

Daniel Alfonso Velásquez Hernández – Sistema de gestión de evidencias DICRI

Guatemala, agosto 10 de 2025

Ministerio Publico – dirección de investigaciones criminalísticas

Manual Técnico – Sistema de gestión de evidencias DICRI

Contenido

Manual Técnico – Sistema de gestión de evidencias DICRI.....	1
1. Introducción.....	3
2. Arquitectura General	3
3. Diagrama ER (se adjunta pdf en entregable).....	5
4. Diagrama de infraestructura	5
5. Diagrama de arquitectura.....	6
Node JS + Express API	6
6. Módulo Auth.....	6
7. Módulo Evidencias.....	8
8. Módulo Expedientes	9
9. Módulo Usuarios.....	11
10. Objeto DB y Login Genérico	12
ReactJS + Nginx	13
Recomendaciones	15
Anexos:	15
docker-compose.yml	15
Multicontainer app.....	16
Consumo y rendimiento del sistema >1.2GB de ram	16
Optimizacion.....	17
Capturas de pantalla de sistema.....	17

1. Introducción

Este documento describe la estructura y funcionamiento del sistema desarrollado para gestión de evidencias.

2. Arquitectura General

El sistema desarrollado implementa una **arquitectura de software en capas** basada en el modelo **Three-Tier Architecture**, complementada con patrones de diseño que garantizan modularidad, escalabilidad y facilidad de mantenimiento.

a. Modelo de 3 capas (Three-Tier Architecture)

- **Capa de Presentación (Presentation Layer)**
Compuesta por el frontend desarrollado en React.js. Esta capa ofrece una interfaz gráfica dinámica y responsiva para el usuario final, gestionando la interacción y el flujo de datos hacia el backend a través de solicitudes HTTP y Web APIs.
- **Capa de Lógica de Negocio (Business Logic Layer)**
Implementada en Node.js con Express.js, donde residen los controladores, validaciones y reglas de negocio. Esta capa procesa las solicitudes recibidas, aplica la lógica correspondiente y coordina la comunicación con la capa de datos.
- **Capa de Acceso a Datos (Data Access Layer)**
Basada en un servidor SQL Server, a la cual se accede mediante un módulo genérico de conexión (db) implementado con el patrón Singleton. Este patrón garantiza que toda la aplicación utilice una única instancia de conexión a la base de datos, optimizando recursos y evitando conflictos.

b. Patrones de Diseño Aplicados

- **MVC (Model-View-Controller)**
Adaptado al contexto del sistema:
 - **Model:** Representado por el acceso a datos a través del módulo db y procedimientos almacenados en SQL Server.
 - **View:** Componentes de React que renderizan la interfaz gráfica.
 - **Controller:** Controladores en Node.js que gestionan la lógica de negocio.
- **Singleton**
Usado en el módulo db para mantener una única conexión persistente a la base de datos.

- **Middleware / Chain of Responsibility**

Empleado en Express para manejo de autenticación, validación y control de errores, permitiendo que las funciones se ejecuten en cadena de forma desacoplada.

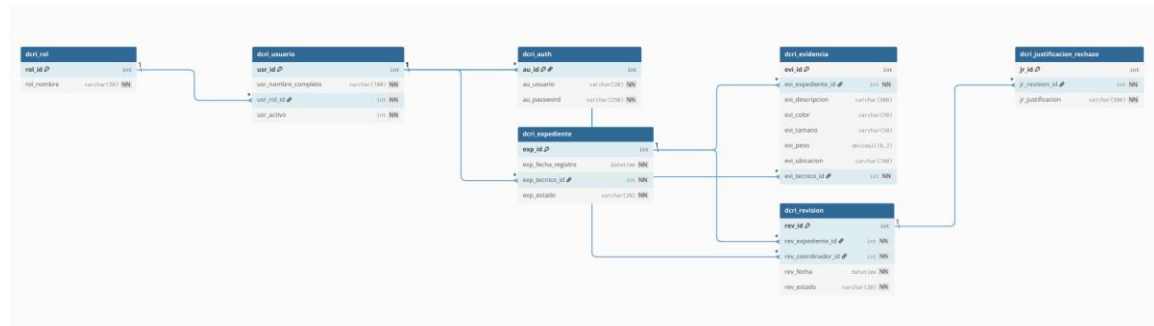
- **Modularidad**

El sistema está dividido en módulos independientes (usuarios, autenticación, expedientes, evidencias, revisiones, justificaciones, etc.), cada uno con su propia estructura de rutas, controladores y lógica de negocio.

