

# WorkFlow CRM Documentos Rodrigo

- Acceso al panel CRM Documentos Rodrigo..... 2**
- Funcionamiento del panel..... 3**
  - Inicio..... 3
  - Estadísticas..... 7
- Backend:..... 7**
  - Función index:..... 8
  - Función show..... 9
  - Función stats..... 9
  - Función store..... 10
  - Función destroy..... 12
  - Función approved..... 13

## Acceso al panel CRM Documentos Rodrigo

Para acceder a este documento tendremos tres opciones una vez hechos los seed, deberemos iniciar sesión con uno de estos tres usuarios:

usuario → admin@icloud.com

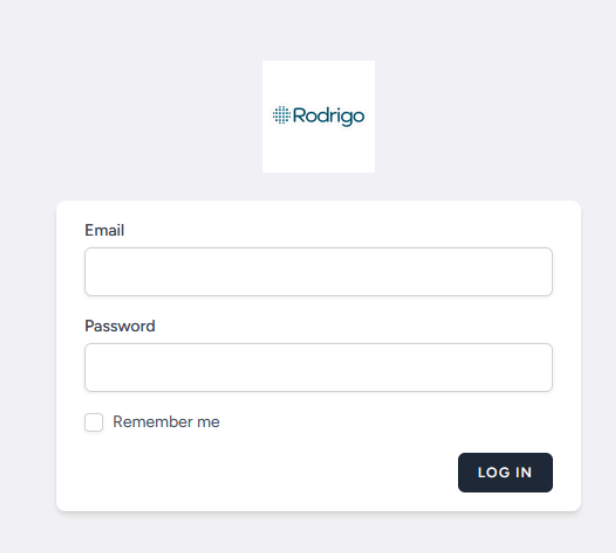
contraseña → admin

usuario → manager@icloud.com

contraseña → manager

usuario → assigned@icloud.com

contraseña → assigned



The image shows a login interface for 'Rodrigo'. At the top center is the 'Rodrigo' logo, which consists of a blue grid icon followed by the word 'Rodrigo' in a blue sans-serif font. Below the logo is a white rectangular form with rounded corners. Inside the form, there are two input fields: the first is labeled 'Email' and the second is labeled 'Password'. Below the password field is a checkbox labeled 'Remember me'. In the bottom right corner of the form is a dark blue button with the text 'LOG IN' in white capital letters.

# Funcionamiento del panel

## Inicio














Una vez iniciemos sesión tendremos acceso al listado de documentos:

Rodrigo Inicio Gráficos Admin ▾

Inicio

Crear Documento Filtros

Documentos:

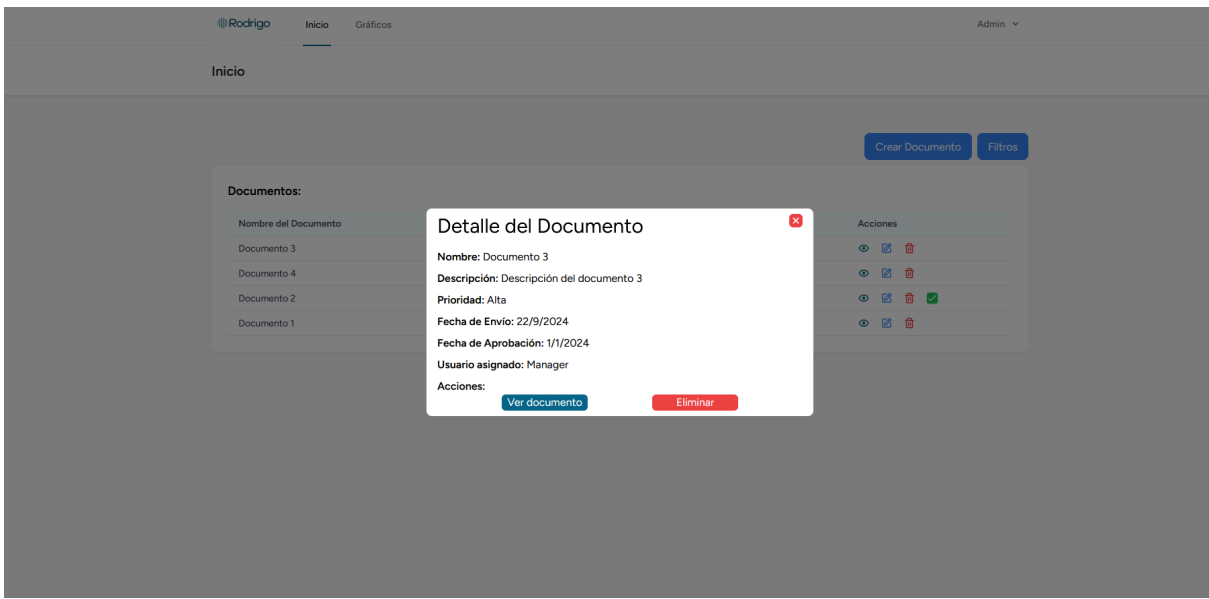
Nombre del Documento	Prioridad	Fecha de Envío	Fecha de Aprobación	Acciones
Documento 3	Alta	22/9/2024	1/1/2024	  
Documento 4	Alta	22/9/2024	1/5/2024	  
Documento 2	Media	22/9/2024	No aprobada	   
Documento 1	Baja	22/9/2024	22/9/2024	  

