Desarrollo de una Aplicación de Control de Entrada y Salida del Personal de OptiSalud

Cristian Danilo García Ardila

Universidad Nacional Abierta u a Distancia - UNAD

Escuela de Ciencias Básicas Tecnología e Ingeriría – ECBTI

Ingeniería de Sistemas

Yopal

2024

Dedicatoria

Este proyecto representa un viaje personal, una odisea que he navegado solo, pero no sin el apoyo silencioso de aquellos que creen en el poder de la innovación. Está dedicado a todos los individuos que valoran la autonomía y la pasión por crear soluciones significativas. Este es un tributo a la determinación y al deseo constante de aprender y crecer. Que este proyecto sea un testimonio de que los sueños individuales pueden materializarse con voluntad y esfuerzo, y que sirva como inspiración para futuros desafíos y logros personales.

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas que han sido parte fundamental en la realización de este proyecto. En primer lugar, agradezco profundamente a mi familia por su apoyo incondicional y por ser mi fuente constante de motivación. Quiero extender mi gratitud a los líderes y colegas de OptiSalud por brindarme la oportunidad de llevar a cabo este proyecto y por proporcionarme los recursos necesarios para hacerlo posible. Especialmente, agradezco al ingeniero Rafael Pérez Holguín por su orientación experta y su inagotable paciencia durante el proceso de desarrollo. También quiero reconocer a la comunidad en línea y a todas las personas que han compartido su conocimiento y experiencias a través de plataformas educativas y foros de desarrollo. Sus valiosas contribuciones han sido esenciales para mi aprendizaje y crecimiento profesional. Finalmente, agradezco a cada usuario potencial que, con sus necesidades y expectativas, ha dado forma a este proyecto. Su confianza en la tecnología que estoy construyendo es mi mayor motivación para crear algo significativo y funcional. Este proyecto no habría sido posible sin el apoyo, la orientación y la confianza de todas estas personas. Gracias por creer en mí y en este proyecto. Estoy eternamente agradecido.

Resumen

El proyecto desarrollado se centra en la creación de un sistema integral de registro de entrada y salida para los empleados de OptiSalud. En respuesta a las dificultades actuales con los registros manuales, este sistema se ha diseñado con tecnologías avanzadas y metodologías ágiles para asegurar una gestión precisa y eficiente del tiempo laboral. Utilizando el lenguaje de programación C# y el entorno de desarrollo Visual Studio, se ha creado una aplicación de escritorio que permite a los empleados registrar sus entradas y salidas de manera sencilla y segura. La aplicación incluye medidas de seguridad como la validación biométrica y el anclaje a equipos específicos para prevenir fraudes y garantizar la integridad de los datos. El proyecto se ha desarrollado siguiendo una metodología ágil, facilitando una planificación flexible y una respuesta rápida a los requisitos cambiantes. Además, se ha contado con la orientación experta del ingeniero Rafael Pérez Holguín, quien ha proporcionado valiosos aportes para el éxito del proyecto. Este sistema no solo aborda las necesidades inmediatas de OptiSalud en términos de gestión del tiempo, sino que también establece una base sólida para futuras innovaciones en el ámbito de la gestión de recursos humanos. Con un enfoque centrado en la usabilidad, la seguridad y la eficiencia, esta aplicación representa un paso significativo hacia un entorno laboral más organizado y productivo en OptiSalud.

Palabras Claves: Tiempo laboral, Aplicación de escritorio, Seguridad, Metodología ágil, Control de acceso.

Abstract

The project focuses on developing a comprehensive system for recording employee check-ins and check-outs at OptiSalud. In response to the current challenges with manual records, this system has been designed using advanced technologies and agile methodologies to ensure accurate and efficient management of work hours. Using the C# programming language and the Visual Studio development environment, a desktop application was created that allows employees to register their check-ins and check-outs in a simple and secure way. The application includes security measures such as biometric validation and device-specific anchoring to prevent fraud and ensure data integrity. The project was developed following an agile methodology, enabling flexible planning and a quick response to changing requirements. Additionally, the expert guidance of Engineer Rafael Pérez Holguín was instrumental in the success of the project, providing valuable input throughout the process. This system not only addresses OptiSalud's immediate needs in terms of time management but also establishes a solid foundation for future innovations in human resource management. With a focus on usability, security, and efficiency, this application represents a significant step toward a more organized and productive work environment at OptiSalud.

Keywords: Time tracking, Biometric validation, Desktop application, C Sharp, Agile methodology.

Tabla de Contenido

Introducción....20

Justificación....21

Objetivos....22

Objetivos General....22

Objetivos específicos....22

Planteamiento del Problema....23

Alcance....25

Población Objetivo.....25

Marco Teórico....26

Análisis Empresarial Previo de Sociedad de Servicios Oculares OptiSalud SAS.....26

Misión:....26

Visión:....26

Cadena de valor....26

Estrategia:....26

Objetivo:....27

Procesos Intervinientes:....27

Valores Corporativos....27

Trabajo Colaborativo:....27

Convicción:....27

Flexibilidad:....27

Integridad: 27
Servicio con Valor:....28
Objetivos....28
Competencias Organizacionales....28
Experiencia en el servicio:....28
Gestionar el cambio:....28
Resultados superiores:....29
Innovar con creatividad: 29
Procesos....29
Estratégicos (Gerenciales y Directivos): 29
Misionales Realización (UNE): 29
Apoyo Operación (GO):....29
Evaluación, Análisis y Mejora: 30
Procedimientos de la Empresa que Serán Sistematizados....34
Situación Actual: 34
Solución Propuesta:....34
Marco Conceptual 36
Desarrollo de Aplicaciones de Escritorio 36
Aplicaciones de Escritorio:....36
Lenguaje de Programación C# y C# WPF: 36
Visual Studio: 36

Frameworks .NET y Bibliotecas C#: 37
Control de Acceso y Biométrico 37
Control de Acceso: 37
Aplicaciones de Ingreso Biométrico: 37
Inyección SQL: 37
Gestión de Bases de Datos 38
Servidor de Base de Datos: 38
SQL (Structured Query Language): 38
Relación de Tablas: 38
Motor de Base de Datos Microsoft SQL Server Express: 38
Infraestructura Tecnológica 39
Sistema Operativo: 39
Servidor de Base de Datos: 39
Tecnología de Autenticación Biométrica 39
Marco Legal 40
Principios Fundamentales: 40
Consentimiento Informado: 40
Seguridad de Datos: 40
Derechos de los Titulares: 40
Confidencialidad y Secreto Profesional: 40
Sanciones por Incumplimiento: 41

Metodología 42
Metodología de Desarrollo Ágil Individualizado 42
Tipo de Investigación del Proyecto: Descriptiva y Experimental 43
Descriptiva: 43
Experimental: 44
Fases del Proyecto en relación con los Objetivos Específicos 44
Análisis de Requisitos y Especificaciones Técnicas: 44
Diseño del Sistema: 44
Desarrollo del Software: 45
Plan de Pruebas: 45
Análisis de resultados de la encuesta sobre la viabilidad del proyecto grado 46
Análisis General de Resultados: 51
Conclusión y Viabilidad del Proyecto 52
Planificación 53
Fase de Planificación 53
Actividad 1: Planificación y análisis (8 agosto 2023 - 27 agosto 2023) 53
Fase de Diseño 53
Actividad 2: Definición de requisitos del sistema (28 agosto 2023 - 5 septiembre 2023) 53
Actividad 3: Diseño de la interfaz de usuario (6 septiembre 2023 - 20 septiembre 2023) 53
Fase de Desarrollo 53

