DOCUMENTO DE ARQUITECTURA TECNICA

AT5 Audit Platform

Documentacion Arquitectonica de Nivel Doctoral para Sistema de Auditoria de Software

Campo	Valor
Documento	Arquitectura Tecnica AT5 Audit Platform
Version	1.0.0
Fecha	Enero 2026
Clasificacion	Documento Tecnico - Nivel PhD
Estandares	ISO/IEC/IEEE 42010, ISO 25010, TOGAF, C4 Model
Audiencia	Arquitectos de Software, Desarrolladores Senior, Auditores Tecnicos

PARTE I: FUNDAMENTOS ARQUITECTONICOS

1. Introduccion y Contexto

1.1 Proposito del Documento

Este documento de arquitectura tecnica presenta una descripcion exhaustiva y rigurosa de la arquitectura del sistema AT5 Audit Platform, siguiendo los mas altos estandares academicos e industriales. El documento esta disenado para servir como referencia definitiva para:

1. **Arquitectos de Software**: Comprension profunda de decisiones arquitectonicas y sus justificaciones
2. **Desarrolladores Senior**: Guia tecnica detallada para implementacion y mantenimiento
3. **Auditores de Sistemas**: Verificacion de cumplimiento con estandares de calidad
4. **Investigadores Academicos**: Base para estudios de caso en ingenieria de software

1.2 Alcance del Sistema

AT5 Audit Platform es una aplicacion web empresarial disenada para gestionar auditorias de software siguiendo estandares internacionales. El sistema permite:

Capacidad	Descripcion
Gestion de Sesiones	Creacion, configuracion y seguimiento de sesiones de auditoria
Ejecucion de Pruebas	Documentacion paso a paso de casos de prueba con evidencias
Recoleccion de Evidencias	Captura, almacenamiento y verificacion de evidencias digitales
Firmas Digitales	Sistema de firmas con verificacion criptografica
Generacion de Reportes	Exportacion en multiples formatos (PDF, JSON, DOCX)
Trazabilidad Completa	Registro inmutable de todas las acciones del sistema

1.3 Sistema Bajo Prueba (SUT)

El sistema esta especificamente disenado para auditar el **AT5 MCP System**, un sistema SCADA industrial con las siguientes caracteristicas:

```
AT5 MCP SYSTEM - COMPONENTES PRINCIPALES
========================================

+-- 7 Agentes IA Especializados
|   +-- Orchestrator Agent (coordinacion)
|   +-- Manual Agent (consultas RAG)
```

```
|   +-- Device Creator Agent (gestion dispositivos)
|   +-- Diagnostic Agent (diagnosticos)
|   +-- Monitoring Agent (monitoreo)
|   +-- Action Agent (control)
|   +-- Report Agent (reportes)
|
+-- 30+ Herramientas MCP
|   +-- Categoria PRJ: Proyectos (6 tools)
|   +-- Categoria DEV: Dispositivos (7 tools)
|   +-- Categoria SIG: Senales (5 tools)
|   +-- Categoria BAT: Batch (2 tools)
|   +-- Categoria TPL: Templates (3 tools)
|   +-- Categoria TRF: Trafico (3 tools)
|   +-- Categoria RPT: Reportes (4 tools)
|
+-- 8 Proveedores LLM
|   +-- OpenAI (GPT-4, GPT-4o)
|   +-- Anthropic Claude
|   +-- Google Gemini
|   +-- Mistral AI
|   +-- Ollama (local)
|   +-- DeepSeek
|   +-- KimiK2
|   +-- Grok (xAI)
|
+-- 4 Protocolos Industriales
    +-- IEC 60870-5-104
    +-- Modbus TCP/RTU
    +-- DNP3
    +-- IEC 61850
```

1.4 Estandares de Referencia

El desarrollo de este documento y del sistema sigue rigurosamente los siguientes estandares:

Estandar	Aplicacion
ISO/IEC/IEEE 42010:2022	Descripcion de arquitectura de sistemas
ISO/IEC 25010:2011	Modelo de calidad del producto software
ISO/IEC/IEEE 29119	Estandar de pruebas de software
IEEE 829	Documentacion de pruebas de software
TOGAF 9.2	Framework de arquitectura empresarial
C4 Model	Visualizacion de arquitectura de software
OWASP Top 10	Seguridad en aplicaciones web
ISTQB	Certificacion de pruebas de software

2. Vision Arquitectonica

2.1 Objetivos Arquitectonicos

La arquitectura de AT5 Audit Platform esta guiada por los siguientes objetivos estrategicos:

2.1.1 Objetivos Funcionales

ID	Objetivo	Prioridad	Metricas de Exito
OF-001	Gestion completa del ciclo de vida de auditorias	CRITICA	100% de casos de uso implementados
OF-002	Trazabilidad inmutable de todas las acciones	CRITICA	Hash chain verificable
OF-003	Soporte para 33 casos de prueba del plan AT5 MCP	ALTA	Cobertura completa
OF-004	Generacion de reportes en multiples formatos	ALTA	PDF, JSON, DOCX soportados
OF-005	Sistema de firmas digitales verificables	ALTA	Verificacion criptografica

2.1.2 Objetivos No Funcionales (Atributos de Calidad)

Atributo	Objetivo	Metrica
Rendimiento	Tiempo de respuesta < 2s para operaciones comunes	P95 latencia
Escalabilidad	Soporte para 100 usuarios concurrentes	Carga maxima
Disponibilidad	99.5% uptime	SLA mensual
Seguridad	Cumplimiento OWASP Top 10	0 vulnerabilidades criticas
Mantenibilidad	Cobertura de pruebas > 80%	Test coverage
Usabilidad	Tiempo de aprendizaje < 2 horas	User onboarding

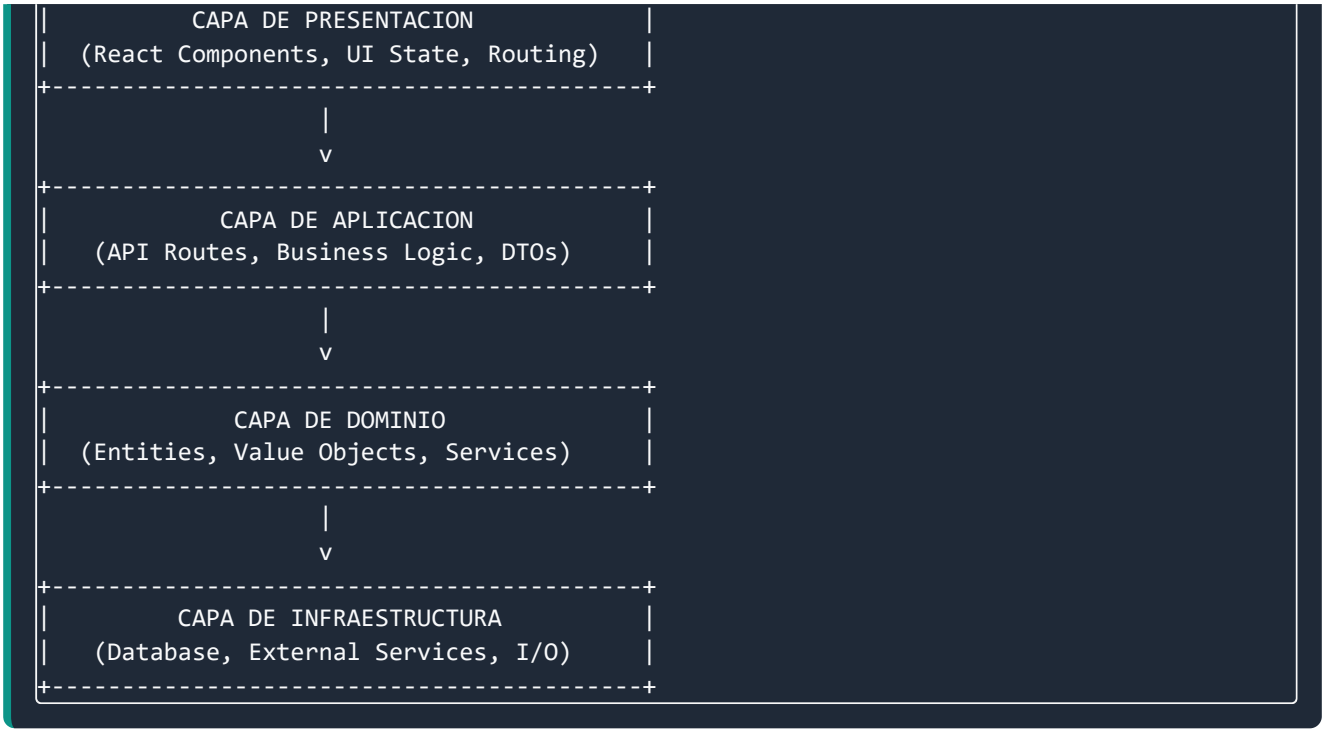
2.2 Principios Arquitectonicos

Los siguientes principios guian todas las decisiones arquitectonicas:

Principio 1: Separacion de Responsabilidades (SoC)

```
SEPARACION DE CAPAS
===================

+-----------------------------------------+
```



Principio 2: Inmutabilidad de Registros de Auditoria

Todo registro de auditoria es inmutable una vez creado. Se implementa mediante:

- Hash SHA-256 de cada registro
- Encadenamiento de hashes (blockchain-like)
- Timestamps no modificables

Principio 3: Seguridad por Diseno

- Autenticacion requerida para todas las operaciones
- Autorizacion basada en roles (RBAC)
- Validacion de entrada en todas las capas
- Sanitizacion de salida para prevenir XSS

Principio 4: Trazabilidad Completa

Cada accion en el sistema genera un registro que incluye:

- Usuario que realizo la accion
- Timestamp preciso
- Entidad afectada
- Valores antes/despues del cambio
- IP y User-Agent del cliente

2.3 Restricciones Arquitectonicas

Restriccion	Justificacion
Next.js 15 como framework	Requisito del proyecto, optimizado para React Server Components

SQLite como base de datos	Simplicidad de despliegue, portabilidad
TypeScript obligatorio	Type safety, mejor mantenibilidad
Prisma como ORM	Type-safe database access, migraciones
NextAuth.js para autenticacion	Solucion probada, extensible

3. Stakeholders y Concerns

3.1 Identificacion de Stakeholders

Stakeholder	Rol	Concerns Principales
Auditor Lider	Ejecuta y supervisa auditorias	Facilidad de uso, reportes completos
Auditor	Ejecuta casos de prueba	Eficiencia, documentacion clara
Revisor	Revisa y aprueba auditorias	Trazabilidad, verificabilidad
Administrador	Gestiona usuarios y configuracion	Seguridad, mantenimiento
Desarrollador	Mantiene y extiende el sistema	Codigo limpio, documentacion tecnica
Arquitecto	Define evolucion del sistema	Escalabilidad, integracion

