**Estructurar la funcionalidad con la siguiente información**

**Estructura en Árbol (Funcionalidad de consultas)**

pira-frontend/

├── public/

│ └── index.html → Entrada principal HTML (meta tags, favicon)

├── src/

│ ├── assets/ → Recursos estáticos (logo, íconos, etc.)

│ ├── components/

│ │ ├── Auth/

│ │ │ └── LoginForm.tsx → Componente de inicio de sesión

│ │ ├── Dashboard/

│ │ │ └── EstadisticasResumen.tsx → Vista de resumen gráfico (coincidencias, tareas, etc.)

│ │ ├── Coincidencias/

│ │ │ └── TablaCoincidencias.tsx → Renderiza coincidencias y su estado de revisión

│ │ ├── Filtros/

│ │ │ └── FiltroConLimpieza.tsx → Filtro reutilizable con persistencia local y botón limpiar

│ │ ├── Historial/

│ │ │ └── TablaHistorial.tsx → Muestra logs de auditoría, filtros y exportación

│ │ └── UI/

│ │ └── Sidebar.tsx → Navegación lateral con validación por rol

│ ├── hooks/

│ │ └── useAuth.ts → Manejador de sesión, token JWT, sliding expiration

│ ├── pages/

│ │ ├── Login.tsx → Vista pública de autenticación

│ │ ├── Coincidencias.tsx → Página principal para consultar resultados

│ │ ├── HistorialTareas.tsx → Página con tareas en segundo plano del usuario

│ │ ├── Auditoria.tsx → Página de validación manual por administradores

│ │ └── Ayuda.tsx → Wiki interna interactiva con opción exportar a PDF

│ ├── services/

│ │ └── api.ts → Axios configurado con token JWT, manejo de errores, CORS

│ ├── store/

│ │ └── authStore.ts → Estado global para sesión, usuario, roles

│ ├── utils/

│ │ └── exportHelpers.ts → Exportación Excel/PDF, impresión directa

│ ├── App.tsx → Definición de rutas, protección por roles

│ └── main.tsx → Punto de entrada de React

├── tailwind.config.ts → Configuración de Tailwind CSS

├── vite.config.ts → Configuración proxy para evitar CORS en desarrollo

└── package.json → Dependencias y scripts

**📘 Descripción Técnica y Funcional por Sección**

**🔐 Autenticación & Seguridad**

**LoginForm.tsx / useAuth.ts**

Autenticación vía API (/api/auth/login)

Almacena JWT en localStorage y renueva automáticamente (sliding expiration)

Expira y cierra sesión automáticamente tras 8 minutos de inactividad

Valida roles (admin, admin\_empresa, consultor) en cada vista protegida

**🧩 Coincidencias y Validaciones**

**Coincidencias.tsx / TablaCoincidencias.tsx**

Interactúa con: coincidencias, personas\_consultadas, observaciones\_validacion

Permite marcar como “Requiere Validación Manual”

Integra botón rápido para exportar a Excel y PDF

Usa iconos de alerta visual (Alto, Medio, Bajo)

**📊 Historial y Auditoría**

**HistorialTareas.tsx / TablaHistorial.tsx**

Carga tareas del usuario o globales si es administrador

Interactúa con tabla: historial\_tareas, usuarios, logs\_acciones

Filtra por estado, fecha, empresa

Indicadores de estado, botón para limpiar filtros, exportación

**Auditoria.tsx**

Muestra coincidencias marcadas como críticas

Permite cambiar su estado a “Revisada”, “Confirmada”

Guarda cambios en tabla logs\_validaciones

**📁 Wiki / Ayuda Interna**

**Ayuda.tsx**

Muestra ayuda interna para el sistema (formato markdown o HTML)

Exportación a PDF

Incluye barra de búsqueda

**🧠 Dashboard**

**EstadisticasResumen.tsx**

Usa recharts o chart.js

Resumen por empresa, SLA promedio, cantidad de alertas por tipo

Conexión con tareas, coincidencias, empresas

**🛡️ Características de Seguridad del Frontend**

| **Medida de Seguridad** | **Implementación** |
| --- | --- |
| 🔐 JWT Expirable | Tokens expiran tras 8 minutos o manualmente |
| 🔁 Renovación de token | Sliding expiration mientras el usuario esté activo |
| 🔐 Protección de Rutas | Validación de rol y sesión activa por componente y por backend |
| ❌ Bloqueo por intentos fallidos | Frontend detecta múltiples errores y restringe temporalmente login |
| 🔒 HTTPS | Requiere TLS 1.2+ en entorno de producción |
| 📦 CORS Seguro | Configurado en vite.config.ts con proxy para evitar errores locales |
| 🗃️ Persistencia controlada | localStorage solo para tokens y preferencias visuales, sin exponer datos |

**🗄️ Tablas con las que Interactúa el Frontend**

| **Tabla** | **Descripción y Uso** |
| --- | --- |
| Usuarios | Obtener datos del usuario y roles desde token JWT |
| Coincidencias | Resultado de consultas por documento/NIT |
| observaciones\_validacion | Validaciones manuales por el equipo |
| historial\_tareas | Procesos asincrónicos (consultas por lote) |
| Empresas | Para filtrar y restringir acceso según empresa |
| logs\_acciones | Bitácora interna, accesible por auditores o administradores |
| versiones\_bases | Información sobre versiones de bases y fechas de actualización |

