

“Sistema de Gestión de Expedientes MP – Manual Técnico Detallado”

Subtítulo:

Documentación exhaustiva para desarrolladores, administradores de sistemas e ingenieros de soporte sobre la arquitectura, implementación y mantenimiento del Sistema de Gestión de Expedientes del Ministerio Público (MP).

Elaborado por:

Jorge Noel Garcia Valenzuela

Fecha:

23 de noviembre del 2025

Introducción

El presente documento describe los aspectos técnicos del Sistema de Gestión de Expedientes, una aplicación web desarrollada para optimizar la administración de expedientes de la Dirección de Investigaciones Criminalísticas (DICRI) y los indicios asociados.

La aplicación implementa flujos de trabajo para el registro, la aprobación y el rechazo de expedientes, junto con una gestión detallada de los indicios por parte de personal técnico y coordinadores. El sistema incorpora un robusto control de acceso y seguridad basado en roles.

Objetivos del Sistema:

- Administrar el ciclo de vida de los expedientes, incluyendo la gestión de sus estados (Pendiente, Aprobado, Rechazado).
- Facilitar el registro y la vinculación de indicios específicos a cada expediente.
- Asegurar la integridad y el control de las operaciones mediante un esquema de permisos basado en roles de usuario (Técnico / Coordinador).
- Proporcionar herramientas de reporte y análisis estadístico, con capacidad de filtrado por estado y rango de fechas.

Tecnologías utilizadas

Frontend: React + Vite, TailwindCSS

Estado / contexto: React Context API ([AuthContext](#))


Backend: Node.js + Express

Base de datos: SQL Server

Conexión a BD: [mssql](#)

Reportes: [chart.js](#) / [react-chartjs-2](#), [jsPDF](#) + [jspdf-autotable](#), [xlsx](#)

Control de versiones: Git + GitHub

 22jgarcia chore: inicial proyecto strutura para backend/frontend folders y giti... 38e8389 · 2 days ago 🕒 2 Commits		
📁 backendMP	chore: initial project setup	2 days ago
📁 frontend	chore: initial project setup	2 days ago
📁 scripts	chore: inicial proyecto strutura para backend/frontend ...	2 days ago
📄 .gitignore	chore: inicial proyecto strutura para backend/frontend ...	2 days ago

