se ha desarrollado el controlador VoucherTotalsController para manejar el nuevo endpoint GET /vouchers/totals.

Descripción del Código:

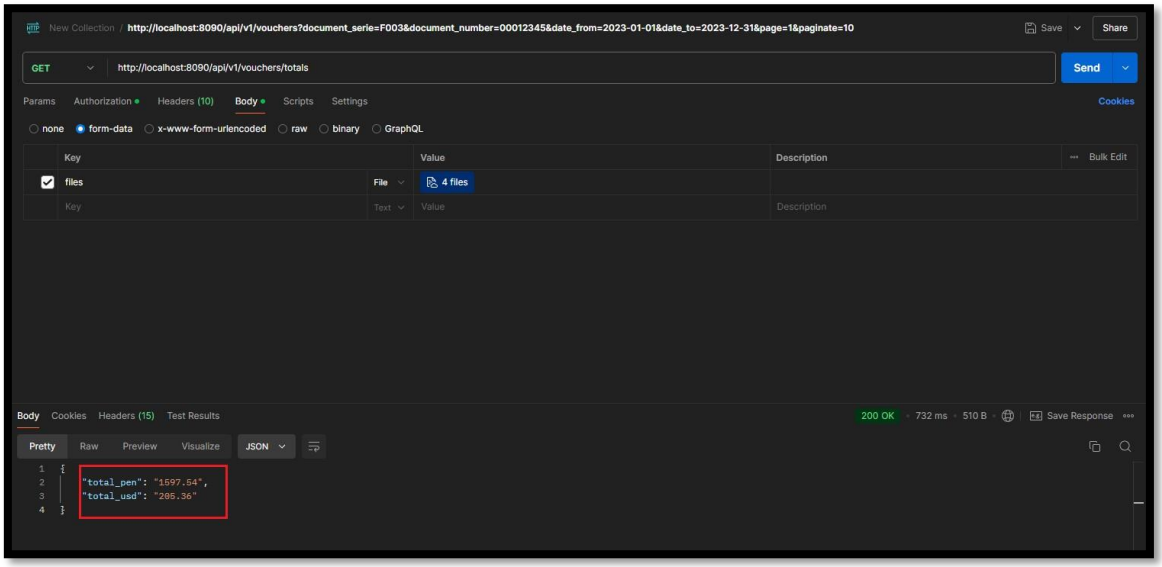
- Se agregó la línea $user = auth()->user(); para obtener el usuario autenticado.
- Cálculo del Total en Soles (PEN): Utiliza una consulta para sumar los montos acumulados en la moneda "PEN" en la tabla vouchers.
- Cálculo del Total en Dólares (USD): Similarmente, realiza una consulta para sumar los montos acumulados en la moneda "USD".
- Se añadió un bloque try-catch para capturar y registrar cualquier excepción que pueda ocurrir durante las consultas a la base de datos.

Ambos totales se devuelven en una respuesta JSON con las claves total_pen y total_usd, proporcionando una visión clara y consolidada de los montos en las dos monedas.

```php
public function getTotals(Request $request): JsonResponse
    {
        try {
            $user = auth()->user();

            $totalPEN = Voucher::where('user_id', $user->id)
                ->where('currency_code', 'PEN')
                ->sum('total_amount');

            $totalUSD = Voucher::where('user_id', $user->id)
                ->where('currency_code', 'USD')
                ->sum('total_amount');

            return response()->json([
                'total_pen' => $totalPEN,
                'total_usd' => $totalUSD,
            ]);

        } catch (Exception $e) {
            Log::error("Error al calcular los totales: " . $e->getMessage());
            return response()->json([
                'error' => 'Error al calcular los totales.'
            ], 500);
        }
    }
```

3.3. Pruebas del Endpoint de Totales de Vouchers en Postman

Se realizaron pruebas en Postman para verificar el correcto funcionamiento del nuevo endpoint /api/v1/vouchers/totals, que calcula y devuelve el total acumulado de montos en soles y dólares.