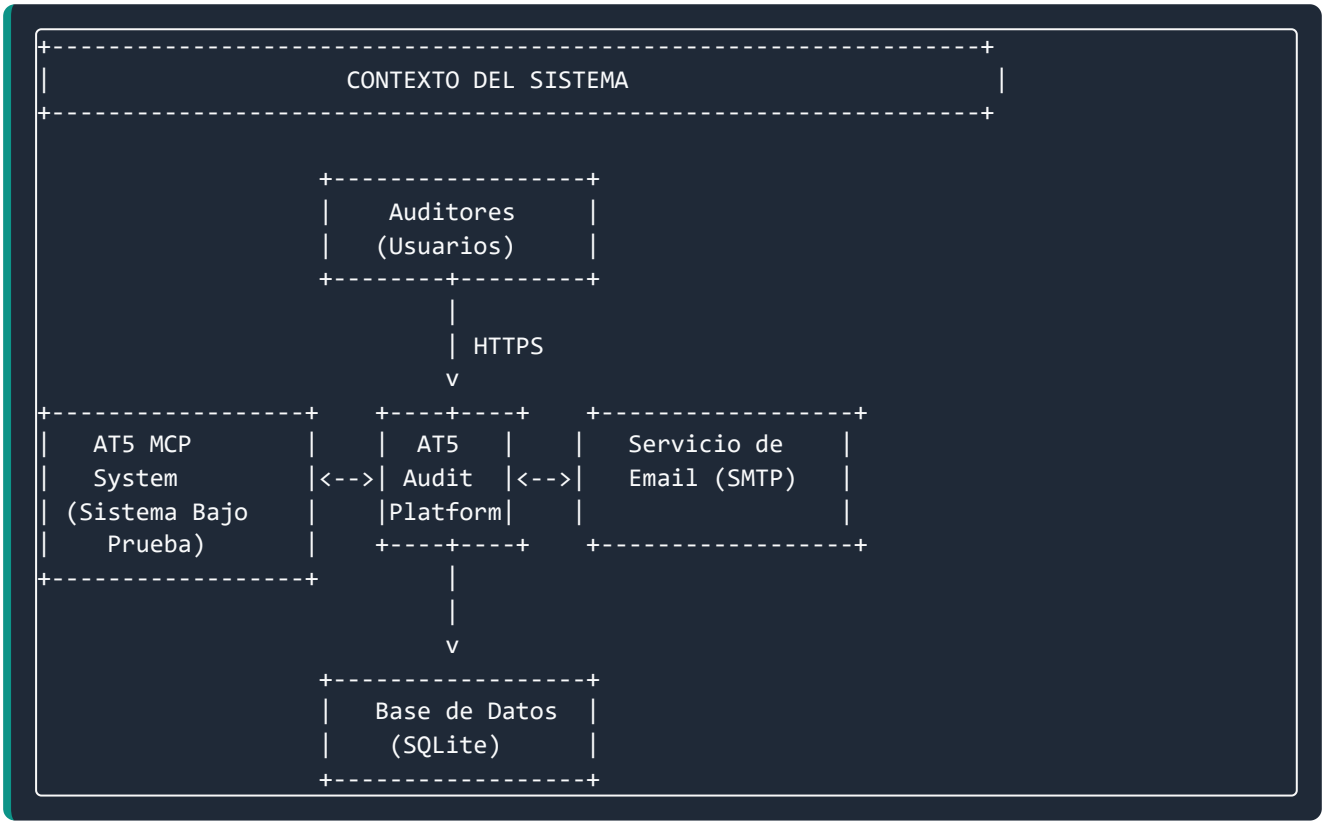
3.2 Matriz de Concerns por Stakeholder

	Auditor Lider	Auditor	Revisor	Admin	Dev	Arquitecto
Funcionalidad	★★★★★	★★★★★	★★★★☆	★★★☆☆	★★★☆☆	★★★★☆
Usabilidad	★★★★★	★★★★★	★★★★☆	★★★☆☆	★★☆☆☆	★★☆☆☆
Rendimiento	★★★☆☆	★★★★☆	★★★☆☆	★★★★☆	★★★★★	★★★★★
Seguridad	★★★★☆	★★★☆☆	★★★★★	★★★★★	★★★★★	★★★★★
Mantenibilidad	★☆☆☆☆	★☆☆☆☆	★☆☆☆☆	★★★☆☆	★★★★★	★★★★★
Escalabilidad	★★☆☆☆	★☆☆☆☆	★☆☆☆☆	★★★★☆	★★★★☆	★★★★★
Trazabilidad	★★★★★	★★★★☆	★★★★★	★★★★★	★★★☆☆	★★★★☆

PARTE II: VISTAS ARQUITECTONICAS

4. Vista de Contexto (C4 Level 1)

4.1 Diagrama de Contexto del Sistema



4.2 Descripcion de Actores Externos

4.2.1 Auditores (Usuarios)

Caracteristica	Descripcion
Tipo	Actor humano
Cantidad estimada	10-50 usuarios
Roles	Admin, Lead Auditor, Auditor, Reviewer, Viewer
Interaccion	Via navegador web (Chrome, Firefox, Edge)
Frecuencia	Diaria durante periodos de auditoria

4.2.2 AT5 MCP System (Sistema Bajo Prueba)

Caracteristica	Descripcion
Tipo	Sistema externo
Tecnologia	WPF (.NET Framework 4.8)
Interaccion	Manual (el auditor opera ambos sistemas)
Protocolos	N/A (no hay integracion directa)

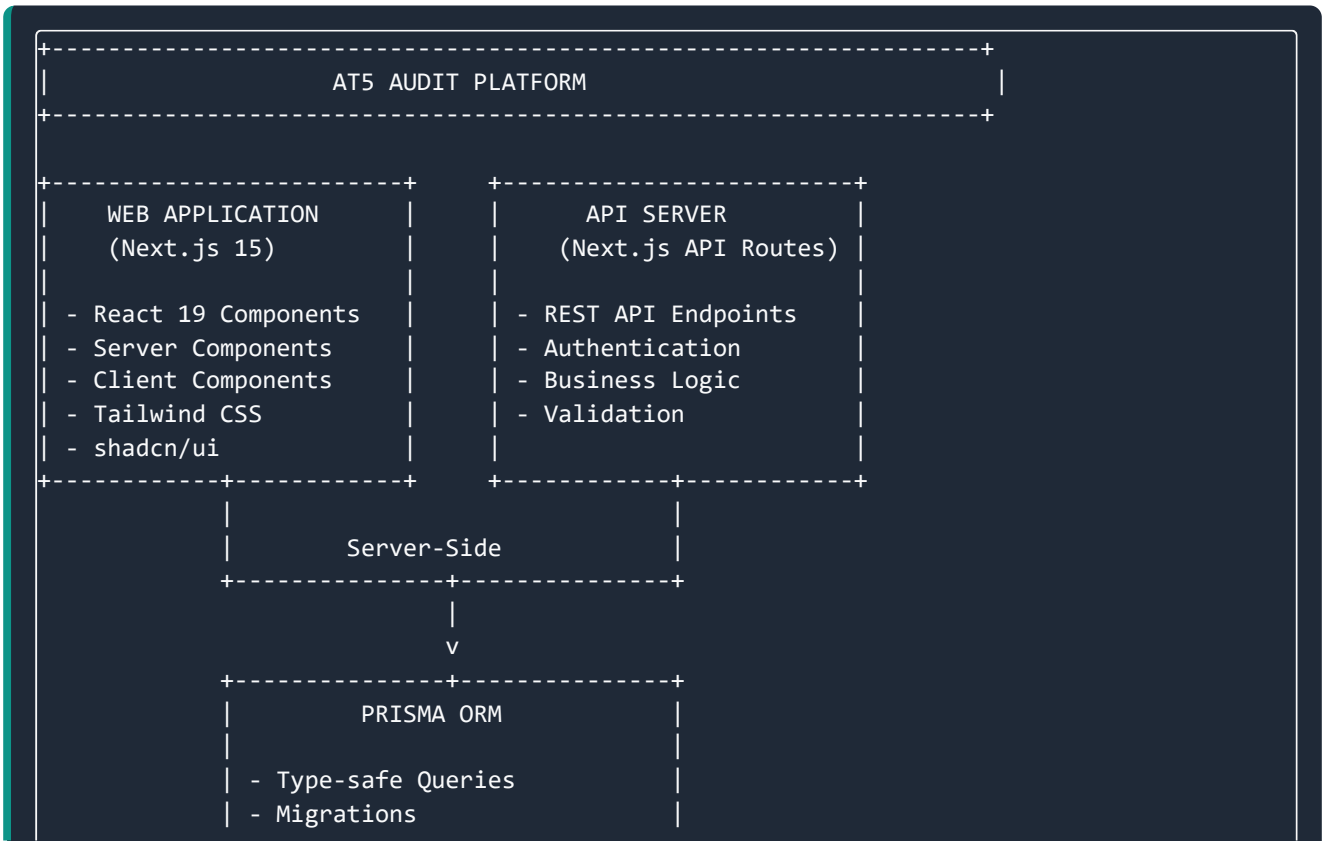
4.2.3 Servicio de Email (Futuro)

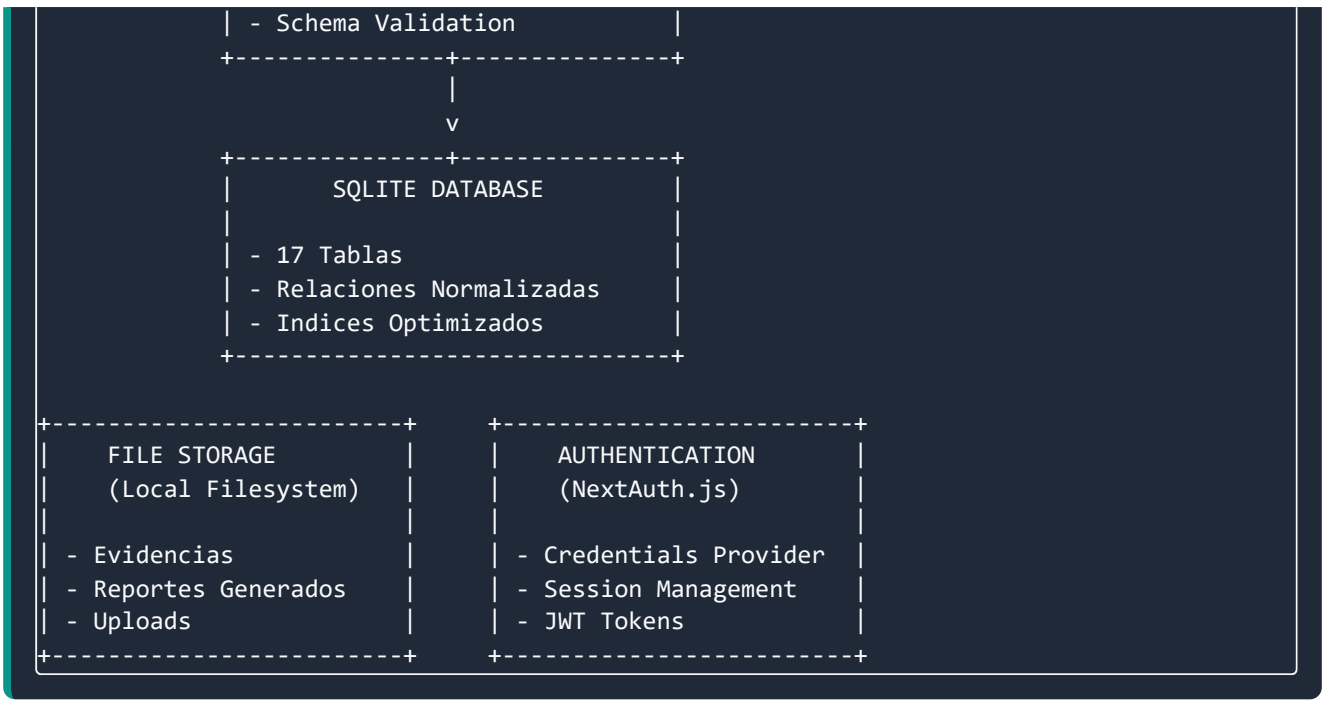
Caracteristica	Descripcion
Tipo	Servicio externo
Proposito	Notificaciones de auditoria
Estado	Planificado para v2.0

---

5. Vista de Contenedores (C4 Level 2)

5.1 Diagrama de Contenedores





5.2 Descripcion Detallada de Contenedores

5.2.1 Web Application Container

Aspecto	Detalle
Tecnologia	Next.js 15.1.0, React 19, TypeScript 5
Responsabilidad	Renderizado de UI, navegacion, estado del cliente
Patron	App Router con Server Components
Styling	Tailwind CSS 3.4, shadcn/ui components
Puerto	3000 (desarrollo), 80/443 (produccion)

Estructura de Directorios:

```
src/
├── app/                    # App Router (Next.js 15)
│   ├── (dashboard)/        # Grupo de rutas autenticadas
│   │   ├── dashboard/      # Panel principal
│   │   ├── sessions/       # Gestion de sesiones
│   │   │   ├── [id]/       # Detalle de sesion
│   │   │   │   ├── execute/  # Ejecucion de pruebas
│   │   │   │   └── page.tsx  # Vista de sesion
│   │   │   ├── new/        # Nueva sesion
│   │   │   └── page.tsx    # Lista de sesiones
│   │   ├── reports/        # Reportes
│   │   ├── metrics/        # Metricas
│   │   ├── audit-log/      # Log de auditoria
│   │   ├── settings/       # Configuracion
│   │   └── layout.tsx      # Layout del dashboard
```

```
│   ├── api/                  # API Routes
│   ├── login/                # Pagina de login
│   ├── layout.tsx            # Root layout
│   └── page.tsx              # Landing page
├── components/               # Componentes React
│   ├── ui/                   # Componentes base (shadcn)
│   ├── layout/               # Componentes de layout
│   ├── sessions/             # Componentes de sesiones
│   ├── dashboard/            # Componentes del dashboard
│   └── providers/            # Context providers
└── lib/                      # Utilidades y configuracion
    ├── auth.ts               # Configuracion NextAuth
    ├── prisma.ts             # Cliente Prisma
    ├── utils.ts              # Utilidades generales
    └── validations.ts        # Esquemas de validacion
```