A la derecha encontraremos los botones de interacción rápida

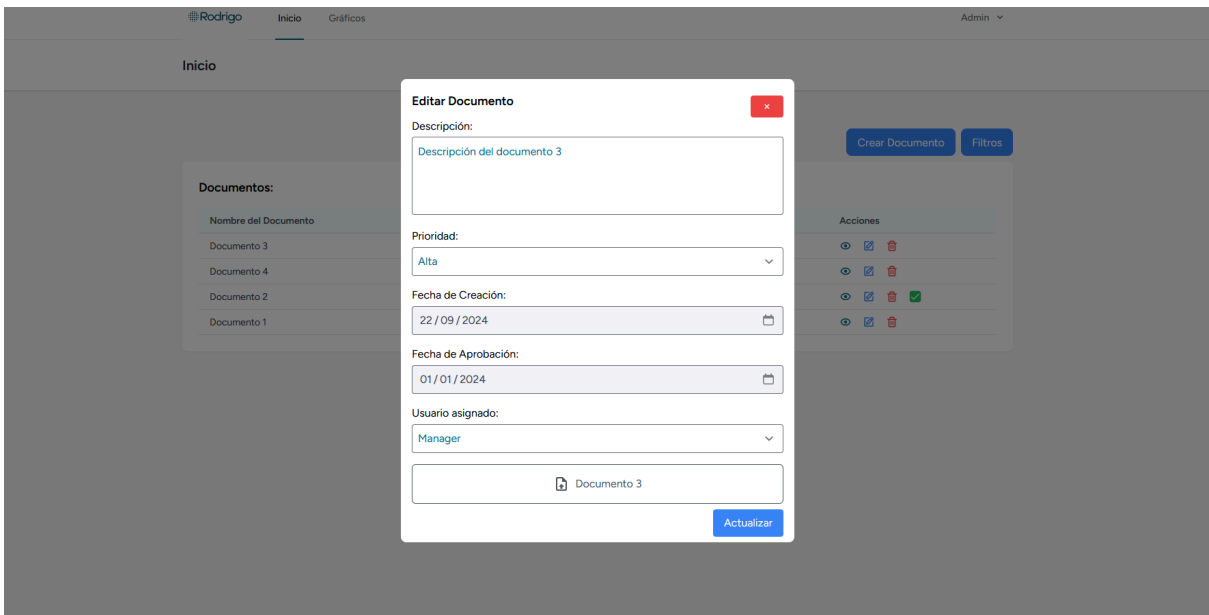


Cada uno de estos tendrá diferentes interacciones

Ver detalles
















Editar



Eliminar

Documentos:

Nombre del Documento	Prioridad	Fecha de Envío	Fecha de Aprobación	Acciones
Documento 3	Alta	22/9/2024	1/1/2024	  
Documento 4	Alta	22/9/2024	1/5/2024	  
Documento 2				   
Documento 1				  

¿Estás seguro de que quieres eliminar este documento?

