ESPECIFICACIÓN TÉCNICA COMPLETA

Plataforma SaaS de Gestión de Cumplimiento Normativo y Clima Organizacional

Nombre del Producto: ComplianceHub Pro
Versión del Documento: 1.0
Fecha: Febrero 2026
Clasificación: Documento Técnico - Especificación de Desarrollo

---

1. RESUMEN EJECUTIVO DEL PRODUCTO

1.1 Visión del Producto

ComplianceHub Pro es una plataforma SaaS empresarial integral diseñada para transformar la manera en que las organizaciones gestionan el cumplimiento normativo, el clima organizacional y el bienestar de sus colaboradores. La plataforma nace como respuesta a la creciente complejidad regulatoria en Latinoamérica, particularmente ante normativas como la Ley Karin (Ley 21.643) en Chile, que exige a las empresas implementar mecanismos robustos de prevención, investigación y sanción del acoso laboral, acoso sexual y violencia en el trabajo.

Más allá del cumplimiento normativo obligatorio, ComplianceHub Pro se posiciona como una solución integral que abarca la gestión de relaciones comunitarias, equidad e inclusión, clima laboral, y analytics avanzado con inteligencia artificial.

1.2 Propuesta de Valor

Pilar	Descripción
Cumplimiento Automatizado	Automatización completa de procesos de compliance con generación de reportes regulatorios, trazabilidad y evidencias
Inteligencia de Datos	Analytics avanzado con IA/ML para predicción de riesgos, detección de patrones y recomendaciones proactivas
Experiencia Unificada	Una sola plataforma para múltiples necesidades de compliance, clima y bienestar organizacional
Escalabilidad Enterprise	Arquitectura multi-tenant robusta que soporta desde PYMEs hasta grandes corporaciones
Flexibilidad Regional	Adaptable a múltiples marcos regulatorios de Latinoamérica y España

1.3 Mercado Objetivo

Segmento Primario

- **Empresas medianas y grandes** (100+ empleados) en Chile, con obligación de cumplimiento de Ley Karin
- **Sectores críticos:** Minería, Retail, Banca, Servicios, Salud, Educación
- **Industrias con alto escrutinio regulatorio** y presencia de comunidades stakeholder

Segmento Secundario

- **Empresas multinacionales** con operaciones en Latinoamérica que requieren gestión centralizada de compliance
- **Organizaciones del sector público** sujetas a normativas de transparencia y buen trato
- **Consultoras de RRHH y compliance** que requieren herramientas para sus clientes

1.4 Diferenciadores Competitivos

1. **Enfoque Holístico:** Única plataforma que integra Ley Karin, relaciones comunitarias, EDI y clima laboral
2. **Perspectiva de Género Nativa:** Diseñado desde su concepción con enfoque de género en investigaciones y análisis
3. **IA Predictiva:** Modelos de machine learning para anticipar riesgos psicosociales y de clima
4. **Multi-Canal Inteligente:** Encuestas y denuncias vía web, móvil, SMS, WhatsApp y modo offline (PWA)
5. **Compliance-as-a-Service:** Actualizaciones automáticas ante cambios regulatorios

6. **Arquitectura Enterprise:** Multi-tenant con aislamiento total, escalabilidad horizontal y alta disponibilidad

---

2. DESCRIPCIÓN FUNCIONAL COMPLETA

2.1 Módulos Principales

MÓDULO 1: Cumplimiento Ley Karin (Chile)

2.1.1 Descripción

Módulo especializado para el cumplimiento integral de la Ley 21.643 que establece obligaciones para la prevención, investigación y sanción del acoso laboral, sexual y violencia en el trabajo.

2.1.2 Funcionalidades

A. Protocolos de Prevención
- Gestión de protocolos de prevención con perspectiva de género
- Versionamiento y publicación de protocolos
- Difusión automatizada a colaboradores
- Registro de aceptación/lectura de protocolos
- Templates predefinidos según industria

B. Canal de Denuncias Confidencial
- Portal web anónimo para denuncias
- Línea telefónica integrada (IVR)
- Canal WhatsApp cifrado
- Formularios configurables por tipo de denuncia:
- Acoso laboral
- Acoso sexual
- Violencia en el trabajo
- Violencia por terceros
- Anonimato garantizado con sistema de token para seguimiento
- Adjuntos y evidencias encriptadas
- Notificaciones automáticas a responsables

C. Gestión de Investigaciones
- Workflow configurable de investigación
- Asignación de investigadores internos/externos
- Gestión de conflictos de interés
- Perspectiva de género en todo el proceso
- Registro de entrevistas y testimonios
- Línea de tiempo de eventos
- Generación de informes de investigación
- Propuestas de medidas y sanciones
- Seguimiento de implementación de medidas

D. Atención Psicológica Temprana
- Derivación automática a red de psicólogos
- Seguimiento de sesiones

- Indicadores de bienestar del denunciante
- Integración con EAP (Employee Assistance Program)
- Encuestas de satisfacción del servicio

E. Riesgos Psicosociales
- Cuestionario SUSESO/ISTAS21 integrado
- Análisis por dimensiones de riesgo
- Mapas de calor por área/departamento
- Planes de intervención
- Seguimiento de indicadores

F. Reportería Regulatoria
- Reporte a SUSESO (Superintendencia de Seguridad Social)
- Reporte a ISL (Instituto de Seguridad Laboral)
- Reporte a Dirección del Trabajo
- Formatos oficiales actualizados
- Trazabilidad completa para fiscalizaciones

2.1.3 Casos de Uso Principales

```
CU-LK-001: Ingresar Denuncia Anónima
Actor: Denunciante (colaborador interno o externo)
Precondición: Acceso al canal de denuncias
Flujo Principal:
1. Usuario accede al portal de denuncias
2. Sistema muestra opciones de tipo de denuncia
3. Usuario selecciona tipo (acoso laboral/sexual/violencia)
4. Sistema presenta formulario con campos requeridos
5. Usuario completa información del incidente
6. Usuario puede adjuntar evidencias (opcional)
7. Usuario decide si denuncia es anónima o identificada
8. Sistema genera token de seguimiento
9. Sistema notifica a responsables designados
10. Sistema envía confirmación al denunciante
Postcondición: Denuncia registrada y en estado "Recibida"

CU-LK-002: Gestionar Investigación
Actor: Investigador Designado
Precondición: Denuncia asignada al investigador
Flujo Principal:
1. Investigador recibe notificación de asignación
2. Sistema verifica ausencia de conflictos de interés
3. Investigador acepta caso
4. Investigador agenda entrevistas
5. Investigador registra testimonios con perspectiva de género
6. Investigador recopila evidencias
7. Sistema genera línea de tiempo automática
8. Investigador elabora informe de investigación
9. Investigador propone medidas/sanciones
10. Sistema notifica conclusión a stakeholders autorizados
Postcondición: Investigación cerrada con resolución

CU-LK-003: Generar Reporte Regulatorio
Actor: Encargado de Compliance
Precondición: Datos del período disponibles
Flujo Principal:
1. Usuario selecciona tipo de reporte (SUSESO/ISL/DT)
2. Usuario selecciona período
3. Sistema valida completitud de datos
4. Sistema genera reporte en formato oficial
5. Usuario revisa y aprueba reporte
6. Sistema permite exportar (PDF/Excel)
7. Sistema registra envío en bitácora
Postcondición: Reporte generado y registrado
```

MÓDULO 2: Gestión de Relaciones Comunitarias

2.2.1 Descripción

Módulo para gestionar el relacionamiento con comunidades, especialmente pueblos originarios, en contextos de consulta ciudadana y proyectos de inversión.

2.2.2 Funcionalidades

A. Mapeo de Stakeholders

- Identificación de comunidades y grupos de interés
- Matriz de poder/interés

- Perfilamiento de líderes y representantes
- Historial de interacciones
- Análisis de sentimiento

B. Gestión de Comunidades Indígenas
- Registro de comunidades según Convenio 169 OIT
- Procesos de consulta previa
- Gestión de permisos y autorizaciones
- Acuerdos de beneficios compartidos
- Calendario de compromisos

C. Participación Ciudadana
- Gestión de procesos de participación
- Convocatorias y difusión
- Registro de asistencia
- Documentación de acuerdos
- Seguimiento de compromisos

D. Acuerdos y Compromisos
- Repositorio de acuerdos
- Workflow de aprobación
- Alertas de vencimiento
- Indicadores de cumplimiento
- Reportes de avance

MÓDULO 3: Equidad, Diversidad e Inclusión (EDI)

2.3.1 Descripción

Módulo integral para gestionar iniciativas de equidad de género, diversidad sexual, inclusión de personas con discapacidad y análisis de brechas.

2.3.2 Funcionalidades

A. Equidad de Género
- Diagnóstico de brechas de género
- Indicadores NCG 461 (CMF Chile)
- Planes de acción con metas
- Monitoreo de avances
- Reportes de equidad

B. Diversidad Sexual (LGBTIQ+)
- Políticas de inclusión
- Protocolos de transición de género
- Indicadores de clima inclusivo
- Capacitaciones especializadas
- Encuestas de percepción

C. Inclusión de Personas con Discapacidad
- Cumplimiento Ley 21.015 (Chile)
- Registro de colaboradores con discapacidad
- Gestión de ajustes razonables

- Indicadores de inclusión
- Reportes a Dirección del Trabajo

D. Análisis de Brechas Salariales
- Importación de datos salariales
- Cálculo de brecha por género
- Análisis multivariable
- Simulación de escenarios de corrección
- Benchmarking sectorial

MÓDULO 4: Clima Laboral y Bienestar Organizacional

2.4.1 Descripción

Módulo para medir, analizar y mejorar el clima organizacional y bienestar de los colaboradores.

2.4.2 Funcionalidades

A. Encuestas de Clima
- Cuestionarios validados científicamente
- Dimensiones configurables
- Benchmark por industria
- Análisis de tendencias
- Segmentación por demografía

B. Pulsos de Engagement
- Encuestas cortas frecuentes
- eNPS (Employee Net Promoter Score)
- Indicadores en tiempo real
- Alertas tempranas
- Comparativas históricas

C. Índices de Bienestar
- Bienestar físico
- Bienestar emocional
- Bienestar financiero
- Bienestar social
- Score integrado

D. Programas de Salud Mental
- Screening de riesgos
- Derivación a especialistas
- Seguimiento de casos
- Indicadores de prevalencia
- ROI de programas

MÓDULO 5: Sistema de Encuestas Inteligente

2.5.1 Descripción

Motor de encuestas flexible y potente que soporta múltiples canales y casos de uso.

2.5.2 Funcionalidades

A. Constructor de Encuestas
- Editor drag & drop
- Tipos de pregunta:
- Opción única / múltiple
- Escala Likert
- NPS
- Matriz
- Ranking
- Texto abierto
- Fecha/hora
- Upload de archivos
- Firma digital
- Lógica condicional (branching)
- Piping de respuestas
- Validaciones personalizadas
- Randomización

B. Multi-Canal
- Encuestas web responsive
- App móvil nativa
- SMS bidireccional
- WhatsApp Business API
- Email con link único
- Kioscos/tablets

C. Modo Offline (PWA)
- Sincronización automática
- Almacenamiento local seguro
- Indicador de conectividad
- Resolución de conflictos

D. Templates por Industria
- Minería
- Retail
- Banca y finanzas
- Salud
- Educación
- Sector público

MÓDULO 6: KPIs y Analytics Avanzado

2.6.1 Descripción

Motor de inteligencia de negocio con dashboards, KPIs configurables y capacidades de IA/ML.

2.6.2 Funcionalidades

A. Dashboard Ejecutivo
- Visualizaciones interactivas
- Widgets configurables

- Filtros dinámicos
- Drill-down multinivel
- Vista móvil optimizada

B. KPIs Configurables
- Biblioteca de KPIs por normativa
- Constructor de indicadores
- Metas y umbrales
- Alertas automáticas
- Histórico y tendencias

C. Analytics con IA/ML
- Análisis de sentimiento (NLP)
- Predicción de rotación
- Detección de anomalías
- Clustering de respuestas
- Recomendaciones automatizadas

D. Benchmarking
- Comparativas sectoriales
- Ranking por industria
- Análisis de gaps
- Best practices

MÓDULO 7: Reportería y Auditoría

2.7.1 Descripción

Sistema completo de generación de reportes con trazabilidad para auditorías.

2.7.2 Funcionalidades

A. Generador de Reportes
- Reportes regulatorios predefinidos
- Constructor de reportes ad-hoc
- Programación de reportes
- Distribución automática
- Formatos: PDF, Excel, Word, CSV, JSON

B. Audit Trail
- Registro de todas las acciones
- Quién, qué, cuándo, desde dónde
- Inmutabilidad de registros
- Exportación para fiscalizadores
- Retención configurable

C. Evidencias Documentales
- Repositorio de evidencias
- Metadatos automáticos
- Cadena de custodia digital
- Hash de integridad
- Exportación forense

MÓDULO 8: Gestión Documental

2.8.1 Descripción

Repositorio centralizado de documentos con versionamiento y firma electrónica.

2.8.2 Funcionalidades

A. Repositorio de Políticas
- Estructura jerárquica
- Categorización por tipo
- Búsqueda full-text
- Metadatos personalizables
- Difusión a colaboradores

B. Versionamiento
- Control de versiones
- Comparación de versiones
- Workflow de aprobación
- Rollback
- Historial de cambios

C. Firma Electrónica
- Firma electrónica simple
- Firma electrónica avanzada
- Integración con proveedores (DocuSign, firmavirtual.cl)
- Trazabilidad de firmas
- Validación de integridad

MÓDULO 9: Capacitación y Formación

2.9.1 Descripción

Módulo LMS para gestionar capacitaciones obligatorias y voluntarias.

2.9.2 Funcionalidades

A. E-Learning
- Cursos SCORM/xAPI
- Videos interactivos
- Evaluaciones
- Progreso en tiempo real
- Gamificación

B. Capacitaciones Obligatorias
- Tracking de cumplimiento
- Recordatorios automáticos
- Reportes a organismos
- Certificaciones automáticas
- Vencimientos y renovaciones

C. Biblioteca de Contenidos
- Contenidos Ley Karin

- Contenidos EDI
- Contenidos de clima
- Contenidos personalizables
- Multi-idioma

MÓDULO 10: Integraciones y API

2.10.1 Descripción

Capa de integración para conectar con sistemas externos.

2.10.2 Funcionalidades

A. API REST
- Documentación OpenAPI 3.0
- Sandbox de pruebas
- Rate limiting configurable
- Versionamiento de API
- SDK en múltiples lenguajes

B. Webhooks
- Eventos configurables
- Reintentos automáticos
- Logs de entregas
- Firma de payloads
- Dashboard de monitoreo

C. Conectores HRIS
- SAP SuccessFactors
- Workday
- Oracle HCM
- BUK (Chile)
- Talana (Chile)

D. SSO
- SAML 2.0
- OAuth 2.0 / OIDC
- Azure AD
- Google Workspace
- Okta

2.2 Requisitos Funcionales Priorizados

Must Have (MVP)

ID	Requisito	Módulo
RF-001	Canal de denuncias anónimo multicanal	Ley Karin
RF-002	Workflow de investigación con perspectiva de género	Ley Karin
RF-003	Reportes regulatorios SUSESO/ISL/DT	Ley Karin
RF-004	Gestión de protocolos de prevención	Ley Karin
RF-005	Cuestionario SUSESO/ISTAS21	Ley Karin
RF-006	Constructor de encuestas básico	Encuestas
RF-007	Dashboard de KPIs principales	Analytics
RF-008	Autenticación SSO (SAML/OAuth)	Core
RF-009	Multi-tenancy con aislamiento de datos	Core
RF-010	Audit trail completo	Auditoría

Should Have (Fase 2)

ID	Requisito	Módulo
RF-011	Encuestas de clima organizacional	Clima
RF-012	Gestión de comunidades y stakeholders	Comunidades
RF-013	Indicadores EDI y brecha salarial	EDI
RF-014	Módulo e-learning básico	Capacitación
RF-015	Integración WhatsApp para encuestas	Encuestas
RF-016	PWA para encuestas offline	Encuestas
RF-017	API REST pública	Integraciones
RF-018	Firma electrónica simple	Documentos
RF-019	Benchmarking sectorial	Analytics
RF-020	Alertas y notificaciones configurables	Core

Nice to Have (Fase 3+)

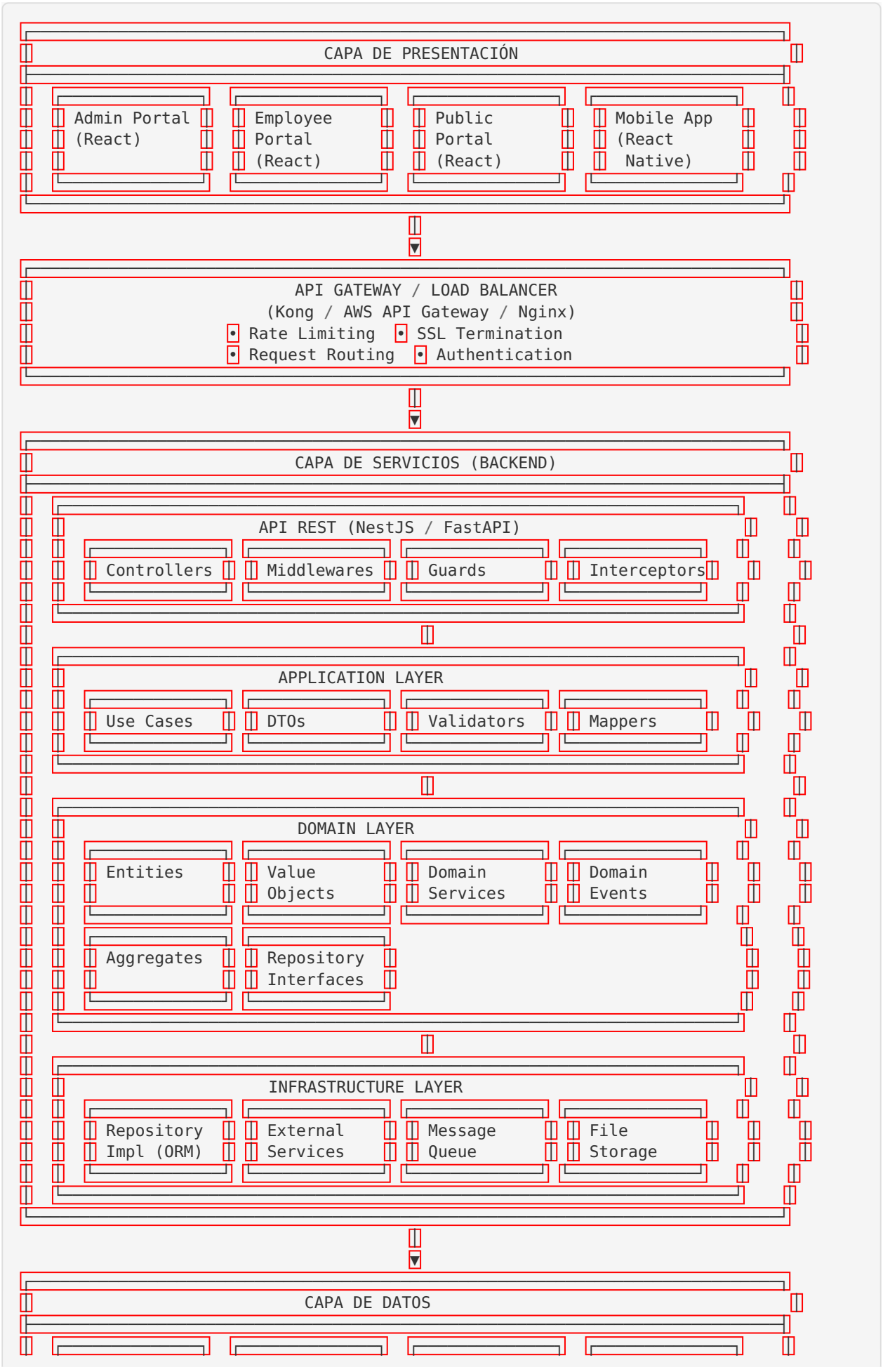
ID	Requisito	Módulo
RF-021	IA predictiva de riesgos	Analytics
RF-022	Análisis de sentimiento NLP	Analytics
RF-023	Integración SAP/Workday	Integraciones
RF-024	App móvil nativa	Mobile
RF-025	Gamificación en capacitaciones	Capacitación
RF-026	Chatbot de asistencia	Core
RF-027	Firma electrónica avanzada	Documentos
RF-028	Multi-idioma completo	Core

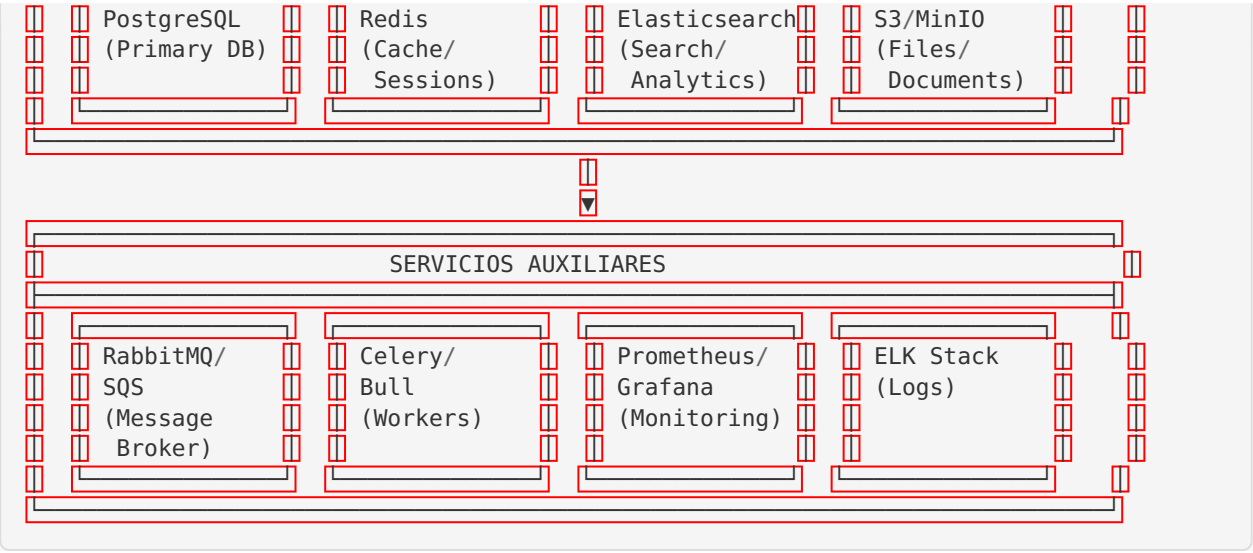
---

3. ARQUITECTURA DE SOFTWARE

3.1 Arquitectura General

3.1.1 Diagrama de Alto Nivel (Descripción Textual)

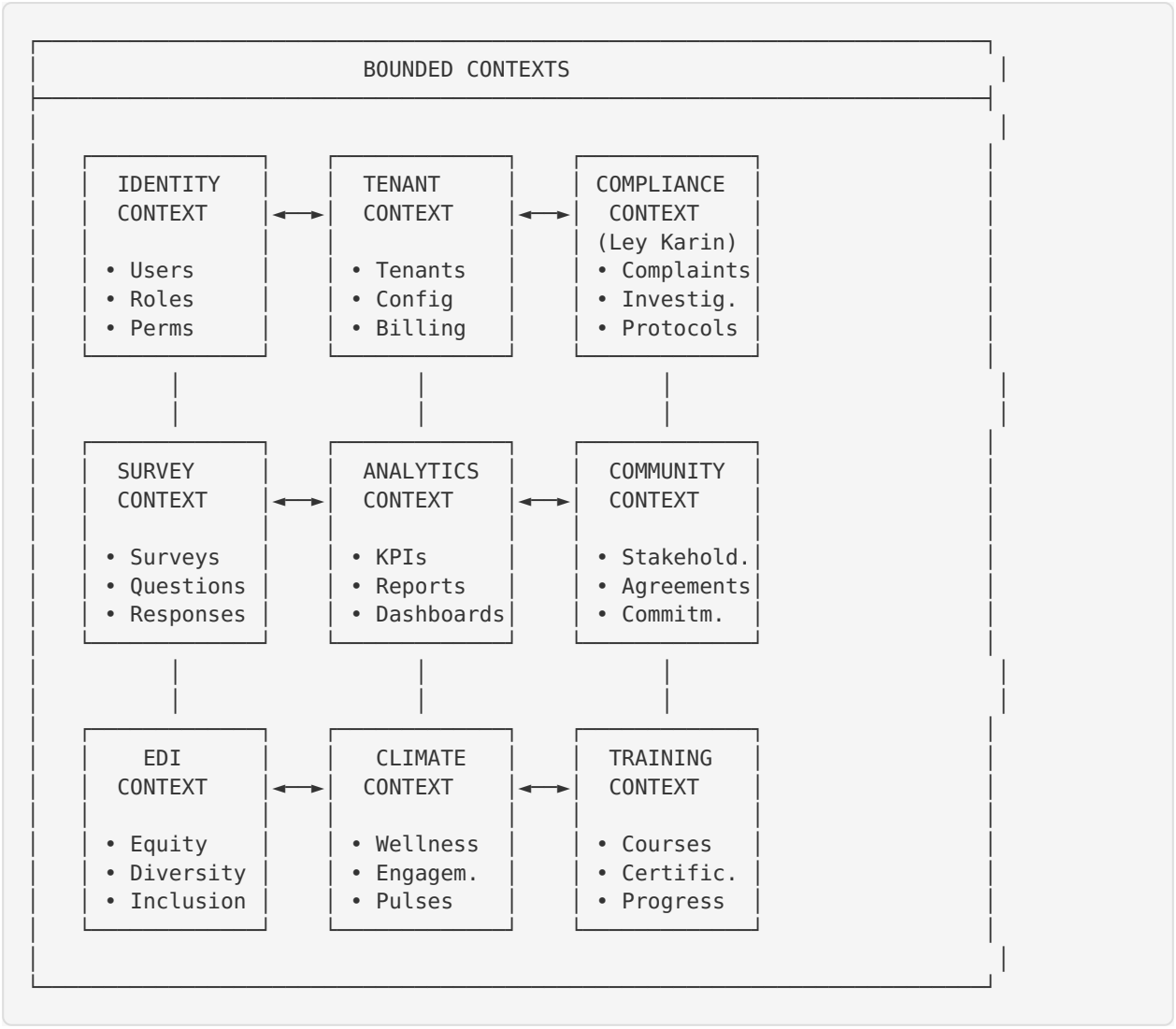

CAPA DE PRESENTACIÓN
Admin Portal (React)
Employee Portal (React)
Public Portal (React)
Mobile App (React Native)
API GATEWAY / LOAD BALANCER
(Kong / AWS API Gateway / Nginx)
• Rate Limiting • SSL Termination
• Request Routing • Authentication
CAPA DE SERVICIOS (BACKEND)
API REST (NestJS / FastAPI)
Controllers
Middlewares
Guards
Interceptors
APPLICATION LAYER
Use Cases
DTOs
Validators
Mappers
DOMAIN LAYER
Entities
Value Objects
Domain Services
Domain Events
Aggregates
Repository Interfaces
INFRASTRUCTURE LAYER
Repository Impl (ORM)
External Services
Message Queue
File Storage
CAPA DE DATOS



3.1.2 Principios Arquitectónicos

Principio	Descripción	Aplicación
SOLID	Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion	Cada clase/módulo tiene una única responsabilidad. Extensión via interfaces.
Clean Architecture	Independencia de frameworks, UI, DB. Testeabilidad.	Capas concéntricas con dependencias hacia el centro (Domain).
Domain-Driven Design	Modelado del dominio con lenguaje ubicuo	Bounded contexts por módulo funcional. Agregados y entidades de dominio.
CQRS	Separación de comandos y consultas	Segregación de operaciones de lectura y escritura para analytics.
Event-Driven	Comunicación mediante eventos	Eventos de dominio para desacoplamiento entre módulos.

