

ESTRUCTURA DEL PROYECTO

...

iso_platform/

- └─ .env
- └─ .env.example
- └─ .gitignore
- └─ composer.json
- └─ README.md
- └─ config/
 - └─ app.php
 - └─ database.php
 - └─ security.php
 - └─ session.php
- └─ src/
 - └─ Core/
 - └─ Database.php
 - └─ Router.php
 - └─ Request.php
 - └─ Response.php
 - └─ Session.php
 - └─ Validator.php
 - └─ Models/
 - └─ Base/
 - └─ Model.php
 - └─ SoftDelete.php
 - └─ Empresa.php
 - └─ Usuario.php
 - └─ Control.php
 - └─ SOA.php
 - └─ Gap.php
 - └─ Accion.php
 - └─ Evidencia.php
 - └─ Requerimiento.php
 - └─ Controllers/
 - └─ Base/
 - └─ Controller.php

- | | └─ AuthController.php
- | | └─ DashboardController.php
- | | └─ ControlController.php
- | | └─ GapController.php
- | | └─ EvidenciaController.php
- | | └─ RequerimientoController.php
- | └─ Middleware/
 - | | └─ AuthMiddleware.php
 - | | └─ RoleMiddleware.php
 - | | └─ RateLimitMiddleware.php
 - | | └─ CsrfMiddleware.php
 - | | └─ TenantMiddleware.php
- | └─ Services/
 - | | └─ AuthService.php
 - | | └─ FileService.php
 - | | └─ CacheService.php
 - | | └─ LogService.php
 - | | └─ MetricsService.php
- | └─ Repositories/
 - | | └─ Base/
 - | | | └─ Repository.php
 - | | └─ UsuarioRepository.php
 - | | └─ ControlRepository.php
 - | | └─ EvidenciaRepository.php
- | └─ Views/
 - | | └─ layouts/
 - | | | └─ app.php
 - | | | └─ auth.php
 - | | | └─ components/
 - | | └─ auth/
 - | | └─ dashboard/
 - | | └─ controles/
 - | | └─ gap/
 - | | └─ evidencias/
 - | | └─ requerimientos/
- └─ public/
 - | └─ index.php
 - | └─ .htaccess
 - | └─ assets/
 - | | └─ css/

```

| | └─ js/
| | └─ images/
| └─ uploads/
└─ storage/
| └─ logs/
| └─ cache/
| └─ sessions/
└─ database/
| └─ migrations/
| └─ seeds/
| └─ schema.sql
└─ tests/
    └─ Unit/
    └─ Integration/
...

```

FASE 1: ARQUITECTURA BASE

1.1 Configuración Inicial

- Instalar Composer con dependencias mínimas: PDO, dotenv
- Configurar autoload PSR-4
- Crear .env con variables: DB, APP_KEY, UPLOAD_PATH, SESSION
- Configurar .htaccess para rewrite y seguridad

1.2 Core Framework

- Database: Singleton PDO con prepared statements, connection pooling
- Router: Sistema de rutas con parámetros dinámicos y middlewares
- Request: Captura y sanitización automática de inputs
- Response: Manejo de headers, JSON, redirects
- Session: Gestión segura con regeneración y fingerprinting
- Validator: Reglas reutilizables para validación de datos

1.3 Seguridad Base

- CSRF: Generación y validación de tokens por sesión
- XSS: Sanitización de inputs/outputs en capa de presentación
- SQL Injection: Prepared statements obligatorios en todos los queries
- Rate Limiting: Control de intentos por IP + user_id

