Especificación de Requisitos de Software (ERS)

Proyecto: Sistema de Administración de Laboratorio Clínico (SALC)
Versión: 2.9
Estado: Final

1. Introducción

1.1 Propósito

Este documento define los requisitos funcionales y no funcionales del **Sistema de Administración de Laboratorio Clínico (SALC)**. El objetivo es la implementación de una aplicación de escritorio para la gestión de las entidades Paciente, Análisis y Usuario, optimizando la trazabilidad, seguridad y eficiencia operativa del laboratorio.

1.2 Alcance

El sistema SALC se centrará exclusivamente en la operativa interna del laboratorio.

- **Funcionalidad IN-SCOPE:**
  - Gestión de entidades maestras: Pacientes, Obras Sociales, Tipos de Análisis y Métricas.
  - Control de acceso y permisos basado en tres roles definidos: Administrador, Médico y Asistente.
  - Ciclo de vida de los análisis: Creación, carga de resultados y validación por firma.
  - Administración del sistema: Gestión de usuarios, catálogos y copias de seguridad.
  - **Visualización de reportes estadísticos (BI):** Generación de gráficos y conteos sobre la performance y los datos del laboratorio (ej. análisis por estado, pacientes por obra social).
- **Funcionalidad OUT-OF-SCOPE:**
  - Gestión de turnos o agenda médica.
  - Módulos de facturación o contabilidad.
  - Portal web para pacientes o médicos externos.
  - Gestión de médicos externos.
  - Integración con sistemas de información hospitalaria (HIS/LIS) en esta versión.

1.3 Personal Involucrado

Los roles de usuario definen el alcance funcional dentro del sistema.

Rol	Alcance Funcional Detallado (Acciones habilitadas)
Administrador	Ejecuta operaciones de Alta, Baja y Modificación (ABM) sobre todas las entidades del sistema (Usuarios, Análisis, Obras Sociales, etc.), **excepto Pacientes**. Configura y gestiona las copias de seguridad de la base de datos. **Podrá**

Rol	Alcance Funcional Detallado (Acciones habilitadas)
	crear, modificar y eliminar las asociaciones entre los tipos de análisis y las métricas que los componen. Además, visualiza reportes estadísticos globales del sistema.
Médico	Crea un nuevo Análisis para un Paciente, asociándose a él. Carga los resultados numéricos del análisis. Valida los resultados, firmando digitalmente el análisis. Visualiza la lista completa de pacientes, pero solo puede acceder al historial detallado de análisis de los pacientes a los que él mismo está asociado. Podrá realizar la modificación de datos de Pacientes y la baja lógica (cambio de estado a "Inactivo") de los mismos. Además, visualiza reportes estadísticos filtrados por su propia actividad.
Asistente	**Podrá realizar el Alta y Modificación de Pacientes.** Visualiza la lista completa de pacientes y puede acceder al historial detallado de análisis de cualquier paciente. Genera el informe en PDF de un Análisis que ya ha sido previamente validado por un Médico. Envía dicho informe al paciente por email o teléfono.

Relación Jerárquica: Cada Asistente está supervisado por un único Médico.

2. Descripción General

2.1 Perspectiva del Producto

El SALC es una **aplicación de escritorio monolítica** desarrollada en **Windows Forms sobre .NET Framework 4.7.2**. Se conecta directamente a una base de datos **SQL Server 2022**.

2.2 Restricciones

- **Plataforma:** Windows 10/11 (x64).
- **Base de Datos:** SQL Server 2022 (Developer/Express Edition).
- **Base de Datos Preexistente:** El sistema se conectará a una base de datos ya creada y poblada.
- **Autenticación:** Acceso por DNI/contraseña. El hash se realizará con **BCrypt**.

