DOCUMENTACIÓN TÉCNICA Y OPERATIVA

SISTEMA INTEGRAL DE SEGUROS CONTINENTAL (SISC)

25 de noviembre de 2025

GESTION DE BASE DE DATOS

David de Jesús Mosco Gasca
Diego Antonio Saviñón Escamilla

ÍNDICE DE CONTENIDOS

CAPÍTULO 1: INTRODUCCIÓN Y ALCANCE

1.1 Resumen Ejecutivo

El Sistema Integral de Seguros Continental (SISC) es una plataforma tecnológica de gestión empresarial (ERP) diseñada para centralizar la operación de aseguradoras con presencia multinacional. El sistema aborda la problemática crítica de la dispersión de datos y la latencia en la toma de decisiones mediante una arquitectura centralizada con nodos lógicos regionales.

La solución garantiza la integridad financiera mediante transacciones ACID, protege la privacidad de los usuarios mediante fragmentación vertical y optimiza la carga operativa mediante algoritmos de asignación automática de tareas.

1.2 Objetivos del Proyecto

Integridad Transaccional: Asegurar que todas las operaciones financieras (pagos y siniestros) sean atómicas, consistentes, aisladas y duraderas.

Automatización de Reglas de Negocio: Reducir la intervención humana en procesos de validación de vigencia y renovación de pólizas.

Seguridad por Diseño: Implementar controles de acceso basados en roles (RBAC) y separación física de datos sensibles (PII).

Inteligencia de Negocios (BI): Proveer visibilidad en tiempo real sobre el rendimiento regional y la productividad de los agentes.

CAPÍTULO 2: ARQUITECTURA DEL SISTEMA

El SISC implementa una arquitectura de Tres Capas (3-Tier Architecture) estrictamente desacopladas para garantizar escalabilidad y mantenibilidad.

2.1 Nivel de Presentación (Frontend)

Tecnologías: HTML5, CSS3 (Diseño responsivo "Glassmorphism"), JavaScript (Fetch API).

Función: Interfaz gráfica reactiva que consume servicios REST de forma asíncrona, eliminando la recarga completa de página para una experiencia de usuario fluida.

2.2 Nivel de Aplicación (Middleware)

Tecnologías: Python 3.10, Flask Framework 3.0.

Función: Actúa como orquestador de peticiones. Implementa la lógica de sesión, validación de tipos de datos y enrutamiento seguro hacia la base de datos. No almacena estado (Stateless).

2.3 Nivel de Datos (Backend)

Tecnologías: Microsoft SQL Server 2022.

Función: Motor de persistencia y ejecución de reglas de negocio críticas. Aloja los Procedimientos Almacenados, Triggers y Vistas Materializadas.

CAPÍTULO 3: DISEÑO DE BASE DE DATOS (DICCIONARIO DE DATOS)

El modelo de datos cumple con la 3ra Forma Normal y utiliza estrategias avanzadas de particionamiento lógico.

3.1 Entidades Maestras y Fragmentación

Tabla: Clientes_General

Descripción: Almacena los identificadores públicos del cliente.

cliente_id (PK): Identificador único autoincremental.

nombre_completo: Nombre legal del asegurado.

email: Correo electrónico único (Índice B-Tree).

pais: Región de residencia (Nodo lógico).

Tabla: Clientes_Sensible (Fragmentación Vertical)

Descripción: Almacena datos de contacto y documentos oficiales. Separada físicamente por motivos de cumplimiento normativo (GDPR).

cliente_id (PK, FK): Vinculación 1:1 con la tabla general.

telefono / direccion: Datos de contacto privado.

numero_documento: ID Oficial (INE, Pasaporte).

Tabla: Agentes

Descripción: Catálogo de usuarios del sistema con roles y credenciales.

agente_id (PK): Identificador del empleado.

usuario / contrasena: Credenciales de acceso cifradas.

puesto: Rol operativo (Admin, Agente, Auditor).

3.2 Entidades Transaccionales

Tabla: Polizas

Descripción: Entidad central que representa el contrato de seguro.

poliza_id (PK): Número de póliza.

prima_anual: Costo del seguro. Constraint: CHECK (prima_anual >= 0).

suma_asegurada: Límite máximo de cobertura financiera.

estado_poliza: Máquina de estados (Activa, Vencida, Cancelada).

Tabla: Siniestros

Descripción: Registro de incidentes o reclamos.

siniestro_id (PK): Número de expediente.

monto_estimado: Valor del reclamo. Constraint: CHECK (monto >= 0).

agente_id (FK): Agente asignado mediante algoritmo Round Robin.

Tabla: Pagos

Descripción: Registro financiero de ingresos.

pago_id (PK): Folio de transacción.

monto: Importe pagado. Su inserción dispara triggers de renovación automática.

3.3 Infraestructura de Auditoría

Tabla: Auditoria_General

Descripción: Bitácora inmutable (Append-Only) para trazabilidad forense. Registra Usuario, Fecha, Acción y Detalles del cambio.

Tabla: Auditoria_Fallos_Transaccion

Descripción: Registro técnico de excepciones capturadas por bloques TRY/CATCH (errores de sistema, timeouts, violaciones de integridad).

CAPÍTULO 4: INGENIERÍA DE SOFTWARE: LÓGICA Y ALGORITMOS

Se justifica la implementación de lógica crítica directamente en el motor de base de datos para maximizar el rendimiento y la consistencia.

4.1 Integridad Transaccional (Modelo ACID)

Componente: sp_registrar_siniestro

El sistema utiliza transacciones explícitas (BEGIN TRANSACTION) para asegurar la atomicidad.

Validación: Verifica vigencia de la póliza y que el monto no exceda la suma_asegurada.

Ejecución: Inserta el siniestro y actualiza el estado de la póliza simultáneamente.

Consistencia: Si ocurre cualquier error, se ejecuta un ROLLBACK total, evitando datos corruptos.

4.2 Algoritmos de Balanceo de Carga (Load Balancing)

Componente: Script de Asignación y sp_reporte_desempeno_agentes.

Para evitar la saturación de un solo agente, el sistema implementa un algoritmo de asignación tipo Round Robin o aleatorio ponderado. Esto asegura que los siniestros se distribuyan equitativamente entre todo el personal activo, permitiendo una medición justa de la productividad ("Estrella" vs "Regular").

4.3 Automatización Reactiva

Componente: sp_registrar_pago.

El sistema no es pasivo; reacciona a eventos financieros. Al detectar un pago en una póliza vencida que cubra el porcentaje mínimo configurado (90%), el sistema dispara automáticamente la reactivación de la póliza y extiende su vigencia un año, sin intervención humana.

CAPÍTULO 5: ESPECIFICACIÓN DE INTERFAZ (API REST)

El Middleware Python expone los siguientes servicios JSON para consumo del cliente web:

Verbo HTTP	Endpoint	Descripción	Payload Requerido
POST	`/api/siniestro/registrar`	Alta de un nuevo reclamo.	`{poliza_id, tipo, monto, agente_id}`
POST	`/api/pago/registrar`	Procesamiento de pagos.	`{poliza_id, monto, metodo}`
POST	`/api/poliza/cancelar`	Baja lógica de contrato.	`{poliza_id}`
GET	`/api/reportes/avanzado`	Dashboard BI (Multi-dataset).	*Ninguno*
GET	`/api/reportes/agentes`	Ranking de productividad.	*Ninguno*
GET	`/api/auditoria/fallos`	Logs de errores técnicos.	*Ninguno*

CAPÍTULO 6: GOBIERNO DE DATOS, SEGURIDAD Y CUMPLIMIENTO

6.1 Protección de Datos Personales (Compliance)

El sistema adhiere a los principios de "Privacidad por Diseño" alineados con normativas como GDPR y LFPDPPP.

Técnica: Fragmentación Vertical.

Justificación: Los datos de contacto (Clientes_Sensible) están físicamente aislados de los datos operativos. Los reportes gerenciales solo acceden a Clientes_General, minimizando la superficie de exposición de datos personales.

6.2 Control de Acceso (RBAC)

Se implementa seguridad a nivel de aplicación mediante decoradores Python (@admin_required, @auditor_required). Estos interceptan cada petición HTTP, validando la firma de la sesión y los privilegios del rol antes de ejecutar cualquier lógica.

