

GESTION DE EVIDENCIA DICRI

Manual Tecnico

Ing. Luis Alfonso Ordoñez

Base de datos

La base de datos está diseñada para gestionar expedientes, sus indicios asociados, el proceso de revisión, y los usuarios que interactúan con el sistema, garantizando trazabilidad, control y consistencia de la información.

El esquema se compone de cinco entidades principales: EXPEDIENTES, INDICIOS, REVISIONES_EXPEDIENTE, USUARIOS y TIPO_USUARIO.

Diagrama ER

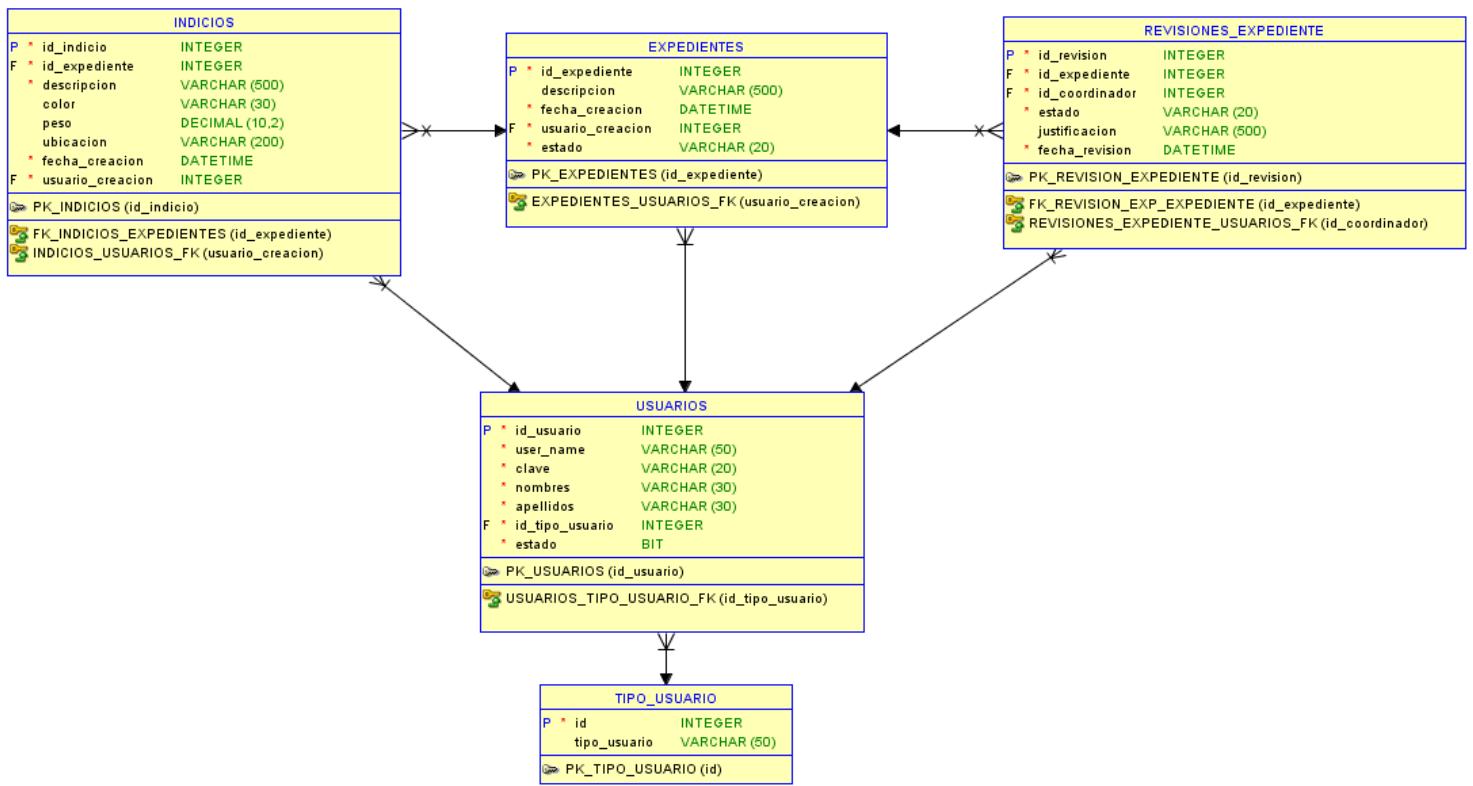


Tabla: EXPEDIENTES - Expedientes Judiciales o de Caso

Esta tabla representa el núcleo del sistema. Almacena la información principal de cada expediente o caso que se registra, incluyendo su descripción, estado actual y el usuario responsable de su creación.