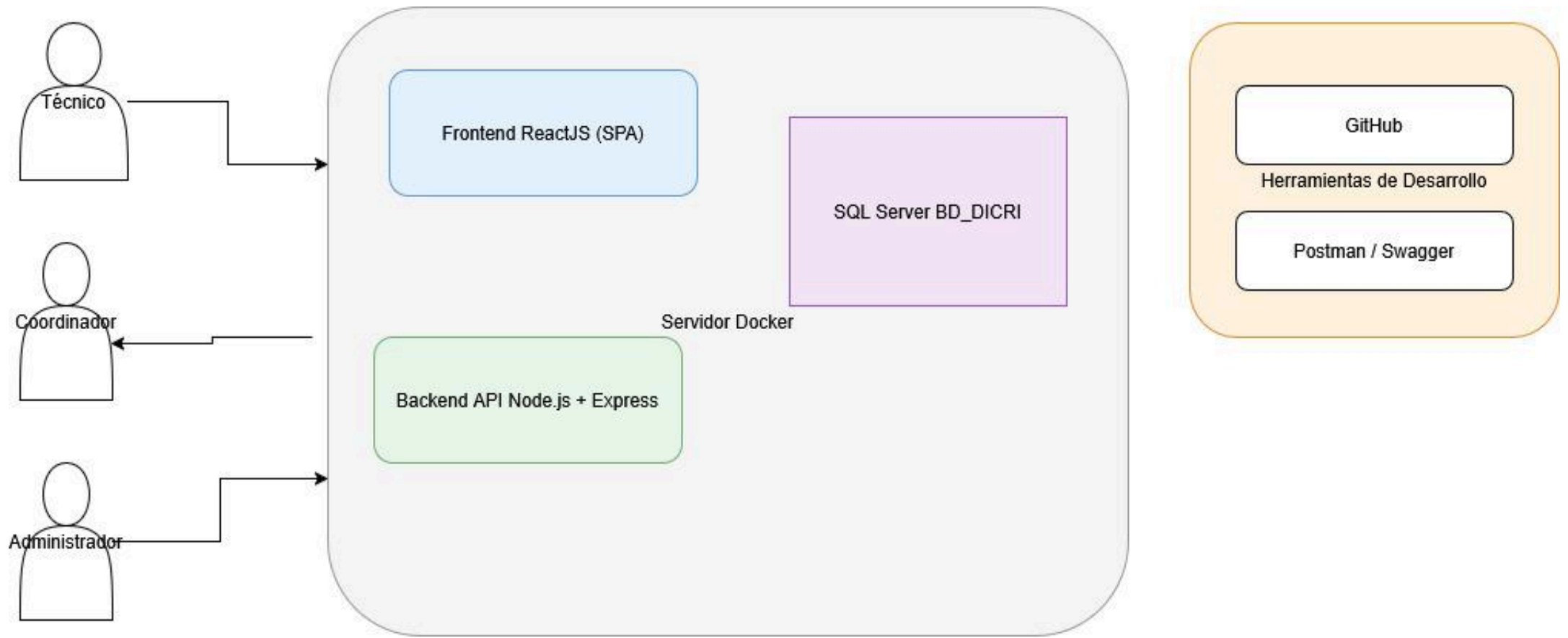
Arquitectura general del sistema

El Cliente React se comunica con la API REST (<http://localhost:3000>), la cual utiliza procedimientos almacenados en SQL Server para las operaciones CRUD.

Endpoints expuestos:

- [/login](#)
- [/expedientes](#), [/expedientes/:id](#), [/expedientes/:id/indicios](#)
- [/expedientes/:id/aprobar](#), [/expedientes/:id/rechazar](#)
- [/indicios](#)
- [/reportes](#), [/reportes/filtro](#), [/reportes/detallado](#)

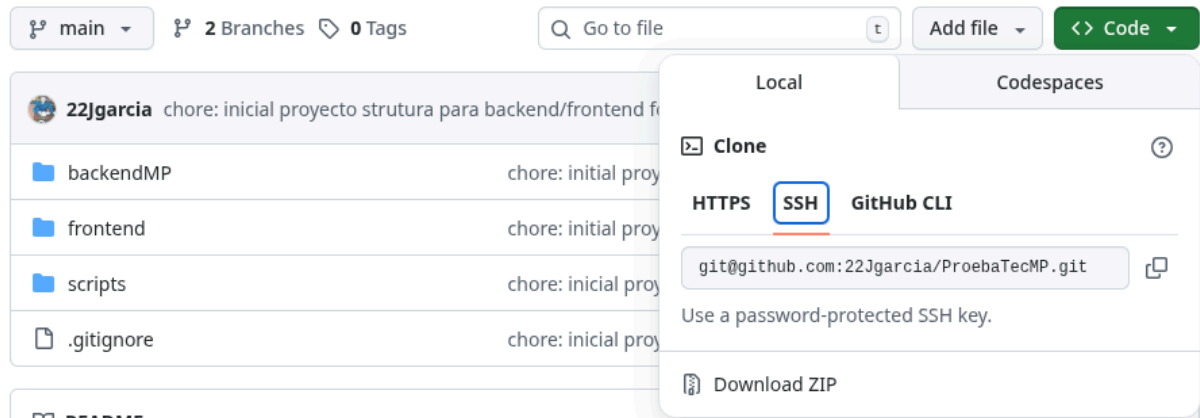
La API usa procedimientos almacenados en SQL Server para las operaciones CRUD.



Instalación y ejecución

Backend

1. Clonar repo:



2. Instalar dependencias:

`npm install`

3. Configurar conexión a SQL Server en `config/db.js`:

servidor, usuario, contraseña, base de datos.