6.3 Auditoría Forense y No Repudio

El sistema garantiza la trazabilidad total. Los triggers de auditoría (trg_Audit_Polizas) operan a nivel de motor de base de datos, haciendo imposible que un usuario modifique un registro sin dejar una huella inmutable en la bitácora.

CAPÍTULO 7: PLAN DE PRUEBAS Y QA

Se ejecutó una batería de pruebas para validar la robustez del sistema ante fallos.

ID Caso	Escenario de Prueba	Resultado Esperado	Resultado Obtenido
QA-01	Siniestro > Suma Asegurada	Rechazo de operación.	Alerta Roja: "Monto excede cobertura".
QA-02	Siniestro en Póliza Vencida	Rechazo de operación.	Error 409: "Póliza Inactiva".
QA-03	Pago de Reactivación	Cambio de estado a "Activa".	Póliza renovada por 1 año.
QA-04	Inyección SQL en Login	Bloqueo de intento.	Consultas parametrizadas rechazan input.
QA-05	Fallo de Transacción	Rollback automático.	Registro en tabla `Auditoria_Fallos`.

CAPÍTULO 8: ESTRATEGIA DE DESPLIEGUE (DEVOPS)

Hoja de ruta sugerida para el paso a producción en infraestructura Cloud.

Contenedorización: Empaquetado del backend Flask mediante Docker (Imagen base: python:3.10-slim), incluyendo controladores ODBC para Linux.

Orquestación: Despliegue en cluster Kubernetes con escalado horizontal automático (HPA) basado en uso de CPU.

Alta Disponibilidad (HA): Configuración de SQL Server Always On Availability Groups con un nodo primario de escritura y réplicas de solo lectura para desahogar la carga de los reportes de BI.

CAPÍTULO 9: MANUAL DE INSTALACIÓN (ADMINISTRADOR)

9.1 Requisitos del Entorno

Motor: SQL Server 2019/2022 Developer Edition.

Runtime: Python 3.10+.

Librerías: pyodbc, flask.

9.2 Procedimiento de Despliegue

Base de Datos: Ejecutar el script maestro SISC_DB_Install.sql en SSMS para crear estructuras y cargar datos semilla.

Backend:

Instalar dependencias: pip install -r requirements.txt.

Configuración: Editar config.py y establecer la variable SERVER_NAME con la instancia local (ej. LOCALHOST\SQLEXPRESS).

Arranque: Ejecutar python app.py y verificar salida en puerto 5000.

CAPÍTULO 10: MANUAL DE OPERACIONES (GUÍA DE USUARIO)

Esta sección describe el flujo de trabajo en la interfaz web del sistema, detallando las interacciones disponibles para cada perfil de usuario.

10.1. ACCESO Y SEGURIDAD

Pantalla: Inicio de Sesión (Login) El sistema cuenta con un único punto de entrada protegido.

1. **Ingreso:** El usuario debe introducir su Usuario y Contraseña asignados.

2. **Validación:** El sistema verifica las credenciales contra la base de datos SQL Server mediante un algoritmo de comparación segura (sp_validar_agente).
3. **Roles:** Dependiendo del perfil (Admin, Agente, Auditor), el menú de navegación superior mostrará u ocultará opciones automáticamente.
    - *Nota:* Si un usuario inactivo intenta entrar, el sistema bloqueará el acceso mostrando un mensaje de error en rojo.

10.2. PERFIL OPERATIVO (AGENTE REGIONAL)

Este perfil es el encargado de la operación diaria (ventas y atención primaria).

A. Registro de Siniestros (Transaccional) *Ruta: Menú "Siniestro"* Esta pantalla inicia una transacción crítica en la base de datos.

1. **Selección de Póliza:** Despliegue la lista "Póliza". El sistema solo mostrará pólizas vigentes o con siniestros previos en curso. Si intenta seleccionar una póliza vencida, esta no aparecerá o el sistema rechazará el envío.
2. **Datos del Incidente:**
    - **Tipo:** Ingrese la naturaleza del evento (Choque, Robo, etc.).
    - **Monto Estimado:** Ingrese el valor aproximado del daño.
    - **Agente:** Selecciónese a sí mismo o al agente responsable.
3. **Confirmación:** Al hacer clic en "Registrar Siniestro", el sistema valida que el monto no exceda la suma asegurada.
    - *Éxito:* Mensaje verde "Siniestro registrado exitosamente".
    - *Error:* Mensaje rojo indicando la causa (ej. "Póliza no vigente").

B. Registro de Pagos y Reactivación Automática *Ruta: Menú "Pago"* Funcionalidad inteligente para gestión de cobranza.

1. **Búsqueda:** Seleccione la póliza del cliente.
    - *Ayuda Visual:* Al seleccionar la póliza, el campo "Monto" se ilumina en verde y sugiere automáticamente el costo de la prima anual.
2. **Procesamiento:** Seleccione el método de pago y confirme.
3. **Resultado Automático:**
    - Si la póliza estaba **Vencida** y el pago cubre el 90% de la prima, el sistema mostrará: *"Pago registrado y vigencia actualizada"*, reactivando el contrato por un año más automáticamente.

C. Emisión de Nuevas Pólizas *Ruta: Menú "Póliza"*

1. Complete los datos del titular y cobertura.

2. **Fragmentación:** Al guardar, el sistema detectará el país del cliente y guardará la copia de seguridad en el nodo regional correspondiente (México o Internacional) de forma transparente para el usuario.

10.3. PERFIL DE CONTROL (GESTOR / ANALISTA)

Funciones avanzadas para la toma de decisiones y supervisión.

A. Gestión y Dictamen de Siniestros *Ruta: Menú "Gestión"* Permite desbloquear pólizas que tienen siniestros en curso.

1. **Bandeja de Entrada:** Seleccione un caso de la lista "Siniestro Pendiente". Solo aparecen casos con estatus "En revisión".
2. **Dictamen:**
    - **Aprobar y Pagar:** Autoriza el desembolso y libera la póliza para futuros trámites.
    - **Rechazar:** Cierra el caso sin pago, liberando también la póliza.
3. **Justificación:** Es obligatorio ingresar comentarios en el área de texto para la auditoría.

B. Cancelación de Pólizas *Ruta: Menú "Cancelar"*

1. Seleccione la póliza a dar de baja.
2. **Advertencia:** Esta acción es irreversible y genera una entrada en la bitácora de seguridad. Al confirmar, la póliza pasa a estado "Cancelada" inmediatamente y se bloquean futuros pagos o siniestros sobre ella.

10.4. PERFIL DE AUDITORÍA Y REPORTES

A. Tableros de Inteligencia (BI) *Ruta: Menú "Reportes"* Visualización de datos en tiempo real para la toma de decisiones estratégicas.

1. **Pestaña Alertas:** Revise la caja superior roja. Si existen pólizas próximas a vencer (30 días), aparecerán listadas ahí.
2. **Pestaña Avanzado:**
    - **Ranking:** Tabla de los 5 países con mayor volumen de ventas.
    - **Heatmap (Tabla Pivote):** Matriz que cruza *Países vs Meses*, permitiendo identificar estacionalidad en la siniestralidad (ej. picos de accidentes en Diciembre).

B. Bitácora de Auditoría (Trazabilidad) *Ruta: Menú "Auditoría" (Solo visible para Analistas de Riesgo)*

1. **Log de Cambios:** Muestra quién modificó qué dato y cuándo. Ideal para investigar fraudes o errores operativos.
2. **Log de Fallos:** Muestra errores técnicos capturados por el sistema (ej. intentos de violar reglas de negocio), útil para el equipo de TI.

CAPÍTULO 11: GLOSARIO TÉCNICO

ACID: (Atomicidad, Consistencia, Aislamiento, Durabilidad). Propiedades que garantizan la validez de una transacción.

PII: (Personally Identifiable Information). Datos que pueden identificar a una persona específica.

