

## **Desarrollo documento de arquitectura DAS**

**MODALIDAD DE ESTUDIOS:**

**Presencial**

**FACULTAD/UNIDAD ACADÉMICA:**

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN Y ELECTRÓNICA

**NOMBRE DE LA CARRERA:**

INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN

**NOMBRES COMPLETOS DE LOS ESTUDIANTES:**

Jean Daniel Villavicencio Samaniego

Sebastián Felipe Mendieta Lima

Orly André Flores Valdiviezo

Alex Fernando Aguirre Rojas

**FECHA:** 3 de Diciembre del 2025

**NOMBRE DEL TUTOR ACADÉMICO:** Bustamante Granda Wayner Xavier

**PERIODO ACADÉMICO:** Octubre 2025 – Febrero 2026

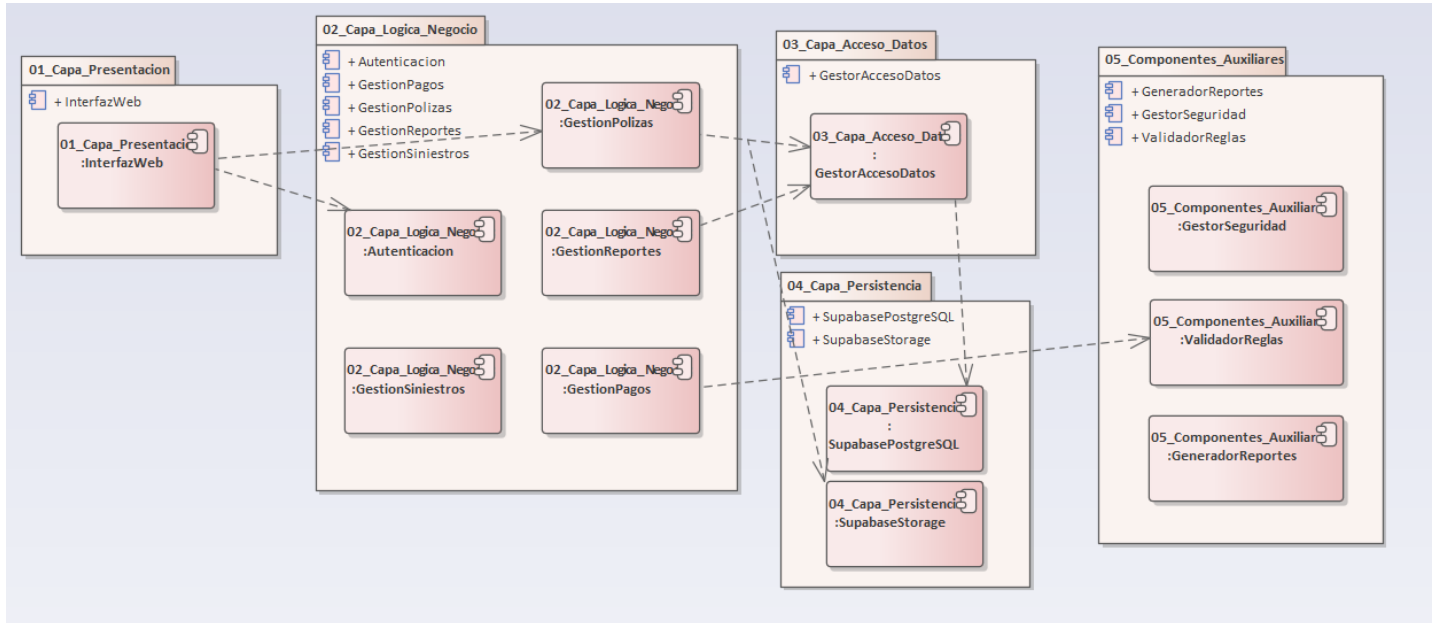
# Sistema de Gestión de Pólizas de Vida - UTPL

## Diseño Arquitectónico

### 1. RESUMEN

#### 1.1. Diagramas de la Arquitectura

Diagrama de Arquitectura de Componentes - Sistema de Seguros UTPL



**Nota:** El diagrama muestra la arquitectura en capas con los componentes principales del sistema y sus interdependencias.

## 1.2. Descripción de los Componentes

Componente	Descripción
Interfaz Web	Aplicación web desarrollada en React que proporciona la interfaz de usuario para todos los actores del sistema.
Gestión de Pólizas	Componente que gestiona el ciclo de vida completo de las pólizas: registro, actualización, renovación y cancelación.
Gestión de Pagos	Gestiona el procesamiento de pagos mensuales, coaseguros y copagos de las pólizas activas.
Gestión de Siniestros	Administra el reporte, seguimiento y resolución de siniestros, incluyendo la carga de evidencias y aprobación.
Gestión de Reportes	Genera reportes estadísticos y analíticos sobre pólizas, pagos, siniestros y estado del sistema.
Seguridad y Autenticación	Gestiona la autenticación de usuarios, autorización basada en roles y seguridad de las sesiones.
Validación de Datos	Valida todos los datos de entrada según las reglas de negocio definidas.
Notificaciones	Envía notificaciones por correo electrónico sobre eventos importantes del sistema.
Gestión de Archivos	Administra el almacenamiento y recuperación de archivos adjuntos como evidencias de siniestros.
Capa de Acceso a Datos	Proporciona acceso a la base de datos mediante el patrón DAO, abstrayendo las operaciones CRUD.