- Password: Hash con Argon2id o Bcrypt cost 12

FASE 2: BASE DE DATOS NORMALIZADA

2.1 Diseño 3FN

- **empresas**: id, nombre, ruc (UNIQUE), contacto, metadata
- **usuarios**: id, empresa_id (FK), email (UNIQUE en empresa), password_hash, rol, estado
- **controles**: id, codigo (UNIQUE), nombre, descripcion, dominio_id (FK)
- **controles_dominio**: id, codigo (UNIQUE), nombre
- **soa_entries**: id, empresa_id (FK), control_id (FK), aplicable, estado, justificacion, UNIQUE(empresa_id, control_id)
- **gap_items**: id, soa_id (FK), brecha, prioridad, avance (computed), estado_gap
- **acciones**: id, gap_id (FK), descripcion, estado, estado_accion, fecha_compromiso
- **evidencias**: id, empresa_id (FK), control_id (FK), archivo, estado_validacion
- **requerimientos_base**: id, numero (UNIQUE), identificador, descripcion
- **empresa_requerimientos**: id, empresa_id (FK), requerimiento_id (FK), estado, UNIQUE(empresa_id, requerimiento_id)
- **requerimientos_controles**: id, requerimiento_base_id (FK), control_id (FK), UNIQUE(requerimiento_base_id, control_id)
- **audit_logs**: id, empresa_id, usuario_id, tabla, accion, datos_previos, datos_nuevos, ip, timestamp

2.2 Índices Críticos

- UNIQUE: ruc, email+empresa_id, control+empresa_id
- INDEX: empresa_id en todas las tablas multi-tenant
- INDEX: estado, estado_gap, estado_accion para filtros
- INDEX: fecha_compromiso, fecha_evaluacion para reportes
- FULLTEXT: descripcion en gap_items para búsquedas

2.3 Constraints

- CASCADE DELETE: empresa → usuarios, soa_entries, evidencias
- CASCADE UPDATE: control_id en todas las referencias
- CHECK: avance BETWEEN 0 AND 100
- CHECK: estado IN (valores permitidos)

FASE 3: MULTITENANCY

3.1 Tenant Isolation

- Middleware que inyecta empresa_id en todos los queries
- Session almacena empresa_id del usuario autenticado
- Queries base con WHERE empresa_id = :tenant obligatorio
- Repository pattern con scope automático por tenant

3.2 Tenant Context

- Clase TenantContext singleton que mantiene empresa_id activa
- Métodos setTenant(), getTenant(), clearTenant()
- Todos los modelos heredan scope automático
- Validación IDOR en cada operación CRUD

3.3 Datos Compartidos

- Tablas maestras SIN empresa_id: controles, controles_dominio, requerimientos_base, requerimientos_controles
- Tablas tenant-specific CON empresa_id: usuarios, soa_entries, gap_items, evidencias

FASE 4: CAPA DE MODELO (Repository Pattern)

4.1 Base Model

- Métodos CRUD genéricos: find(), findAll(), create(), update(), delete()
- Soft delete automático con deleted_at
- Timestamps automáticos: created_at, updated_at
- Query builder simple para condiciones WHERE

4.2 Repositories

- Lógica de negocio separada del modelo
- Métodos específicos: findByEmail(), findWithEvidencias()
- Caché de queries frecuentes con TTL
- Transacciones para operaciones complejas

4.3 Relaciones

- Eager loading manual: WITH joins en queries
- Lazy loading con métodos get*(): getEvidencias(), getGaps()
- Evitar N+1 queries con batch loading

FASE 5: AUTENTICACIÓN Y AUTORIZACIÓN

5.1 Registro Multi-Tenant

- Wizard 3 pasos: Empresa → Admin → Confirmación
- Crear empresa + usuario admin + trigger SOA en transacción
- Email debe ser único dentro de empresa
- Validación RUC contra base externa (opcional)

5.2 Login

- Identificador: email + empresa_id O username único global
- Rate limiting: 5 intentos por 15 minutos
- Sesión: almacenar user_id, empresa_id, rol, permisos
- Fingerprinting: validar IP + User-Agent en cada request

5.3 Roles y Permisos

- Roles fijos: super_admin, admin_empresa, auditor, consultor
- Permisos por rol en array de configuración
- Middleware RoleMiddleware valida permisos antes de controller
- super_admin puede cambiar de tenant para soporte

FASE 6: MÓDULOS FUNCIONALES

6.1 Dashboard

- Métricas en tiempo real con caché de 5 minutos
- Cards: Cumplimiento, Gaps críticos, Evidencias pendientes, Acciones vencidas
- Gráficos: Avance por dominio, Timeline de implementación
- Queries optimizados con índices y agregaciones en BD

