Documento de Desarrollo Backend (MVP)

Sistema de Gestión Integral de Pólizas y Siniestros

> Objetivo: construir un backend **sencillo, funcional y rápido**, centrado en: registro de siniestros, control por estados, expediente documental, beneficiarios, liquidación, pagos, pólizas/vigencias, alertas y reportes tabulares.

---

1) Alcance backend (qué incluye / qué NO)

Incluye (MVP)

1. **Autenticación y autorización** (JWT + roles).
2. **Portal público**: endpoint para crear reporte de siniestro + acuse.
3. **Gestión de siniestros**:
4. listado, filtros, detalle,
5. cambios de estado (flujo por etapas),
6. bitácora/auditoría de eventos,
7. checklist documental y carga de archivos.
8. **Beneficiarios y firmas** (estado + archivos firmados).
9. **Liquidación** (registro, archivo, monto, observaciones).
10. **Pago/cierre** (registro de pago, comprobante, doc contable, cierre de caso).
11. **Pólizas**: CRUD mínimo, vigencias, pagos, estado.
12. **Alertas** por plazos y vencimientos (jobs programados).
13. **Reportes simples** (consultas agregadas + export CSV desde frontend).

No incluye (para no complejizar)

- Motor BPM o workflow avanzado.
- Firma electrónica avanzada (solo “subir PDF firmado” + estado).
- Integraciones reales con ERP/Finanzas/Broker (solo captura de referencias y adjuntos).
- Data warehouse/BI (solo reportes tabulares).

---

2) Stack recomendado (rápido y mantenible)

Lenguaje y framework

- **Node.js 20 + TypeScript**
- **NestJS** (estructura modular, DI, validación, guards)

Persistencia

- **PostgreSQL**
- **Prisma ORM** (migraciones, tipado, rapidez)

Auth

- **JWT** (access token) + refresh opcional (MVP: solo access)
- Hash: **bcrypt**

Validación

- **class-validator + class-transformer** (DTOs Nest)

Jobs/colas (alertas)

- **BullMQ + Redis** (recomendado)
- Alternativa sin Redis: cron + consultas directas (si el entorno no permite Redis)

Archivos

- **S3 compatible** (MinIO/AWS) recomendado
- Alternativa MVP: almacenamiento local + URL firmada interna

Observabilidad

- **pino** (logger) o `nestjs-pino`
- **Swagger (OpenAPI)**

Tests

- **Jest** (unit)
- **supertest** (e2e)

---

3) Arquitectura backend por capas (simple)

Capas (por módulo)

1. **Controller (API)**: rutas, validación DTO, respuesta.
2. **Service (Aplicación)**: reglas, orquestación.
3. **Repository (Prisma)**: acceso DB.
4. **Domain (Tipos/Enums)**: estados, roles, transiciones.

Regla práctica: - Controllers sin lógica de negocio. - Services con reglas y transiciones. - Prisma sólo en repositorios/servicios (sin queries dispersas).