## 1.3. Interdependencias de los componentes

Componente1	Componente2	Interdependencia
Interfaz Web	Gestión de Pólizas	La interfaz consume servicios REST para operaciones CRUD de pólizas

Componente1	Componente2	Interdependencia
Gestión de Pólizas	Capa de Acceso a Datos	Utiliza DAOs para persistir información de pólizas en la base de datos
Gestión de Siniestros	Gestión de Archivos	Requiere almacenar evidencias adjuntas a los reportes de siniestros
Gestión de Pagos	Validación de Datos	Valida montos, fechas y estado de póliza antes de procesar pagos
Gestión de Reportes	Capa de Acceso a Datos	Consulta datos agregados de múltiples entidades para generar reportes

## Historia de Revisiones

Fecha	Versión	Descripción	Autor
03/12/2024	1.0	Versión inicial de arquitectura de componentes	Equipo UTPL

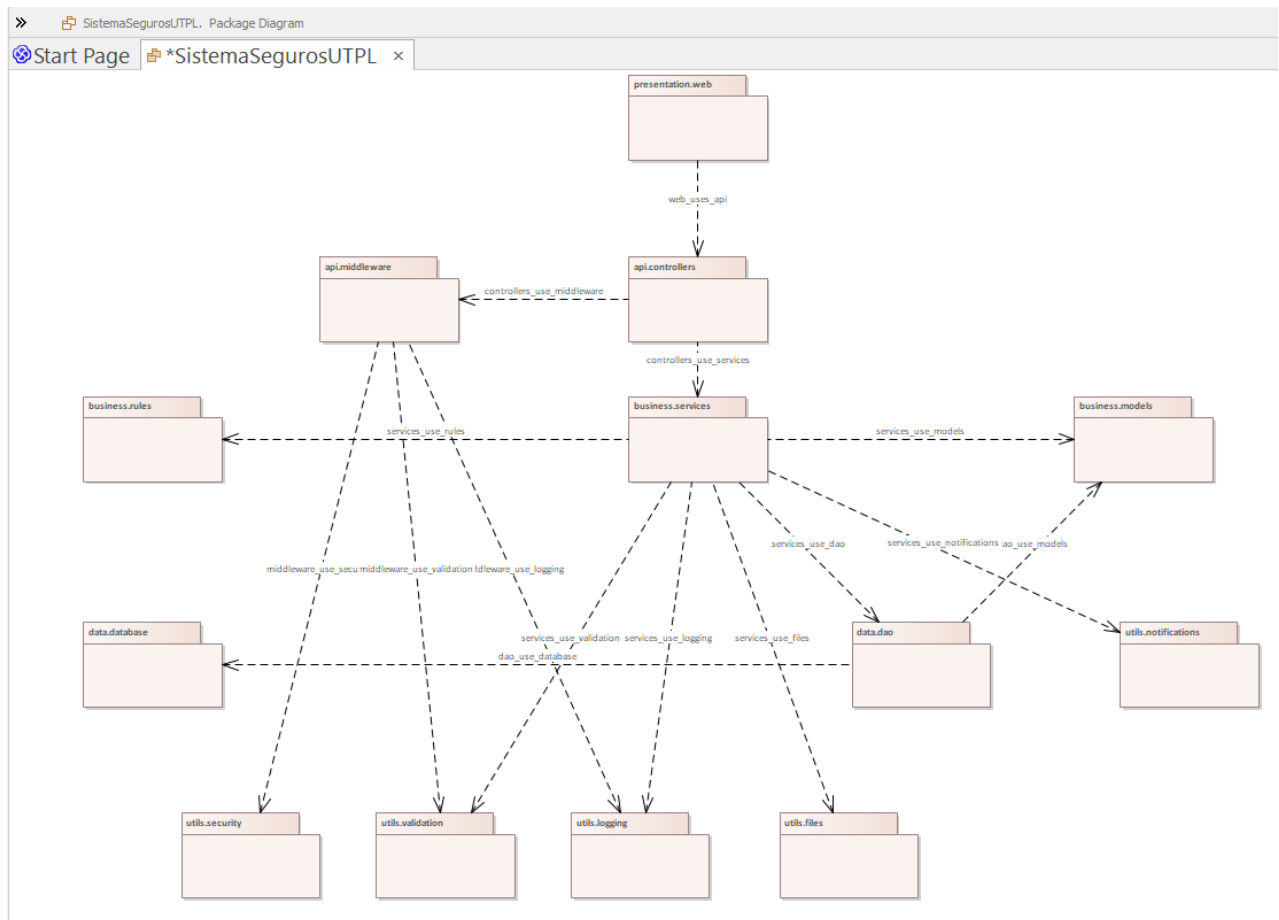
# **Sistema de Gestión de Pólizas de Vida - UTPL**

## **Diseño Arquitectónico**

### **1. RESUMEN**

#### **1.1. Diagramas de la Arquitectura**

Diagrama de Arquitectura de Paquetes - Sistema de Seguros UTPL



**Nota:** El diagrama muestra la organización en paquetes y módulos del sistema siguiendo el patrón de arquitectura por capas.