6.2 Controles

- Lista con filtros: dominio, estado, aplicabilidad
- Detalle con formulario evaluación inline
- Actualización AJAX sin reload (fetch API)
- Validación IDOR: control debe pertenecer a empresa

35 KB

6.3 GAP Analysis

- CRUD completo con transacciones
- Avance calculado automáticamente con trigger o computed column
- Solo permitir GAPS en controles aplicables
- Soft delete en cascada: gap → acciones

6.4 Evidencias

- Upload con validación MIME real (finfo_file)
- Almacenamiento: /uploads/{empresa_id}/{año}/{mes}/{hash}.ext
- Escaneo antivirus básico: buscar patrones maliciosos
- Metadata en BD: tamaño, tipo, hash SHA256
- Descarga con validación IDOR y rate limiting

6.5 Requerimientos

- Completitud automática con triggers o cron
- Validación: TODOS los controles implementados + TODAS las evidencias aprobadas
- Vista de progreso con barra y checklist
- Exportar reporte en PDF (TCPDF)

FASE 7: SERVICIOS TRANSVERSALES

7.1 File Service

- Upload con validación: tamaño, tipo, contenido
- Almacenamiento organizado por tenant
- Generación de nombres únicos con hash
- Cleanup de archivos huérfanos con cron
- Prevención path traversal

7.2 Cache Service

- Implementación simple con archivos en /storage/cache
- Métodos: get(), set(), delete(), flush()
- TTL configurable por tipo de dato
- Caché de queries frecuentes y conteos

7.3 Log Service

- Logs separados: app.log, error.log, security.log, audit.log
- Rotación diaria con cleanup de +30 días
- Contexto: user_id, empresa_id, IP, timestamp
- Niveles: DEBUG, INFO, WARNING, ERROR, CRITICAL

7.4 Metrics Service

- Contadores: requests, queries, uploads, logins
- Promedios: response time, query time
- Alertas: errores 500, rate limit hits, logins fallidos
- Dashboard de métricas para super_admin

FASE 8: OPTIMIZACIÓN DE RENDIMIENTO

8.1 Database

- Connection pooling con PDO persistent connections
- Índices compuestos para queries complejos
- EXPLAIN ANALYZE en queries lentos
- Particionamiento de audit_logs por fecha
- Read replicas para reportes (futuro)

8.2 Caché

- Caché de sesiones en archivos con garbage collection
- Caché de queries agregadas (conteos, estadísticas)
- Caché de vistas parciales (cards del dashboard)
- Invalidación inteligente al modificar datos

8.3 Assets

- CSS minificado y concatenado
- JS minificado y defer/async
- Imágenes optimizadas y lazy loading
- CDN para Tailwind CSS
- Versionado de assets con hash

8.4 Queries

- Paginación obligatoria en listados (+50 registros)
- Lazy loading de relaciones
- Batch queries para actualizaciones múltiples
- Evitar SELECT * siempre especificar columnas

FASE 9: SEGURIDAD AVANZADA

9.1 Protección IDOR

- Validar empresa_id en TODOS los queries
- Middleware TenantMiddleware automático
- Nunca confiar en IDs de URL sin validación
- Logging de intentos IDOR sospechosos

9.2 Rate Limiting Granular

- Login: 5/15min por IP + email
- Uploads: 10/hora por usuario
- API calls: 100/minuto por usuario
- Forms: 20/5min por usuario
- Storage en sesión + BD para persistencia

9.3 Auditoría

- Log TODOS los cambios en datos críticos
- Audit trail con before/after values
- IP, user_agent, timestamp en cada log
- Retención de 2 años para compliance
- Exportación de logs para análisis forense

9.4 Hardening

- Headers: CSP, HSTS, X-Frame-Options, X-Content-Type-Options
- Cookie flags: HttpOnly, Secure, SameSite=Strict
- Deshabilitar directory listing
- Ocultar versión PHP y Apache
- Permisos archivos: 644 para código, 600 para .env

FASE 10: UI/UX PROFESIONAL

10.1 Design System

- Variables CSS para colores, espaciados, tipografía
- Componentes reutilizables: botones, cards, modals, forms
- Estados: hover, active, disabled, loading
- Feedback visual inmediato en todas las acciones
- Skeleton loaders para cargas asíncrona