📄 **TABLA: USUARIOS**

| **Campo** | **Tipo** | **Restricciones** | **Descripción** |
| --- | --- | --- | --- |
| Id | UUID | PRIMARY KEY, NOT NULL | Identificador único del usuario |
| Nombre | VARCHAR(100) | NOT NULL | Nombre completo del usuario |
| Email | VARCHAR(150) | UNIQUE, NOT NULL | Correo electrónico del usuario |
| password\_hash | VARCHAR(255) | NOT NULL | Contraseña cifrada del usuario |
| Rol | VARCHAR(30) | NOT NULL | Rol del usuario (admin, validador, consulta, etc.) |
| Activo | BOOLEAN | DEFAULT TRUE | Indica si el usuario está activo o no |
| empresa\_id | UUID | FOREIGN KEY → empresas.id, NULLABLE | Empresa asociada, si aplica |
| ultimo\_login | TIMESTAMP |  | Fecha y hora del último inicio de sesión |
| fecha\_creacion | TIMESTAMP | DEFAULT NOW() | Fecha de creación del usuario |
| token\_invalido | BOOLEAN | DEFAULT FALSE | Usado para invalidar sesiones activas |

**TABLA: COINCIDENCIAS**

| **Campo** | **Tipo** | **Restricciones** | **Descripción** |
| --- | --- | --- | --- |
| Id | UUID | PRIMARY KEY, NOT NULL | Identificador único de la coincidencia |
| registro\_id | UUID | NOT NULL | Identificador del registro consultado (fuente externa o base cargada) |
| base\_id | UUID | FOREIGN KEY → versiones\_bases.id, NOT NULL | Relacionado con la versión de la base que generó la coincidencia |
| tipo\_documento | VARCHAR(50) |  | Tipo de documento (CC, NIT, PASAPORTE, etc.) |
| Documento | VARCHAR(100) | INDEX, NOT NULL | Número de documento asociado a la coincidencia |
| nombre\_detectado | VARCHAR(255) |  | Nombre encontrado en la coincidencia |
| tipo\_alerta | VARCHAR(20) | DEFAULT 'medio' | Nivel de alerta: alto, medio, bajo |
| estado\_validacion | VARCHAR(20) | DEFAULT 'sin\_validar' | Estado actual: pendiente, en\_revision, confirmado, descartado |
| requiere\_revision | BOOLEAN | DEFAULT FALSE | Si la coincidencia requiere validación manual |
| Observación | TEXT |  | Observación del validador (si existe) |
| usuario\_validador | UUID | FOREIGN KEY → usuarios.id, NULLABLE | Usuario que realizó la validación |
| fecha\_detectada | TIMESTAMP | DEFAULT NOW() | Fecha en que se detectó la coincidencia |
| fecha\_validacion | TIMESTAMP | NULLABLE | Fecha de la validación manual |

📄 **TABLA: OBSERVACIONES\_VALIDACION**

| **Campo** | **Tipo** | **Restricciones** | **Descripción** |
| --- | --- | --- | --- |
| Id | UUID | PRIMARY KEY, NOT NULL | Identificador único de la observación |
| coincidencia\_id | UUID | FOREIGN KEY → coincidencias.id, NOT NULL | Referencia a la coincidencia asociada |
| usuario\_id | UUID | FOREIGN KEY → usuarios.id, NOT NULL | Usuario que realizó la observación |
| estado\_anterior | VARCHAR(20) |  | Estado anterior de la coincidencia |
| estado\_nuevo | VARCHAR(20) |  | Estado actualizado tras la observación (pendiente, confirmado, etc.) |
| Observación | TEXT |  | Texto explicativo de la observación |
| fecha\_observacion | TIMESTAMP | DEFAULT NOW() | Fecha y hora de creación de la observación |

📄 **TABLA: HISTORIAL\_TAREAS**

| **Campo** | **Tipo** | **Restricciones** | **Descripción** |
| --- | --- | --- | --- |
| Id | UUID | PRIMARY KEY, NOT NULL | Identificador único de la tarea |
| usuario\_id | UUID | FOREIGN KEY → usuarios.id, NOT NULL | Usuario que ejecutó la tarea |
| tipo\_tarea | VARCHAR(50) | NOT NULL | Tipo de tarea (ej. consulta, carga\_base, respaldo) |
| descripcion | TEXT |  | Descripción detallada de la tarea |
| Estado | VARCHAR(20) | DEFAULT 'pendiente' | Estado de la tarea (pendiente, en\_proceso, completada, fallida) |
| Progreso | INTEGER | DEFAULT 0 | Porcentaje de progreso (0-100) |
| fecha\_inicio | TIMESTAMP | DEFAULT NOW() | Momento en que se inició la tarea |
| fecha\_fin | TIMESTAMP |  | Fecha de finalización, si aplica |
| resultado | TEXT |  | Resultado textual (ej. “3 coincidencias encontradas”, errores, etc.) |
| Origen | VARCHAR(30) |  | Módulo o funcionalidad desde donde se originó |
| Detalles | JSONB |  | Campo opcional para incluir detalles estructurados (ej. archivos, filtros) |

🏢 **TABLA:** EMPRESAS

| **Campo** | **Tipo** | **Restricciones** | **Descripción** |
| --- | --- | --- | --- |
| Id | UUID | PRIMARY KEY, NOT NULL | Identificador único de la empresa |
| Nombre | VARCHAR(255) | UNIQUE, NOT NULL | Nombre legal de la empresa |
| Nit | VARCHAR(20) | UNIQUE, NOT NULL | Número de identificación tributaria (NIT) |
| Estado | VARCHAR(20) | DEFAULT 'activo' | Estado de la empresa (activo, inactivo, etc.) |
| correo\_contacto | VARCHAR(255) |  | Correo de contacto principal |
| Teléfono | VARCHAR(30) |  | Teléfono o celular |
| Dirección | TEXT |  | Dirección de la empresa |
| creado\_en | TIMESTAMP | DEFAULT NOW() | Fecha de creación del registro |
| actualizado\_en | TIMESTAMP | ON UPDATE CURRENT\_TIMESTAMP (si aplica) | Última actualización del registro |
| Representante | VARCHAR(255) |  | Nombre del representante legal |
| notificar\_alertas | BOOLEAN | DEFAULT TRUE | Indica si la empresa desea recibir notificaciones automáticas |