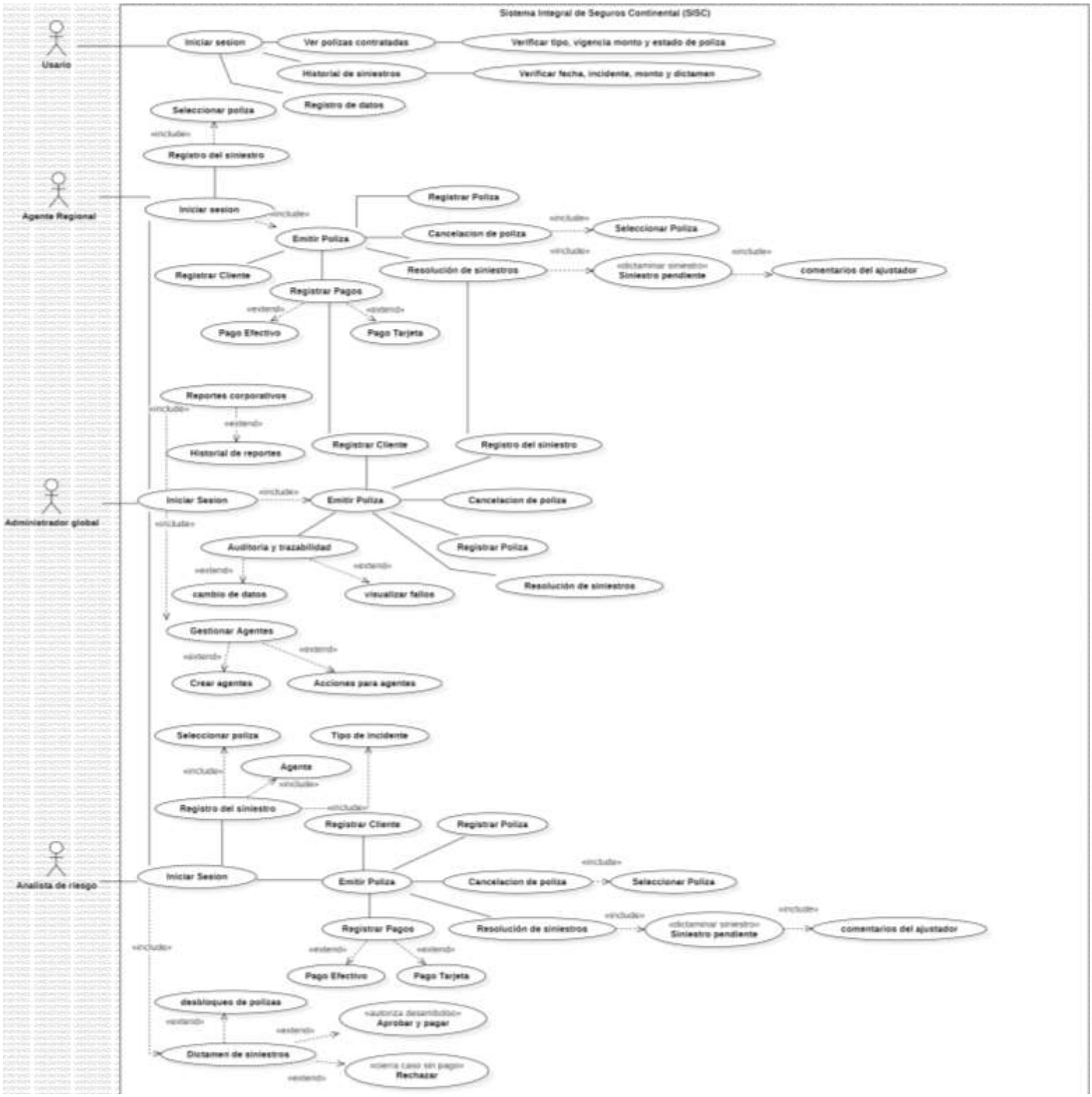
RBAC: (Role-Based Access Control). Método de restricción de acceso al sistema basado en los roles de los usuarios individuales.

Trigger: Procedimiento que se ejecuta automáticamente en respuesta a ciertos eventos en una tabla o vista.

Middleware: Software que actúa como puente entre un sistema operativo o base de datos y aplicaciones.

CAPÍTULO 12: ANEXOS

12.1 Diagrama de Casos de Uso



1. Descripción General del Sistema

El **Sistema Integral de Seguros Continental (SISC)** es una plataforma centralizada diseñada para la gestión del ciclo de vida de pólizas de seguro. El sistema interconecta a cuatro actores clave (Usuario, Agente Regional, Administrador Global y Analista de Riesgo) para garantizar la trazabilidad de la información, desde la contratación y cobranza hasta la dictaminación de siniestros.

2. Actores del Sistema

- **Usuario:** Cliente final o asegurado que interactúa con el sistema para consultas y reportes básicos.
- **Agente Regional:** Operador encargado de la gestión comercial, emisión de pólizas y primer contacto en siniestros.
- **Administrador Global:** Rol de supervisión encargado de la auditoría, gestión de usuarios, agentes y control de calidad de datos.
- **Analista de Riesgo:** Especialista encargado de la evaluación financiera, validación de incidentes y autorización de desembolsos.

3. Módulos Funcionales y Casos de Uso

A continuación, se describen las funciones agrupadas por el perfil de seguridad correspondiente.

3.1. Módulo de Autoservicio (Perfil: Usuario)

Este módulo permite al cliente visualizar el estado de sus servicios contratados.

- **Autenticación:** Acceso seguro mediante *Iniciar Sesión*.
- **Gestión de Pólizas:**
    - **Ver Pólizas Contratadas:** Permite verificar tipo de cobertura, vigencia, montos asegurados y estado actual de la póliza.
    - **Seleccionar Póliza:** Punto de partida para iniciar trámites sobre un contrato específico.
- **Gestión de Siniestros (Lado Cliente):**
    - **Historial de Siniestros:** Consulta de incidentes pasados, verificando fecha, montos y dictamen final.
    - **Registro de Siniestro:** El usuario puede reportar un incidente, lo cual dispara el *Registro de datos* inicial del evento.

3.2. Módulo de Operación Comercial (Perfil: Agente Regional)

Enfocado en la venta, administración de cartera y seguimiento operativo.

- **Emisión y Gestión de Pólizas:**
    - **Registrar Cliente:** Alta de nuevos asegurados en la base de datos.
    - **Emitir Póliza:** Proceso que *incluye* el registro formal de la póliza en el sistema.
    - **Cancelación de Póliza:** Permite dar de baja un contrato, requiriendo previamente *Seleccionar la Póliza* afectada.
- **Gestión Financiera (Cobranza):**
    - **Registrar Pagos:** Módulo flexible que se *extiende* para aceptar dos modalidades: **Pago en Efectivo** y **Pago con Tarjeta**.
- **Resolución Operativa de Siniestros:**
    - Gestión de incidentes que *incluye* la revisión de siniestros pendientes y la captura de **comentarios del ajustador** para nutrir el expediente.
- **Inteligencia de Negocio:**
    - **Reportes Corporativos:** Generación de informes que se *extiende* a la visualización del historial de reportes.

3.3. Perfil: Administrador Global

Enfocado en la integridad del sistema y la gestión de recursos humanos.