3. Requisitos Específicos

3.1 Requisitos de Interfaces

Código	Interfaz	Especificación Técnica
UI-01	Interfaz de Usuario	Interfaz gráfica MDI (Multiple Document Interface) implementada en Windows Forms, siguiendo la guía de estilo de la sección 4.4.
UI-02	Interfaz de Reportes (Admin)	Formulario (`FrmReportesAdmin`) que debe contener controles de gráficos (ej. `System.Windows.Forms.DataVisualization.Charting.Chart`) para mostrar datos agregados de todo el sistema.
UI-03	Interfaz de Reportes (Médico)	Formulario (`FrmReportesMedico`) que debe contener controles de gráficos para mostrar datos agregados, filtrados por el DNI del médico logueado.
HW-01	Hardware	Cliente: CPU 2 núcleos, 4 GB RAM, 10 GB HDD. Conectividad TCP/IP a la red local.
SW-01	Software	Cliente: .NET Framework 4.7.2 Runtime. Servidor: SQL Server 2022.
COM-01	Comunicaciones	Conexión directa a la base de datos mediante ADO.NET sobre TCP/IP, puerto 1433 por defecto.

3.2 Requisitos Funcionales (RF)

Código	Nombre	Actor(es)	Descripción Detallada
RF-01	Autenticar Usuario	Todos	El sistema debe validar el DNI y la contraseña del usuario contra el hash almacenado en usuarios.password_hash . Si la validación es exitosa, se concederá acceso según el id_rol asociado.
RF-02	ABM de Usuarios	Administrador	El sistema permitirá crear, leer, modificar y eliminar registros en las tablas usuarios, medicos y asistentes, manteniendo la integridad referencial.
RF-03	ABM de Pacientes	Médico, Asistente	**El sistema permitirá al Asistente el alta y modificación de la entidad pacientes. El Médico podrá modificar los datos del paciente y realizar una baja lógica (cambio de estado a "Inactivo").**
RF-04	ABM de Catálogos	Administrador	El sistema permitirá el ABM de obras_sociales, tipos_analisis y metricas.
RF-05	Crear Análisis	Médico	Un usuario Médico podrá crear un nuevo registro en la tabla analisis, asociando un dni_paciente y un id_tipo_analisis. Su DNI se registrará en dni_carga.

Código	Nombre	Actor(es)	Descripción Detallada
RF-06	Cargar Resultados	Médico	Sobre un análisis existente en estado "Sin verificar", el Médico podrá insertar o modificar registros en analisis_metrica, registrando el valor numérico de cada métrica.
RF-07	Validar Análisis	Médico	El Médico podrá cambiar el id_estado de un análisis a "Verificado". Al hacerlo, su DNI se registrará en dni_firma y se establecerá la fecha_firma. Los resultados en analisis_metrica se vuelven inmutables.
RF-08	Generar y Enviar Informe	Asistente	El sistema debe generar un informe en formato PDF. Un Asistente solo puede hacerlo para análisis con estado "Verificado”.
RF-09	Visualizar Historial	Médico, Asistente	El sistema mostrará una lista de todos los pacientes. Un Asistente podrá ver el historial detallado de cualquier paciente. Un Médico solo podrá ver el historial detallado de los pacientes asociados a los análisis que él cargó (analisis.dni_carga).
RF-10	Gestionar Backups	Administrador	El sistema proveerá una interfaz para ejecutar y programar copias de seguridad de la base de datos SQL Server.

Código	Nombre	Actor(es)	Descripción Detallada
RF-11	Gestionar Relación Tipo de Análisis - Métrica	Administrador	El sistema permitirá al Administrador crear, modificar y eliminar las asociaciones entre los diferentes tipos de análisis y las métricas que los componen, utilizando la tabla `tipo_analisis_metrica`. Esto incluye la capacidad de asignar una o varias métricas a un tipo de análisis y viceversa.
RF-12	Visualizar Reportes Globales	Administrador	El sistema debe proveer al Administrador una pantalla (`FrmReportesAdmin`) que muestre, como mínimo, los siguientes reportes gráficos: Conteo de análisis agrupados por estado, Conteo de pacientes agrupados por obra social, y Conteo de usuarios agrupados por rol.
RF-13	Visualizar Reportes Personales	Médico	El sistema debe proveer al Médico una pantalla (`FrmReportesMedico`) que muestre, como mínimo, los siguientes reportes gráficos filtrados por su DNI: Conteo de análisis cargados por él (agrupados por estado) y Conteo de análisis firmados por él (agrupados por tipo).