**📋 Tabla: logs\_acciones**

| **Campo** | **Tipo** | **Restricciones** | **Descripción** |
| --- | --- | --- | --- |
| Id | UUID | PRIMARY KEY, NOT NULL | Identificador único del log |
| usuario\_id | UUID | FOREIGN KEY → usuarios(id) | Usuario que ejecutó la acción |
| ip\_origen | VARCHAR(45) |  | Dirección IP desde la cual se hizo la acción |
| Acción | VARCHAR(255) | NOT NULL | Descripción corta de la acción (ej. "Eliminó coincidencia", "Inicio sesión") |
| Detalle | TEXT |  | Descripción técnica detallada del evento |
| Tipo | VARCHAR(50) | DEFAULT 'informativo' | Puede ser: informativo, critico, seguridad, configuracion, etc. |
| recurso\_afectado | VARCHAR(100) |  | Nombre del recurso afectado (ej. nombre de tabla o módulo) |
| Fecha | TIMESTAMP | DEFAULT NOW() | Fecha y hora exacta del evento |
| Estado | VARCHAR(20) | DEFAULT 'visible' | Visibilidad del log (visible, oculto, interno) |
| Resultado | VARCHAR(50) |  | Resultado del evento (éxito, fallo, bloqueado, etc.) |

📋 **Tabla: versiones\_bases**

| **Campo** | **Tipo** | **Restricciones** | **Descripción** |
| --- | --- | --- | --- |
| Id | UUID | PRIMARY KEY, NOT NULL | Identificador único de la versión |
| base\_id | UUID | FOREIGN KEY → bases\_datos(id) | Referencia a la base cargada |
| nombre\_base | VARCHAR(255) | NOT NULL | Nombre legible de la base (ej. “Sancionados\_UIAF”) |
| version\_num | INTEGER | DEFAULT 1, NOT NULL | Número de versión (incremental por base) |
| fecha\_carga | TIMESTAMP | DEFAULT NOW() | Fecha y hora en que se cargó esta versión |
| usuario\_id | UUID | FOREIGN KEY → usuarios(id) | Usuario que subió esta versión |
| Observaciones | TEXT |  | Detalles técnicos o aclaraciones sobre la versión |
| Disponible | BOOLEAN | DEFAULT TRUE | Si esta versión está disponible para consulta |
| hash\_integridad | VARCHAR(128) |  | Hash (ej. SHA-256) para validar integridad de la base |
| empresas\_autorizadas | TEXT[] |  | Lista de IDs de empresas que pueden consultar esta versión |

**MIGRACIÓN ALEMBIC (ALEMBIC/VERSIONS/XXXX\_VERSIONES\_BASES.PY)**

from alembic import op

import sqlalchemy as sa

import uuid

# Identificación

revision = 'xxxx\_versiones\_bases'

down\_revision = 'anterior\_revision\_id'

branch\_labels = None

depends\_on = None

def upgrade():

op.create\_table(

'versiones\_bases',

sa.Column('id', sa.UUID(), primary\_key=True, default=uuid.uuid4),

sa.Column('base\_id', sa.UUID(), sa.ForeignKey('bases\_datos.id', ondelete='CASCADE'), nullable=False),

sa.Column('nombre\_base', sa.String(length=255), nullable=False),

sa.Column('version\_num', sa.Integer(), nullable=False, server\_default='1'),

sa.Column('fecha\_carga', sa.TIMESTAMP(), server\_default=sa.text('CURRENT\_TIMESTAMP')),

sa.Column('usuario\_id', sa.UUID(), sa.ForeignKey('usuarios.id')),

sa.Column('observaciones', sa.Text(), nullable=True),

sa.Column('disponible', sa.Boolean(), server\_default='true'),

sa.Column('hash\_integridad', sa.String(length=128), nullable=True),

sa.Column('empresas\_autorizadas', sa.ARRAY(sa.Text()), nullable=True)

)

def downgrade():

op.drop\_table('versiones\_bases')

**Resumen técnico de la tabla versiones\_bases**

| **Aspecto** | **Detalle** |
| --- | --- |
| Control de versiones | Incremental por base (version\_num) |
| Seguridad | Hash de integridad + acceso por empresa |
| Trazabilidad | Usuario que cargó, fecha, observaciones |
| Relación con otras tablas | bases\_datos, usuarios, empresas |
| Disponibilidad controlada | Campo disponible para activar/desactivar versiones específicas |
| Consultas segmentadas | Soporte para asignar visibilidad por empresa (empresas\_autorizadas) |

✅ models/versiones\_bases.py

from sqlalchemy import Column, String, Integer, Boolean, ForeignKey, DateTime, Text, ARRAY

from sqlalchemy.dialects.postgresql import UUID

from sqlalchemy.sql import func

import uuid

from app.db.base\_class import Base # Asegúrate de importar tu clase Base

class VersionBase(Base):

\_\_tablename\_\_ = "versiones\_bases"

id = Column(UUID(as\_uuid=True), primary\_key=True, default=uuid.uuid4)

base\_id = Column(UUID(as\_uuid=True), ForeignKey("bases\_datos.id", ondelete="CASCADE"), nullable=False)

nombre\_base = Column(String(255), nullable=False)

version\_num = Column(Integer, nullable=False, default=1)

fecha\_carga = Column(DateTime(timezone=True), server\_default=func.now())

usuario\_id = Column(UUID(as\_uuid=True), ForeignKey("usuarios.id"), nullable=True)

observaciones = Column(Text, nullable=True)

disponible = Column(Boolean, default=True)

hash\_integridad = Column(String(128), nullable=True)

empresas\_autorizadas = Column(ARRAY(Text), nullable=True)