4. Ejecutar servidor:

`npm run dev`

5. El backend queda escuchando en <http://localhost:3000>.

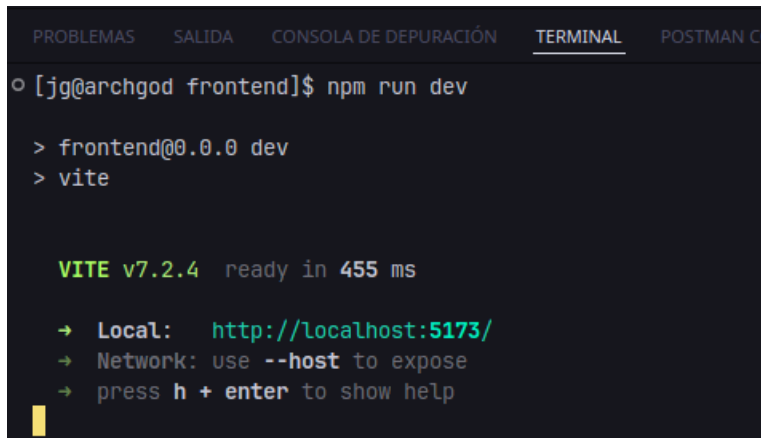
```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  POST
○ [jg@archgod backendMP]$ npm run dev

> backendmp@1.0.0 dev
> nodemon server.js

[nodemon] 3.1.11
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Ejecutandose en http://localhost:3000
```

Frontend

1. Clonar repo o ir a carpeta:
cd frontendMP
npm install
npm run dev
2. La app se sirve en <http://localhost:5173>.



```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  POSTMAN CO  
[jg@archgod frontend]$ npm run dev  
  
> frontend@0.0.0 dev  
> vite  
  
VITE v7.2.4 ready in 455 ms  
  
→ Local:   http://localhost:5173/  
→ Network: use --host to expose  
→ press h + enter to show help
```

Modelo de datos y procedimientos almacenados

Tablas principales

Usuarios

- id, nombre, rol

Expedientes

- id, numero, descripcion, estado,
- fecha_registro,
- justificacion_rechazo,
- usuario_creacion, usuario_aprobacion, usuario_rechazo,
- fecha_aprobacion

Indicios

- id, expediente_id (FK),
- descripcion, cantidad,
- color, tamano, peso, ubicacion,
- tecnico



Procedimientos almacenados

SP_RegistrarExpediente

SP_ListarExpedientes

SP_ObtenerExpediente

SP_AprobarExpediente

SP_RechazarExpediente

SP_ReporteExpedientesPorEstado

SP_ReporteExpedientesDetallado

SP_RegistrarIndicio

Backend

Rutas de usuarios

Archivo: [routes/usuarios.routes.js](#)

- [POST /usuarios/login](#)
Recibe [nombre](#), busca en tabla [Usuarios](#), retorna [{ id, nombre, rol }](#).



```
1 const { Router } = require("express");
2 const controller = require("../controllers/usuarios.controller");
3
4 const router = Router();
5
6 router.get("/", controller.listarUsuarios);
7 router.post("/login", controller.login);
8
9 module.exports = router;
10
```

Rutas de expedientes

Archivo: [routes/expedientes.routes.js](#)

- GET /expedientes
- POST /expedientes (solo coordinador)
- GET /expedientes/:id
- GET /expedientes/:id/indicios
- PUT /expedientes/:id/aprobar

- PUT /expedientes/:id/rechazar

El middleware `requireRole` basado en header `rol` y `usuario`.

```

1 function requireRole(rolesPermitidos = []) {
2   return (req, res, next) => {
3     const rol = req.headers["rol"];
4
5     if (!rol) {
6       return res.status(401).json({ error: "Rol no enviado en headers" });
7     }
8     if (!rolesPermitidos.includes(rol)) {
9       return res.status(403).json({ error: "Acceso denegado: rol no autorizado" });
10    }
11
12    next();
13  };
14 }
15
16 module.exports = requireRole;
17

```

Rutas de reportes

GET /reportes → resumen por estado

GET /reportes/filtro → con filtros `estado`, `fechaInicio`, `fechaFin`

GET /reportes/detallado → listado detallado (para PDF/Excel)

```

1 const { Router } = require("express");
2 const controller = require("../controllers/reportes.controller");
3 const requireRole = require("../middleware/roles");
4
5 const router = Router();
6
7 router.get("/", requireRole(["coordinador"]), controller.listarReportes);
8 router.get("/estados", requireRole(["coordinador"]), controller.reportePorEstado);
9 router.get("/filtro", requireRole(["coordinador"]), controller.reporteFiltro);
10 router.get("/detallado", requireRole(["coordinador"]), controller.reporteDetallado);
11
12 module.exports = router;
13