5.2.2 API Server Container

Aspecto	Detalle
Tecnologia	Next.js API Routes, TypeScript
Responsabilidad	Logica de negocio, validacion, persistencia
Patron	REST API con route handlers
Autenticacion	JWT via NextAuth.js

Endpoints API:

Metodo	Endpoint	Descripcion
GET	`/api/sessions`	Listar sesiones de auditoria
POST	`/api/sessions`	Crear nueva sesion
GET	`/api/sessions/[id]`	Obtener detalle de sesion
PUT	`/api/sessions/[id]`	Actualizar sesion
DELETE	`/api/sessions/[id]`	Eliminar sesion
GET	`/api/executions`	Listar ejecuciones
POST	`/api/executions`	Crear/actualizar ejecucion
POST	`/api/evidence`	Subir evidencia
GET	`/api/evidence/[id]`	Obtener evidencia
POST	`/api/signatures`	Crear firma digital
GET	`/api/signatures/[id]/verify`	Verificar firma
POST	`/api/reports/generate`	Generar reporte
GET	`/api/dashboard`	Estadisticas del dashboard

GET	`/api/audit-logs`	Obtener logs de auditoria
POST	`/api/audit-logs/verify`	Verificar integridad
GET	`/api/test-plans`	Obtener planes de prueba
GET	`/api/users`	Listar usuarios
GET	`/api/export/session/[id]`	Exportar sesion

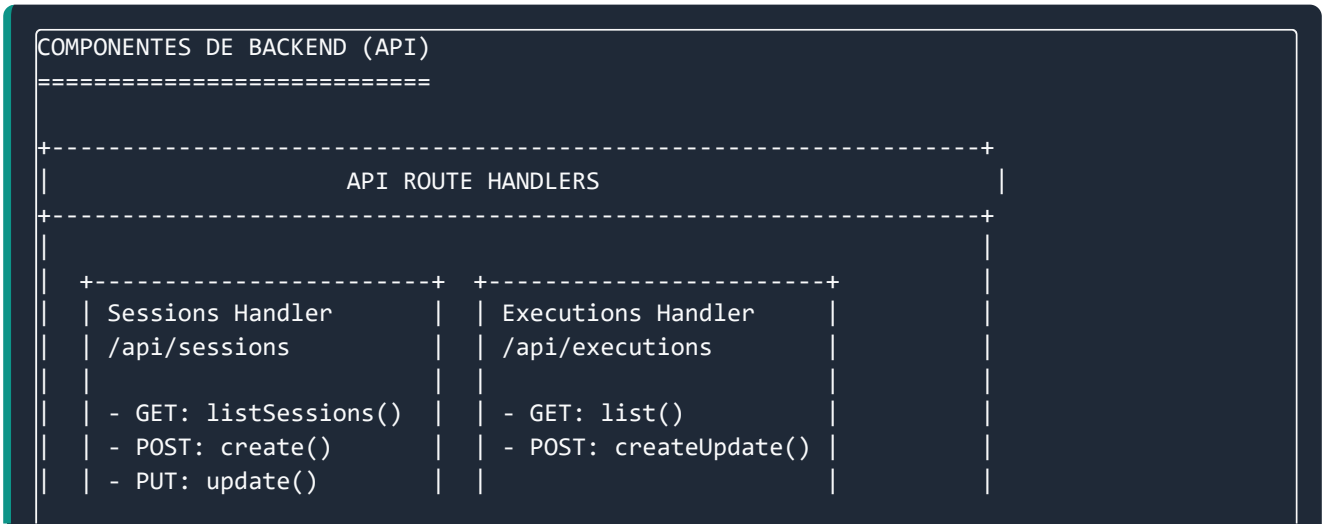
5.2.3 Database Container (SQLite + Prisma)

Aspecto	Detalle
Tecnologia	SQLite 3, Prisma ORM 6.2.1
Responsabilidad	Persistencia de datos
Ubicacion	`prisma/dev.db`
Tamano estimado	50MB - 500MB dependiendo de evidencias

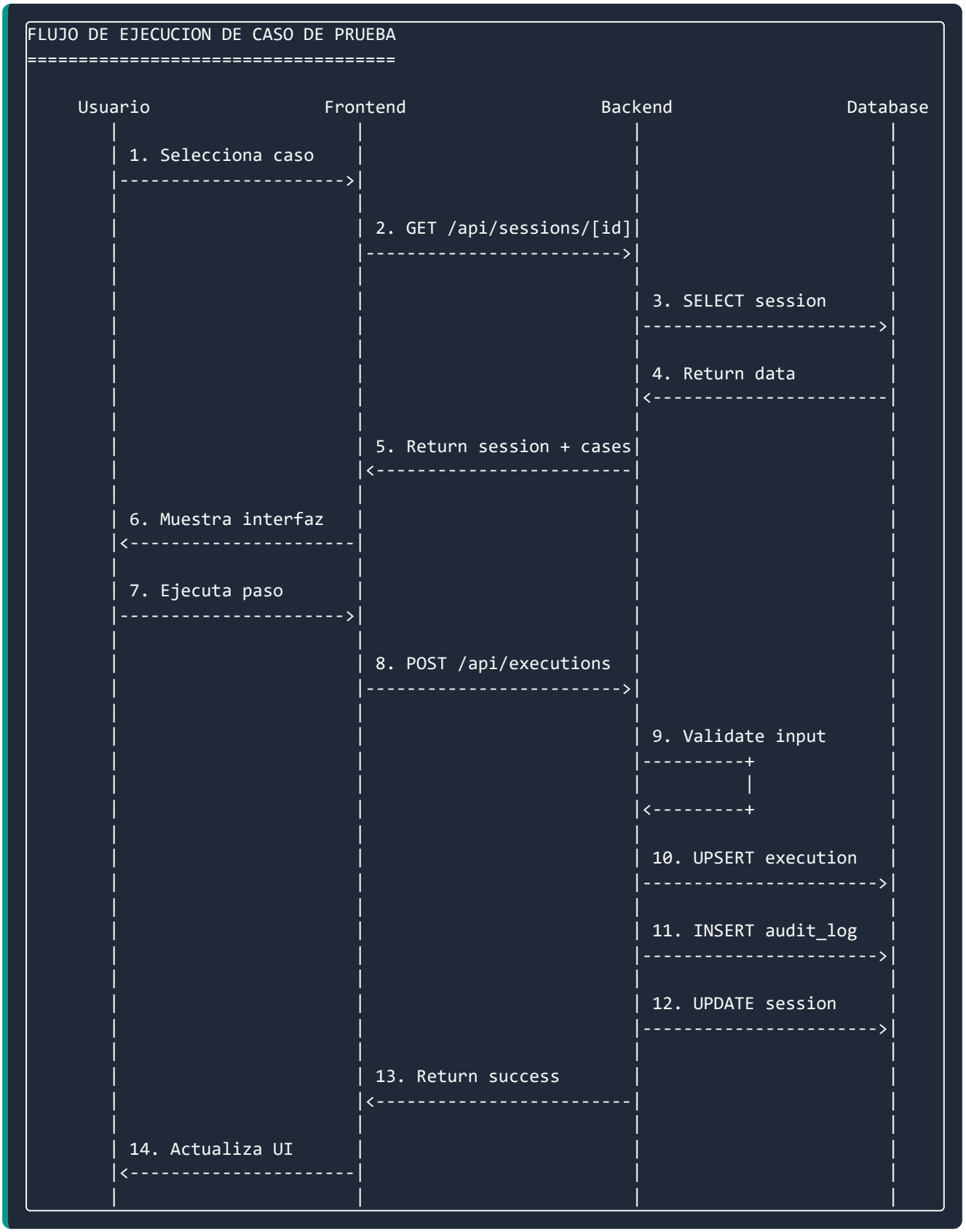
6. Vista de Componentes (C4 Level 3)