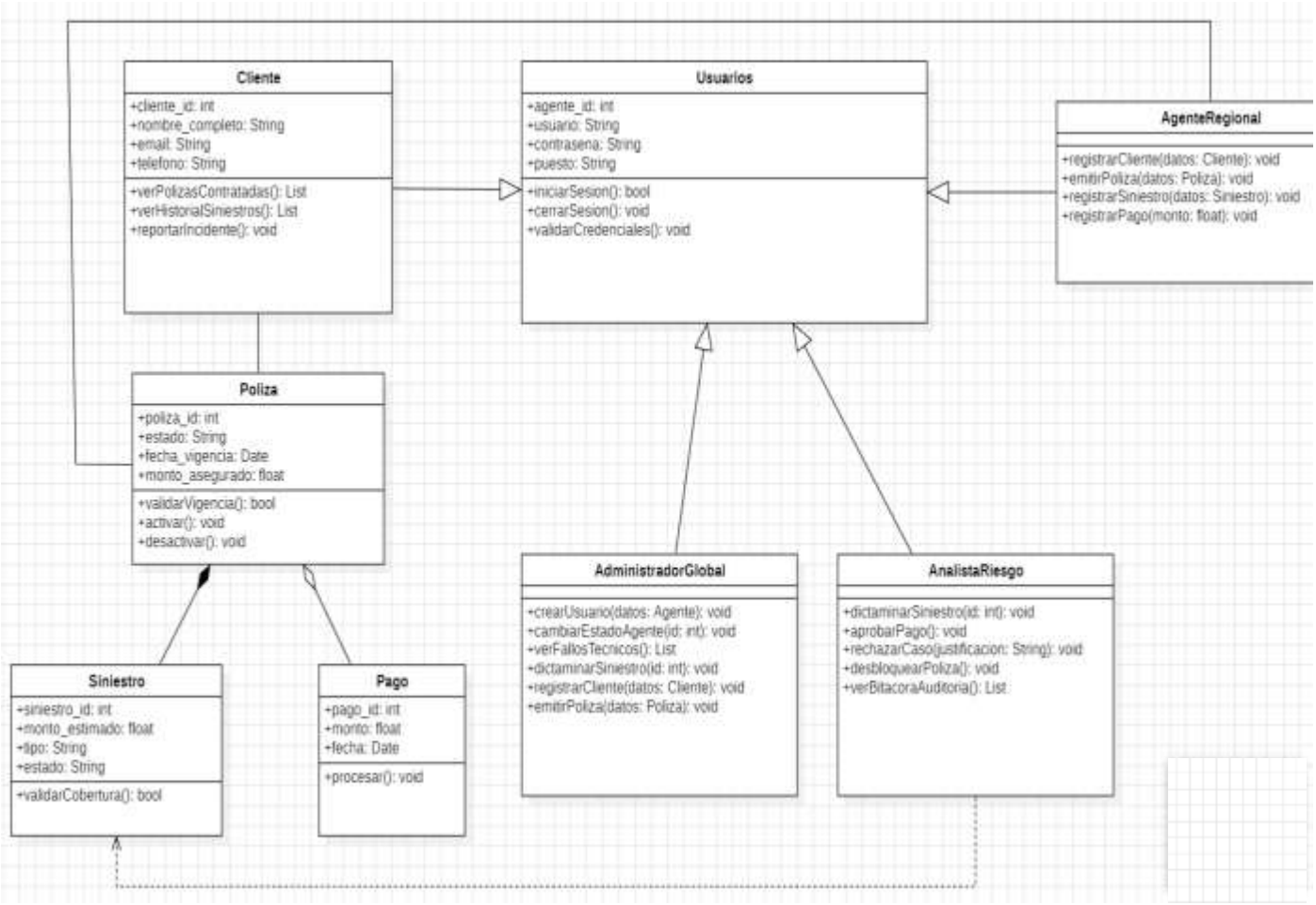
- **Auditoría y Trazabilidad:**
    - Monitorización de cambios de datos críticos.
    - **Visualización de fallos** del sistema para mantenimiento preventivo/correctivo.
- **(Agentes):**
    - **Crear Agentes:** Alta de nuevos operadores en el sistema.
    - **Acciones para Agentes:** Modificación de permisos o baja de personal.
- **Supervisión General:** El administrador mantiene capacidades redundantes para *Emitir Pólizas*, *Registrar Siniestros* y *Resolver Siniestros* en casos de escalamiento.

3.4. Perfil: Analista de Riesgo

Este es el módulo crítico para la rentabilidad de la aseguradora, donde se toman decisiones financieras.

- **Gestión Avanzada de Siniestros:**
    - **Selección y Tipificación:** Selección de póliza para definir el Tipo de Incidente y vincularlo con el Agente y Cliente correspondiente.
    - **Registro de Siniestro:** Validación técnica del reporte.
- **Dictamen y Flujo Financiero:**
    - **Dictamen de Siniestros:** Proceso de evaluación que deriva en dos resoluciones excluyentes:
        1. **Aprobar y Pagar:** *Autoriza el desembolso* económico al asegurado.
        2. **Rechazar:** *Cierra el caso sin pago* tras la evaluación negativa.
- **Control de Excepciones:**
    - **Desbloqueo de Pólizas:** Función administrativa para reactivar pólizas suspendidas por falta de pago o investigación.

12.2 Diagrama de Clases (Modelo Entidad-Relación)



12.3 Diagrama de Clases Modelo Entidad Relación

Descripción General

El diagrama de clases del **Sistema Integral de Seguros Continental (SISC)** representa la estructura estática del sistema bajo el paradigma de Programación Orientada a Objetos (POO). Este modelo define la arquitectura lógica de las entidades de negocio y los controladores de seguridad, garantizando la integridad referencial y la escalabilidad modular descrita en la arquitectura del sistema.

A continuación, se detallan los componentes principales y sus interacciones:

1. Jerarquía de Usuarios y Seguridad (Patrón de Herencia)

Para optimizar la reutilización de código y centralizar la autenticación, se implementó una **clase** denominada **Usuarios**.

- **Función:** Centraliza atributos comunes (usuario, contrasena, puesto) y métodos de seguridad (iniciarSesion, validarCredenciales).
- **Subclases (Roles):** Mediante una relación de **Generalización (Herencia)**, los actores específicos heredan la funcionalidad base pero implementan sus propios métodos operativos:
    - **AgenteRegional:** Hereda el acceso pero añade métodos de venta y operación como emitirPoliza() y registrarPago().
    - **AnalistaRiesgo:** Extiende la funcionalidad para tareas de auditoría y toma de decisiones financieras como dictaminarSiniestro().
    - **AdministradorGlobal:** Posee métodos exclusivos de gestión de infraestructura como crearUsuario() y verFallosTecnicos().

2. Núcleo del Negocio

Las clases de entidad representan las tablas transaccionales de la base de datos y contienen la lógica de negocio crítica.

- **Clase Poliza:** Es la entidad central del sistema. Contiene la lógica de estado (validarVigencia, activar) y actúa como el contenedor principal de la información del asegurado.
- **Clase Siniestro:** Representa los incidentes reportados. Incluye métodos de validación como validarCobertura() para asegurar que los montos no excedan la suma asegurada antes de ser procesados.
- **Clase Pago:** Encapsula las transacciones financieras. Su método procesar() dispara la lógica de renovación automática si se cumplen las condiciones de negocio (cubrir el 90% de la prima en pólizas vencidas).

3. Definición de Relaciones y Multiplicidad

Las conexiones entre clases se han diseñado siguiendo reglas estrictas de integridad de datos:

- **Composición (Póliza — Siniestro):** Se definió una relación de **Composición fuerte** entre Poliza y Siniestro.
    - *Justificación Técnica:* Un siniestro no puede existir de forma autónoma; depende existencialmente de un contrato de seguro. Si una póliza se elimina físicamente, sus siniestros asociados deben desaparecer para mantener la consistencia.
- **Agregación (Póliza — Pago):** Se estableció una relación de **Agregación** entre póliza y Pago.
    - *Justificación Técnica:* Aunque los pagos pertenecen a una póliza, son registros contables que pueden existir independientemente para fines de auditoría financiera o reportes históricos, incluso si la póliza cambia de estado.
- **Dependencia (Analista --> Siniestro):** Existe una relación de dependencia (línea punteada) donde el AnalistaRiesgo utiliza la clase Siniestro.
    - *Justificación Técnica:* El analista no "posee" el siniestro, pero depende de sus datos para ejecutar el método dictaminarSiniestro() y cambiar su estado.
- **Asociación (Cliente — Póliza):** Relación directa donde un Cliente puede tener múltiples instancias de Poliza (1 a *), permitiendo la gestión de cartera por usuario.

4. Patrones de Diseño Aplicados

- **Encapsulamiento:** Todos los atributos sensibles (como contrasena o monto_asegurado) se definen como privados o protegidos, exponiéndose solo a través de métodos públicos controlados.
- **Alta Cohesión:** Cada clase tiene una responsabilidad única (ej. AgenteRegional solo gestiona ventas, no audita fallos), cumpliendo con el principio de responsabilidad única.

CAPÍTULO 13: DOCUMENTACIÓN DETALLADA DEL CÓDIGO FUENTE

A continuación se presenta el desglose técnico, línea por línea, de los archivos que componen la solución, detallando la lógica, sintaxis y función de cada instrucción para garantizar la total comprensión del flujo del sistema.