## 1.2. Descripción de los módulos

Módulo	Descripción
presentation.web	Paquete que contiene los componentes React de la interfaz web: páginas, formularios, vistas y navegación.
business.services	Contiene la lógica de negocio del sistema: servicios de pólizas, pagos, siniestros y reportes.
business.rules	Implementa las reglas de negocio definidas: validaciones de fechas, montos, estados de póliza y elegibilidad.
business.models	Define las entidades del dominio: Póliza, Pago, Siniestro, Usuario, etc.
data.dao	Capa de acceso a datos con objetos DAO para cada entidad, implementando operaciones CRUD.
data.database	Gestiona la conexión a la base de datos PostgreSQL mediante Supabase SDK.

Módulo	Descripción
utils.validation	Utilidades para validación de datos: formato de fechas, cédulas, emails y campos requeridos.
utils.security	Funciones de seguridad: encriptación, gestión de tokens JWT y control de acceso basado en roles.
utils.notifications	Servicios de notificaciones por correo electrónico y mensajes del sistema.
utils.files	Gestión de archivos adjuntos: carga, descarga y almacenamiento en Supabase Storage.
utils.logging	Sistema de logging para auditoría y seguimiento de operaciones del sistema.
api.controllers	Controladores REST API que exponen los endpoints del sistema al frontend.
api.middleware	Middleware para autenticación, validación de requests y manejo de errores HTTP.

### 1.3. Interdependencias de los módulos

Módulo1	Módulo2	Interdependencia
presentation.web	api.controllers	La interfaz consume los endpoints REST expuestos por los controladores
api.controllers	business.services	Los controladores invocan los servicios de la capa de negocio
business.services	business.rules	Los servicios aplican las reglas de negocio antes de operar
business.services	data.dao	Los servicios utilizan los DAOs para persistir datos
data.dao	data.database	Los DAOs acceden a la base de datos mediante el módulo de conexión
api.controllers	api.middleware	Los controllers pasan por el middleware de autenticación y validación

## Historia de Revisiones

Fecha	Versión	Descripción	Autor
03/12/2024	1.0	Versión inicial de arquitectura de paquetes	Equipo UTPL

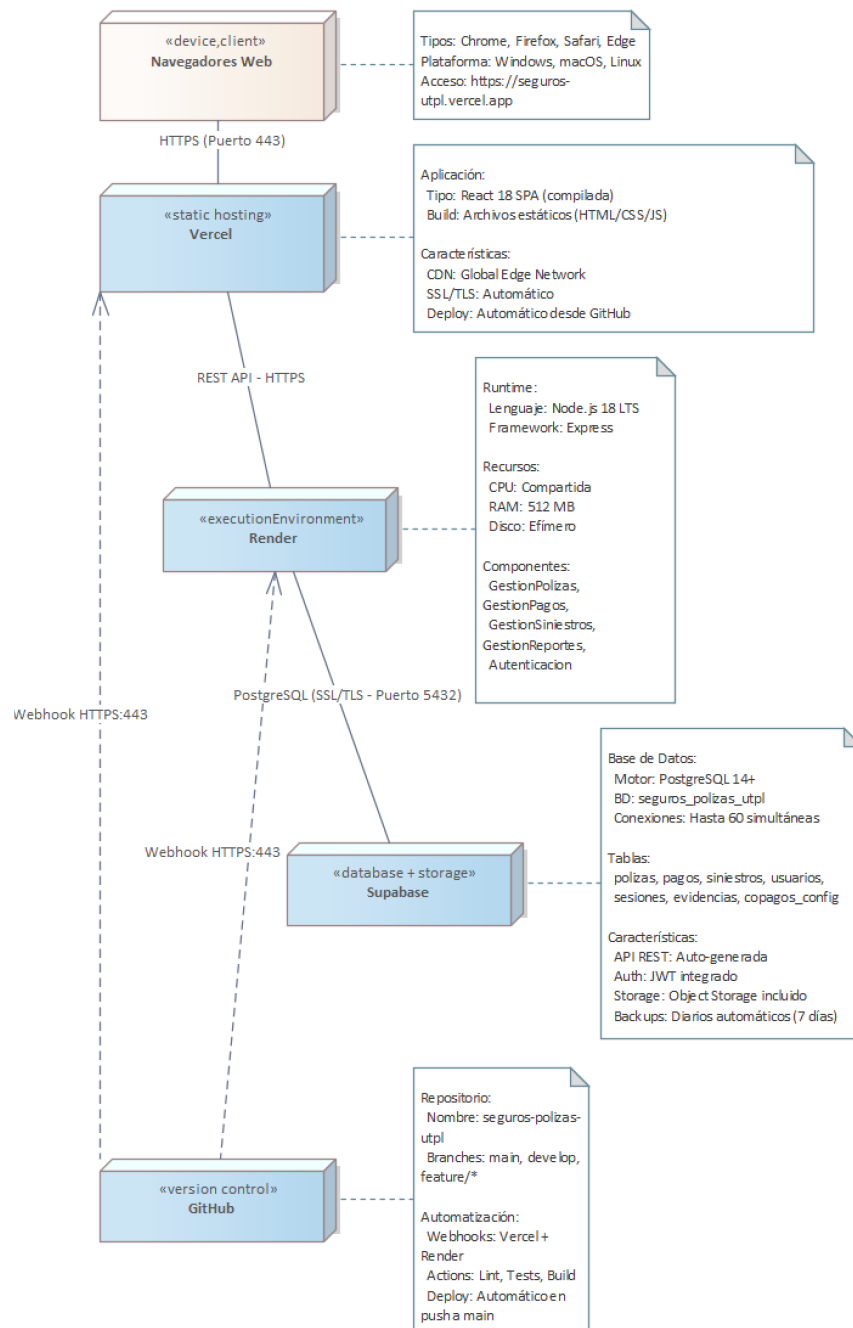
# **Sistema de Gestión de Pólizas de Vida - UTPL**