Actividad 4: Diseño de la arquitectura de software (21 septiembre 2023 - 30 septiembre 2023) 53
Actividad 5: Desarrollo del software (1 octubre 2023 - 29 diciembre 2023) 53
Fase de Pruebas 54
Actividad 6: Pruebas unitarias y de integración (30 diciembre 2023 – 8 enero 2023) 54
Actividad 7: Pruebas de sistema y correcciones (9 enero 2024 - 18 enero 2024) 54
Fase de Finalización 54
Actividad 8: Preparación para el lanzamiento (19 enero 2024 - 28 enero 2024) 54
Actividad 9: Lanzamiento y seguimiento (29 enero 2024 - 7 febrero 2024) 54
Presupuesto 56
Diseño 59
Diagrama de Flujos 59
Descripción del proceso de registro de asistencia para empleados: 60
Descripción del proceso de autenticación en el sistema: 61
Descripción del proceso de gestión de registros: 62
Descripción del proceso de generación y exportación de informes: 64
Descripción del proceso de gestión de usuarios: 66
Diagrama de casos de usos 67
Actor: Empleados 67
Explicación para el diagrama de casos de uso: registro de Entrada/Salida con Huella Dactilar 68

Actor: Empleados con privilegios....68

Explicación para el diagrama de casos de uso: Sistema de gestión para empleados con privilegios. 69

Diagrama de Dominio 70

Comportamiento del Sistema Basado en el Diagrama....71

Diagrama de clases 71

Diagrama de objetos....73

Diagramas de secuencias....75

Diagrama de componentes de arquitectura MVC....83

Usuario (Empleado o Administrador): 83

Interfaz de Usuario (Vista): 84

Controlador: 84

Modelo: 84

Base de Datos:....85

Modelo entidad – relación. 86

Entidades y Atributos....86

Relaciones 88

Procedimientos almacenados....89

Desarrollo y función de la aplicación 93

Clases....95

ConfiguracionBD 95

DatoEmpleado....95

Empleado 96
NumeroTemp 97
ServicioUsuario 97
UIAdmin 98
Usuarios 98
ViewModel 99
Estilos y recursos 99
Formularios 102
Login 102
CaptureHuella 103
EnrollmentWindow 105
InicioSesion 107
Administración 109
Pag_Registro 110
Informe 112
Users 114
CambioPass 115
Árbol de carpetas del proyecto 117
Descripción de Carpetas y Archivos 119
Dependencias 119
Marcos de trabajo y Paquetes 119

Properties....119
Contenidos....119
Estilos....119
FormsPrincipales....119
Huella....120
Imágenes....120
Mensajes....120
Pruebas....120
App.config....120
App.xaml....120
ConfiguracionBD.cs....121
DatoEmpleado.cs....121
Empleado.cs....121
Ingreso.xaml....121
NumeroTemp.cs....121
ServicioUsuario.cs....121
UIAdmin.cs....121
Usuarios.cs....122
ViewModel.cs....122
Repositorio del proyecto....122
Acceso a proyecto....122

Conclusiones.....123

Lista de referencias 125

Anexos 127

Encuesta de Viabilidad del Sistema de Registro Automático en OptiSalud. 127

Lista de Tablas

Tabla 1 Respuesta a la pregunta 01 46

Tabla 2 Respuesta a la pregunta 02 48

Tabla 3 Respuesta a la pregunta 03 49

Tabla 4 Respuesta a la pregunta 04 50

Tabla 5 Respuesta a la pregunta 05 51

Tabla 6 Consulta y Metodología.....56

Tabla 7 Recursos Humanos.....56

Tabla 8 Recursos Técnicos.....56

Tabla 9 Gastos Operativos 57

Tabla 10 Gastos adicionales.....57

Tabla 11 Muestra General de Presupuesto.....58

Lista de Figuras

Figura 1 Mapa de Procesos....30

Figura 2 Modelo Evaluación por Competencias Familias de Cargos 30

Figura 3 Organigrama Gerente General 31

Figura 4 Organigrama Subgerente Institucional....32

Figura 5 Diagrama de respuesta a la pregunta 01....46

Figura 6 Diagrama de respuesta a la pregunta 02....47

Figura 7 Diagrama de respuesta a la pregunta 03....48

Figura 8 Diagrama de respuesta a la pregunta 04....49

Figura 9 Diagrama de respuesta a la pregunta 05....50

Figura 10 Diagrama de actividades 54

Figura 11 Diagrama de flujo del proceso de registro de asistencia para empleados 59

Figura 12 Diagrama de flujo del proceso de autenticación en el sistema....60

Figura 13 Diagrama de flujo del proceso de gestión de registros....61

Figura 14 Diagrama de flujo del proceso de generación y exportación de informes....63

Figura 15 Diagrama de flujo del proceso de gestión de usuarios....65

Figura 16 Diagrama de casos de uso: registro de Entrada/Salida con Huella Dactilar...67

Figura 17 Diagrama de Casos de Uso: Sistema de gestión para empleados con privilegios....68

Figura 18 Diagrama de dominio....70

Figura 19 Diagrama de clases....71

Figura 20 Diagrama de objetos 73

Figura 21 Diagrama de Secuencia para Registro de Asistencia Biométrica (Empleado)75

Figura 22 Diagrama de secuencia para inicio de sesión (Administrador).....77

Figura 23 Diagrama de secuencia para gestión de usuarios (CRUD).....78

Figura 24 Diagrama de secuencia para exportación de informes 80

Figura 25 Diagrama de secuencia para cambio de contraseña (Administrador) 81

Figura 26 Diagrama de arquitectura MVC 83

Figura 27 Modelo entidad – relación.....88

Figura 28 Código MySql, actualiza empleado.....89

Figura 29 Código MySql, actualiza empleado huella 90

Figura 30 Código MySql, insertar empleado.....91

Figura 31 Código MySql, muestra empleado.....91

Figura 32 Código MySql, obtener empleado.....92

Figura 33 Código MySql, obtener usuarios.....92

Figura 34 Muestra de dependencias utilizadas.....93

Figura 35 Estructura de datos de la aplicación 93

Figura 36 Muestra de código C# de la clase, conexión base de datos 95

Figura 37 Muestra de código C# de la clase, dato empleado 96

Figura 38 Muestra de código C# de la clase, empleados 96

Figura 39 Muestra de código C# de la clase, numero temporal 97

Figura 40 Muestra de código C# de la clase, servicios con usuarios 97

Figura 41 Muestra de código C# de la clase, UI Admin 98

Figura 42 Muestra de código C# de la clase, usuarios 98

Figura 43 Muestra de código C# de la clase, view model 99

Figura 44 Muestra de código XMAL, estilos y recursos 101

Figura 45 Muestra de código C# de módulo, Login 102

Figura 46 Módulo Login con huella dactilar 102

Figura 47 Módulo de registro de horario de empleados 103

Figura 48 Muestra de código C# de módulo, captura de huella 104

Figura 49 Módulo para captura de huella dactilar 105

Figura 50 Muestra de código C# de módulo, de alojamiento de huella 105

Figura 51 Muestra de código C# de módulo, inicio de sesión 107

Figura 52 Módulo de inicio de sesión 108

Figura 53 Modulo administración 109

Figura 54 Muestra de código C# de módulo, administración 110

Figura 55 Modulo registro de empleado 110

Figura 56 Módulo de creación, edición, de empleado 111

Figura 57 Muestra de código C# de módulo, registro de empleado 112

Figura 58 Módulo informe 112

Figura 59 Muestra de código C# de módulo, servicios con usuarios 113

Figura 60 Módulo de registro de usuarios 114

Figura 61 Muestra de código C# de módulo, usuarios 115

Figura 62 Módulo de cambio de contraseña 115

Figura 63 Muestra de código C# de módulo, cambiar contraseña 116

Introducción

En el mundo empresarial actual, gestionar de manera precisa y eficiente el tiempo laboral es crucial para el buen funcionamiento de cualquier organización. OptiSalud se enfrenta al desafío de mejorar el registro de las entradas y salidas de sus empleados en su sede administrativa. La necesidad de un sistema robusto y confiable que elimine errores y prevenga fraudes es esencial para asegurar la integridad y la transparencia de los datos.