13.1. BACKEND Y CONTROLADORES (PYTHON/FLASK)

Archivo: app.py *Este es el orquestador principal del sistema. Gestiona las rutas web, la seguridad de sesión y la exposición de la API REST.*

- **Líneas 1-5:** Importación de librerías. Flask (framework web), request (manejo de datos entrantes), session (manejo de cookies de usuario), DBConnector (nuestra clase personalizada para base de datos), wraps (para decoradores de seguridad) y decimal/datetime (para formateo de datos).
- **Línea 8:** app = Flask(__name__): Inicializa la instancia de la aplicación Flask.
- **Línea 9:** app.secret_key = ...: Define la llave criptográfica para firmar las cookies de sesión y evitar que sean manipuladas.
- **Línea 10:** db = DBConnector(): Crea una instancia global del conector a base de datos para ser usada en toda la app.
- **Líneas 14-26 (Función process_row_safe):** Función auxiliar para convertir filas de base de datos (tuplas) a diccionarios JSON.
  - **Línea 16:** Convierte la fila pyodbc a una lista mutable.
  - **Líneas 19-25:** Itera sobre cada columna. Si detecta tipos de datos complejos como Decimal (dinero) o datetime (fechas), los convierte a float o string respectivamente para evitar errores al generar el JSON.
- **Líneas 30-36:** Función de seguridad para proteger rutas.
  - **Línea 33:** Verifica si en la session existe la marca logged_in y si el rol del usuario es "Administrador Global".
  - **Línea 34:** Si no cumple, redirige al usuario al inicio (index), bloqueando el acceso.
- **Líneas 38-44:** Similar al anterior, pero valida específicamente que el rol sea "Analista de Riesgo".

- **Líneas 46-52:** before_request indica que esto se ejecuta antes de cualquier petición.
  - **Línea 47:** Define rutas excluidas de validación.
  - **Línea 49:** Si el usuario no está logueado, lo fuerza a ir a la página de Login.
  - **Líneas 50-52:** Si está logueado, carga los datos del usuario en la variable global g para usarlos en las vistas.
- **Líneas 56-72 (login):**
  - **Línea 58:** Si ya está logueado, lo manda al inicio.
  - **Línea 60:** Captura usuario y contraseña del formulario HTML.

    - **Línea 62:** Llama al SP sp_validar_agente en base de datos.
    - **Línea 63:** Valida si el SP retornó éxito (1 o True).
    - **Líneas 65-68:** Si es correcto, guarda los datos en session y redirige.
    - **Línea 70:** Si falla, recarga la página mostrando el error devuelto por la BD.
- **Líneas 74-77 (logout):** Limpia la sesión (session.clear()) cerrando el acceso y redirige al login.
- **Líneas 81-110 (Rutas de Vistas UI):**
    - Cada función (index, nuevo_cliente_ui, etc.) simplemente renderiza la plantilla HTML correspondiente (ej. render_template('registrar_poliza.html')), pasando el contexto del usuario actual.
- **Líneas 114-123 (API registrar_siniestro_api):**
    - **Línea 116:** Recibe los datos del formulario.
    - **Línea 117:** Valida en Python que el monto no sea negativo (capa extra de seguridad).
    - **Línea 118:** Prepara la tupla de parámetros para el SP.
    - **Línea 119:** Ejecuta sp_registrar_siniestro usando db.execute_sp.
    - **Línea 120:** Retorna un JSON con éxito (200) o error (400) según el resultado del SP.
- **Líneas 125-131 (API actualizar_siniestro_api):** Llama a sp_gestion_siniestro para aprobar/rechazar casos.
- **Líneas 133-139 (API crear_cliente_api):** Mapea los 7 campos del formulario al SP sp_crear_cliente.
- **Líneas 141-148 (API registrar_poliza_api):**
    - **Línea 143:** Valida que la prima anual no sea negativa.
    - **Línea 144:** Convierte los datos y llama a sp_crear_poliza.
- **Líneas 150-157 (API registrar_pago_api):**
    - **Línea 152:** Valida que el monto sea mayor a 0.
    - **Línea 153:** Ejecuta sp_registrar_pago, disparando la lógica de renovación automática en BD.
- **Líneas 159-165 (API cancelar_poliza_api):** Ejecuta la baja lógica mediante sp_cancelar_poliza.
- **Líneas 169-232 (APIs de Reportes y Listados):**
    - Estas rutas (`/api/reportes/...`, `/api/clientes/listar`, etc.) ejecutan SPs de lectura (`SELECT`) y usan `process_row_safe` para devolver los datos en formato JSON limpio para que el Frontend los dibuje en las tablas.
- **Líneas 236-267 (APIs de Admin):**
    - Rutas protegidas con @admin_required para crear agentes, resetear contraseñas o cambiar estados.
- **Líneas 269-285 (APIs de Auditoría):**
    - Rutas protegidas con @auditor_required para leer las tablas Auditoria_General y Auditoria_Fallos_Transaccion.
- **Línea 287:** app.run(debug=True): Arranca el servidor web en modo desarrollo.

Archivo: db_connector.py *Módulo encargado de la abstracción de la conexión a SQL Server.*

- **Líneas 1-3:** Importa pyodbc y la clase Config.
- **Línea 6:** Clase DBConnector.
- **Líneas 10-15 (get_connection):** Intenta establecer conexión usando el DSN definido en configuración. Si falla, captura la excepción y la imprime en consola.
- **Líneas 17-64 (Método execute_sp):** Método central para ejecutar Procedimientos Almacenados.
    - **Línea 26:** Crea un cursor para ejecutar comandos.
    - **Líneas 27-30:** Construye dinámicamente la query SQL (ej: {CALL sp_nombre(?, ?)}) basándose en la cantidad de parámetros recibidos.
    - **Línea 32:** fetch_one=True se usa para SPs transaccionales que devuelven un solo mensaje de estado.
    - **Líneas 34-40:** fetch_one=False se usa para reportes. Itera sobre los resultados incluyendo nextset para manejar múltiples tablas si el SP devuelve más de una.
    - **Línea 42:** conn.commit(): **CRÍTICO**. Confirma la transacción en la base de datos.
    - **Líneas 46-60 (Manejo de Errores):**
        - Si el SP falla (entra al CATCH de SQL), Python captura el error pyodbc.Error.
        - **Línea 57:** Ejecuta un conn.commit() explícito incluso en el error. **Esto es esencial** para asegurar que el registro insertado en Auditoria_Fallos_Transaccion dentro de la BD se guarde y no se pierda al cerrarse la conexión.

Archivo: config.py

- **Línea 4:** Define SERVER_NAME. Debe apuntar a la instancia local (ej. DAVIDMOSCO\SQLEXPRESS).
- **Línea 6:** Construye la cadena de conexión DSN usando autenticación de Windows (Trusted_Connection=yes) para mayor seguridad (evita hardcodear contraseñas en el código).

13.2. BASE DE DATOS Y LÓGICA SQL

Archivo: Scrip final base de datos Proyecto.sql

- **Línea 19:** CREATE DATABASE CorporateInsuranceDB: Crea el contenedor de datos.
- **Líneas 28-38 (Tabla Clientes):** Estructura original. Define email y numero_documento como UNIQUE para evitar duplicados.
- **Líneas 41-50 (Tabla Agentes):** Almacena usuarios, usuario es único. estado controla el acceso lógico.
- **Líneas 53-64 (Tabla Polizas):** Tabla transaccional principal. Relaciona con Clientes mediante FK.
- **Líneas 68-84 (Tablas Polizas_Nodo_Mexico e Internacional):** Tablas espejo para simular la **fragmentación horizontal**. Reciben copias de datos según la región.
- **Líneas 87-98 (Tabla Siniestros):** Relaciona con Pólizas y Agentes.
- **Líneas 101-108 (Tabla Pagos):** Registra flujo de efectivo.
- **Líneas 111-125 (Tablas de Auditoría):**
  - Auditoria_General: Bitácora de operaciones de negocio.
  - Auditoria_Fallos_Transaccion: Bitácora técnica para errores TRY/CATCH.
- **Líneas 130-132 (Índices):** Optimización. Crea índices en columnas muy consultadas como estado_siniestro y fecha_reporte para acelerar el Dashboard.
- **Líneas 138-151 (Vista vw_Alertas_Vencimiento):** Consulta precompilada que calcula DATEDIFF (días restantes) para detectar pólizas por vencer en <30 días.
- **Línea 168 (sp_validar_agente):** SP de Login. Compara el hash de contraseña y verifica que el agente esté "Activo".
- **Líneas 180-190 (sp_admin_crear_agente):** Usa TRY/CATCH para evitar crear usuarios duplicados.
- **Líneas 193-222 (sp_registrar_siniestro - Transaccional):**
  - **Línea 199:** BEGIN TRANSACTION. Inicia bloque atómico.
  - **Líneas 201-207:** Validaciones de negocio (Estado de póliza). Usa THROW para abortar si la póliza no es válida.
  - **Línea 210:** Inserta el siniestro.
  - **Línea 212:** Actualiza el estado de la póliza a "Con Siniestro Reportado".
  - **Línea 213:** COMMIT. Confirma cambios.
  - **Líneas 216-220 (CATCH):** Si algo falla, hace ROLLBACK y guarda el error en Auditoria_Fallos_Transaccion.
- **Líneas 226-267 (sp_registrar_pago - Lógica Compleja):**
  - **Líneas 249-261: Automatización**. Si el pago es suficiente (>90%), verifica el estado actual. Si es "Vencida", la reactiva y renueva fechas. Si es "Activa", extiende la vigencia un año más.