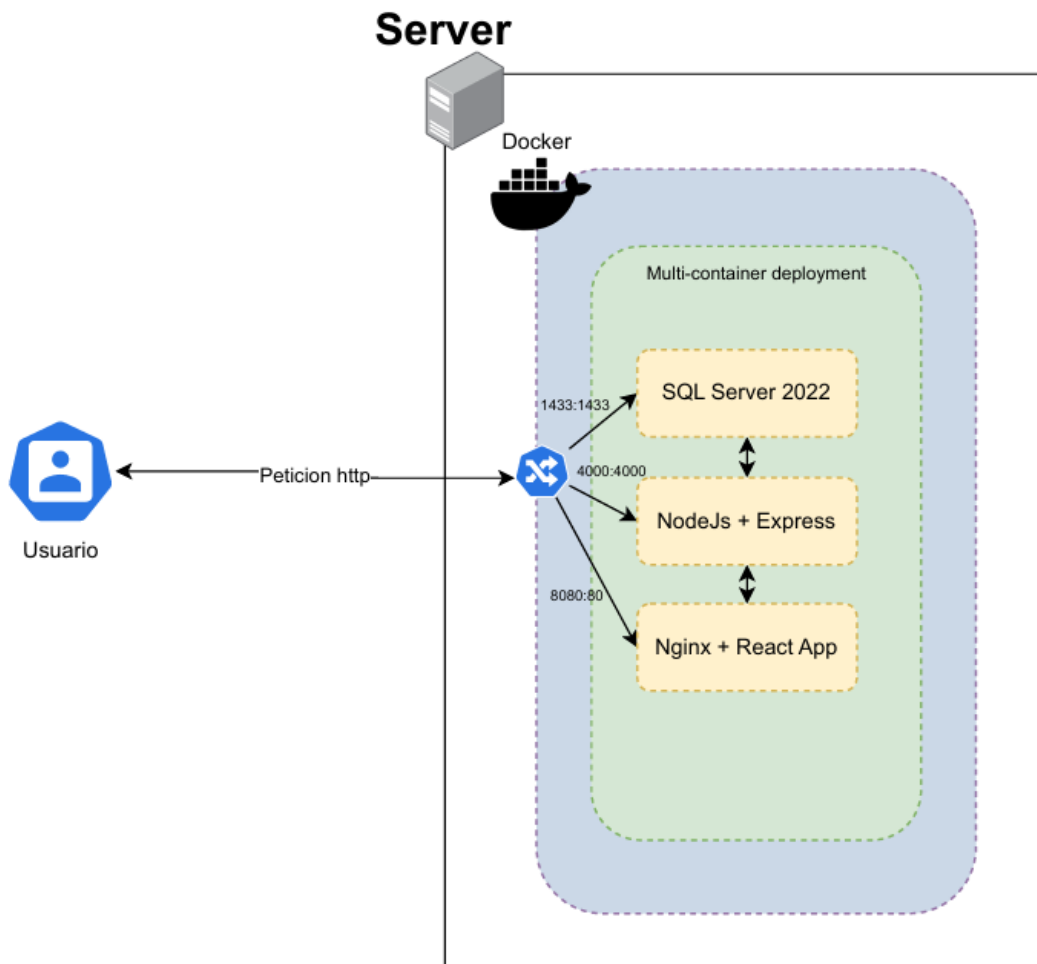
c. Flujo General del Sistema

1. El usuario interactúa con la **interfaz web en React**, generando acciones que envían solicitudes HTTP al backend.
2. Las peticiones llegan a las rutas definidas en **Express**, donde se aplican middlewares de seguridad y validaciones.
3. Los controladores procesan la lógica de negocio correspondiente y solicitan datos al módulo db.
4. El módulo db (Singleton) se comunica con la base de datos **SQL Server**, ejecutando consultas o procedimientos almacenados.
5. La respuesta se devuelve al controlador, que la envía al frontend en formato JSON.
6. El frontend renderiza la información en la interfaz para el usuario final.

3. Diagrama ER (se adjunta pdf en entregable)



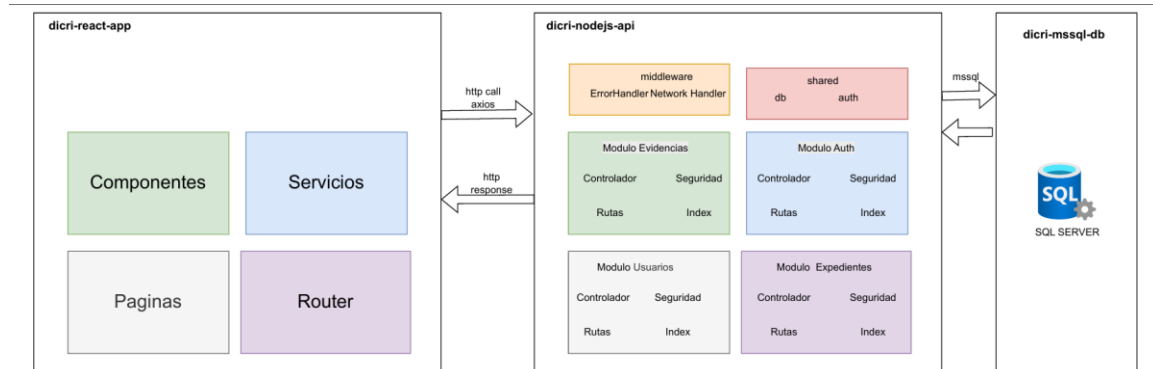
4. Diagrama de infraestructura



La infraestructura, Nginx está configurado como servidor web para la entrega eficiente de la aplicación React. El entorno de despliegue está basado en contenedores Docker, y se ha desarrollado un archivo Docker Compose que orquesta múltiples contenedores, tanto backend como frontend poseen su propio dockerfile, facilitando la integración de los distintos servicios

que componen el sistema. Esta arquitectura multicontenedor permite una gestión escalable, portabilidad entre ambientes y despliegues consistentes en distintos entornos.

5. Diagrama de arquitectura



Node JS + Express API

Se adjuntan dos archivos para documentar pruebas y definiciones en postman y swagger llamados:

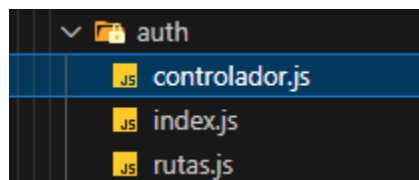
1. dicri-api-swagger.yaml
2. DICRI.postman_collection.json

6. Módulo Auth

Descripción del módulo Auth.