El estado del expediente puede cambiar a lo largo del tiempo conforme se realizan **revisiones**, y sirve como referencia principal para relacionar indicios y controles de validación.

Campo	Tipo de Dato	Clave	Descripción
id_expediente	INTEGER	PK	Identificador único del expediente.
descripcion	VARCHAR(500)		Descripción detallada del expediente o caso.
fecha_creacion	DATETIME		Fecha y hora en que se registró el expediente.
estado	VARCHAR(20)		Estado actual del expediente (Ej.: Pendiente, Aprobado, Rechazado).
usuario_creacion	INTEGER	FK	Usuario que creó el expediente (USUARIOS.id_usuario).

Tabla: INDICIOS - Evidencia o Indicios Asociados a un Expediente

Esta tabla almacena los indicios o evidencias relacionadas a un expediente específico. Cada indicio contiene información descriptiva, características físicas y ubicación, permitiendo un control detallado de los elementos asociados al caso.

Los indicios dependen completamente de un expediente, por lo que su existencia está ligada a este mediante una relación directa.

Campo	Tipo de Dato	Clave	Descripción
id_indicio	INTEGER	PK	Identificador único del indicio.
id_expediente	INTEGER	FK	Expediente al que pertenece el indicio.
descripcion	VARCHAR(500)		Descripción del indicio o evidencia.
color	VARCHAR(30)		Color del indicio (clasificación visual).
peso	DECIMAL(10,2)		Peso aproximado del indicio.
ubicacion	VARCHAR(200)		Ubicación física o lógica del indicio.
fecha_creacion	DATETIME		Fecha y hora de registro del indicio.
usuario_creacion	INTEGER	FK	Usuario que registró el indicio.

Tabla: REVISIONES_EXPEDIENTE - Historial de Revisiones del Expediente

Esta tabla permite llevar un historial de todas las revisiones realizadas a un expediente. Cada revisión es ejecutada por un coordinador y contiene un estado y una justificación, lo que permite auditar decisiones, validar procesos y mantener transparencia.

El estado de la última revisión suele reflejarse directamente en el estado del expediente.

Campo	Tipo de Dato	Clave	Descripción
id_revision	INTEGER	PK	Identificador único de la revisión.
id_expediente	INTEGER	FK	Expediente revisado.
id_coordinador	INTEGER	FK	Usuario coordinador que realizó la revisión.
estado	VARCHAR(20)		Resultado de la revisión (Aprobado / Rechazado).
justificacion	VARCHAR(500)		Motivo o comentario de la decisión.
fecha_revision	DATETIME		Fecha y hora de la revisión.

Tabla: USUARIOS - Usuarios del Sistema

Esta tabla almacena la información de todas las personas que interactúan con el sistema.

Los usuarios pueden tener distintos roles y responsabilidades, como crear expedientes, registrar indicios o realizar revisiones.

El sistema puede habilitar o deshabilitar usuarios mediante el campo estado.

Campo	Tipo de Dato	Clave	Descripción
id_usuario	INTEGER	PK	Identificador único del usuario.
user_name	VARCHAR(50)		Nombre de usuario para autenticación.
clave	VARCHAR(20)		Contraseña del usuario (recomendado hash).
nombres	VARCHAR(50)		Nombre(s) del usuario.
apellidos	VARCHAR(50)		Apellidos del usuario.
id_tipo_usuario	INTEGER	FK	Tipo o rol del usuario.

Tabla: TIPO_USUARIO - Roles y Perfiles de Usuario

Esta tabla define los roles del sistema, permitiendo controlar permisos y responsabilidades.