- **Líneas 337-353 (trg_Validar_Vigencia_Siniestro):** Trigger INSTEAD OF INSERT. Intercepta el registro de siniestros. Si la póliza no es válida, bloquea la inserción y lanza error 16 (Grave), protegiendo la integridad de datos a bajo nivel.
- **Líneas 357-378 (trg_Replicacion_Fragmentacion):** Trigger AFTER INSERT. Después de crear una póliza, detecta la región (Mexico vs Otro) e inserta una copia en la tabla de nodo correspondiente (Polizas_Nodo_Mexico o Internacional), simulando replicación distribuida.

Archivo: Scrip datos masivos.sql

- **Sección 1:** Inserta 50 clientes "famosos" manualmente para tener datos de prueba reconocibles.
- **Sección 2 (Líneas 60-78):** Script procedural (WHILE loop) que genera 70 pólizas. Usa RAND() para asignar primas, fechas y estados aleatorios, vinculándolas a clientes al azar (ORDER BY NEWID()).
- **Sección 3:** Genera siniestros aleatorios pero **solo** para pólizas activas (manteniendo integridad referencial).

13.3. FRONTEND E INTERFAZ DE USUARIO (HTML5 + JAVASCRIPT)

Esta capa implementa la lógica de presentación. Se utiliza **Jinja2** (motor de plantillas de Flask) para el renderizado dinámico del lado del servidor (SSR) en la carga inicial, y **JavaScript (Fetch API)** para la interactividad asíncrona sin recargas (SPA behavior).

Archivo: login.html *Punto de entrada al sistema. Gestiona la autenticación inicial.*

- **Líneas 8-22 (CSS):** Define las variables de color (:root) y la animación fadeIn para que el formulario aparezca suavemente.
- **Líneas 25-27:** Contenedor principal con efecto "Glassmorphism" (fondo semitransparente).
- **Líneas 29-31 (Jinja Logic):** Bloque condicional de Flask. Si el Backend devuelve un error de credenciales, inyecta un div rojo con el mensaje; si no, este código HTML no se genera.
- **Línea 33:** El formulario envía los datos vía POST a la ruta /login de Python explicada en la sección 13.1.

Archivo: **registrar_siniestro.html** *Interfaz crítica para el registro transaccional de incidentes.*

- **Líneas 33-54 (Barra de Navegación Dinámica):**
  - **Línea 35:** Inyecta el nombre del agente logueado.
  - **Líneas 47-53:** Bloques Solo muestra los enlaces "Admin" o "Auditoría" si el rol del usuario en sesión coincide, ocultando opciones no autorizadas a nivel visual.
- **Líneas 73-82 (Formulario):** Define los inputs. Nota que los <select> inician vacíos porque se llenarán dinámicamente con JS.
- **Líneas 91-115 (Script: Carga de Datos):**
  - **Línea 91:** Función autoejecutable async que corre al abrir la página.
  - **Línea 93:** Solicita al Backend la lista de pólizas en formato JSON.
  - **Líneas 98-102:** Filtra en el navegador para mostrar solo pólizas 'Activas' o 'Con Siniestro'.
  - **Línea 103:** Transforma cada objeto póliza en una etiqueta HTML <option>, mostrando ID, Cliente y Tipo para facilitar la selección.
  - **Línea 108:** Carga similar para obtener la lista de agentes activos desde /api/agentes/listar_activos.
- **Líneas 117-129 (Script: Envío del Formulario):**
  - **Línea 117:** Intercepta el botón "Guardar".
  - **Línea 118:** Evita que la página se recargue (comportamiento clásico).
  - **Línea 121:** Empaqueta los datos y los envía al endpoint /api/siniestro/registrar.
  - **Línea 124:** Recibe la respuesta JSON. Si success es true, muestra mensaje verde y recarga la página en 2 segundos.

Archivo: **registrar_pago.html** *Interfaz para cobranza y reactivación automática de pólizas.*

- **Líneas 85-87:** Campos de entrada. El campo monto tiene un placeholder dinámico.
- **Líneas 100-116 (Script: Lógica de Interfaz):**
  - **Línea 109:** Guarda el valor de la prima_anual dentro de un atributo de datos (dataset.monto) en cada opción del <select>.
  - **Líneas 118-125:** selectPoliza.addEventListener('change').
  - **Línea 121:** Cuando el agente selecciona una póliza, el sistema extrae la prima anual del dataset y pre-llena automáticamente el campo de monto, sugiriendo el pago completo.
  - **Línea 122:** Añade un efecto visual (borde verde) para indicar al usuario que el monto fue sugerido.

Archivo: auditoria.html *Dashboard de seguridad para el Analista de Riesgo.*

- **Líneas 62-65 (Tabs):** Botones HTML que llaman a la función openTab().
- **Líneas 82-90 (Función openTab):** Lógica de pestañas. Oculta todos los div con clase .tab-content y muestra solo el seleccionado por ID. Si se abre la pestaña 'Cambios', dispara cargarCambios().
- **Línea 93 (Función formatearFecha):** Convierte fechas ISO de SQL a formato local legible ("25/11/2025 10:00").
- **Líneas 98-115 (Función cargarCambios):**
    - **Línea 101:** Consulta la bitácora inmutable.
    - **Líneas 105-112:** Genera filas de tabla (<tr>) dinámicamente.
    - **Línea 107:** Colorea el nombre de la tabla afectada para rápida identificación visual.
- **Líneas 117-133 (Función cargarFallos):** Similar a la anterior, pero consume /api/auditoria/fallos. Muestra los errores capturados por los bloques TRY/CATCH del SQL Server.

Archivo: reporte_siniestros.html *Centro de inteligencia de negocios (BI) y alertas.*

- **Líneas 95-98 (Alertas):** Contenedor div id="alertas-vencimiento" oculto por defecto (display: none en CSS).
- **Líneas 147-157 (Script: cargarAlertasVencimiento):**
    - **Línea 148:** Consulta la vista de base de datos a través de la API.
    - **Línea 151:** Si hay datos (array length > 0), cambia el estilo a display: block (visible).
    - **Línea 153:** Inyecta HTML mostrando cuántos días faltan para que la póliza expire.
- **Líneas 168-185 (Filtros en Cascada):**
    - **Línea 168:** Escucha cambios en el select de cliente.
    - **Línea 175:** fetch(/api/polizas/listar?cliente_id=${clienteId}). Hace una petición al servidor pidiendo **solo** las pólizas de ese cliente específico.
    - **Línea 178:** Reconstruye el segundo <select> (Pólizas) con los datos recibidos. Esto evita cargar miles de pólizas innecesarias (Optimización).
- **Líneas 207-212 (cargarReporteAvanzado):**
    - **Línea 210:** Renderiza la tabla de Ranking de Países.
    - **Línea 211:** Renderiza la tabla Pivote (Heatmap mensual). Mapea dinámicamente las columnas de los meses (Enero-Diciembre) usando las llaves del JSON devuelto por el SP sp_reporte_avanzado_corporativo.