Este proyecto responde a la necesidad crítica de optimizar la administración del recurso humano en OptiSalud a través de un software de registro de entrada y salida. Desarrollado bajo una metodología ágil, el proyecto busca automatizar el proceso de registro y garantizar la precisión de los datos mediante técnicas de validación biométrica y análisis estadístico.

En este proyecto se exploran los desafíos actuales en la gestión del tiempo, las tecnologías disponibles para mejorar la precisión del registro laboral y la implementación de un sistema integral que resuelva estas dificultades. Bajo la guía del ingeniero Rafael Pérez Holguín, esta iniciativa representa un paso importante hacia la eficiencia operativa y la fiabilidad en la gestión de recursos humanos en OptiSalud.

En este contexto, el trabajo se centrará en la planificación, diseño e implementación de un sistema de registro de entrada y salida que no solo responde a las necesidades específicas de OptiSalud, sino que también destaca por su dedicación y excelencia en el desarrollo de software para entornos empresariales.

Justificación

Este proyecto es fundamental para optimizar el proceso de registro de entrada y salida de los empleados en la sede administrativa de OptiSalud. La implementación de un sistema digitalizado no solo asegura precisión y transparencia en el registro de las horas laborales, minimizando errores y promoviendo la integridad en los datos, sino que también mejora la eficiencia operativa al reducir la carga administrativa y permitir que los recursos humanos se concentren en tareas estratégicas. Además, el uso de técnicas biométricas fortalece la seguridad y previene fraudes, garantizando la autenticidad de los registros y generando confianza en el sistema.

La adaptabilidad y escalabilidad del sistema permiten que crezca junto con OptiSalud, manteniéndose alineado con futuros cambios y expansiones. Esta iniciativa no solo resuelve necesidades inmediatas, sino que posiciona a OptiSalud como un referente en innovación y competitividad, fortaleciendo su presencia en el mercado y preparándola para afrontar los desafíos laborales emergentes.

En resumen, este proyecto representa una inversión estratégica que optimiza los recursos, mejora la precisión y establece una base tecnológica sólida para el futuro de OptiSalud. Su viabilidad está respaldada por la integración de tecnologías avanzadas, prácticas eficientes y la capacidad del sistema para evolucionar de acuerdo con las necesidades cambiantes de la organización.

Objetivos

Objetivos General

Desarrollar el sistema SCESO-Soft para proporcionar información confiable sobre los horarios de trabajo de los empleados de OptiSalud.

Objetivos específicos

Analizar los requisitos y especificaciones técnicas para el desarrollo del sistema SCESO-Soft.

Diseñar el sistema que permitirá sistematizar el control de horarios de trabajo de los empleados de OptiSalud.

Desarrollar la solución de software para controlar las entradas y salidas de los empleados de OptiSalud.

Elaborar un plan de pruebas para la depuración y mejora del proyecto.

Planteamiento del Problema

OptiSalud enfrenta un problema significativo en la obtención de información precisa y completa sobre los registros de entrada y salida de sus empleados. Actualmente, los métodos de recolección de datos varían entre las diferentes sedes, lo que genera información inconsistente y no confiable. Esto ha resultado en una falta de uniformidad en los registros a lo largo del tiempo.

La ausencia de un modelo unificado para registrar la entrada y salida de los empleados ha llevado a la falta de información consolidada y precisa, lo que dificulta el análisis, el seguimiento y la toma de decisiones. Sin indicadores claros y cuantificables, la evaluación de la asistencia y el rendimiento laboral, así como la detección de anomalías, se vuelve ineficiente.

La falta de un sistema centralizado y estandarizado para el registro ha generado ineficiencias en la gestión de recursos humanos. Esto ha impedido la identificación de patrones de asistencia, el reconocimiento de retrasos o ausencias frecuentes, y el cálculo preciso de las horas trabajadas. Como consecuencia, la empresa ha experimentado pérdidas en productividad y recursos.

Para ilustrar la magnitud del problema, se identifican los siguientes puntos clave:

- Inconsistencias en los registros: Las discrepancias en los registros de entrada y salida entre sedes han generado un desequilibrio en la información, dificultando el seguimiento preciso de la asistencia.
- Duplicidad de registros: Se han detectado casos en los que los empleados registran múltiples veces su entrada o salida, lo que impide tener una visión clara de las horas trabajadas y complica el cálculo de la remuneración.
- Dificultades en la generación de informes: El proceso manual de recolección y consolidación de datos impide generar informes y análisis eficientes sobre la asistencia y el rendimiento laboral, retrasando la toma de decisiones.
- Ausencia de medidas de seguridad: La falta de un sistema robusto ha permitido fraudes, como el registro de horas ficticias, ya que no existen mecanismos efectivos para evitar estos comportamientos.

La implementación del software propuesto busca resolver estos problemas y mejorar considerablemente la gestión de los registros de entrada y salida en OptiSalud. Se espera lograr una mayor precisión en los datos, simplificar su análisis y facilitar la toma de decisiones basadas en información actualizada y confiable.

Alcance

Población Objetivo.

El proyecto de desarrollo de la aplicación SCESO-Soft se enfoca en los empleados de la sede administrativa principal de OptiSalud en Yopal, conformada por aproximadamente 60 empleados. Este grupo es el núcleo del personal administrativo y desempeña funciones cruciales en el funcionamiento diario de la organización.

El objetivo principal es optimizar la precisión y eficiencia en el registro de las horas de trabajo, incluyendo el control del horario de almuerzo. Además, se implementarán medidas de seguridad y validación para garantizar la integridad de los datos y prevenir fraudes.

La implementación exitosa de este proyecto beneficiará directamente a los empleados al simplificar y agilizar el proceso de registro, contribuyendo a una gestión más eficiente de la asistencia y del rendimiento laboral. También se espera un impacto positivo en la toma de decisiones basada en datos por parte del equipo de recursos humanos de OptiSalud, mejorando la gestión de personal en la sede principal.

El proyecto no solo contempla la implementación del software, sino también la capacitación de los empleados para su uso efectivo. Se espera que esta capacitación permita a todos los miembros de la población objetivo aprovechar al máximo las funcionalidades del sistema, optimizando su desempeño y contribuyendo a la mejora operativa de la empresa.

Marco Teórico

Análisis Empresarial Previo de Sociedad de Servicios Oculares OptiSalud SAS.