3.3 Requisitos No Funcionales (RNF)

Código	Requisito	Criterio de Aceptación
RNF-01	Rendimiento	Las consultas y carga de datos en grillas no deben exceder los 3 segundos. La generación de informes PDF no debe exceder los 5 segundos. **La generación y renderizado de los gráficos de reportes estadísticos no debe exceder los 5 segundos.**
RNF-02	Seguridad	Todas las contraseñas deben almacenarse hasheadas con BCrypt. Todas las consultas a la base de datos deben usar SqlParameter para prevenir inyección SQL.
RNF-03	Usabilidad	La interfaz debe ser consistente con la guía de estilo de la sección 4.4, garantizando una navegación intuitiva y predecible para usuarios de aplicaciones de escritorio Windows.
RNF-04	Disponibilidad	El sistema deberá estar disponible durante el horario laboral del laboratorio (99% de uptime en dicho período).
RNF-05	Mantenibilidad	El código fuente deberá estar estructurado siguiendo estrictamente la arquitectura MVP de 3 capas y los principios SOLID descritos en la

		sección 4.2.

4. Apéndices

4.1 Stack Tecnológico

- **Lenguaje:** C#
- **Framework:** .NET Framework 4.7.2
- **Interfaz de Usuario:** Windows Forms
- **Acceso a Datos:** ADO.NET (System.Data.SqlClient)
- **Base de Datos:** SQL Server 2022
- **Hashing de Contraseñas:** BCrypt.Net-Next (o similar compatible)

4.2 Arquitectura y Patrones de Diseño (v2.6)

El sistema se desarrollará sobre una arquitectura de 3 capas lógicas, aplicando el patrón **Model-View-Presenter (MVP)** para la capa de presentación, con el fin de separar la lógica de la interfaz de usuario y maximizar la testabilidad y mantenibilidad del código.

4.2.1. Estructura de Capas

A continuación, se detalla la arquitectura del sistema, dividida en capas y el patrón de diseño Model-View-Presenter (Passive View) utilizado:

1. Arquitectura en Capas

- **Capa de Presentación (UI - User Interface):**
  - **Componentes:** Formularios de Windows Forms, controles de usuario y Presenters.
  - **Responsabilidad:** Se encarga de la interfaz de usuario, captura las interacciones del usuario y delega las operaciones lógicas al Presenter. Es una capa "pasiva" sin lógica de negocio.
  - **Nomenclatura:** Ejemplos incluyen `FrmGestionPacientes.cs` y `PacientesPresenter.cs`.
- **Capa de Lógica de Negocio (BLL - Business Logic Layer):**
  - **Componentes:** Clases de Servicio.
  - **Responsabilidad:** Orquesta las reglas de negocio, realiza validaciones complejas y coordina las operaciones entre la capa de presentación y la de acceso a datos.
  - **Nomenclatura:** Ejemplos incluyen `PacienteService.cs` y `AnalisisService.cs`.
- **Capa de Acceso a Datos (DAL - Data Access Layer):**
  - **Componentes:** Clases de Repositorio.

    - **Responsabilidad:** Encapsula la lógica de interacción con la base de datos SQL Server utilizando ADO.NET, realizando operaciones CRUD (Crear, Leer, Actualizar, Borrar).
    - **Nomenclatura:** Ejemplos incluyen `PacienteRepository.cs` y `UsuarioRepository.cs`.