Modulo utilizado para gestionar el inicio de sesión con los algoritmos de seguridad de encriptación y JWT.

Estructura de carpetas del módulo



Fragmento de código del controlador.

```
1  const TABLA = 'dcric_auth';
2  const auth = require('.././auth');
3  const bcrypt = require('bcrypt');
4
5
6  module.exports = function(dbinyectada){
7
8      let db = dbinyectada;
9
10     if(!db){
11         db = require('.././DB/mssql');
12     }
13
14     async function login(usuario, password) {
15         const data = await db.query(TABLA, { au_usuario: usuario });
16
17         if (data.length === 0) {
18             throw new Error('Usuario no encontrado');
19         }
20
21         const user = data[0]; // tomar la primera fila
22         return bcrypt.compare(password, user.au_password)
23             .then(resultado => {
24                 if (resultado) {
25                     return auth.asignarToken({ ...user });
26                 } else {
27                     throw new Error('Información inválida');
28                 }
29             });
30     }
31 }
```

Fragmento de código de las rutas

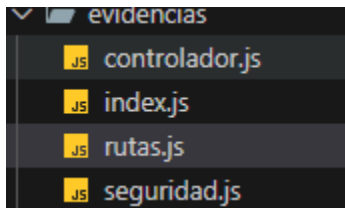
```
1  const express = require('express');
2
3  const respuesta = require('.././red/respuestas');
4
5  const controlador = require('./index');
6
7  const router = express.Router();
8
9  router.post('/login', login);
10 router.get('/login/:user', getRol);
11
12 async function login(req,res,next){
13     try{
14         const token = await controlador.login(req.body.usuario, req.b
15         respuesta.success(req,res,token,200);
16     }
17     catch(err){
18         next(err);
19     }
20 };
21
22
23 async function getRol(req,res,next){
24     try{
25         const item = await controlador.uno(req.params.user);
26         respuesta.success(req,res,item,200);
27     }
28     catch(err){
29         next(err);
30     }
31 };
```

7. Módulo Evidencias

Descripción del módulo Evidencias

Modulo en el cual se gestionan las llamadas respectivas al crud de las mismas. Se definen las reglas de ruteo y disponibilizacion de endpoint.

Estructura de carpetas del módulo



Fragmento de código de rutas

```
1  const express = require('express');
2
3  const respuesta = require('../red/respuestas');
4  const seguridad = require('../seguridad');
5  const controlador = require('../index');
6
7  const router = express.Router();
8
9  router.get('/', seguridad(), todos);
10 router.get('/:id', uno);
11 router.post('/', agregar);
12 router.patch('/', agregar);
13 router.delete('/:id', eliminar);
14 router.put('/', actualizar);
15
16
17 async function todos(req,res,next){
18
19   try{
20     const items = await controlador.todos();
21     respuesta.success(req,res,items,200)
22   }
23   catch(err){
24     next(err);
25   }
26 };
27
28 async function uno(req,res,next){
29   try{
30     const items = await controlador.uno(req.params.id);
31     respuesta.success(req,res,items,200);
32   }
```

Fragmento de código de controlador


```
const TABLA = 'dcric_evidencia';

module.exports = function(dbinyectada){

  let db = dbinyectada;

  if(!db){
    db = require('../DB/mssql');
  }

  function todos()
  {
    return db.todos(TABLA);
  }

  function uno(id)
  {
    return db.uno(TABLA,id,'evi_expediente_id');
  }

  function agregar(body)
  {
    return db.agregar(TABLA,body, 'evi_id');
  }

  function actualizar(id, body) {
    return db.actualizar(TABLA, id, body, 'evi_id');
  }

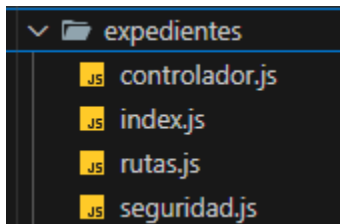
  function eliminar(id)
```

8. Módulo Expedientes

Descripción del módulo Expedientes

Gestiona la creación, consulta, actualización y eliminación lógica de expedientes, con validación de datos y filtros por criterios específicos. Usa endpoints REST en Express.js, procedimientos almacenados en SQL Server y responde en JSON para integrarse con React.

Estructura de carpetas del módulo



Fragmento de código del controlador

```
backend > src > modulos > expedientes > controladores > <unknown> > exports > actuali
5  module.exports = function(dbinyectada){
13
14      function todos()
15      {
16          return db.todos(TABLA);
17      }
18
19      function uno(id)
20      {
21          return db.uno(TABLA,id);
22      }
23
24      function agregar(body)
25      {
26          return db.agregar(TABLA,body, 'exp_id');
27      }
28
29      function actualizar(id, body) {
30          return db.actualizar(TABLA, id, body,'exp_id');
31      }
32
33      function eliminar(id)
34      {
35          return db.eliminar(TABLA,id);
36      }
37
38      return {
39          todos,
40          uno,
41          eliminar,
42          agregar,
43          actualizar
44      }
```