OptiSalud forma parte del sector de prestación de servicios de salud en el área de la visión y desarrolla su misión bajo los principios de integralidad y generación de valor. La atención que ofrece está dividida en dos canales: el canal comercial, encargado de la venta de lentes, anteojos y gafas para la población en general, y el canal institucional, vinculado al sistema de seguridad social, que ofrece servicios de optometría, oftalmología, supra especialidades, y otros tratamientos médicos. Además, incluye atención domiciliaria, terapia física y rehabilitación para pacientes afiliados a las EPS con las que mantiene contratos.

Los dos canales cuentan con la oferta de servicios y productos para la protección visual y constituyen así las 2 "Unidades de Negocio Estratégicas (UNE)" de OptiSalud.

Misión:

Proporcionar una atención integral para la salud visual y ocular, con altos estándares de calidad y un talento humano competente desde el ser y el hacer. Haciendo uso de tecnologías e infraestructura, innovadoras y eficientes, que contribuyen a mejorar la calidad de vida y experiencia de los usuarios.

Visión:

Incrementar nuestra cobertura en los departamentos presencia de con OptiSalud, integrando IOS centros satélites, articulando las UNE para una expansión rentable y sostenible, consolidando nuestro liderazgo en el sector.

Cadena de valor

Estrategia:

Permitir un acceso a los servicios y productos de forma ágil, integral y sostenible; considerando el entorno competitivo. Se sugiere la siguiente cadena de valor.

Objetivo:

Asegurar el cumplimiento de la misión, realizando seguimientos periódicos a la gestión de los procesos y el cumplimiento de manera conjunta, por gestión de un propósito.

Procesos Intervinientes:

Estratégico, Misional, Evaluación y Apoyo (Calidad, Gestión Humana, Experiencia)

Valores Corporativos

Trabajo Colaborativo:

Trabamos en equipo para garantizar un desempeño costo eficiente desde los diferentes niveles de responsabilidad. Los resultados individuales y colectivos en cada proceso responden a la calidad y oportunidad productiva para cumplir con los objetivos de la organización.

Convicción:

Nos enorgullece trabajar en OptiSalud, por eso estamos comprometidos con ser parte de su reconocimiento y reputación. Nos identificamos con los valores familiares y el sentido de la marca, por eso nuestras actitudes y comportamientos están motivados por una sinergia positiva entre las diferentes poblaciones de interés.

Flexibilidad:

Comprendemos el crecimiento y fortalecimiento de las UNE, la diversidad política, geográfica y cultural de los territorios donde hacemos presencia, la visión de expansión, las demandantes y cambiantes necesidades de las EPS, los usuarios y los clientes, acondicionando nuestra infraestructura física, humana y tecnológica, por eso tenemos apertura al cambio y asumimos con responsabilidad nuevos retos personales y laborales.

Integridad:

En OptiSalud entregamos valor a nuestros aliados, proveedores, prestadores, usuarios y clientes. Por eso creamos espacios diferenciales y confortables, adquirimos productos,

herramientas y personal con altos estándares de calidad y confiabilidad, diseñamos modelos de atención para garantizar más que un servicio accesible: una experiencia de bienestar y salud.

Servicio con Valor:

Aquí cobra sentido el liderazgo de OptiSalud, como principio fundamental en el abordaje integral del ser humano con empatía, asertividad, agilidad, flexibilidad, respeto e inclusión.

Objetivos

Contexto de sostenibilidad redefinición de estrategias para adaptar la gestión a los cambios del entorno, el crecimiento y continuar la prestación óptima del servicio; con lo cual se garantice la perdurabilidad de la organización, respondiendo en equilibrio por sus impactos sociales y económicos a través de una estructura y gestión organizacional sólida.

Alineado con la historia de OptiSalud y SER, se prioriza proteger el empleo de los trabajadores, el cuidado de la salud y el bienestar de los equipos de trabajo y los usuarios. Considerando la transformación tecnológica y del talento humano para soportar la gestión y mejorar los modelos de atención.

Competencias Organizacionales

Un conjunto de comportamientos conductas deben y que caracterizar el desempeño y actuaciones para afrontar las situaciones de la Vida laboral a través de la gestión del desarrollo y la incorporación.

Experiencia en el servicio:

Procura soluciones sorprendentes, dejando huella en cliente interno y externo con el interés genuino de satisfacer las necesidades y expectativas de los demás.

Gestionar el cambio:

Esta presto(a) a los cambios con buena actitud e interés a través de la autogestión e iniciativa para generar ideas y soluciones.

Resultados superiores:

Aporta significativamente a los objetivos y tiende a mejorar. Plantea retos, es persistente, solicita ayuda si lo requiere, al final logra superar expectativas.

Innovar con creatividad:

Se vincula emocionalmente con las funciones asignadas, por tanto, busca constantemente nuevos modelos y entornos de relacionamiento y gestión.

Procesos

Estratégicos (Gerenciales y Directivos):

Definen la estrategia de la organización, permiten la creación de metas para obtener resultados a largo plazo. Se consideran factores clave para fijar políticas, estrategias y objetivos.

Misionales Realización (UNE):

Definen la misión y el objetivo de la empresa, hacen realidad la satisfacción de las necesidades del usuario, con la realización del producto o la prestación del servicio. Son considerados fundamentales en la operación: 1. Ingreso al usuario al servicio asistencial, 2. Prestación de servicios asistenciales, egreso al usuario al servicio asistencial y servicio de apoyo al proceso de servicio asistencial.

Apoyo Operación (GO):

Son aquellos que apoyan los procesos estratégicos, misionales y de monitoreo y evaluación. Generalmente están dirigidos a provisionar los recursos y orden para los demás procesos.

Evaluación, Análisis y Mejora:

Son los procesos que miden, recopilan datos y analizan para mejorar el desempeño, la eficacia y eficiencia haciendo el seguimiento necesario, para tomar acciones correctivas y preventivas, como parte integral de los procesos estratégicos, de apoyo y los misionales.

En la siguiente Figura 1 se explica el mapa de procesos:

Figura 1

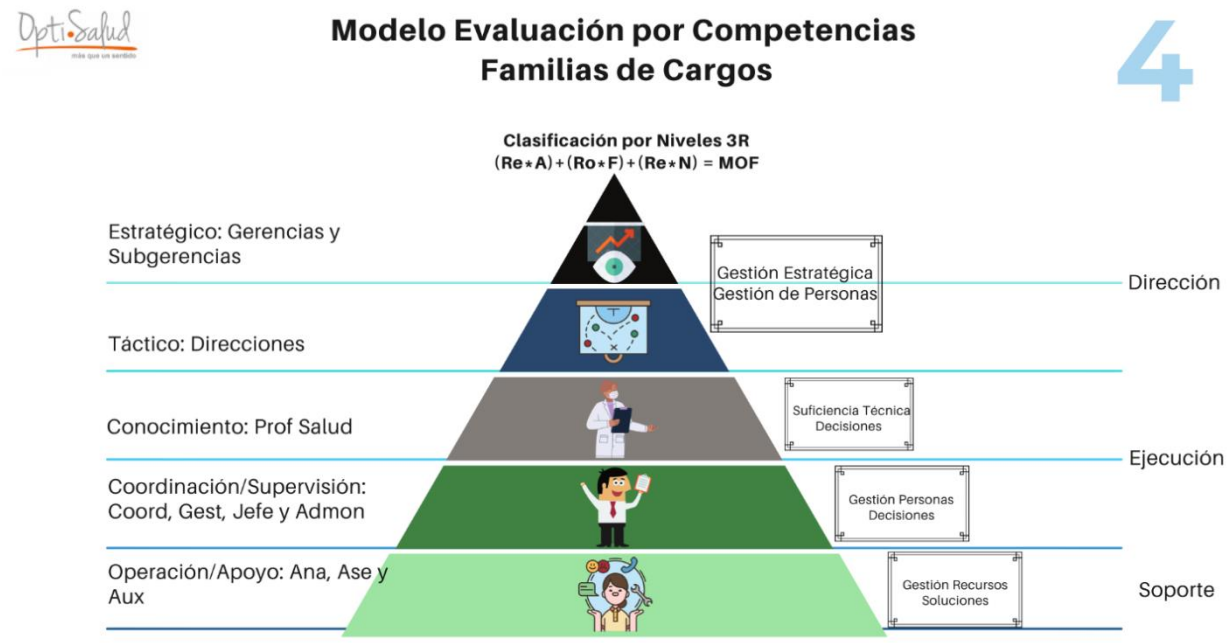
Mapa de Procesos.