## **Diseño de Despliegue**

### **1. RESUMEN**

#### **1.1. Diagramas de Despliegue**

Diagrama de Arquitectura de Despliegue - Sistema de Seguros UTPL



**Nota:** El diagrama muestra la infraestructura cloud simplificada del sistema desplegado en plataformas gratuitas.

## 1.2. Descripción de los Nodos

Nodo	Descripción
Cliente Web	Navegador web (Chrome, Firefox, Edge, Safari) que ejecuta la aplicación React del usuario. Compatible con Windows, Mac y Linux.
Vercel CDN	Plataforma de hosting para el frontend React. Distribuye contenido estático globalmente mediante CDN. Tier gratuito: despliegues ilimitados, dominio personalizado, SSL automático.
Servidor Backend	Instancia Render con Node.js y Express. Especificaciones: 512 MB RAM, CPU compartida, disco efímero. Tier gratuito con suspensión automática tras 15 minutos de inactividad.
Base de Datos	Supabase PostgreSQL Database. Almacena todas las entidades del sistema: pólizas, pagos, siniestros, usuarios. Tier gratuito: 500 MB de almacenamiento, conexiones ilimitadas.
Storage	Supabase Storage para archivos adjuntos. Almacena evidencias de siniestros (imágenes, PDFs). Tier gratuito: 1 GB de almacenamiento, transferencia ilimitada.
GitHub Repository	Repositorio Git para control de versiones del código fuente. Integrado con CI/CD para despliegues automáticos a Vercel y Render.

### 1.3. Interrelaciones de los Nodos

Nodo1	Nodo2	Interrelación
Cliente Web	Vercel CDN	HTTPS: El navegador carga la aplicación React desde Vercel CDN
Cliente Web	Servidor Backend	REST API HTTPS: La aplicación consume endpoints del backend
Servidor Backend	Base de Datos	PostgreSQL Protocol: El backend ejecuta queries mediante Supabase SDK
Servidor Backend	Storage	HTTPS: El backend gestiona carga y descarga de archivos

Nodo1	Nodo2	Interrelación
GitHub Repository	Vercel CDN	GitHub Webhooks: Despliegue automático del frontend en cada push
GitHub Repository	Servidor Backend	GitHub Webhooks: Despliegue automático del backend en cada push

## 2. SERVIDOR BACKEND

### 2.1. Descripción

Servidor de aplicaciones Node.js desplegado en Render. Es el único servidor del sistema, ejecutando toda la lógica de negocio, validaciones y orquestación de servicios. Expone una API REST para el frontend y se comunica con Supabase para datos y archivos.

#### Especificaciones Técnicas:

**Plataforma:** Render Free Tier

**Runtime:** Node.js v18+

**Framework:** Express.js

**RAM:** 512 MB

**CPU:** Compartida

**Disco:** Efímero (no persistente)

**Suspensión:** Automática tras 15 minutos de inactividad

### 2.2. Composición

El servidor backend contiene los siguientes módulos y componentes desplegados:

- API Controllers: Endpoints REST para todas las operaciones
- Business Services: Gestión de Pólizas, Pagos, Siniestros, Reportes
- Business Rules: Validaciones y reglas de negocio
- Data Access Layer: DAOs y conexión a Supabase
- Security Module: Autenticación JWT y autorización RBAC
- Notification Service: Envío de emails
- File Management: Integración con Supabase Storage

### 2.3. Interrelaciones

El servidor backend se relaciona con los siguientes nodos:

1. **Cliente Web:** Recibe peticiones HTTPS REST desde el frontend React
2. **Base de Datos Supabase:** Ejecuta queries mediante Supabase SDK para operaciones CRUD
3. **Storage Supabase:** Gestiona carga/descarga de archivos adjuntos
4. **GitHub:** Recibe actualizaciones automáticas vía webhooks para CI/CD

## 3. SUPABASE (BASE DE DATOS Y STORAGE)

### 3.1. Descripción

Supabase es una plataforma Backend-as-a-Service que proporciona base de datos PostgreSQL, autenticación y almacenamiento de archivos. En este proyecto maneja toda la persistencia de datos y archivos del sistema.