📌 Explicación Técnica

| **Campo** | **Descripción** |
| --- | --- |
| Id | Identificador único UUID |
| base\_id | Referencia a la base cargada (relación con bases\_datos) |
| nombre\_base | Nombre o alias asignado por el usuario |
| version\_num | Control incremental de versiones |
| fecha\_carga | Fecha y hora de la carga (automática) |
| usuario\_id | Usuario responsable de la carga (opcional) |
| Observaciones | Campo libre para notas u observaciones |
| Disponible | Permite desactivar una versión sin eliminarla |
| hash\_integridad | Validación de integridad del archivo subido |
| empresas\_autorizadas | Lista de empresas (por ID o nombre) autorizadas a consultar esta versión |

📁 schemas/versiones\_base.py

from pydantic import BaseModel, Field

from typing import Optional, List

from uuid import UUID

from datetime import datetime

class VersionBaseBase(BaseModel):

base\_id: UUID

nombre\_base: str

version\_num: int = Field(..., ge=1)

observaciones: Optional[str] = None

empresas\_autorizadas: Optional[List[str]] = None

disponible: Optional[bool] = True

class VersionBaseCreate(VersionBaseBase):

usuario\_id: Optional[UUID]

class VersionBaseUpdate(BaseModel):

nombre\_base: Optional[str]

observaciones: Optional[str]

disponible: Optional[bool]

empresas\_autorizadas: Optional[List[str]]

class VersionBaseOut(VersionBaseBase):

id: UUID

fecha\_carga: datetime

usuario\_id: Optional[UUID]

hash\_integridad: Optional[str]

class Config:

orm\_mode = True

✅ Explicación por esquema

| **Esquema** | **Uso** |
| --- | --- |
| VersionBaseBase | Esquema base para herencia |
| VersionBaseCreate | Para peticiones POST de nuevas versiones |
| VersionBaseUpdate | Para PATCH o PUT en actualizaciones parciales |
| VersionBaseOut | Para respuestas JSON completas desde la API (con orm\_mode = True) |

| **Área** | **Recomendación** |
| --- | --- |
| ✅ QA | Ejecutar pruebas funcionales completas por rol (Administrador, Validador, Consultor). |
| ✅ Seguridad | Revisar logs de accesos, vencimiento de tokens, intentos de login fallidos. |
| ✅ Rendimiento | Hacer pruebas de estrés con múltiples cargas concurrentes. |
| ✅ Respaldo | Simular una restauración completa a una fecha anterior. |
| ✅ UX | Validar usabilidad real con usuarios de prueba. |

**Código fuente**

public/index.html

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<meta name="author" content="PIRA RIESGOS SAS" />

<meta name="description" content="Sistema de consulta avanzada y verificación de datos para cumplimiento normativo y análisis de riesgo." />

<meta name="theme-color" content="#2563eb" />

<link rel="icon" href="/favicon.ico" />

<title>APP CONSULTAS PIRA</title>

</head>

<body class="bg-gray-100 text-gray-900">

<div id="root"></div>

<script type="module" src="/src/main.tsx"></script>

</body>

</html>

src/components/auth/LoginForm.tsx

import React, { useState } from "react";

import axios from "axios";

import { useNavigate } from "react-router-dom";

const LoginForm = () => {

const [email, setEmail] = useState("");

const [contrasena, setContrasena] = useState("");

const [error, setError] = useState("");

const navigate = useNavigate();

const manejarEnvio = async (e: React.FormEvent) => {

e.preventDefault();

setError("");

try {

const respuesta = await axios.post("/api/auth/login", {

email,

password: contrasena,

});

const token = respuesta.data.access\_token;

const rol = respuesta.data.rol;

localStorage.setItem("token", token);

localStorage.setItem("rol", rol);

localStorage.setItem("usuario\_email", email);

navigate("/panel");

} catch (err: any) {

console.error(err);

setError("Credenciales inválidas o error de conexión.");

}

};

return (

<div className="min-h-screen flex items-center justify-center bg-gradient-to-br from-blue-100 to-blue-300 p-4">

<form

onSubmit={manejarEnvio}

className="bg-white p-8 rounded-2xl shadow-lg w-full max-w-sm space-y-6"

>

<h2 className="text-2xl font-bold text-center text-blue-800">

Iniciar Sesión

</h2>

{error && <p className="text-red-600 text-sm text-center">{error}</p>}

<div>

<label className="block text-sm text-gray-700">Correo electrónico</label>

<input

type="email"

value={email}

onChange={(e) => setEmail(e.target.value)}

required

className="mt-1 w-full px-4 py-2 border rounded-xl focus:outline-none focus:ring-2 focus:ring-blue-400"

/>

</div>

<div>

<label className="block text-sm text-gray-700">Contraseña</label>

<input

type="password"

value={contrasena}

onChange={(e) => setContrasena(e.target.value)}

required

className="mt-1 w-full px-4 py-2 border rounded-xl focus:outline-none focus:ring-2 focus:ring-blue-400"

/>

</div>

<button

type="submit"

className="w-full bg-blue-600 hover:bg-blue-700 text-white font-semibold py-2 rounded-xl transition duration-200"

>

Ingresar

</button>

</form>

</div>

);

};

export default LoginForm;

src/components/LoginForm.tsx

import React, { useState } from "react";

import axios from "axios";

interface LoginFormProps {

onLoginSuccess: (token: string) => void;