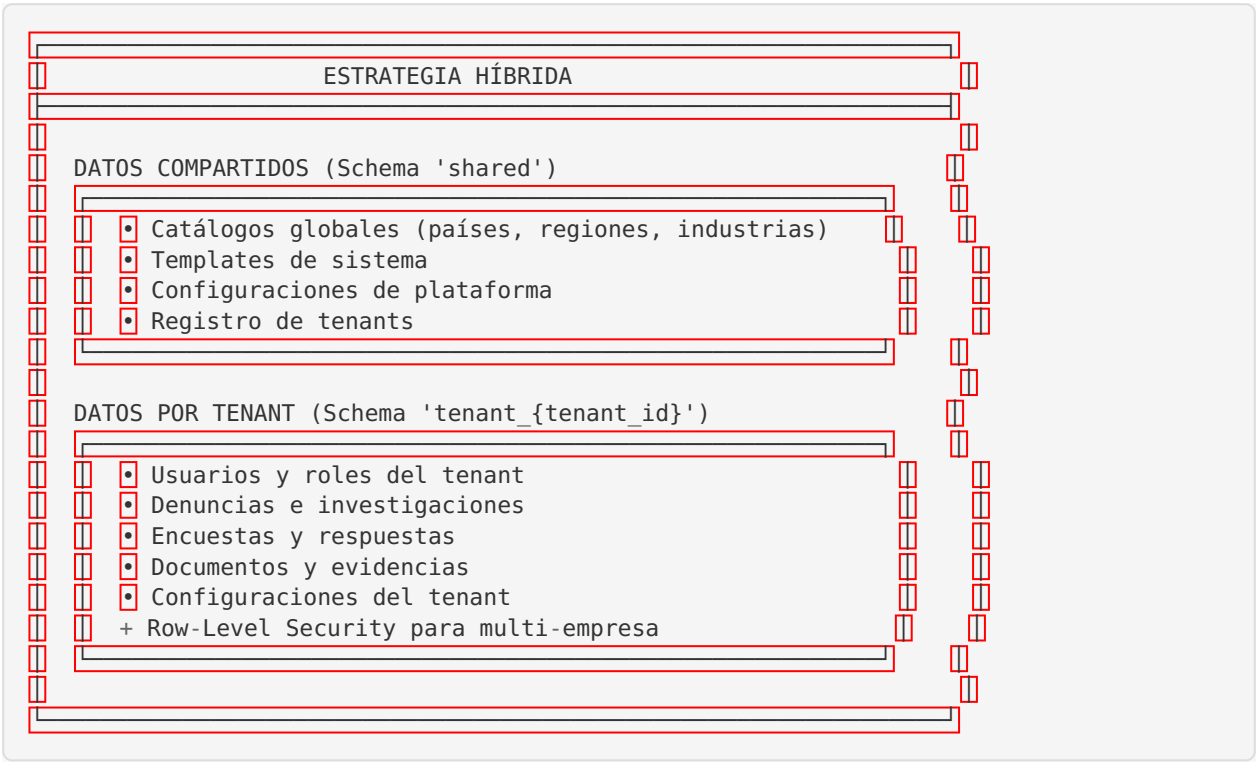
3.1.3 Bounded Contexts (DDD)



3.2 Arquitectura Multi-Tenant

3.2.1 Estrategia de Aislamiento

Modelo Seleccionado: Schema por Tenant + Row-Level Security


ESTRATEGIA HÍBRIDA
DATOS COMPARTIDOS (Schema 'shared')
• Catálogos globales (países, regiones, industrias)
• Templates de sistema
• Configuraciones de plataforma
• Registro de tenants
DATOS POR TENANT (Schema 'tenant_{tenant_id}')
• Usuarios y roles del tenant
• Denuncias e investigaciones
• Encuestas y respuestas
• Documentos y evidencias
• Configuraciones del tenant
+ Row-Level Security para multi-empresa

3.2.2 Resolución de Tenant

```typescript
// Middleware de resolución de tenant
@Injectable()
export class TenantMiddleware implements NestMiddleware {
  constructor(
    private readonly tenantService: TenantService,
    private readonly configService: ConfigService,
  ) {}

  async use(req: Request, res: Response, next: NextFunction) {
    // 1. Extraer tenant de múltiples fuentes (prioridad)
    const tenantId = this.extractTenantId(req);

    if (!tenantId) {
      throw new UnauthorizedException('Tenant not identified');
    }

    // 2. Validar tenant activo
    const tenant = await this.tenantService.validateTenant(tenantId);

    if (!tenant || !tenant.isActive) {
      throw new ForbiddenException('Tenant inactive or not found');
    }

    // 3. Inyectar contexto de tenant en request
    req['tenantContext'] = {
      tenantId: tenant.id,
      schemaName: `tenant_${tenant.id}`,
      config: tenant.config,
      features: tenant.features,
    };

    // 4. Configurar schema de base de datos
    await this.setDatabaseSchema(tenant.id);

    next();
  }

  private extractTenantId(req: Request): string | null {
    // Prioridad 1: Header X-Tenant-ID
    const headerTenant = req.headers['x-tenant-id'] as string;
    if (headerTenant) return headerTenant;

    // Prioridad 2: Subdominio
    const host = req.headers.host || '';
    const subdomain = host.split('.')[0];
    if (subdomain && subdomain !== 'www' && subdomain !== 'app') {
      return subdomain;
    }

    // Prioridad 3: JWT claim
    const token = req.headers.authorization?.replace('Bearer ', '');
    if (token) {
      const decoded = this.jwtService.decode(token) as any;
      if (decoded?.tenantId) return decoded.tenantId;
    }

    // Prioridad 4: Query param (solo para desarrollo)
    if (this.configService.get('NODE_ENV') === 'development') {
      return req.query.tenant as string;
    }

    return null;
```

```
  }
}
```

3.2.3 Configuración PostgreSQL con Row-Level Security

```
-- Habilitar RLS en tablas sensibles
ALTER TABLE complaints ENABLE ROW LEVEL SECURITY;

-- Política de aislamiento por tenant
CREATE POLICY tenant_isolation_policy ON complaints
  USING (tenant_id = current_setting('app.current_tenant_id')::uuid);

-- Política de aislamiento por empresa (dentro del tenant)
CREATE POLICY company_isolation_policy ON complaints
  USING (
    company_id IN (
      SELECT company_id
      FROM user_company_access
      WHERE user_id = current_setting('app.current_user_id')::uuid
    )
  );

-- Función para configurar contexto de sesión
CREATE OR REPLACE FUNCTION set_tenant_context(
  p_tenant_id uuid,
  p_user_id uuid
) RETURNS void AS $$
BEGIN
  PERFORM set_config('app.current_tenant_id', p_tenant_id::text, true);
  PERFORM set_config('app.current_user_id', p_user_id::text, true);
END;
$$ LANGUAGE plpgsql;
```

3.2.4 Consideraciones de Seguridad Multi-Tenant

Aspecto	Implementación
Aislamiento de datos	Schema separation + RLS + validación en application layer
Encriptación	Claves de encriptación únicas por tenant (key per tenant)
Auditoría	Logs segregados por tenant, inmutables
Backups	Backups por tenant con posibilidad de restore individual
Rate Limiting	Límites configurables por tenant según plan
Feature Flags	Habilitación de features por tenant

3.3 Arquitectura Backend

3.3.1 Estructura de Capas

```
src/
├── main.ts                          # Entry point
├── app.module.ts                    # Root module
│
├── core/                            # CORE (Cross-cutting concerns)
│   ├── config/                      # Configuración
│   │   ├── database.config.ts
│   │   ├── redis.config.ts
│   │   ├── jwt.config.ts
│   │   └── app.config.ts
│   ├── guards/                      # Guards de autenticación/autorización
│   │   ├── jwt-auth.guard.ts
│   │   ├── roles.guard.ts
│   │   ├── permissions.guard.ts
│   │   └── tenant.guard.ts
│   ├── interceptors/                # Interceptors
│   │   ├── logging.interceptor.ts
│   │   ├── transform.interceptor.ts
│   │   ├── timeout.interceptor.ts
│   │   └── cache.interceptor.ts
│   ├── filters/                     # Exception filters
│   │   ├── http-exception.filter.ts
│   │   ├── validation.filter.ts
│   │   └── business-exception.filter.ts
│   ├── decorators/                  # Custom decorators
│   │   ├── current-user.decorator.ts
│   │   ├── current-tenant.decorator.ts
│   │   ├── permissions.decorator.ts
│   │   └── api-paginated.decorator.ts
│   ├── middleware/                  # Middlewares
│   │   ├── tenant.middleware.ts
│   │   ├── correlation-id.middleware.ts
│   │   └── request-logging.middleware.ts
│   └── pipes/                       # Validation pipes
│       ├── validation.pipe.ts
│       └── parse-uuid.pipe.ts
│
├── shared/                          # SHARED (Utilidades compartidas)
│   ├── dto/                         # DTOs base
│   │   ├── pagination.dto.ts
│   │   ├── api-response.dto.ts
│   │   └── base-entity.dto.ts
│   ├── interfaces/                  # Interfaces compartidas
│   │   ├── repository.interface.ts
│   │   ├── service.interface.ts
│   │   └── paginated-result.interface.ts
│   ├── utils/                       # Utilidades
│   │   ├── date.utils.ts
│   │   ├── string.utils.ts
│   │   ├── crypto.utils.ts
│   │   └── validation.utils.ts
│   ├── constants/                   # Constantes
│   │   ├── error-codes.ts
│   │   ├── permissions.ts
│   │   └── regex-patterns.ts
│   └── exceptions/                  # Excepciones personalizadas
│       ├── business.exception.ts
│       ├── validation.exception.ts
│       └── domain.exception.ts
│
├── infrastructure/                   # INFRASTRUCTURE LAYER
│   ├── database/                    # Configuración de BD
```

```
│   │   ├── database.module.ts
│   │   ├── typeorm.config.ts
│   │   ├── migrations/
│   │   └── seeds/
│   ├── cache/                          # Redis/Cache
│   │   ├── cache.module.ts
│   │   └── cache.service.ts
│   ├── queue/                          # Message queues
│   │   ├── queue.module.ts
│   │   ├── processors/
│   │   └── producers/
│   ├── storage/                        # File storage
│   │   ├── storage.module.ts
│   │   ├── storage.service.ts
│   │   └── providers/
│   ├── email/                          # Email service
│   │   ├── email.module.ts
│   │   ├── email.service.ts
│   │   └── templates/
│   ├── external/                       # External services
│   │   ├── whatsapp/
│   │   ├── sms/
│   │   └── signature/
│   └── search/                         # Elasticsearch
│       ├── search.module.ts
│       └── search.service.ts
│
└── modules/                            # DOMAIN MODULES (Bounded Contexts)
    │
    ├── identity/                       # Identity Context
    │   ├── identity.module.ts
    │   ├── application/                # Application layer
    │   │   ├── commands/
    │   │   │   ├── create-user.command.ts
    │   │   │   ├── update-user.command.ts
    │   │   │   └── handlers/
    │   │   ├── queries/
    │   │   │   ├── get-user.query.ts
    │   │   │   └── handlers/
    │   │   ├── dto/
    │   │   │   ├── create-user.dto.ts
    │   │   │   ├── update-user.dto.ts
    │   │   │   └── user-response.dto.ts
    │   │   └── services/
    │   │       └── auth.service.ts
    │   ├── domain/                     # Domain layer
    │   │   ├── entities/
    │   │   │   ├── user.entity.ts
    │   │   │   └── role.entity.ts
    │   │   ├── value-objects/
    │   │   │   ├── email.vo.ts
    │   │   │   └── password.vo.ts
    │   │   ├── events/
    │   │   │   ├── user-created.event.ts
    │   │   │   └── user-updated.event.ts
    │   │   ├── repositories/
    │   │   │   └── user.repository.interface.ts
    │   │   └── services/
    │   │       └── password-hasher.interface.ts
    │   ├── infrastructure/             # Infrastructure impl
    │   │   ├── persistence/
    │   │   │   ├── user.repository.ts
    │   │   │   └── user.schema.ts
```

```
│   │   └── services/
│   │       └── bcrypt-password-hasher.ts
│   └── presentation/            # Presentation layer (Controllers)
│       ├── controllers/
│       │   ├── auth.controller.ts
│       │   └── users.controller.ts
│       └── guards/
│           └── local-auth.guard.ts
│
├── compliance/                  # Compliance Context (Ley Karin)
│   ├── compliance.module.ts
│   ├── application/
│   │   ├── commands/
│   │   │   ├── create-complaint.command.ts
│   │   │   ├── assign-investigator.command.ts
│   │   │   ├── close-investigation.command.ts
│   │   │   └── handlers/
│   │   ├── queries/
│   │   │   ├── get-complaints.query.ts
│   │   │   ├── get-investigation.query.ts
│   │   │   └── handlers/
│   │   ├── dto/
│   │   │   ├── create-complaint.dto.ts
│   │   │   ├── complaint-response.dto.ts
│   │   │   └── investigation.dto.ts
│   │   └── services/
│   │       ├── complaint.service.ts
│   │       ├── investigation.service.ts
│   │       └── protocol.service.ts
│   ├── domain/
│   │   ├── entities/
│   │   │   ├── complaint.entity.ts
│   │   │   ├── investigation.entity.ts
│   │   │   ├── protocol.entity.ts
│   │   │   └── psychosocial-risk.entity.ts
│   │   ├── value-objects/
│   │   │   ├── complaint-type.vo.ts
│   │   │   ├── investigation-status.vo.ts
│   │   │   └── tracking-token.vo.ts
│   │   ├── aggregates/
│   │   │   └── complaint.aggregate.ts
│   │   ├── events/
│   │   │   ├── complaint-created.event.ts
│   │   │   ├── investigation-started.event.ts
│   │   │   └── investigation-closed.event.ts
│   │   ├── repositories/
│   │   │   ├── complaint.repository.interface.ts
│   │   │   └── investigation.repository.interface.ts
│   │   └── services/
│   │       ├── anonymizer.service.interface.ts
│   │       └── conflict-of-interest.service.ts
│   ├── infrastructure/
│   │   ├── persistence/
│   │   │   ├── complaint.repository.ts
│   │   │   ├── complaint.schema.ts
│   │   │   └── investigation.repository.ts
│   │   └── services/
│   │       └── token-anonymizer.service.ts
│   └── presentation/
│       ├── controllers/
│       │   ├── complaints.controller.ts
│       │   ├── investigations.controller.ts
│       │   └── protocols.controller.ts
```

```
│       └── public/
│           └── anonymous-complaint.controller.ts
│
├── surveys/                    # Survey Context
│   ├── surveys.module.ts
│   ├── application/
│   ├── domain/
│   ├── infrastructure/
│   └── presentation/
│
├── analytics/                  # Analytics Context
│   ├── analytics.module.ts
│   ├── application/
│   ├── domain/
│   ├── infrastructure/
│   └── presentation/
│
├── community/                  # Community Relations Context
│   └── ...
│
├── edi/                        # EDI Context
│   └── ...
│
├── climate/                    # Climate Context
│   └── ...
│
├── training/                   # Training Context
│   └── ...
│
└── documents/                  # Documents Context
    └── ...
```

3.3.2 Patrón MVC + Clean Architecture

```
// ============================================
// PRESENTATION LAYER (Controller - Vista/API)
// ============================================
@Controller('api/v1/complaints')
@UseGuards(JwtAuthGuard, PermissionsGuard)
@ApiTags('Complaints')
export class ComplaintsController {
  constructor(
    private readonly commandBus: CommandBus,
    private readonly queryBus: QueryBus,
  ) {}

  @Post()
  @Permissions('complaints.create')
  @ApiOperation({ summary: 'Create new complaint' })
  @ApiResponse({ status: 201, type: ComplaintResponseDto })
  async create(
    @Body() dto: CreateComplaintDto,
    @CurrentUser() user: UserContext,
    @CurrentTenant() tenant: TenantContext,
  ): Promise<ApiResponse<ComplaintResponseDto>> {
    const command = new CreateComplaintCommand({
      ...dto,
      tenantId: tenant.id,
      createdBy: user.id,
    });

    const result = await this.commandBus.execute(command);

    return ApiResponse.created(result);
  }

  @Get()
  @Permissions('complaints.read')
  @ApiPaginatedResponse(ComplaintResponseDto)
  async findAll(
    @Query() pagination: PaginationDto,
    @Query() filters: ComplaintFiltersDto,
    @CurrentTenant() tenant: TenantContext,
  ): Promise<PaginatedResponse<ComplaintResponseDto>> {
    const query = new GetComplaintsQuery({
      tenantId: tenant.id,
      ...pagination,
      ...filters,
    });

    return this.queryBus.execute(query);
  }
}

// ============================================
// APPLICATION LAYER (Service/Use Case)
// ============================================
@CommandHandler(CreateComplaintCommand)
export class CreateComplaintHandler
  implements ICommandHandler<CreateComplaintCommand>
{
  constructor(
    private readonly complaintRepository: IComplaintRepository,
    private readonly anonymizerService: IAnonymizerService,
    private readonly eventBus: EventBus,
    private readonly notificationService: NotificationService,
```

```
  ) {}

  async execute(command: CreateComplaintCommand): Promise<ComplaintResponseDto> {
    // 1. Crear entidad de dominio
    const complaint = Complaint.create({
      type: ComplaintType.fromString(command.type),
      description: command.description,
      isAnonymous: command.isAnonymous,
      tenantId: command.tenantId,
      reporterId: command.isAnonymous ? null : command.reporterId,
    });

    // 2. Generar token de seguimiento si es anónima
    if (command.isAnonymous) {
      const token = await this.anonymizerService.generateTrackingToken();
      complaint.setTrackingToken(token);
    }

    // 3. Validar reglas de negocio de dominio
    complaint.validate();

    // 4. Persistir
    const saved = await this.complaintRepository.save(complaint);

    // 5. Publicar evento de dominio
    this.eventBus.publish(new ComplaintCreatedEvent(saved));

    // 6. Notificar a responsables
    await this.notificationService.notifyComplaintReceivers(saved);

    // 7. Retornar DTO
    return ComplaintMapper.toResponseDto(saved);
  }
}

// ============================================
// DOMAIN LAYER (Model/Entity)
// ============================================
export class Complaint extends AggregateRoot {
  private _id: ComplaintId;
  private _type: ComplaintType;
  private _description: string;
  private _status: ComplaintStatus;
  private _isAnonymous: boolean;
  private _trackingToken: TrackingToken | null;
  private _tenantId: TenantId;
  private _reporterId: UserId | null;
  private _createdAt: Date;
  private _updatedAt: Date;

  private constructor(props: ComplaintProps) {
    super();
    this._id = props.id || ComplaintId.generate();
    this._type = props.type;
    this._description = props.description;
    this._status = ComplaintStatus.RECEIVED;
    this._isAnonymous = props.isAnonymous;
    this._trackingToken = null;
    this._tenantId = props.tenantId;
    this._reporterId = props.reporterId;
    this._createdAt = new Date();
    this._updatedAt = new Date();
  }
```

```
  static create(props: CreateComplaintProps): Complaint {
    // Validaciones de creación
    if (!props.description || props.description.trim().length < 10) {
      throw new DomainException('Description must be at least 10 characters');
    }

    return new Complaint(props);
  }

  // Reglas de negocio encapsuladas en el dominio
  assignInvestigator(investigatorId: UserId, assignedBy: UserId): void {
    if (this._status !== ComplaintStatus.RECEIVED) {
      throw new DomainException('Cannot assign investigator to non-received com-
plaint');
    }

    // Validar que no haya conflicto de interés
    if (this._reporterId && this._reporterId.equals(investigatorId)) {
      throw new DomainException('Investigator cannot be the same as reporter');
    }

    this._status = ComplaintStatus.UNDER_INVESTIGATION;
    this._updatedAt = new Date();

    this.addDomainEvent(new InvestigatorAssignedEvent({
      complaintId: this._id,
      investigatorId,
      assignedBy,
    }));
  }

  validate(): void {
    const errors: string[] = [];

    if (!ComplaintType.isValid(this._type)) {
      errors.push('Invalid complaint type');
    }

    if (this._description.length > 5000) {
      errors.push('Description exceeds maximum length');
    }

    if (errors.length > 0) {
      throw new ValidationException(errors);
    }
  }

  // Getters (inmutabilidad)
  get id(): ComplaintId { return this._id; }
  get type(): ComplaintType { return this._type; }
  get status(): ComplaintStatus { return this._status; }
  // ...
}

// Value Object
export class ComplaintType extends ValueObject<string> {
  private static readonly VALID_TYPES = [
    'WORKPLACE_HARASSMENT',
    'SEXUAL_HARASSMENT',
    'WORKPLACE_VIOLENCE',
    'THIRD_PARTY_VIOLENCE',
  ];
```

```
  private constructor(value: string) {
    super(value);
  }

  static fromString(value: string): ComplaintType {
    if (!this.isValid(value)) {
      throw new DomainException(`Invalid complaint type: ${value}`);
    }
    return new ComplaintType(value);
  }

  static isValid(value: string): boolean {
    return this.VALID_TYPES.includes(value);
  }

  isWorkplaceRelated(): boolean {
    return this.value === 'WORKPLACE_HARASSMENT' || this.value === 'WORK-
PLACE_VIOLENCE';
  }
}

// ============================================
// INFRASTRUCTURE LAYER (Repository Implementation)
// ============================================
@Injectable()
export class ComplaintRepository implements IComplaintRepository {
  constructor(
    @InjectRepository(ComplaintEntity)
    private readonly repository: Repository<ComplaintEntity>,
    private readonly eventBus: EventBus,
  ) {}

  async save(complaint: Complaint): Promise<Complaint> {
    const entity = ComplaintMapper.toEntity(complaint);
    const saved = await this.repository.save(entity);

    // Publicar eventos de dominio pendientes
    const events = complaint.getDomainEvents();
    for (const event of events) {
      await this.eventBus.publish(event);
    }
    complaint.clearDomainEvents();

    return ComplaintMapper.toDomain(saved);
  }

  async findById(id: ComplaintId): Promise<Complaint | null> {
    const entity = await this.repository.findOne({
      where: { id: id.value },
      relations: ['investigation', 'attachments'],
    });

    return entity ? ComplaintMapper.toDomain(entity) : null;
  }

  async findByTenant(
    tenantId: TenantId,
    options: FindComplaintsOptions,
  ): Promise<PaginatedResult<Complaint>> {
    const queryBuilder = this.repository
      .createQueryBuilder('complaint')
      .where('complaint.tenant_id = :tenantId', { tenantId: tenantId.value });
```

```
    // Aplicar filtros
    if (options.status) {
      queryBuilder.andWhere('complaint.status = :status', { status: options.status });
    }

    if (options.type) {
      queryBuilder.andWhere('complaint.type = :type', { type: options.type });
    }

    if (options.dateFrom) {
      queryBuilder.andWhere('complaint.created_at >= :dateFrom', { dateFrom: op-
tions.dateFrom });
    }

    // Paginación
    const [entities, total] = await queryBuilder
      .orderBy('complaint.created_at', 'DESC')
      .skip(options.offset)
      .take(options.limit)
      .getManyAndCount();

    return {
      data: entities.map(ComplaintMapper.toDomain),
      total,
      page: options.page,
      pageSize: options.limit,
      totalPages: Math.ceil(total / options.limit),
    };
  }
}
```

3.3.3 API REST Design

Convenciones de Endpoints

```
# Base URL: https://api.compliancehub.com/api/v1

# Recursos principales
/tenants                    # Gestión de tenants (super admin)
/auth                       # Autenticación
/users                      # Usuarios
/roles                      # Roles y permisos
/complaints                 # Denuncias
/investigations             # Investigaciones
/protocols                  # Protocolos de prevención
/surveys                    # Encuestas
/responses                  # Respuestas de encuestas
/kpis                       # Indicadores
/reports                    # Reportes
/documents                  # Documentos
/training                   # Capacitaciones

# Convenciones:
# - Usar sustantivos en plural
# - Recursos anidados para relaciones claras
# - Query params para filtros, ordenamiento, paginación
# - Verbos HTTP para acciones (GET, POST, PUT, PATCH, DELETE)
```

