Documento de Desarrollo de Base de Datos (Simulada, MVP)

Sistema de Gestión Integral de Pólizas y Siniestros

> Objetivo: definir una **base de datos PostgreSQL simulada** (sin data real), que soporte el flujo MVP de pólizas y siniestros con: estados, documentos, beneficiarios, liquidación, pago, alertas, auditoría y reportes simples.

---

1) Alcance de la base de datos

Incluye (MVP)

- Esquema relacional en PostgreSQL.
- Catálogos mínimos (roles, estados, tipos de documento, tipos de alerta).
- Tablas operativas para:
- pólizas, vigencias y pagos,
- siniestros + expediente documental,
- beneficiarios y firmas,
- liquidación,
- pagos y contabilidad,
- alertas,
- auditoría.
- Estrategia de **datos simulados (seed)** para demo y pruebas.

No incluye (para mantener simple)

- Dimensional model / data mart.
- Data masking real ni replicación.
- Integración con fuentes externas.

---

2) Motor, convenciones y estándares

Motor

- **PostgreSQL 14+** (ideal 15/16).

Convenciones

- Tablas en `snake_case`.
- PK: `uuid`.
- Fechas: `timestamptz`.
- `created_at`, `updated_at` obligatorios.
- Eliminación lógica solo donde sea útil (`is_active`, `deleted_at`), pero en MVP evitar complejidad.

Reglas generales

- Relaciones 1:N con FK y `ON DELETE RESTRICT` (evitar borrados accidentales).
- Archivos se referencian con `files.id` (no guardar binarios en DB).

---

3) ERD lógico (texto)

Relación general

- `polizas (1) → (N) poliza_vigencias`
- `polizas (1) → (N) poliza_pagos`
- `siniestros (1) → (N) documentos`
- `siniestros (1) → (N) beneficiarios`
- `siniestros (1) → (1) liquidaciones`
- `siniestros (1) → (1) pagos`
- `files (1) → (N) documentos`
- `files (1) → (N) beneficiarios (firma_file_id)`
- `files (1) → (N) liquidaciones (liquidacion_file_id)`
- `files (1) → (N) pagos (comprobante_file_id)`
- `files (1) → (N) poliza_pagos (factura_file_id)`
- `users (1) → (N) siniestros (assigned_to_user_id)`
- `users (1) → (N) audit_events`
- `alertas` referencia (`ref_type`, `ref_id`) a `siniestros` o `polizas`.

---

4) Enumeraciones / catálogos (DB enums)

En PostgreSQL podemos usar `ENUM` por simplicidad y consistencia.

Roles

- `role_enum`: `ADMIN`, `GESTOR`, `FINANZAS`, `ASEGURADORA`

Estado siniestro

- `siniestro_estado_enum`: `RECIBIDO`, `EN_VALIDACION`, `BENEFICIARIOS`, `LIQUIDACION`, `PAGO`, `CERRADO`

Tipo siniestro

- `siniestro_tipo_enum`: `NATURAL`, `ACCIDENTE`, `DESCONOCIDO`

Documentos

- `documento_estado_enum`: `PENDIENTE`, `RECIBIDO`, `RECHAZADO`
- `documento_tipo_enum`:
- Base: `CEDULA_FALLECIDO`, `CERTIFICADO_DEFUNCION`, `POSESION_EFECTIVA`, `CEDULAS_BENEFICIARIOS`, `CUENTAS_BANCARIAS`, `CERTIFICADO_MATRICULA`
- Accidente: `ACTA_LEVANTAMIENTO`, `PARTE_POLICIAL`, `AUTOPSIA`, `ALCOHOLEMIA`

Beneficiarios

- `firma_estado_enum`: `PENDIENTE`, `RECIBIDA`

Liquidación

- `liquidacion_estado_enum`: `ENVIADA`, `OBSERVADA`, `APROBADA`

Pagos

- `pago_estado_enum`: `PENDIENTE`, `EJECUTADO`

Pólizas

- `poliza_estado_enum`: `ACTIVA`, `INACTIVA`
- `vigencia_estado_enum`: `ABIERTA`, `CERRADA`
- `poliza_pago_estado_enum`: `PENDIENTE`, `PAGADO`

Alertas

- `alerta_tipo_enum`: `PLAZO_60D`, `PLAZO_15D`, `PLAZO_72H`, `VENCIMIENTO_POLIZA`, `PAGO_POLIZA`
- `alerta_severidad_enum`: `INFO`, `WARNING`, `CRITICAL`
- `alerta_ref_type_enum`: `SINIestro`, `POLIZA`

> Nota: mantener exactamente como lo use el backend (ideal: `SINIestro` → corregir a `SINIESTRO` si se decide estándar).