10.2 Responsive

- Mobile-first con Tailwind breakpoints
- Sidebar colapsable en móviles
- Tablas con scroll horizontal
- Modals full-screen en móviles
- Touch-friendly: botones +44px

10.3 Accesibilidad

- Contraste mínimo WCAG AA
- Labels en todos los inputs
- Focus visible en teclado
- ARIA labels en iconos
- Navegación completa por teclado

10.4 Feedback

- Toasts para notificaciones temporales
- Confirmaciones en acciones destructivas
- Validación en tiempo real en formularios
- Progress bars en procesos largos
- Estados de carga con spinners

FASE 11: TESTING Y QA

11.1 Unit Tests

- PHPUnit para lógica de negocio
- Test de modelos: CRUD, validaciones
- Test de servicios: auth, file, cache
- Cobertura mínima: 70%

11.2 Integration Tests

- Test de flujos completos: registro → login → crear gap
- Test de transacciones y rollbacks
- Test de middleware chain
- Test de rate limiting

11.3 Security Tests

- SQL injection attempts

- XSS payloads
- CSRF bypass attempts
- IDOR exploitation attempts
- File upload exploits

11.4 Load Tests

- Apache Bench: 100 usuarios concurrentes
- Identificar cuellos de botella
- Optimizar queries lentos
- Ajustar límites de rate limiting

FASE 12: DEPLOYMENT Y CI/CD

12.1 Environment Setup

- Crear .env.production con valores seguros
- Configurar permisos: www-data para uploads y storage
- Configurar cron para cleanup y cache warming
- SSL con Let's Encrypt
- Firewall: solo puertos 80, 443

12.2 CI Pipeline

- Git hooks: pre-commit para linting
- GitHub Actions: test suite en push
- Static analysis con PHPStan
- Security scan con dependencies audit
- Build automático de assets

12.3 Deployment

- Script de deployment con backup previo
- Migrations automáticas en deploy
- Cache clearing post-deploy
- Rollback automático si falla health check
- Zero-downtime con blue-green deploy

12.4 Monitoring

- Health checks: /health endpoint
- Logs centralizados con rotación

- Alertas: errores 500, queries lentos, disco lleno
- Métricas: requests/min, avg response time
- Backup diario automatizado de BD

ORDEN DE IMPLEMENTACIÓN

1. **Semana 1-2**: Fase 1 (Arquitectura) + Fase 2 (BD)
2. **Semana 3**: Fase 3 (Multi-tenancy) + Fase 4 (Models)
3. **Semana 4**: Fase 5 (Auth) + Fase 6.1 (Dashboard)
4. **Semana 5**: Fase 6.2-6.3 (Controles + GAP)
5. **Semana 6**: Fase 6.4-6.5 (Evidencias + Requerimientos)
6. **Semana 7**: Fase 7 (Servicios) + Fase 8 (Optimización)
7. **Semana 8**: Fase 9 (Seguridad) + Fase 10 (UI/UX)
8. **Semana 9**: Fase 11 (Testing) + Fase 12 (CI/CD)
9. **Semana 10**: QA final + deployment producción

ENTREGABLES POR FASE

- Código funcional y testeado
- Documentación de API interna
- Diagramas de flujo actualizados
- Tests unitarios y de integración
- Logs de cambios en CHANGELOG.md

ANÁLISIS COMPLETO: ISO 27001 COMPLIANCE PLATFORM

1. PROPÓSITO Y CONTEXTO DE NEGOCIO

¿Qué es la plataforma?

Una **herramienta de gestión de cumplimiento de seguridad de la información** basada en la norma **ISO 27001:2022**. Permite a las organizaciones:

- **Evaluar** su nivel de cumplimiento contra los 93 controles del Anexo A
- **Documentar** la aplicabilidad de cada control (SOA - Statement of Applicability)
- **Identificar brechas** (GAPs) entre el estado actual y el deseado
- **Planificar acciones** correctivas para cerrar esas brechas

...eliminar acciones correctivas para cerrar esas brechas

- **Gestionar evidencias** que demuestren la implementación de controles
- **Cumplir requerimientos** documentales obligatorios para certificación
- **Generar reportes** ejecutivos y técnicos para auditorías

¿Por qué existe?