Fuente: https://optisalud.com/wp-content/uploads/2023/11/Brochure-Corporativo_20231024_130812_0000.pdf

Nota. Tomada de Brochure Corporativo, Mapa de Procesos de OptiSalud, 2023.

Figura 2

Modelo Evaluación por Competencias Familias de Cargos.

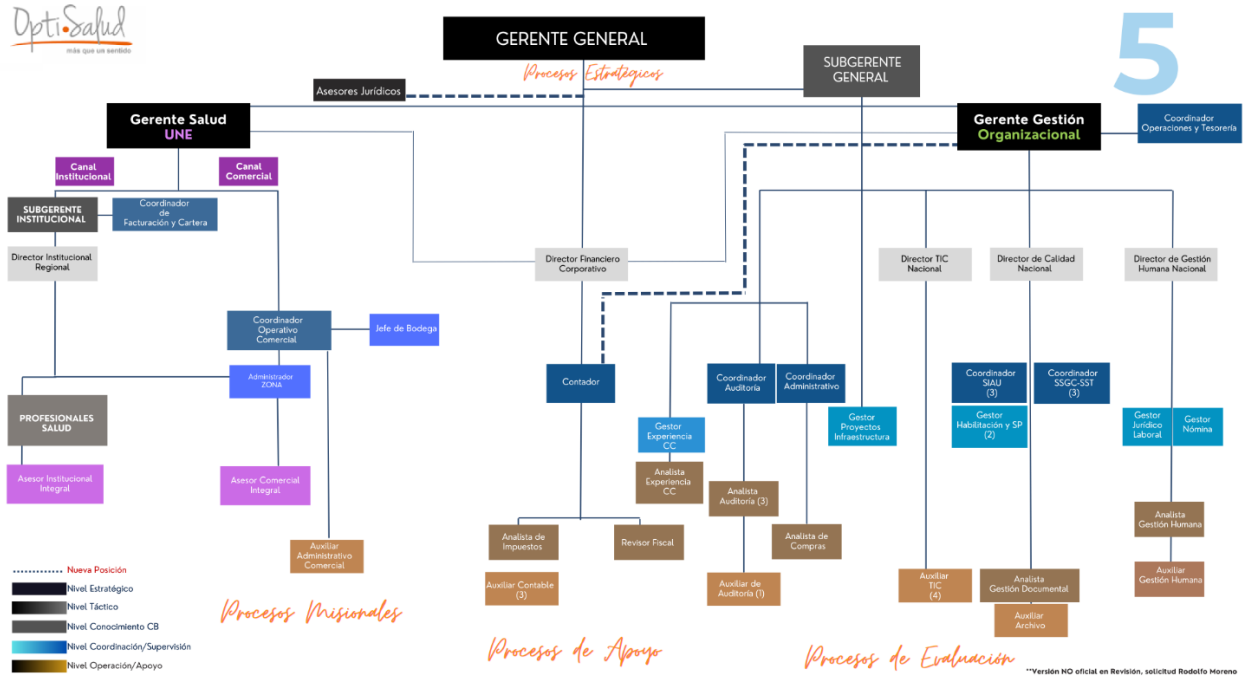


Fuente: https://optisalud.com/wp-content/uploads/2023/11/Brochure-Corporativo_20231024_130812_0000.pdf

Nota. Tomada de Brochure Corporativo, Evaluación por Competencias Familias de Cargos de OptiSalud, 2023.

Figura 3

Organigrama Gerente General.

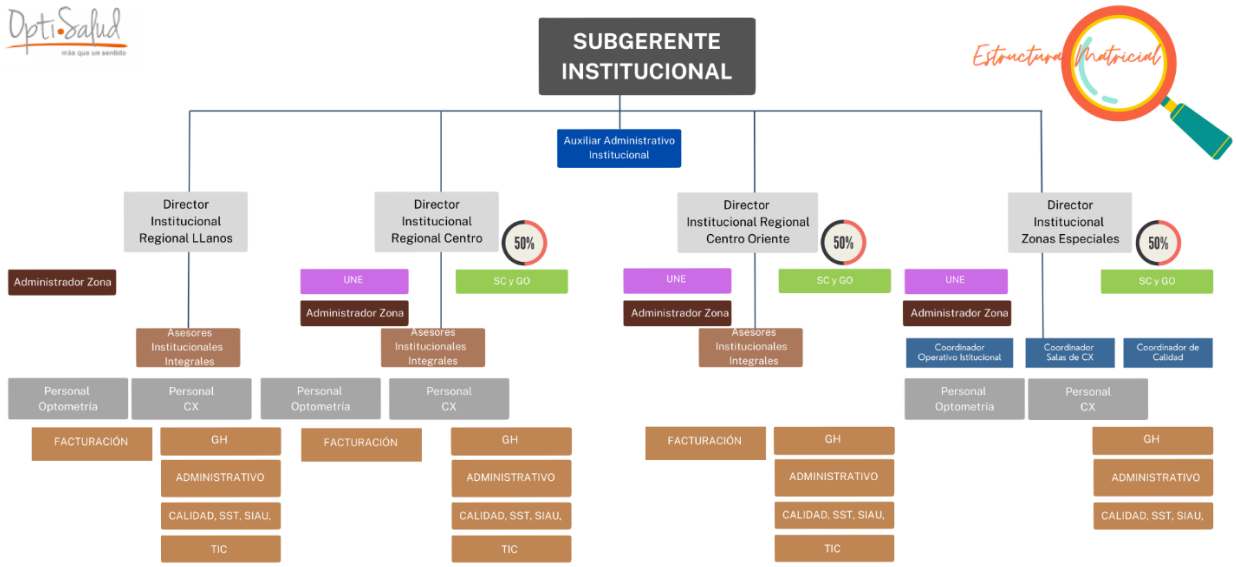


Fuente: https://optisalud.com/wp-content/uploads/2023/11/Brochure-Corporativo_20231024_130812_0000.pdf

Nota. Tomada de Brochure Corporativo, Organigrama Gerente General de OptiSalud, 2023.

Figura 4

Organigrama Subgerente Institucional.



Fuente: https://optisalud.com/wp-content/uploads/2023/11/Brochure-Corporativo_20231024_130812_0000.pdf

Nota. Tomada de Brochure Corporativo, Organigrama Subgerente Institucional de OptiSalud, 2023.