Fragmento de código de las rutas

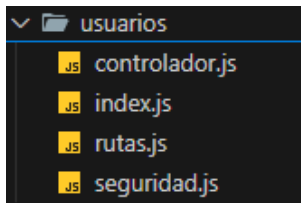
```
backend > src > modulos > expedientes > rutas.js > actualizar > result
1  const express = require('express');
2
3  const respuesta = require('../red/respuestas');
4  const seguridad = require('../seguridad');
5  const controlador = require('./index');
6
7  const router = express.Router();
8
9  router.get('/', seguridad(), todos);
10 router.get('/:id', uno);
11 router.post('/', agregar);
12 router.patch('/', agregar);
13 router.delete('/:id', eliminar);
14 router.put('/', actualizar);
15
16
17 async function todos(req,res,next){
18     try{
19         const items = await controlador.todos();
20         respuesta.success(req,res,items,200)
21     }
22     catch(err){
23         next(err);
24     }
25 }
26
27
28 async function uno(req,res,next){
29     try{
30         const items = await controlador.uno(req.params.id);
31         respuesta.success(req,res,items,200);
32     }
33     catch(err){
34         next(err);
35     }
36 }
```

9. Módulo Usuarios

Descripción del módulo Usuarios

Administra usuarios, incluyendo alta, baja, modificación, asignación de roles y control de acceso. Las rutas están protegidas con middleware de autenticación, con validaciones previas y controladores desacoplados que interactúan con la base de datos.

Estructura de carpetas del módulo



Fragmento de código del controlador

```
backend > src > modulos > usuarios > controlador.js > <unknown> > export
1  const TABLA = 'dcri_usuario';
2  const auth = require('../auth');
3
4
5  module.exports = function(dbinyectada){
6
7    let db = dbinyectada;
8
9    if(!db){
10     db = require('../DB/mssql');
11   }
12
13   function todos(){
14     {
15       return db.todos(TABLA);
16     }
17   }
18
19   function uno(id,idColumn = 'usr_id'){
20     {
21       return db.uno(TABLA,id,idColumn);
22     }
23   }
24
25   async function agregar(body){
26     {
27       const usuario = {
28         usr_id: body.usr_id,
29         usr_nombre_completo: body.usr_nombre_completo,
30         usr_rol_id: body.usr_rol_id,
31         usr_activo: body.usr_activo
32       }
33     }
34   }
35 }
```

Fragmento de código de las rutas

```
backend > src > modulos > usuarios > rutas.js > uno > items
15  async function todos(req,res,next){
16    const items = await controlador.todos();
17    respuesta.success(req,res,items,200)
18  }
19  catch(err){
20    next(err);
21  }
22  };
23
24  async function uno(req,res,next){
25    try{
26      const items = await controlador.uno(req.params.id);
27      respuesta.success(req,res,items,200);
28    }
29    catch(err){
30      next(err);
31    }
32  };
33
34  async function agregar(req,res,next){
35    try{
36      const items = await controlador.agregar(req.body);
37      if(req.body.usr_id == 0){
38        {
39          mensaje = 'Item guardado con exito';
40        }
41      }
42    }
43  }
```

10. Objeto DB y Login Genérico

Descripción del objeto db genérico y su uso en los diferentes módulos

Es la capa única de conexión a la base de datos SQL Server, implementada con patrón Singleton para optimizar recursos. Ejecuta consultas y procedimientos almacenados de forma asíncrona, gestionando conexiones y errores eficientemente

Fragmento de código del objeto db

```
1 // db/mssql.js
2 const sql = require('mssql');
3 const config = require('../config');
4
5 const dbconfig = {
6   server: config.sql.server,
7   user: config.sql.user,
8   password: config.sql.password,
9   database: config.sql.database,
10  options: {
11    encrypt: false, // Desactivar para local
12    trustServerCertificate: true // Necesario para dev
13  },
14  pool: {
15    max: 10,
16    min: 0,
17    idleTimeoutMillis: 30000
18  }
19 };
20
21 let pool;
22
23 async function conSql() {
24   try {
25     if (pool) {
26       await pool.close();
27     }
28     pool = await sql.connect(dbconfig);
29     console.log('DB CONECTADA!');
30   } catch (err) {
31     console.error('[db err]', err);
32     setTimeout(conSql, 2000);
33   }
34 }
```

Descripción y código del módulo de login genérico

Se encarga de autenticar usuarios mediante validación de credenciales y generación de tokens JWT con expiración. Protege endpoints restringidos con middleware, usando bcrypt para comparar contraseñas encriptadas y un endpoint /login en Express.

```
backend > src > auth > index.js > asignarToken
1  const jwt = require('jsonwebtoken');
2  //const auth = require('../modulos/auth');
3  const config = require('../config');
4
5  const secret = config.jwt.secret;
6  const error = require('../middleware/errores')
7
8
9  function verificarToken(token){
10     return jwt.verify(token,secret);
11 }
12
13 function asignarToken(data)
14 {
15     return jwt.sign(data,secret);
16 }
17
18 const chequearToken = {
19     confirmarToken: function(req,id){
20         const decodificado = decodificarCabecera(req);
```