```

Frontend

Login

Archivo: `Login.jsx`

- Input de nombre.
- Llamada a `/usuarios/login`.
- Guarda `{nombre, rol}` en `localStorage` y en `AuthContext`.
- Navega a `/expedientes`.



```
1 import { useContext, useState } from "react";
2 import { AuthContext } from "../context/AuthContext";
3 import { useNavigate } from "react-router-dom";
4 import logoMP from "../assets/Logo.jpg";
5
6 export default function Login() {
7   const { login } = useContext(AuthContext);
8   const navigate = useNavigate();
9
10  const [nombre, setNombre] = useState("");
11
12  const handleLogin = async () => {
13    const res = await fetch("http://localhost:3000/usuarios/login", {
14      method: "POST",
15      headers: { "Content-Type": "application/json" },
16      body: JSON.stringify({ nombre }),
17    });
18
19    const data = await res.json();
20
21    if (data.error) {
22      alert("Usuario no encontrado");
23      return;
24    }
25
26    // GUARDAR usuario completo
27    login(data.nombre, data.rol);
28
29    navigate("/expedientes");
30  };
31
32  return (
33    <div className="min-h-screen flex items-center justify-center bg-gray-100">
34      <div className="bg-white shadow-lg p-8 rounded-xl w-full max-w-md text-center">
35        <img src={logoMP} alt="MP" className="w-28 mx-auto mb-4 opacity-90" />
36
37        <h1 className="text-3xl font-bold text-gray-800 mb-6">
38          Sistema de Expedientes
39        </h1>
40
41        <input
42          type="text"
43          className="w-full p-3 border rounded-lg mb-5 text-lg"
44          placeholder="Ingresa tu nombre"
45          value={nombre}
46          onChange={(e) => setNombre(e.target.value)}
47        />
48
49        <button
50          onClick={handleLogin}
51          className="w-full py-3 bg-blue-600 text-white rounded-lg text-lg font-semibold hover:bg-blue-700 transition"
52        >
53          Ingresar
54        </button>
55      </div>
56    </div>
57  );
58 }
```

Listado de expedientes

Archivo: [Expedientes.jsx](#)

Muestra cards con:

Número

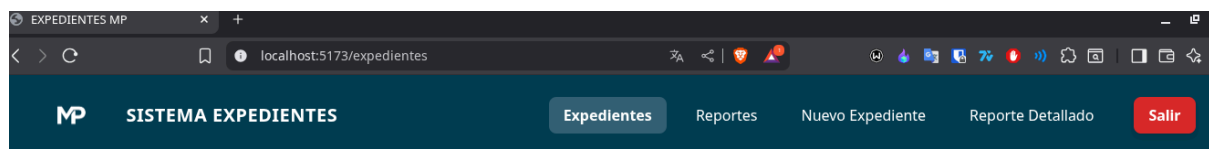
Descripción

Estado con badge de color

“Registrado por ...”