}

const LoginForm: React.FC<LoginFormProps> = ({ onLoginSuccess }) => {

const [email, setEmail] = useState("");

const [contrasena, setContrasena] = useState("");

const [error, setError] = useState("");

const handleLogin = async (e: React.FormEvent) => {

e.preventDefault();

setError("");

try {

const response = await axios.post("/api/auth/login", {

email,

password: contrasena,

});

const token = response.data?.access\_token;

if (token) {

localStorage.setItem("token", token); // Guarda el token

onLoginSuccess(token);

} else {

setError("No se recibió el token del servidor.");

}

} catch (err: any) {

setError("Credenciales incorrectas o error de conexión.");

}

};

return (

<div className="max-w-md mx-auto p-8 bg-white shadow-xl rounded-2xl mt-10">

<h2 className="text-2xl font-bold mb-6 text-center">Iniciar sesión</h2>

{error && <div className="text-red-600 text-sm mb-4">{error}</div>}

<form onSubmit={handleLogin} className="space-y-4">

<div>

<label className="block mb-1 text-sm font-medium">Correo electrónico</label>

<input

type="email"

className="w-full p-2 border border-gray-300 rounded-md"

value={email}

onChange={(e) => setEmail(e.target.value)}

required

autoComplete="email"

/>

</div>

<div>

<label className="block mb-1 text-sm font-medium">Contraseña</label>

<input

type="password"

className="w-full p-2 border border-gray-300 rounded-md"

value={contrasena}

onChange={(e) => setContrasena(e.target.value)}

required

autoComplete="current-password"

/>

</div>

<button

type="submit"

className="w-full bg-blue-600 text-white py-2 rounded-md hover:bg-blue-700 transition"

>

Ingresar

</button>

</form>

</div>

);

};

export default LoginForm;

✅ src/components/EstadisticasResumen.tsx

import React, { useEffect, useState } from "react";

import axios from "axios";

import { Card, CardContent } from "@/components/ui/card";

import { PieChart, Pie, Cell, Tooltip, ResponsiveContainer } from "recharts";

const COLORS = ["#34D399", "#FBBF24", "#EF4444", "#3B82F6"];

const EstadisticasResumen: React.FC = () => {

const [resumen, setResumen] = useState({

tareas\_totales: 0,

tareas\_completadas: 0,

coincidencias: 0,

usuarios\_activos: 0,

});

useEffect(() => {

axios.get("/api/estadisticas/resumen").then((res) => {

setResumen(res.data);

});

}, []);

const dataChart = [

{ name: "Completadas", value: resumen.tareas\_completadas },

{ name: "Pendientes", value: resumen.tareas\_totales - resumen.tareas\_completadas },

];

return (

<div className="grid md:grid-cols-2 gap-6 p-6">

<Card>

<CardContent className="p-4">

<h3 className="text-lg font-semibold mb-2">Tareas</h3>

<p>Total: {resumen.tareas\_totales}</p>

<p>Completadas: {resumen.tareas\_completadas}</p>

<div className="h-48 mt-4">

<ResponsiveContainer width="100%" height="100%">

<PieChart>

<Pie

data={dataChart}

dataKey="value"

nameKey="name"

cx="50%"

cy="50%"

outerRadius={60}

label

>

{dataChart.map((entry, index) => (

<Cell key={`cell-${index}`} fill={COLORS[index % COLORS.length]} />

))}

</Pie>

<Tooltip />

</PieChart>

</ResponsiveContainer>

</div>

</CardContent>

</Card>

<Card>

<CardContent className="p-4">

<h3 className="text-lg font-semibold mb-2">Resumen General</h3>

<ul className="text-sm space-y-1">

<li>Coincidencias encontradas: <strong>{resumen.coincidencias}</strong></li>

<li>Usuarios activos: <strong>{resumen.usuarios\_activos}</strong></li>

</ul>

</CardContent>

</Card>

</div>

);

};

export default EstadisticasResumen;

TablaCoincidencias.tsx

// src/components/TablaCoincidencias.tsx

import React from "react";

import { Table, TableHeader, TableRow, TableHead, TableCell, TableBody } from "@/components/ui/table";

import { Badge } from "@/components/ui/badge";

import { Coincidencia } from "@/types";

interface Props {

coincidencias: Coincidencia[];

}

const TablaCoincidencias: React.FC<Props> = ({ coincidencias }) => {

return (

<div className="overflow-x-auto">

<Table>

<TableHeader>

<TableRow>

<TableHead>Documento</TableHead>

<TableHead>Nombre</TableHead>

<TableHead>Base Coincidente</TableHead>

<TableHead>Nivel</TableHead>

<TableHead>Estado</TableHead>

</TableRow>

</TableHeader>

<TableBody>

{coincidencias.map((c) => (

<TableRow key={c.id}>

<TableCell>{c.documento}</TableCell>

<TableCell>{c.nombre}</TableCell>

<TableCell>{c.base\_origen}</TableCell>

<TableCell>

<Badge

className={

c.nivel === "Alto"

? "bg-red-600"

: c.nivel === "Medio"

? "bg-yellow-500"

: "bg-green-600"

}

>

{c.nivel}

</Badge>

</TableCell>

<TableCell>{c.estado}</TableCell>

</TableRow>

))}

</TableBody>

</Table>

</div>

);

};

export default TablaCoincidencias;

FiltroConLimpieza.tsx

// src/components/FiltroConLimpieza.tsx

import React from "react";

import { Input } from "@/components/ui/input";

import { Button } from "@/components/ui/button";

import { XIcon } from "lucide-react";

interface Props {

filtros: { [clave: string]: string };

setFiltros: (val: { [clave: string]: string }) => void;

onBuscar: () => void;