---

4) Estructura de carpetas (archivo por archivo)

```
backend/
  src/
    main.ts
    app.module.ts
    config/
      env.validation.ts
```

```
    configuration.ts
  common/
    filters/
      http-exception.filter.ts
    interceptors/
      audit.interceptor.ts
    guards/
      jwt-auth.guard.ts
      roles.guard.ts
    decorators/
      roles.decorator.ts
      user.decorator.ts
    pipes/
      zod.pipe.ts (opcional)
    utils/
      dates.ts
      pagination.ts
  prisma/
    prisma.module.ts
    prisma.service.ts
  auth/
    auth.module.ts
    auth.controller.ts
    auth.service.ts
    strategies/jwt.strategy.ts
    dto/login.dto.ts
  users/
    users.module.ts
    users.controller.ts (opcional MVP)
    users.service.ts
    dto/
    users.repository.ts
  files/
    files.module.ts
    files.controller.ts
    files.service.ts
    providers/
      s3.provider.ts
    dto/
  polizas/
    polizas.module.ts
    polizas.controller.ts
    polizas.service.ts
    polizas.repository.ts
    dto/
  siniestros/
    siniestros.module.ts
    siniestros.controller.ts
    siniestros.public.controller.ts
    siniestros.service.ts
    siniestros.repository.ts
```

```
    transitions/
      estados.enum.ts
      transitions.map.ts
    dto/
      crear-reporte-publico.dto.ts
      actualizar-siniestro.dto.ts
      cambiar-estado.dto.ts
      list-siniestros.query.ts
  documentos/
    documentos.module.ts
    documentos.controller.ts
    documentos.service.ts
    documentos.repository.ts
    dto/
  beneficiarios/
    beneficiarios.module.ts
    beneficiarios.controller.ts
    beneficiarios.service.ts
    beneficiarios.repository.ts
    dto/
  liquidacion/
    liquidacion.module.ts
    liquidacion.controller.ts
    liquidacion.service.ts
    dto/
  pagos/
    pagos.module.ts
    pagos.controller.ts
    pagos.service.ts
    dto/
  alertas/
    alertas.module.ts
    alertas.controller.ts
    alertas.service.ts
    jobs/
      alertas.scheduler.ts
    dto/
  reportes/
    reportes.module.ts
    reportes.controller.ts
    reportes.service.ts
    queries/
      siniestralidad.sql
      estado-siniestros.sql
prisma/
  schema.prisma
  migrations/
test/
  e2e/
Dockerfile
```

```
docker-compose.yml
.env.example
```

---

5) Modelo de datos (PostgreSQL + Prisma)

Se define **lo mínimo** para soportar el flujo completo.

Entidades principales

`User`

- `id` (uuid)
- `email` (unique)
- `passwordHash`
- `role` enum: `GESTOR | FINANZAS | ASEGURADORA | ADMIN`
- `name`
- `isActive`
- `createdAt`, `updatedAt`

`Poliza`

- `id`
- `codigo` (unique)
- `nombre`
- `tipo` `{POLIZA_TIPO}`
- `prima` (numeric)
- `moneda` (string)
- `estado` enum: `ACTIVA | INACTIVA`
- `createdAt`, `updatedAt`

`PolizaVigencia`

- `id`
- `polizaId`
- `desde` (date)
- `hasta` (date)
- `estado` enum: `ABIERTA | CERRADA`
- (opcional) `configJson` (JSONB: reglas/copagos/beneficios)

`PolizaPago`

- `id`
- `polizaId`
- `fecha`
- `monto`
- `estado` enum: `PENDIENTE | PAGADO`
- `facturaFileId` (nullable)

`Siniestro`

- `id`
- `caseCode` (unique, legible: ej. `SIN-2026-000123`)
- `tipo` enum: `NATURAL | ACCIDENTE | DESCONOCIDO`
- `estado` enum (ver sección 6)
- `fechaDefuncion`
- `observaciones` (text)

Fallecido (embebido como columnas simples) - `fallecidoNombre` - `fallecidoCedula`

Reportante - `reportanteNombre` - `reportanteRelacion` - `reportanteEmail` - `reportanteTelefono`

Relaciones - `polizaId` (nullable) - `polizaVigenciaId` (nullable) - `assignedToUserId` (nullable)

Fechas de control - `fechaReporte` (createdAt) - `fechaEnvioAseguradora` (nullable) - `fechaLiquidacion` (nullable) - `fechaFirmaRecibida` (nullable) - `fechaPago` (nullable) - `fechaCierre` (nullable)

`Documento`

- `id`
- `siniestroId`
- `tipo` enum (ver checklist)
- `estado` enum: `PENDIENTE | RECIBIDO | RECHAZADO`
- `motivoRechazo` (nullable)
- `fileId` (nullable)
- `createdAt`, `updatedAt`

`Beneficiario`

- `id`
- `siniestroId`
- `nombre`
- `cedula`
- `relacion`
- `porcentaje` (numeric)
- `banco`
- `cuenta`
- `estadoFirma` enum: `PENDIENTE | RECIBIDA`
- `firmaFileId` (nullable)

`Liquidacion`

- `id`
- `siniestroId` (unique)
- `montoLiquidado`
- `notasAseguradora` (text)
- `liquidacionFileId` (nullable)
- `estado` enum: `ENVIADA | OBSERVADA | APROBADA`