Archivo: **admin_agentes.html** *Panel de administración de usuarios (ABM).*

- **Líneas 119-145 (Script: cargarAgentes):**
  - **Línea 125:** Lógica ternaria para determinar el color y texto del botón. Si el agente está 'Activo', el botón será Rojo ('Desactivar'); si no, Verde ('Activar').
  - **Líneas 134-135:** Inyecta botones con eventos onclick que pasan el ID del agente a las funciones cambiarEstado() y resetPass().
- **Líneas 155-159 (Función cambiarEstado):**
  - **Línea 157:** fetch(..., method: 'POST'). Envía un JSON {agente_id, nuevo_estado} al endpoint de actualización. Al finalizar, recarga la tabla automáticamente (cargarAgentes()) para reflejar el cambio sin recargar la página completa.

14.DESCRIPCIÓN FUNCIONAL DE MÓDULOS WEB

1. PESTAÑA: INICIO

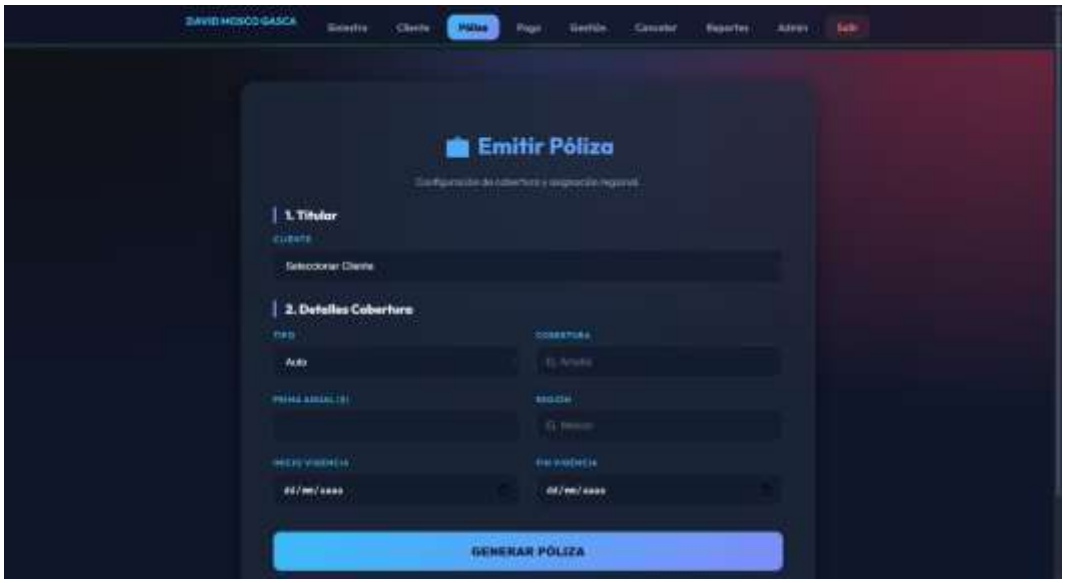
- **Propósito:** Panel de aterrizaje y centro de notificaciones personalizadas.
- **Funcionalidad:**
    - **Validación de Sesión:** Al cargar, el sistema verifica la integridad del token de sesión. Si la sesión ha expirado o es inválida, redirige forzosamente al Login.
    - **Personalización:** Renderiza un saludo dinámico extrayendo el nombre del agente desde la variable de sesión segura, confirmando visualmente la identidad del usuario activo.
    - **Navegación Rápida:** Provee accesos directos a los módulos más utilizados según el rol detectado (Operativo vs. Administrativo).

2. PESTAÑA: SINIESTRO

- **Propósito:** Captura y alta de reclamos o accidentes en tiempo real.
- **Funcionalidad y Lógica:**
    - **Filtrado Inteligente de Pólizas:** El selector de pólizas no muestra todas las opciones; ejecuta una consulta asíncrona (AJAX) que filtra y despliega únicamente contratos con estatus "Vigente" o con siniestros previos abiertos, previniendo errores humanos de selección.
    - **Validación de Cobertura (Backend):** Al enviar el formulario, el sistema compara el Monto Estimado contra la Suma Asegurada en la base de datos.
        - *Escenario de Éxito:* Se genera un folio único y se actualiza el estado de la póliza a "Con Siniestro".
        - *Escenario de Bloqueo:* Si el monto excede la cobertura o la póliza está vencida, el motor SQL rechaza la transacción y la interfaz muestra una alerta crítica sin recargar la página.

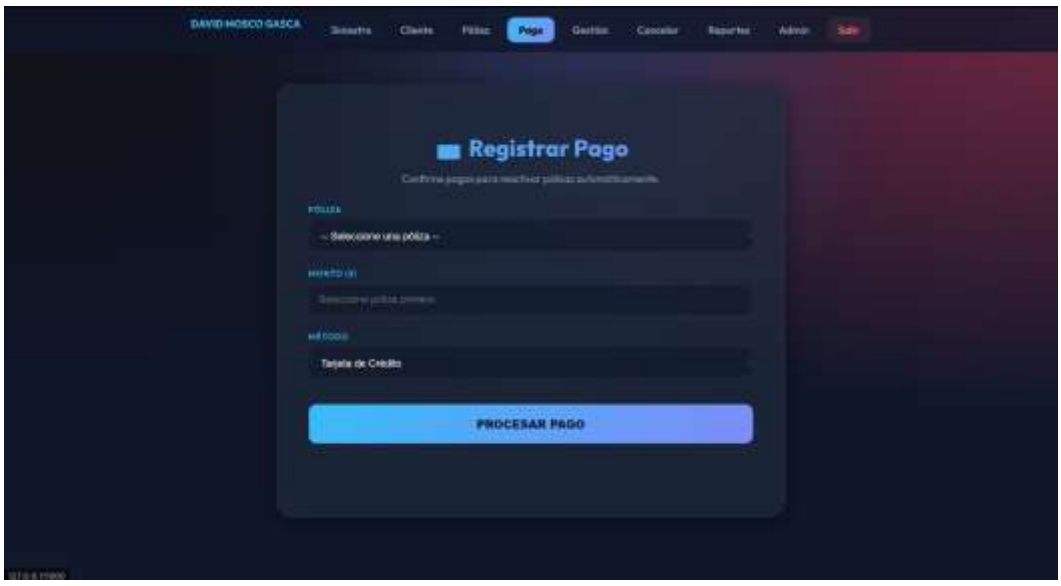
3. PESTAÑA: PÓLIZA (EMISIÓN Y FRAGMENTACIÓN)

- **Propósito:** "Onboarding" de nuevos clientes y formalización de contratos.
- **Funcionalidad y Lógica:**
    - **Captura Unificada:** Permite ingresar datos demográficos del cliente y condiciones financieras (Prima, Suma Asegurada) en un solo flujo.
    - **Distribución Transparente (Fragmentación):** Aunque el usuario interactúa con un solo formulario, el backend analiza el campo "País". Si el asegurado es extranjero, el sistema enruta automáticamente el almacenamiento de datos hacia el nodo Internacional y los datos sensibles hacia la tabla segura Clientes_Sensible, cumpliendo normativas de privacidad sin intervención del agente.



4. PESTAÑA: PAGO

- **Propósito:** Gestión de tesorería y reactivación de servicios suspendidos.
- **Funcionalidad y Lógica:**
    - **Asistencia de Cobro:** Al seleccionar una póliza, el sistema recupera la Prima Anual pactada y pre-llena el campo de monto, resaltándolo con un indicador visual verde para sugerir el pago total.
    - **Motor de Reactivación:** Si se procesa un pago para una póliza con estatus "Vencida" y el monto cubre el adeudo mínimo configurado (90%), el sistema dispara un procedimiento almacenado que:
        1. Registra el ingreso financiero.
        2. Cambia el estatus de la póliza a "Activa".
        3. Extiende la fecha de vigencia por 12 meses automáticamente.