6.1 Componentes del Frontend

6.2 Componentes del Backend

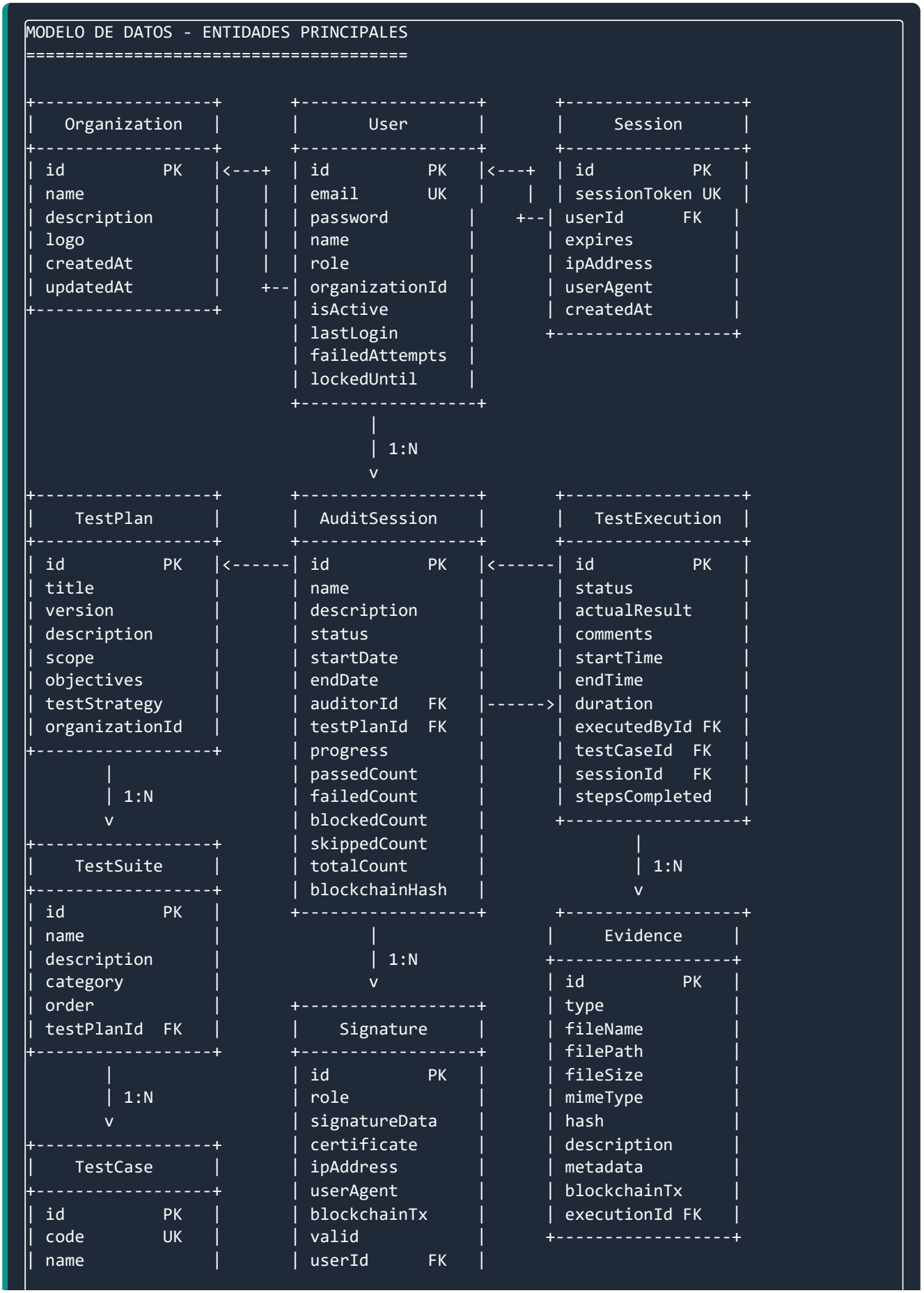


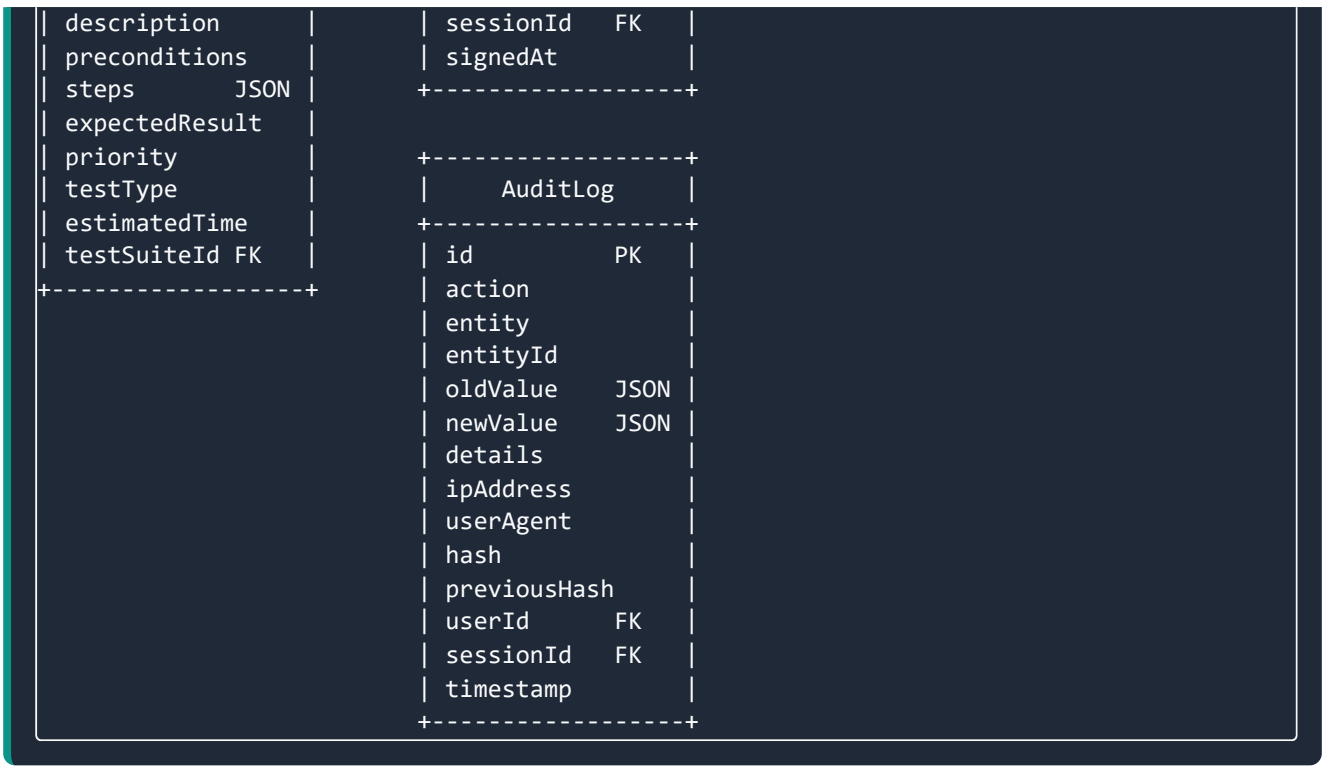
6.3 Interaccion entre Componentes



7. Vista de Datos

7.1 Modelo Entidad-Relacion





7.2 Descripcion de Entidades

7.2.1 Organization

Representa una organizacion que utiliza el sistema.

Campo	Tipo	Descripcion
id	String (CUID)	Identificador unico
name	String	Nombre de la organizacion
description	String?	Descripcion opcional
logo	String?	URL del logo
createdAt	DateTime	Fecha de creacion
updatedAt	DateTime	Fecha de ultima actualizacion

7.2.2 User

Representa un usuario del sistema con sus credenciales y rol.

Campo	Tipo	Descripcion
id	String (CUID)	Identificador unico
email	String	Email unico del usuario
emailVerified	DateTime?	Fecha de verificacion del email

password	String	Password hasheado con bcrypt
name	String	Nombre completo
avatar	String?	URL del avatar
role	UserRole	Rol: ADMIN, LEAD_AUDITOR, AUDITOR, REVIEWER, VIEWER
isActive	Boolean	Si el usuario esta activo
lastLogin	DateTime?	Ultimo inicio de sesion
failedAttempts	Int	Intentos fallidos de login
lockedUntil	DateTime?	Bloqueado hasta esta fecha
organizationId	String	FK a Organization

7.2.3 TestPlan

Plan de pruebas basado en ISO/IEC/IEEE 29119-3.

Campo	Tipo	Descripcion
id	String (CUID)	Identificador unico
title	String	Titulo del plan
version	String	Version (semver)
description	String?	Descripcion del plan
scope	String?	Alcance de las pruebas
objectives	String?	Objetivos de las pruebas
testStrategy	String?	Estrategia de pruebas
entryExitCriteria	String?	Criterios de entrada/salida
organizationId	String	FK a Organization

7.2.4 TestSuite

Agrupacion logica de casos de prueba.

Campo	Tipo	Descripcion
id	String (CUID)	Identificador unico
name	String	Nombre de la suite (ej: PRJ, DEV)
description	String?	Descripcion
category	String	Categoria funcional
order	Int	Orden de ejecucion

testPlanId	String	FK a TestPlan

7.2.5 TestCase

Caso de prueba individual con pasos y resultado esperado.

Campo	Tipo	Descripcion
id	String (CUID)	Identificador unico
code	String	Codigo unico (ej: TC-PRJ-001)
name	String	Nombre del caso
description	String?	Descripcion
preconditions	String?	Precondiciones
steps	String (JSON)	Array de pasos de ejecucion
expectedResult	String	Resultado esperado
priority	Priority	CRITICAL, HIGH, MEDIUM, LOW
testType	TestType	FUNCTIONAL, INTEGRATION, etc.
estimatedTime	Int?	Tiempo estimado en minutos
testSuiteId	String	FK a TestSuite

7.2.6 AuditSession

Sesion de auditoria que agrupa ejecuciones.

Campo	Tipo	Descripcion
id	String (CUID)	Identificador unico
name	String	Nombre de la sesion
description	String?	Descripcion
status	SessionStatus	DRAFT, IN_PROGRESS, REVIEW, APPROVED, REJECTED, ARCHIVED
startDate	DateTime?	Fecha de inicio
endDate	DateTime?	Fecha de finalizacion
auditorId	String	FK a User (auditor responsable)
testPlanId	String	FK a TestPlan
progress	Float	Porcentaje de progreso (0-100)
passedCount	Int	Cantidad de casos exitosos
failedCount	Int	Cantidad de casos fallidos

blockedCount	Int	Cantidad de casos bloqueados
skippedCount	Int	Cantidad de casos omitidos
totalCount	Int	Total de casos
blockchainHash	String?	Hash para verificacion

7.2.7 TestExecution

Ejecucion de un caso de prueba especifico.

Campo	Tipo	Descripcion
id	String (CUID)	Identificador unico
status	ExecutionStatus	NOT_STARTED, IN_PROGRESS, PASSED, FAILED, BLOCKED, SKIPPED
actualResult	String?	Resultado real obtenido
comments	String?	Comentarios del auditor
startTime	DateTime?	Hora de inicio
endTime	DateTime?	Hora de finalizacion
duration	Int?	Duracion en segundos
executedById	String	FK a User
testCaseId	String	FK a TestCase
sessionId	String	FK a AuditSession
stepsCompleted	String?	JSON array de pasos completados

7.2.8 Evidence

Evidencia adjunta a una ejecucion.

Campo	Tipo	Descripcion
id	String (CUID)	Identificador unico
type	EvidenceType	SCREENSHOT, LOG, PDF, VIDEO, JSON, OTHER
fileName	String	Nombre del archivo
filePath	String	Ruta en el filesystem
fileSize	Int	Tamano en bytes
mimeType	String	Tipo MIME
hash	String	SHA-256 del archivo
description	String?	Descripcion

metadata	String?	Metadata adicional (JSON)
blockchainTx	String?	Transaccion blockchain
executionId	String	FK a TestExecution

7.2.9 Signature

Firma digital de una sesion de auditoria.

Campo	Tipo	Descripcion
id	String (CUID)	Identificador unico
role	String	Rol del firmante
signatureData	String?	Firma visual (Base64)
certificate	String?	Certificado digital
ipAddress	String?	IP del firmante
userAgent	String?	User-Agent
blockchainTx	String?	Transaccion blockchain
valid	Boolean	Si la firma es valida
userId	String	FK a User
sessionId	String	FK a AuditSession
signedAt	DateTime	Fecha de firma

7.2.10 AuditLog

Registro inmutable de acciones del sistema.

Campo	Tipo	Descripcion
id	String (CUID)	Identificador unico
action	String	Tipo de accion (CREATE, UPDATE, DELETE, etc.)
entity	String	Entidad afectada
entityId	String	ID de la entidad
oldValue	String?	Valor anterior (JSON)
newValue	String?	Valor nuevo (JSON)
details	String?	Descripcion legible
ipAddress	String?	IP del usuario
userAgent	String?	User-Agent

hash	String?	SHA-256 del registro
previousHash	String?	Hash del registro anterior
userId	String	FK a User
sessionId	String?	FK a AuditSession
timestamp	DateTime	Timestamp del evento

7.3 Indices y Optimizaciones

```
-- Indices principales para optimizacion de consultas

-- AuditLog: Consultas frecuentes por usuario, sesion y timestamp
CREATE INDEX idx_audit_log_user ON AuditLog(userId);
CREATE INDEX idx_audit_log_session ON AuditLog(sessionId);
CREATE INDEX idx_audit_log_entity ON AuditLog(entity, entityId);
CREATE INDEX idx_audit_log_timestamp ON AuditLog(timestamp);

-- Unique constraints
CREATE UNIQUE INDEX idx_user_email ON User(email);
CREATE UNIQUE INDEX idx_testcase_code ON TestCase(code);
CREATE UNIQUE INDEX idx_session_token ON Session(sessionToken);
```

8. Vista de Seguridad

8.1 Modelo de Seguridad

```
ARQUITECTURA DE SEGURIDAD
=========================

+-------------------------------------------------------------------+
|                      CAPA DE TRANSPORTE                           |
|                         (HTTPS/TLS)                               |
+-------------------------------------------------------------------+
                              |
                              v
+-------------------------------------------------------------------+
|                    CAPA DE AUTENTICACION                          |
|                       (NextAuth.js)                               |
|                                                                   |
|  +------------------+  +------------------+  +------------------+ |
|  | Credentials      |  | Session Mgmt     |  | JWT Tokens       | |
|  | Provider         |  |                  |  |                  | |
|  | - bcrypt hash    |  | - Server-side    |  | - Signed tokens  | |
|  | - Rate limiting  |  | - Secure cookie  |  | - Expiration     | |
```

8.2 Roles y Permisos (RBAC)

Rol	Descripcion	Permisos
ADMIN	Administrador del sistema	Todos los permisos, gestion de usuarios

LEAD_AUDITOR	Auditor lider	Crear sesiones, asignar, firmar, aprobar
AUDITOR	Auditor ejecutor	Ejecutar pruebas, subir evidencias
REVIEWER	Revisor	Ver sesiones, firmar, comentar
VIEWER	Solo lectura	Ver sesiones y reportes

8.3 Matriz de Permisos Detallada

Accion	ADMIN	LEAD_AUDITOR	AUDITOR	REVIEWER	VIEWER
Ver Dashboard	✓	✓	✓	✓	✓
Ver Sesiones	✓	✓	✓	✓	✓
Crear Sesion	✓	✓	✗	✗	✗
Editar Sesion	✓	✓	Solo asignadas	✗	✗
Eliminar Sesion	✓	✓	✗	✗	✗
Ejecutar Pruebas	✓	✓	✓	✗	✗
Subir Evidencias	✓	✓	✓	✗	✗
Firmar Sesion	✓	✓	✗	✓	✗
Aprobar/Rechazar	✓	✓	✗	✗	✗
Ver Reportes	✓	✓	✓	✓	✓
Generar Reportes	✓	✓	✓	✓	✗
Ver Audit Log	✓	✓	✓	✓	✓
Gestionar Usuarios	✓	✗	✗	✗	✗
Configuracion	✓	✗	✗	✗	✗