Endpoints Principales por Módulo

```
# ========================================
# AUTENTICACIÓN Y USUARIOS
# ========================================
POST   /auth/login                    # Login con credenciales
POST   /auth/refresh                  # Refresh token
POST   /auth/logout                   # Logout
POST   /auth/forgot-password          # Solicitar reset password
POST   /auth/reset-password           # Ejecutar reset password
POST   /auth/mfa/setup                # Configurar 2FA
POST   /auth/mfa/verify               # Verificar código 2FA

GET    /users                         # Listar usuarios (admin)
POST   /users                         # Crear usuario
GET    /users/:id                     # Obtener usuario
PUT    /users/:id                     # Actualizar usuario
DELETE /users/:id                     # Eliminar usuario (soft delete)
GET    /users/me                      # Usuario actual
PUT    /users/me                      # Actualizar perfil propio

# ========================================
# LEY KARIN - DENUNCIAS
# ========================================
# Endpoint público (sin auth, para denuncias anónimas)
POST   /public/complaints             # Crear denuncia anónima
GET    /public/complaints/:token      # Consultar estado por token

# Endpoints autenticados
GET    /complaints                    # Listar denuncias
POST   /complaints                    # Crear denuncia (identificada)
GET    /complaints/:id                # Obtener denuncia
PUT    /complaints/:id                # Actualizar denuncia
GET    /complaints/:id/timeline       # Timeline de eventos
POST   /complaints/:id/attachments    # Adjuntar evidencia
GET    /complaints/:id/attachments    # Listar adjuntos
GET    /complaints/export             # Exportar denuncias (CSV/Excel)

# Filtros disponibles (query params):
# ?status=RECEIVED,UNDER_INVESTIGATION,CLOSED
# ?type=WORKPLACE_HARASSMENT,SEXUAL_HARASSMENT
# ?dateFrom=2024-01-01&dateTo=2024-12-31
# ?page=1&pageSize=20&sortBy=createdAt&sortOrder=DESC

# ========================================
# LEY KARIN - INVESTIGACIONES
# ========================================
GET    /investigations                        # Listar investigaciones
POST   /investigations                        # Crear investigación
GET    /investigations/:id                    # Obtener investigación
PUT    /investigations/:id                    # Actualizar investigación
POST   /investigations/:id/assign             # Asignar investigador
POST   /investigations/:id/interviews         # Registrar entrevista
GET    /investigations/:id/interviews         # Listar entrevistas
POST   /investigations/:id/findings           # Agregar hallazgo
POST   /investigations/:id/close              # Cerrar investigación
GET    /investigations/:id/report             # Generar informe
POST   /investigations/:id/measures           # Proponer medidas

# ========================================
# LEY KARIN - PROTOCOLOS
# ========================================
GET    /protocols                     # Listar protocolos
POST   /protocols                     # Crear protocolo
```

```
GET    /protocols/:id                  # Obtener protocolo
PUT    /protocols/:id                  # Actualizar protocolo
POST   /protocols/:id/publish          # Publicar versión
GET    /protocols/:id/versions         # Historial de versiones
POST   /protocols/:id/acknowledge      # Registrar lectura/aceptación
GET    /protocols/:id/acknowledgments # Listar aceptaciones

# ========================================
# LEY KARIN - RIESGOS PSICOSOCIALES
# ========================================
GET    /psychosocial-risks                     # Listar evaluaciones
POST   /psychosocial-risks                     # Crear evaluación
GET    /psychosocial-risks/:id                 # Obtener evaluación
GET    /psychosocial-risks/:id/results         # Resultados por dimensión
GET    /psychosocial-risks/:id/heatmap         # Mapa de calor
POST   /psychosocial-risks/:id/action-plan     # Crear plan de acción

# ========================================
# ENCUESTAS
# ========================================
GET    /surveys                        # Listar encuestas
POST   /surveys                        # Crear encuesta
GET    /surveys/:id                    # Obtener encuesta
PUT    /surveys/:id                    # Actualizar encuesta
DELETE /surveys/:id                    # Eliminar encuesta
POST   /surveys/:id/duplicate          # Duplicar encuesta
POST   /surveys/:id/publish            # Publicar encuesta
POST   /surveys/:id/close              # Cerrar encuesta
GET    /surveys/:id/preview            # Vista previa

# Preguntas de encuesta
GET    /surveys/:id/questions          # Listar preguntas
POST   /surveys/:id/questions          # Crear pregunta
PUT    /surveys/:id/questions/:qid     # Actualizar pregunta
DELETE /surveys/:id/questions/:qid     # Eliminar pregunta
POST   /surveys/:id/questions/reorder # Reordenar preguntas

# Distribución
POST   /surveys/:id/distribute         # Enviar encuesta
GET    /surveys/:id/distributions      # Historial de envíos

# Respuestas (endpoint público para responder)
POST   /public/surveys/:token/responses  # Enviar respuestas
GET    /public/surveys/:token            # Obtener encuesta para responder

# Respuestas (admin)
GET    /surveys/:id/responses            # Listar respuestas
GET    /surveys/:id/responses/:rid       # Ver respuesta individual
GET    /surveys/:id/responses/export     # Exportar respuestas
GET    /surveys/:id/analytics            # Analytics de respuestas

# ========================================
# ANALYTICS Y KPIs
# ========================================
GET    /kpis                           # Listar KPIs configurados
POST   /kpis                           # Crear KPI
GET    /kpis/:id                       # Obtener KPI
PUT    /kpis/:id                       # Actualizar KPI
GET    /kpis/:id/values                # Valores históricos
GET    /kpis/:id/trend                 # Tendencia

GET    /dashboards                     # Listar dashboards
POST   /dashboards                     # Crear dashboard
```

```
GET    /dashboards/:id                # Obtener dashboard
PUT    /dashboards/:id                # Actualizar dashboard
GET    /dashboards/:id/widgets        # Widgets del dashboard

GET    /analytics/summary             # Resumen ejecutivo
GET    /analytics/compliance          # Métricas de compliance
GET    /analytics/climate             # Métricas de clima
GET    /analytics/predictions         # Predicciones IA

# ========================================
# REPORTES
# ========================================
GET    /reports                       # Listar reportes
POST   /reports                       # Generar reporte
GET    /reports/:id                   # Obtener reporte
GET    /reports/:id/download          # Descargar reporte
GET    /reports/templates             # Templates disponibles

# Reportes regulatorios
POST   /reports/regulatory/suseso     # Generar reporte SUSESO
POST   /reports/regulatory/isl        # Generar reporte ISL
POST   /reports/regulatory/dt         # Generar reporte Dir. Trabajo

# ========================================
# DOCUMENTOS
# ========================================
GET    /documents                     # Listar documentos
POST   /documents                     # Subir documento
GET    /documents/:id                 # Obtener documento
PUT    /documents/:id                 # Actualizar metadatos
DELETE /documents/:id                 # Eliminar documento
GET    /documents/:id/download        # Descargar documento
GET    /documents/:id/versions        # Versiones del documento
POST   /documents/:id/sign            # Solicitar firma

# ========================================
# CAPACITACIONES
# ========================================
GET    /courses                       # Listar cursos
POST   /courses                       # Crear curso
GET    /courses/:id                   # Obtener curso
PUT    /courses/:id                   # Actualizar curso
GET    /courses/:id/modules           # Módulos del curso
POST   /courses/:id/enroll            # Inscribir usuarios
GET    /courses/:id/progress          # Progreso de participantes

GET    /my-training                   # Mis capacitaciones
GET    /my-training/:id/progress      # Mi progreso en curso
POST   /my-training/:id/complete      # Completar módulo

# ========================================
# WEBHOOKS
# ========================================
GET    /webhooks                      # Listar webhooks configurados
POST   /webhooks                      # Crear webhook
PUT    /webhooks/:id                  # Actualizar webhook
DELETE /webhooks/:id                  # Eliminar webhook
GET    /webhooks/:id/logs             # Logs de entregas
POST   /webhooks/:id/test             # Probar webhook
```

3.3.4 Autenticación y Autorización

JWT Structure

```typescript
// Access Token Payload
interface AccessTokenPayload {
  sub: string;          // User ID
  email: string;
  tenantId: string;
  companyIds: string[]; // Companies user has access to
  roles: string[];      // Role names
  permissions: string[];// Flattened permissions
  iat: number;          // Issued at
  exp: number;          // Expiration (15 min)
  type: 'access';
}

// Refresh Token Payload
interface RefreshTokenPayload {
  sub: string;
  tenantId: string;
  jti: string;          // Token ID for revocation
  iat: number;
  exp: number;          // Expiration (7 days)
  type: 'refresh';
}
```

RBAC (Role-Based Access Control)

```
// Modelo de permisos granulares
const PERMISSIONS = {
  // Complaints
  'complaints.create': 'Create complaints',
  'complaints.read': 'View complaints',
  'complaints.read.own': 'View own complaints only',
  'complaints.read.all': 'View all complaints',
  'complaints.update': 'Update complaints',
  'complaints.delete': 'Delete complaints',
  'complaints.assign': 'Assign investigators',
  'complaints.export': 'Export complaints data',

  // Investigations
  'investigations.create': 'Create investigations',
  'investigations.read': 'View investigations',
  'investigations.update': 'Update investigations',
  'investigations.close': 'Close investigations',
  'investigations.read.confidential': 'Access confidential details',

  // Surveys
  'surveys.create': 'Create surveys',
  'surveys.read': 'View surveys',
  'surveys.update': 'Update surveys',
  'surveys.delete': 'Delete surveys',
  'surveys.publish': 'Publish surveys',
  'surveys.responses.read': 'View survey responses',
  'surveys.responses.export': 'Export responses',

  // Reports
  'reports.create': 'Generate reports',
  'reports.read': 'View reports',
  'reports.regulatory': 'Generate regulatory reports',

  // Users (Admin)
  'users.create': 'Create users',
  'users.read': 'View users',
  'users.update': 'Update users',
  'users.delete': 'Delete users',
  'users.roles.assign': 'Assign roles',

  // Settings
  'settings.read': 'View settings',
  'settings.update': 'Update settings',
  'tenant.manage': 'Manage tenant configuration',
};

// Roles predefinidos
const ROLES = {
  SUPER_ADMIN: {
    name: 'Super Administrator',
    permissions: ['*'], // All permissions
  },
  TENANT_ADMIN: {
    name: 'Tenant Administrator',
    permissions: [
      'users.*',
      'settings.*',
      'reports.*',
      'complaints.*',
      'investigations.*',
      'surveys.*',
    ],
```

```
  },
  COMPLIANCE_OFFICER: {
    name: 'Compliance Officer',
    permissions: [
      'complaints.*',
      'investigations.*',
      'reports.create',
      'reports.read',
      'reports.regulatory',
      'surveys.read',
      'surveys.responses.read',
    ],
  },
  INVESTIGATOR: {
    name: 'Investigator',
    permissions: [
      'complaints.read.all',
      'investigations.read',
      'investigations.update',
      'investigations.close',
    ],
  },
  HR_MANAGER: {
    name: 'HR Manager',
    permissions: [
      'complaints.read',
      'surveys.*',
      'reports.read',
      'users.read',
    ],
  },
  EMPLOYEE: {
    name: 'Employee',
    permissions: [
      'complaints.create',
      'complaints.read.own',
      'surveys.responses.create',
    ],
  },
};
```

Implementación de Guards

```typescript
@Injectable()
export class PermissionsGuard implements CanActivate {
  constructor(private reflector: Reflector) {}

  canActivate(context: ExecutionContext): boolean {
    const requiredPermissions = this.reflector.getAllAndOverride<string[]>(
      PERMISSIONS_KEY,
      [context.getHandler(), context.getClass()],
    );

    if (!requiredPermissions || requiredPermissions.length === 0) {
      return true;
    }

    const request = context.switchToHttp().getRequest();
    const user = request.user as AuthenticatedUser;

    // Super admin tiene todos los permisos
    if (user.permissions.includes('*')) {
      return true;
    }

    // Verificar si tiene alguno de los permisos requeridos
    return requiredPermissions.some((permission) =>
      this.hasPermission(user.permissions, permission),
    );
  }

  private hasPermission(userPermissions: string[], required: string): boolean {
    // Permiso exacto
    if (userPermissions.includes(required)) {
      return true;
    }

    // Permiso wildcard (e.g., "complaints.*" matches "complaints.create")
    const [resource, action] = required.split('.');
    const wildcardPermission = `${resource}.*`;

    return userPermissions.includes(wildcardPermission);
  }
}
```

3.3.5 Manejo de Errores

```typescript
// Códigos de error estandarizados
enum ErrorCode {
  // Validation (400)
  VALIDATION_ERROR = 'VALIDATION_ERROR',
  INVALID_INPUT = 'INVALID_INPUT',
  MISSING_REQUIRED_FIELD = 'MISSING_REQUIRED_FIELD',

  // Authentication (401)
  INVALID_CREDENTIALS = 'INVALID_CREDENTIALS',
  TOKEN_EXPIRED = 'TOKEN_EXPIRED',
  TOKEN_INVALID = 'TOKEN_INVALID',

  // Authorization (403)
  INSUFFICIENT_PERMISSIONS = 'INSUFFICIENT_PERMISSIONS',
  TENANT_ACCESS_DENIED = 'TENANT_ACCESS_DENIED',
  RESOURCE_ACCESS_DENIED = 'RESOURCE_ACCESS_DENIED',

  // Not Found (404)
  RESOURCE_NOT_FOUND = 'RESOURCE_NOT_FOUND',
  USER_NOT_FOUND = 'USER_NOT_FOUND',
  COMPLAINT_NOT_FOUND = 'COMPLAINT_NOT_FOUND',

  // Conflict (409)
  RESOURCE_ALREADY_EXISTS = 'RESOURCE_ALREADY_EXISTS',
  DUPLICATE_ENTRY = 'DUPLICATE_ENTRY',

  // Business Logic (422)
  BUSINESS_RULE_VIOLATION = 'BUSINESS_RULE_VIOLATION',
  INVALID_STATE_TRANSITION = 'INVALID_STATE_TRANSITION',
  CONFLICT_OF_INTEREST = 'CONFLICT_OF_INTEREST',

  // Rate Limit (429)
  RATE_LIMIT_EXCEEDED = 'RATE_LIMIT_EXCEEDED',

  // Server (500)
  INTERNAL_ERROR = 'INTERNAL_ERROR',
  DATABASE_ERROR = 'DATABASE_ERROR',
  EXTERNAL_SERVICE_ERROR = 'EXTERNAL_SERVICE_ERROR',
}

// Formato de respuesta de error
interface ErrorResponse {
  success: false;
  error: {
    code: ErrorCode;
    message: string;
    details?: Record<string, any>;
    validationErrors?: ValidationError[];
    timestamp: string;
    traceId: string;
    path: string;
  };
}

// Exception Filter global
@Catch()
export class GlobalExceptionFilter implements ExceptionFilter {
  constructor(
    private readonly logger: LoggerService,
    private readonly configService: ConfigService,
  ) {}
```

```
  catch(exception: unknown, host: ArgumentsHost): void {
    const ctx = host.switchToHttp();
    const response = ctx.getResponse<Response>();
    const request = ctx.getRequest<Request>();
    const traceId = request.headers['x-trace-id'] as string || uuid();

    let status = HttpStatus.INTERNAL_SERVER_ERROR;
    let errorResponse: ErrorResponse;

    if (exception instanceof HttpException) {
      status = exception.getStatus();
      const exceptionResponse = exception.getResponse() as any;

      errorResponse = {
        success: false,
        error: {
          code: exceptionResponse.code || this.mapStatusToCode(status),
          message: exceptionResponse.message || exception.message,
          details: exceptionResponse.details,
          validationErrors: exceptionResponse.validationErrors,
          timestamp: new Date().toISOString(),
          traceId,
          path: request.path,
        },
      };
    } else if (exception instanceof DomainException) {
      status = HttpStatus.UNPROCESSABLE_ENTITY;
      errorResponse = {
        success: false,
        error: {
          code: ErrorCode.BUSINESS_RULE_VIOLATION,
          message: exception.message,
          timestamp: new Date().toISOString(),
          traceId,
          path: request.path,
        },
      };
    } else {
      // Error inesperado
      this.logger.error('Unhandled exception', exception, traceId);

      errorResponse = {
        success: false,
        error: {
          code: ErrorCode.INTERNAL_ERROR,
          message: this.configService.get('NODE_ENV') === 'production'
            ? 'An unexpected error occurred'
            : (exception as Error).message,
          timestamp: new Date().toISOString(),
          traceId,
          path: request.path,
        },
      };
    }

    // Log del error
    this.logger.error(
      `[${traceId}] ${request.method} ${request.path} - ${status} - ${errorRes-
ponse.error.code}`,
      { exception, body: request.body, query: request.query },
    );

    response.status(status).json(errorResponse);
```

```
    }
}
```

3.4 Arquitectura Frontend

3.4.1 Estructura de Proyecto (React + TypeScript)

```
frontend/
├── public/
│   ├── index.html
│   ├── manifest.json              # PWA manifest
│   ├── service-worker.js          # Service worker para offline
│   └── assets/
│       └── images/
│
├── src/
│   ├── index.tsx                  # Entry point
│   ├── App.tsx                    # Root component
│   ├── routes.tsx                 # Route definitions
│   │
│   ├── config/                    # Configuration
│   │   ├── api.config.ts
│   │   ├── theme.config.ts
│   │   ├── i18n.config.ts
│   │   └── feature-flags.ts
│   │
│   ├── core/                      # Core functionality
│   │   ├── api/                   # API client
│   │   │   ├── axios-instance.ts
│   │   │   ├── interceptors/
│   │   │   └── error-handler.ts
│   │   ├── auth/                  # Authentication
│   │   │   ├── auth.context.tsx
│   │   │   ├── auth.provider.tsx
│   │   │   ├── use-auth.hook.ts
│   │   │   └── protected-route.tsx
│   │   ├── tenant/                # Multi-tenant
│   │   │   ├── tenant.context.tsx
│   │   │   └── use-tenant.hook.ts
│   │   ├── permissions/           # Authorization
│   │   │   ├── permissions.context.tsx
│   │   │   ├── use-permissions.hook.ts
│   │   │   └── can.component.tsx
│   │   └── error-boundary/
│   │       └── error-boundary.tsx
│   │
│   ├── shared/                    # Shared resources
│   │   ├── components/            # Reusable components
│   │   │   ├── ui/                # Basic UI components
│   │   │   │   ├── Button/
│   │   │   │   ├── Input/
│   │   │   │   ├── Select/
│   │   │   │   ├── Modal/
│   │   │   │   ├── Table/
│   │   │   │   ├── Card/
│   │   │   │   ├── Tabs/
│   │   │   │   ├── Toast/
│   │   │   │   └── index.ts
│   │   │   ├── forms/             # Form components
│   │   │   │   ├── FormField/
│   │   │   │   ├── FormSelect/
│   │   │   │   ├── FormDatePicker/
│   │   │   │   └── FormFileUpload/
│   │   │   ├── layout/            # Layout components
│   │   │   │   ├── Header/
│   │   │   │   ├── Sidebar/
│   │   │   │   ├── Footer/
│   │   │   │   └── PageContainer/
│   │   │   └── data-display/      # Data display
```

```
│   │   │   │   ├── DataTable/
│   │   │   │   ├── KPICard/
│   │   │   │   ├── Chart/
│   │   │   │   └── Timeline/
│   │   │   ├── feedback/          # Feedback components
│   │   │   │   ├── Loading/
│   │   │   │   ├── Empty/
│   │   │   │   ├── Error/
│   │   │   │   └── Confirmation/
│   │   ├── hooks/                 # Custom hooks
│   │   │   ├── use-api.hook.ts
│   │   │   ├── use-debounce.hook.ts
│   │   │   ├── use-pagination.hook.ts
│   │   │   ├── use-form.hook.ts
│   │   │   └── use-local-storage.hook.ts
│   │   ├── utils/                 # Utilities
│   │   │   ├── date.utils.ts
│   │   │   ├── format.utils.ts
│   │   │   ├── validation.utils.ts
│   │   │   └── storage.utils.ts
│   │   ├── types/                 # TypeScript types
│   │   │   ├── api.types.ts
│   │   │   ├── user.types.ts
│   │   │   └── common.types.ts
│   │   └── constants/             # Constants
│   │       ├── routes.ts
│   │       ├── query-keys.ts
│   │       └── status.ts
│   │
│   ├── store/                     # State management (Zustand/Redux)
│   │   ├── index.ts
│   │   ├── slices/
│   │   │   ├── auth.slice.ts
│   │   │   ├── ui.slice.ts
│   │   │   └── notifications.slice.ts
│   │   └── middleware/
│   │
│   ├── services/                  # API services
│   │   ├── auth.service.ts
│   │   ├── users.service.ts
│   │   ├── complaints.service.ts
│   │   ├── investigations.service.ts
│   │   ├── surveys.service.ts
│   │   ├── reports.service.ts
│   │   └── analytics.service.ts
│   │
│   ├── features/                  # Feature modules
│   │   │
│   │   ├── auth/                  # Authentication module
│   │   │   ├── pages/
│   │   │   │   ├── LoginPage.tsx
│   │   │   │   ├── ForgotPasswordPage.tsx
│   │   │   │   └── ResetPasswordPage.tsx
│   │   │   ├── components/
│   │   │   │   ├── LoginForm.tsx
│   │   │   │   └── MFASetup.tsx
│   │   │   └── hooks/
│   │   │       └── use-login.hook.ts
│   │   │
│   │   ├── dashboard/             # Dashboard module
│   │   │   ├── pages/
│   │   │   │   └── DashboardPage.tsx
│   │   │   ├── components/
```

```
│   │   │   │   ├── SummaryCards.tsx
│   │   │   │   ├── RecentComplaints.tsx
│   │   │   │   ├── ComplianceChart.tsx
│   │   │   │   └── AlertsWidget.tsx
│   │   │   └── hooks/
│   │   │       └── use-dashboard-data.hook.ts
│   │   │
│   │   ├── complaints/            # Complaints module
│   │   │   ├── pages/
│   │   │   │   ├── ComplaintsListPage.tsx
│   │   │   │   ├── ComplaintDetailPage.tsx
│   │   │   │   └── CreateComplaintPage.tsx
│   │   │   ├── components/
│   │   │   │   ├── ComplaintsTable.tsx
│   │   │   │   ├── ComplaintForm.tsx
│   │   │   │   ├── ComplaintTimeline.tsx
│   │   │   │   ├── ComplaintFilters.tsx
│   │   │   │   └── AnonymityToggle.tsx
│   │   │   └── hooks/
│   │   │       ├── use-complaints.hook.ts
│   │   │       └── use-complaint-form.hook.ts
│   │   │
│   │   ├── investigations/        # Investigations module
│   │   │   ├── pages/
│   │   │   │   ├── InvestigationsListPage.tsx
│   │   │   │   └── InvestigationDetailPage.tsx
│   │   │   ├── components/
│   │   │   │   ├── InvestigationWorkflow.tsx
│   │   │   │   ├── InterviewForm.tsx
│   │   │   │   ├── FindingsPanel.tsx
│   │   │   │   └── MeasuresProposal.tsx
│   │   │   └── hooks/
│   │   │
│   │   ├── surveys/               # Surveys module
│   │   │   ├── pages/
│   │   │   │   ├── SurveysListPage.tsx
│   │   │   │   ├── SurveyBuilderPage.tsx
│   │   │   │   ├── SurveyPreviewPage.tsx
│   │   │   │   └── SurveyResultsPage.tsx
│   │   │   ├── components/
│   │   │   │   ├── builder/
│   │   │   │   │   ├── QuestionTypes.tsx
│   │   │   │   │   ├── QuestionEditor.tsx
│   │   │   │   │   ├── LogicBuilder.tsx
│   │   │   │   │   └── SurveyCanvas.tsx
│   │   │   │   ├── respondent/
│   │   │   │   │   ├── SurveyRenderer.tsx
│   │   │   │   │   └── QuestionRenderer.tsx
│   │   │   │   └── results/
│   │   │   │       ├── ResponsesTable.tsx
│   │   │   │       └── ResultsCharts.tsx
│   │   │   └── hooks/
│   │   │
│   │   ├── analytics/             # Analytics module
│   │   │   ├── pages/
│   │   │   │   ├── AnalyticsDashboard.tsx
│   │   │   │   └── KPIConfigPage.tsx
│   │   │   ├── components/
│   │   │   │   ├── ChartWidget.tsx
│   │   │   │   ├── KPIBuilder.tsx
│   │   │   │   └── TrendAnalysis.tsx
│   │   │   └── hooks/
│   │   │
```

```
│   │   ├── reports/              # Reports module
│   │   │   └── ...
│   │   │
│   │   ├── users/                # Users management
│   │   │   └── ...
│   │   │
│   │   └── settings/             # Settings module
│   │       └── ...
│   │
│   └── styles/                   # Global styles
│       ├── globals.css
│       ├── variables.css
│       └── themes/
│           ├── light.css
│           └── dark.css
│
├── tests/                        # Tests
│   ├── unit/
│   ├── integration/
│   └── e2e/
│
├── .env.example
├── .eslintrc.js
├── .prettierrc
├── tsconfig.json
├── vite.config.ts
└── package.json
```

3.4.2 Gestión de Estado

```
// store/slices/auth.slice.ts
import { create } from 'zustand';
import { persist, createJSONStorage } from 'zustand/middleware';

interface User {
  id: string;
  email: string;
  firstName: string;
  lastName: string;
  roles: string[];
  permissions: string[];
  tenantId: string;
}

interface AuthState {
  user: User | null;
  accessToken: string | null;
  refreshToken: string | null;
  isAuthenticated: boolean;
  isLoading: boolean;

  // Actions
  setUser: (user: User) => void;
  setTokens: (accessToken: string, refreshToken: string) => void;
  logout: () => void;
  setLoading: (isLoading: boolean) => void;
}

export const useAuthStore = create<AuthState>()(
  persist(
    (set) => ({
      user: null,
      accessToken: null,
      refreshToken: null,
      isAuthenticated: false,
      isLoading: true,

      setUser: (user) =>
        set({ user, isAuthenticated: true, isLoading: false }),

      setTokens: (accessToken, refreshToken) =>
        set({ accessToken, refreshToken }),

      logout: () =>
        set({
          user: null,
          accessToken: null,
          refreshToken: null,
          isAuthenticated: false,
          isLoading: false,
        }),

      setLoading: (isLoading) => set({ isLoading }),
    }),
    {
      name: 'auth-storage',
      storage: createJSONStorage(() => sessionStorage),
      partialize: (state) => ({
        accessToken: state.accessToken,
        refreshToken: state.refreshToken,
      }),
    },
```

```
  ),
);

// React Query para server state
// services/complaints.service.ts
import { useQuery, useMutation, useQueryClient } from '@tanstack/react-query';
import { apiClient } from '@/core/api/axios-instance';

export const COMPLAINTS_QUERY_KEY = 'complaints';

export const useComplaints = (filters: ComplaintFilters) => {
  return useQuery({
    queryKey: [COMPLAINTS_QUERY_KEY, filters],
    queryFn: () => apiClient.get('/complaints', { params: filters }),
    staleTime: 5 * 60 * 1000, // 5 minutes
    cacheTime: 10 * 60 * 1000, // 10 minutes
  });
};

export const useCreateComplaint = () => {
  const queryClient = useQueryClient();

  return useMutation({
    mutationFn: (data: CreateComplaintDto) =>
      apiClient.post('/complaints', data),
    onSuccess: () => {
      queryClient.invalidateQueries({ queryKey: [COMPLAINTS_QUERY_KEY] });
    },
  });
};
```