No

Sí














Aprobar

RodrigoInicioGráficosAdmin

Inicio

Crear DocumentoFiltros

Documentos:

Nombre del Documento	Prioridad	Fecha de Envío	Fecha de Aprobación	Acciones
Documento 3	Alta	22/9/2024	1/1/2024	  
Documento 4	Alta	22/9/2024	1/5/2024	  
Documento 2				   
Documento 1				  

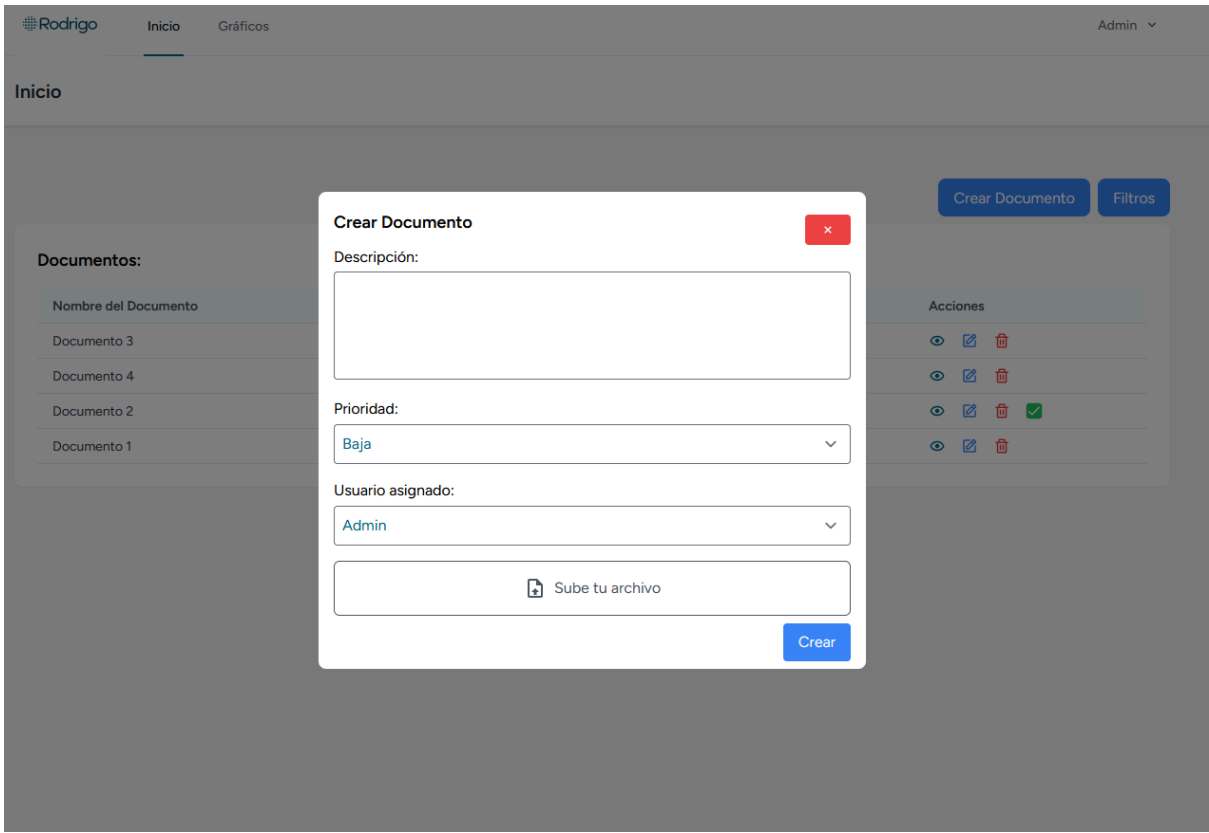
¿Estás seguro de que quieres aprobar este documento?