2. Patrón de Diseño: Model-View-Presenter (Passive View)

- **View (Vista):**
    - Es el formulario (`System.Windows.Forms.Form`) que implementa una interfaz (`IView`) para desacoplarse del Presenter. No contiene lógica de decisión.
    - **Ejemplo:** `IFrmPacienteView` define propiedades (`NombrePaciente`, `ApellidoPaciente`) y eventos (`GuardarClick`), implementados por `FrmPaciente.cs`.
- **Presenter (Presentador):**
    - Actúa como intermediario, conteniendo la lógica de la UI. Se comunica con la BLL para obtener y enviar datos, y actualiza la Vista a través de su interfaz.
    - **Ejemplo:** `PacientePresenter` recibe el evento `GuardarClick` de la Vista, invoca a `PacienteService` para validar y guardar datos, y luego notifica a la Vista el resultado.
- **Model (Modelo):**
    - Representa las entidades de negocio (POCOs - Plain Old CLR Objects), que son objetos simples que contienen los datos.
    - **Ejemplo:** `class Paciente { public int Dni { get; set; } ... }`.

3. Flujo de Ejecución (Ejemplo: Guardar un Paciente):

`FrmPaciente (View)` -> Usuario hace clic en Guardar -> `Presenter.OnGuardarClick()` -> `PacienteService.GuardarPaciente(modelo)` -> `PacienteRepository.Update(modelo)` -> Ejecución de `SQLCommand` con `ADO.NET`.

4. Flujo de Navegación (Reportes):

Los usuarios (Administrador y Médico) accederán a sus reportes mediante un botón dedicado en su panel principal (ej. `FrmPanelAdministrador` -> Botón "Reportes") o a través del menú superior "Reportes".

4.2.3. Buenas Prácticas y Principios

- **Principio de Responsabilidad Única (SOLID - S):** Cada clase tiene un único propósito:
    - Form: Solo para mostrar y recibir datos.
    - Presenter: Solo lógica de presentación.
    - Service: Solo lógica de negocio.
    - Repository: Solo acceso a datos de una entidad.

- **Seguridad - Prevención de Inyección SQL:** Todas las consultas SQL en la capa DAL **deben** utilizar SqlParameter para pasar valores. Queda terminantemente prohibida la concatenación de strings para construir consultas.
- **Validación de Datos:**
  - **Nivel UI (Presenter):** Se realizan validaciones de formato, tipo y obligatoriedad (ej. que un DNI sea numérico, que un email tenga formato válido). Se utilizará el control ErrorProvider para notificar al usuario.
  - **Nivel BLL (Service):** Se realizan validaciones de reglas de negocio (ej. que un médico no pueda validar su propio análisis si la regla lo prohibiera).
- **Reutilización de Código:** Se creará una clase DbConnection en la capa DAL para gestionar de forma centralizada la cadena de conexión y la apertura/cierre de conexiones a la base de datos.

4.3 Modelo de Datos (Diccionario de Datos) - v2.5

Esta sección describe la estructura detallada de la base de datos.

Tablas de Catálogo

Tabla: roles		
Columna	**Tipo de Dato SQL**	**Restricciones y Descripción**
id_rol	INT	PK, IDENTITY. Identificador único del rol.
nombre_rol	NVARCHAR(50)	NOT NULL, UNIQUE. Nombre del rol ('Administrador', 'Médico', 'Asistente').

Tabla: estados_analisis		
Columna	**Tipo de Dato SQL**	**Restricciones y Descripción**
id_estado	INT	PK, IDENTITY. Identificador único del estado.
descripcion	NVARCHAR(50)	NOT NULL, UNIQUE. Descripción ('Sin verificar', 'Verificado').