Especificaciones Base de Datos:

**Motor:** PostgreSQL 15

**Almacenamiento:** 500 MB (tier gratuito)

**Conexiones:** Ilimitadas

**API:** REST y SDK JavaScript

Especificaciones Storage:

**Almacenamiento:** 1 GB (tier gratuito)

**Transferencia:** Ilimitada

**Tipos soportados:** Imágenes (JPEG, PNG), PDFs, documentos

### 3.2. Composición

Supabase contiene las siguientes tablas y buckets:

**Tablas:** usuarios, polizas, pagos, siniestros, beneficiarios, roles, auditorias

**Bucket Storage:** evidencias-siniestros (para archivos adjuntos)

## Historia de Revisiones

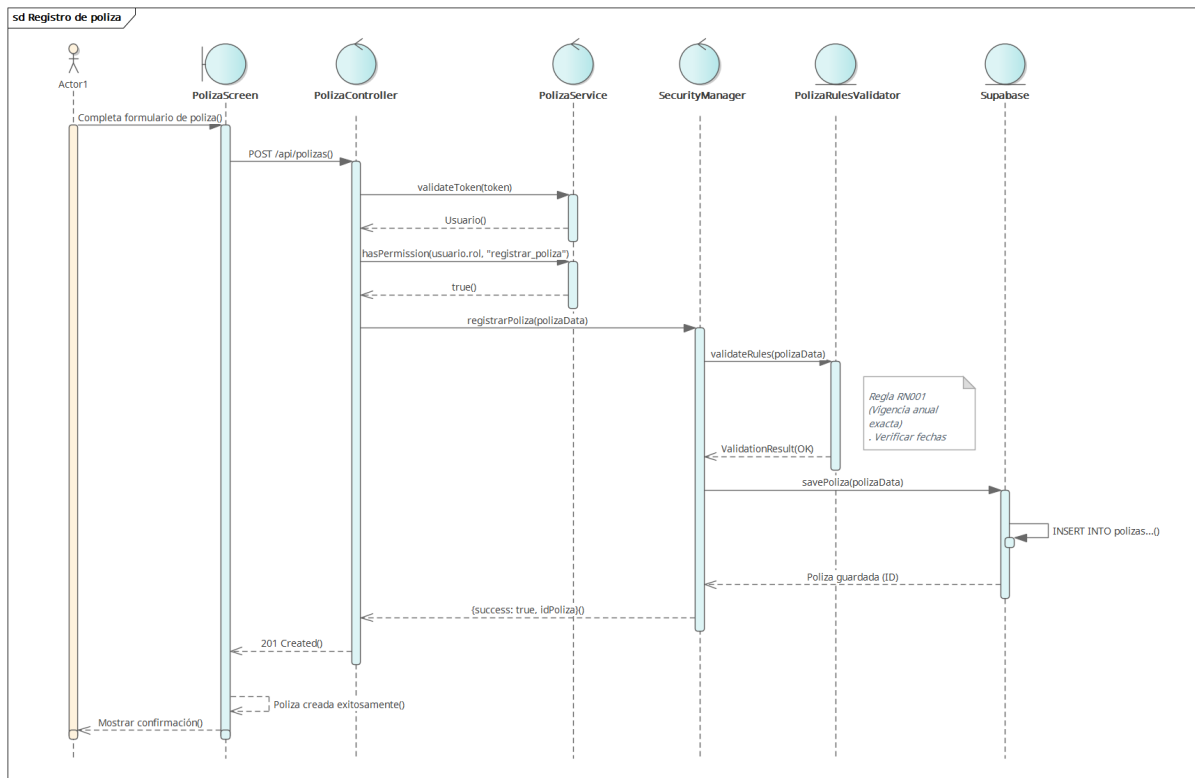
Fecha	Versión	Descripción	Autor
03/12/2024	1.0	Versión inicial de arquitectura de despliegue cloud	Equipo UTPL