4. Eliminación de comprobantes

4.1. Implementación de Endpoint para Eliminación de Comprobantes

Se ha implementado una nueva funcionalidad para permitir la eliminación de comprobantes a través de un nuevo endpoint en la API. Esta funcionalidad permite eliminar un comprobante específico mediante su identificador único (id).

```php
Route::prefix('vouchers')->group(
    function () {
        Route::get('/', GetVouchersHandler::class);
        Route::post('/', StoreVouchersHandler::class);
        Route::get('/totals', [VoucherTotalsController::class, 'getTotals']);
        Route::delete('/{id}', DeleteVoucherHandler::class);
    }
);
```

4.2. Implementación de Eliminación de Comprobantes por ID en el Controlador DeleteVoucherHandler

Se ha añadido la lógica para eliminar comprobantes específicos por su identificador (id) en el controlador DeleteVoucherHandler. Esta funcionalidad permite realizar operaciones de eliminación de manera eficiente y segura, y se ha implementado siguiendo estándares de código limpio y buenas prácticas.

Explicación del Código

Verificación del usuario autenticado:

- Se agregó la línea $user = auth()->user(); para obtener el usuario actualmente autenticado.

Búsqueda del Comprobante:

- Utiliza Voucher::find($id) para localizar el comprobante.
- Si no se encuentra, devuelve un error 404 (No Encontrado).

Eliminación del Comprobante:

- Si el comprobante existe, se elimina con $voucher->delete().
- Se recomienda agregar manejo de excepciones para posibles fallos en la eliminación.

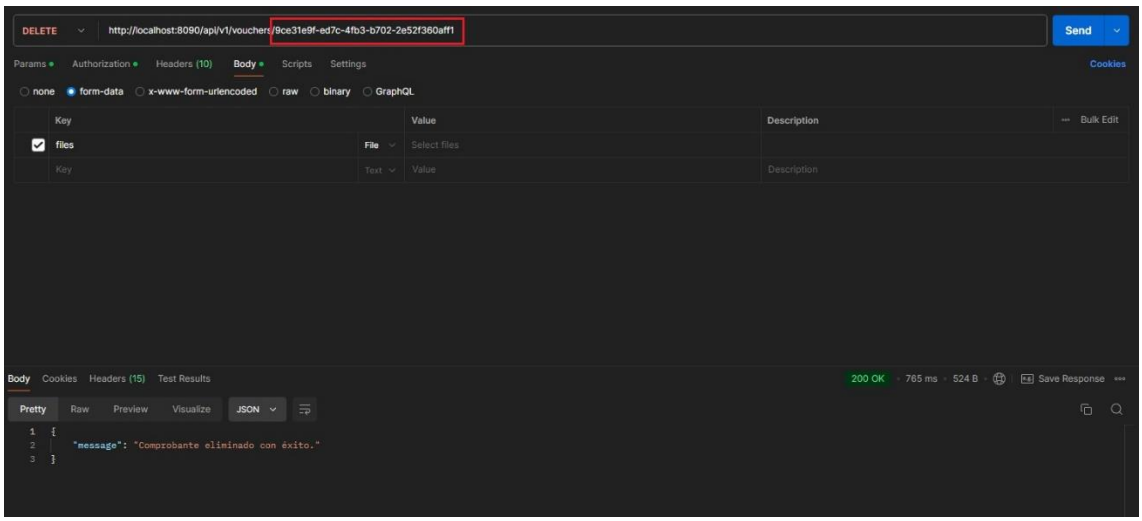
Respuesta de Éxito:

- Retorna un mensaje de éxito con código 200 (Éxito).

```php
public function __invoke($id): JsonResponse
    {
        $user = auth()->user();
        $voucher = Voucher::where('id', $id)->where('user_id', $user->id)->first();

        if (!$voucher) {
            return response()->json([
                'message' => 'Comprobante no encontrado o no pertenece al usuario.',
            ], 404);
        }

        $voucher->delete();

        return response()->json([
            'message' => 'Comprobante eliminado con éxito.',
        ], 200);
    }
```

4.3. Pruebas de Eliminación de Comprobantes en Postman

Se realizaron pruebas en Postman para verificar el funcionamiento del nuevo endpoint que permite la eliminación de comprobantes por ID. Se configuró una petición DELETE a la URL http://localhost:8090/api/v1/vouchers/9ce31e9f-ed7c-4fb3-b702-2e52f360aff1. Las pruebas confirmaron que la funcionalidad de eliminación se comporta según lo esperado, manejando adecuadamente tanto los casos de éxito como los errores.



5. Filtro en listado de comprobantes

5.1. Implementación de Filtros en Listado de Comprobantes por Serie, Número y Rango de Fechas

Se ha añadido la capacidad de filtrar el listado de comprobantes por serie, número y un rango de fechas en el endpoint de listado de comprobantes. Esto se hizo mediante la modificación del controlador GetVouchersHandler.php.

Explicación del Código

- Obtener el Usuario: Se obtiene el usuario autenticado mediante auth()->user(), lo que permite aplicar filtros solo a los comprobantes del usuario actual.
- Definición de Filtros: Se define un array $filters para mapear los parámetros de la solicitud a las columnas de la base de datos.
- Construcción de Consulta: Se construye una consulta base para los comprobantes del usuario. Luego, se aplican los filtros si están presentes en la solicitud.
- Ejecución y Paginación: Se ejecuta la consulta paginada, devolviendo un mensaje si no se encuentran resultados y los datos en formato JSON si hay resultados.
- Respuestas: Se devuelven mensajes claros dependiendo del resultado de la consulta, asegurando una respuesta adecuada según si se encuentran o no los comprobantes.

```php
public function __invoke(GetVouchersRequest $request): Response
    {
        $user = auth()->user();

        // Array de filtros con sus respectivos campos
        $filters = [
            'document_serie' => 'document_serie',
            'document_number' => 'document_number',
            'date_from' => ['created_at', '>='],
            'date_to' => ['created_at', '<='],
        ];

        // Construcción de la consulta con filtros
        $query = Voucher::where('user_id', $user->id);

        foreach ($filters as $requestKey => $dbColumn) {
            if ($request->filled($requestKey)) {
                $filterValue = $request->input($requestKey);
                is_array($dbColumn)
                    ? $query->whereDate($dbColumn[0], $dbColumn[1], $filterValue)
                    : $query->where($dbColumn, $filterValue);
            }
        }

```

```php
$vouchers = $query->paginate(
            perPage: $request->query('paginate'),
            page: $request->query('page')
        );

        if ($vouchers->isEmpty()) {
            return response([
                'message' => 'No se encontraron comprobantes con los filtros aplicados.',
            ], 404);
        }

        return response([
            'data' => VoucherResource::collection($vouchers),
        ], 200);
    }
```