Procedimientos de la Empresa que Serán Sistematizados

Situación Actual:

En OptiSalud, los procesos manuales y descentralizados para el registro de entrada y salida de los empleados generan inconsistencias y falta de uniformidad en la información recopilada. Esta situación complica la evaluación de la asistencia y el rendimiento laboral, además de aumentar el riesgo de fraudes debido a la ausencia de medidas de seguridad efectivas. Asimismo, la generación de informes se ve afectada por la falta de un sistema centralizado, lo que dificulta la toma de decisiones basada en datos confiables y actualizados.

Solución Propuesta:

El proyecto SCESO-Soft ofrece una solución digitalizada y centralizada que mejora de manera integral estos procesos. Mediante el uso de una aplicación desarrollada en C#, los empleados podrán registrar sus horas de trabajo de manera precisa y sencilla. Además, se incorporarán medidas de seguridad, como la validación biométrica, para prevenir fraudes y garantizar la autenticidad de los registros.

La implementación de SCESO-Soft también permitirá:

- Seguridad mejorada: Técnicas biométricas garantizarán la autenticidad de los registros, eliminando la posibilidad de fraudes como el registro de horas ficticias.
- Eficiencia en la generación de informes: El uso de una base de datos en SQL Server facilitará la creación de informes precisos y en tiempo real, mejorando el análisis de la asistencia y el rendimiento laboral.
- Escalabilidad y adaptabilidad: SCESO-Soft se diseñará con la capacidad de ajustarse a futuros cambios y expansiones, asegurando que el sistema crezca junto con las necesidades de la empresa.

Marco Conceptual

El marco conceptual del proyecto se basa en los principios y tecnologías que sustentan el desarrollo del sistema de registro de entrada y salida para los empleados de OptiSalud. Este apartado define los conceptos clave que guiarán el diseño y la implementación del sistema. A continuación, se describen los principales componentes del marco conceptual:

Desarrollo de Aplicaciones de Escritorio

Aplicaciones de Escritorio:

Las aplicaciones de escritorio son ideales para entornos empresariales que requieren interfaces de usuario intuitivas y funcionales. Ofrecen un alto rendimiento y control directo sobre los recursos del sistema, lo que las convierte en una opción adecuada para el manejo eficiente de datos sensibles y voluminosos. En el contexto de OptiSalud, una aplicación de escritorio garantiza estabilidad, seguridad y rapidez en el registro y procesamiento de la información relacionada con las horas de trabajo, minimizando la dependencia de conexiones a internet y mejorando la experiencia del usuario en un entorno controlado.

Lenguaje de Programación C# y C# WPF:

C# es un lenguaje de programación desarrollado por Microsoft utilizado para implementar la lógica de la aplicación y la conexión con la base de datos. Se utilizará en conjunto con Windows Presentation Foundation (WPF) para crear interfaces de usuario atractivas y funcionales para las aplicaciones de escritorio.

Visual Studio:

Visual Studio es un entorno de desarrollo integrado que proporciona herramientas y características para escribir, depurar y compilar aplicaciones de software. Es una herramienta esencial para el desarrollo eficiente y la gestión del proyecto.

Frameworks .NET y Bibliotecas C#:

Los Frameworks .NET y las bibliotecas C# proporcionan funcionalidades predefinidas que aceleran el desarrollo y mejoran la calidad del software. Estos incluyen ASP.NET Core para desarrollo web, Unity para juegos multiplataforma, Xamarin para aplicaciones móviles y Windows Presentation Foundation (WPF) para interfaces de usuario de aplicaciones de escritorio.

Control de Acceso y Biométrico

Control de Acceso:

El control de acceso es una medida de seguridad que garantiza que solo el personal autorizado pueda acceder a recursos específicos dentro de una organización. Se basa en la autenticación, que puede realizarse a través de métodos tradicionales, como contraseñas, o mediante tecnologías más avanzadas, como sistemas biométricos.

Aplicaciones de Ingreso Biométrico:

Estas aplicaciones utilizan características físicas únicas, como huellas dactilares o reconocimiento facial, para verificar la identidad de los empleados. Esto mejora la autenticación durante el registro de entrada y salida, proporcionando un nivel adicional de seguridad.

Inyección SQL:

La inyección de código es una vulnerabilidad de seguridad que surge cuando no se validan correctamente los datos antes de incluirlos en una consulta a la base de datos. Esta brecha permite que usuarios malintencionados alteren el comportamiento de las consultas, lo que puede comprometer la integridad de la información almacenada. Para evitar estos riesgos, es fundamental aplicar mecanismos de validación y filtrado de datos, asegurando que solo entradas autorizadas interactúen con el sistema de gestión de bases de datos.

Gestión de Bases de Datos

Servidor de Base de Datos:

Un servidor de base de datos es una plataforma que permite el almacenamiento y gestión de grandes volúmenes de información de manera estructurada y eficiente. Utiliza un Sistema de Gestión de Bases de Datos para organizar, acceder y recuperar los datos de forma rápida y segura. Existen distintos tipos de servidores, como bases de datos relacionales, NoSQL y en la nube, cada uno con características específicas que responden a diferentes necesidades de almacenamiento y procesamiento de datos.

SQL (Structured Query Language):

SQL es un lenguaje de programación esencial para gestionar bases de datos relacionales. Permite realizar consultas, actualizaciones y modificaciones en la información almacenada en la base de datos. Además, asegura la eficiencia y facilita la manipulación de datos de manera precisa.

Relación de Tablas:

Las relaciones entre tablas permiten vincular y recuperar datos de múltiples tablas, evitando la duplicación de datos y mejorando la integridad y flexibilidad de estos. Se implementarán relaciones uno a uno, uno a varios y varios a varios para mantener la coherencia de los registros de entrada y salida.

Motor de Base de Datos Microsoft SQL Server Express:

En el servidor físico, se implementó Microsoft SQL Server Express como motor de base de datos. Esta elección se basa en la confiabilidad, escalabilidad y capacidad de SQL Server Express para gestionar grandes volúmenes de datos. Aunque es una versión gratuita, ofrece un rendimiento sólido y una serie de características avanzadas, como soporte para transacciones, consultas complejas y seguridad de datos. Esto lo hace adecuado para aplicaciones empresariales que requieren un almacenamiento de datos eficiente y seguro.

Infraestructura Tecnológica

Sistema Operativo:

Un sistema operativo es el software que permite la gestión de los recursos del hardware y proporciona una plataforma estable para la ejecución de aplicaciones. Windows 10 Pro, como sistema operativo, ofrece un entorno confiable y compatible con diversas herramientas de desarrollo y gestión empresarial, lo que facilita la implementación de soluciones tecnológicas robustas.

Servidor de Base de Datos:

Un servidor físico dedicado a la gestión de bases de datos proporciona un entorno seguro y eficiente para almacenar grandes volúmenes de información. Al centralizar los datos en un solo servidor, se mejora el control de acceso, el rendimiento y la seguridad de la información, garantizando que la infraestructura pueda crecer conforme a las necesidades de la organización.

Tecnología de Autenticación Biométrica

La autenticación biométrica es un método de verificación de identidad basado en características físicas únicas, como las huellas digitales. Esta tecnología garantiza un control de acceso seguro y preciso, reduciendo el riesgo de fraudes y mejorando la eficiencia en la gestión de asistencia y tiempo laboral. Los dispositivos biométricos permiten una verificación rápida y no intrusiva, lo que los convierte en una solución ideal para entornos empresariales donde la seguridad y la precisión son prioritarias.