---

5) Esquema físico (DDL sugerido)

> Este DDL es guía para implementación rápida. Puede migrarse con Prisma o SQL.

5.1 Extensiones

- `uuid-ossp` o generar UUID en app.

5.2 Tablas

`users`

Campos: - `id uuid pk` - `email varchar(255) unique not null` - `password_hash text not null` - `role role_enum not null` - `name varchar(200) not null` - `is_active boolean not null default true` - `created_at timestamptz not null default now()` - `updated_at timestamptz not null default now()`

Índices: - `idx_users_role`

`files`

Campos: - `id uuid pk` - `bucket varchar(200) not null` - `path text not null` - `original_name text not null` - `mime_type varchar(120) not null` - `size bigint not null` - `checksum varchar(128) null` - `created_at timestamptz not null default now()`

Índices: - `idx_files_created_at`

`polizas`

Campos: - `id uuid pk` - `codigo varchar(50) unique not null` - `nombre varchar(200) not null` - `tipo varchar(80) not null`

> Plantilla: `{POLIZA_TIPO}` - `prima numeric(14,2) not null default 0` - `moneda varchar(10) not null default 'USD'` - `estado poliza_estado_enum not null default 'ACTIVA'` - `created_at timestamptz not null default now()` - `updated_at timestamptz not null default now()`

Índices: - `idx_polizas_estado`

`poliza_vigencias`

Campos: - `id uuid pk` - `poliza_id uuid fk → polizas(id)` - `desde date not null` - `hasta date not null` - `estado vigencia_estado_enum not null default 'ABIERTA'` - `config_json jsonb null`

> Reglas/copagos/beneficios del plan (simulado) - `created_at timestamptz not null default now()` - `updated_at timestamptz not null default now()`

Restricciones: - Regla: solo 1 vigencia ABIERTA por póliza (validar por app; opcional constraint parcial).

Índices: - `idx_vigencias_poliza` - `idx_vigencias_hasta`

`poliza_pagos`

Campos: - `id uuid pk` - `poliza_id uuid fk → polizas(id)` - `fecha date not null` - `monto numeric(14,2) not null` - `estado poliza_pago_estado_enum not null default 'PENDIENTE'` - `factura_file_id uuid fk → files(id) null` - `created_at timestamptz not null default now()` - `updated_at timestamptz not null default now()`

Índices: - `idx_poliza_pagos_poliza` - `idx_poliza_pagos_estado`

`siniestros`

Campos: - `id uuid pk` - `case_code varchar(30) unique not null` - `tipo siniestro_tipo_enum not null default 'DESCONOCIDO'` - `estado siniestro_estado_enum not null default 'RECIBIDO'` - `fecha_defuncion date not null` - `observaciones text null`

Fallecido: - `fallecido_nombre varchar(200) not null` - `fallecido_cedula varchar(30) not null`

Reportante: - `reportante_nombre varchar(200) not null` - `reportante_relacion varchar(80) not null` - `reportante_email varchar(200) not null` - `reportante_telefono varchar(50) not null`

Relaciones: - `poliza_id uuid fk → polizas(id) null` - `poliza_vigencia_id uuid fk → poliza_vigencias(id) null` - `assigned_to_user_id uuid fk → users(id) null`

Fechas control: - `fecha_envio_aseguradora timestamptz null` - `fecha_liquidacion timestamptz null` - `fecha_firma_recibida timestamptz null` - `fecha_pago timestamptz null` - `fecha_cierre timestamptz null`

Auditoría: - `created_at timestamptz not null default now()` - `updated_at timestamptz not null default now()`

Índices: - `idx_siniestros_estado` - `idx_siniestros_tipo` - `idx_siniestros_fecha_defuncion` - `idx_siniestros_fallecido_cedula` - `idx_siniestros_assigned_to`

`documentos`

Campos: - `id uuid pk` - `siniestro_id uuid fk → siniestros(id)` - `tipo documento_tipo_enum not null` - `estado documento_estado_enum not null default 'PENDIENTE'` - `motivo_rechazo text null` - `file_id uuid fk → files(id) null` - `created_at timestamptz not null default now()` - `updated_at timestamptz not null default now()`

Restricciones: - Un tipo de documento por siniestro (opcional): `unique (siniestro_id, tipo)`

Índices: - `idx_documentos_siniestro` - `idx_documentos_estado`

`beneficiarios`

Campos: - `id uuid pk` - `siniestro_id uuid fk → siniestros(id)` - `nombre varchar(200) not null` - `cedula varchar(30) not null` - `relacion varchar(80) not null` - `porcentaje numeric(5,2) not null default 0`