- `createdAt`, `updatedAt`

`Pago`

- `id`
- `siniestroId` (unique)
- `estado` enum: `PENDIENTE | EJECUTADO`
- `docContable` (nullable)
- `obsFinanzas` (nullable)
- `comprobanteFileId` (nullable)
- `fechaPago` (nullable)

`File`

- `id`
- `bucket` / `path`
- `originalName`
- `mimeType`
- `size`
- `checksum` (opcional)
- `createdAt`

`AuditEvent`

- `id`
- `entity` (string: `SINIestro`, `POLIZA`, etc.)
- `entityId`
- `action` (string)
- `meta` (JSONB)
- `actorUserId` (nullable)
- `createdAt`

`Alerta`

- `id`
- `tipo` enum: `PLAZO_60D | PLAZO_15D | PLAZO_72H | VENCIMIENTO_POLIZA | PAGO_POLIZA`
- `severidad` enum: `INFO | WARNING | CRITICAL`
- `mensaje`
- `refType` enum: `SINIestro | POLIZA`
- `refId`
- `fechaLimite`
- `isResolved` boolean
- `createdAt`

---

6) Estados del siniestro y transiciones (máquina simple)

Estados (MVP)

- `RECIBIDO`
- `EN_VALIDACION`
- `BENEFICIARIOS`
- `LIQUIDACION`
- `PAGO`
- `CERRADO`

Transiciones permitidas

- `RECIBIDO → EN_VALIDACION`
- `EN_VALIDACION → BENEFICIARIOS`
- `BENEFICIARIOS → LIQUIDACION`
- `LIQUIDACION → PAGO`
- `PAGO → CERRADO`

Reglas de negocio (plantillas): - Para pasar `EN_VALIDACION → BENEFICIARIOS`: checklist base completo (`Documento.estado=RECIBIDO` para tipos requeridos). - Para pasar `BENEFICIARIOS → LIQUIDACION`: beneficiarios suman 100% `{TOTAL_PORCENTAJE}=100` y firmas recibidas (o bandera `{PERMITIR_SIN_FIRMAS}` en false). - Para pasar `LIQUIDACION → PAGO`: liquidación registrada con monto y archivo. - Para pasar `PAGO → CERRADO`: pago ejecutado + doc contable `{docContable}` (si rol Finanzas lo exige).

Implementación: - `transitions.map.ts` define allowed transitions. - `SiniestrosService.cambiarEstado()` valida: 1) rol, 2) transición, 3) precondiciones, 4) actualiza estado, 5) crea `AuditEvent`.

---

7) Checklist documental (configurable)

Tipos de documento (enum)

Base (Natural): - `CEDULA_FALLECIDO` - `CERTIFICADO_DEFUNCION` - `POSESION_EFECTIVA` - `CEDULAS_BENEFICIARIOS` - `CUENTAS_BANCARIAS` - `CERTIFICADO_MATRICULA`

Accidente (adicional): - `ACTA_LEVANTAMIENTO` - `PARTE_POLICIAL` - `AUTOPSIA` - `ALCOHOLEMIA`

Configuración

- Tabla `DocumentoChecklistConfig` (opcional) o JSON en `.env`/DB:
- requerido por `tipo` (Natural/Accidente)
- permite marca “opcional”.

MVP recomendado: - Hardcode en backend (enum + arrays) para velocidad.

---

8) API (rutas y contratos mínimos)

Convención: respuestas JSON consistentes.

8.1 Auth

- `POST /auth/login`
- body: `{ email, password }`
- resp: `{ accessToken, user: { id, name, role } }`

8.2 Público

- `POST /public/siniestros`
- body: `CrearReportePublicoDto`
- resp: `{ caseCode, id }`

8.3 Siniestros

- `GET /siniestros` (auth)
- query: `search, estado, tipo, fechaDesde, fechaHasta, page, pageSize`
- resp: `{ items, pageInfo }`
- `GET /siniestros/:id` (auth)
- resp: `SiniestroDetail`
- `PATCH /siniestros/:id` (auth)
- body: campos editables (observaciones, asignación, poliza, etc.)
- `POST /siniestros/:id/estado` (auth)
- body: `{ nextEstado, notas? }`