3.4.3 Componente de Permisos

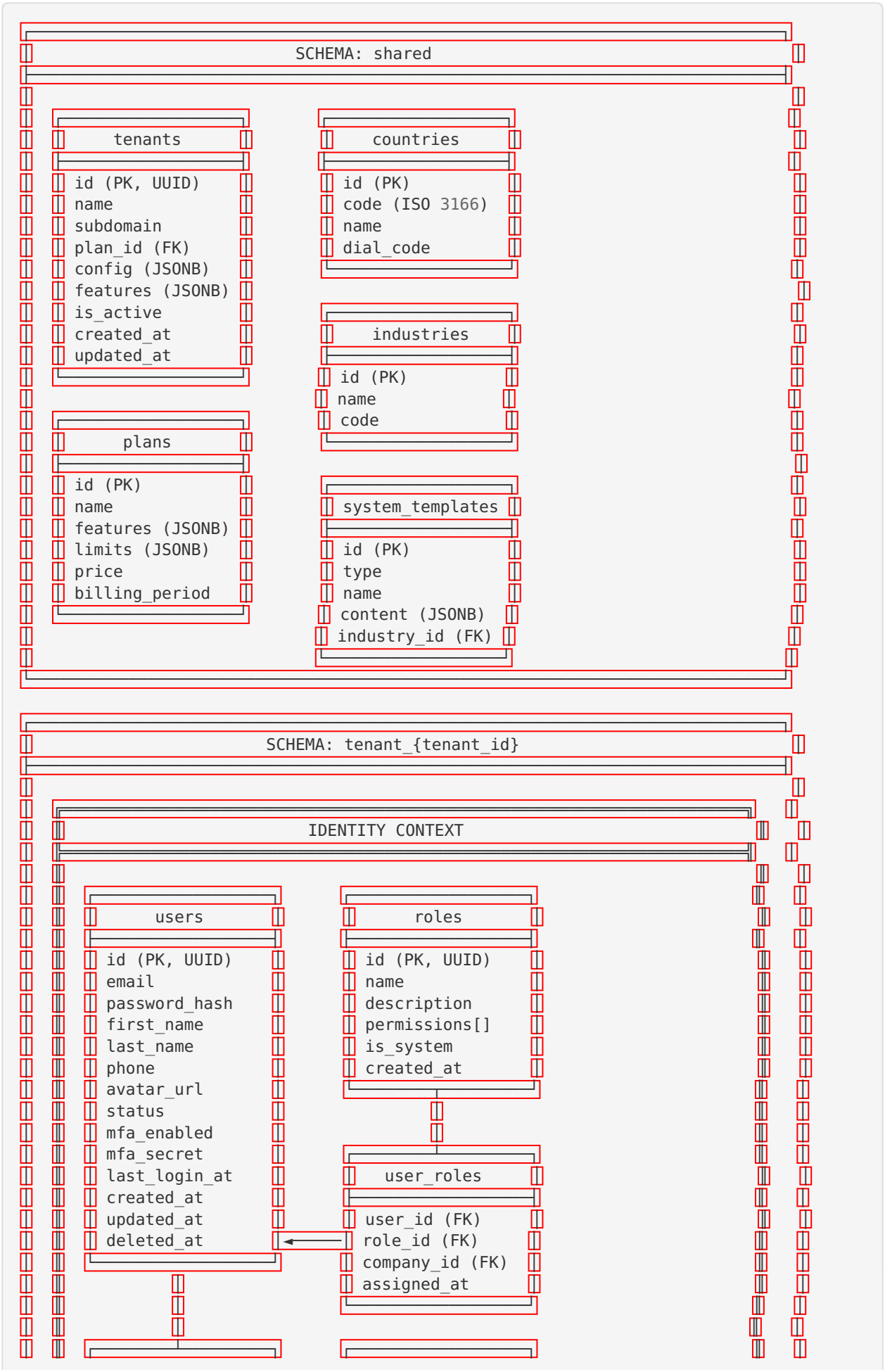
```tsx
// core/permissions/can.component.tsx
import { usePermissions } from './use-permissions.hook';

interface CanProps {
  permission: string | string[];
  children: React.ReactNode;
  fallback?: React.ReactNode;
}

export const Can: React.FC<CanProps> = ({
  permission,
  children,
  fallback = null,
}) => {
  const { hasPermission, hasAnyPermission } = usePermissions();

  const permitted = Array.isArray(permission)
    ? hasAnyPermission(permission)
    : hasPermission(permission);

  return permitted ? <>{children}</> : <>{fallback}</>;
};

// Uso
<Can permission="complaints.create">
  <Button onClick={openCreateModal}>Nueva Denuncia</Button>
</Can>

<Can
  permission={['reports.create', 'reports.regulatory']}
  fallback={<p>Sin acceso a reportes</p>}
>
  <ReportsSection />
</Can>
```

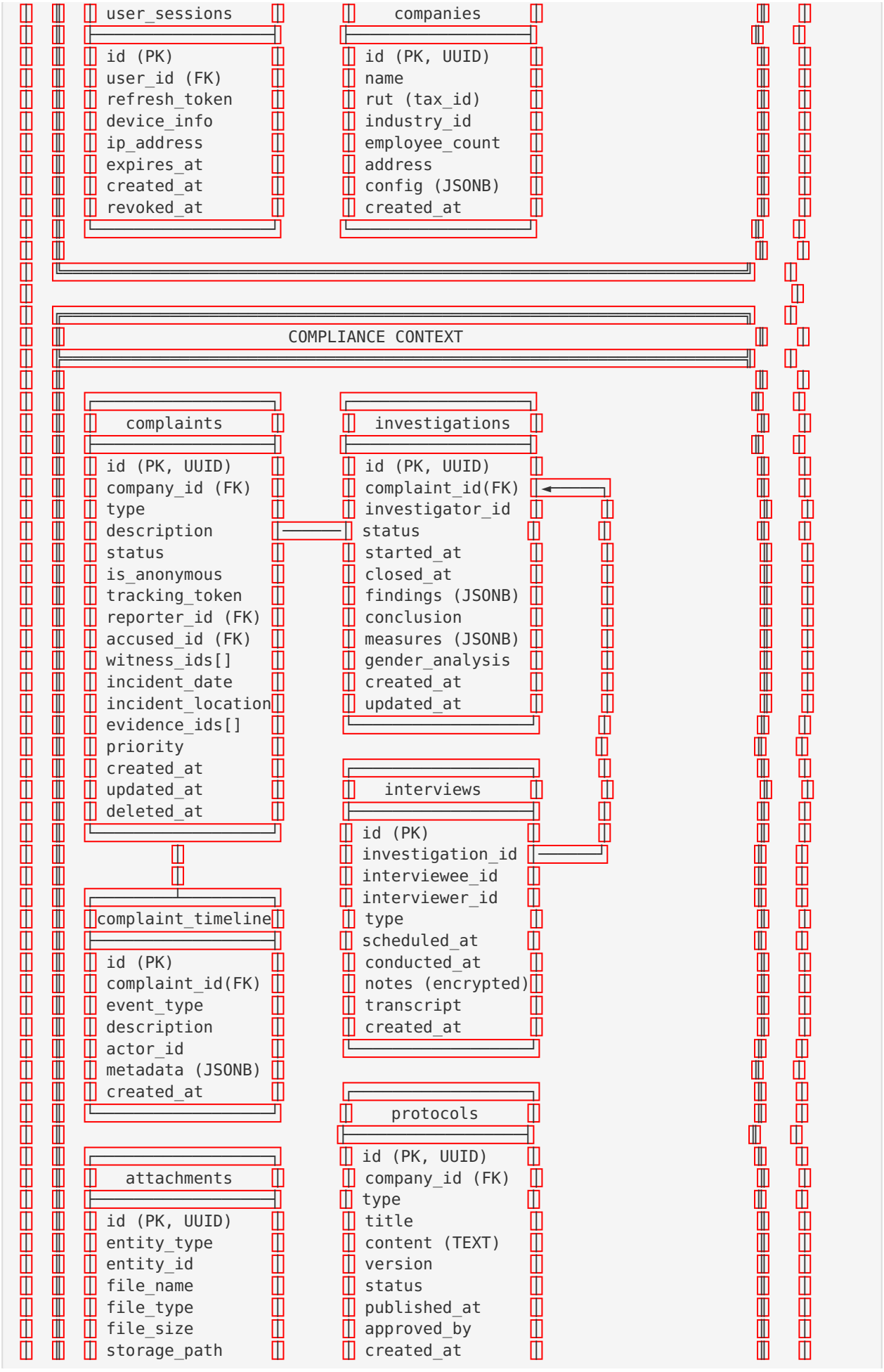
3.4.4 Aplicaciones Frontend

Aplicación	Descripción	Usuarios
Admin Portal	Panel de administración completo	Admins, Compliance Officers, HR
Employee Portal	Portal para colaboradores	Todos los empleados
Public Portal	Denuncias anónimas y encuestas públicas	Público general
Mobile App	App React Native	Empleados en terreno

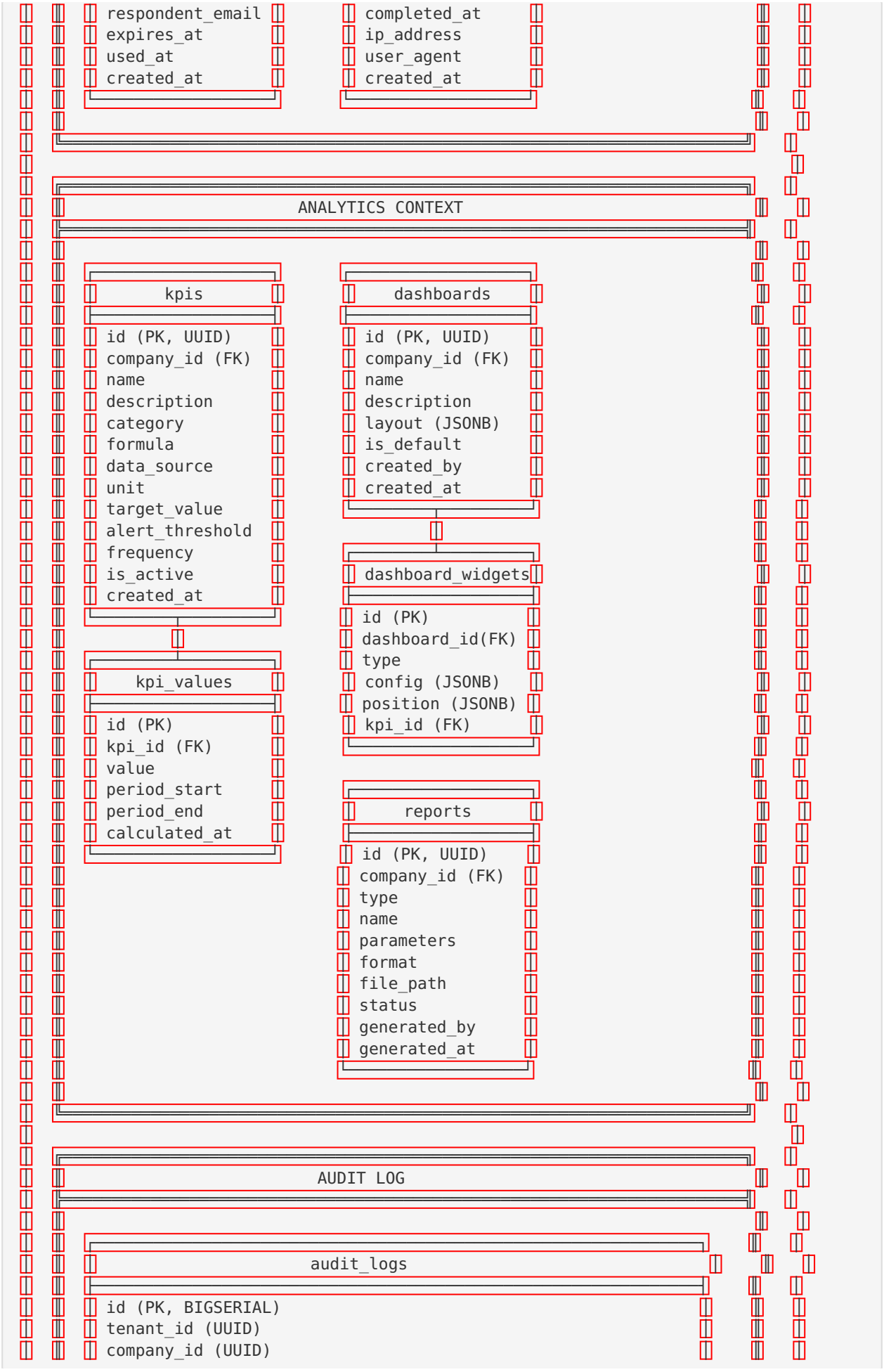
3.5 Base de Datos

3.5.1 Modelo de Datos (ERD Textual)


SCHEMA: shared
tenants
id (PK, UUID)
name
subdomain
plan_id (FK)
config (JSONB)
features (JSONB)
is_active
created_at
updated_at
countries
id (PK)
code (ISO 3166)
name
dial_code
industries
id (PK)
name
code
plans
id (PK)
name
features (JSONB)
limits (JSONB)
price
billing_period
system_templates
id (PK)
type
name
content (JSONB)
industry_id (FK)
SCHEMA: tenant_{tenant_id}
IDENTITY CONTEXT
users
id (PK, UUID)
email
password_hash
first_name
last_name
phone
avatar_url
status
mfa_enabled
mfa_secret
last_login_at
created_at
updated_at
deleted_at
roles
id (PK, UUID)
name
description
permissions[]
is_system
created_at
user_roles
user_id (FK)
role_id (FK)
company_id (FK)
assigned_at


user_sessions
id (PK)
user_id (FK)
refresh_token
device_info
ip_address
expires_at
created_at
revoked_at
companies
id (PK, UUID)
name
rut (tax_id)
industry_id
employee_count
address
config (JSONB)
created_at
COMPLIANCE CONTEXT
complaints
id (PK, UUID)
company_id (FK)
type
description
status
is_anonymous
tracking_token
reporter_id (FK)
accused_id (FK)
witness_ids[]
incident_date
incident_location
evidence_ids[]
priority
created_at
updated_at
deleted_at
investigations
id (PK, UUID)
complaint_id(FK)
investigator_id
status
started_at
closed_at
findings (JSONB)
conclusion
measures (JSONB)
gender_analysis
created_at
updated_at
interviews
id (PK)
investigation_id
interviewee_id
interviewer_id
type
scheduled_at
conducted_at
notes (encrypted)
transcript
created_at
complaint_timeline
id (PK)
complaint_id(FK)
event_type
description
actor_id
metadata (JSONB)
created_at
protocols
id (PK, UUID)
company_id (FK)
type
title
content (TEXT)
version
status
published_at
approved_by
created_at
attachments
id (PK, UUID)
entity_type
entity_id
file_name
file_type
file_size
storage_path


checksum
is_encrypted
uploaded_by
created_at
updated_at
protocol_ack
id (PK)
protocol_id (FK)
user_id (FK)
acknowledged_at
ip_address
psychosocial_eval
id (PK, UUID)
company_id (FK)
survey_id (FK)
status
start_date
end_date
results (JSONB)
risk_level
created_at
SURVEY CONTEXT
surveys
id (PK, UUID)
company_id (FK)
title
description
type
status
settings (JSONB)
starts_at
ends_at
created_by
created_at
updated_at
deleted_at
questions
id (PK, UUID)
survey_id (FK)
type
text
options (JSONB)
validations
logic (JSONB)
order
section_id (FK)
required
created_at
sections
id (PK, UUID)
survey_id (FK)
title
description
order
distributions
id (PK)
survey_id (FK)
channel
recipients[]
scheduled_at
sent_at
status
responses
id (PK, UUID)
survey_id (FK)
respondent_id
token (unique)
status
answers (JSONB)
metadata (JSONB)
started_at
response_tokens
id (PK)
survey_id (FK)
token (unique)


respondent_email
expires_at
used_at
created_at
completed_at
ip_address
user_agent
created_at
ANALYTICS CONTEXT
kpis
id (PK, UUID)
company_id (FK)
name
description
category
formula
data_source
unit
target_value
alert_threshold
frequency
is_active
created_at
kpi_values
id (PK)
kpi_id (FK)
value
period_start
period_end
calculated_at
dashboards
id (PK, UUID)
company_id (FK)
name
description
layout (JSONB)
is_default
created_by
created_at
dashboard_widgets
id (PK)
dashboard_id(FK)
type
config (JSONB)
position (JSONB)
kpi_id (FK)
reports
id (PK, UUID)
company_id (FK)
type
name
parameters
format
file_path
status
generated_by
generated_at
AUDIT LOG
audit_logs
id (PK, BIGSERIAL)
tenant_id (UUID)
company_id (UUID)

```
    user_id (UUID)
    action (VARCHAR) -- CREATE, UPDATE, DELETE, READ, LOGIN, etc
    entity_type (VARCHAR) -- complaint, investigation, user, etc
    entity_id (UUID)
    old_values (JSONB) -- Estado anterior
    new_values (JSONB) -- Estado nuevo
    ip_address (INET)
    user_agent (TEXT)
    request_id (UUID) -- Correlation ID
    additional_data (JSONB)
    created_at (TIMESTAMPTZ)
    -- Particionado por mes: audit_logs_2024_01, audit_logs_2024_02...
    -- Índices: (tenant_id, created_at), (entity_type, entity_id)
    -- Tabla inmutable (append-only)
```

3.5.2 Índices y Optimizaciones

```sql
-- Índices principales por tabla

-- Users
CREATE INDEX idx_users_email ON users(email);
CREATE INDEX idx_users_status ON users(status) WHERE deleted_at IS NULL;

-- Complaints
CREATE INDEX idx_complaints_company ON complaints(company_id);
CREATE INDEX idx_complaints_status ON complaints(status);
CREATE INDEX idx_complaints_type ON complaints(type);
CREATE INDEX idx_complaints_created ON complaints(created_at DESC);
CREATE INDEX idx_complaints_tracking ON complaints(tracking_token) WHERE is_anonymous
= true;

-- Investigations
CREATE INDEX idx_investigations_complaint ON investigations(complaint_id);
CREATE INDEX idx_investigations_investigator ON investigations(investigator_id);
CREATE INDEX idx_investigations_status ON investigations(status);

-- Surveys
CREATE INDEX idx_surveys_company ON surveys(company_id);
CREATE INDEX idx_surveys_status ON surveys(status);
CREATE INDEX idx_surveys_dates ON surveys(starts_at, ends_at);

-- Responses
CREATE INDEX idx_responses_survey ON responses(survey_id);
CREATE INDEX idx_responses_token ON responses(token);
CREATE INDEX idx_responses_status ON responses(status);

-- Audit logs (partitioned table)
CREATE INDEX idx_audit_logs_tenant_date ON audit_logs(tenant_id, created_at DESC);
CREATE INDEX idx_audit_logs_entity ON audit_logs(entity_type, entity_id);
CREATE INDEX idx_audit_logs_user ON audit_logs(user_id, created_at DESC);
```

3.5.3 Estrategia de Migraciones

```typescript
// Usando TypeORM Migrations

// migrations/1708000000000-CreateComplaintsTable.ts
import { MigrationInterface, QueryRunner, Table, Index } from 'typeorm';

export class CreateComplaintsTable1708000000000 implements MigrationInterface {
  name = 'CreateComplaintsTable1708000000000';

  public async up(queryRunner: QueryRunner): Promise<void> {
    // Crear tabla
    await queryRunner.createTable(
      new Table({
        name: 'complaints',
        columns: [
          {
            name: 'id',
            type: 'uuid',
            isPrimary: true,
            default: 'uuid_generate_v4()',
          },
          {
            name: 'company_id',
            type: 'uuid',
            isNullable: false,
          },
          {
            name: 'type',
            type: 'varchar',
            length: '50',
            isNullable: false,
          },
          {
            name: 'description',
            type: 'text',
            isNullable: false,
          },
          {
            name: 'status',
            type: 'varchar',
            length: '30',
            default: "'RECEIVED'",
          },
          {
            name: 'is_anonymous',
            type: 'boolean',
            default: false,
          },
          {
            name: 'tracking_token',
            type: 'varchar',
            length: '64',
            isNullable: true,
            isUnique: true,
          },
          {
            name: 'reporter_id',
            type: 'uuid',
            isNullable: true,
          },
          {
            name: 'created_at',
            type: 'timestamptz',
```

```
            default: 'now()',
          },
          {
            name: 'updated_at',
            type: 'timestamptz',
            default: 'now()',
          },
          {
            name: 'deleted_at',
            type: 'timestamptz',
            isNullable: true,
          },
        ],
      }),
      true,
    );

    // Crear índices
    await queryRunner.createIndex(
      'complaints',
      new Index({
        name: 'idx_complaints_company',
        columnNames: ['company_id'],
      }),
    );

    // Habilitar RLS
    await queryRunner.query(`
      ALTER TABLE complaints ENABLE ROW LEVEL SECURITY;

      CREATE POLICY complaints_tenant_isolation ON complaints
        USING (company_id IN (
          SELECT company_id FROM user_company_access
          WHERE user_id = current_setting('app.current_user_id')::uuid
        ));
    `);
  }

  public async down(queryRunner: QueryRunner): Promise<void> {
    await queryRunner.dropTable('complaints');
  }
}
```

3.6 Seguridad

3.6.1 Matriz de Seguridad

Capa	Medida	Implementación
Transporte	TLS 1.3	Certificados SSL/TLS en load balancer
Autenticación	JWT + 2FA	Access tokens cortos (15 min), refresh tokens rotatorios
Autorización	RBAC + ABAC	Permisos granulares + políticas basadas en atributos
Datos en reposo	AES-256	Campos sensibles encriptados (columnas, S3)
API	Rate limiting	Límites por tenant, IP y usuario
Input	Validación estricta	DTOs con class-validator, sanitización
SQL	Prepared statements	ORM con parameterización automática
XSS	CSP headers	Content-Security-Policy restrictivo
CSRF	Tokens	SameSite cookies + CSRF tokens

3.6.2 Encriptación de Datos Sensibles

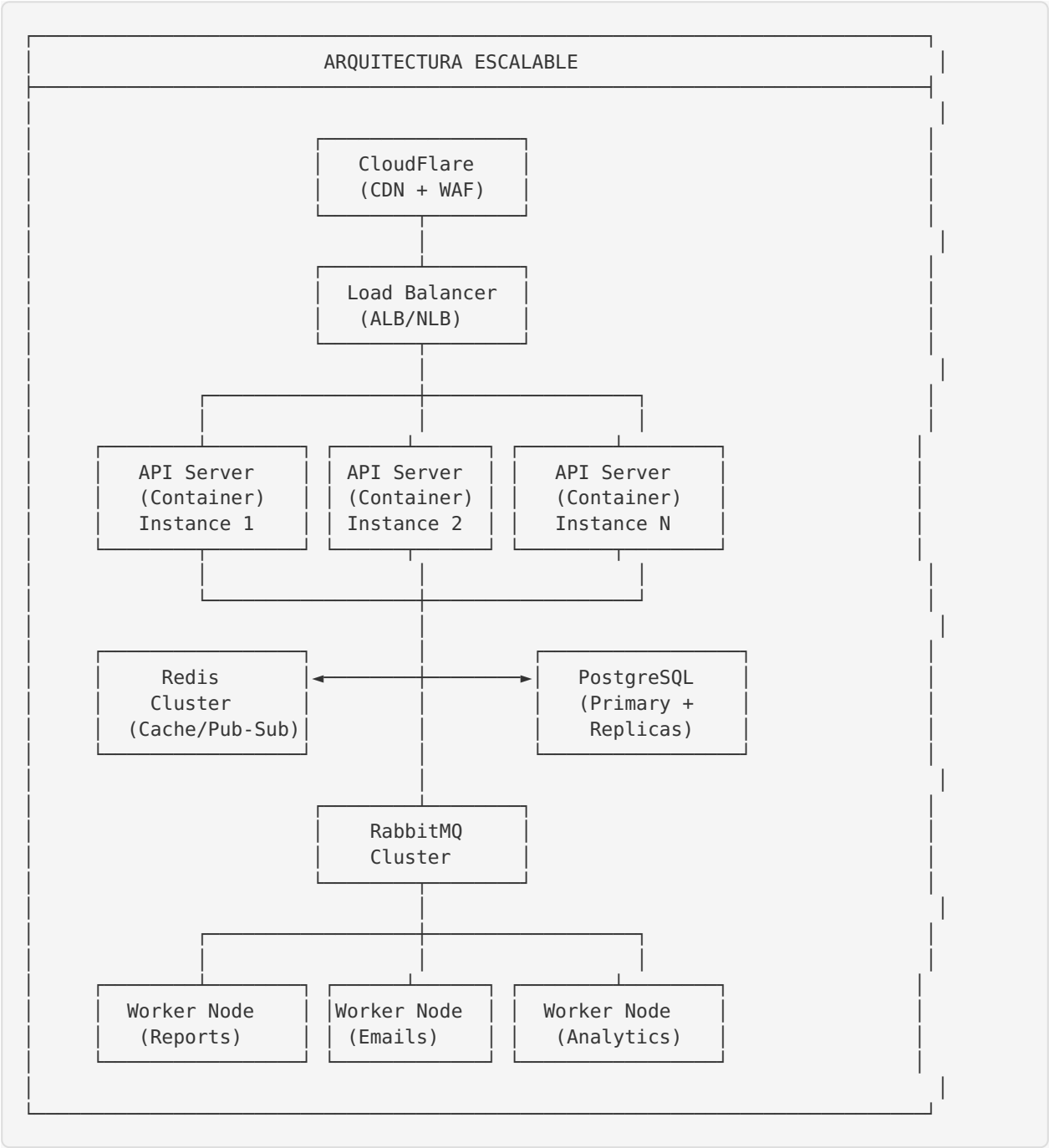
```typescript
// Campos que requieren encriptación
const ENCRYPTED_FIELDS = {
  complaints: ['description', 'witness_names'],
  interviews: ['notes', 'transcript'],
  users: ['phone', 'address'],
  attachments: ['file_content'], // Solo si se almacena en DB
};

// Servicio de encriptación
@Injectable()
export class EncryptionService {
  private readonly algorithm = 'aes-256-gcm';
  private readonly keyLength = 32;
  private readonly ivLength = 16;
  private readonly authTagLength = 16;

  constructor(
    @Inject('TENANT_ENCRYPTION_KEY')
    private readonly tenantKeyService: TenantKeyService,
  ) {}

  async encrypt(plaintext: string, tenantId: string): Promise<EncryptedData> {
    const key = await this.tenantKeyService.getKey(tenantId);
    const iv = crypto.randomBytes(this.ivLength);

    const cipher = crypto.createCipheriv(this.algorithm, key, iv);

    let encrypted = cipher.update(plaintext, 'utf8', 'hex');
    encrypted += cipher.final('hex');

    const authTag = cipher.getAuthTag();

    return {
      ciphertext: encrypted,
      iv: iv.toString('hex'),
      authTag: authTag.toString('hex'),
      version: 1, // Para rotación de claves
    };
  }

  async decrypt(data: EncryptedData, tenantId: string): Promise<string> {
    const key = await this.tenantKeyService.getKey(tenantId, data.version);
    const iv = Buffer.from(data.iv, 'hex');
    const authTag = Buffer.from(data.authTag, 'hex');

    const decipher = crypto.createDecipheriv(this.algorithm, key, iv);
    decipher.setAuthTag(authTag);

    let decrypted = decipher.update(data.ciphertext, 'hex', 'utf8');
    decrypted += decipher.final('utf8');

    return decrypted;
  }
}
```