Fecha de registro

Al hacer clic, navega a </expedientes/:id>.



```

1 import { useEffect, useState } from "react";
2 import { Link } from "react-router-dom";
3
4 export default function Expedientes() {
5   const [lista, setLista] = useState([]);
6
7   useEffect(() => {
8     const cargar = async () => {
9       const res = await fetch("http://localhost:3000/expedientes");
10      const data = await res.json();
11      setLista(data);
12    };
13    cargar();
14  }, []);
15
16  const badgeClass = (estado) => {
17    switch (estado.toLowerCase()) {
18      case "pendiente":
19        return "bg-gray-200 text-gray-700 border-gray-300";
20      case "aprobado":
21        return "bg-gray-100 text-green-800 border-green-300";
22      case "rechazado":
23        return "bg-gray-100 text-red-800 border-red-300";
24      default:
25        return "bg-gray-200 text-gray-700 border-gray-300";
26    }
27  };
28
29  return (
30    <div className="p-8 max-w-6xl mx-auto">
31      <h2 className="text-3xl font-bold text-slate-900 tracking-tight mb-8">
32        Gestión de Expedientes
33      </h2>
34
35      {lista.length === 0 && (
36        <p className="text-gray-600 text-sm">No hay expedientes registrados.</p>
37      )}
38
39      <div className="grid sm:grid-cols-2 lg:grid-cols-3 gap-6">
40        {lista.map((exp) => (
41          <Link
42            to={` /expedientes/${exp.id}`}
43            key={exp.id}
44            className="
45              block
46              bg-white
47              border border-slate-200
48              rounded-lg
49              p-6
50              shadow-sm
51              transition-all
52              duration-200
53              hover:shadow-xl
54              hover:-translate-y-1.5
55              hover:border-slate-400
56            "
57          >
58            { /* Header */ }
59            <div className="flex justify-between items-center mb-3">
60              <h3 className="text-xl font-semibold text-slate-900 tracking-tight">
61                EXP-{exp.id.toString().padStart(3, "0")}
62              </h3>
63
64              <span
65                className={`px-2.5 py-1 text-xs font-medium rounded border ${badgeClass(
66                  exp.estado
67                )}`}
68              >
69                {exp.estado.toUpperCase()}
70              </span>
80            </div>
71
72            { /* Descripción */ }
73            <p className="text-sm text-slate-700 mb-6 leading-snug line-clamp-3">
74              {exp.descripcion}
75            </p>
76
77            <p className="mt-1 text-gray-600 text-sm">
78              Registrado por:{" "}
79              <span className="font-semibold">{exp.usuario_creacion}</span>
80            </p>
81
82            { /* Fecha */ }
83            <p className="text-xs text-slate-500 border-t pt-3">
84              Fecha de registro:{" "}
85              <span className="text-slate-700">
86                {new Date(exp.fecha_registro).toLocaleString()}
87              </span>
88            </p>
89          </Link>
90        ))}
91      </div>
92    </div>
93  );
94 }
95
96

```

Detalle del expediente

Archivo: `DetalleExpediente.jsx`

Muestra:

Número, descripción, estado, fecha.

Botones para coordinador:

Aprobar (PUT `/expedientes/:id/aprobar`)

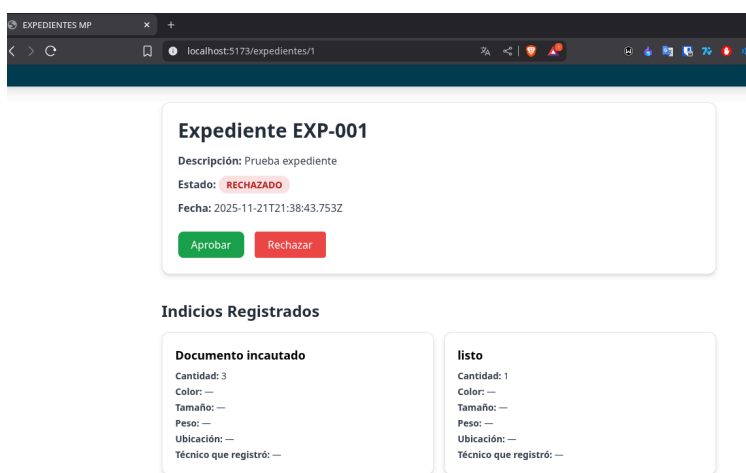
Rechazar con modal y justificación (PUT `/rechazar` con `justificacion`)

Lista de indicios registrados.

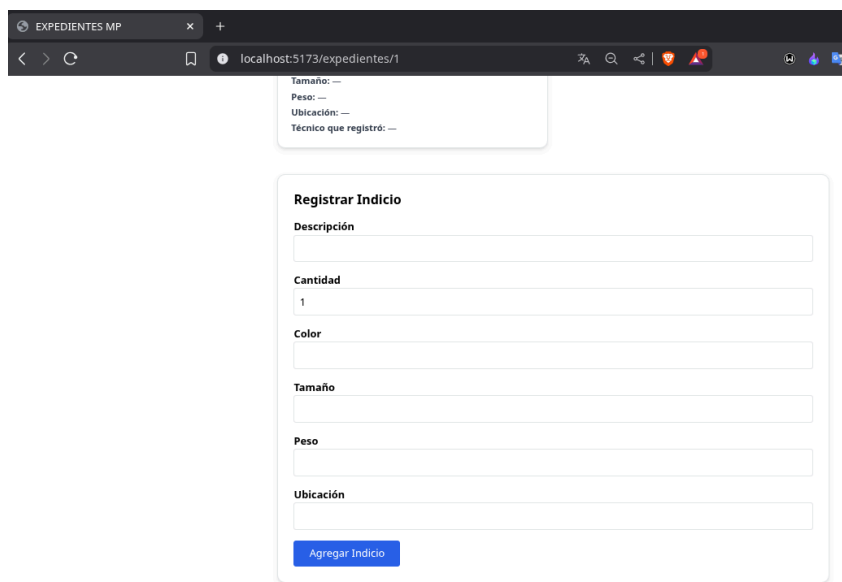
Formulario para **técnico**:

descripción, cantidad, color, tamaño, peso, ubicación

Se envía a `POST /indicios`.