La certificación ISO 27001 requiere:

1. Evaluar TODOS los 93 controles de seguridad
2. Documentar cuáles aplican y cuáles no (con justificación)
3. Implementar los controles aplicables
4. Demostrar con evidencias la implementación
5. Mantener 7 documentos maestros obligatorios
6. Pasar auditorías internas y externas

Sin una plataforma, esto se hace con **hojas de cálculo, documentos dispersos y control manual**, lo que genera:

- Pérdida de información
- Inconsistencias entre versiones
- Dificultad para auditar cambios
- Imposibilidad de medir progreso en tiempo real
- Riesgo de incumplimiento

2. LÓGICA DE NEGOCIO CORE

Conceptos Fundamentales

A. CONTROLES ISO 27001

- **93 controles obligatorios** organizados en 4 dominios:
 - **5. Organizacionales** (37 controles): políticas, roles, procesos
 - **6. Personas** (8 controles): selección, capacitación, concienciación
 - **7. Físicos** (14 controles): seguridad de instalaciones, equipos
 - **8. Tecnológicos** (34 controles): redes, cifrado, accesos, incidentes
- **Cada control tiene**:
 - Código único (ej: 5.1, 8.20)
 - Nombre descriptivo
 - Descripción de qué requiere

B. APLICABILIDAD (SOA)

- La norma NO obliga a implementar los 93 controles
- ****Solo los que sean relevantes para los riesgos de la organización****
- Cada empresa debe evaluar control por control:
 - ****Aplicable****: Sí es relevante para nuestros riesgos → debe implementarse
 - ****No aplicable****: No es relevante → justificar por qué no
- ****Estados de implementación**** (solo para controles aplicables):
 - ****Implementado****: Control completamente funcional
 - ****Parcial****: Control en proceso de implementación
 - ****No implementado****: Control aplicable pero aún no se ha trabajado en él

C. ANÁLISIS DE BRECHAS (GAP)

- ****GAP = Brecha entre estado actual y deseado****
- Solo se crean GAPs en controles ****aplicables no implementados o parciales****
- Cada GAP tiene:
 - ****Descripción de la brecha****: qué falta
 - ****Objetivo****: qué se quiere lograr
 - ****Prioridad****: alta/media/baja según riesgo
 - ****Plan de acción****: acciones específicas para cerrar el GAP
 - ****Avance automático****: calculado según acciones completadas

D. ACCIONES CORRECTIVAS

- ****Tareas específicas**** para cerrar un GAP
- Cada acción tiene:
 - Descripción clara
 - Responsable asignado
 - Fecha compromiso
 - Estado: pendiente → en progreso → completada
- ****Cálculo automático de avance del GAP****:
 - Si un GAP tiene 5 acciones y 2 están completadas → 40% de avance
 - Al llegar al 100% → GAP se cierra automáticamente

E. EVIDENCIAS

- ****Documentos que demuestran**** la implementación de controles
- Solo controles ****aplicables**** pueden tener evidencias
- Tipos comunes:
 - Políticas firmadas

- Registros de capacitacion
 - Capturas de pantalla de configuraciones
 - Certificados de terceros
 - Actas de comité de seguridad
 - Informes de auditorías internas
-
- ****Flujo de validación****:
 - Usuario sube evidencia → estado: ****pendiente****
 - Auditor/líder revisa → aprueba o rechaza
 - Solo evidencias ****aprobadas**** cuentan para completitud

F. REQUERIMIENTOS BASE

- ****7 documentos maestros obligatorios**** según ISO 27001:
 1. Manual de políticas de seguridad
 2. Inventario de activos de información
 3. Plan anual de capacitaciones
 4. Estrategia de concientización
 5. Evidencia de cumplimiento de plan y estrategia
 6. Manual de gestión de incidentes
 7. Evidencia de monitoreo continuo
- ****Completitud automática****:
 - Cada requerimiento está vinculado a múltiples controles
 - Se marca como ****completado**** automáticamente cuando:
 - TODOS sus controles asociados están ****implementados****
 - TODOS tienen al menos UNA evidencia ****aprobada****
 - NO hay evidencias pendientes o rechazadas