# Sistema de Gestión de Pólizas de Vida - UTPL

## Diagramas de Secuencia

### 1. DIAGRAMA - REGISTRO DE PÓLIZA

Diagrama de Secuencia: Registro de nueva póliza de seguro de vida



#### 1.1. Descripción de las Funciones

**validarDatosEmpleado():** Valida que el empleado pertenece a UTPL y está activo en el sistema de RRHH.

**crearPoliza():** Crea el registro de la póliza en la base de datos con estado 'Activa' y fecha de inicio.

**registrarBeneficiarios():** Asocia los beneficiarios designados a la póliza con sus porcentajes de beneficio.

**generarCertificado():** Genera el certificado de póliza en formato PDF con todos los detalles del seguro.

**enviarNotificacion():** Envía correo electrónico al asegurado confirmando el registro de la póliza.

## 1.2. Descripción de las Condiciones

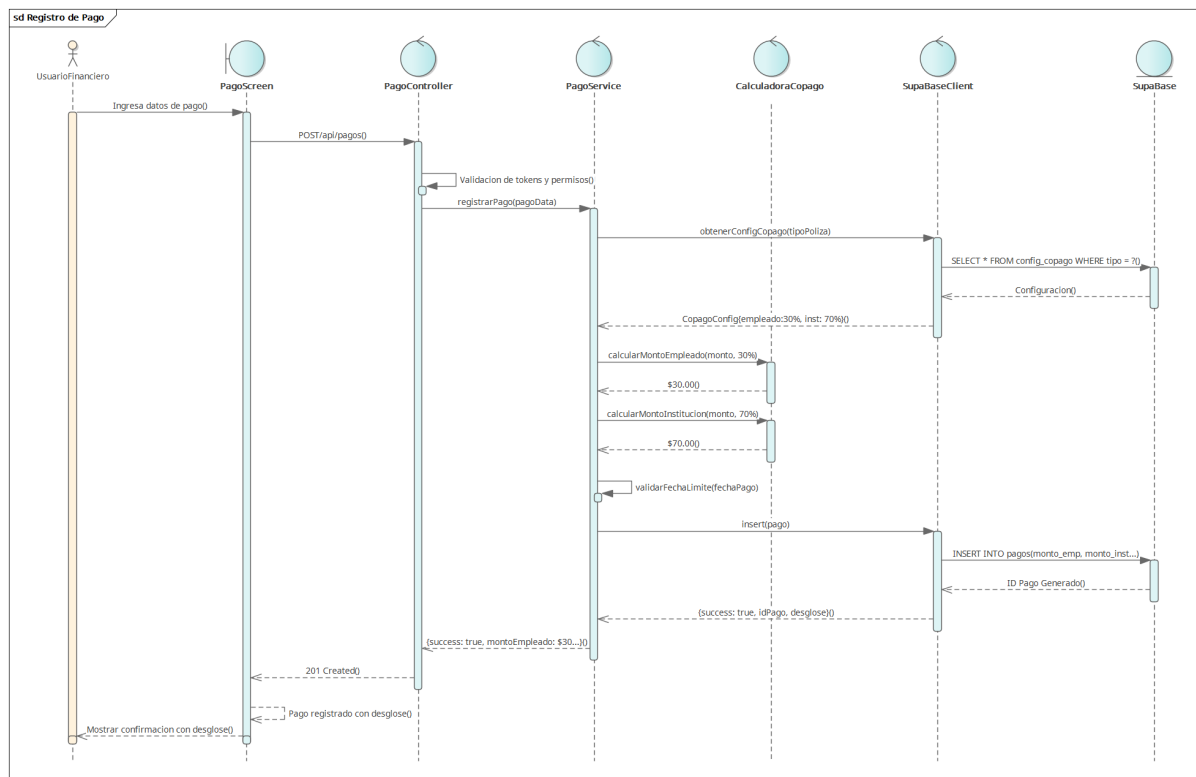
**¿Datos válidos?:** Verifica que todos los campos requeridos están completos y cumplen formato (cédula, email, fechas).

**¿Empleado activo?:** Confirma que el empleado está activo en RRHH y no tiene restricciones para contratar seguro.

**¿Beneficiarios válidos?:** Verifica que la suma de porcentajes de beneficiarios sea 100% y los datos sean correctos.

## 2. DIAGRAMA - PROCESAMIENTO DE PAGO

Diagrama de Secuencia: Procesamiento de pago mensual de póliza



## 2.1. Descripción de las Funciones

**verificarPolizaActiva():** Verifica que la póliza está en estado 'Activa' y no ha vencido.

**calcularMontoPago():** Calcula el monto total incluyendo coaseguro (20%) y copago (10%).

**procesarPagoBanco():** Envía la solicitud de débito al sistema bancario del asegurado.

**registrarPago():** Almacena el registro del pago en el sistema con fecha, monto y estado.

**actualizarEstadoPoliza():** Actualiza el estado de la póliza según el resultado del pago.

## 2.2. Descripción de las Condiciones

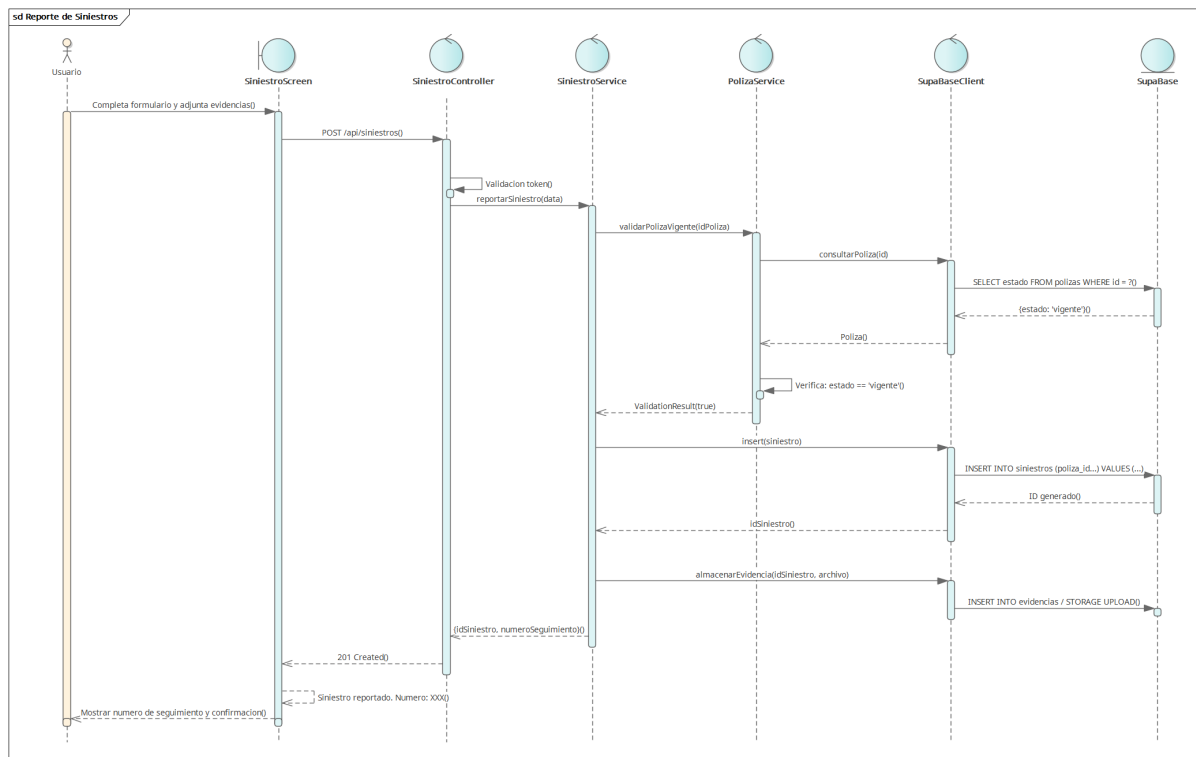
**¿Póliza activa?:** Verifica que la póliza no está vencida ni cancelada.