3.6.3 Headers de Seguridad

```
// Configuración de Helmet
app.use(
  helmet({
    contentSecurityPolicy: {
      directives: {
        defaultSrc: ["'self'"],
        scriptSrc: ["'self'", "'unsafe-inline'", 'cdn.jsdelivr.net'],
        styleSrc: ["'self'", "'unsafe-inline'", 'fonts.googleapis.com'],
        imgSrc: ["'self'", 'data:', 'https:'],
        connectSrc: ["'self'", 'wss:', process.env.API_URL],
        fontSrc: ["'self'", 'fonts.gstatic.com'],
        objectSrc: ["'none'"],
        mediaSrc: ["'self'"],
        frameSrc: ["'none'"],
      },
    },
    crossOriginEmbedderPolicy: true,
    crossOriginOpenerPolicy: true,
    crossOriginResourcePolicy: { policy: 'same-site' },
    dnsPrefetchControl: true,
    frameguard: { action: 'deny' },
    hidePoweredBy: true,
    hsts: {
      maxAge: 31536000,
      includeSubDomains: true,
      preload: true,
    },
    ieNoOpen: true,
    noSniff: true,
    originAgentCluster: true,
    permittedCrossDomainPolicies: { permittedPolicies: 'none' },
    referrerPolicy: { policy: 'strict-origin-when-cross-origin' },
    xssFilter: true,
  }),
);
```

3.7 Escalabilidad y Performance

3.7.1 Estrategia de Escalamiento



3.7.2 Estrategia de Caching

```
// Configuración de cache multinivel
const CACHE_CONFIG = {
  // L1: In-memory cache (por instancia)
  local: {
    ttl: 60, // 1 minuto
    max: 1000, // items
    items: ['user_permissions', 'tenant_config', 'feature_flags'],
  },

  // L2: Redis cache (compartido)
  redis: {
    // Datos de sesión
    sessions: { ttl: 3600 * 24 * 7 }, // 7 días

    // Datos de referencia
    catalogs: { ttl: 3600 * 24 }, // 24 horas

    // Resultados de queries
    queries: {
      complaints_list: { ttl: 60 },
      dashboard_summary: { ttl: 300 },
      kpi_values: { ttl: 900 },
    },

    // Rate limiting
    rate_limits: { ttl: 60 },
  },

  // L3: CDN cache (estáticos)
  cdn: {
    assets: { ttl: 31536000 }, // 1 año
    api_public: { ttl: 300 }, // 5 min para datos públicos
  },
};

// Implementación del cache decorator
function Cacheable(options: CacheOptions) {
  return function (
    target: any,
    propertyKey: string,
    descriptor: PropertyDescriptor,
  ) {
    const originalMethod = descriptor.value;

    descriptor.value = async function (...args: any[]) {
      const cacheKey = `${options.prefix}:${JSON.stringify(args)}`;
      const cached = await this.cacheService.get(cacheKey);

      if (cached) {
        return cached;
      }

      const result = await originalMethod.apply(this, args);
      await this.cacheService.set(cacheKey, result, options.ttl);

      return result;
    };

    return descriptor;
  };
}
```

```
// Uso
@Cacheable({ prefix: 'complaints_list', ttl: 60 })
async getComplaints(filters: ComplaintFilters): Promise<ComplaintList> {
  // Query a la base de datos
}
```

3.7.3 Procesamiento Asíncrono

```
// Definición de queues y jobs
const QUEUES = {
  // Queue de alta prioridad
  HIGH_PRIORITY: {
    name: 'high-priority',
    jobs: ['send_urgent_notification', 'process_complaint_alert'],
    concurrency: 10,
  },

  // Queue de procesamiento normal
  DEFAULT: {
    name: 'default',
    jobs: ['send_email', 'generate_report', 'sync_external'],
    concurrency: 5,
  },

  // Queue de procesamiento pesado
  HEAVY: {
    name: 'heavy-processing',
    jobs: ['generate_analytics', 'bulk_import', 'export_data'],
    concurrency: 2,
  },

  // Queue de tareas programadas
  SCHEDULED: {
    name: 'scheduled',
    jobs: ['daily_report', 'weekly_digest', 'data_cleanup'],
    concurrency: 1,
  },
};

// Implementación de job processor
@Processor('default')
export class EmailProcessor {
  constructor(
    private readonly emailService: EmailService,
    private readonly logger: LoggerService,
  ) {}

  @Process('send_email')
  async handleSendEmail(job: Job<SendEmailJobData>): Promise<void> {
    const { to, template, data, tenantId } = job.data;

    this.logger.log(`Processing email job ${job.id} for tenant ${tenantId}`);

    try {
      await this.emailService.send({
        to,
        template,
        data,
        tenantId,
      });

      this.logger.log(`Email sent successfully: ${job.id}`);
    } catch (error) {
      this.logger.error(`Failed to send email: ${job.id}`, error);
      throw error; // Re-throw para retry automático
    }
  }

  @OnQueueFailed()
  async handleFailed(job: Job, error: Error): Promise<void> {
```

```
    this.logger.error(
      `Job ${job.id} failed after ${job.attemptsMade} attempts`,
      error,
    );

    if (job.attemptsMade >= job.opts.attempts!) {
      // Notificar fallo permanente
      await this.notificationService.notifyJobFailed(job);
    }
  }
}
```

3.8 Integraciones

3.8.1 API REST Pública

```yaml
openapi: 3.0.3
info:
  title: ComplianceHub Pro API
  version: 1.0.0
  description: API pública para integración con sistemas externos

servers:
  - url: https://api.compliancehub.com/v1
    description: Production
  - url: https://api-sandbox.compliancehub.com/v1
    description: Sandbox

security:
  - BearerAuth: []
  - ApiKeyAuth: []

paths:
  /complaints:
    get:
      summary: List complaints
      tags: [Complaints]
      parameters:
        - $ref: '#/components/parameters/PageParam'
        - $ref: '#/components/parameters/PageSizeParam'
        - name: status
          in: query
          schema:
            type: string
            enum: [RECEIVED, UNDER_INVESTIGATION, CLOSED]
      responses:
        '200':
          description: Successful response
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ComplaintListResponse'
        '401':
          $ref: '#/components/responses/Unauthorized'
        '403':
          $ref: '#/components/responses/Forbidden'

components:
  securitySchemes:
    BearerAuth:
      type: http
      scheme: bearer
      bearerFormat: JWT
    ApiKeyAuth:
      type: apiKey
      in: header
      name: X-API-Key

  schemas:
    Complaint:
      type: object
      properties:
        id:
          type: string
          format: uuid
        type:
          type: string
          enum: [WORKPLACE_HARASSMENT, SEXUAL_HARASSMENT, WORKPLACE_VIOLENCE]
```

```
        status:
          type: string
          enum: [RECEIVED, UNDER_INVESTIGATION, CLOSED]
        createdAt:
          type: string
          format: date-time
      required: [id, type, status, createdAt]
```

3.8.2 Webhooks

```
// Eventos disponibles para webhooks
const WEBHOOK_EVENTS = {
  // Complaints
  'complaint.created': 'When a new complaint is submitted',
  'complaint.updated': 'When a complaint is updated',
  'complaint.status_changed': 'When complaint status changes',

  // Investigations
  'investigation.started': 'When an investigation begins',
  'investigation.closed': 'When an investigation is closed',

  // Surveys
  'survey.published': 'When a survey is published',
  'survey.completed': 'When a survey receives all responses',
  'response.submitted': 'When a survey response is submitted',

  // Reports
  'report.generated': 'When a report is generated',

  // Users
  'user.created': 'When a new user is created',
  'user.deactivated': 'When a user is deactivated',
};

// Payload de webhook
interface WebhookPayload {
  id: string;                    // Unique event ID
  event: string;                 // Event type
  timestamp: string;             // ISO 8601
  tenant_id: string;             // Tenant identifier
  data: Record<string, any>;     // Event-specific data
  signature: string;             // HMAC-SHA256 signature
}

// Servicio de webhooks
@Injectable()
export class WebhookService {
  async dispatch(event: string, data: any, tenantId: string): Promise<void> {
    // 1. Obtener webhooks suscritos a este evento
    const webhooks = await this.webhookRepository.findByEventAndTenant(
      event,
      tenantId,
    );

    // 2. Preparar payload
    const payload: WebhookPayload = {
      id: uuid(),
      event,
      timestamp: new Date().toISOString(),
      tenant_id: tenantId,
      data,
      signature: '', // Se calcula después
    };

    // 3. Encolar entregas
    for (const webhook of webhooks) {
      // Calcular firma
      payload.signature = this.sign(payload, webhook.secret);

      await this.queue.add('webhook_delivery', {
        webhookId: webhook.id,
        url: webhook.url,
```

```
        payload,
        attempt: 1,
      });
    }
  }

  private sign(payload: WebhookPayload, secret: string): string {
    const body = JSON.stringify(payload);
    return crypto
      .createHmac('sha256', secret)
      .update(body)
      .digest('hex');
  }
}
```

3.8.3 Conectores HRIS

```typescript
// Interface para conectores HRIS
interface HRISConnector {
  name: string;
  authenticate(credentials: HRISCredentials): Promise<HRISConnection>;
  syncEmployees(connection: HRISConnection): Promise<SyncResult>;
  syncOrganization(connection: HRISConnection): Promise<SyncResult>;
  handleWebhook(payload: any): Promise<void>;
}

// Implementación para SAP SuccessFactors
@Injectable()
export class SAPSuccessFactorsConnector implements HRISConnector {
  name = 'SAP SuccessFactors';

  async authenticate(credentials: SAPCredentials): Promise<HRISConnection> {
    const tokenResponse = await axios.post(
      `${credentials.apiUrl}/oauth/token`,
      {
        grant_type: 'client_credentials',
        client_id: credentials.clientId,
        client_secret: credentials.clientSecret,
        company_id: credentials.companyId,
      },
    );

    return {
      accessToken: tokenResponse.data.access_token,
      expiresAt: new Date(Date.now() + tokenResponse.data.expires_in * 1000),
      refreshToken: tokenResponse.data.refresh_token,
      metadata: { companyId: credentials.companyId },
    };
  }

  async syncEmployees(connection: HRISConnection): Promise<SyncResult> {
    const employees = await this.fetchAllEmployees(connection);

    const result: SyncResult = {
      total: employees.length,
      created: 0,
      updated: 0,
      errors: [],
    };

    for (const employee of employees) {
      try {
        const mapped = this.mapToInternalFormat(employee);
        const existing = await this.userRepository.findByExternalId(
          'sap_sf',
          employee.personIdExternal,
        );

        if (existing) {
          await this.userRepository.update(existing.id, mapped);
          result.updated++;
        } else {
          await this.userRepository.create(mapped);
          result.created++;
        }
      } catch (error) {
        result.errors.push({
          externalId: employee.personIdExternal,
          error: error.message,
```

```
        });
      }
    }

    return result;
  }

  private mapToInternalFormat(sapEmployee: SAPEmployee): CreateUserDto {
    return {
      externalId: sapEmployee.personIdExternal,
      externalSource: 'sap_sf',
      email: sapEmployee.email,
      firstName: sapEmployee.firstName,
      lastName: sapEmployee.lastName,
      department: sapEmployee.department,
      position: sapEmployee.jobTitle,
      managerId: sapEmployee.managerId,
      hireDate: sapEmployee.startDate,
      status: this.mapStatus(sapEmployee.status),
    };
  }
}
```

4. STACK TECNOLÓGICO RECOMENDADO

4.1 Backend

Componente	Tecnología	Justificación
Runtime	Node.js 20 LTS	Performance, ecosistema, async nativo
Framework	NestJS 10	Arquitectura enterprise, TypeScript nativo, modular
ORM	TypeORM / Prisma	Type-safe, migrations, multi-database
Validación	class-validator	Decoradores, integración NestJS
API Docs	Swagger/OpenAPI	Estándar industria, generación automática
Auth	Passport.js + JWT	Flexibilidad, estrategias múltiples
Queue	BullMQ	Redis-backed, confiable, UI dashboard
Testing	Jest	Coverage, mocking, snapshot testing

4.2 Frontend

Componente	Tecnología	Justificación
Framework	React 18	Ecosistema maduro, comunidad, hooks
Language	TypeScript 5	Type safety, mejor DX
Build	Vite	Velocidad, HMR, ESBuild
State	Zustand + React Query	Simplicidad + cache de servidor
Forms	React Hook Form	Performance, validación
UI	Shadcn/ui + Tailwind	Componentes accesibles, customizable
Charts	Recharts / Tremor	React nativo, responsive
Testing	Vitest + Testing Library	Velocidad, integración Vite

4.3 Base de Datos

Componente	Tecnología	Uso
Primary DB	PostgreSQL 16	Datos transaccionales, JSONB, RLS
Cache	Redis 7	Sessions, cache, pub/sub, rate limiting
Search	Elasticsearch 8	Full-text search, analytics
Time Series	TimescaleDB	Métricas, KPIs históricos

4.4 Infraestructura

Componente	Tecnología	Uso
Cloud	AWS / GCP	Infraestructura principal
Containers	Docker	Empaquetado consistente
Orchestration	Kubernetes (EKS/GKE)	Orquestación, auto-scaling
CDN	CloudFlare	Cache, WAF, DDoS protection
Storage	S3 / GCS	Archivos, documentos, backups
Secrets	AWS Secrets Manager / Vault	Gestión de credenciales

4.5 CI/CD y DevOps

Componente	Tecnología	Uso
VCS	GitHub	Código, PRs, Actions
CI/CD	GitHub Actions	Pipelines automatizados
IaC	Terraform	Infraestructura como código
Registry	ECR / GCR	Imágenes Docker
Monitoring	Datadog / Grafana	Métricas, alertas
Logging	ELK Stack	Logs centralizados
APM	New Relic / Datadog	Performance monitoring

4.6 Testing

Tipo	Herramienta	Cobertura Mínima
Unit	Jest / Vitest	80%
Integration	Supertest	Endpoints críticos
E2E	Playwright	Flujos principales
Load	k6	Escenarios de carga
Security	OWASP ZAP	Vulnerabilidades

5. MODELO DE DATOS DETALLADO

5.1 Entidades Principales

5.1.1 Tenant (Shared Schema)

```
interface Tenant {
  id: UUID;                      // Identificador único
  name: string;                  // Nombre del tenant
  subdomain: string;             // Subdominio único
  planId: UUID;                  // Plan contratado
  config: {
    timezone: string;            // Zona horaria default
    locale: string;              // Idioma default
    dateFormat: string;          // Formato de fechas
    branding: {
      logo: string;
      primaryColor: string;
      secondaryColor: string;
    };
    notifications: {
      emailEnabled: boolean;
      smsEnabled: boolean;
      whatsappEnabled: boolean;
    };
    security: {
      mfaRequired: boolean;
      passwordPolicy: PasswordPolicy;
      sessionTimeout: number;    // minutos
    };
  };
  features: string[];            // Features habilitados
  limits: {
    maxUsers: number;
    maxCompanies: number;
    maxStorageGB: number;
    maxApiCallsPerMonth: number;
  };
  isActive: boolean;
  trialEndsAt: Date | null;
  createdAt: Date;
  updatedAt: Date;
}
```

5.1.2 User (Tenant Schema)

```typescript
interface User {
  id: UUID;
  email: string;                // Único por tenant
  passwordHash: string;         // Bcrypt hash
  firstName: string;
  lastName: string;
  phone?: string;               // Encriptado
  avatarUrl?: string;
  status: UserStatus;           // ACTIVE, INACTIVE, PENDING, SUSPENDED
  mfaEnabled: boolean;
  mfaSecret?: string;           // Encriptado
  lastLoginAt?: Date;
  lastLoginIp?: string;
  failedLoginAttempts: number;
  lockedUntil?: Date;
  passwordChangedAt: Date;
  mustChangePassword: boolean;
  preferences: {
    language: string;
    timezone: string;
    notifications: NotificationPreferences;
  };
  metadata: Record<string, any>; // Datos adicionales flexibles
  createdAt: Date;
  updatedAt: Date;
  deletedAt?: Date;             // Soft delete
}

enum UserStatus {
  ACTIVE = 'ACTIVE',
  INACTIVE = 'INACTIVE',
  PENDING = 'PENDING',          // Invitación pendiente
  SUSPENDED = 'SUSPENDED',      // Suspendido por admin
}
```

5.1.3 Complaint (Tenant Schema)

```
interface Complaint {
  id: UUID;
  companyId: UUID;               // Empresa dentro del tenant
  type: ComplaintType;
  description: string;           // Encriptado
  status: ComplaintStatus;
  priority: ComplaintPriority;
  isAnonymous: boolean;
  trackingToken?: string;        // Solo para anónimas

  // Personas involucradas
  reporterId?: UUID;             // Null si anónima
  accusedInfo: {                 // Puede ser null si no se identifica
    userId?: UUID;
    name?: string;
    department?: string;
    position?: string;
  };
  witnessIds: UUID[];

  // Detalles del incidente
  incidentDate?: Date;
  incidentLocation?: string;
  incidentDescription: string;   // Encriptado

  // Categorización
  categories: string[];          // e.g., ['verbal', 'psychological']
  severity: number;              // 1-5

  // Datos de canal
  channel: ComplaintChannel;     // WEB, PHONE, WHATSAPP, EMAIL
  channelMetadata: Record<string, any>;

  // Atención psicológica
  psychologicalSupport: {
    required: boolean;
    status?: 'PENDING' | 'IN_PROGRESS' | 'COMPLETED';
    providerId?: UUID;
    sessions: PsychologicalSession[];
  };

  // Relaciones
  investigationId?: UUID;
  attachmentIds: UUID[];

  // Auditoría
  createdAt: Date;
  updatedAt: Date;
  deletedAt?: Date;
  createdBy?: UUID;              // Sistema si es anónima
}

enum ComplaintType {
  WORKPLACE_HARASSMENT = 'WORKPLACE_HARASSMENT',
  SEXUAL_HARASSMENT = 'SEXUAL_HARASSMENT',
  WORKPLACE_VIOLENCE = 'WORKPLACE_VIOLENCE',
  THIRD_PARTY_VIOLENCE = 'THIRD_PARTY_VIOLENCE',
}

enum ComplaintStatus {
  RECEIVED = 'RECEIVED',
  UNDER_REVIEW = 'UNDER_REVIEW',
```

```
  UNDER_INVESTIGATION = 'UNDER_INVESTIGATION',
  PENDING_RESOLUTION = 'PENDING_RESOLUTION',
  RESOLVED = 'RESOLVED',
  CLOSED = 'CLOSED',
  DISMISSED = 'DISMISSED',
}
```

5.1.4 Investigation (Tenant Schema)

```
interface Investigation {
  id: UUID;
  complaintId: UUID;
  companyId: UUID;

  // Investigador
  investigatorId: UUID;
  investigatorAcceptedAt?: Date;
  conflictOfInterestDeclaration: {
    declared: boolean;
    declaredAt: Date;
    hasConflict: boolean;
    details?: string;
  };

  // Estado
  status: InvestigationStatus;
  startedAt: Date;
  dueDate: Date;                  // Plazo legal
  closedAt?: Date;

  // Proceso
  methodology: string;            // Descripción de metodología
  interviewIds: UUID[];
  evidenceIds: UUID[];

  // Hallazgos
  findings: {
    facts: Finding[];             // Hechos establecidos
    analysis: string;             // Análisis con perspectiva de género
    credibilityAssessment: CredibilityAssessment;
  };

  // Conclusión
  conclusion: {
    determination: 'SUBSTANTIATED' | 'UNSUBSTANTIATED' | 'INCONCLUSIVE';
    summary: string;
    genderPerspectiveAnalysis: string;
    aggravatingFactors: string[];
    mitigatingFactors: string[];
  };

  // Medidas propuestas
  proposedMeasures: ProposedMeasure[];

  // Reporte final
  finalReport: {
    documentId: UUID;
    generatedAt: Date;
    approvedBy?: UUID;
    approvedAt?: Date;
  };

  createdAt: Date;
  updatedAt: Date;
}

interface ProposedMeasure {
  type: MeasureType;              // DISCIPLINARY, ORGANIZATIONAL, SUPPORT
  description: string;
  targetUserId?: UUID;
  deadline?: Date;
```

```
  status: 'PROPOSED' | 'APPROVED' | 'IMPLEMENTED' | 'REJECTED';
  implementedAt?: Date;
  implementedBy?: UUID;
  notes?: string;
}
```

5.1.5 Survey (Tenant Schema)

```typescript
interface Survey {
  id: UUID;
  companyId: UUID;

  // Información básica
  title: string;
  description?: string;
  type: SurveyType;              // CLIMATE, PSYCHOSOCIAL, PULSE, CUSTOM

  // Estado y ciclo de vida
  status: SurveyStatus;
  version: number;

  // Configuración de tiempo
  startsAt?: Date;
  endsAt?: Date;
  timezone: string;

  // Configuración
  settings: {
    isAnonymous: boolean;
    allowMultipleResponses: boolean;
    showProgressBar: boolean;
    randomizeQuestions: boolean;
    randomizeOptions: boolean;
    requireAllQuestions: boolean;
    showQuestionNumbers: boolean;
    allowBack: boolean;
    showSummaryAtEnd: boolean;
    redirectUrl?: string;
    thankYouMessage: string;
  };

  // Notificaciones
  notifications: {
    sendReminders: boolean;
    reminderDays: number[];      // Días antes de cierre
    notifyOnCompletion: boolean;
    completionEmails: string[];
  };

  // Segmentación
  targetAudience: {
    type: 'ALL' | 'SPECIFIC_USERS' | 'DEPARTMENTS' | 'CRITERIA';
    userIds?: UUID[];
    departmentIds?: UUID[];
    criteria?: AudienceCriteria;
  };

  // Secciones y preguntas (estructura separada)
  sectionIds: UUID[];

  // Distribución
  distributions: Distribution[];

  // Estadísticas (calculadas)
  stats: {
    totalInvited: number;
    totalResponses: number;
    completionRate: number;
    averageCompletionTime: number;
    lastResponseAt?: Date;
```

```
  };

  createdBy: UUID;
  createdAt: Date;
  updatedAt: Date;
  deletedAt?: Date;
}

interface Question {
  id: UUID;
  surveyId: UUID;
  sectionId?: UUID;

  // Contenido
  type: QuestionType;
  text: string;
  description?: string;
  placeholder?: string;

  // Opciones (para tipos con opciones)
  options: QuestionOption[];

  // Configuración
  required: boolean;
  order: number;

  // Validaciones
  validations: {
    minLength?: number;
    maxLength?: number;
    minValue?: number;
    maxValue?: number;
    pattern?: string;
    customValidation?: string;
  };

  // Lógica condicional
  logic: {
    showIf?: LogicCondition;
    skipTo?: UUID;              // Question ID
    piping?: {
      sourceQuestionId: UUID;
      transform?: string;
    };
  };

  // Scoring (para encuestas con puntuación)
  scoring?: {
    enabled: boolean;
    weights: Record<string, number>;
    dimension?: string;         // Para agrupar por dimensión
  };

  createdAt: Date;
  updatedAt: Date;
}

enum QuestionType {
  SINGLE_CHOICE = 'SINGLE_CHOICE',
  MULTIPLE_CHOICE = 'MULTIPLE_CHOICE',
  TEXT_SHORT = 'TEXT_SHORT',
  TEXT_LONG = 'TEXT_LONG',
  RATING = 'RATING',
```

```
  SCALE = 'SCALE',
  LIKERT = 'LIKERT',
  NPS = 'NPS',
  MATRIX_SINGLE = 'MATRIX_SINGLE',
  MATRIX_MULTIPLE = 'MATRIX_MULTIPLE',
  RANKING = 'RANKING',
  DATE = 'DATE',
  TIME = 'TIME',
  FILE_UPLOAD = 'FILE_UPLOAD',
  SIGNATURE = 'SIGNATURE',
}
```

5.2 Estrategia de Soft Deletes

```
// Mixin para soft deletes
export abstract class SoftDeletableEntity {
  @Column({ type: 'timestamptz', nullable: true })
  deletedAt?: Date;

  @Column({ type: 'uuid', nullable: true })
  deletedBy?: string;

  // Método para soft delete
  softDelete(userId: string): void {
    this.deletedAt = new Date();
    this.deletedBy = userId;
  }

  // Método para restore
  restore(): void {
    this.deletedAt = undefined;
    this.deletedBy = undefined;
  }

  get isDeleted(): boolean {
    return this.deletedAt !== undefined;
  }
}

// Subscriber para filtrar automáticamente
@EventSubscriber()
export class SoftDeleteSubscriber implements EntitySubscriberInterface {
  beforeFind(event: FindManyOptions<any>): void {
    if (!event.where) {
      event.where = {};
    }
    event.where.deletedAt = IsNull();
  }
}
```

6. ESPECIFICACIONES DE MÓDULOS

6.1 Módulo: Cumplimiento Ley Karin

6.1.1 Descripción Funcional

El módulo de Cumplimiento Ley Karin es el núcleo del sistema para empresas chilenas. Implementa todos los requerimientos de la Ley 21.643 incluyendo:
- Gestión de protocolos de prevención con perspectiva de género
- Canal de denuncias multicanal con opción de anonimato
- Workflow de investigaciones con perspectiva de género
- Atención psicológica temprana
- Gestión de riesgos psicosociales (SUSESO/ISTAS21)
- Reportería regulatoria a organismos fiscalizadores

6.1.2 Casos de Uso Detallados

```
CU-LK-001: Gestionar Protocolos de Prevención
Actor Principal: Encargado de Compliance
Actores Secundarios: Gerente RRHH, Trabajadores

Flujo Principal:
1. Usuario accede al módulo de protocolos
2. Sistema muestra protocolos vigentes
3. Usuario selecciona "Crear nuevo protocolo" o "Editar existente"
4. Sistema presenta editor con secciones obligatorias:
   - Definiciones (acoso laboral, sexual, violencia)
   - Medidas de prevención
   - Procedimiento de denuncia
   - Proceso de investigación
   - Medidas de protección
   - Sanciones aplicables
5. Usuario completa/edita contenido
6. Sistema valida completitud según Ley Karin
7. Usuario envía a revisión
8. Revisor aprueba/rechaza con observaciones
9. Sistema publica versión y archiva anterior
10. Sistema inicia proceso de difusión

Postcondiciones:
- Protocolo publicado y disponible
- Notificación enviada a trabajadores
- Proceso de aceptación iniciado

Reglas de Negocio:
- RN-001: Protocolo debe incluir perspectiva de género
- RN-002: Protocolo debe actualizarse al menos anualmente
- RN-003: Todos los trabajadores deben aceptar protocolo en 30 días
- RN-004: Mantener historial de versiones por 5 años
```