3. FLUJO DE USO COMPLETO

FASE 1: ONBOARDING (Registro)

```

1. Super Admin o Empresa nueva accede a /registro
2. Completa wizard de 3 pasos:

#### PASO 1: Datos de la empresa

- Nombre comercial

- RUC (validado como único)
- Sector (salud, finanzas, manufactura, etc.)
- Datos de contacto

PASO 2: Usuario administrador

- Nombre completo
- Email corporativo (será el username)
- Contraseña fuerte (validada con requisitos)
- Confirmar contraseña

PASO 3: Confirmación y resumen

- Mostrar datos ingresados
- Términos y condiciones
- Botón "Crear cuenta"

3. Al confirmar, el sistema ejecuta TRANSACCIÓN:

- Crea empresa en BD
- Crea usuario admin vinculado a esa empresa
- TRIGGER automático crea:
  - \* 93 registros en soa\_entries (estado inicial: no\_implementado)
  - \* 7 registros en empresa\_requerimientos (estado: pendiente)
- Genera sesión con empresa\_id y usuario\_id
- Redirige a /dashboard con mensaje de bienvenida

4. Sistema envía email de confirmación (opcional)

...

### FASE 2: EVALUACIÓN INICIAL (SOA)

...

Usuario Admin/Auditor ingresa a:

1. /controles (Lista de 93 controles)

- Ve tabla con columnas:
  - \* Código (5.1, 8.20, etc.)
  - \* Nombre del control
  - \* Dominio (Organizacional, Físico, etc.)
  - \* Aplicabilidad (por defecto: Sí)
  - \* Estado (por defecto: No implementado)



- \* Evidencias (contador)

- Puede filtrar por:

- \* Dominio
- \* Estado de implementación
- \* Aplicabilidad

2. Click en control específico → /controles/{id}

- Ve detalle completo del control
- Formulario de evaluación con opciones:

A. ¿Es aplicable a su organización?

- ☐ Sí, es aplicable
- ☐ No es aplicable → muestra campo "Justificación"

B. Si es aplicable, ¿cuál es su estado?

- ☐ Implementado (control ya está funcionando)
- ☐ Parcialmente implementado (en proceso)
- ☐ No implementado (aún no se ha trabajado)

C. Botón "Guardar evaluación"

- Backend valida datos
- Actualiza soa\_entries
- Si marca "No aplicable" → requiere justificación obligatoria
- Si marca "Implementado" → sistema verifica si hay evidencias
- Muestra mensaje de éxito

3. Repite proceso para los 93 controles

- Sistema guarda progreso automáticamente
- Dashboard muestra % de controles evaluados

...

### FASE 3: IDENTIFICACIÓN DE BRECHAS (GAP)

...

Tras evaluar todos los controles, el usuario:

1. Identifica controles "No implementados" o "Parciales" en /controles
2. Para cada uno, puede crear un GAP:

/gap/crear

- Selecciona control (solo muestra controles APLICABLES no implementados)
  - Completa formulario:
    - \* Descripción de la brecha (qué falta específicamente)
    - \* Objetivo de mejora (qué se quiere lograr)
    - \* Prioridad (alta/media/baja)
    - \* Responsable (persona a cargo)
    - \* Fecha estimada de cierre
  - Agrega acciones correctivas (mínimo 1):
    - \* Descripción de la tarea
    - \* Responsable específico
    - \* Fecha compromiso
- [Botón: Agregar otra acción]

- Botón "Crear GAP"
- Backend ejecuta TRANSACCIÓN:
- \* Crea registro en gap\_items (avance inicial: 0%)
  - \* Crea N registros en acciones (estado: pendiente)
  - \* Log de auditoría
  - \* Redirige a /gap/{id}

3. Vista de GAP creado: /gap/{id}

- Muestra información del GAP
- Lista de acciones con estados
- Botón "Marcar acción como completada" por cada acción
- Barra de progreso automática
- Cuando avance = 100% → GAP se cierra automáticamente

...

### FASE 4: GESTIÓN DE EVIDENCIAS

...

Usuario necesita demostrar implementación de controles:

1. /evidencias/subir

- Selecciona control (solo controles APLICABLES)
- Completa formulario:
  - \* Tipo de evidencia (Política, Registro, Certificado, etc.)

- \* Descripción clara (qué demuestra este archivo)
- \* Archivo (PDF, DOCX, XLSX, PNG, JPG)
  - Validación tamaño máximo: 10MB
  - Validación MIME real (no solo extensión)
  - Escaneo anti-malware básico

- Botón "Subir evidencia"

Backend procesa:

- \* Valida archivo (tipo, tamaño, contenido)
- \* Genera nombre único: SHA256(timestamp + random) + extensión
- \* Almacena en: /uploads/{empresa\_id}/{año}/{mes}/{archivo}