Tabla: obras_sociales		
Columna	**Tipo de Dato SQL**	**Restricciones y Descripción**
id_obra_social	INT	PK, IDENTITY. ID único.
cuit	NVARCHAR(13)	NOT NULL, UNIQUE. CUIT de la entidad.
nombre	NVARCHAR(50)	NOT NULL. Nombre de la obra social.
estado	NVARCHAR(20)	NOT NULL. CHECK ('Activo', 'Inactivo').

Tabla: tipos_analisis		
Columna	**Tipo de Dato SQL**	**Restricciones y Descripción**
id_tipo_analisis	INT	PK, IDENTITY. ID único.
descripcion	NVARCHAR(100)	NOT NULL. Nombre del análisis.
estado	NVARCHAR(20)	NOT NULL. CHECK ('Activo', 'Inactivo').

Tabla: metricas		
Columna	**Tipo de Dato SQL**	**Restricciones y Descripción**
id_metrica	INT	PK, IDENTITY. ID único.
nombre	NVARCHAR(100)	NOT NULL. Nombre de la métrica (ej. 'Glucosa').
unidad_medida	NVARCHAR(20)	NOT NULL. Unidad (ej.

		'mg/dL').
valor_maximo	DECIMAL(10,2)	NULL. Valor de referencia máximo.
valor_minimo	DECIMAL(10,2)	NULL. Valor de referencia mínimo.
estado	NVARCHAR(20)	NOT NULL. CHECK ('Activo', 'Inactivo').

Tabla: tipo_analisis_metrica		
Columna	**Tipo de dato SQL**	**Restricciones y Descripción**
id_tipo_analisis	**INT**	**PK, FK a tipos_analisis.**
id_metrica	**INT**	**PK, FK a metricas.**
		Define qué métricas componen cada tipo de análisis. Relación N:N. Clave primaria compuesta por (id_tipo_analisis, id_metrica).
		Clave foránea a `tipos_analisis(id_tipo_analisis)` con `ON DELETE CASCADE`.
		Clave foránea a `metricas(id_metrica)` con `ON DELETE CASCADE`.

Tablas de Entidades Principales

Tabla: usuarios (Datos comunes)		
Columna	**Tipo de Dato SQL**	**Restricciones y Descripción**
dni	INT	PK. Clave primaria.
nombre	NVARCHAR(50)	NOT NULL.
apellido	NVARCHAR(50)	NOT NULL.
email	NVARCHAR(100)	NOT NULL, UNIQUE.
password_hash	NVARCHAR(255)	NOT NULL. Hash de la contraseña (BCrypt).
id_rol	INT	NOT NULL, FK a roles.id_rol.
estado	NVARCHAR(20)	NOT NULL. CHECK ('Activo', 'Inactivo').

Tabla: medicos (Extensión de usuarios)		
Columna	**Tipo de Dato SQL**	**Restricciones y Descripción**
dni	INT	PK, FK a usuarios.dni.
nro_matricula	INT	NOT NULL, UNIQUE.
especialidad	NVARCHAR(50)	NOT NULL.

Tabla: asistentes (Extensión de usuarios)		

Columna	Tipo de Dato SQL	Restricciones y Descripción
dni	INT	PK, FK a usuarios.dni.
dni_supervisor	INT	NOT NULL, FK a medicos.dni.
fecha_ingreso	DATE	NOT NULL.

Tabla: pacientes		
Columna	**Tipo de Dato SQL**	**Restricciones y Descripción**
dni	INT	PK.
nombre	NVARCHAR(50)	NOT NULL.
apellido	NVARCHAR(50)	NOT NULL.
fecha_nac	DATE	NOT NULL.
sexo	CHAR(1)	NOT NULL. CHECK ('M', 'F', 'X').
email	NVARCHAR(100)	NULL.
telefono	NVARCHAR(20)	NULL.
id_obra_social	INT	NULL, FK a obras_sociales.id_obra_social.
estado	NVARCHAR(20)	NOT NULL. CHECK ('Activo', 'Inactivo').