No

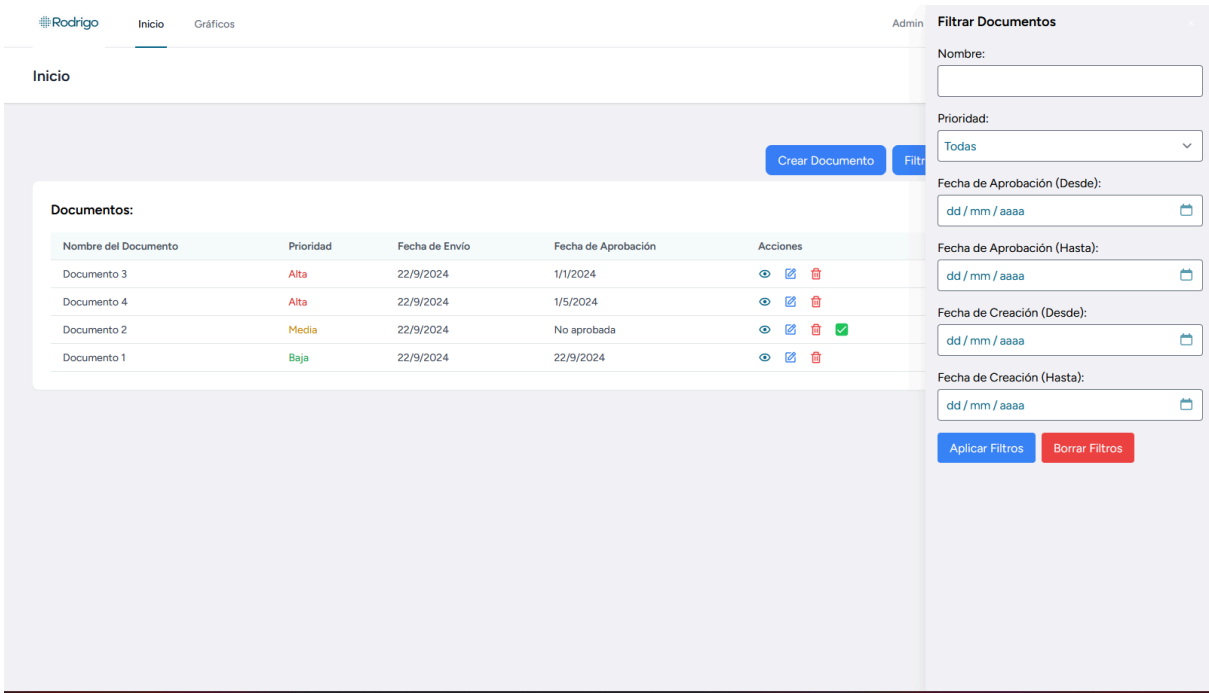
Sí

Además, podremos crear un documento o filtrar estos:

Crear Documentos:

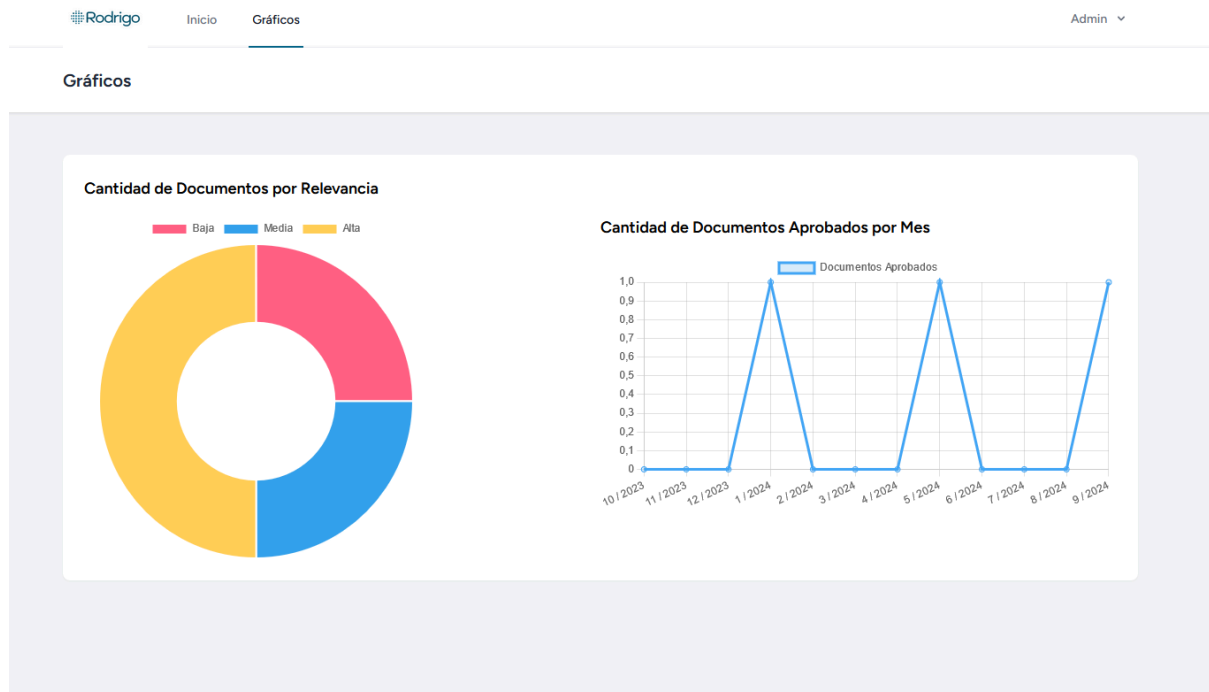


Filtros de documentos



## Estadísticas

Y por último podremos ver las estadísticas de los documentos en el apartado de gráficos:



## Backend:

En la parte del backend encontramos las validaciones para comprobar el rango de usuarios para accesos a rutas en el **app\Http\Middleware\UserPermission.php** este no dejará pasar a los usuarios que no tengan suficiente rango.

Además encontramos las diversas funciones de interacción con documentos en el **app\Http\Controllers\DocumentController.php** y también con los usuarios en el **app\Http\Controllers\UserController.php**.

## Función index:

Será la encargada de devolver los documentos agrupados por prioridad y comprobar que solo los documentos de ese usuario son los que este pueda ver.

```
1 public function index()
2 {
3     $user = Auth::user();
4
5     $documents = Document::where('status', 1)
6         ->get(['id', 'name', 'date_submitted', 'date_approved', 'priority', 'user_id'])
7         ->map(function ($document) {
8             $document->detail = route('documents.show', $document->id);
9             return $document;
10        })
11        ->groupBy(function ($document) {
12            return $document->priority;
13        });
14
15    $groupedDocuments = $documents->flatMap(function ($group) {
16        return $group;
17    });
18
19    if ($user->permissions === 0) {
20        Log::info('Todos los documentos:', $groupedDocuments->toArray());
21        $groupedDocuments = $groupedDocuments->filter(function($document) use ($user) {
22            return $document->user_id === $user->id;
23        });
24    }
25
26    $groupedDocuments = $groupedDocuments->map(function($document) {
27        unset($document->user_id);
28        return $document;
29    });
30
31
32    return response()->json($groupedDocuments);
33 }
```



## Función show

Será la encargada de devolver el detalle del documento reemplazando el id por el nombre de usuario para evitar filtrado de datos al front desde el backend

```
1 public function show(String $id)
2     {
3         $document = Document::with('user')->findOrFail($id);
4
5         $document->username = $document->user->name;
6         unset($document->user_id);
7
8         return response()->json($document);
9     }
```

## Función stats

Se encarga de renderizar la vista de gráficos devolviendo los datos necesarios para esta vista.

```
public function stats(){

    // Get all documents active

    $documents = Document::where('status', 1)->get();

    return Inertia::render('Graficos', [

        'documents' => $documents

    ]);

}
```

## Función store

Encargada de guardar los documentos y además crear el documento en la base de datos. En caso de error esta devuelve los errores a la vista traducidos al español para ser mostrados.

```
1 public function store(Request $request)
2 {
3     try{
4         $validatedData = $request->validate([
5             'name' => 'required|string|max:50',
6             'description' => 'required|string|max:255',
7             'priority' => 'required|integer|min:1|max:3',
8             'document' => 'required|file|mimes:pdf,doc,docx',
9             'user_id' => 'required|integer|exists:users,id',
10        ]);
11
12        $filePath = $request->file('document')->store('documents');
13
14
15
16        $document = Document::create([
17            'name' => $validatedData['name'],
18            'description' => $validatedData['description'],
19            'priority' => (int)$validatedData['priority'],
20            'date_approved' => null,
21            'date_submitted' => now(),
22            'url' => $filePath,
23            'user_id' => (int)$validatedData['user_id'],
24        ]);
25
26        return response()->json($document, 201);
27    }catch (ValidationException $e){
28        return response()->json([
29            'errors' => $e->errors()
30        ], 422);
31    }catch (Exception $e){
32        return response()->json($e->getMessage(), 500);
33    }
34 }
```