Marco Legal

El desarrollo del proyecto de control de registro de entrada y salida de empleados de OptiSalud está respaldado por la legislación colombiana, en particular, la Ley Estatutaria 1581 de 2012, la cual regula el manejo y protección de datos personales en Colombia. De acuerdo con esta ley, se establecen los principios, derechos y procedimientos para garantizar la privacidad y seguridad de la información personal.

Principios Fundamentales:

El proyecto se alinea con los principios fundamentales de la Ley 1581, tales como el principio de legalidad, finalidad, libertad, veracidad o calidad, transparencia, acceso y circulación restringida, seguridad y confidencialidad.

Consentimiento Informado:

El sistema implementado asegurará que los empleados de OptiSalud otorguen su consentimiento informado para el registro de sus datos de entrada y salida. Se respetará el derecho a la autodeterminación informativa de cada individuo.

Seguridad de Datos:

El proyecto garantizará la implementación de medidas técnicas, humanas y administrativas necesarias para otorgar seguridad a los registros, evitando su adulteración, pérdida, consulta, uso o acceso no autorizado.

Derechos de los Titulares:

El sistema permitirá a los empleados ejercer sus derechos como titulares de datos personales, incluyendo el derecho a conocer, actualizar, rectificar y suprimir la información.

Confidencialidad y Secreto Profesional:

Se preservará la confidencialidad de los datos recolectados, garantizando el secreto profesional y adoptando medidas para evitar su acceso por parte de terceros no autorizados.

Sanciones por Incumplimiento:

El proyecto estará sujeto a las sanciones establecidas por la ley en caso de incumplimiento de las disposiciones sobre protección de datos personales.

Este proyecto se desarrollará en estricto cumplimiento de la Ley Estatutaria 1581 de 2012 y demás normativas colombianas aplicables, asegurando así el respeto de los derechos fundamentales de privacidad y protección de datos de los empleados de OptiSalud.

Metodología

Metodología de Desarrollo Ágil Individualizado

Este proyecto sigue un enfoque de desarrollo ágil personalizado, donde un único desarrollador asume los roles de Scrum Master, Product Owner y Equipo de Desarrollo. La metodología se ajusta a un entorno individual, pero mantiene los principios fundamentales del desarrollo ágil. Se organiza el trabajo en sprints (iteraciones cortas de una a dos semanas), lo que permite entregar resultados incrementales y adaptarse a cambios de manera rápida y efectiva.

Los eventos clave en esta metodología incluyen:

- Sprint Planning: Al inicio de cada iteración, se define qué tareas del Product Backlog se abordarán en el sprint, priorizando las más críticas para el avance del proyecto.
- Daily Standup: Aunque en este contexto no hay un equipo para reportar, el desarrollador realiza una revisión diaria del progreso, ayudándose de herramientas como un tablero Kanban para gestionar las tareas.
- Sprint Review: Al final de cada sprint, se revisan los resultados y se recoge retroalimentación de los interesados clave, ajustando el Product Backlog para las siguientes iteraciones.
- Sprint Retrospective: Después de cada entrega, el desarrollador reflexiona sobre las mejoras posibles en el proceso y las implementa en la siguiente iteración.

El uso de esta metodología ofrece varios beneficios:

- Entrega más rápida de valor: Al trabajar en iteraciones cortas y con un enfoque en la funcionalidad esencial, el cliente recibe incrementos funcionales del producto con mayor rapidez.
- Adaptación constante a los cambios: La retroalimentación continua permite ajustar las prioridades y características del producto en tiempo real, lo que asegura que el proyecto esté alineado con las necesidades emergentes del cliente.
- Mayor calidad del producto: A través de pruebas rigurosas en cada fase del desarrollo y una refactorización continua del código, se minimizan los defectos y se mejora la estructura del software.

Tipo de Investigación del Proyecto: Descriptiva y Experimental

Descriptiva:

- Descripción detallada de los requisitos y especificaciones técnicas:

En esta fase, se llevará a cabo una investigación exhaustiva para identificar y detallar todos los requisitos que el sistema SCESO-Soft debe cumplir. Esto incluirá tanto los aspectos funcionales como no funcionales del software, además de los requisitos técnicos específicos. El objetivo es obtener una comprensión completa de lo que se espera lograr con el sistema.

- Análisis del sistema actual de registro de entrada y salida en OptiSalud:

Se realizará un análisis detallado del sistema actual de registro de entrada y salida de los empleados en OptiSalud. Esto incluirá la identificación de los procesos y procedimientos existentes, así como las deficiencias y limitaciones del sistema actual. Este análisis servirá como base para el diseño y desarrollo de SCESO-Soft, asegurando que el nuevo sistema aborde y mejore las deficiencias identificadas

Experimental:

- Diseño y desarrollo de un sistema experimental, SCESO-Soft:

En esta fase se implementará el sistema propuesto, traduciendo los requisitos y especificaciones técnicas en un diseño tangible y funcional, el software SCESO-Soft. El término "experimental" se refiere a que este sistema es nuevo para OptiSalud, y su implementación representa una prueba directa de su funcionamiento y eficacia.

- Pruebas y mejora iterativa del sistema en un entorno real:

Una vez implementado, SCESO-Soft se someterá a pruebas en un entorno operativo real. Estas pruebas incluirán casos de uso tanto simulados como reales para evaluar el rendimiento del sistema, la precisión en el registro de los horarios de entrada y salida, así como la efectividad de las medidas de seguridad implementadas. Con base en los resultados obtenidos, el sistema se depurará y mejorará de forma iterativa y continua, garantizando su eficacia y eficiencia.

Fases del Proyecto en relación con los Objetivos Específicos

Análisis de Requisitos y Especificaciones Técnicas:

En esta fase inicial, se realizará una investigación detallada para comprender completamente los requisitos del sistema, lo que implica identificar todas las funciones y características necesarias. La documentación resultante servirá como base sólida para el diseño y desarrollo subsiguiente.

Diseño del Sistema:

Con base en los requisitos establecidos, se procederá al desarrollo del diseño de SCESO-Soft. Esto implica planificar cómo se estructurará el sistema y qué características serán prioritarias para la sistematización del control de horarios.

Desarrollo del Software:

La implementación del software se llevará a cabo según el diseño establecido. Durante esta fase, se traducirán los planes y diseños en un sistema funcional que aborde las necesidades específicas de OptiSalud.

Plan de Pruebas:

Se desarrollará un plan detallado para evaluar el rendimiento y la funcionalidad del sistema. Las pruebas se realizarán para asegurar que SCESO-Soft cumpla con los requisitos y para identificar posibles áreas de mejora. Las pruebas, depuración y mejoras continuas garantizarán la calidad y confiabilidad del sistema.

Análisis de resultados de la encuesta sobre la viabilidad del proyecto grado

Pregunta 01. ¿Cómo calificaría la eficiencia del proceso actual de registrar sus horas de trabajo de manera manual?

Objetivo: Evaluar la percepción de los empleados sobre la eficiencia del método actual de registro de horas laborales.

Figura 5

Diagrama de respuesta a la pregunta 01.

Fuente: Autoría propia.

Análisis: De 53 encuestados, los resultados son los siguientes: 29 (55%) indican que el proceso es muy ineficiente, 23 (43%) indican que es ineficiente y 1 (2%) se mantuvo neutral. Estos resultados reflejan una clara insatisfacción con el sistema actual de registro manual, lo que subraya la necesidad de un sistema más eficiente y confiable.