Tablas Transaccionales

Tabla: analisis		
Columna	**Tipo de Dato SQL**	**Restricciones y**

		Descripción
id_analisis	INT	PK, IDENTITY.
id_tipo_analisis	INT	NOT NULL, FK a tipos_analisis.
id_estado	INT	NOT NULL, FK a estados_analisis.
dni_paciente	INT	NOT NULL, FK a pacientes.
dni_carga	INT	NOT NULL, FK a medicos. Médico que crea el análisis.
dni_firma	INT	NULL, FK a medicos. Médico que valida el análisis.
fecha_creacion	DATETIME2	NOT NULL.
fecha_firma	DATETIME2	NULL.
observaciones	NVARCHAR(MAX)	NULL.

Tabla: analisis_metrica		
Columna	**Tipo de Dato SQL**	**Restricciones y Descripción**
id_analisis	INT	PK, FK a analisis.
id_metrica	INT	PK, FK a metricas.
resultado	DECIMAL(10,2)	NOT NULL. Valor numérico del resultado.
observaciones	NVARCHAR(500)	NULL.

4.4 Arquitectura de Información y Guía de Estilo UX/UI (v2.6)

4.4.1. Arquitectura de Información (AI)

La estructura de navegación será jerárquica y orientada a tareas, partiendo de un formulario principal que actúa como contenedor.

- **Contenedor Principal:** La aplicación se iniciará en un formulario principal (FrmPrincipal) que contendrá un control MenuStrip (menú clásico) y un ToolStrip (barra de herramientas con iconos) para las funciones más comunes. Este formulario funcionará como un contenedor MDI (Multiple Document Interface).
- **Flujo de Navegación:** El usuario accederá a las diferentes secciones (ABM de Pacientes, Gestión de Análisis, etc.) a través del menú. Cada sección se abrirá como una ventana hija dentro del contenedor MDI.
    - **Ejemplo:** Menú: Archivo -> Gestionar -> Pacientes -> Se abre FrmGestionPacientes.

4.4.2. Guía de Estilo UX/UI

El diseño de la interfaz será utilitario, clásico y consistente, emulando la estética de aplicaciones de escritorio tradicionales de Windows para minimizar la curva de aprendizaje.

- **Layout y Composición:**
    - **Formularios de Gestión (Grillas):** La parte superior contendrá un ToolStrip con botones para Nuevo, Editar, Eliminar y Buscar. El área principal será ocupada por un DataGridView que mostrará los datos.
    - **Formularios de Edición/Alta:** Serán diálogos modales. Los campos se agruparán lógicamente utilizando controles GroupBox o TabControl. La alineación de etiquetas y campos será consistente.
    - **Botones de Acción:** Los botones Aceptar/Guardar y Cancelar siempre estarán ubicados en la esquina inferior derecha del formulario.
- **Paleta de Colores:** Se utilizará la paleta de colores por defecto del sistema operativo para garantizar una apariencia nativa.
    - **Fondo:** SystemColors.Control (gris estándar).
    - **Texto:** SystemColors.ControlText (negro estándar).
    - **Selección:** SystemColors.Highlight (azul estándar).
- **Tipografía:**
    - **Fuente Estándar:** Microsoft Sans Serif o Segoe UI, en tamaño 8.25pt o 9pt (predeterminado por Visual Studio para Windows Forms).
    - **Jerarquía:** Los títulos de los GroupBox y las etiquetas (Label) importantes usarán la propiedad Font.Bold = true.
- **Interacción y Feedback:**
    - **Orden de Foco:** El TabIndex de los controles seguirá un orden lógico de izquierda a derecha y de arriba hacia abajo.
    - **Notificaciones:** Se utilizará MessageBox.Show() para mostrar mensajes de confirmación (ej. "¿Confirma la eliminación?"), información y error.
    - **Validación:** El control ErrorProvider se usará para indicar visualmente los errores de validación en los campos de entrada, mostrando un ícono de error junto al control.