Indicios UI:



Reporte resumido

Archivo: [Reporte.jsx](#)

Filtros:

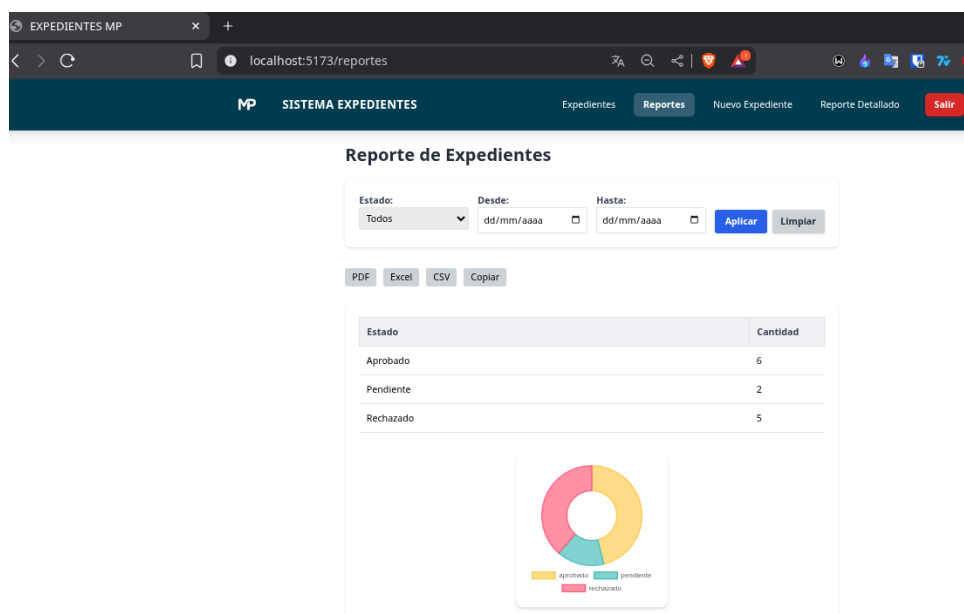
Estado

Fecha inicio / fin

Tabla con columnas: estado y cantidad.

Gráfica [Doughnut](#) por estado.

Botones de exportación (PDF, Excel, CSV, Copiar).



Reporte detallado

Archivo: [ReporteDetallado.jsx](#)

Tabla con columnas:

número, descripción, estado, fecha_registro,

usuario_creacion, justificación, usuario_aprobación, usuario_rechazo, fecha_aprobación...

Botones de exportación reutilizando [ReportButtons](#).



Número	Descripción	Estado	Fecha Registro	Usuario Crea	Justificación Rechaz
EXP-014	testtt 14	Aprobado	29/11/2025, 12:19:25	Coordinador	por no me gusto

EXP-005	as	Aprobado	22/11/2025, 12:43:18 p. m.		Lorem Ipsum
EXP-004	test-expediente	Rechazado	22/11/2025, 12:23:16 a. m.		—
EXP-002	Prueba de expediente sis	Rechazado	22/11/2025, 12:22:08 a. m.		—
3	test	Rechazado	22/11/2025, 12:20:46 a. m.		test rechazo
EXP-001	Prueba expediente	Rechazado	21/11/2025, 3:38:43 p. m.		Lorem Ipsum

Seguridad y control de acceso

Autenticación ligera: login por nombre contra tabla **Usuarios**.

Autorización por rol:

AuthContext guarda usuario y rol.

Rutas frontend protegidas con RutaPrivada.

Backend usa middleware `requireRole(["coordinador"])` o `["tecnico"]`.

Rutas accesibles:

Técnico: expedientes, detalle, registrar indicios.

Coordinador: todo + crear, aprobar, rechazar, reportes.