}

const FiltroConLimpieza: React.FC<Props> = ({ filtros, setFiltros, onBuscar }) => {

const limpiarFiltros = () => {

const camposVacios = Object.fromEntries(Object.keys(filtros).map((k) => [k, ""]));

setFiltros(camposVacios);

onBuscar();

};

return (

<div className="flex flex-wrap items-end gap-2 p-2 bg-muted/50 rounded-xl">

<Input

placeholder="Buscar por nombre..."

value={filtros.nombre || ""}

onChange={(e) => setFiltros({ ...filtros, nombre: e.target.value })}

className="w-64"

/>

<Input

placeholder="Buscar por documento..."

value={filtros.documento || ""}

onChange={(e) => setFiltros({ ...filtros, documento: e.target.value })}

className="w-64"

/>

<Button onClick={onBuscar} variant="default">

Buscar

</Button>

<Button onClick={limpiarFiltros} variant="ghost" className="text-destructive">

<XIcon className="w-4 h-4 mr-1" />

Limpiar filtros

</Button>

</div>

);

};

export default FiltroConLimpieza;

TablaHistorial.tsx

// src/components/TablaHistorial.tsx

import React from "react";

import { Table, TableHeader, TableRow, TableHead, TableBody, TableCell } from "@/components/ui/table";

import { format } from "date-fns";

interface HistorialItem {

id: number;

fecha: string;

usuario: string;

accion: string;

tipo: string;

estado: string;

}

interface Props {

datos: HistorialItem[];

}

const TablaHistorial: React.FC<Props> = ({ datos }) => {

return (

<div className="overflow-auto rounded-xl border shadow-sm bg-white">

<Table>

<TableHeader>

<TableRow>

<TableHead>ID</TableHead>

<TableHead>Fecha</TableHead>

<TableHead>Usuario</TableHead>

<TableHead>Acción</TableHead>

<TableHead>Tipo</TableHead>

<TableHead>Estado</TableHead>

</TableRow>

</TableHeader>

<TableBody>

{datos.map((item) => (

<TableRow key={item.id}>

<TableCell>{item.id}</TableCell>

<TableCell>{format(new Date(item.fecha), "dd/MM/yyyy HH:mm")}</TableCell>

<TableCell>{item.usuario}</TableCell>

<TableCell>{item.accion}</TableCell>

<TableCell>{item.tipo}</TableCell>

<TableCell>{item.estado}</TableCell>

</TableRow>

))}

</TableBody>

</Table>

</div>

);

};

export default TablaHistorial;

Sidebar.tsx

// src/components/layout/Sidebar.tsx

import React from "react";

import { Link, useLocation } from "react-router-dom";

import {

LayoutDashboard,

Search,

Users,

FileText,

ShieldCheck,

Clock,

HelpCircle,

Settings,

LogOut,

Archive,

} from "lucide-react";

import useAuth from "@/hooks/useAuth";

const Sidebar = () => {

const { pathname } = useLocation();

const { logout, user } = useAuth();

const menu = [

{ to: "/panel", label: "Inicio", icon: <LayoutDashboard /> },

{ to: "/consultas", label: "Consultas", icon: <Search /> },

{ to: "/usuarios", label: "Usuarios", icon: <Users />, roles: ["admin"] },

{ to: "/bases", label: "Bases", icon: <FileText />, roles: ["admin"] },

{ to: "/coincidencias", label: "Coincidencias", icon: <ShieldCheck /> },

{ to: "/tareas", label: "Historial Tareas", icon: <Clock /> },

{ to: "/auditoria", label: "Auditoría", icon: <Archive />, roles: ["admin", "validador"] },

{ to: "/wiki", label: "Ayuda", icon: <HelpCircle /> },

{ to: "/configuracion", label: "Configuración", icon: <Settings />, roles: ["admin"] },

];

return (

<aside className="w-64 h-screen bg-gray-900 text-white flex flex-col justify-between">

<nav className="p-4">

<h2 className="text-xl font-bold mb-6">PIRA RIESGOS</h2>

<ul className="space-y-2">

{menu

.filter((item) => !item.roles || item.roles.includes(user?.rol || ""))

.map((item) => (

<li key={item.to}>

<Link

to={item.to}

className={`flex items-center gap-2 p-2 rounded hover:bg-gray-700 ${

pathname === item.to ? "bg-gray-800 font-semibold" : ""

}`}

>

{item.icon}

{item.label}

</Link>

</li>

))}

</ul>

</nav>

<div className="p-4 border-t border-gray-800">

<button

onClick={logout}

className="flex items-center gap-2 p-2 w-full rounded hover:bg-red-700"

>

<LogOut /> Cerrar sesión

</button>

</div>

</aside>

);

};

export default Sidebar;

useAuth.ts

// src/hooks/useAuth.ts

import { useEffect, useState } from "react";

import { useNavigate } from "react-router-dom";

import api from "@/services/api";

import { User } from "@/types";

const useAuth = () => {

const [user, setUser] = useState<User | null>(null);

const navigate = useNavigate();

const loadUser = async () => {

try {

const res = await api.get("/auth/me");

setUser(res.data);

} catch (err) {

logout();

}

};

const logout = () => {

localStorage.removeItem("token");

setUser(null);

navigate("/login");

};

const refreshToken = async () => {

try {

const res = await api.post("/auth/refresh");

localStorage.setItem("token", res.data.token);

} catch {

logout();

}

};

useEffect(() => {

const token = localStorage.getItem("token");

if (token) loadUser();

}, []);

return { user, logout, refreshToken };

};

export default useAuth;