- \* Crea registro en evidencias (estado: pendiente)
- \* Notifica a auditor (opcional)

2. /evidencias (Repositorio)

- Lista todas las evidencias de la empresa
- Filtros: control, tipo, estado
- Para cada evidencia:
  - \* Icono del tipo de archivo
  - \* Descripción
  - \* Control asociado
  - \* Estado (badge: pendiente/aprobada/rechazada)
  - \* Botón "Descargar" (con validación IDOR)
  - \* Botón "Validar" (solo para auditores)

3. Auditor valida evidencia:

- Click en "Validar"
- Modal con opciones:
  - \* ☐ Aprobar
  - \* ☐ Rechazar
  - \* Campo: Comentarios
- Al aprobar:
  - \* Actualiza estado\_validacion = 'aprobada'
  - \* Sistema verifica si se completó algún requerimiento automáticamente
  - \* Notifica al usuario que subió la evidencia

...

...

...

Usuario ve progreso de documentos obligatorios:

1. /requerimientos

- Checklist de 7 requerimientos con estado visual:
  - [ ] 1. Manual de políticas → Pendiente (0/3 controles)
  - [~] 2. Inventario de activos → En proceso (5/8 controles)
  - [✓] 3. Plan de capacitaciones → Completado (automático)
- Para cada requerimiento:
  - \* Ver controles asociados (expandible)
  - \* Ver evidencias aprobadas
  - \* Estado actualizado automáticamente

2. Lógica de completitud automática:

- Sistema ejecuta verificación cada vez que:
  - \* Se marca un control como "Implementado"
  - \* Se aprueba una evidencia
- Proceso de verificación:
  - a. Obtiene todos los controles del requerimiento
  - b. Verifica que TODOS estén en estado "implementado"
  - c. Verifica que TODOS tengan evidencias aprobadas
  - d. Si se cumple todo → marca requerimiento como "completado"
  - e. Notifica al usuario del logro

...

### FASE 6: MONITOREO Y REPORTE

...

1. /dashboard

- Métricas en tiempo real:
  - \* % Cumplimiento general (controles implementados / aplicables)
  - \* GAPs críticos (alta prioridad con < 50% avance)
  - \* Evidencias pendientes de validación
  - \* Acciones vencidas (fecha compromiso pasada)
  - \* Requerimientos completados (X/7)
- Gráficos:
  - \* Avance por dominio (barras horizontales)

- \* Timeline de implementación (últimos 30 días)
- \* Distribución de GAPs por prioridad (pie chart)

2. /reportes

- Generar SOA en PDF
  - \* Tabla completa de 93 controles
  - \* Aplicabilidad y justificaciones
  - \* Estados de implementación
  - \* Firmado digitalmente
- Reporte ejecutivo
  - \* Resumen de cumplimiento
  - \* GAPs críticos
  - \* Próximos hitos
- Reporte de auditoría
  - \* Cambios en últimos 90 días
  - \* Evidencias agregadas
  - \* Controles modificados

...

---

## 4. REGLAS DE NEGOCIO CRÍTICAS

### R1: Aplicabilidad de Controles

...

SI control.aplicable = 0 (No aplicable):

- DEBE tener justificación escrita
- NO puede tener GAPs
- NO puede tener evidencias
- NO cuenta para % de cumplimiento

SI control.aplicable = 1 (Aplicable):

- DEBE tener estado de implementación
- PUEDE tener GAPs si no está implementado
- PUEDE tener evidencias
- SÍ cuenta para % de cumplimiento

...

### R2: Creación de GAPs

...

SOLO SE PUEDE crear GAP si:

- ✓ control.aplicable = 1
- ✓ control.estado IN ('no\_implementado', 'parcial')

NO SE PUEDE crear GAP si:

- ✗ control.aplicable = 0
- ✗ control.estado = 'implementado'

...

### R3: Cálculo de Avance de GAP

...

avance = (acciones\_completadas / total\_acciones) \* 100

DONDE:

acciones\_completadas = COUNT(acciones WHERE estado='completada' AND estado\_accion='activo')

total\_acciones = COUNT(acciones WHERE estado\_accion='activo')

Si avance = 100%:

- Marcar fecha\_real\_cierre = HOY
- Notificar al responsable
- Actualizar dashboard

NOTA: Solo se cuentan acciones con estado\_accion='activo'

(las eliminadas lógicamente no afectan el cálculo)

...

### R4: Soft Delete en Cascada

...

AL ELIMINAR un GAP (soft delete):

1. Marcar gap\_items.estado\_gap = 'eliminado'
2. Marcar TODAS sus acciones.estado\_accion = 'eliminada'
3. NO eliminar físicamente de la BD
4. Registrar en audit\_logs

CONSULTAS posteriores DEBEN filtrar:

WHERE estado\_gap = 'activo'