8.4 Documentos

- `POST /siniestros/:id/documentos`
- multipart: `file` + `tipo`
- crea/actualiza `Documento` con `estado=RECIBIDO`
- `PATCH /siniestros/:id/documentos/:docId`
- body: `{ estado, motivoRechazo? }`
- `GET /siniestros/:id/documentos`

8.5 Beneficiarios

- `GET /siniestros/:id/beneficiarios`

- `POST /siniestros/:id/beneficiarios`
- `PATCH /beneficiarios/:beneficiarioId`
- `DELETE /beneficiarios/:beneficiarioId`

Firmas: - `POST /beneficiarios/:id/firma` (multipart) - marca `estadoFirma=RECIBIDA`

8.6 Liquidación

- `POST /siniestros/:id/liquidacion` (auth)
- body/multipart: `montoLiquidado`, `notasAseguradora`, `estado`, `file`
- `GET /siniestros/:id/liquidacion`

8.7 Pago

- `POST /siniestros/:id/pago` (auth)
- body/multipart: `{ fechaPago, docContable, obsFinanzas, file? }`
- `GET /siniestros/:id/pago`

8.8 Pólizas

- `GET /polizas`
- `POST /polizas`
- `GET /polizas/:id`
- `PATCH /polizas/:id`

Vigencias - `POST /polizas/:id/vigencias` (abrir) - `POST /polizas/vigencias/:vigenciaId/cerrar`

Pagos - `POST /polizas/:id/pagos` (monto, fecha, estado, factura file?)

8.9 Alertas

- `GET /alertas`
- `POST /alertas/:id/resolver`

8.10 Reportes

- `GET /reportes/siniestralidad?desde&hasta&polizaId?`
- `GET /reportes/estado-siniestros?desde&hasta&estado?`
- `GET /reportes/costos-polizas?desde&hasta`
- `GET /reportes/poblacion-asegurada?polizaId`

---

9) Autorización (RBAC simple)

Reglas por rol (MVP)

- `GESTOR`:
- CRUD siniestros, documentos, beneficiarios, liquidación, cierre.
- `FINANZAS`:
- leer siniestros + registrar pago + doc contable.
- `ASEGURADORA` (opcional):
- leer siniestros asignados + registrar liquidación.
- `ADMIN`:
- administrar usuarios (opcional MVP).

Implementación: - Decorador `@Roles('GESTOR')` + `RolesGuard`. - En algunos endpoints validar adicional por estado (ej. pago solo en estado `PAGO`).

---

10) Alertas y plazos (jobs)

Parámetros (config)

- `{PLAZO_REPORTE_DIAS}=60`
- `{PLAZO_LIQUIDACION_DIAS_HABILES}=15` (en MVP se puede manejar como días calendario si no hay calendario laboral)
- `{PLAZO_PAGO_HORAS}=72`
- `{UMBRAL_CRITICO_DIAS}=10`

Cálculos (MVP)

- **PLAZO_60D**: `fechaDefuncion + 60 días` (o según regla real) → alerta si falta <= umbral.
- **PLAZO_15D**: desde `{fechaEnvioAseguradora}`.
- **PLAZO_72H**: desde `{fechaFirmaRecibida}` o `{fechaLiquidacion}` según negocio.
- **VENCIMIENTO_POLIZA**: `{vigencia.hasta}`.

Frecuencia

- Job diario 06:00: genera/actualiza alertas.
- Job cada hora: solo para reglas 72h (si quieren más exactitud).

Idempotencia

- Al generar alertas:
- buscar alerta existente por `(tipo, refId, fechaLimite)`.
- actualizar severidad/mensaje, no duplicar.

---

11) Manejo de archivos (documentos/firmas/liquidación)

Flujo

1. Cliente sube archivo (multipart) a `FilesController` o directamente en módulo correspondiente.
2. Backend guarda metadata en tabla `File`.
3. Guarda el binario:
4. S3: `bucket`, `path`.
5. Local: `uploads/{id}-{originalName}`.
6. Vincula `fileId` al registro (Documento, Beneficiario.firma, Liquidación, Pago, PolizaPago).