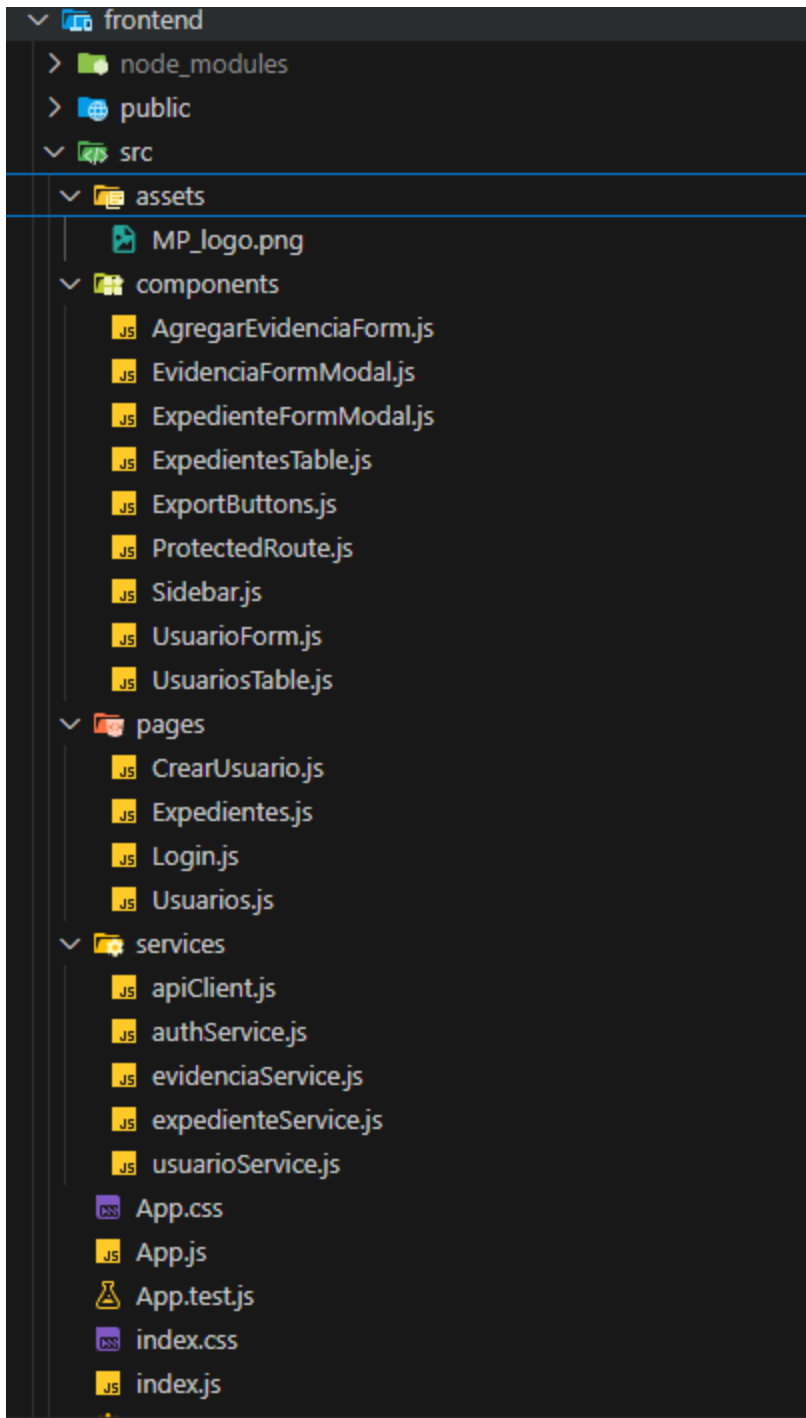
ReactJS + Nginx

La aplicación frontend está desarrollada en React.js y se despliega mediante un servidor Nginx, encargado de servir el contenido estático optimizado para ambientes productivos. La arquitectura del proyecto está basada en una organización modular que incluye componentes reutilizables, páginas funcionales, servicios para la comunicación con la API y un sistema de enrutamiento (React Router) que controla la navegación interna de la aplicación.

Se implementa un mecanismo de autenticación mediante una pantalla de login, y se protegen las rutas críticas del sistema para garantizar que solo usuarios autenticados puedan acceder a las funcionalidades restringidas. Entre las páginas principales se encuentran: Crear Usuario, Expedientes, Login y Gestión de Usuarios, cada una respaldada por servicios específicos que encapsulan las llamadas HTTP y la lógica de interacción con el backend.

Para optimizar el rendimiento y la experiencia de usuario, los componentes están diseñados para maximizar la reutilización y minimizar renderizados innecesarios, haciendo uso eficiente del ciclo de vida y hooks de React.

A continuación, se presentan capturas de como fue estructurado el proyecto:

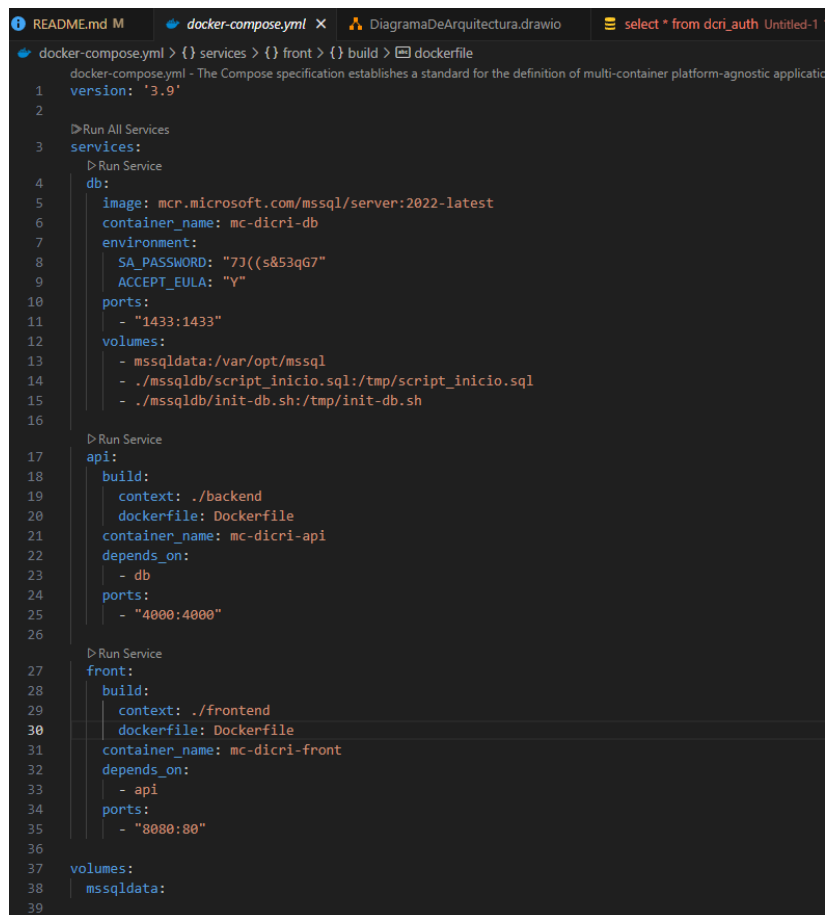


Recomendaciones

- En el repositorio de github y en el entregable, se adjunta un README.md con el paso a paso para levantar el proyecto.
- Es compatible con cualquier entorno que tenga git, Docker, Docker compose y node.























Anexos:

docker-compose.yml

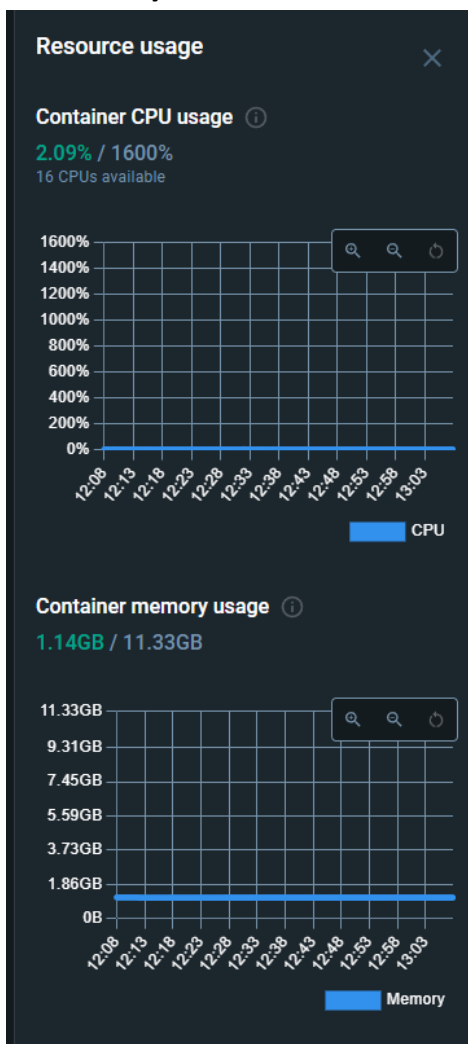


```
1 docker-compose.yml > {} services > {} front > {} build > {} dockerfile
2
3 docker-compose.yml - The Compose specification establishes a standard for the definition of multi-container platform-agnostic applications
4 version: '3.9'
5
6 > Run All Services
7 services:
8   > Run Service
9   db:
10     image: mcr.microsoft.com/mssql/server:2022-latest
11     container_name: mc-dicri-db
12     environment:
13       SA_PASSWORD: "7J((s&53q67"
14       ACCEPT_EULA: "Y"
15     ports:
16       - "1433:1433"
17     volumes:
18       - mssqldata:/var/opt/mssql
19       - ./mssqldb/script_inicio.sql:/tmp/script_inicio.sql
20       - ./mssqldb/init-db.sh:/tmp/init-db.sh
21
22   > Run Service
23   api:
24     build:
25       context: ./backend
26       dockerfile: Dockerfile
27     container_name: mc-dicri-api
28     depends_on:
29       - db
30     ports:
31       - "4000:4000"
32
33   > Run Service
34   front:
35     build:
36       context: ./frontend
37       dockerfile: Dockerfile
38     container_name: mc-dicri-front
39     depends_on:
40       - api
41     ports:
42       - "8080:80"
43
44 volumes:
45   mssqldata:
```

Multicontainer app


<input type="checkbox"/>		gestor_evidencias_s	Running (3/3)	1.63%	  
<input type="checkbox"/>		mc-dicri-db fc32512b071b 	mcr.micr Running 1433:1433 	1.63%	  
<input type="checkbox"/>		mc-dicri-api 64e5f8cd465a 	gestor_evic Running 4000:4000 	0%	  
<input type="checkbox"/>		mc-dicri-front d3e7aa9ff64b 	gestor_evic Running 8080:80 	0%	  