Login.tsx

// src/pages/Login.tsx

import React, { useState } from "react";

import { useNavigate } from "react-router-dom";

import api from "@/services/api";

const Login = () => {

const navigate = useNavigate();

const [email, setEmail] = useState("");

const [password, setPassword] = useState("");

const [error, setError] = useState("");

const handleSubmit = async (e: React.FormEvent) => {

e.preventDefault();

try {

const res = await api.post("/auth/login", { email, password });

localStorage.setItem("token", res.data.token);

navigate("/");

} catch (err) {

setError("Credenciales incorrectas. Inténtalo de nuevo.");

}

};

return (

<div className="min-h-screen flex items-center justify-center bg-gray-100 p-6">

<div className="w-full max-w-md bg-white shadow-lg rounded-xl p-8">

<h2 className="text-2xl font-bold mb-6 text-center">Iniciar sesión</h2>

<form onSubmit={handleSubmit} className="space-y-4">

<div>

<label className="block text-sm font-medium">Correo</label>

<input

type="email"

value={email}

onChange={(e) => setEmail(e.target.value)}

required

className="w-full px-4 py-2 border rounded-xl"

/>

</div>

<div>

<label className="block text-sm font-medium">Contraseña</label>

<input

type="password"

value={password}

onChange={(e) => setPassword(e.target.value)}

required

className="w-full px-4 py-2 border rounded-xl"

/>

</div>

{error && <p className="text-red-500 text-sm">{error}</p>}

<button

type="submit"

className="w-full bg-blue-600 text-white py-2 rounded-xl hover:bg-blue-700"

>

Entrar

</button>

</form>

</div>

</div>

);

};

export default Login;

Coincidencias.tsx

// src/pages/Coincidencias.tsx

import React, { useEffect, useState } from "react";

import TablaCoincidencias from "@/components/TablaCoincidencias";

import FiltroConLimpieza from "@/components/FiltroConLimpieza";

import { getCoincidencias } from "@/services/api";

import { Coincidencia } from "@/types";

const Coincidencias = () => {

const [coincidencias, setCoincidencias] = useState<Coincidencia[]>([]);

const [filtros, setFiltros] = useState({ empresa: "", estado: "", fecha: "" });

const cargarCoincidencias = async () => {

const data = await getCoincidencias(filtros);

setCoincidencias(data);

};

useEffect(() => {

cargarCoincidencias();

}, [filtros]);

const limpiarFiltros = () => {

setFiltros({ empresa: "", estado: "", fecha: "" });

};

return (

<div className="p-6">

<h1 className="text-2xl font-bold mb-4">Coincidencias Detectadas</h1>

<FiltroConLimpieza filtros={filtros} setFiltros={setFiltros} onLimpiar={limpiarFiltros} />

<TablaCoincidencias data={coincidencias} />

</div>

);

};

export default Coincidencias;

HistorialTareas.tsx

// src/pages/HistorialTareas.tsx

import React, { useEffect, useState } from "react";

import TablaHistorial from "@/components/TablaHistorial";

import FiltroConLimpieza from "@/components/FiltroConLimpieza";

import { getHistorialTareas } from "@/services/api";

import { Tarea } from "@/types";

const HistorialTareas = () => {

const [tareas, setTareas] = useState<Tarea[]>([]);

const [filtros, setFiltros] = useState({ empresa: "", fecha: "", estado: "" });

const cargarTareas = async () => {

const data = await getHistorialTareas(filtros);

setTareas(data);

};

useEffect(() => {

cargarTareas();

}, [filtros]);

const limpiarFiltros = () => {

setFiltros({ empresa: "", fecha: "", estado: "" });

};

return (

<div className="p-6">

<h1 className="text-2xl font-bold mb-4">Historial de Tareas</h1>

<FiltroConLimpieza filtros={filtros} setFiltros={setFiltros} onLimpiar={limpiarFiltros} />

<TablaHistorial data={tareas} />

</div>

);

};

export default HistorialTareas;

Auditoria.tsx

// src/pages/Auditoria.tsx

import React, { useEffect, useState } from "react";

import { getAuditoriaEventos } from "@/services/api";

import TablaAuditoria from "@/components/TablaAuditoria";

import FiltroConLimpieza from "@/components/FiltroConLimpieza";

import { EventoAuditoria } from "@/types";

const Auditoria = () => {

const [eventos, setEventos] = useState<EventoAuditoria[]>([]);

const [filtros, setFiltros] = useState({ usuario: "", tipo: "", fecha: "" });

const cargarAuditoria = async () => {

const data = await getAuditoriaEventos(filtros);

setEventos(data);

};

useEffect(() => {

cargarAuditoria();

}, [filtros]);

const limpiarFiltros = () => {

setFiltros({ usuario: "", tipo: "", fecha: "" });

};

return (

<div className="p-6">

<h1 className="text-2xl font-bold mb-4">Registro de Auditoría</h1>

<FiltroConLimpieza filtros={filtros} setFiltros={setFiltros} onLimpiar={limpiarFiltros} />

<TablaAuditoria data={eventos} />

</div>

);

};

export default Auditoria;

Ayuda.tsx

// src/pages/Ayuda.tsx

import React from "react";

import WikiAyuda from "./wiki/WikiAyuda";

const Ayuda = () => {

return (

<div className="p-6">

<WikiAyuda />

</div>

);

};

export default Ayuda;

api.ts

// src/services/api.ts

import axios from "axios";

import { getToken } from "./authStore";

const API\_URL = import.meta.env.VITE\_API\_URL || "http://localhost:8000";

const api = axios.create({

baseURL: API\_URL,

headers: {

"Content-Type": "application/json",

},

});