6.1.3 Endpoints API

```
# Protocolos
GET    /api/v1/protocols                          # Listar protocolos
POST   /api/v1/protocols                          # Crear protocolo
GET    /api/v1/protocols/:id                      # Obtener protocolo
PUT    /api/v1/protocols/:id                      # Actualizar protocolo
DELETE /api/v1/protocols/:id                      # Eliminar protocolo
POST   /api/v1/protocols/:id/publish              # Publicar protocolo
GET    /api/v1/protocols/:id/versions             # Historial de versiones
GET    /api/v1/protocols/:id/acknowledgments      # Aceptaciones
POST   /api/v1/protocols/:id/acknowledge          # Aceptar protocolo (empleado)

# Denuncias (Autenticado)
GET    /api/v1/complaints                         # Listar denuncias
POST   /api/v1/complaints                         # Crear denuncia identificada
GET    /api/v1/complaints/:id                     # Obtener denuncia
PUT    /api/v1/complaints/:id                     # Actualizar denuncia
GET    /api/v1/complaints/:id/timeline            # Timeline de eventos
POST   /api/v1/complaints/:id/attachments         # Subir adjunto
GET    /api/v1/complaints/:id/attachments         # Listar adjuntos
GET    /api/v1/complaints/stats                   # Estadísticas

# Denuncias (Público - Anónimas)
POST   /api/v1/public/complaints                  # Crear denuncia anónima
GET    /api/v1/public/complaints/:token           # Consultar por token
POST   /api/v1/public/complaints/:token/message   # Enviar mensaje al investigador
GET    /api/v1/public/complaints/:token/messages  # Ver mensajes

# Investigaciones
GET    /api/v1/investigations                     # Listar investigaciones
POST   /api/v1/investigations                     # Crear investigación
GET    /api/v1/investigations/:id                 # Obtener investigación
PUT    /api/v1/investigations/:id                 # Actualizar investigación
POST   /api/v1/investigations/:id/assign          # Asignar investigador
POST   /api/v1/investigations/:id/accept          # Aceptar asignación
POST   /api/v1/investigations/:id/conflict-check  # Declarar conflicto de interés
POST   /api/v1/investigations/:id/interviews      # Registrar entrevista
GET    /api/v1/investigations/:id/interviews      # Listar entrevistas
POST   /api/v1/investigations/:id/findings        # Agregar hallazgo
POST   /api/v1/investigations/:id/measures        # Proponer medidas
POST   /api/v1/investigations/:id/close           # Cerrar investigación
GET    /api/v1/investigations/:id/report          # Descargar informe

# Riesgos Psicosociales
GET    /api/v1/psychosocial-risks                 # Listar evaluaciones
POST   /api/v1/psychosocial-risks                 # Crear evaluación
GET    /api/v1/psychosocial-risks/:id             # Obtener evaluación
GET    /api/v1/psychosocial-risks/:id/results     # Resultados
GET    /api/v1/psychosocial-risks/:id/heatmap     # Mapa de calor
POST   /api/v1/psychosocial-risks/:id/action-plan # Plan de intervención

# Reportes Regulatorios
POST   /api/v1/reports/regulatory/suseso          # Generar reporte SUSESO
POST   /api/v1/reports/regulatory/isl             # Generar reporte ISL
POST   /api/v1/reports/regulatory/dt              # Generar reporte Dir. Trabajo
GET    /api/v1/reports/regulatory/templates       # Templates disponibles
GET    /api/v1/reports/regulatory/history         # Historial de reportes
```

6.1.4 Reglas de Negocio

ID	Regla	Validación
RN-LK-001	Denuncias anónimas deben generar token de seguimiento único	Automático
RN-LK-002	Investigador no puede ser el denunciante ni el acusado	Validación en asignación
RN-LK-003	Investigador debe declarar conflicto de interés antes de aceptar	Workflow obligatorio
RN-LK-004	Investigación debe completarse en plazo legal (30 días hábiles)	Alerta automática
RN-LK-005	Toda investigación debe incluir análisis con perspectiva de género	Campo obligatorio
RN-LK-006	Denunciante debe recibir notificación de inicio de investigación	Automático
RN-LK-007	Atención psicológica debe ofrecerse en casos de acoso sexual	Automático
RN-LK-008	Medidas propuestas deben documentarse y dar seguimiento	Workflow
RN-LK-009	Reportes a organismos deben generarse en formato oficial	Templates
RN-LK-010	Evidencias deben mantener cadena de custodia digital	Inmutabilidad + hash

6.1.5 Permisos Requeridos

```
const LEY_KARIN_PERMISSIONS = {
  // Protocolos
  'protocols.read': 'Ver protocolos',
  'protocols.create': 'Crear protocolos',
  'protocols.update': 'Editar protocolos',
  'protocols.publish': 'Publicar protocolos',
  'protocols.delete': 'Eliminar protocolos',

  // Denuncias
  'complaints.read': 'Ver denuncias',
  'complaints.read.own': 'Ver propias denuncias',
  'complaints.read.all': 'Ver todas las denuncias',
  'complaints.create': 'Crear denuncias',
  'complaints.update': 'Actualizar denuncias',
  'complaints.delete': 'Eliminar denuncias',
  'complaints.assign': 'Asignar investigadores',
  'complaints.export': 'Exportar denuncias',

  // Investigaciones
  'investigations.read': 'Ver investigaciones',
  'investigations.read.confidential': 'Ver detalles confidenciales',
  'investigations.create': 'Crear investigaciones',
  'investigations.update': 'Actualizar investigaciones',
  'investigations.close': 'Cerrar investigaciones',
  'investigations.assign': 'Asignar investigadores',

  // Riesgos Psicosociales
  'psychosocial.read': 'Ver evaluaciones psicosociales',
  'psychosocial.create': 'Crear evaluaciones',
  'psychosocial.manage': 'Gestionar planes de intervención',

  // Reportes
  'reports.regulatory': 'Generar reportes regulatorios',
};
```

6.2 Módulo: Sistema de Encuestas

6.2.1 Descripción Funcional

Motor de encuestas empresarial flexible que soporta:
- Constructor visual drag & drop
- Múltiples tipos de preguntas
- Lógica condicional avanzada
- Distribución multicanal
- Modo offline (PWA)
- Analytics en tiempo real

6.2.2 Endpoints API

```
# Encuestas
GET    /api/v1/surveys                    # Listar encuestas
POST   /api/v1/surveys                    # Crear encuesta
GET    /api/v1/surveys/:id                # Obtener encuesta
PUT    /api/v1/surveys/:id                # Actualizar encuesta
DELETE /api/v1/surveys/:id                # Eliminar encuesta
POST   /api/v1/surveys/:id/duplicate      # Duplicar encuesta
POST   /api/v1/surveys/:id/publish        # Publicar
POST   /api/v1/surveys/:id/close          # Cerrar
GET    /api/v1/surveys/:id/preview        # Vista previa

# Secciones
GET    /api/v1/surveys/:id/sections       # Listar secciones
POST   /api/v1/surveys/:id/sections       # Crear sección
PUT    /api/v1/surveys/:id/sections/:sid  # Actualizar sección
DELETE /api/v1/surveys/:id/sections/:sid  # Eliminar sección
POST   /api/v1/surveys/:id/sections/reorder # Reordenar

# Preguntas
GET    /api/v1/surveys/:id/questions      # Listar preguntas
POST   /api/v1/surveys/:id/questions      # Crear pregunta
PUT    /api/v1/surveys/:id/questions/:qid # Actualizar pregunta
DELETE /api/v1/surveys/:id/questions/:qid # Eliminar pregunta
POST   /api/v1/surveys/:id/questions/reorder # Reordenar
POST   /api/v1/surveys/:id/questions/bulk # Importar masivo

# Distribución
POST   /api/v1/surveys/:id/distribute     # Enviar encuesta
GET    /api/v1/surveys/:id/distributions  # Historial de envíos
POST   /api/v1/surveys/:id/remind         # Enviar recordatorio

# Respuestas (Admin)
GET    /api/v1/surveys/:id/responses      # Listar respuestas
GET    /api/v1/surveys/:id/responses/:rid # Ver respuesta
DELETE /api/v1/surveys/:id/responses/:rid # Eliminar respuesta
GET    /api/v1/surveys/:id/responses/export # Exportar

# Respuestas (Público)
GET    /api/v1/public/surveys/:token      # Obtener encuesta
POST   /api/v1/public/surveys/:token/responses # Enviar respuesta
PUT    /api/v1/public/surveys/:token/responses/:rid # Actualizar (si permitido)

# Analytics
GET    /api/v1/surveys/:id/analytics      # Resumen analytics
GET    /api/v1/surveys/:id/analytics/questions/:qid # Por pregunta
GET    /api/v1/surveys/:id/analytics/segments # Por segmento
GET    /api/v1/surveys/:id/analytics/trends # Tendencias

# Templates
GET    /api/v1/survey-templates           # Listar templates
GET    /api/v1/survey-templates/:id       # Obtener template
POST   /api/v1/survey-templates/:id/use   # Usar template
```

6.2.3 Reglas de Negocio

ID	Regla	Implementación
RN-SV-001	Encuesta publicada no puede modificar preguntas	Estado inmutable
RN-SV-002	Token de respuesta único por respondente	Generación UUID
RN-SV-003	Respuestas anónimas no almacenan identificador	Null en respondent_id
RN-SV-004	Lógica condicional debe validarse sin ciclos	Validación de grafo
RN-SV-005	Piping solo de preguntas anteriores	Validación de orden
RN-SV-006	Encuesta cerrada no acepta respuestas	Validación de estado

---

6.3 Módulo: Analytics y KPIs

6.3.1 Descripción Funcional

Sistema de inteligencia de negocio con:
- KPIs configurables y calculados
- Dashboards personalizables
- Visualizaciones interactivas
- Predicciones con IA/ML
- Alertas automáticas
- Benchmarking sectorial

6.3.2 Endpoints API

```
# KPIs
GET    /api/v1/kpis                       # Listar KPIs
POST   /api/v1/kpis                       # Crear KPI
GET    /api/v1/kpis/:id                   # Obtener KPI
PUT    /api/v1/kpis/:id                   # Actualizar KPI
DELETE /api/v1/kpis/:id                   # Eliminar KPI
GET    /api/v1/kpis/:id/values            # Valores históricos
GET    /api/v1/kpis/:id/trend             # Tendencia
POST   /api/v1/kpis/:id/calculate         # Forzar cálculo

# Dashboards
GET    /api/v1/dashboards                 # Listar dashboards
POST   /api/v1/dashboards                 # Crear dashboard
GET    /api/v1/dashboards/:id             # Obtener dashboard
PUT    /api/v1/dashboards/:id             # Actualizar dashboard
DELETE /api/v1/dashboards/:id             # Eliminar dashboard
POST   /api/v1/dashboards/:id/duplicate   # Duplicar
GET    /api/v1/dashboards/:id/widgets     # Widgets

# Widgets
POST   /api/v1/dashboards/:id/widgets     # Crear widget
PUT    /api/v1/dashboards/:id/widgets/:wid # Actualizar widget
DELETE /api/v1/dashboards/:id/widgets/:wid # Eliminar widget
POST   /api/v1/dashboards/:id/widgets/layout # Actualizar layout

# Analytics General
GET    /api/v1/analytics/summary          # Resumen ejecutivo
GET    /api/v1/analytics/compliance       # Métricas compliance
GET    /api/v1/analytics/climate          # Métricas clima
GET    /api/v1/analytics/edi              # Métricas EDI
GET    /api/v1/analytics/trends           # Tendencias generales

# Predicciones IA
GET    /api/v1/analytics/predictions/turnover  # Predicción rotación
GET    /api/v1/analytics/predictions/risks     # Predicción riesgos
GET    /api/v1/analytics/predictions/climate   # Predicción clima

# Alertas
GET    /api/v1/alerts                     # Listar alertas
POST   /api/v1/alerts                     # Crear alerta
PUT    /api/v1/alerts/:id                 # Actualizar alerta
DELETE /api/v1/alerts/:id                 # Eliminar alerta
POST   /api/v1/alerts/:id/acknowledge     # Reconocer alerta

# Benchmarking
GET    /api/v1/analytics/benchmark        # Benchmark sectorial
GET    /api/v1/analytics/benchmark/compare # Comparar con industria
```

---

7. ESTÁNDARES DE CÓDIGO Y BUENAS PRÁCTICAS

7.1 Backend (NestJS + TypeScript)

7.1.1 Convenciones de Nomenclatura

```
// ============================================
// ARCHIVOS
// ============================================
// Kebab-case para archivos
user.controller.ts
user.service.ts
user.repository.ts
user.entity.ts
user.dto.ts
user.interface.ts
user.guard.ts
user.decorator.ts
user.pipe.ts
user.interceptor.ts
user.filter.ts

// Sufijos obligatorios
*.controller.ts    // Controllers
*.service.ts       // Services
*.repository.ts    // Repositories
*.entity.ts        // Entities
*.dto.ts           // DTOs
*.interface.ts     // Interfaces
*.guard.ts         // Guards
*.module.ts        // Modules

// ============================================
// CLASES
// ============================================
// PascalCase para clases
class UserController {}
class UserService {}
class CreateUserDto {}
class User {} // Entidad

// Sufijos descriptivos
class JwtAuthGuard extends AuthGuard {}
class LoggingInterceptor implements NestInterceptor {}
class ValidationPipe implements PipeTransform {}
class HttpExceptionFilter implements ExceptionFilter {}

// ============================================
// INTERFACES
// ============================================
// PascalCase, prefijo I solo si es necesario distinguir
interface User {}
interface IUserRepository {} // Si hay clase User
interface CreateUserOptions {}

// ============================================
// VARIABLES Y FUNCIONES
// ============================================
// camelCase
const userName = 'John';
function getUserById(id: string) {}
async function createUser(data: CreateUserDto) {}

// ============================================
// CONSTANTES
// ============================================
// UPPER_SNAKE_CASE para constantes
const MAX_LOGIN_ATTEMPTS = 5;
```

```
const DEFAULT_PAGE_SIZE = 20;
const API_VERSION = 'v1';

// ============================================
// ENUMS
// ============================================
// PascalCase para enum, UPPER_SNAKE_CASE para valores
enum UserStatus {
  ACTIVE = 'ACTIVE',
  INACTIVE = 'INACTIVE',
  PENDING = 'PENDING',
}

enum ComplaintType {
  WORKPLACE_HARASSMENT = 'WORKPLACE_HARASSMENT',
  SEXUAL_HARASSMENT = 'SEXUAL_HARASSMENT',
}

// ============================================
// TIPOS
// ============================================
// PascalCase
type UserId = string;
type ComplaintStatus = 'RECEIVED' | 'UNDER_INVESTIGATION' | 'CLOSED';
```

7.1.2 Estructura de Archivos por Módulo

```
modules/
└── complaints/
    ├── complaints.module.ts          # Módulo NestJS
    │
    ├── application/                  # Capa de aplicación
    │   ├── commands/                 # Comandos (CQRS)
    │   │   ├── create-complaint/
    │   │   │   ├── create-complaint.command.ts
    │   │   │   └── create-complaint.handler.ts
    │   │   └── index.ts
    │   ├── queries/                  # Queries (CQRS)
    │   │   ├── get-complaints/
    │   │   │   ├── get-complaints.query.ts
    │   │   │   └── get-complaints.handler.ts
    │   │   └── index.ts
    │   ├── dto/                      # DTOs
    │   │   ├── create-complaint.dto.ts
    │   │   ├── update-complaint.dto.ts
    │   │   ├── complaint-response.dto.ts
    │   │   ├── complaint-filters.dto.ts
    │   │   └── index.ts
    │   ├── mappers/                  # Mappers
    │   │   └── complaint.mapper.ts
    │   └── services/                 # Application services
    │       └── complaint-notification.service.ts
    │
    ├── domain/                       # Capa de dominio
    │   ├── entities/                 # Entidades de dominio
    │   │   └── complaint.entity.ts
    │   ├── value-objects/            # Value Objects
    │   │   ├── complaint-type.vo.ts
    │   │   ├── complaint-status.vo.ts
    │   │   └── tracking-token.vo.ts
    │   ├── aggregates/               # Aggregates
    │   │   └── complaint.aggregate.ts
    │   ├── events/                   # Eventos de dominio
    │   │   ├── complaint-created.event.ts
    │   │   └── complaint-status-changed.event.ts
    │   ├── repositories/             # Interfaces de repositorio
    │   │   └── complaint.repository.interface.ts
    │   ├── services/                 # Servicios de dominio
    │   │   └── anonymizer.service.interface.ts
    │   └── exceptions/               # Excepciones de dominio
    │       └── complaint.exceptions.ts
    │
    ├── infrastructure/               # Capa de infraestructura
    │   ├── persistence/              # Implementación de persistencia
    │   │   ├── complaint.repository.ts
    │   │   ├── complaint.schema.ts   # Schema TypeORM/Prisma
    │   │   └── complaint.mapper.ts   # Mapper DB <-> Domain
    │   └── services/                 # Implementación de servicios
    │       └── token-anonymizer.service.ts
    │
    └── presentation/                 # Capa de presentación
        ├── controllers/
        │   ├── complaints.controller.ts
        │   └── complaints-public.controller.ts
        ├── guards/
        │   └── complaint-access.guard.ts
        └── decorators/
            └── complaint-id.decorator.ts
```

7.1.3 DTOs con Validación

```typescript
// dto/create-complaint.dto.ts
import {
  IsString,
  IsEnum,
  IsBoolean,
  IsOptional,
  IsUUID,
  IsDateString,
  MinLength,
  MaxLength,
  ValidateNested,
  IsArray,
  ArrayMaxSize,
} from 'class-validator';
import { Type } from 'class-transformer';
import { ApiProperty, ApiPropertyOptional } from '@nestjs/swagger';

export class CreateComplaintDto {
  @ApiProperty({
    enum: ComplaintType,
    description: 'Tipo de denuncia según Ley Karin',
    example: ComplaintType.WORKPLACE_HARASSMENT,
  })
  @IsEnum(ComplaintType, {
    message: 'Tipo de denuncia inválido',
  })
  type: ComplaintType;

  @ApiProperty({
    description: 'Descripción detallada del incidente',
    minLength: 50,
    maxLength: 5000,
    example: 'Descripción del incidente ocurrido...',
  })
  @IsString()
  @MinLength(50, {
    message: 'La descripción debe tener al menos 50 caracteres',
  })
  @MaxLength(5000, {
    message: 'La descripción no puede exceder 5000 caracteres',
  })
  description: string;

  @ApiPropertyOptional({
    description: 'Indica si la denuncia es anónima',
    default: false,
  })
  @IsOptional()
  @IsBoolean()
  isAnonymous?: boolean = false;

  @ApiPropertyOptional({
    description: 'Fecha del incidente',
    example: '2024-01-15',
  })
  @IsOptional()
  @IsDateString()
  incidentDate?: string;

  @ApiPropertyOptional({
    description: 'Ubicación del incidente',
    maxLength: 200,
```

```
  })
  @IsOptional()
  @IsString()
  @MaxLength(200)
  incidentLocation?: string;

  @ApiPropertyOptional({
    description: 'Información del acusado',
    type: AccusedInfoDto,
  })
  @IsOptional()
  @ValidateNested()
  @Type(() => AccusedInfoDto)
  accusedInfo?: AccusedInfoDto;

  @ApiPropertyOptional({
    description: 'IDs de testigos',
    type: [String],
    maxItems: 10,
  })
  @IsOptional()
  @IsArray()
  @ArrayMaxSize(10)
  @IsUUID('4', { each: true })
  witnessIds?: string[];
}

export class AccusedInfoDto {
  @ApiPropertyOptional()
  @IsOptional()
  @IsUUID()
  userId?: string;

  @ApiPropertyOptional()
  @IsOptional()
  @IsString()
  @MaxLength(100)
  name?: string;

  @ApiPropertyOptional()
  @IsOptional()
  @IsString()
  @MaxLength(100)
  department?: string;

  @ApiPropertyOptional()
  @IsOptional()
  @IsString()
  @MaxLength(100)
  position?: string;
}
```

7.1.4 Controller con Documentación

```
// controllers/complaints.controller.ts
import {
  Controller,
  Get,
  Post,
  Put,
  Delete,
  Body,
  Param,
  Query,
  UseGuards,
  HttpStatus,
  HttpCode,
  ParseUUIDPipe,
} from '@nestjs/common';
import {
  ApiTags,
  ApiOperation,
  ApiResponse,
  ApiBearerAuth,
  ApiParam,
  ApiQuery,
} from '@nestjs/swagger';
import { CommandBus, QueryBus } from '@nestjs/cqrs';

@ApiTags('Complaints')
@ApiBearerAuth()
@Controller('api/v1/complaints')
@UseGuards(JwtAuthGuard, PermissionsGuard)
export class ComplaintsController {
  constructor(
    private readonly commandBus: CommandBus,
    private readonly queryBus: QueryBus,
  ) {}

  @Post()
  @Permissions('complaints.create')
  @HttpCode(HttpStatus.CREATED)
  @ApiOperation({
    summary: 'Create a new complaint',
    description: 'Creates a new complaint. Can be anonymous or identified.',
  })
  @ApiResponse({
    status: HttpStatus.CREATED,
    description: 'Complaint created successfully',
    type: ComplaintResponseDto,
  })
  @ApiResponse({
    status: HttpStatus.BAD_REQUEST,
    description: 'Validation error',
  })
  @ApiResponse({
    status: HttpStatus.UNAUTHORIZED,
    description: 'Unauthorized',
  })
  async create(
    @Body() dto: CreateComplaintDto,
    @CurrentUser() user: AuthenticatedUser,
    @CurrentTenant() tenant: TenantContext,
  ): Promise<ApiResponse<ComplaintResponseDto>> {
    const command = new CreateComplaintCommand({
      ...dto,
```

```
      tenantId: tenant.id,
      companyId: user.companyId,
      reporterId: dto.isAnonymous ? undefined : user.id,
    });

    const result = await this.commandBus.execute(command);

    return ApiResponse.created(result, 'Denuncia creada exitosamente');
  }

  @Get()
  @Permissions('complaints.read')
  @ApiOperation({
    summary: 'List complaints',
    description: 'Returns a paginated list of complaints with optional filters',
  })
  @ApiPaginatedResponse(ComplaintResponseDto)
  @ApiQuery({ name: 'status', required: false, enum: ComplaintStatus })
  @ApiQuery({ name: 'type', required: false, enum: ComplaintType })
  @ApiQuery({ name: 'dateFrom', required: false, type: String })
  @ApiQuery({ name: 'dateTo', required: false, type: String })
  async findAll(
    @Query() pagination: PaginationDto,
    @Query() filters: ComplaintFiltersDto,
    @CurrentUser() user: AuthenticatedUser,
    @CurrentTenant() tenant: TenantContext,
  ): Promise<PaginatedResponse<ComplaintResponseDto>> {
    const query = new GetComplaintsQuery({
      tenantId: tenant.id,
      companyIds: user.companyIds,
      userId: user.hasPermission('complaints.read.all') ? undefined : user.id,
      ...pagination,
      ...filters,
    });

    return this.queryBus.execute(query);
  }

  @Get(':id')
  @Permissions('complaints.read')
  @ApiOperation({ summary: 'Get complaint by ID' })
  @ApiParam({ name: 'id', type: String, format: 'uuid' })
  @ApiResponse({
    status: HttpStatus.OK,
    type: ComplaintDetailResponseDto,
  })
  @ApiResponse({
    status: HttpStatus.NOT_FOUND,
    description: 'Complaint not found',
  })
  async findOne(
    @Param('id', ParseUUIDPipe) id: string,
    @CurrentUser() user: AuthenticatedUser,
  ): Promise<ApiResponse<ComplaintDetailResponseDto>> {
    const query = new GetComplaintByIdQuery({ id, userId: user.id });
    const result = await this.queryBus.execute(query);

    if (!result) {
      throw new NotFoundException('Denuncia no encontrada');
    }

    return ApiResponse.ok(result);
  }
```

```
  @Get(':id/timeline')
  @Permissions('complaints.read')
  @ApiOperation({ summary: 'Get complaint timeline' })
  async getTimeline(
    @Param('id', ParseUUIDPipe) id: string,
  ): Promise<ApiResponse<TimelineEventDto[]>> {
    const query = new GetComplaintTimelineQuery({ complaintId: id });
    const result = await this.queryBus.execute(query);

    return ApiResponse.ok(result);
  }
}
```

7.1.5 Manejo de Errores

```
// shared/exceptions/business.exception.ts
export class BusinessException extends HttpException {
  constructor(
    public readonly code: string,
    message: string,
    public readonly details?: Record<string, any>,
  ) {
    super(
      {
        code,
        message,
        details,
      },
      HttpStatus.UNPROCESSABLE_ENTITY,
    );
  }
}

// Excepciones específicas del dominio
export class ComplaintNotFoundException extends BusinessException {
  constructor(complaintId: string) {
    super(
      'COMPLAINT_NOT_FOUND',
      `Denuncia con ID ${complaintId} no encontrada`,
      { complaintId },
    );
  }
}

export class InvestigatorConflictException extends BusinessException {
  constructor(investigatorId: string, reason: string) {
    super(
      'INVESTIGATOR_CONFLICT_OF_INTEREST',
      'El investigador tiene conflicto de interés',
      { investigatorId, reason },
    );
  }
}

export class ComplaintStatusTransitionException extends BusinessException {
  constructor(currentStatus: string, targetStatus: string) {
    super(
      'INVALID_STATUS_TRANSITION',
      `No es posible cambiar de estado ${currentStatus} a ${targetStatus}`,
      { currentStatus, targetStatus },
    );
  }
}
```