Endpoints utilitarios (opcional)

- `GET /files/:id/download` (firma URL o stream)

Seguridad: - Validar tamaño máximo `{MAX_FILE_MB}` - Validar mime types permitidos `{ALLOWED_MIME}`

---

12) Auditoría y trazabilidad

Qué auditar (mínimo)

- Creación de siniestro
- Cambio de estado
- Subida/rechazo de documentos
- CRUD beneficiarios
- Registro de liquidación
- Registro de pago
- Cierre

Implementación: - Interceptor `audit.interceptor.ts` para eventos genéricos. - En servicios, crear `AuditEvent` explícito para acciones críticas.

---

13) Manejo de errores y respuesta estándar

Respuesta estándar

- `200/201`: `{ data, meta? }`
- `400`: validación
- `401`: no autenticado
- `403`: no autorizado
- `404`: no existe
- `409`: conflicto (ej. transición inválida)
- `422`: negocio (precondición no cumple)

Filtro global

- `HttpExceptionFilter` para formatear:
- `{ code, message, details? }`

---

14) Paginación, filtros y orden

Convención

- query: `page=1`, `pageSize=20`, `sort=updatedAt:desc`
- respuesta:
- `items: []`
- `pageInfo: { page, pageSize, totalItems, totalPages }`

Implementación: - `common/utils/pagination.ts` - Prisma `skip/take`.

---

15) Reportes (consultas simples)

Principio

- Reportes como **consultas agregadas** (Prisma raw SQL si conviene).

Ejemplos: - Siniestralidad: conteo siniestros por póliza/tipo/mes. - Estado siniestros: conteo por estado. - Costos pólizas: suma primas/pagos. - Población asegurada: en MVP puede venir de tabla/CSV importado por backend (si existe).

---

16) Seguridad básica (MVP)

- JWT con expiración `{JWT_EXPIRES_IN}`.
- Rate limit en público `/public/siniestros` (evitar spam) `{PUBLIC_RATE_LIMIT}`.
- Sanitizar inputs.
- CORS permitido a frontend.
- Logs sin datos sensibles (no loggear contraseñas ni tokens).

---

17) DevOps rápido (local)

docker-compose (dev)

- `postgres`
- `redis` (si BullMQ)
- `minio` (opcional)
- `backend`

Scripts

- `npm run prisma:migrate`

- `npm run seed` (usuarios y catálogos)
- `npm run start:dev`

---

18) Seed (datos mínimos)

- Usuarios:
- admin
- gestor1
- finanzas1
- Póliza demo + vigencia abierta.
- Checklist hardcode.

---

19) Plan por fases y entregables (alineado a frontend)

Fase 0 — Setup (0.5–1 día)

Entregables: NestJS base, Prisma+Postgres, Swagger, Auth JWT.

Fase 1 — Público (1 día)

Entregables: `POST /public/siniestros`, generación `caseCode`, auditoría.

Fase 2 — Siniestros core (2–3 días)

Entregables: listar, filtrar, detalle, cambio de estado con reglas.

Fase 3 — Documentos + Beneficiarios (2 días)

Entregables: endpoints documentos, checklist, firmas/archivos beneficiarios.

Fase 4 — Liquidación + Pago + Pólizas (2–3 días)

Entregables: liquidación, pago/doc contable, pólizas/vigencias/pagos.

Fase 5 — Alertas + Reportes (1–2 días)

Entregables: job alertas, endpoints alertas, endpoints reportes agregados.

---

20) Notas para mantenerlo simple

- Estado del siniestro como **fuente de verdad** (una columna, sin workflows complejos).
- Checklist documental: arrays por tipo (hardcode) antes que tablas complejas.
- Alertas: job diario + idempotencia.
- Archivos: S3/MinIO si se puede; si no, local con cuidado.

---

21) Variables de negocio (plantillas parametrizables)

- `{PLAZO_REPORTE_DIAS}=60`
- `{PLAZO_LIQUIDACION_DIAS_HABILES}=15`
- `{PLAZO_PAGO_HORAS}=72`
- `{MAX_FILE_MB}=10`
- `{ALLOWED_MIME}=["application/pdf","image/jpeg","image/png"]`
- `{JWT_EXPIRES_IN}="8h"`
- `{PUBLIC_RATE_LIMIT}="20/min"`
- `{TOTAL_PORCENTAJE}=100`
- `{PERMITIR_SIN_FIRMAS}=false`