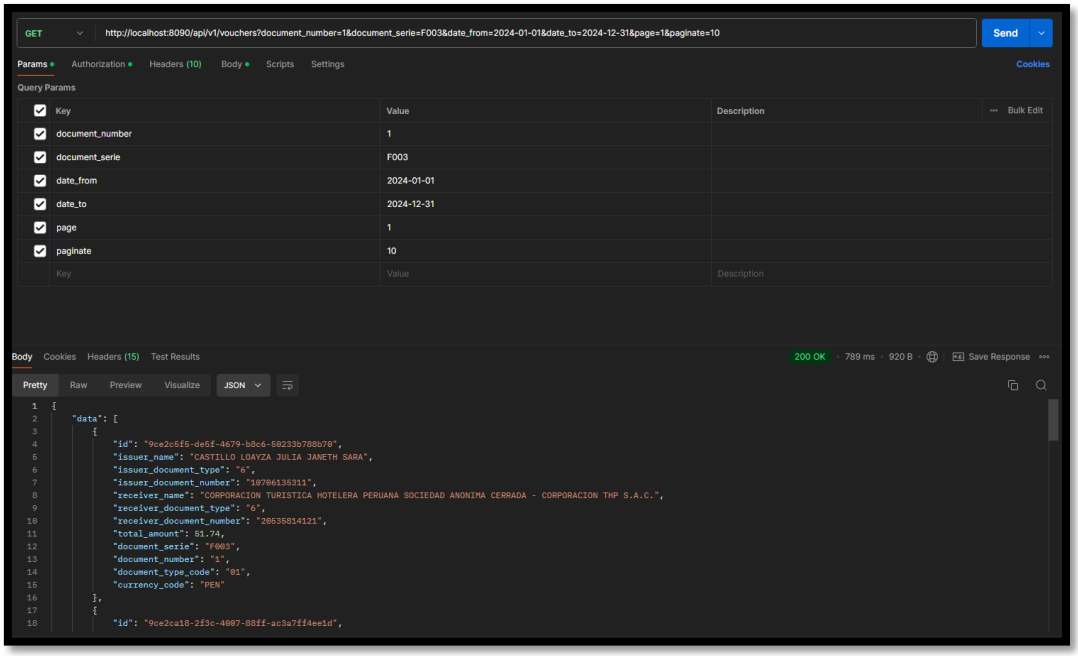
5.2. Implementación de Filtros en el Endpoint de Listado de Comprobantes

Se ha ampliado la funcionalidad del endpoint de listado de comprobantes para incluir nuevos filtros que permiten una búsqueda más detallada. Estos filtros incluyen la serie y número del documento, así como un rango de fechas basado en las fechas de creación.

```php
public function rules(): array
    {
        return [
            'page' => ['required', 'int', 'gt:0'],
            'paginate' => ['required', 'int', 'gt:0'],

            // NUEVOS FILTROS
            'document_serie' => ['nullable', 'string', 'max:4'],
            'document_number' => ['nullable', 'string', 'max:8'],
            'date_from' => ['nullable', 'date'],
            'date_to' => ['nullable', 'date'],
        ];
    }
```

5.3. Pruebas de Filtros en el Endpoint de Listado de Comprobantes en Postman

Se realizaron pruebas para verificar el funcionamiento de los nuevos filtros implementados en el endpoint de listado de comprobantes. La URL utilizada para las pruebas fue http://localhost:8090/api/v1/vouchers, donde se aplicaron diferentes combinaciones de filtros para asegurar que el sistema devuelve los resultados esperados según los parámetros especificados.


GET
http://localhost:8090/api/v1/vouchers?document_number=1&document_serie=F003&date_from=2024-01-01&date_to=2024-12-31&page=1&paginate=10
Send
Params
Authorization
Headers (10)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
document_number
1
document_serie
F003
date_from
2024-01-01
date_to
2024-12-31
page
1
paginate
10
Body
Cookies
Headers (15)
Test Results
200 OK
789 ms
920 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"data": [
{
"id": "9ce2c5f5-de5f-4679-b8c6-50233b788b70",
"issuer_name": "CASTILLO LOAYZA JULIA JANETH SARA",
"issuer_document_type": "6",
"issuer_document_number": "10706135311",
"receiver_name": "CORPORACION TURISTICA HOTELERA PERUANA SOCIEDAD ANONIMA CERRADA - CORPORACION THP S.A.C.",
"receiver_document_type": "6",
"receiver_document_number": "20535814121",
"total_amount": 51.74,
"document_serie": "F003",
"document_number": "1",
"document_type_code": "01",
"currency_code": "PEN"
},
{
"id": "9ce2ca18-2f3c-4007-88ff-ac3a7ff4ee1d",