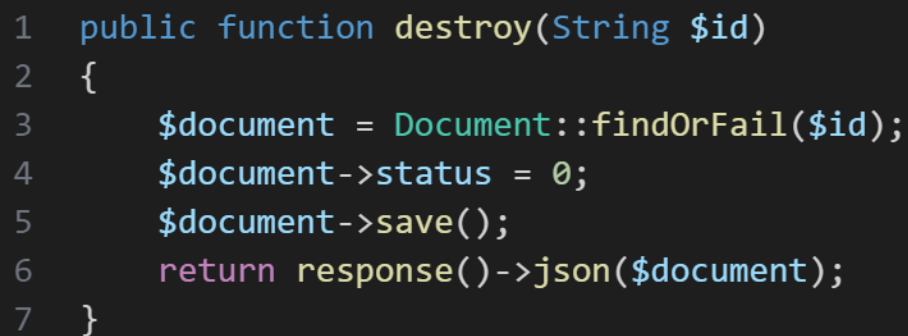
## Función update

Se encarga de actualizar los datos del registro

```
1 public function update(Request $request, String $id)
2 {
3     try{
4         $validatedData = $request->validate([
5             'name' => 'string|max:50|min:1',
6             'description' => 'string|max:255|min:1',
7             'priority' => 'integer|min:1|max:3',
8             'document' => 'nullable|file|mimes:pdf,doc,docx',
9             'user_id' => 'required|integer|exists:users,id',
10            'date_submitted' => 'required|date',
11            'date_approved' => 'nullable|date',
12        ]);
13
14
15
16        $document = Document::findOrFail($id);
17
18
19        if($request->hasFile('document')){
20            $filePath = $request->file('document')->store('documents');
21            $document->url = $filePath;
22        }
23
24        $document->name = $validatedData['name'];
25        $document->description = $validatedData['description'];
26        $document->priority = (int)$validatedData['priority'];
27        $document->user_id = (int)$validatedData['user_id'];
28        $document->date_submitted = $validatedData['date_submitted'];
29        $document->date_approved = $validatedData['date_approved'];
30        $document->save();
31
32        return response()->json($document);
33    }catch (ValidationException $e){
34        return response()->json([
35            'errors' => $e->errors()
36        ], 422);
37    }
38 }
```

## Función destroy

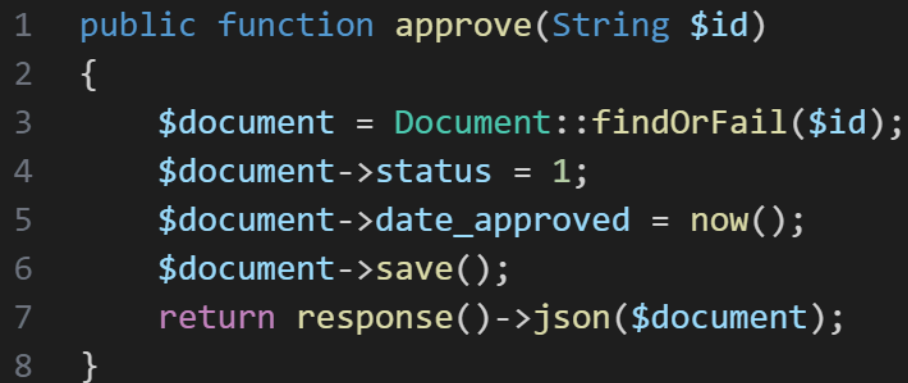
Es la encargada de hacer el borrado lógico de los documentos, es decir, que por seguridad las funciones de borrado serán lógicas en la base de datos. Pasando de estado 1 a 0.



```
1 public function destroy(String $id)
2 {
3     $document = Document::findOrFail($id);
4     $document->status = 0;
5     $document->save();
6     return response()->json($document);
7 }
```

## Función approved

Encargada de actualizar el documento sólo el status comprobando que sea 1 y actualizando la fecha de aprobación



```
1 public function approve(String $id)
2 {
3     $document = Document::findOrFail($id);
4     $document->status = 1;
5     $document->date_approved = now();
6     $document->save();
7     return response()->json($document);
8 }
```