> Regla: suma = `{TOTAL_PORCENTAJE}=100` - `banco varchar(120) not null` - `cuenta varchar(60) not null` - `estado_firma firma_estado_enum not null default 'PENDIENTE'` - `firma_file_id uuid fk → files(id) null` - `created_at timestamptz not null default now()` - `updated_at timestamptz not null default now()`

Restricciones: - Evitar duplicados por cédula dentro del siniestro: `unique (siniestro_id, cedula)`

Índices: - `idx_beneficiarios_siniestro`

`liquidaciones`

Campos: - `id uuid pk` - `siniestro_id uuid fk → siniestros(id) unique` - `monto_liquidado numeric(14,2) not null default 0` - `notas_aseguradora text null` - `estado liquidacion_estado_enum not null default 'ENVIADA'` - `liquidacion_file_id uuid fk → files(id) null` - `created_at timestamptz not null default now()` - `updated_at timestamptz not null default now()`

Índices: - `idx_liquidaciones_estado`

`pagos`

Campos: - `id uuid pk` - `siniestro_id uuid fk → siniestros(id) unique` - `estado pago_estado_enum not null default 'PENDIENTE'` - `doc_contable varchar(80) null` - `obs_finanzas text null` - `comprobante_file_id uuid fk → files(id) null` - `fecha_pago timestamptz null` - `created_at timestamptz not null default now()` - `updated_at timestamptz not null default now()`

Índices: - `idx_pagos_estado`

`alertas`

Campos: - `id uuid pk` - `tipo alerta_tipo_enum not null` - `severidad alerta_severidad_enum not null` - `mensaje text not null` - `ref_type alerta_ref_type_enum not null` - `ref_id uuid not null` - `fecha_limite timestamptz not null` - `is_resolved boolean not null default false` -

`created_at timestamptz not null default now()` - `updated_at timestamptz not null default now()`

Índices: - `idx_alertas_tipo` - `idx_alertas_resueltas` - `idx_alertas_fecha_limite` - `idx_alertas_ref (ref_type, ref_id)`

Restricciones (idempotencia): - `unique (tipo, ref_type, ref_id, fecha_limite)`

`audit_events`

Campos: - `id uuid pk` - `entity varchar(60) not null`

> Ej: `SINIESTRO`, `POLIZA` - `entity_id uuid not null` - `action varchar(80) not null` - `meta jsonb null` - `actor_user_id uuid fk → users(id) null` - `created_at timestamptz not null default now()`

Índices: - `idx_audit_entity (entity, entity_id)` - `idx_audit_created_at`

---

6) Restricciones y reglas (DB vs App)

Reglas que SÍ se pueden reforzar en DB

- Unicidad:
- `polizas.codigo`
- `siniestros.case_code`
- `documentos (siniestro_id, tipo)`
- `beneficiarios (siniestro_id, cedula)`
- `liquidaciones.siniestro_id` (1:1)
- `pagos.siniestro_id` (1:1)
- `alertas (tipo, ref_type, ref_id, fecha_limite)`

Reglas que se validan en aplicación (para rapidez)

- Transiciones de estado.
- Precondiciones (checklist completo, firmas completas, etc.).
- Suma de porcentajes de beneficiarios = 100.
- “Solo una vigencia ABIERTA por póliza”.

---

7) Índices recomendados (rendimiento MVP)

Consultas típicas

- Bandeja de siniestros por estado/fecha.
- Búsqueda por cédula y case_code.
- Alertas próximas a vencer.

- Reportes agregados por fechas.

Índices clave (ya listados)

- `siniestros(estado, updated_at desc)` (opcional compuesto)
- `siniestros(fallecido_cedula)`
- `alertas(is_resolved, fecha_limite)`
- `documentos(siniestro_id)`
- `beneficiarios(siniestro_id)`

---

8) Vistas / consultas para reportes (simples)

En MVP, puede hacerse desde el backend con SQL. Opcional crear `VIEW`.

8.1 Siniestralidad (por mes y tipo)

- Agrupar `siniestros` por `date_trunc('month', created_at)` y `tipo`.

8.2 Estado de siniestros

- Conteo por `estado`.

8.3 Costos de pólizas

- Suma de `polizas.prima` y/o `poliza_pagos.monto` por rango.

8.4 Alertas críticas

- `alertas` donde `is_resolved=false` y `severidad='CRITICAL'`.