Consumo y rendimiento del sistema >1.2GB de ram



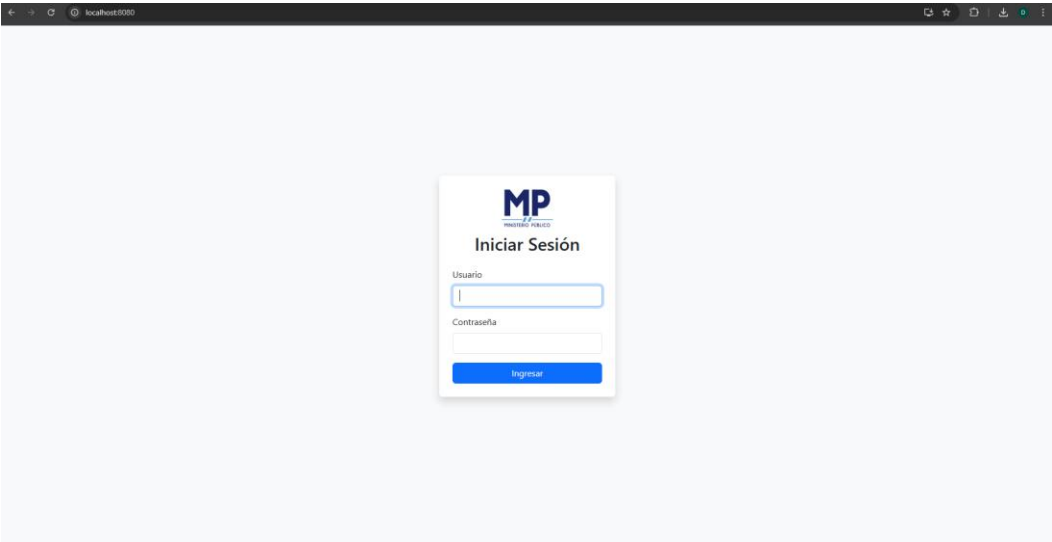
Optimizacion

Imágenes creadas livianas para correr en entornos con pocos recursos

Name	Size	Last pushed 
danielvh01/dicri-react-app	114.3 MB	about 2 hours ago
danielvh01/dicri-api	86.9 MB	about 2 hours ago

Capturas de pantalla de sistema

Login



Menu de expedientes



Usuario: divelasquez

Rol:

Expedientes

Usuarios

Cerrar sesión

Listado de Expedientes - DICRI

+ Nuevo Expediente

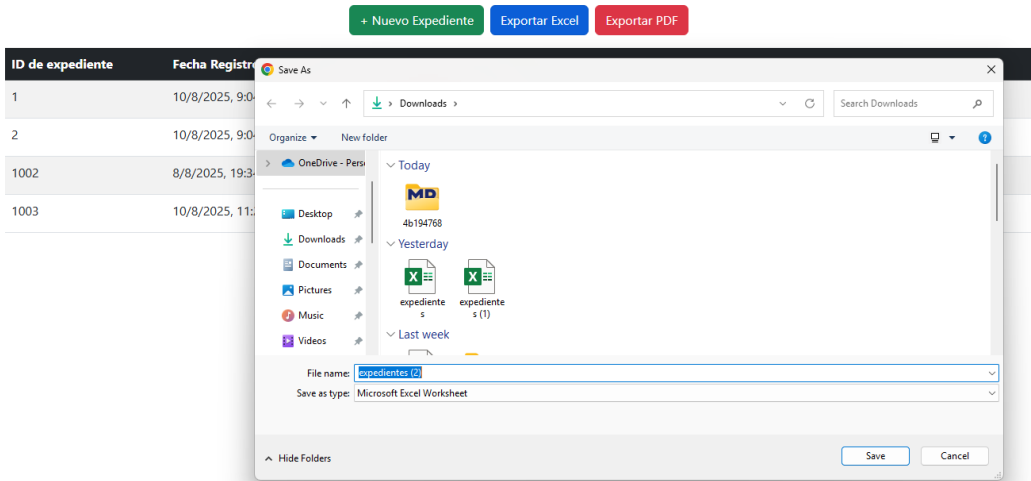
Exportar Excel

Exportar PDF

ID de expediente	Fecha Registro	ID del técnico	Estado	Acciones
1	10/8/2025, 9:04:12	1	Pendiente	<div>Ver evidencias</div> <div>+ Agregar Evidencia</div> <div>Editar expediente</div>
2	10/8/2025, 9:04:12	2	Pendiente	<div>Ver evidencias</div> <div>+ Agregar Evidencia</div> <div>Editar expediente</div>
1002	8/8/2025, 19:34:43	2	Pendiente	<div>+ Agregar Evidencia</div> <div>Editar expediente</div>
1003	10/8/2025, 11:20:53	2	En Proceso	<div>+ Agregar Evidencia</div> <div>Editar expediente</div>