**¿Pago exitoso?:** Confirma que el banco aprobó y procesó el pago correctamente.

**¿Dentro de plazo?:** Verifica que el pago se realiza antes del día 5 del mes.

## 3. DIAGRAMA - REPORTE DE SINIESTRO

Diagrama de Secuencia: Reporte y procesamiento de siniestro



## 3.1. Descripción de las Funciones

**validarElegibilidad():** Verifica que la póliza está activa y al día con los pagos para reportar siniestro.

**cargarEvidencias():** Sube los archivos adjuntos (facturas, informes médicos) a Supabase Storage.

**crearSiniestro():** Crea el registro del siniestro en estado 'Pendiente' con todos los detalles.

**notificarAseguradora():** Envía notificación automática a la aseguradora sobre el nuevo siniestro.

**evaluarSiniestro():** La aseguradora revisa las evidencias y determina aprobación o rechazo.

**actualizarEstadoSiniestro():** Cambia el estado del siniestro a 'Aprobado' o 'Rechazado' según evaluación.

## 3.2. Descripción de las Condiciones

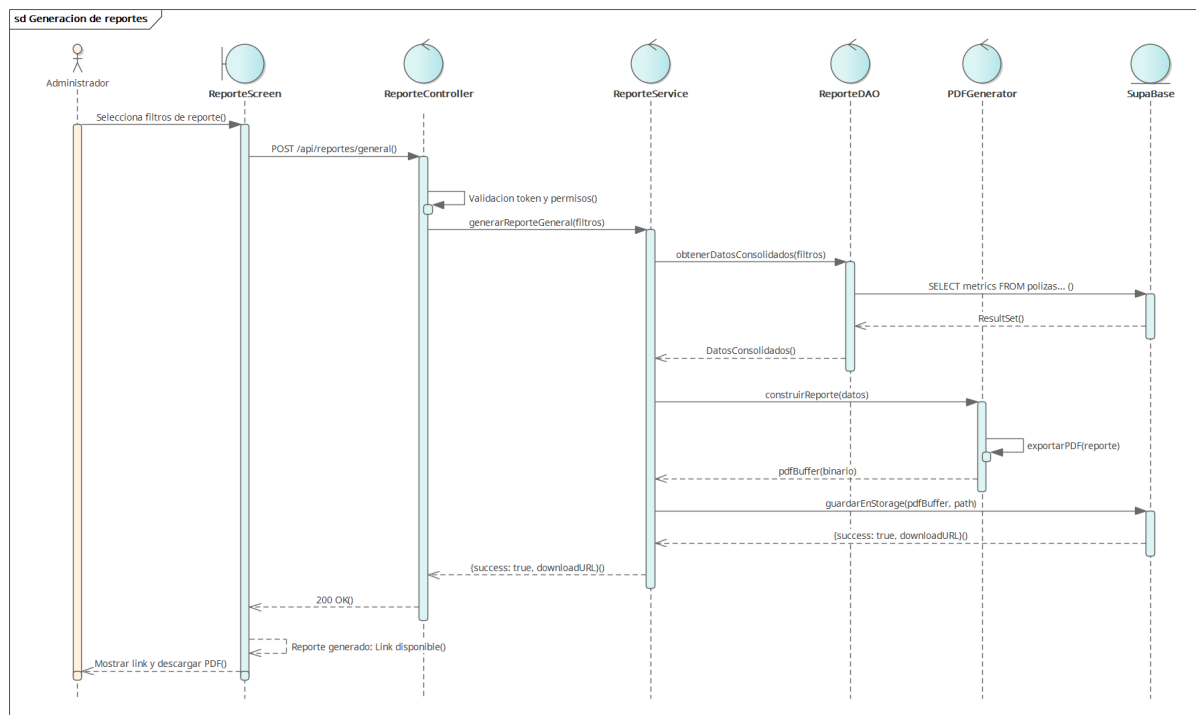
**¿Póliza elegible?:** Verifica que la póliza está activa y sin pagos pendientes.