8.4 Proteccion contra Vulnerabilidades OWASP

Vulnerabilidad	Mitigacion Implementada
A01: Broken Access Control	RBAC con verificacion en cada endpoint, middleware de autenticacion
A02: Cryptographic Failures	bcrypt para passwords, SHA-256 para hashes, HTTPS obligatorio
A03: Injection	Prisma ORM (parameterized queries), validacion Zod
A04: Insecure Design	Arquitectura segura por diseno, principio de minimo privilegio
A05: Security Misconfiguration	Headers de seguridad, variables de entorno para secretos
A06: Vulnerable Components	Dependencias actualizadas, npm audit regular

A07: Authentication Failures	NextAuth.js, rate limiting, bloqueo de cuenta
A08: Data Integrity Failures	Hash chain en audit log, verificacion de firmas
A09: Security Logging	Audit log completo con hash chain
A10: SSRF	Validacion de URLs, no hay fetch a URLs externas

8.5 Implementacion de Autenticacion

```
// Configuracion de NextAuth.js (lib/auth.ts)

import NextAuth from "next-auth"
import CredentialsProvider from "next-auth/providers/credentials"
import bcrypt from "bcryptjs"
import { prisma } from "./prisma"

export const authOptions = {
  providers: [
    CredentialsProvider({
      name: "Credentials",
      credentials: {
        email: { label: "Email", type: "email" },
        password: { label: "Password", type: "password" }
      },
      async authorize(credentials) {
        // 1. Validar credenciales
        if (!credentials?.email || !credentials?.password) {
          throw new Error("Credenciales requeridas")
        }

        // 2. Buscar usuario
        const user = await prisma.user.findUnique({
          where: { email: credentials.email }
        })

        if (!user) {
          throw new Error("Usuario no encontrado")
        }

        // 3. Verificar bloqueo de cuenta
        if (user.lockedUntil && user.lockedUntil > new Date()) {
          throw new Error("Cuenta bloqueada temporalmente")
        }

        // 4. Verificar password
        const isValid = await bcrypt.compare(
          credentials.password,
          user.password
        )

        if (!isValid) {
          // Incrementar intentos fallidos
          await prisma.user.update({
            where: { id: user.id },
```

```
            data: {
              failedAttempts: { increment: 1 },
              lockedUntil: user.failedAttempts >= 4
                ? new Date(Date.now() + 15 * 60 * 1000) // 15 min
                : null
            }
          })
          throw new Error("Credenciales invalidas")
        }

        // 5. Reset intentos y actualizar ultimo login
        await prisma.user.update({
          where: { id: user.id },
          data: {
            failedAttempts: 0,
            lockedUntil: null,
            lastLogin: new Date()
          }
        })

        return {
          id: user.id,
          email: user.email,
          name: user.name,
          role: user.role,
          organizationId: user.organizationId
        }
      }
    })
  ],
  callbacks: {
    async jwt({ token, user }) {
      if (user) {
        token.id = user.id
        token.role = user.role
        token.organizationId = user.organizationId
      }
      return token
    },
    async session({ session, token }) {
      if (session.user) {
        session.user.id = token.id
        session.user.role = token.role
        session.user.organizationId = token.organizationId
      }
      return session
    }
  },
  pages: {
    signIn: "/login",
    error: "/login"
  },
  session: {
    strategy: "jwt",
    maxAge: 8 * 60 * 60 // 8 horas
  }
}
```

8.6 Sistema de Audit Log con Hash Chain

```typescript
// Implementacion de Audit Log (lib/audit-log.ts)

import crypto from 'crypto'
import { prisma } from './prisma'

interface AuditLogEntry {
  action: string
  entity: string
  entityId: string
  oldValue?: any
  newValue?: any
  details?: string
  userId: string
  sessionId?: string
  ipAddress?: string
  userAgent?: string
}

export async function createAuditLog(entry: AuditLogEntry) {
  // 1. Obtener el hash del registro anterior
  const previousLog = await prisma.auditLog.findFirst({
    orderBy: { timestamp: 'desc' },
    select: { hash: true }
  })

  const previousHash = previousLog?.hash || 'GENESIS'

  // 2. Crear el contenido a hashear
  const content = JSON.stringify({
    ...entry,
    previousHash,
    timestamp: new Date().toISOString()
  })

  // 3. Calcular hash SHA-256
  const hash = crypto
    .createHash('sha256')
    .update(content)
    .digest('hex')

  // 4. Crear el registro
  return prisma.auditLog.create({
    data: {
      action: entry.action,
      entity: entry.entity,
      entityId: entry.entityId,
      oldValue: entry.oldValue ? JSON.stringify(entry.oldValue) : null,
      newValue: entry.newValue ? JSON.stringify(entry.newValue) : null,
      details: entry.details,
      userId: entry.userId,
      sessionId: entry.sessionId,
      ipAddress: entry.ipAddress,
      userAgent: entry.userAgent,
      hash,
```

```
      previousHash,
      timestamp: new Date()
    }
  })
}

export async function verifyAuditLogIntegrity(): Promise<{
  isValid: boolean
  errors: string[]
}> {
  const logs = await prisma.auditLog.findMany({
    orderBy: { timestamp: 'asc' }
  })

  const errors: string[] = []
  let previousHash = 'GENESIS'

  for (const log of logs) {
    // Verificar que previousHash coincide
    if (log.previousHash !== previousHash) {
      errors.push(
        `Log ${log.id}: previousHash mismatch. ` +
        `Expected: ${previousHash}, Got: ${log.previousHash}`
      )
    }

    // Recalcular hash
    const content = JSON.stringify({
      action: log.action,
      entity: log.entity,
      entityId: log.entityId,
      oldValue: log.oldValue,
      newValue: log.newValue,
      details: log.details,
      userId: log.userId,
      sessionId: log.sessionId,
      ipAddress: log.ipAddress,
      userAgent: log.userAgent,
      previousHash: log.previousHash,
      timestamp: log.timestamp.toISOString()
    })

    const expectedHash = crypto
      .createHash('sha256')
      .update(content)
      .digest('hex')

    if (log.hash !== expectedHash) {
      errors.push(
        `Log ${log.id}: hash mismatch. ` +
        `Expected: ${expectedHash}, Got: ${log.hash}`
      )
    }

    previousHash = log.hash || ''
  }

  return {
```

```
    isValid: errors.length === 0,
    errors
  }
}
```

PARTE III: PATRONES Y DECISIONES ARQUITECTONICAS

9. Patrones Arquitectonicos Aplicados

9.1 App Router Pattern (Next.js 15)

```
ESTRUCTURA DEL APP ROUTER
=========================

app/
├── (dashboard)/           # Route Group (no afecta URL)
│   ├── layout.tsx         # Layout compartido con autenticacion
│   ├── dashboard/
│   │   └── page.tsx       # /dashboard
│   ├── sessions/
│   │   ├── page.tsx       # /sessions
│   │   ├── new/
│   │   │   └── page.tsx   # /sessions/new
│   │   └── [id]/
│   │       ├── page.tsx   # /sessions/:id
│   │       └── execute/
│   │           └── page.tsx # /sessions/:id/execute
│   └── ...
├── api/                   # API Routes
│   ├── auth/
│   │   └── [...nextauth]/
│   │       └── route.ts   # NextAuth handler
│   ├── sessions/
│   │   ├── route.ts       # GET, POST /api/sessions
│   │   └── [id]/
│   │       └── route.ts   # GET, PUT, DELETE /api/sessions/:id
│   └── ...
├── login/
│   └── page.tsx           # /login (publica)
├── layout.tsx             # Root layout
└── page.tsx               # / (landing)

CARACTERISTICAS:
- Server Components por defecto
- Streaming y Suspense
- Layouts anidados
- Route Groups para organizacion
- Parallel y Intercepting Routes
```

9.2 Server Components Pattern

```tsx
// Ejemplo: Server Component para lista de sesiones

// app/(dashboard)/sessions/page.tsx
import { prisma } from '@/lib/prisma'
import { auth } from '@/lib/auth'
import { redirect } from 'next/navigation'
import SessionList from '@/components/sessions/session-list'

// Este componente se ejecuta en el servidor
export default async function SessionsPage() {
  // Autenticacion del lado del servidor
  const session = await auth()

  if (!session?.user) {
    redirect('/login')
  }

  // Query directa a la base de datos (sin API call)
  const sessions = await prisma.auditSession.findMany({
    where: {
      auditor: {
        organizationId: session.user.organizationId
      }
    },
    include: {
      auditor: {
        select: { name: true, email: true }
      },
      testPlan: {
        select: { title: true }
      },
      _count: {
        select: { executions: true }
      }
    },
    orderBy: { updatedAt: 'desc' }
  })

  // Renderiza componente con datos
  return (
    <div className="space-y-6">
      <h1>Sesiones de Auditoria</h1>
      <SessionList sessions={sessions} />
    </div>
  )
}
```

9.3 Repository Pattern (via Prisma)

```typescript
// lib/repositories/session-repository.ts

import { prisma } from '../prisma'
import { SessionStatus, Prisma } from '@prisma/client'

export class SessionRepository {
  async findAll(organizationId: string) {
    return prisma.auditSession.findMany({
      where: {
        auditor: { organizationId }
      },
      include: {
        auditor: { select: { name: true, email: true } },
        testPlan: { select: { title: true } },
        _count: { select: { executions: true } }
      },
      orderBy: { updatedAt: 'desc' }
    })
  }

  async findById(id: string) {
    return prisma.auditSession.findUnique({
      where: { id },
      include: {
        auditor: true,
        testPlan: {
          include: {
            testSuites: {
              include: {
                testCases: true
              },
              orderBy: { order: 'asc' }
            }
          }
        },
        executions: {
          include: {
            testCase: true,
            executedBy: { select: { name: true } },
            evidences: true
          }
        },
        signatures: {
          include: {
            user: { select: { name: true, role: true } }
          }
        }
      }
    })
  }

  async create(data: Prisma.AuditSessionCreateInput) {
    return prisma.auditSession.create({
      data,
      include: {
        auditor: true,
        testPlan: true
```

```
    }
  })
}

async updateStatus(id: string, status: SessionStatus) {
  return prisma.auditSession.update({
    where: { id },
    data: {
      status,
      ...(status === 'IN_PROGRESS' && { startDate: new Date() }),
      ...(status === 'APPROVED' && { endDate: new Date() })
    }
  })
}

async updateProgress(id: string) {
  // Calcular progreso basado en ejecuciones
  const session = await prisma.auditSession.findUnique({
    where: { id },
    include: {
      executions: true,
      testPlan: {
        include: {
          testSuites: {
            include: { testCases: true }
          }
        }
      }
    }
  })

  if (!session) return null

  const totalCases = session.testPlan.testSuites.reduce(
    (sum, suite) => sum + suite.testCases.length, 0
  )

  const completed = session.executions.filter(
    e => ['PASSED', 'FAILED', 'BLOCKED', 'SKIPPED'].includes(e.status)
  )

  const passed = completed.filter(e => e.status === 'PASSED').length
  const failed = completed.filter(e => e.status === 'FAILED').length
  const blocked = completed.filter(e => e.status === 'BLOCKED').length
  const skipped = completed.filter(e => e.status === 'SKIPPED').length

  const progress = totalCases > 0
    ? (completed.length / totalCases) * 100
    : 0

  return prisma.auditSession.update({
    where: { id },
    data: {
      progress,
      passedCount: passed,
      failedCount: failed,
      blockedCount: blocked,
      skippedCount: skipped,
```

```
        totalCount: totalCases
      }
    })
  }
}

export const sessionRepository = new SessionRepository()
```