```
WHERE estado_gap = activo
AND estado_accion = 'activo'
'''
```

### ### R5: Completitud de Requerimientos

'''

PARA CADA requerimiento\_base:

1. Obtener controles asociados desde requerimientos\_controles
2. Filtrar solo controles APLICABLES de la empresa

SI algún control NO está implementado:

→ estado = 'pendiente' o 'en\_proceso'

SI TODOS los controles implementados:

VERIFICAR evidencias:

PARA CADA control:

SI NO tiene evidencias aprobadas:

→ estado = 'en\_proceso'

→ DETENER verificación

SI TODOS tienen evidencias aprobadas:

→ estado = 'completado'

→ fecha\_entrega = HOY

→ Agregar nota automática en observaciones

'''

### ### R6: Validación IDOR (Multi-Tenancy)

'''

EN CADA QUERY:

1. Obtener empresa\_id del usuario en sesión
2. Agregar WHERE empresa\_id = :tenant\_id
3. Si query retorna 0 resultados:
  - Log intento IDOR en security.log
  - Retornar error 403 Forbidden
  - NO revelar si el recurso existe

EJEMPLO - Descargar evidencia:

```
SELECT * FROM evidencias
```

```
WHERE id = :evidencia_id
```

```
AND empresa_id = :tenant_id ← OBLIGATORIO
```

LIMIT 1

SI no encuentra registro:

→ "No tiene permisos para acceder a este recurso"

```

R7: Rate Limiting por Acción

```

Login: 5 intentos / 15 minutos (por IP + email)

Uploads: 10 archivos / hora (por usuario)

Forms: 20 envíos / 5 minutos (por usuario)

API: 100 requests / minuto (por usuario)

Downloads: 50 archivos / hora (por usuario)

Almacenamiento: session + cache + BD

Bloqueo: temporal con contador exponencial

AL EXCEDER límite:

- Bloquear temporalmente
- Mostrar tiempo restante
- Log en security.log
- Notificar a admin si es repetitivo

```

R8: Auditoría de Cambios

```

EN cambios a datos críticos, registrar en audit\_logs:

- usuario\_id: quién hizo el cambio
- empresa\_id: tenant afectado
- tabla: qué tabla se modificó
- accion: INSERT/UPDATE/DELETE
- datos\_previos: JSON con valores anteriores
- datos\_nuevos: JSON con valores nuevos
- ip: dirección IP del usuario
- timestamp: cuándo ocurrió

TABLAS AUDITADAS:

- soa\_entries (cambios en evaluación de controles)
- gap\_items (creación/modificación de GAPs)