7.1.6 Testing Unitario

```
// __tests__/complaints.service.spec.ts
import { Test, TestingModule } from '@nestjs/testing';
import { CreateComplaintHandler } from '../application/commands/create-complaint/
create-complaint.handler';

describe('CreateComplaintHandler', () => {
  let handler: CreateComplaintHandler;
  let complaintRepository: jest.Mocked<IComplaintRepository>;
  let eventBus: jest.Mocked<EventBus>;
  let anonymizerService: jest.Mocked<IAnonymizerService>;

  beforeEach(async () => {
    const module: TestingModule = await Test.createTestingModule({
      providers: [
        CreateComplaintHandler,
        {
          provide: 'IComplaintRepository',
          useValue: {
            save: jest.fn(),
            findById: jest.fn(),
          },
        },
        {
          provide: EventBus,
          useValue: {
            publish: jest.fn(),
          },
        },
        {
          provide: 'IAnonymizerService',
          useValue: {
            generateTrackingToken: jest.fn(),
          },
        },
      ],
    }).compile();

    handler = module.get<CreateComplaintHandler>(CreateComplaintHandler);
    complaintRepository = module.get('IComplaintRepository');
    eventBus = module.get(EventBus);
    anonymizerService = module.get('IAnonymizerService');
  });

  describe('execute', () => {
    it('should create an identified complaint successfully', async () => {
      // Arrange
      const command = new CreateComplaintCommand({
        type: ComplaintType.WORKPLACE_HARASSMENT,
        description: 'Test description with enough characters to pass validation',
        isAnonymous: false,
        tenantId: 'tenant-123',
        companyId: 'company-123',
        reporterId: 'user-123',
      });

      const savedComplaint = Complaint.create({
        ...command,
        id: 'complaint-123',
      });

      complaintRepository.save.mockResolvedValue(savedComplaint);
```

```
    // Act
    const result = await handler.execute(command);

    // Assert
    expect(result).toBeDefined();
    expect(result.id).toBe('complaint-123');
    expect(result.isAnonymous).toBe(false);
    expect(complaintRepository.save).toHaveBeenCalledTimes(1);
    expect(eventBus.publish).toHaveBeenCalledWith(
      expect.any(ComplaintCreatedEvent),
    );
  });

  it('should create an anonymous complaint with tracking token', async () => {
    // Arrange
    const command = new CreateComplaintCommand({
      type: ComplaintType.SEXUAL_HARASSMENT,
      description: 'Anonymous complaint description with enough characters',
      isAnonymous: true,
      tenantId: 'tenant-123',
      companyId: 'company-123',
    });

    const trackingToken = 'ABC123XYZ';
    anonymizerService.generateTrackingToken.mockResolvedValue(trackingToken);

    const savedComplaint = Complaint.create({
      ...command,
      id: 'complaint-456',
      trackingToken,
    });

    complaintRepository.save.mockResolvedValue(savedComplaint);

    // Act
    const result = await handler.execute(command);

    // Assert
    expect(result.isAnonymous).toBe(true);
    expect(result.trackingToken).toBe(trackingToken);
    expect(result.reporterId).toBeNull();
    expect(anonymizerService.generateTrackingToken).toHaveBeenCalled();
  });

  it('should throw error for invalid complaint type', async () => {
    // Arrange
    const command = new CreateComplaintCommand({
      type: 'INVALID_TYPE' as ComplaintType,
      description: 'Test description',
      tenantId: 'tenant-123',
      companyId: 'company-123',
    });

    // Act & Assert
    await expect(handler.execute(command)).rejects.toThrow(
      ValidationException,
    );
  });

  it('should throw error when description is too short', async () => {
    // Arrange
    const command = new CreateComplaintCommand({
      type: ComplaintType.WORKPLACE_HARASSMENT,
```

```
        description: 'Too short',
        tenantId: 'tenant-123',
        companyId: 'company-123',
      });

      // Act & Assert
      await expect(handler.execute(command)).rejects.toThrow(
        'Description must be at least 50 characters',
      );
    });
  });
});
```

7.2 Frontend (React + TypeScript)

7.2.1 Convenciones de Nomenclatura

```
// ============================================
// ARCHIVOS
// ============================================
// PascalCase para componentes
ComplaintsList.tsx
ComplaintForm.tsx
ComplaintCard.tsx

// camelCase para hooks, utils, services
useComplaints.ts
useComplaintForm.ts
complaints.service.ts
date.utils.ts

// Kebab-case para estilos CSS modules
complaints-list.module.css

// ============================================
// COMPONENTES
// ============================================
// PascalCase
const ComplaintsList: React.FC<ComplaintsListProps> = () => {};
const ComplaintCard: React.FC<ComplaintCardProps> = () => {};

// ============================================
// HOOKS
// ============================================
// camelCase con prefijo use
const useComplaints = () => {};
const useComplaintForm = () => {};
const useDebounce = () => {};

// ============================================
// INTERFACES Y TYPES
// ============================================
// PascalCase, sufijo Props para props de componentes
interface ComplaintsListProps {}
interface ComplaintFormProps {}
type ComplaintStatus = 'RECEIVED' | 'UNDER_INVESTIGATION' | 'CLOSED';

// ============================================
// CONSTANTES
// ============================================
const QUERY_KEYS = {
  COMPLAINTS: 'complaints',
  COMPLAINT_DETAIL: 'complaint-detail',
};

const STATUS_LABELS: Record<ComplaintStatus, string> = {
  RECEIVED: 'Recibida',
  UNDER_INVESTIGATION: 'En Investigación',
  CLOSED: 'Cerrada',
};
```

7.2.2 Estructura de Componentes

```tsx
// features/complaints/components/ComplaintCard/ComplaintCard.tsx
import { memo } from 'react';
import { formatDate } from '@/shared/utils/date.utils';
import { Badge } from '@/shared/components/ui/Badge';
import { Card } from '@/shared/components/ui/Card';
import { ComplaintStatusBadge } from '../ComplaintStatusBadge';
import { ComplaintCardProps } from './ComplaintCard.types';
import styles from './ComplaintCard.module.css';

/**
 * Card component to display complaint summary
 * @param complaint - Complaint data to display
 * @param onClick - Callback when card is clicked
 * @param isSelected - Whether the card is currently selected
 */
export const ComplaintCard = memo<ComplaintCardProps>(
  ({ complaint, onClick, isSelected = false }) => {
    const handleClick = () => {
      onClick?.(complaint.id);
    };

    const handleKeyDown = (e: React.KeyboardEvent) => {
      if (e.key === 'Enter' || e.key === ' ') {
        e.preventDefault();
        handleClick();
      }
    };

    return (
      <Card
        className={cn(styles.card, isSelected && styles.selected)}
        onClick={handleClick}
        onKeyDown={handleKeyDown}
        role="button"
        tabIndex={0}
        aria-pressed={isSelected}
        data-testid="complaint-card"
      >
        <div className={styles.header}>
          <span className={styles.id}>#{complaint.id.slice(0, 8)}</span>
          <ComplaintStatusBadge status={complaint.status} />
        </div>

        <div className={styles.content}>
          <Badge variant="outline">{COMPLAINT_TYPE_LABELS[complaint.type]}</Badge>
          <p className={styles.description}>
            {truncate(complaint.description, 150)}
          </p>
        </div>

        <div className={styles.footer}>
          <span className={styles.date}>
            {formatDate(complaint.createdAt, 'dd MMM yyyy')}
          </span>
          {complaint.isAnonymous && (
            <Badge variant="secondary">Anónima</Badge>
          )}
        </div>
      </Card>
    );
  },
);
```

```
ComplaintCard.displayName = 'ComplaintCard';
```

7.2.3 Custom Hooks

```typescript
// features/complaints/hooks/useComplaints.ts
import { useQuery, useMutation, useQueryClient } from '@tanstack/react-query';
import { useState, useCallback, useMemo } from 'react';
import { complaintsService } from '@/services/complaints.service';
import { QUERY_KEYS } from '@/shared/constants/query-keys';
import { useToast } from '@/shared/hooks/useToast';

interface UseComplaintsOptions {
  initialFilters?: ComplaintFilters;
  pageSize?: number;
}

export const useComplaints = (options: UseComplaintsOptions = {}) => {
  const { initialFilters = {}, pageSize = 20 } = options;
  const queryClient = useQueryClient();
  const { showToast } = useToast();

  // State
  const [filters, setFilters] = useState<ComplaintFilters>(initialFilters);
  const [page, setPage] = useState(1);

  // Query
  const {
    data,
    isLoading,
    isError,
    error,
    refetch,
  } = useQuery({
    queryKey: [QUERY_KEYS.COMPLAINTS, filters, page, pageSize],
    queryFn: () => complaintsService.getAll({
      ...filters,
      page,
      pageSize,
    }),
    staleTime: 5 * 60 * 1000, // 5 minutes
    keepPreviousData: true,
  });

  // Mutations
  const createMutation = useMutation({
    mutationFn: complaintsService.create,
    onSuccess: (newComplaint) => {
      queryClient.invalidateQueries([QUERY_KEYS.COMPLAINTS]);
      showToast({
        type: 'success',
        message: 'Denuncia creada exitosamente',
      });
    },
    onError: (error: ApiError) => {
      showToast({
        type: 'error',
        message: error.message || 'Error al crear la denuncia',
      });
    },
  });

  const updateMutation = useMutation({
    mutationFn: ({ id, data }: { id: string; data: UpdateComplaintDto }) =>
      complaintsService.update(id, data),
    onSuccess: (updatedComplaint) => {
      queryClient.invalidateQueries([QUERY_KEYS.COMPLAINTS]);
```

```
      queryClient.setQueryData(
        [QUERY_KEYS.COMPLAINT_DETAIL, updatedComplaint.id],
        updatedComplaint,
      );
      showToast({
        type: 'success',
        message: 'Denuncia actualizada',
      });
    },
  });

  // Callbacks
  const updateFilters = useCallback((newFilters: Partial<ComplaintFilters>) => {
    setFilters((prev) => ({ ...prev, ...newFilters }));
    setPage(1); // Reset to first page
  }, []);

  const clearFilters = useCallback(() => {
    setFilters({});
    setPage(1);
  }, []);

  const goToPage = useCallback((newPage: number) => {
    setPage(newPage);
  }, []);

  // Derived state
  const pagination = useMemo(() => ({
    currentPage: page,
    totalPages: data?.totalPages || 1,
    totalItems: data?.total || 0,
    hasNextPage: page < (data?.totalPages || 1),
    hasPrevPage: page > 1,
  }), [page, data]);

  return {
    // Data
    complaints: data?.data || [],
    pagination,

    // Loading states
    isLoading,
    isError,
    error,
    isCreating: createMutation.isLoading,
    isUpdating: updateMutation.isLoading,

    // Actions
    createComplaint: createMutation.mutateAsync,
    updateComplaint: updateMutation.mutateAsync,

    // Filters
    filters,
    updateFilters,
    clearFilters,

    // Pagination
    goToPage,
    nextPage: () => goToPage(page + 1),
    prevPage: () => goToPage(page - 1),

    // Refresh
    refetch,
```

```
  };
};
```

7.2.4 Testing de Componentes

```tsx
// features/complaints/components/ComplaintCard/__tests__/ComplaintCard.test.tsx
import { render, screen, fireEvent } from '@testing-library/react';
import userEvent from '@testing-library/user-event';
import { ComplaintCard } from '../ComplaintCard';
import { mockComplaint } from '@/tests/mocks/complaints';

describe('ComplaintCard', () => {
  const defaultProps = {
    complaint: mockComplaint,
    onClick: jest.fn(),
  };

  beforeEach(() => {
    jest.clearAllMocks();
  });

  it('renders complaint information correctly', () => {
    render(<ComplaintCard {...defaultProps} />);

    expect(screen.getByText(/#{mockComplaint.id.slice(0, 8)}/)).toBeInTheDocument();
    expect(screen.getByText(mockComplaint.description.slice(0, 150))).toBeInTheDocu-
ment();
    expect(screen.getByTestId('complaint-status-badge')).toHaveTextContent('Recibida')
;
  });

  it('shows anonymous badge for anonymous complaints', () => {
    const anonymousComplaint = { ...mockComplaint, isAnonymous: true };
    render(<ComplaintCard {...defaultProps} complaint={anonymousComplaint} />);

    expect(screen.getByText('Anónima')).toBeInTheDocument();
  });

  it('calls onClick when clicked', async () => {
    const user = userEvent.setup();
    render(<ComplaintCard {...defaultProps} />);

    await user.click(screen.getByTestId('complaint-card'));

    expect(defaultProps.onClick).toHaveBeenCalledWith(mockComplaint.id);
  });

  it('calls onClick when Enter key is pressed', async () => {
    const user = userEvent.setup();
    render(<ComplaintCard {...defaultProps} />);

    const card = screen.getByTestId('complaint-card');
    card.focus();
    await user.keyboard('{Enter}');

    expect(defaultProps.onClick).toHaveBeenCalledWith(mockComplaint.id);
  });

  it('applies selected styles when isSelected is true', () => {
    render(<ComplaintCard {...defaultProps} isSelected />);

    const card = screen.getByTestId('complaint-card');
    expect(card).toHaveClass('selected');
    expect(card).toHaveAttribute('aria-pressed', 'true');
  });

  it('is accessible with keyboard navigation', () => {
```

```
    render(<ComplaintCard {...defaultProps} />);

    const card = screen.getByTestId('complaint-card');
    expect(card).toHaveAttribute('role', 'button');
    expect(card).toHaveAttribute('tabIndex', '0');
  });
});
```

7.2.5 Accesibilidad (WCAG)

```
// Checklist de accesibilidad para componentes

// 1. Etiquetas ARIA apropiadas
<button aria-label="Cerrar modal">
  <CloseIcon aria-hidden="true" />
</button>

// 2. Roles semánticos
<nav role="navigation" aria-label="Navegación principal">
<main role="main">
<aside role="complementary">

// 3. Estados de focus visibles
const focusStyles = `
  outline: 2px solid var(--color-primary);
  outline-offset: 2px;
`;

// 4. Contraste de colores (mínimo 4.5:1 para texto)
// Usar herramientas como axe-core para validar

// 5. Textos alternativos
<img src={logo} alt="ComplianceHub Pro - Logo" />

// 6. Formularios accesibles
<label htmlFor="email">Correo electrónico</label>
<input
  id="email"
  type="email"
  aria-describedby="email-hint email-error"
  aria-invalid={hasError}
/>
<span id="email-hint">Ingresa tu correo corporativo</span>
{hasError && (
  <span id="email-error" role="alert">
    El correo no es válido
  </span>
)}

// 7. Navegación por teclado
const handleKeyDown = (e: KeyboardEvent) => {
  switch (e.key) {
    case 'ArrowDown':
      focusNextItem();
      break;
    case 'ArrowUp':
      focusPrevItem();
      break;
    case 'Escape':
      closeMenu();
      break;
  }
};

// 8. Anuncios de estado con live regions
<div aria-live="polite" aria-atomic="true">
  {isLoading ? 'Cargando...' : `${count} resultados encontrados`}
</div>
```

7.3 Herramientas de Calidad

7.3.1 ESLint Configuration (Backend)

```
// .eslintrc.js (Backend - NestJS)
module.exports = {
  parser: '@typescript-eslint/parser',
  parserOptions: {
    project: 'tsconfig.json',
    tsconfigRootDir: __dirname,
    sourceType: 'module',
  },
  plugins: [
    '@typescript-eslint/eslint-plugin',
    'import',
    'jest',
  ],
  extends: [
    'plugin:@typescript-eslint/recommended',
    'plugin:@typescript-eslint/recommended-requiring-type-checking',
    'plugin:import/errors',
    'plugin:import/warnings',
    'plugin:import/typescript',
    'plugin:jest/recommended',
    'prettier',
  ],
  root: true,
  env: {
    node: true,
    jest: true,
  },
  ignorePatterns: ['.eslintrc.js', 'dist', 'node_modules'],
  rules: {
    // TypeScript
    '@typescript-eslint/explicit-function-return-type': 'error',
    '@typescript-eslint/explicit-module-boundary-types': 'error',
    '@typescript-eslint/no-explicit-any': 'warn',
    '@typescript-eslint/no-unused-vars': ['error', { argsIgnorePattern: '^_' }],
    '@typescript-eslint/no-floating-promises': 'error',
    '@typescript-eslint/await-thenable': 'error',
    '@typescript-eslint/no-misused-promises': 'error',
    '@typescript-eslint/naming-convention': [
      'error',
      {
        selector: 'interface',
        format: ['PascalCase'],
      },
      {
        selector: 'class',
        format: ['PascalCase'],
      },
      {
        selector: 'enum',
        format: ['PascalCase'],
      },
      {
        selector: 'enumMember',
        format: ['UPPER_CASE'],
      },
    ],

    // Imports
    'import/order': [
      'error',
      {
        groups: [
```

```
          'builtin',
          'external',
          'internal',
          'parent',
          'sibling',
          'index',
        ],
        'newlines-between': 'always',
        alphabetize: { order: 'asc', caseInsensitive: true },
      },
    ],
    'import/no-unresolved': 'error',
    'import/no-cycle': 'error',

    // General
    'no-console': 'warn',
    'prefer-const': 'error',
    'no-duplicate-imports': 'error',
  },
};
```

7.3.2 ESLint Configuration (Frontend)

```
// .eslintrc.js (Frontend - React)
module.exports = {
  parser: '@typescript-eslint/parser',
  parserOptions: {
    ecmaVersion: 2022,
    sourceType: 'module',
    ecmaFeatures: {
      jsx: true,
    },
    project: './tsconfig.json',
  },
  plugins: [
    '@typescript-eslint',
    'react',
    'react-hooks',
    'jsx-a11y',
    'import',
  ],
  extends: [
    'eslint:recommended',
    'plugin:@typescript-eslint/recommended',
    'plugin:react/recommended',
    'plugin:react-hooks/recommended',
    'plugin:jsx-a11y/recommended',
    'plugin:import/typescript',
    'prettier',
  ],
  settings: {
    react: {
      version: 'detect',
    },
  },
  rules: {
    // React
    'react/react-in-jsx-scope': 'off',
    'react/prop-types': 'off',
    'react/jsx-no-target-blank': 'error',
    'react/jsx-key': 'error',
    'react/no-array-index-key': 'warn',
    'react/self-closing-comp': 'error',

    // React Hooks
    'react-hooks/rules-of-hooks': 'error',
    'react-hooks/exhaustive-deps': 'warn',

    // Accessibility
    'jsx-a11y/anchor-is-valid': 'error',
    'jsx-a11y/click-events-have-key-events': 'error',
    'jsx-a11y/no-static-element-interactions': 'error',

    // TypeScript
    '@typescript-eslint/no-unused-vars': ['error', { argsIgnorePattern: '^_' }],
    '@typescript-eslint/explicit-function-return-type': 'off',
    '@typescript-eslint/no-explicit-any': 'warn',
  },
};
```

7.3.3 Pre-commit Hooks (Husky + lint-staged)

```
// package.json
{
  "husky": {
    "hooks": {
      "pre-commit": "lint-staged",
      "commit-msg": "commitlint -E HUSKY_GIT_PARAMS"
    }
  },
  "lint-staged": {
    "*.{ts,tsx}": [
      "eslint --fix",
      "prettier --write"
    ],
    "*.{json,md,yml,yaml}": [
      "prettier --write"
    ],
    "*.{ts,tsx}": [
      "jest --bail --findRelatedTests"
    ]
  },
  "commitlint": {
    "extends": ["@commitlint/config-conventional"]
  }
}
```

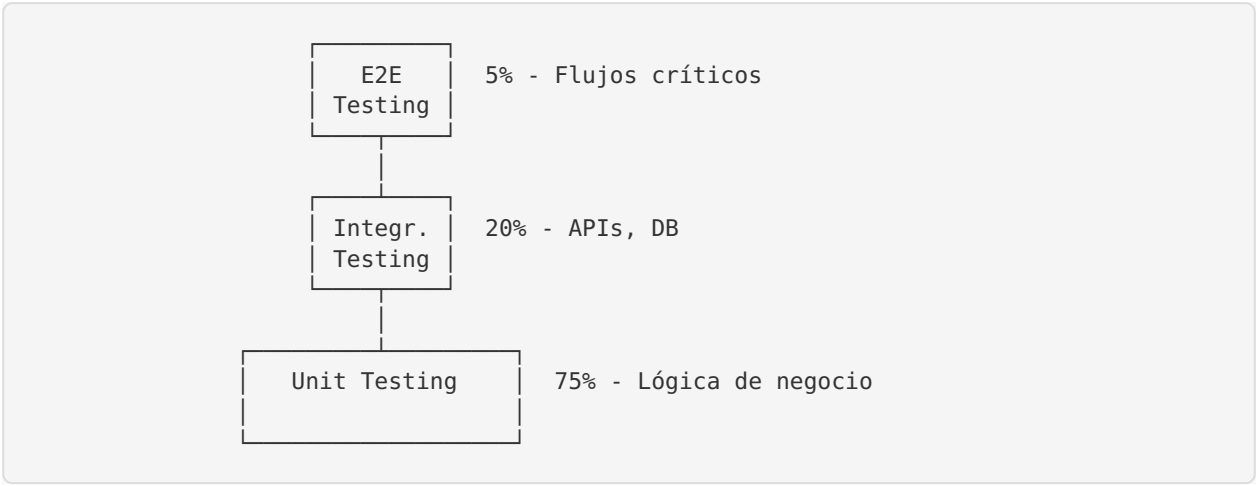
7.3.4 Code Review Checklist

```
## Code Review Checklist

### General
- [ ] El código cumple con los requisitos del ticket
- [ ] El código es legible y autoexplicativo
- [ ] No hay código comentado o dead code
- [ ] Los nombres de variables/funciones son descriptivos
- [ ] No hay magic numbers (usar constantes)

### Arquitectura
- [ ] Respeta la separación de capas (Clean Architecture)
- [ ] No hay dependencias circulares
- [ ] Los módulos están desacoplados
- [ ] Se respeta el principio de responsabilidad única

### Seguridad
- [ ] No hay credenciales hardcodeadas
- [ ] Se validan todos los inputs del usuario
- [ ] Se usan prepared statements para queries
- [ ] Se sanitizan outputs para prevenir XSS
- [ ] Se implementa autorización adecuada

### Performance
- [ ] No hay N+1 queries
- [ ] Se usa paginación para listas grandes
- [ ] Se implementa caching donde corresponde
- [ ] No hay operaciones bloqueantes en el thread principal

### Testing
- [ ] Hay tests unitarios para la lógica de negocio
- [ ] Los tests cubren casos edge
- [ ] Los tests son determinísticos
- [ ] El coverage cumple el mínimo (80%)

### API
- [ ] Los endpoints siguen convenciones REST
- [ ] Se documentan con Swagger/OpenAPI
- [ ] Se manejan errores apropiadamente
- [ ] Se retornan códigos HTTP correctos

### Frontend
- [ ] Componentes son accesibles (WCAG)
- [ ] Se manejan estados de loading/error
- [ ] Responsive design funciona
- [ ] No hay memory leaks (cleanup en useEffect)

### Documentación
- [ ] El código complejo está documentado
- [ ] Los DTOs tienen ejemplos en Swagger
- [ ] El README está actualizado si es necesario
```

---

8. ESTRATEGIA DE TESTING

8.1 Pirámide de Testing



8.2 Testing por Capa

Backend

Capa	Tipo de Test	Herramienta	Cobertura Mínima
Domain	Unit	Jest	90%
Application	Unit + Integration	Jest + Supertest	85%
Infrastructure	Integration	Jest + TestContainers	80%
Presentation	Integration	Supertest	80%

Frontend

Capa	Tipo de Test	Herramienta	Cobertura Mínima
Components	Unit	Vitest + Testing Library	80%
Hooks	Unit	Vitest	85%
Services	Unit	Vitest	90%
Pages	Integration	Vitest + MSW	70%
E2E	E2E	Playwright	Flujos críticos

8.3 Ejemplos de Tests

Test de Integración (API)

```
// __tests__/integration/complaints.integration.spec.ts
import { Test, TestingModule } from '@nestjs/testing';
import { INestApplication } from '@nestjs/common';
import * as request from 'supertest';
import { AppModule } from '../../src/app.module';
import { setupTestDatabase, cleanupTestDatabase } from '../helpers/database';
import { createTestUser, getAuthToken } from '../helpers/auth';

describe('Complaints API (Integration)', () => {
  let app: INestApplication;
  let authToken: string;
  let testUser: User;

  beforeAll(async () => {
    await setupTestDatabase();

    const moduleFixture: TestingModule = await Test.createTestingModule({
      imports: [AppModule],
    }).compile();

    app = moduleFixture.createNestApplication();
    app.useGlobalPipes(new ValidationPipe());
    await app.init();

    testUser = await createTestUser({
      permissions: ['complaints.create', 'complaints.read'],
    });
    authToken = await getAuthToken(testUser);
  });

  afterAll(async () => {
    await cleanupTestDatabase();
    await app.close();
  });

  describe('POST /api/v1/complaints', () => {
    it('should create a complaint successfully', async () => {
      const createDto = {
        type: 'WORKPLACE_HARASSMENT',
        description: 'This is a test complaint with enough characters to pass
validation requirements',
        isAnonymous: false,
        incidentDate: '2024-01-15',
      };

      const response = await request(app.getHttpServer())
        .post('/api/v1/complaints')
        .set('Authorization', `Bearer ${authToken}`)
        .set('X-Tenant-ID', testUser.tenantId)
        .send(createDto)
        .expect(201);

      expect(response.body.success).toBe(true);
      expect(response.body.data).toMatchObject({
        type: 'WORKPLACE_HARASSMENT',
        status: 'RECEIVED',
        isAnonymous: false,
      });
      expect(response.body.data.id).toBeDefined();
    });

    it('should return 400 for invalid complaint type', async () => {
```

```
      const createDto = {
        type: 'INVALID_TYPE',
        description: 'Test description',
      };

      const response = await request(app.getHttpServer())
        .post('/api/v1/complaints')
        .set('Authorization', `Bearer ${authToken}`)
        .set('X-Tenant-ID', testUser.tenantId)
        .send(createDto)
        .expect(400);

      expect(response.body.error.code).toBe('VALIDATION_ERROR');
    });

    it('should return 401 without auth token', async () => {
      const createDto = {
        type: 'WORKPLACE_HARASSMENT',
        description: 'Test description',
      };

      await request(app.getHttpServer())
        .post('/api/v1/complaints')
        .send(createDto)
        .expect(401);
    });
  });

  describe('GET /api/v1/complaints', () => {
    it('should return paginated complaints', async () => {
      const response = await request(app.getHttpServer())
        .get('/api/v1/complaints')
        .set('Authorization', `Bearer ${authToken}`)
        .set('X-Tenant-ID', testUser.tenantId)
        .query({ page: 1, pageSize: 10 })
        .expect(200);

      expect(response.body.success).toBe(true);
      expect(response.body.data).toBeInstanceOf(Array);
      expect(response.body.pagination).toMatchObject({
        page: 1,
        pageSize: 10,
      });
    });

    it('should filter by status', async () => {
      const response = await request(app.getHttpServer())
        .get('/api/v1/complaints')
        .set('Authorization', `Bearer ${authToken}`)
        .set('X-Tenant-ID', testUser.tenantId)
        .query({ status: 'RECEIVED' })
        .expect(200);

      response.body.data.forEach((complaint: any) => {
        expect(complaint.status).toBe('RECEIVED');
      });
    });
  });
});
```