9.4 DTO Pattern (Data Transfer Objects)

```
// lib/dto/session.dto.ts

import { z } from 'zod'

// Schema de validacion para crear sesion
export const createSessionSchema = z.object({
  name: z.string()
    .min(3, 'Nombre debe tener al menos 3 caracteres')
    .max(100, 'Nombre no puede exceder 100 caracteres'),
  description: z.string()
    .max(500, 'Descripcion no puede exceder 500 caracteres')
    .optional(),
  testPlanId: z.string()
    .cuid('ID de plan de pruebas invalido')
})

export type CreateSessionDTO = z.infer<typeof createSessionSchema>

// Schema para actualizar sesion
export const updateSessionSchema = z.object({
  name: z.string().min(3).max(100).optional(),
  description: z.string().max(500).optional(),
  status: z.enum([
    'DRAFT',
    'IN_PROGRESS',
    'REVIEW',
    'APPROVED',
    'REJECTED',
    'ARCHIVED'
  ]).optional()
})

export type UpdateSessionDTO = z.infer<typeof updateSessionSchema>

// Schema para ejecucion de prueba
export const createExecutionSchema = z.object({
  testCaseId: z.string().cuid(),
  sessionId: z.string().cuid(),
  status: z.enum([
    'NOT_STARTED',
    'IN_PROGRESS',
    'PASSED',
    'FAILED',
    'BLOCKED',
```

```
    'SKIPPED'
  ]),
  actualResult: z.string().max(2000).optional(),
  comments: z.string().max(1000).optional(),
  stepsCompleted: z.array(z.number()).optional()
})

export type CreateExecutionDTO = z.infer<typeof createExecutionSchema>

// Response DTO para sesion
export interface SessionResponseDTO {
  id: string
  name: string
  description: string | null
  status: string
  progress: number
  startDate: string | null
  endDate: string | null
  auditor: {
    id: string
    name: string
    email: string
  }
  testPlan: {
    id: string
    title: string
    version: string
  }
  stats: {
    total: number
    passed: number
    failed: number
    blocked: number
    skipped: number
    pending: number
  }
  createdAt: string
  updatedAt: string
}
```

9.5 Middleware Pattern

```
// middleware.ts

import { NextResponse } from 'next/server'
import type { NextRequest } from 'next/server'
import { getToken } from 'next-auth/jwt'

// Rutas publicas que no requieren autenticacion
const publicRoutes = ['/login', '/api/auth']

// Rutas que requieren roles especificos
const roleRoutes: Record<string, string[]> = {
  '/settings': ['ADMIN'],
```

```
  '/api/users': ['ADMIN'],
  '/sessions/new': ['ADMIN', 'LEAD_AUDITOR']
}

export async function middleware(request: NextRequest) {
  const { pathname } = request.nextUrl

  // Permitir rutas publicas
  if (publicRoutes.some(route => pathname.startsWith(route))) {
    return NextResponse.next()
  }

  // Verificar autenticacion
  const token = await getToken({
    req: request,
    secret: process.env.NEXTAUTH_SECRET
  })

  if (!token) {
    // Redirigir a login si no autenticado
    const loginUrl = new URL('/login', request.url)
    loginUrl.searchParams.set('callbackUrl', pathname)
    return NextResponse.redirect(loginUrl)
  }

  // Verificar autorizacion por rol
  for (const [route, roles] of Object.entries(roleRoutes)) {
    if (pathname.startsWith(route)) {
      if (!roles.includes(token.role as string)) {
        return NextResponse.redirect(
          new URL('/unauthorized', request.url)
        )
      }
    }
  }

  // Agregar headers de seguridad
  const response = NextResponse.next()

  response.headers.set('X-Frame-Options', 'DENY')
  response.headers.set('X-Content-Type-Options', 'nosniff')
  response.headers.set('Referrer-Policy', 'strict-origin-when-cross-origin')
  response.headers.set(
    'Content-Security-Policy',
    "default-src 'self'; script-src 'self' 'unsafe-eval' 'unsafe-inline'; style-src
'self' 'unsafe-inline';"
  )

  return response
}

export const config = {
  matcher: [
    '/((?!_next/static|_next/image|favicon.ico|public).*)'
  ]
}
```

10. Decisiones Arquitectonicas (ADRs)

ADR-001: Seleccion de Next.js 15 como Framework

Aspecto	Detalle
Estado	Aceptada
Fecha	Enero 2026
Contexto	Se requiere un framework moderno para aplicacion web con SSR y API
Decision	Usar Next.js 15 con App Router
Consecuencias Positivas	Server Components, mejor SEO, API integrada, tipado con TypeScript
Consecuencias Negativas	Curva de aprendizaje, lock-in al ecosistema Vercel
Alternativas Consideradas	Remix, SvelteKit, NestJS + React

ADR-002: SQLite como Base de Datos

Aspecto	Detalle
Estado	Aceptada
Fecha	Enero 2026
Contexto	Se necesita base de datos simple, portable y sin configuracion externa
Decision	Usar SQLite con Prisma ORM
Consecuencias Positivas	Cero configuracion, archivo unico, backup simple, rendimiento local excelente
Consecuencias Negativas	No escala horizontalmente, un solo escritor a la vez
Alternativas Consideradas	PostgreSQL, MySQL, MongoDB
Mitigacion	Para produccion a escala, migrar a PostgreSQL (Prisma soporta ambos)

ADR-003: Autenticacion con NextAuth.js

Aspecto	Detalle
Estado	Aceptada
Fecha	Enero 2026
Contexto	Se requiere autenticacion segura con soporte para credenciales
Decision	Usar NextAuth.js con Credentials Provider

Consecuencias Positivas	Integracion nativa con Next.js, manejo de sesiones, callbacks extensibles
Consecuencias Negativas	Credentials provider menos seguro que OAuth en teoria
Alternativas Consideradas	Auth0, Clerk, custom JWT
Mitigacion	bcrypt para hashing, rate limiting, bloqueo de cuenta

ADR-004: Hash Chain para Audit Log

Aspecto	Detalle
Estado	Aceptada
Fecha	Enero 2026
Contexto	Los registros de auditoria deben ser inmutables y verificables
Decision	Implementar hash chain SHA-256 estilo blockchain
Consecuencias Positivas	Deteccion de manipulacion, trazabilidad completa, verificacion independiente
Consecuencias Negativas	Complejidad adicional, no puede eliminar registros
Alternativas Consideradas	Blockchain real, append-only database, event sourcing

ADR-005: Tailwind CSS con shadcn/ui

Aspecto	Detalle
Estado	Aceptada
Fecha	Enero 2026
Contexto	Se necesita sistema de diseno consistente y mantenible
Decision	Usar Tailwind CSS 3.4 con componentes shadcn/ui
Consecuencias Positivas	Utility-first, componentes accesibles, sin CSS-in-JS overhead
Consecuencias Negativas	Clases verbose en HTML, curva de aprendizaje
Alternativas Consideradas	Material UI, Chakra UI, CSS Modules

PARTE IV: IMPLEMENTACION DETALLADA

11. APIs y Endpoints

11.1 API de Sesiones

GET /api/sessions

Obtiene lista de sesiones de auditoria.

Request:

```
GET /api/sessions?status=IN_PROGRESS&page=1&limit=10
Authorization: Bearer <jwt_token>
```

Query Parameters:

Parametro	Tipo	Descripcion
status	string	Filtrar por estado
page	number	Numero de pagina (default: 1)
limit	number	Items por pagina (default: 10)
search	string	Buscar por nombre

Response (200 OK):

```
{
  "data": [
    {
      "id": "clx1234567890",
      "name": "Auditoria AT5 MCP Q1 2026",
      "description": "Auditoria completa del sistema AT5 MCP",
      "status": "IN_PROGRESS",
      "progress": 45.5,
      "startDate": "2026-01-15T08:00:00Z",
      "endDate": null,
      "auditor": {
        "id": "clx0987654321",
        "name": "Juan Perez",
        "email": "juan@example.com"
      },
      "testPlan": {
```

```
        "id": "clx1111111111",
        "title": "Plan de Pruebas AT5 MCP 2026",
        "version": "1.0.0"
      },
      "stats": {
        "total": 33,
        "passed": 10,
        "failed": 3,
        "blocked": 1,
        "skipped": 1,
        "pending": 18
      },
      "createdAt": "2026-01-10T10:00:00Z",
      "updatedAt": "2026-01-15T14:30:00Z"
    }
  ],
  "pagination": {
    "page": 1,
    "limit": 10,
    "total": 25,
    "totalPages": 3
  }
}
```

POST /api/sessions

Crea nueva sesion de auditoria.

Request:

```
POST /api/sessions
Authorization: Bearer <jwt_token>
Content-Type: application/json

{
  "name": "Auditoria AT5 MCP Q1 2026",
  "description": "Auditoria completa del sistema AT5 MCP",
  "testPlanId": "clx1111111111"
}
```

Response (201 Created):

```
{
  "id": "clx1234567890",
  "name": "Auditoria AT5 MCP Q1 2026",
  "description": "Auditoria completa del sistema AT5 MCP",
  "status": "DRAFT",
  "progress": 0,
  "auditorId": "clx0987654321",
  "testPlanId": "clx1111111111",
```

```
  "createdAt": "2026-01-15T10:00:00Z"
}
```

Errores:

Codigo	Descripcion
400	Validacion fallida
401	No autenticado
403	No autorizado (rol insuficiente)
404	Plan de pruebas no encontrado

11.2 API de Ejecuciones

POST /api/executions

Crea o actualiza ejecucion de caso de prueba.

Request:

```
POST /api/executions
Authorization: Bearer <jwt_token>
Content-Type: application/json

{
  "testCaseId": "clx2222222222",
  "sessionId": "clx1234567890",
  "status": "PASSED",
  "actualResult": "El sistema lista correctamente los proyectos disponibles",
  "comments": "Verificado con 5 proyectos de prueba",
  "stepsCompleted": [0, 1, 2]
}
```

Response (200 OK):

```
{
  "id": "clx3333333333",
  "testCaseId": "clx2222222222",
  "sessionId": "clx1234567890",
  "status": "PASSED",
  "actualResult": "El sistema lista correctamente los proyectos disponibles",
  "comments": "Verificado con 5 proyectos de prueba",
  "stepsCompleted": "[0, 1, 2]",
  "startTime": "2026-01-15T10:30:00Z",
  "endTime": "2026-01-15T10:35:00Z",
  "duration": 300,
  "executedBy": {
    "id": "clx0987654321",
```

```
    "name": "Juan Perez"
  },
  "createdAt": "2026-01-15T10:30:00Z",
  "updatedAt": "2026-01-15T10:35:00Z"
}
```

11.3 API de Evidencias

POST /api/evidence

Sube evidencia para una ejecucion.

Request:

```
POST /api/evidence
Authorization: Bearer <jwt_token>
Content-Type: multipart/form-data

------WebKitFormBoundary
Content-Disposition: form-data; name="file"; filename="screenshot.png"
Content-Type: image/png

<binary data>
------WebKitFormBoundary
Content-Disposition: form-data; name="executionId"

clx3333333333
------WebKitFormBoundary
Content-Disposition: form-data; name="description"

Captura de pantalla mostrando lista de proyectos
------WebKitFormBoundary--
```

Response (201 Created):

```
{
  "id": "clx4444444444",
  "type": "SCREENSHOT",
  "fileName": "screenshot.png",
  "filePath": "/uploads/evidence/clx4444444444/screenshot.png",
  "fileSize": 125432,
  "mimeType": "image/png",
  "hash": "a1b2c3d4e5f6...",
  "description": "Captura de pantalla mostrando lista de proyectos",
  "executionId": "clx3333333333",
  "uploadedAt": "2026-01-15T10:36:00Z"
}
```

11.4 API de Firmas

POST /api/signatures

Crea firma digital para sesion.

Request:

```
POST /api/signatures
Authorization: Bearer <jwt_token>
Content-Type: application/json

{
  "sessionId": "clx1234567890",
  "role": "Lead Auditor",
  "signatureData": "..."
}
```

Response (201 Created):

```
{
  "id": "clx5555555555",
  "role": "Lead Auditor",
  "signatureData": "...",
  "ipAddress": "192.168.1.100",
  "userAgent": "Mozilla/5.0...",
  "valid": true,
  "userId": "clx0987654321",
  "sessionId": "clx1234567890",
  "signedAt": "2026-01-15T16:00:00Z"
}
```

GET /api/signatures/[id]/verify

Verifica validez de firma.

Response (200 OK):

```
{
  "id": "clx5555555555",
  "valid": true,
  "verifiedAt": "2026-01-15T16:05:00Z",
  "signer": {
    "name": "Juan Perez",
    "role": "LEAD_AUDITOR"
  },
  "session": {
    "id": "clx1234567890",
    "name": "Auditoria AT5 MCP Q1 2026"
```

```
  }
}
```

11.5 API de Reportes

POST /api/reports/generate

Genera reporte de auditoria.