Cada usuario pertenece a un tipo específico, lo que facilita la gestión de accesos y funcionalidades dentro de la aplicación.

Campo	Tipo de Dato	Clave	Descripción
id	INTEGER	PK	Identificador del tipo de usuario.
tipo_usuario	VARCHAR(50)		Nombre del rol (Administrador, Coordinador, Operador).

Resumen de relaciones

Tabla Origen	FK	Tabla Destino	Cardinalidad	Descripción
EXPEDIENTES	usuario_creacion	USUARIOS	N:1	Un usuario puede crear múltiples expedientes.
INDICIOS	id_expediente	EXPEDIENTES	N:1	Un expediente puede contener varios indicios.
INDICIOS	usuario_creacion	USUARIOS	N:1	Un usuario puede registrar múltiples indicios.
REVISIONES_EXPEDIENTE	id_expediente	EXPEDIENTES	N:1	Un expediente puede tener varias revisiones.
REVISIONES_EXPEDIENTE	id_coordinador	USUARIOS	N:1	Un coordinador puede realizar múltiples revisiones.
USUARIOS	id_tipo_usuario	TIPO_USUARIO	N:1	Un rol puede ser asignado a muchos usuarios.

Procedimientos almacenados y funciones

Los procedimientos almacenados encapsulan la **lógica de negocio del sistema**, permitiendo controlar validaciones, transacciones y mensajes de respuesta desde la base de datos, garantizando integridad y consistencia.

Procedimientos para la tabla USUARIOS

SP_CREA_USUARI

Crea un nuevo usuario en el sistema asignándole un tipo de usuario específico. Maneja errores mediante control de excepciones.

Parámetros:

- @user_name (VARCHAR): Nombre de usuario.
- @clave (VARCHAR): Contraseña del usuario.
- @nombres (VARCHAR): Nombres del usuario.
- @apellidos (VARCHAR): Apellidos del usuario.
- @id_tipo_usuario (INT): Rol del usuario.
- @mensaje (OUTPUT): Mensaje de resultado.

Retorno:

- 0: Usuario creado correctamente.
- 1: Error durante la creación.

SP_MODIFICA_USUARIO

Actualiza la información básica de un usuario existente y permite activar o desactivar su estado.

Reglas de negocio:

No permite modificar usuarios inexistentes.

Parámetros:

- @id_usuario (INT)
- @user_name (VARCHAR)
- @nombres (VARCHAR)
- @apellidos (VARCHAR)
- @id_tipo_usuario (INT)
- @estado (BIT)
- @mensaje (OUTPUT)

Retorno:

- 0: Usuario modificado.
- 2: Usuario no existe.

SP_ELIMINA_USUARIO

Realiza una eliminación lógica del usuario, desactivándolo en el sistema.

Parámetros:

- @id_usuario (INT)
- @mensaje (OUTPUT)

Retorno:

- 0: Usuario desactivado.
- 2: Usuario no existe.

Procedimientos para la tabla EXPEDIENTES

SP_CREA_EXPEDIENTE

Registra un nuevo expediente en el sistema y devuelve el identificador generado.

Parámetros:

- @descripcion (VARCHAR)
- @usuario_creacion (INT)
- @estado (VARCHAR)
- @mensaje (OUTPUT)
- @id_expediente (OUTPUT)

Retorno:

- 0: Expediente creado.
- 1: Error en la creación.

SP_MODIFICA_EXPEDIENTE

Actualiza la descripción y estado de un expediente existente.

Parámetros:

- @id_expediente (INT)
- @descripcion (VARCHAR)
- @estado (VARCHAR)
- @mensaje (OUTPUT)

Retorno:

- 0: Expediente modificado.
- 2: Expediente no existe.

SP_ELIMINA_EXPEDIENTE

Elimina un expediente siempre que no esté aprobado.

Reglas de negocio:

No se puede eliminar un expediente aprobado.

Parámetros:

- @id_expediente (INT)
- @mensaje (OUTPUT)

Retorno:

- 0: Expediente eliminado.
- 2: Expediente no existe.
- 3: Expediente aprobado, no eliminable.