// Interceptor para incluir token en cada solicitud

api.interceptors.request.use((config) => {

const token = getToken();

if (token) {

config.headers.Authorization = `Bearer ${token}`;

}

return config;

});

// Ejemplo: Obtener coincidencias con filtros

export const getCoincidencias = async (filtros: any) => {

const response = await api.post("/coincidencias/buscar", filtros);

return response.data;

};

// Obtener historial de tareas

export const getHistorialTareas = async (filtros: any) => {

const response = await api.post("/tareas/historial", filtros);

return response.data;

};

// Obtener eventos de auditoría

export const getAuditoriaEventos = async (filtros: any) => {

const response = await api.post("/auditoria/listar", filtros);

return response.data;

};

// Exportar coincidencias o tareas

export const exportarResultados = async (tipo: string, filtros: any) => {

const response = await api.post(`/exportar/${tipo}`, filtros, {

responseType: "blob",

});

return response.data;

};

export default api;

authStore.ts

// src/services/authStore.ts

const TOKEN\_KEY = "pira\_token";

export const setToken = (token: string) => {

localStorage.setItem(TOKEN\_KEY, token);

};

export const getToken = (): string | null => {

return localStorage.getItem(TOKEN\_KEY);

};

export const clearToken = () => {

localStorage.removeItem(TOKEN\_KEY);

};

export const isAuthenticated = (): boolean => {

const token = getToken();

return !!token;

};

exportHelpers.ts

// src/utils/exportHelpers.ts

import \* as XLSX from "xlsx";

import jsPDF from "jspdf";

import autoTable from "jspdf-autotable";

export const exportToExcel = (data: any[], fileName: string) => {

const ws = XLSX.utils.json\_to\_sheet(data);

const wb = XLSX.utils.book\_new();

XLSX.utils.book\_append\_sheet(wb, ws, "Datos");

XLSX.writeFile(wb, `${fileName}.xlsx`);

};

export const exportToPDF = (columns: string[], data: any[], fileName: string) => {

const doc = new jsPDF();

autoTable(doc, {

head: [columns],

body: data.map((row) => columns.map((col) => row[col] ?? "")),

});

doc.save(`${fileName}.pdf`);

};

App.tsx

// src/App.tsx

import React from "react";

import { BrowserRouter as Router, Routes, Route, Navigate } from "react-router-dom";

import { AuthProvider } from "./auth/authStore";

import Login from "./pages/Login";

import Coincidencias from "./pages/Coincidencias";

import HistorialTareas from "./pages/HistorialTareas";

import Auditoria from "./pages/Auditoria";

import Ayuda from "./pages/wiki/WikiAyuda";

import Sidebar from "./components/layout/Sidebar";

import useAuth from "./hooks/useAuth";

const ProtectedRoute = ({ children }: { children: JSX.Element }) => {

const { isAuthenticated } = useAuth();

return isAuthenticated ? children : <Navigate to="/login" replace />;

};

function App() {

return (

<AuthProvider>

<Router>

<div className="flex min-h-screen">

<Sidebar />

<main className="flex-1 bg-gray-100 p-4">

<Routes>

<Route path="/login" element={<Login />} />

<Route path="/" element={<ProtectedRoute><Coincidencias /></ProtectedRoute>} />

<Route path="/tareas" element={<ProtectedRoute><HistorialTareas /></ProtectedRoute>} />

<Route path="/auditoria" element={<ProtectedRoute><Auditoria /></ProtectedRoute>} />

<Route path="/ayuda" element={<ProtectedRoute><Ayuda /></ProtectedRoute>} />

<Route path="\*" element={<Navigate to="/" replace />} />

</Routes>

</main>

</div>

</Router>

</AuthProvider>

);

}

export default App;

main.tsx

// src/main.tsx

import React from "react";

import ReactDOM from "react-dom/client";

import App from "./App";

import "./index.css";

const root = ReactDOM.createRoot(document.getElementById("root") as HTMLElement);

root.render(

<React.StrictMode>

<App />

</React.StrictMode>

);

vite.config.ts

// vite.config.ts

import { defineConfig } from "vite";

import react from "@vitejs/plugin-react";

import path from "path";

// https://vitejs.dev/config/

export default defineConfig({

plugins: [react()],

resolve: {

alias: {

"@": path.resolve(\_\_dirname, "./src"),

},

},

server: {

port: 5173,

proxy: {

"/api": {

target: "http://localhost:8000", // Cambia según tu backend

changeOrigin: true,

secure: false,

},

},

},

});

package.json

{

"name": "pira-frontend",

"version": "1.0.0",

"private": true,

"scripts": {

"dev": "vite",

"build": "vite build",

"preview": "vite preview",

"lint": "eslint . --ext .ts,.tsx"

},

"dependencies": {

"@emotion/react": "^11.11.1",

"@emotion/styled": "^11.11.0",

"@tanstack/react-query": "^5.18.0",

"axios": "^1.6.8",

"classnames": "^2.5.1",

"lucide-react": "^0.310.0",

"react": "^18.2.0",

"react-dom": "^18.2.0",

"react-router-dom": "^6.22.3",

"xlsx": "^0.18.5",

"jspdf": "^2.5.1",

"tailwind-merge": "^2.2.1",

"zustand": "^4.5.2"

},

"devDependencies": {

"@types/react": "^18.2.37",

"@types/react-dom": "^18.2.15",

"@vitejs/plugin-react": "^4.2.1",

"autoprefixer": "^10.4.17",

"eslint": "^8.57.0",

"eslint-plugin-react": "^7.34.1",

"postcss": "^8.4.38",

"tailwindcss": "^3.4.3",

"typescript": "^5.4.2",

"vite": "^5.2.8"

}

}