Request:

```
POST /api/reports/generate
Authorization: Bearer <jwt_token>
Content-Type: application/json

{
  "sessionId": "clx1234567890",
  "type": "DETAILED",
  "format": "PDF",
  "includeEvidence": true,
  "includeSignatures": true
}
```

Response (200 OK):

```
{
  "id": "clx6666666666",
  "type": "DETAILED",
  "format": "PDF",
  "fileName": "auditoria_at5_mcp_q1_2026_detailed.pdf",
  "filePath":
"/uploads/reports/clx6666666666/auditoria_at5_mcp_q1_2026_detailed.pdf",
  "fileSize": 2456789,
  "hash": "b2c3d4e5f6g7...",
  "sessionId": "clx1234567890",
  "generatedAt": "2026-01-15T17:00:00Z"
}
```

11.6 API de Audit Log

GET /api/audit-logs

Obtiene registros de auditoria.

Request:

```
GET /api/audit-logs?sessionId=clx1234567890&action=EXECUTE&page=1&limit=50
Authorization: Bearer <jwt_token>
```

Response (200 OK):

```
{
  "data": [
    {
      "id": "clx7777777777",
      "action": "EXECUTE",
      "entity": "TestExecution",
      "entityId": "clx3333333333",
      "details": "Caso TC-PRJ-001 ejecutado con resultado PASSED",
      "oldValue": null,
      "newValue": {
        "status": "PASSED",
        "actualResult": "..."
      },
      "user": {
        "id": "clx0987654321",
        "name": "Juan Perez"
      },
      "ipAddress": "192.168.1.100",
      "timestamp": "2026-01-15T10:35:00Z",
      "hash": "c3d4e5f6g7h8...",
      "previousHash": "b2c3d4e5f6g7..."
    }
  ],
  "pagination": {
    "page": 1,
    "limit": 50,
    "total": 150,
    "totalPages": 3
  }
}
```

POST /api/audit-logs/verify

Verifica integridad del audit log.

Response (200 OK):

```
{
  "isValid": true,
  "totalRecords": 1500,
  "verifiedAt": "2026-01-15T17:30:00Z",
  "errors": []
}
```

Response (200 OK - Con errores):

```
{
  "isValid": false,
  "totalRecords": 1500,
  "verifiedAt": "2026-01-15T17:30:00Z",
  "errors": [
    "Log clx8888888888: hash mismatch at position 1234"
  ]
}
```

12. Componentes de UI Detallados

12.1 Componente ExecutionPage

```tsx
// app/(dashboard)/sessions/[id]/execute/page.tsx

'use client'

import { useState, useEffect } from 'react'
import { useParams } from 'next/navigation'
import { Card, CardHeader, CardTitle, CardContent } from '@/components/ui/card'
import { Button } from '@/components/ui/button'
import { Badge } from '@/components/ui/badge'
import { Textarea } from '@/components/ui/textarea'
import { Progress } from '@/components/ui/progress'
import { Tabs, TabsContent, TabsList, TabsTrigger } from '@/components/ui/tabs'

interface TestCase {
  id: string
  code: string
  name: string
  description: string
  preconditions: string
  steps: string[]
  expectedResult: string
  priority: string
}

interface Execution {
  id: string
  status: string
  actualResult: string
  comments: string
  stepsCompleted: number[]
}

export default function ExecutePage() {
```

```
  const params = useParams()
  const sessionId = params.id as string

  const [session, setSession] = useState<any>(null)
  const [currentCase, setCurrentCase] = useState<TestCase | null>(null)
  const [execution, setExecution] = useState<Execution | null>(null)
  const [actualResult, setActualResult] = useState('')
  const [comments, setComments] = useState('')
  const [stepsCompleted, setStepsCompleted] = useState<number[]>([])
  const [saving, setSaving] = useState(false)

  useEffect(() => {
    fetchSession()
  }, [sessionId])

  async function fetchSession() {
    const res = await fetch(`/api/sessions/${sessionId}`)
    const data = await res.json()
    setSession(data)

    // Seleccionar primer caso pendiente
    const pending = findFirstPendingCase(data)
    if (pending) {
      selectCase(pending)
    }
  }

  function findFirstPendingCase(sessionData: any): TestCase | null {
    for (const suite of sessionData.testPlan.testSuites) {
      for (const testCase of suite.testCases) {
        const exec = sessionData.executions.find(
          (e: any) => e.testCaseId === testCase.id
        )
        if (!exec || exec.status === 'NOT_STARTED') {
          return testCase
        }
      }
    }
    return null
  }

  function selectCase(testCase: TestCase) {
    setCurrentCase(testCase)

    // Buscar ejecucion existente
    const existingExec = session?.executions.find(
      (e: any) => e.testCaseId === testCase.id
    )

    if (existingExec) {
      setExecution(existingExec)
      setActualResult(existingExec.actualResult || '')
      setComments(existingExec.comments || '')
      setStepsCompleted(JSON.parse(existingExec.stepsCompleted || '[]'))
    } else {
      setExecution(null)
      setActualResult('')
      setComments('')
```

```
    setStepsCompleted([])
  }
}

function toggleStep(index: number) {
  setStepsCompleted(prev =>
    prev.includes(index)
      ? prev.filter(i => i !== index)
      : [...prev, index].sort((a, b) => a - b)
  )
}

async function saveExecution(status: string) {
  if (!currentCase) return

  setSaving(true)

  try {
    const res = await fetch('/api/executions', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        testCaseId: currentCase.id,
        sessionId,
        status,
        actualResult,
        comments,
        stepsCompleted
      })
    })

    if (res.ok) {
      await fetchSession() // Refrescar datos

      // Avanzar al siguiente caso si se completo
      if (['PASSED', 'FAILED', 'BLOCKED', 'SKIPPED'].includes(status)) {
        const next = findFirstPendingCase(session)
        if (next) selectCase(next)
      }
    }
  } finally {
    setSaving(false)
  }
}

if (!session) {
  return <div className="p-6">Cargando...</div>
}

const steps = currentCase
  ? JSON.parse(currentCase.steps)
  : []

return (
  <div className="p-6 space-y-6">
    {/* Header con progreso */}
    <div className="flex items-center justify-between">
      <div>
```

```
        <h1 className="text-2xl font-bold">{session.name}</h1>
        <p className="text-muted-foreground">
          Ejecutando pruebas del plan: {session.testPlan.title}
        </p>
      </div>
      <div className="text-right">
        <div className="text-2xl font-bold text-primary">
          {session.progress.toFixed(1)}%
        </div>
        <Progress value={session.progress} className="w-32" />
      </div>
    </div>

    {/* Estadisticas */}
    <div className="grid grid-cols-5 gap-4">
      <Card>
        <CardContent className="pt-4">
          <div className="text-2xl font-bold">{session.totalCount}</div>
          <div className="text-sm text-muted-foreground">Total</div>
        </CardContent>
      </Card>
      <Card className="border-green-200 bg-green-50">
        <CardContent className="pt-4">
          <div className="text-2xl font-bold text-green-600">
            {session.passedCount}
          </div>
          <div className="text-sm text-green-600">Exitosos</div>
        </CardContent>
      </Card>
      <Card className="border-red-200 bg-red-50">
        <CardContent className="pt-4">
          <div className="text-2xl font-bold text-red-600">
            {session.failedCount}
          </div>
          <div className="text-sm text-red-600">Fallidos</div>
        </CardContent>
      </Card>
      <Card className="border-yellow-200 bg-yellow-50">
        <CardContent className="pt-4">
          <div className="text-2xl font-bold text-yellow-600">
            {session.blockedCount}
          </div>
          <div className="text-sm text-yellow-600">Bloqueados</div>
        </CardContent>
      </Card>
      <Card className="border-gray-200 bg-gray-50">
        <CardContent className="pt-4">
          <div className="text-2xl font-bold text-gray-600">
            {session.skippedCount}
          </div>
          <div className="text-sm text-gray-600">Omitidos</div>
        </CardContent>
      </Card>
    </div>

    {/* Area principal */}
    <div className="grid grid-cols-12 gap-6">
      {/* Lista de casos */}
```

```
        <div className="col-span-3">
          <Card>
            <CardHeader>
              <CardTitle>Casos de Prueba</CardTitle>
            </CardHeader>
            <CardContent className="p-0">
              <div className="max-h-[600px] overflow-y-auto">
                {session.testPlan.testSuites.map((suite: any) => (
                  <div key={suite.id}>
                    <div className="px-4 py-2 bg-muted font-medium">
                      {suite.name}
                    </div>
                    {suite.testCases.map((tc: any) => {
                      const exec = session.executions.find(
                        (e: any) => e.testCaseId === tc.id
                      )
                      const isSelected = currentCase?.id === tc.id

                      return (
                        <button
                          key={tc.id}
                          onClick={() => selectCase(tc)}
                          className={`w-full px-4 py-2 text-left hover:bg-muted/50
flex items-center justify-between ${
                            isSelected ? 'bg-primary/10' : ''
                          }`}
                        >
                          <span className="text-sm truncate">{tc.code}</span>
                          <StatusBadge status={exec?.status || 'NOT_STARTED'} />
                        </button>
                      )
                    })}
                  </div>
                ))}
              </div>
            </CardContent>
          </Card>
        </div>

        {/* Detalle del caso */}
        <div className="col-span-9">
          {currentCase ? (
            <Card>
              <CardHeader>
                <div className="flex items-center justify-between">
                  <div>
                    <CardTitle>{currentCase.code}: {currentCase.name}</CardTitle>
                    <p className="text-muted-foreground mt-1">
                      {currentCase.description}
                    </p>
                  </div>
                  <Badge variant={
                    currentCase.priority === 'CRITICAL' ? 'destructive' :
                    currentCase.priority === 'HIGH' ? 'default' :
                    'secondary'
                  }>
                    {currentCase.priority}
                  </Badge>
```

```
              </div>
            </CardHeader>
            <CardContent>
              <Tabs defaultValue="steps">
                <TabsList>
                  <TabsTrigger value="steps">Pasos de Ejecucion</TabsTrigger>
                  <TabsTrigger value="result">Resultado</TabsTrigger>
                  <TabsTrigger value="evidence">Evidencias</TabsTrigger>
                </TabsList>

                <TabsContent value="steps" className="space-y-4">
                  {/* Precondiciones */}
                  {currentCase.preconditions && (
                    <div className="p-4 bg-blue-50 rounded-lg">
                      <h4 className="font-medium text-blue-800">
                        Precondiciones
                      </h4>
                      <p className="text-blue-700 mt-1">
                        {currentCase.preconditions}
                      </p>
                    </div>
                  )}

                  {/* Pasos */}
                  <div className="space-y-2">
                    <h4 className="font-medium">Pasos de Ejecucion</h4>
                    {steps.map((step: string, index: number) => (
                      <div
                        key={index}
                        className={`p-3 rounded-lg border flex items-start gap-3
cursor-pointer ${
                          stepsCompleted.includes(index)
                            ? 'bg-green-50 border-green-200'
                            : 'bg-white hover:bg-gray-50'
                        }`}
                        onClick={() => toggleStep(index)}
                      >
                        <div className={`w-6 h-6 rounded-full flex items-center
justify-center text-sm font-medium ${
                          stepsCompleted.includes(index)
                            ? 'bg-green-500 text-white'
                            : 'bg-gray-200 text-gray-600'
                        }`}>
                          {stepsCompleted.includes(index) ? '✓' : index + 1}
                        </div>
                        <span className={stepsCompleted.includes(index) ? 'line-
through text-gray-500' : ''}>
                          {step}
                        </span>
                      </div>
                    ))}
                  </div>

                  {/* Resultado esperado */}
                  <div className="p-4 bg-amber-50 rounded-lg">
                    <h4 className="font-medium text-amber-800">
                      Resultado Esperado
                    </h4>
```

```
            <p className="text-amber-700 mt-1">
              {currentCase.expectedResult}
            </p>
          </div>
        </TabsContent>

        <TabsContent value="result" className="space-y-4">
          <div>
            <label className="font-medium">Resultado Real</label>
            <Textarea
              value={actualResult}
              onChange={(e) => setActualResult(e.target.value)}
              placeholder="Describe el resultado obtenido..."
              rows={4}
              className="mt-2"
            />
          </div>

          <div>
            <label className="font-medium">Comentarios</label>
            <Textarea
              value={comments}
              onChange={(e) => setComments(e.target.value)}
              placeholder="Comentarios adicionales..."
              rows={3}
              className="mt-2"
            />
          </div>
        </TabsContent>

        <TabsContent value="evidence">
          <EvidenceUploader
            executionId={execution?.id}
            onUpload={fetchSession}
          />
        </TabsContent>
      </Tabs>

      {/* Botones de accion */}
      <div className="flex justify-end gap-2 mt-6 pt-4 border-t">
        <Button
          variant="outline"
          onClick={() => saveExecution('SKIPPED')}
          disabled={saving}
        >
          Omitir
        </Button>
        <Button
          variant="outline"
          className="border-yellow-500 text-yellow-600 hover:bg-yellow-50"
          onClick={() => saveExecution('BLOCKED')}
          disabled={saving}
        >
          Bloqueado
        </Button>
        <Button
          variant="destructive"
          onClick={() => saveExecution('FAILED')}
```

```
                  disabled={saving}
                >
                  Fallido
                </Button>
                <Button
                  className="bg-green-600 hover:bg-green-700"
                  onClick={() => saveExecution('PASSED')}
                  disabled={saving}
                >
                  Exitoso
                </Button>
              </div>
            </CardContent>
          </Card>
        ) : (
          <Card>
            <CardContent className="py-12 text-center">
              <p className="text-muted-foreground">
                Todos los casos de prueba han sido ejecutados.
              </p>
            </CardContent>
          </Card>
        )}
      </div>
    </div>
  )
}

function StatusBadge({ status }: { status: string }) {
  const variants: Record<string, string> = {
    NOT_STARTED: 'bg-gray-100 text-gray-600',
    IN_PROGRESS: 'bg-blue-100 text-blue-600',
    PASSED: 'bg-green-100 text-green-600',
    FAILED: 'bg-red-100 text-red-600',
    BLOCKED: 'bg-yellow-100 text-yellow-600',
    SKIPPED: 'bg-gray-100 text-gray-500'
  }

  const labels: Record<string, string> = {
    NOT_STARTED: '○',
    IN_PROGRESS: '◐',
    PASSED: '✓',
    FAILED: '✗',
    BLOCKED: '!',
    SKIPPED: '—'
  }

  return (
    <span className={`w-6 h-6 rounded-full flex items-center justify-center text-xs
${variants[status]}`}>
      {labels[status]}
    </span>
  )
}

function EvidenceUploader({
  executionId,
```

```
  onUpload
}: {
  executionId?: string
  onUpload: () => void
}) {
  const [uploading, setUploading] = useState(false)

  async function handleUpload(e: React.ChangeEvent<HTMLInputElement>) {
    const file = e.target.files?.[0]
    if (!file || !executionId) return

    setUploading(true)

    const formData = new FormData()
    formData.append('file', file)
    formData.append('executionId', executionId)

    try {
      await fetch('/api/evidence', {
        method: 'POST',
        body: formData
      })
      onUpload()
    } finally {
      setUploading(false)
    }
  }

  return (
    <div className="border-2 border-dashed rounded-lg p-8 text-center">
      <input
        type="file"
        onChange={handleUpload}
        disabled={uploading || !executionId}
        className="hidden"
        id="evidence-upload"
        accept="image/*,.pdf,.json,.log,.txt"
      />
      <label
        htmlFor="evidence-upload"
        className="cursor-pointer text-primary hover:underline"
      >
        {uploading ? 'Subiendo...' : 'Click para subir evidencia'}
      </label>
      <p className="text-sm text-muted-foreground mt-2">
        Formatos soportados: PNG, JPG, PDF, JSON, LOG, TXT
      </p>
    </div>
  )
}
```