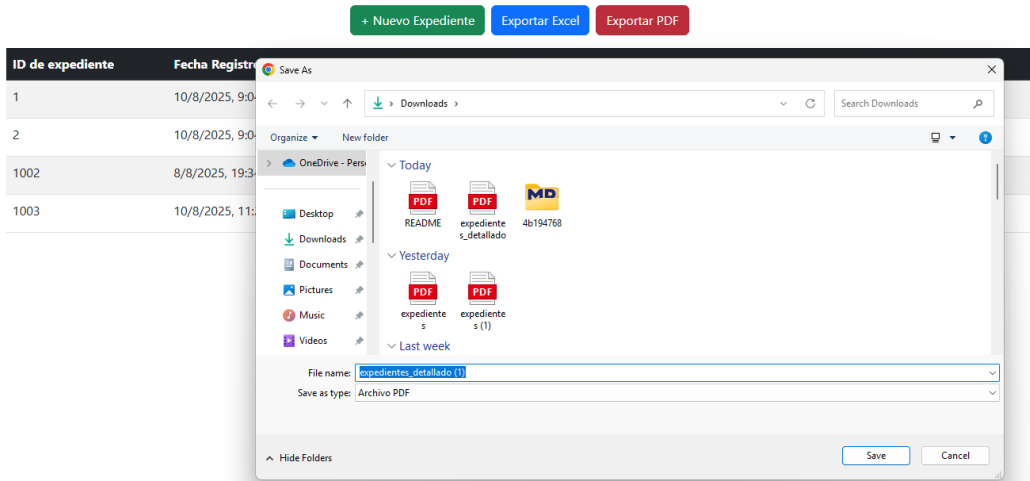
Exportar a Excel

Listado de Expedientes - DICRI



Exportar a pdf

Listado de Expedientes - DICRI



Reporte generado

Reporte Detallado de Expedientes DICRI

Expediente ID: 1
Fecha de registro: 2025-08-10T15:04:12.023Z
ID Técnico: 1
Estado: Pendiente

#	Descripción	Color	Tamaño	Peso (kg)	Ubicación
1	Fragmento de material recolectado	Marrón	Mediano	12.5	Almacén 1 - Caja 3
2	Muestra orgánica en soporte	Rojo oscuro	Pequeño	5.2	Estante A4

Expediente ID: 2
Fecha de registro: 2025-08-10T15:04:12.023Z
ID Técnico: 2
Estado: Pendiente

#	Descripción	Color	Tamaño	Peso (kg)	Ubicación
1	Elemento sintético adherido	Transparente	Grande	45.3	Zona refrigerada

Expediente ID: 1002
Fecha de registro: 2025-08-09T01:34:43.630Z
ID Técnico: 2
Estado: Pendiente
Evidencias: No tiene

Expediente ID: 1003
Fecha de registro: 2025-08-10T17:20:53.220Z
ID Técnico: 2
Estado: En Proceso
Evidencias: No tiene

Formulario de nuevo expediente

Listado de Expedientes - DICRI

+ Nuevo Expediente

Exportar Excel

Exportar PDF

ID de expediente	Fecha Registro	ID del técnico	Estado	Acciones
1	10/8/2025, 9:04:12	1	Pendiente	<div>Ver evidencias</div> <div>+ Agregar Evidencia</div> <div>Editar expediente</div>
2	10/8/2025, 9:04:12	2	Pendiente	<div>Ver evidencias</div> <div>+ Agregar Evidencia</div> <div>Editar expediente</div>
1002	8/8/2025, 19:34:43	2	Pendiente	<div>Ver evidencias</div> <div>+ Agregar Evidencia</div> <div>Editar expediente</div>
1003	10/8/2025, 11:20:53	2	En Proceso	<div>Ver evidencias</div> <div>+ Agregar Evidencia</div> <div>Editar expediente</div>

Nuevo Expediente

Fecha Registro:

10/08/2025 19:14

ID Técnico:

Estado:

Pendiente

Guardar

Cancelar

Visualización de evidencias asociadas por expedientes

Listado de Expedientes - DICRI

+ Nuevo ExpedienteExportar ExcelExportar PDF

ID de expediente	Fecha Registro	ID del técnico	Estado	Acciones
1	10/8/2025, 9:04:12	1	Pendiente	Ocultar evidencias+ Agregar EvidenciaEditar expediente

Descripción: Fragmento de material recolectado

Color: Marrón

Tamaño: Mediano

Peso: 12.5

Ubicación: Almacén 1 - Caja 3

Editar Evidencia

Descripción: Muestra orgánica en soporte

Color: Rojo oscuro

Tamaño: Pequeño

Peso: 5.2

Ubicación: Estante A4

Editar Evidencia

Ingreso de nueva evidencia

Listado de Expedientes - DICRI

+ Nuevo ExpedienteExportar ExcelExportar PDF

ID de expediente	Fecha Registro	ID del técnico	Estado	Acciones
1	10/8/2025, 9:04:12	1	Pendiente	Ver evidencias+ Agregar EvidenciaEditar expediente
2	10/8/2025, 9:04:12			Ver evidencias+ Agregar EvidenciaEditar expediente
1002	8/8/2025, 19:34:43			Ver evidencias+ Agregar EvidenciaEditar expediente
1003	10/8/2025, 11:20:53			Ver evidencias+ Agregar EvidenciaEditar expediente

Nueva Evidencia

Descripción:

Color:

Tamaño:

Peso:

Ubicación:

ID Técnico:

Guardar

Cancelar

Gestion de usuarios

MP

Usuario: dvelasquez

Rol: -

Expedientes

Usuarios

Gestión de Usuarios

Crear usuario nuevo

ID	Nombre Completo	Rol ID	Activo	Acciones	
1	usuario1	1	No	Editar	Activar
2	usuario2	1	Sí	Editar	Desactivar
3	usuario3	2	Sí	Editar	Desactivar
1002	Gilberto Santacruz	2	No	Editar	Activar
1003	Daniel Velasquez	2	Sí	Editar	Desactivar

Creacion de nuevo usuario

Crear Usuario

Nombre completo *

Usuario *

Contraseña *

Rol

Coordinador

▼

Activo


Sí

▼

Crear Usuario

Cancelar

Sidebar para n menus a futuro



Usuario: dvelasquez

Rol: -

Expedientes

Usuarios

Cerrar sesión

ID de expediente
1
2
1002
1003