Procedimientos para la tabla INDICIOS

SP_CREACION_INDICIOS

Registra un indicio asociado a un expediente y devuelve el ID generado.

Parámetros:

- @id_expediente (INT)
- @descripcion (VARCHAR)
- @color (VARCHAR)
- @peso (DECIMAL)
- @ubicacion (VARCHAR)
- @usuario_creacion (INT)
- @mensaje (OUTPUT)
- @id_indicio (OUTPUT)

Retorno:

- 0: Indicio creado.
- 1: Error en la creación.

SP_MODIFICA_INDICIO

Actualiza la información descriptiva de un indicio existente.

Parámetros:

- @id_indicio (INT)
- @descripcion (VARCHAR)
- @color (VARCHAR)
- @peso (DECIMAL)
- @ubicacion (VARCHAR)
- @mensaje (OUTPUT)

Retorno:

- 0: Indicio modificado.
- 2: Indicio no existe.

SP_ELIMINA_INDICIO

Elimina físicamente un indicio del sistema.

Parámetros:

- @id_indicio (INT)
- @mensaje (OUTPUT)

Retorno:

- 0: Indicio eliminado.
- 2: Indicio no existe.

Procedimientos para la tabla REVISIONES_EXPEDIENTE

SP_CREA_REVISION_EXPEDIENTE

Registra una revisión para un expediente y actualiza automáticamente el estado del expediente, asegurando consistencia mediante transacciones.

Parámetros:

- @id_expediente (INT)
- @id_coordinador (INT)
- @estado (VARCHAR)
- @justificacion (VARCHAR)
- @mensaje (OUTPUT)
- @id_revision (OUTPUT)

Retorno:

- 0: Revisión creada correctamente.
- 1: Error en transacción.
- 2: Expediente no existe.

SP_MODIFICA_REVISION_EXPEDIENTE

Actualiza el estado y justificación de una revisión existente y sincroniza el estado del expediente asociado.

Parámetros:

- @id_revision (INT)
- @estado (VARCHAR)
- @justificacion (VARCHAR)
- @mensaje (OUTPUT)

Retorno:

- 0: Revisión actualizada.
- 1: Error en transacción.
- 2: Revisión no existe.

SP_ELIMINA_REVISION_EXPEDIENTE

Elimina una revisión del expediente.

Parámetros:

- @id_revision (INT)
- @mensaje (OUTPUT)

Retorno:

- 0: Revisión eliminada correctamente.

Funciones del sistema

FN_LOGIN_USER

Función utilizada para validar las credenciales de un usuario activo durante el proceso de autenticación.

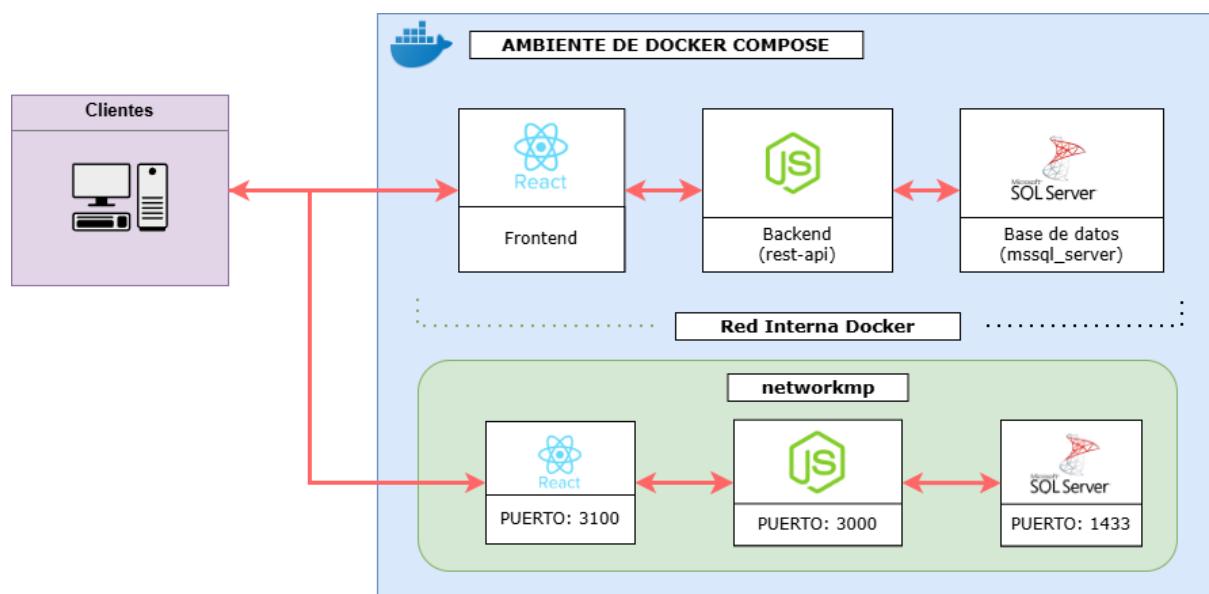
Parámetros:

- @user_name (VARCHAR)
- @clave (VARCHAR)

Retorno:

- 1: Credenciales válidas y usuario activo.
- 0: Credenciales inválidas o usuario inactivo.

Diagrama de la arquitectura



Documentación de Endpoints

El siguiente enlace llevará a la documentación de los endpoints generados en Postman.

<https://www.postman.com/bold-space-678637/workspace/test-dicri-laoc/documentation/12923225-a541ae33-ae4c-4821-9240-aae04cf71d66>

Documentación de API

Index.js

Es el archivo principal de la aplicación.

```
src > index.js > ...
1  import 'dotenv/config'
2  import app from './app.js'
3
4
5  const PORT = process.env.PORT || 3000
6
7  app.listen(PORT)
8  console.log('server iniciado' )
```

App.js

Es el archivo donde se importan todas las dependencias y rutas necesarias para la aplicación.

```
src > app.js > ...
1  import express from 'express'
2  import expedientesRoutes from './routes/expedientes.routes.js'
3  import indiciosRoutes from './routes/indicios.routes.js'
4  import revisionExpedientesRoutes from './routes/revisiones_expediente.routes.js'
5
6  const app = express()
7
8  app.use(express.json())
9  app.use(expedientesRoutes)
10 app.use(indiciosRoutes)
11 app.use(revisionExpedientesRoutes)
12
13 export default app
```

Expedientes.router.js

Es el archivo donde se crean las rutas necesarias para los expedientes.

```
src > routes > expedientes.routes.js > default
1  import { Router } from "express";
2  import {
3      createExpediente,
4      deleteExpediente,
5      getAllExpedientes,
6      getExpediente,
7      updateExpediente,
8  } from "../controllers/expedientes.controllers.js";
9
10 const router = Router();
11
12 router.get("/expedientes", getAllExpedientes);
13 router.get("/expedientes/:id", getExpediente);
14 router.post("/expedientes/add", createExpediente);
15 router.put("/expedientes/update/:id", updateExpediente);
16 router.delete("/expedientes/delete/:id", deleteExpediente);
17
18
19 export default router;
```

Expedientes.controler.js

Es el archivo donde se crea la funcionalidad de las rutas para los expedientes.

```
src > controllers > js expedientes.controllers.js > updateExpediente > mensaje
1 import sql from "mssql"
2
3 import { getConnection } from "../database/conn.js"
4
5 export const getAllExpedientes = async (req, res) => {
6   const pool = await getConnection()
7   var query = 'SELECT * FROM EXPEDIENTES'
8
9   const result = await pool.request().query(query)
10
11  res.json(result.recordset)
12}
13
14 export const getExpediente = async (req, res) => {
15   const {id} = req.params
16
17   try{
18     const pool = await getConnection()
19     const request = pool.request()
20     const query = 'SELECT * FROM EXPEDIENTES WHERE id_expediente = @idExpediente'
21
22     request.input('idExpediente', sql.Int, id)
23
24     const result = await request.query(query)
25
26     if (result.recordset.length === 0) {
27       return res.status(404).json({
28         mensaje: "Expediente no encontrado"
29       });
30     }
31
32     res.json(result.recordset[0]);
33
34   } catch (error) {
35     console.error(error);
36     res.status(500).json({
```