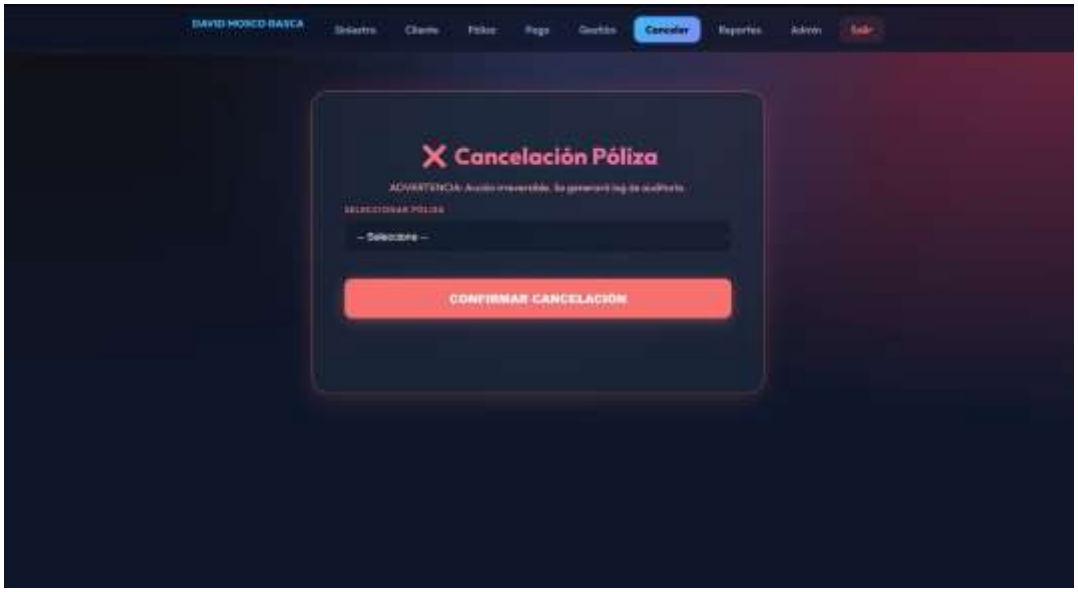


5. PESTAÑA: GESTIÓN (DICTAMEN Y AUTORIZACIÓN)

- **Propósito:** Bandeja de entrada para la supervisión y resolución de siniestros.
- **Funcionalidad:**
    - **Filtrado de Estado:** La tabla muestra exclusivamente siniestros con estatus "En Revisión".
    - **Flujo de Aprobación:**
        - **Aprobar:** Autoriza la dispersión de fondos y libera la póliza para futuros trámites.
        - **Rechazar:** Cierra el caso sin impacto financiero, liberando la póliza.
    - **Auditoría de Decisión:** Es obligatorio ingresar una justificación en el campo de texto antes de procesar el dictamen, asegurando trazabilidad para futuras auditorías.

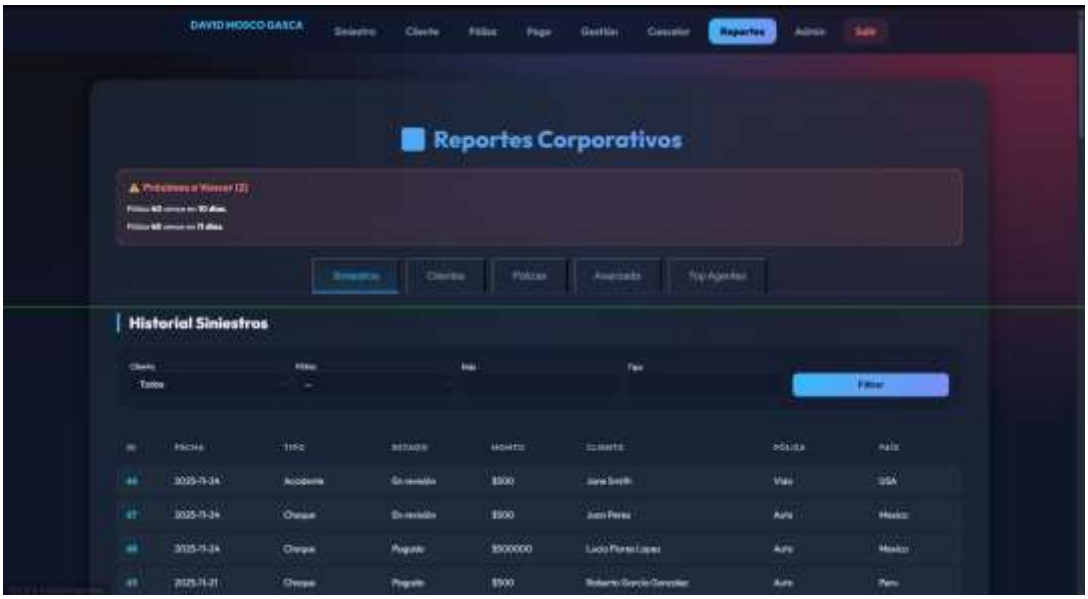
6. PESTAÑA: CANCELAR (BAJA DE SERVICIOS)

- **Propósito:** Terminación anticipada de contratos de seguro.
- **Funcionalidad:**
    - **Baja Lógica (Soft Delete):** Al confirmar la cancelación, el sistema **no elimina** físicamente el registro de la base de datos (para preservar la historia). En su lugar, actualiza el estatus a "Cancelada", bloqueando inmediatamente la posibilidad de registrar nuevos siniestros o pagos sobre dicha póliza.



7. PESTAÑA: REPORTES (INTELIGENCIA DE NEGOCIOS - BI)

- **Propósito:** Visualización de KPIs y monitoreo de riesgos.
- **Funcionalidad:**
    - **Alertas Preventivas:** Un módulo oculto consulta en tiempo real la base de datos. Si detecta pólizas próximas a vencer (ventana de 30 días), despliega automáticamente un contenedor de alertas rojas en la cabecera.
    - **Análisis de Estacionalidad (Heatmap):** Genera una tabla pivote dinámica cruzando *Regiones Geográficas* vs. *Meses del Año*, permitiendo identificar patrones de siniestralidad (ej. aumento de accidentes en diciembre en la zona norte).



DESCRIPCIÓN DEL MÓDULO: ADMINISTRADOR GLOBAL

*(Visible exclusivamente para usuarios con rol 'Administrador Global' en la pestaña **ADMIN**)*

Este módulo crítico permite la gestión integral del ciclo de vida de los usuarios (Agentes y Auditores) y el control de acceso al sistema (RBAC).

A. Alta de Usuarios (Provisioning)

- **Descripción:** Formulario para la creación de nuevas credenciales de acceso.
- **Lógica de Negocio:**
    - Permite asignar roles específicos (Agente Regional, Analista de Riesgo, Administrador Global) que determinarán qué pestañas podrá ver el nuevo usuario.
    - Valida que el nombre de usuario no exista previamente en la base de datos para evitar duplicidad de identidades.

B. Control de Estado (Activación/Bloqueo)

- **Descripción:** Mecanismo de control de acceso inmediato.
- **Funcionalidad:**
    - En la tabla de agentes, cada fila presenta un botón de estado dinámico.
    - **Lógica Toggle:**
        - Si el agente está **Activo**, el botón se muestra Rojo con el texto "Desactivar". Al pulsarlo, revoca el acceso al sistema inmediatamente sin borrar al usuario.
        - Si el agente está **Inactivo**, el botón se muestra Verde con el texto "Activar", restaurando el acceso.
    - Esta acción se ejecuta mediante una petición asíncrona, actualizando la interfaz sin recargar la página.

C. Restablecimiento de Seguridad (Password Reset)

- **Descripción:** Herramienta de soporte para recuperación de cuentas.
- **Funcionalidad:**
    - Permite al Administrador forzar el cambio de contraseña de un agente a un valor temporal predeterminado. Esto es esencial cuando un usuario olvida sus credenciales o se sospecha que su cuenta ha sido comprometida.

DESCRIPCIÓN DEL MÓDULO: AUDITORÍA

(Visible exclusivamente para usuarios con rol 'Analista de Riesgo' o Auditores)

- **Propósito:** Garantizar el No-Repudio y la trazabilidad técnica.
- **Vistas Disponibles:**
    1. **Bitácora de Operaciones:** Despliega un historial inmutable de cambios en los datos (Data Lineage). Muestra el "Valor Anterior" y "Valor Nuevo" de cada campo modificado, identificando al usuario responsable y la fecha exacta (Timestamp).
    2. **Bitácora de Fallos Técnicos:** Visualiza las excepciones capturadas por los bloques TRY/CATCH del motor de base de datos, permitiendo al equipo de TI diagnosticar intentos de violación de reglas de negocio o errores de infraestructura.