Test E2E (Playwright)

```
// e2e/complaints.spec.ts
import { test, expect } from '@playwright/test';
import { loginAs, seedTestData } from './helpers';

test.describe('Complaints Flow', () => {
  test.beforeEach(async ({ page }) => {
    await seedTestData();
    await loginAs(page, 'compliance_officer');
  });

  test('should create a new complaint', async ({ page }) => {
    // Navigate to complaints
    await page.goto('/complaints');
    await expect(page.getByRole('heading', { name: 'Denuncias' })).toBeVisible();

    // Click create button
    await page.getByRole('button', { name: 'Nueva Denuncia' }).click();

    // Fill form
    await page.getByLabel('Tipo de Denuncia').selectOption('WORKPLACE_HARASSMENT');
    await page.getByLabel('Descripción').fill(
      'Esta es una descripción de prueba para validar el flujo de creación de
denuncias en el sistema'
    );
    await page.getByLabel('Fecha del Incidente').fill('2024-01-15');

    // Submit
    await page.getByRole('button', { name: 'Crear Denuncia' }).click();

    // Verify success
    await expect(page.getByText('Denuncia creada exitosamente')).toBeVisible();
    await expect(page.getByText('Acoso Laboral')).toBeVisible();
    await expect(page.getByText('Recibida')).toBeVisible();
  });

  test('should view complaint details', async ({ page }) => {
    await page.goto('/complaints');

    // Click on first complaint
    await page.getByTestId('complaint-card').first().click();

    // Verify detail page
    await expect(page.getByRole('heading', { name: /Denuncia #/ })).toBeVisible();
    await expect(page.getByText('Timeline')).toBeVisible();
    await expect(page.getByText('Información del Incidente')).toBeVisible();
  });

  test('should assign investigator to complaint', async ({ page }) => {
    await page.goto('/complaints');
    await page.getByTestId('complaint-card').first().click();

    // Click assign
    await page.getByRole('button', { name: 'Asignar Investigador' }).click();

    // Select investigator
    await page.getByLabel('Investigador').selectOption({ label: 'Juan Pérez' });
    await page.getByRole('button', { name: 'Confirmar' }).click();

    // Verify assignment
    await expect(page.getByText('Investigador asignado')).toBeVisible();
    await expect(page.getByText('En Investigación')).toBeVisible();
```

```
  });
});
```

---

9. CI/CD Y DEVOPS

9.1 Pipeline de CI/CD

```yaml
# .github/workflows/ci-cd.yml
name: CI/CD Pipeline

on:
  push:
    branches: [main, develop]
  pull_request:
    branches: [main, develop]

env:
  NODE_VERSION: '20'
  REGISTRY: ghcr.io
  IMAGE_NAME: ${{ github.repository }}

jobs:
  # ============================================
  # LINT & TYPE CHECK
  # ============================================
  lint:
    name: Lint & Type Check
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4

      - name: Setup Node.js
        uses: actions/setup-node@v4
        with:
          node-version: ${{ env.NODE_VERSION }}
          cache: 'npm'

      - name: Install dependencies
        run: npm ci

      - name: Run ESLint
        run: npm run lint

      - name: Run TypeScript check
        run: npm run type-check

  # ============================================
  # UNIT TESTS
  # ============================================
  test-unit:
    name: Unit Tests
    runs-on: ubuntu-latest
    needs: lint
    steps:
      - uses: actions/checkout@v4

      - name: Setup Node.js
        uses: actions/setup-node@v4
        with:
          node-version: ${{ env.NODE_VERSION }}
          cache: 'npm'

      - name: Install dependencies
        run: npm ci

      - name: Run unit tests
        run: npm run test:unit -- --coverage

      - name: Upload coverage
```

```yaml
      uses: codecov/codecov-action@v3
      with:
        files: ./coverage/lcov.info
        fail_ci_if_error: true
        verbose: true

  # ============================================
  # INTEGRATION TESTS
  # ============================================
  test-integration:
    name: Integration Tests
    runs-on: ubuntu-latest
    needs: lint
    services:
      postgres:
        image: postgres:16
        env:
          POSTGRES_USER: test
          POSTGRES_PASSWORD: test
          POSTGRES_DB: compliance_test
        ports:
          - 5432:5432
        options: >-
          --health-cmd pg_isready
          --health-interval 10s
          --health-timeout 5s
          --health-retries 5
      redis:
        image: redis:7
        ports:
          - 6379:6379

    steps:
      - uses: actions/checkout@v4

      - name: Setup Node.js
        uses: actions/setup-node@v4
        with:
          node-version: ${{ env.NODE_VERSION }}
          cache: 'npm'

      - name: Install dependencies
        run: npm ci

      - name: Run migrations
        run: npm run migration:run
        env:
          DATABASE_URL: postgres://test:test@localhost:5432/compliance_test

      - name: Run integration tests
        run: npm run test:integration
        env:
          DATABASE_URL: postgres://test:test@localhost:5432/compliance_test
          REDIS_URL: redis://localhost:6379

  # ============================================
  # E2E TESTS
  # ============================================
  test-e2e:
    name: E2E Tests
    runs-on: ubuntu-latest
    needs: [test-unit, test-integration]
    if: github.event_name == 'push' && github.ref == 'refs/heads/main'
```

```yaml
    steps:
      - uses: actions/checkout@v4

      - name: Setup Node.js
        uses: actions/setup-node@v4
        with:
          node-version: ${{ env.NODE_VERSION }}
          cache: 'npm'

      - name: Install dependencies
        run: npm ci

      - name: Install Playwright browsers
        run: npx playwright install --with-deps

      - name: Start application
        run: npm run start:test &
        env:
          NODE_ENV: test

      - name: Wait for application
        run: npx wait-on http://localhost:3000

      - name: Run E2E tests
        run: npm run test:e2e

      - name: Upload test artifacts
        uses: actions/upload-artifact@v3
        if: failure()
        with:
          name: playwright-report
          path: playwright-report/

  # ============================================
  # BUILD & PUSH
  # ============================================
  build:
    name: Build & Push
    runs-on: ubuntu-latest
    needs: [test-unit, test-integration]
    if: github.event_name == 'push'
    permissions:
      contents: read
      packages: write
    steps:
      - uses: actions/checkout@v4

      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3

      - name: Login to Registry
        uses: docker/login-action@v3
        with:
          registry: ${{ env.REGISTRY }}
          username: ${{ github.actor }}
          password: ${{ secrets.GITHUB_TOKEN }}

      - name: Extract metadata
        id: meta
        uses: docker/metadata-action@v5
        with:
          images: ${{ env.REGISTRY }}/${{ env.IMAGE_NAME }}
          tags: |
```

```
            type=ref,event=branch
            type=sha,prefix=
            type=raw,value=latest,enable=${{ github.ref == 'refs/heads/main' }}

      - name: Build and push
        uses: docker/build-push-action@v5
        with:
          context: .
          push: true
          tags: ${{ steps.meta.outputs.tags }}
          labels: ${{ steps.meta.outputs.labels }}
          cache-from: type=gha
          cache-to: type=gha,mode=max

  # ============================================
  # DEPLOY TO STAGING
  # ============================================
  deploy-staging:
    name: Deploy to Staging
    runs-on: ubuntu-latest
    needs: build
    if: github.ref == 'refs/heads/develop'
    environment: staging
    steps:
      - uses: actions/checkout@v4

      - name: Configure AWS credentials
        uses: aws-actions/configure-aws-credentials@v4
        with:
          aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          aws-region: us-east-1

      - name: Deploy to EKS
        run: |
          aws eks update-kubeconfig --name compliance-staging
          kubectl set image deployment/api api=${{ env.REGISTRY }}/$
{{ env.IMAGE_NAME }}:${{ github.sha }}
          kubectl rollout status deployment/api

  # ============================================
  # DEPLOY TO PRODUCTION
  # ============================================
  deploy-production:
    name: Deploy to Production
    runs-on: ubuntu-latest
    needs: [build, test-e2e]
    if: github.ref == 'refs/heads/main'
    environment: production
    steps:
      - uses: actions/checkout@v4

      - name: Configure AWS credentials
        uses: aws-actions/configure-aws-credentials@v4
        with:
          aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          aws-region: us-east-1

      - name: Deploy to EKS
        run: |
          aws eks update-kubeconfig --name compliance-production
          kubectl set image deployment/api api=${{ env.REGISTRY }}/$
```

```
{{ env.IMAGE_NAME }}:${{ github.sha }}
          kubectl rollout status deployment/api

      - name: Notify deployment
        uses: slackapi/slack-github-action@v1
        with:
          payload: |
            {
              "text": "Production deployment completed",
              "blocks": [
                {
                  "type": "section",
                  "text": {
                    "type": "mrkdwn",
                    "text": "✅ *Production Deployment Successful*\n\nCommit: `$
{{ github.sha }}`\nBy: ${{ github.actor }}"
                  }
                }
              ]
            }
        env:
          SLACK_WEBHOOK_URL: ${{ secrets.SLACK_WEBHOOK }}
```

9.2 Ambientes

Ambiente	URL	Propósito	Deploy
Development	localhost	Desarrollo local	Manual
Staging	staging.compliancehub.com	QA y pruebas	Push a `develop`
Production	app.compliancehub.com	Producción	Push a `main`

9.3 Feature Flags

```
// Configuración de Feature Flags
const FEATURE_FLAGS = {
  // Flags por ambiente
  development: {
    AI_PREDICTIONS: true,
    WHATSAPP_CHANNEL: true,
    ADVANCED_ANALYTICS: true,
    NEW_SURVEY_BUILDER: true,
  },
  staging: {
    AI_PREDICTIONS: true,
    WHATSAPP_CHANNEL: true,
    ADVANCED_ANALYTICS: true,
    NEW_SURVEY_BUILDER: false,
  },
  production: {
    AI_PREDICTIONS: false,
    WHATSAPP_CHANNEL: true,
    ADVANCED_ANALYTICS: true,
    NEW_SURVEY_BUILDER: false,
  },
};

// Flags por tenant (override)
const TENANT_FEATURE_FLAGS = {
  'tenant-abc': {
    AI_PREDICTIONS: true, // Beta tester
  },
};
```

---

10. PLAN DE DESARROLLO

10.1 Fases de Desarrollo

Fase 1: Fundamentos y Core (Semanas 1-8)

Objetivo: Establecer arquitectura base, autenticación y módulo core de Ley Karin

Entregables:
- [ ] Setup de proyecto (monorepo, CI/CD, linters)
- [ ] Arquitectura multi-tenant implementada
- [ ] Sistema de autenticación (JWT, MFA, SSO)
- [ ] Sistema de autorización (RBAC)
- [ ] Canal de denuncias (web, básico)
- [ ] Workflow de investigaciones básico
- [ ] Gestión de protocolos
- [ ] Dashboard básico
- [ ] Audit trail

Criterios de Aceptación:
- Usuario puede crear cuenta y autenticarse
- Usuario puede crear denuncia anónima o identificada
- Compliance officer puede gestionar investigaciones
- Administrador puede gestionar protocolos
- Todas las acciones quedan registradas en audit log

Fase 2: Módulos Principales (Semanas 9-16)

Objetivo: Implementar encuestas, analytics y módulos EDI/Clima

Entregables:
- [ ] Constructor de encuestas completo
- [ ] Sistema de distribución (email, link)
- [ ] Encuesta SUSESO/ISTAS21 integrada
- [ ] Módulo de KPIs configurables
- [ ] Dashboard analytics
- [ ] Módulo EDI básico
- [ ] Módulo Clima básico
- [ ] API REST pública
- [ ] Documentación API (Swagger)

Criterios de Aceptación:
- Usuario puede crear encuestas con lógica condicional
- Encuesta de riesgos psicosociales funcional
- Dashboard muestra KPIs en tiempo real
- API documentada y funcional

Fase 3: Integraciones y Features Avanzados (Semanas 17-24)

Objetivo: Multicanal, integraciones y features avanzados

Entregables:
- [ ] Canal WhatsApp para encuestas
- [ ] Modo offline (PWA)
- [ ] Integración SSO (SAML, OAuth)
- [ ] Webhooks
- [ ] Reportería regulatoria (SUSESO, ISL, DT)
- [ ] Módulo de capacitaciones básico
- [ ] Firma electrónica simple
- [ ] Benchmarking sectorial
- [ ] App móvil (React Native) - MVP

Criterios de Aceptación:
- Encuestas funcionan vía WhatsApp
- PWA funciona sin conexión
- SSO integrado con Azure AD / Google
- Reportes regulatorios generados automáticamente

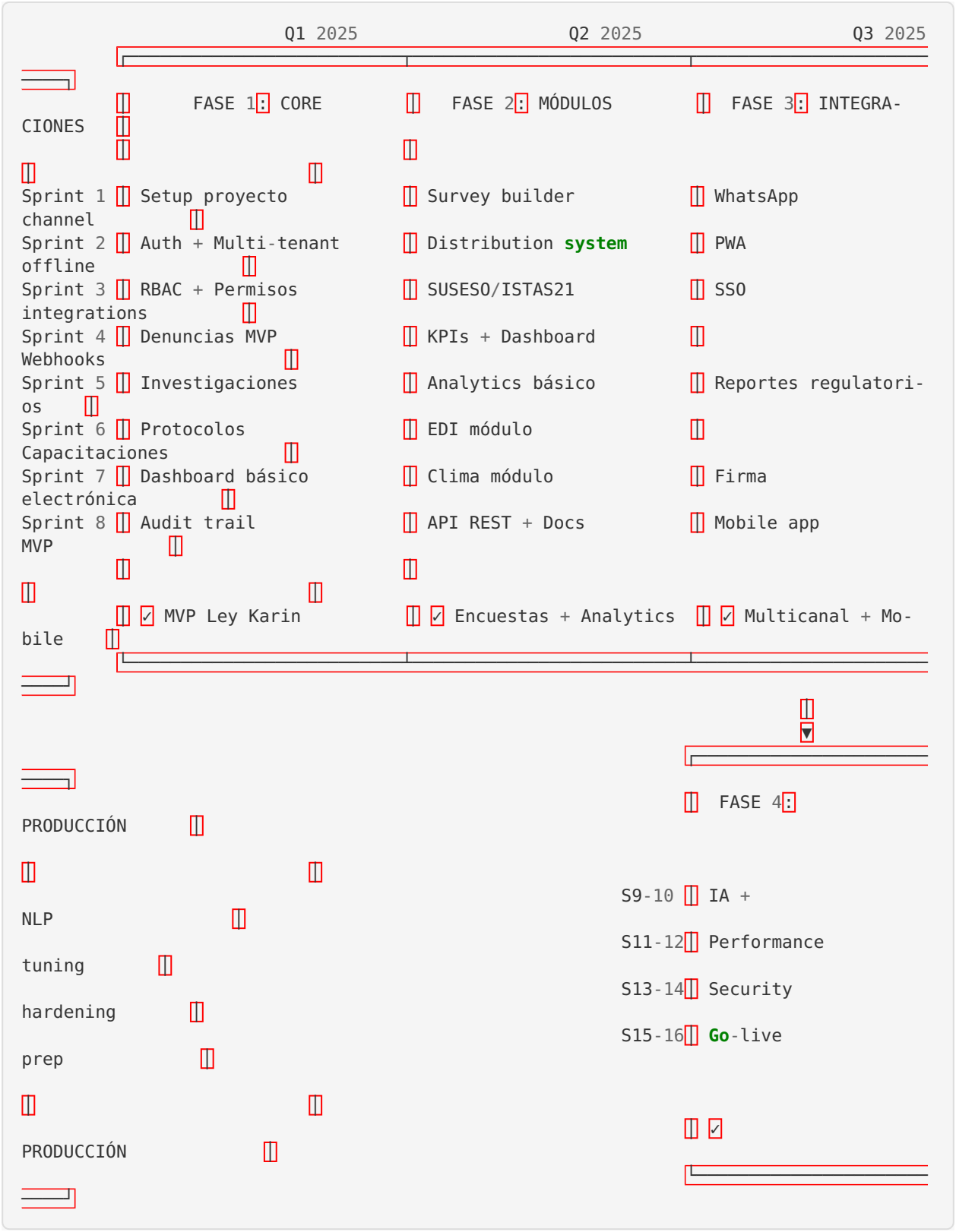
Fase 4: Optimización y Producción (Semanas 25-32)

Objetivo: Preparar para producción con IA y optimizaciones

Entregables:
- [ ] Analytics con IA (predicciones básicas)
- [ ] Análisis de sentimiento NLP
- [ ] Optimización de performance
- [ ] Tests de carga y seguridad
- [ ] Documentación completa
- [ ] Onboarding y demos
- [ ] Hardening de seguridad
- [ ] Disaster recovery plan

Criterios de Aceptación:
- Sistema soporta 10,000 usuarios concurrentes
- Tiempo de respuesta < 200ms p95
- Tests de seguridad pasados (OWASP)
- Documentación completa para usuarios y desarrolladores

10.2 Roadmap Visual



11. CONSIDERACIONES PARA AUDITORÍAS DE CÓDIGO

11.1 Checklist de Auditoría Backend

Seguridad

Item	Descripción	Severidad
[ ]	No hay credenciales hard-codeadas	Crítica
[ ]	Todos los endpoints sensibles requieren autenticación	Crítica
[ ]	RBAC implementado correcta-mente	Crítica
[ ]	Validación de inputs en todos los endpoints	Alta
[ ]	Sanitización de outputs (pre-vención XSS)	Alta
[ ]	Prepared statements / ORM sin raw queries	Alta
[ ]	Rate limiting implementado	Media
[ ]	CORS configurado restrictiva-mente	Media
[ ]	Headers de seguridad config-urados	Media
[ ]	Logging sin datos sensibles	Media

Arquitectura

Item	Descripción	Severidad
[ ]	Separación clara de capas (Clean Architecture)	Alta
[ ]	No hay dependencias circulares	Alta
[ ]	Principios SOLID respetados	Media
[ ]	DTOs para entrada/salida de API	Media
[ ]	Repositorios con interfaces	Media
[ ]	Manejo de errores consistente	Media
[ ]	Logging estructurado	Media

Código

Item	Descripción	Severidad
[ ]	Type hints en todas las funciones	Media
[ ]	No hay any explícito sin justificación	Media
[ ]	Funciones con responsabilidad única	Media
[ ]	No hay código duplicado (DRY)	Baja
[ ]	Nomenclatura consistente	Baja
[ ]	No hay magic numbers	Baja
[ ]	Comentarios donde es necesario	Baja

Testing

Item	Descripción	Severidad
[ ]	Coverage > 80%	Alta
[ ]	Tests unitarios para lógica de negocio	Alta
[ ]	Tests de integración para APIs	Media
[ ]	Tests no tienen dependencias externas	Media
[ ]	Tests son determinísticos	Media

11.2 Checklist de Auditoría Frontend

Se

guridad

Item	Descripción	Severidad
[ ]	No hay tokens/keys en código cliente	Crítica
[ ]	Validación de permisos antes de renderizar	Alta
[ ]	Sanitización de contenido dinámico	Alta
[ ]	HTTPS en todas las peticiones	Alta
[ ]	No hay localStorage de datos sensibles	Media

Accesibilidad (WCAG 2.1)

Item	Descripción	Severidad
[ ]	Contraste de colores ≥ 4.5:1	Alta
[ ]	Navegación por teclado funcional	Alta
[ ]	Etiquetas ARIA apropiadas	Alta
[ ]	Alt text en imágenes	Media
[ ]	Focus visible en elementos interactivos	Media
[ ]	Formularios con labels asociados	Media
[ ]	Errores anunciados con aria-live	Media

Performance

Item	Descripción	Severidad
[ ]	Bundle size optimizado (< 500KB gzip)	Alta
[ ]	Lazy loading de rutas	Media
[ ]	Imágenes optimizadas	Media
[ ]	No hay memory leaks (cleanup useEffect)	Media
[ ]	Memoización donde corresponde	Baja
[ ]	No hay re-renders innecesarios	Baja

Código

Item	Descripción	Severidad
[ ]	Componentes con responsabilidad única	Media
[ ]	Props tipadas correctamente	Media
[ ]	Custom hooks extraídos	Baja
[ ]	Consistencia en estructura	Baja
[ ]	No hay prop drilling excesivo	Baja

---

12. GLOSARIO Y REFERENCIAS

12.1 Términos Técnicos

Término	Definición
Aggregate	Patrón DDD que agrupa entidades relacionadas bajo una raíz
Bounded Context	Límite conceptual donde un modelo de dominio es válido
CQRS	Command Query Responsibility Segregation - Separación de lectura y escritura
DTO	Data Transfer Object - Objeto para transferir datos entre capas
JWT	JSON Web Token - Estándar para tokens de autenticación
Multi-tenant	Arquitectura que soporta múltiples clientes en una instancia
PWA	Progressive Web App - Aplicación web con capacidades nativas
RBAC	Role-Based Access Control - Control de acceso basado en roles
RLS	Row-Level Security - Seguridad a nivel de fila en bases de datos
SSO	Single Sign-On - Autenticación única para múltiples sistemas
Value Object	Objeto inmutable definido por sus atributos, sin identidad

12.2 Referencias Ley Karin (Ley 21.643)

Concepto	Descripción Legal
Acoso Laboral	Toda conducta que constituya agresión u hostigamiento, ejercida por el empleador o por uno o más trabajadores, en contra de otro u otros, por cualquier medio, y que tenga como resultado para el o los afectados menoscabo, maltrato o humillación, o bien que amenace o perjudique su situación laboral o sus oportunidades en el empleo
Acoso Sexual	El que una persona realice, en forma indebida, por cualquier medio, requerimientos de carácter sexual, no consentidos por quien los recibe y que amenacen o perjudiquen su situación laboral o sus oportunidades en el empleo
Violencia en el Trabajo	Conductas que afecten a los trabajadores con ocasión de la prestación de servicios, ejercidas por terceros ajenos a la relación laboral (clientes, proveedores, usuarios)
Perspectiva de Género	Enfoque que considera las diferencias de género y sus implicancias en el análisis de situaciones de acoso y violencia
Protocolo de Prevención	Documento que establece las medidas para prevenir el acoso y violencia, y los procedimientos para su denuncia e investigación
SUSESO	Superintendencia de Seguridad Social - Organismo fiscalizador
ISL	Instituto de Seguridad Laboral - Organismo de seguridad social
Dirección del Trabajo	Organismo encargado de fiscalizar el cumplimiento laboral

12.3 Normativas y Estándares Aplicables

Normativa	Ámbito	Aplicabilidad
Ley 21.643 (Ley Karin)	Acoso y violencia laboral	Chile - Obligatorio
Ley 21.015	Inclusión laboral personas con discapacidad	Chile - Obligatorio
NCG 461 CMF	Reportes de equidad de género	Chile - Empresas reguladas
SUSESO/ISTAS21	Riesgos psicosociales	Chile - Obligatorio
Convenio 169 OIT	Pueblos indígenas	Chile - Obligatorio
GDPR	Protección de datos	UE - Si aplica
ISO 27001	Seguridad de información	Recomendado
OWASP Top 10	Seguridad aplicaciones web	Recomendado
WCAG 2.1	Accesibilidad web	Recomendado

12.4 Documentación Adicional Recomendada

Documentación Técnica

- NestJS Documentation (https://docs.nestjs.com/)
- React Documentation (https://react.dev/)
- TypeORM Documentation (https://typeorm.io/)
- PostgreSQL Documentation (https://www.postgresql.org/docs/)
- Redis Documentation (https://redis.io/docs/)

Documentación Regulatoria

- Ley 21.643 - Biblioteca del Congreso Nacional (https://www.bcn.cl/leychile/navegar?idNorma=1200246)
- Guía SUSESO Riesgos Psicosociales (https://www.suseso.cl/)
- Dirección del Trabajo - Normativa (https://www.dt.gob.cl/)

Estándares de Desarrollo

- Conventional Commits (https://www.conventionalcommits.org/)
- Semantic Versioning (https://semver.org/)
- 12-Factor App (https://12factor.net/)

ANEXO A: Configuración Docker

```
# Dockerfile
FROM node:20-alpine AS base

# Dependencias de build
FROM base AS deps
WORKDIR /app
COPY package*.json ./
RUN npm ci --only=production

# Build
FROM base AS builder
WORKDIR /app
COPY package*.json ./
RUN npm ci
COPY . .
RUN npm run build

# Producción
FROM base AS runner
WORKDIR /app

ENV NODE_ENV=production

RUN addgroup --system --gid 1001 nodejs
RUN adduser --system --uid 1001 nestjs

COPY --from=deps /app/node_modules ./node_modules
COPY --from=builder /app/dist ./dist

USER nestjs

EXPOSE 3000

CMD ["node", "dist/main.js"]
```

```yaml
# docker-compose.yml
version: '3.8'

services:
  api:
    build: .
    ports:
      - "3000:3000"
    environment:
      - NODE_ENV=production
      - DATABASE_URL=postgres://user:pass@db:5432/compliance
      - REDIS_URL=redis://redis:6379
    depends_on:
      - db
      - redis
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:3000/health"]
      interval: 30s
      timeout: 10s
      retries: 3

  db:
    image: postgres:16-alpine
    volumes:
      - postgres_data:/var/lib/postgresql/data
    environment:
      - POSTGRES_USER=user
      - POSTGRES_PASSWORD=pass
      - POSTGRES_DB=compliance
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U user -d compliance"]
      interval: 10s
      timeout: 5s
      retries: 5

  redis:
    image: redis:7-alpine
    volumes:
      - redis_data:/data
    healthcheck:
      test: ["CMD", "redis-cli", "ping"]
      interval: 10s
      timeout: 5s
      retries: 5

volumes:
  postgres_data:
  redis_data:
```

ANEXO B: Variables de Entorno

```
# .env.example

# ============================================
# APPLICATION
# ============================================
NODE_ENV=development
PORT=3000
API_VERSION=v1
APP_NAME=ComplianceHub
APP_URL=http://localhost:3000

# ============================================
# DATABASE
# ============================================
DATABASE_URL=postgres://user:password@localhost:5432/compliance
DATABASE_SSL=false
DATABASE_LOGGING=true
DATABASE_SYNCHRONIZE=false

# ============================================
# REDIS
# ============================================
REDIS_URL=redis://localhost:6379
REDIS_PREFIX=compliance:

# ============================================
# JWT
# ============================================
JWT_SECRET=your-super-secret-jwt-key-min-32-chars
JWT_ACCESS_EXPIRATION=15m
JWT_REFRESH_EXPIRATION=7d

# ============================================
# ENCRYPTION
# ============================================
ENCRYPTION_KEY=your-32-character-encryption-key

# ============================================
# EMAIL
# ============================================
SMTP_HOST=smtp.example.com
SMTP_PORT=587
SMTP_USER=user@example.com
SMTP_PASS=password
EMAIL_FROM=noreply@compliancehub.com

# ============================================
# STORAGE (S3)
# ============================================
S3_BUCKET=compliance-files
S3_REGION=us-east-1
S3_ACCESS_KEY=your-access-key
S3_SECRET_KEY=your-secret-key

# ============================================
# EXTERNAL SERVICES
# ============================================
WHATSAPP_API_URL=https://api.whatsapp.com
WHATSAPP_API_TOKEN=your-token

# ============================================
# MONITORING
```

```
# ============================================
SENTRY_DSN=https://your-sentry-dsn
LOG_LEVEL=debug

# ============================================
# FEATURE FLAGS
# ============================================
FF_AI_PREDICTIONS=false
FF_WHATSAPP_CHANNEL=true
```

FIN DEL DOCUMENTO

Documento generado como especificación técnica para el desarrollo de ComplianceHub Pro.
Versión 1.0 - Febrero 2026
Confidencial - Uso interno para equipo de desarrollo