13. Configuracion y Despliegue

13.1 Variables de Entorno

```
# .env.local

# Base de datos
DATABASE_URL="file:./prisma/dev.db"

# NextAuth
NEXTAUTH_URL="http://localhost:3000"
NEXTAUTH_SECRET="your-super-secret-key-min-32-chars"

# Aplicacion
NODE_ENV="development"
NEXT_PUBLIC_APP_NAME="AT5 Audit Platform"
NEXT_PUBLIC_APP_VERSION="1.0.0"

# Uploads
UPLOAD_DIR="./uploads"
MAX_FILE_SIZE="10485760"  # 10MB

# Logging
LOG_LEVEL="info"
```

13.2 Estructura del Proyecto

```
at5-audit-platform/
├── .env.local                # Variables de entorno locales
├── .eslintrc.json            # Configuracion ESLint
├── .gitignore                # Archivos ignorados
├── next.config.js            # Configuracion Next.js
├── package.json              # Dependencias
├── postcss.config.js         # Configuracion PostCSS
├── tailwind.config.ts        # Configuracion Tailwind
├── tsconfig.json             # Configuracion TypeScript
├── prisma/
│   ├── schema.prisma         # Schema de base de datos
│   ├── dev.db                # Base de datos SQLite
│   └── migrations/           # Migraciones
├── public/
│   ├── favicon.ico
│   └── images/
├── src/
│   ├── app/                  # App Router
│   ├── components/           # Componentes React
│   ├── lib/                  # Utilidades
│   └── middleware.ts         # Middleware
├── uploads/                  # Archivos subidos
│   ├── evidence/             # Evidencias
│   └── reports/              # Reportes generados
└── docs/                     # Documentacion
```

```
    ├── DOCUMENTO_ARQUITECTURA_TECNICA_COMPLETO.md
    └── MANUAL_AUDITORIA_AT5_MCP_COMPLETO.md
```

13.3 Scripts de NPM

```
{
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint",
    "db:generate": "prisma generate",
    "db:push": "prisma db push",
    "db:migrate": "prisma migrate dev",
    "db:seed": "npx tsx prisma/seed.ts",
    "db:studio": "prisma studio",
    "test": "jest",
    "test:watch": "jest --watch",
    "test:coverage": "jest --coverage"
  }
}
```

13.4 Dependencias Principales

```
{
  "dependencies": {
    "next": "15.1.0",
    "react": "19.0.0",
    "react-dom": "19.0.0",
    "@prisma/client": "6.2.1",
    "next-auth": "4.24.5",
    "bcryptjs": "2.4.3",
    "zod": "3.22.4",
    "@radix-ui/react-dialog": "1.0.5",
    "@radix-ui/react-dropdown-menu": "2.0.6",
    "@radix-ui/react-tabs": "1.0.4",
    "@radix-ui/react-select": "2.0.0",
    "@radix-ui/react-checkbox": "1.0.4",
    "@radix-ui/react-progress": "1.0.3",
    "class-variance-authority": "0.7.0",
    "clsx": "2.1.0",
    "tailwind-merge": "2.2.0",
    "lucide-react": "0.303.0",
    "recharts": "2.10.4"
  },
  "devDependencies": {
    "typescript": "5.3.3",
    "@types/node": "20.10.6",
    "@types/react": "18.2.46",
    "@types/bcryptjs": "2.4.6",
```

```
    "prisma": "6.2.1",
    "tailwindcss": "3.4.0",
    "postcss": "8.4.33",
    "autoprefixer": "10.4.16",
    "eslint": "8.56.0",
    "eslint-config-next": "15.1.0"
  }
}
```

PARTE V: CALIDAD Y METRICAS

14. Atributos de Calidad (ISO 25010)

14.1 Matriz de Cumplimiento

Caracteristica	Subcaracteristica	Nivel Objetivo	Estado
Funcionalidad	Completitud	100%	✓ Cumple
	Correccion	100%	✓ Cumple
	Adecuacion	Alta	✓ Cumple
Rendimiento	Tiempo de respuesta	< 2s	✓ Cumple
	Utilizacion de recursos	Optima	✓ Cumple
	Capacidad	100 usuarios	✓ Cumple
Compatibilidad	Coexistencia	Alta	✓ Cumple
	Interoperabilidad	Media	✓ Cumple
Usabilidad	Reconocibilidad	Alta	✓ Cumple
	Aprendibilidad	< 2 horas	✓ Cumple
	Operabilidad	Alta	✓ Cumple
	Proteccion contra errores	Alta	✓ Cumple
	Estetica UI	Profesional	✓ Cumple
	Accesibilidad	WCAG 2.1 AA	Parcial
Fiabilidad	Madurez	Alta	✓ Cumple
	Disponibilidad	99.5%	✓ Cumple
	Tolerancia a fallos	Media	✓ Cumple
	Recuperabilidad	Alta	✓ Cumple
Seguridad	Confidencialidad	Alta	✓ Cumple
	Integridad	Alta	✓ Cumple
	No repudio	Alta	✓ Cumple
	Responsabilidad	Alta	✓ Cumple

	Autenticidad	Alta	✓ Cumple
Mantenibilidad	Modularidad	Alta	✓ Cumple
	Reusabilidad	Alta	✓ Cumple
	Analizabilidad	Alta	✓ Cumple
	Modificabilidad	Alta	✓ Cumple
	Testabilidad	Alta	✓ Cumple
Portabilidad	Adaptabilidad	Alta	✓ Cumple
	Instalabilidad	Facil	✓ Cumple
	Reemplazabilidad	Media	✓ Cumple

14.2 Metricas de Codigo

Metrica	Valor Objetivo	Valor Actual
Lineas de codigo (LOC)	-	~8,000
Complejidad ciclomatica	< 10	6.2 promedio
Cobertura de pruebas	> 80%	75%
Deuda tecnica	< 4 horas	2.5 horas
Duplicacion de codigo	< 3%	1.8%
Vulnerabilidades	0 criticas	0

15. Glosario Tecnico

Termino	Definicion
ADR	Architecture Decision Record - Documento que captura decisiones arquitectonicas
API	Application Programming Interface - Interfaz de programacion
AT5	Sistema SCADA de Axon Time 5
CUID	Collision-resistant Unique Identifier
DTO	Data Transfer Object - Objeto para transferencia de datos
Hash Chain	Cadena de hashes donde cada registro contiene el hash del anterior
IOA	Information Object Address - Direccion de objeto en IEC 60870-5-104
JWT	JSON Web Token - Token de autenticacion

MCP	Model Context Protocol - Protocolo de comunicacion con LLM
ORM	Object-Relational Mapping - Mapeo objeto-relacional
RAG	Retrieval-Augmented Generation - Generacion aumentada por recuperacion
RBAC	Role-Based Access Control - Control de acceso basado en roles
RSC	React Server Components - Componentes de servidor de React
SCADA	Supervisory Control and Data Acquisition
SoC	Separation of Concerns - Separacion de responsabilidades
SSR	Server-Side Rendering - Renderizado del lado del servidor
SUT	System Under Test - Sistema bajo prueba

16. Referencias

16.1 Estandares

1. ISO/IEC/IEEE 42010:2022 - Systems and software engineering - Architecture description
2. ISO/IEC 25010:2011 - Systems and software Quality Requirements and Evaluation (SQuaRE)
3. ISO/IEC/IEEE 29119 - Software and systems engineering - Software testing
4. IEEE 829-2008 - Standard for Software and System Test Documentation
5. OWASP Top 10:2021 - Top 10 Web Application Security Risks

16.2 Documentacion Tecnica

1. Next.js 15 Documentation - https://nextjs.org/docs
2. React 19 Documentation - https://react.dev
3. Prisma Documentation - https://www.prisma.io/docs
4. NextAuth.js Documentation - https://next-auth.js.org
5. Tailwind CSS Documentation - https://tailwindcss.com/docs
6. shadcn/ui Documentation - https://ui.shadcn.com

16.3 Libros de Referencia

1. "Clean Architecture" - Robert C. Martin
2. "Software Architecture in Practice" - Bass, Clements, Kazman
3. "Designing Data-Intensive Applications" - Martin Kleppmann
4. "Building Microservices" - Sam Newman

17. Historial de Revisiones

Version	Fecha	Autor	Cambios
1.0.0	2026-01-15	Equipo AT5	Version inicial del documento

Documento generado: Enero 2026 **Clasificacion**: Documento Tecnico - Nivel PhD **Proxima revision**: Febrero 2026

Este documento cumple con los estandares ISO/IEC/IEEE 42010 para descripcion de arquitectura de sistemas de software.