**¿Evidencias completas?:** Confirma que se adjuntaron todos los documentos requeridos (facturas, informes).

**¿Siniestro aprobado?:** Determina si la aseguradora aprueba el siniestro según evaluación.

## 4. DIAGRAMA - GENERACIÓN DE REPORTES

Diagrama de Secuencia: Generación de reportes estadísticos



### 4.1. Descripción de las Funciones

**validarPermisos():** Verifica que el usuario tiene rol autorizado para generar reportes (Admin, Asesor).

**consultarDatos():** Ejecuta queries agregadas en la base de datos según filtros del reporte.

**procesarEstadisticas():** Calcula métricas, promedios y totales para el reporte.

**generarGraficos():** Crea visualizaciones gráficas de los datos (barras, líneas, pastel).

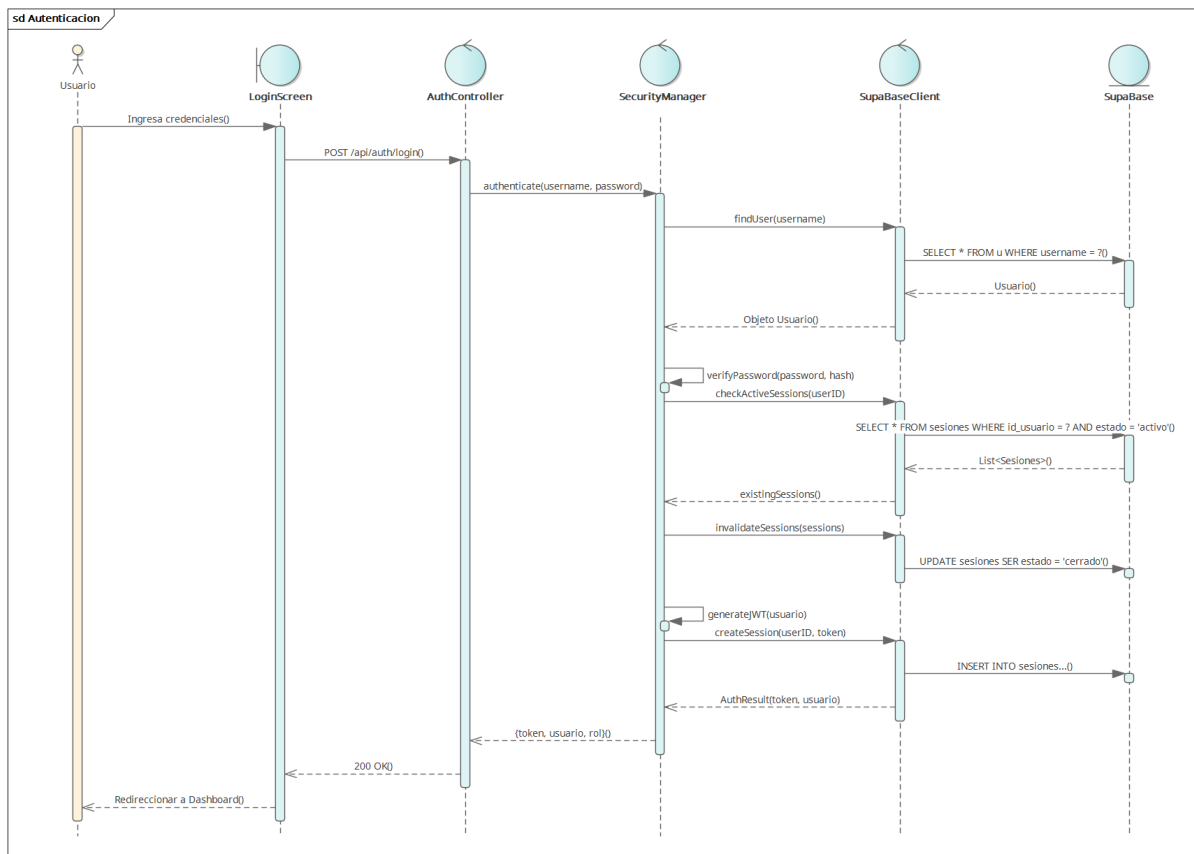
**exportarReporte():** Genera el archivo final en formato PDF o Excel según preferencia del usuario.

## 4.2. Descripción de las Condiciones

**¿Usuario autorizado?:** Verifica que el usuario tiene permisos para acceder a reportes.

**¿Filtros válidos?:** Confirma que el rango de fechas y filtros seleccionados son válidos.

**¿Hay datos?:** Verifica que existen datos en el período seleccionado para generar el reporte.



## Historia de Revisiones

Fecha	Versión	Descripción	Autor
03/12/2024	1.0	Versión inicial de diagramas de secuencia	Equipo UTPL

## Bitácora de reunión y organización de equipo.

Actividad	Responsable(S)
Lectura del archivo enviado con especificaciones del proyecto	Jean Villavicencio Sebastian Mendieta Orly Flores Alex Aguirre
Escribir inquietudes en archivo Notion compartido	Jean Villavicencio Sebastian Mendieta Orly Flores Alex Aguirre
Reunión con encargada del proyecto	Jean Villavicencio Sebastian Mendieta Orly Flores Alex Aguirre
DPlaneación y discusión de Diagramas de secuencia.	Jean Villavicencio Sebastian Mendieta Orly Flores Alex Aguirre
Planeación y discusión de Diagramas de componentes.	Jean Villavicencio Sebastian Mendieta Orly Flores Alex Aguirre
Planeación y discusión de Diagramas de paquetes.	Jean Villavicencio Sebastian Mendieta Orly Flores Alex Aguirre
Planeación y discusión de Diagramas de despliegue.	Jean Villavicencio Sebastian Mendieta Orly Flores Alex Aguirre
Hacer diagramas en borrador para mayor organización	Jean Villavicencio Sebastian Mendieta
Pasar diagramas del borrador	Orly Flores Alex Aguirre
Elaboración del archivo para entrega final	Jean Villavicencio Sebastian Mendieta Orly Flores Alex Aguirre