---

9) Datos simulados (seed) — estrategia

Principios

- Generar datos consistentes y repetibles.
- Distribución realista de estados y plazos.
- Mantener volumen pequeño (para demo rápida).

Volumen recomendado (MVP demo)

- Usuarios: 4 (admin, gestor, finanzas, aseguradora)
- Pólizas: 3
- Vigencias: 3–6
- Siniestros: 30
- Documentos por siniestro: 6–10
- Beneficiarios por siniestro: 1–3
- Liquidaciones: ~60% de casos en estados posteriores
- Pagos: ~30% ejecutados
- Alertas: generadas por job o seed directo

Distribución de estados (ejemplo)

- `RECIBIDO`: 20%
- `EN_VALIDACION`: 25%
- `BENEFICIARIOS`: 20%
- `LIQUIDACION`: 20%
- `PAGO`: 10%
- `CERRADO`: 5%

---

10) Seed: ejemplos de registros (plantillas)

10.1 Usuarios

- `{email: "admin@demo.com", role: ADMIN}`
- `{email: "gestor@demo.com", role: GESTOR}`
- `{email: "finanzas@demo.com", role: FINANZAS}`
- `{email: "aseguradora@demo.com", role: ASEGURADORA}`

10.2 Pólizas

- `POL-001`: `{POLIZA_TIPO}="VIDA"`, prima 5000
- `POL-002`: `{POLIZA_TIPO}="ACCIDENTES"`, prima 3200
- `POL-003`: `{POLIZA_TIPO}="SALUD"`, prima 12000

10.3 Siniestros (case_code)

Formato recomendado: - `SIN-{YYYY}-{NNNNNN}` Ej: - `SIN-2026-000001`

Campos simulados: - `fecha_defuncion`: entre hoy-5d y hoy-80d - `tipo`: NATURAL 70%, ACCIDENTE 25%, DESCONOCIDO 5%

---

11) Jobs y datos simulados de alertas

Reglas seed (si no corren jobs)

- Para cada siniestro crear alerta `PLAZO_60D` con `fecha_limite = fecha_defuncion + {PLAZO_REPORTE_DIAS}`.
- Para siniestros con `fecha_envio_aseguradora` crear `PLAZO_15D`.
- Para siniestros con `fecha_firma_recibida` crear `PLAZO_72H`.
- Para pólizas/vigencias próximas a vencer crear `VENCIMIENTO_POLIZA`.

Severidad: - `CRITICAL` si faltan <= `{UMBRAL_CRITICO_DIAS}` o <= 12h para 72h.

---

12) Migraciones (flujo simple)

Estrategia

- Migraciones incrementales.
- Primero crear enums, luego tablas principales, luego índices/constraints.

Orden sugerido

1. Enums
2. users, files
3. polizas, poliza_vigencias, poliza_pagos
4. siniestros
5. documentos, beneficiarios
6. liquidaciones, pagos
7. alertas
8. audit_events

---

13) Backups y mantenimiento (mínimo)

MVP (entorno de demo)

- Backup manual (dump) diario:
- `pg_dump -Fc dbname > backup.dump`
- Restore:
- `pg_restore -d dbname backup.dump`

---

14) Seguridad de datos (simulado)

- No guardar documentos en DB.
- Evitar almacenar números sensibles más allá de cédula (si aplica, enmascarar en UI).
- Logs sin PII completa (en demo se tolera, en prod se ajusta).

---

15) Parámetros de negocio (plantillas)

- `{PLAZO_REPORTE_DIAS}=60`
- `{PLAZO_LIQUIDACION_DIAS_HABILES}=15`
- `{PLAZO_PAGO_HORAS}=72`
- `{UMBRAL_CRITICO_DIAS}=10`
- `{TOTAL_PORCENTAJE}=100`
- `{POLIZA_TIPO}` lista libre
- `{MONEDA_DEFAULT}="USD"`

---

16) Checklist de aceptación (DB)

- Se puede crear siniestro y consultar por bandeja (estado/fecha).
- Se pueden asociar documentos y actualizar su estado.
- Se pueden registrar beneficiarios y firma.
- Se puede registrar liquidación (1:1) y pago (1:1).
- Se pueden generar alertas sin duplicados (unique compuesto).
- Se pueden ejecutar reportes agregados básicos por rango.

---

17) Nota de coherencia con Backend

Este diseño corresponde al backend MVP: - Estados simples y transiciones en aplicación. - Archivos como metadata + almacenamiento externo. - Alertas por job idempotente.

Si más adelante se requiere: calendario laboral (días hábiles), multi-aseguradora, multi-empresa, o trazabilidad avanzada, se extiende con tablas adicionales.