Tabla 1

Respuesta a la pregunta 01

¿Cómo calificaría la eficiencia del proceso actual de registrar sus horas de trabajo de manera manual?		
Muy Ineficiente	29	55%
Ineficiente	23	43%
Neutral	1	2%

Eficiente	0	0%
Muy eficiente	0	0%
Total, encuestados	53	100%

Nota. Muestras de encuesta de viabilidad del proyecto.

Pregunta 02. ¿Con qué frecuencia encuentra errores o discrepancias en los registros manuales de tiempo laboral?

Objetivo: Determinar la frecuencia con la que los empleados experimentan errores en el registro manual de horas

Figura 6

Diagrama de respuesta a la pregunta 02.

Fuente: Autoría propia.

Análisis: De 53 encuestados, 41 (77%) informan que encuentran errores muy frecuentemente, mientras que 12 (23%) señalan que encuentran errores con frecuencia. Ninguno de los encuestados indicó que los errores ocurran de manera ocasional, rara o nunca. Este resultado pone de manifiesto que los errores en los registros manuales son una problemática común y generalizada. Los errores mencionados incluyen discrepancias en la hora de entrada o salida, omisiones en el registro y duplicación de datos, lo que afecta la precisión en el conteo de horas laborales.

Tabla 2

Respuesta a la pregunta 02

¿Con qué frecuencia encuentra errores o discrepancias en los registros manuales de tiempo laboral?		
Muy Frecuentemente	41	77%
Frecuentemente	12	23%
Ocasionalmente	0	2%
Raramente	0	0%
Nunca	0	0%
Total, encuestados	53	100%

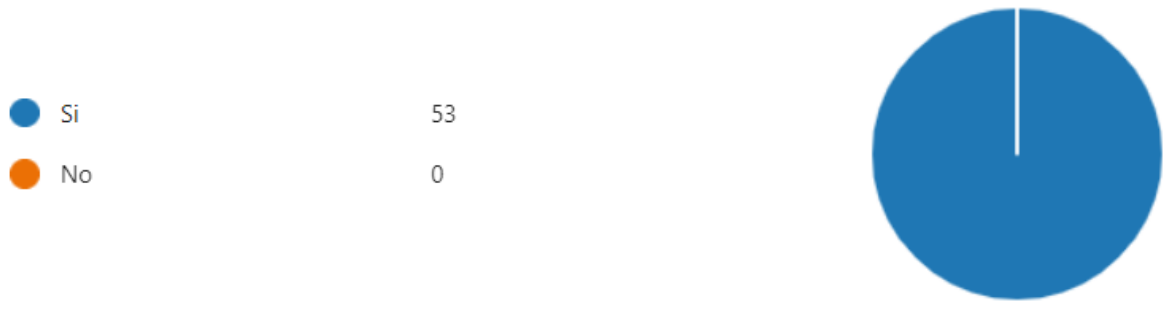
Nota. Muestras de encuesta de viabilidad del proyecto.

Pregunta 03. ¿Cree que los errores en el registro manual de horas afectan la exactitud del conteo de sus horas laborales?

Objetivo: Identificar si los empleados consideran que los errores en los registros manuales impactan la precisión de sus horas laborales.

Figura 7

Diagrama de respuesta a la pregunta 03.



Fuente: Autoría propia.

Análisis: De los 53 encuestados, todos (100%) coinciden en que los errores en el registro manual afectan la exactitud del conteo de sus horas laborales. Esta percepción unánime confirma el impacto negativo de los errores manuales, lo que puede llevar a una remuneración incorrecta y a una disminución de la satisfacción laboral.

Tabla 3

Respuesta a la pregunta 03

¿Cree que los errores en el registro manual de horas afectan la exactitud del conteo de sus horas laborales?		
Si	53	100%
No	0	0%
Total, encuestados	53	100%

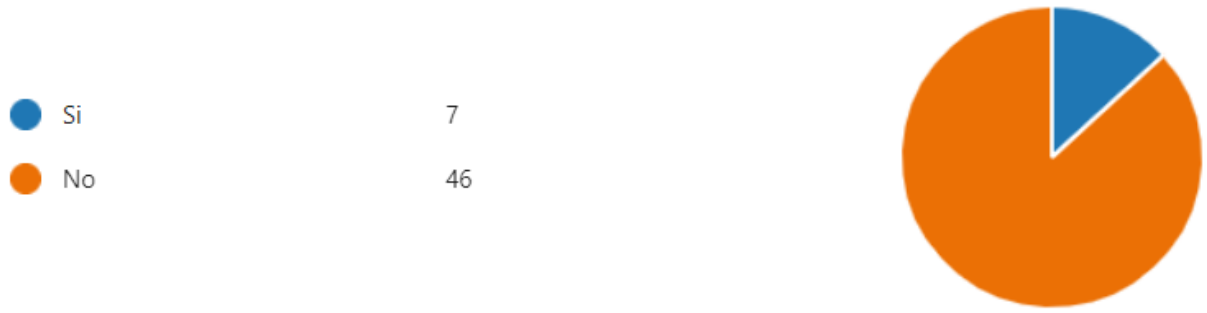
Nota. Muestras de encuesta de viabilidad del proyecto.

Pregunta 04. ¿Conoce sistemas automatizados para el registro de llegada y salida de horarios laborales?

Objetivo: Medir el nivel de conocimiento y familiaridad de los empleados con los sistemas automatizados de registro de tiempo.

Figura 8

Diagrama de respuesta a la pregunta 04.



Fuente: Autoría propia.

Análisis: De 53 encuestados, 7 (13%) indicaron conocer sistemas automatizados, mientras que 46 (87%) señalaron que no están familiarizados con ellos. Este resultado evidencia una falta de familiaridad general con las soluciones tecnológicas disponibles, lo que sugiere la necesidad de programas de capacitación para que los empleados se adapten mejor a nuevas tecnologías.

Tabla 4

Respuesta a la pregunta 04

¿Conoce sistemas automatizados para el registro de llegada y salida de horarios laborales?		
Si	7	13%
No	46	87%
Total, encuestados	53	100%

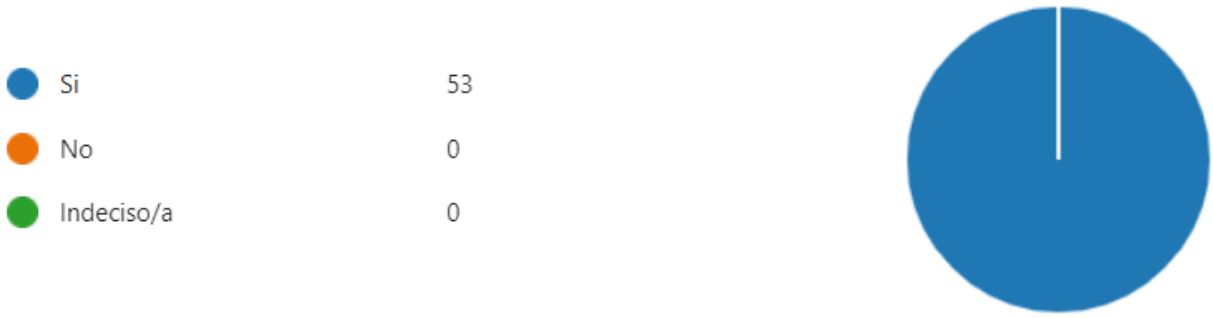
Nota. Muestras de encuesta de viabilidad del proyecto.

Pregunta 05. ¿Estaría de acuerdo con la implementación de un sistema automatizado para el registro de horas laborales?

Objetivo: Evaluar el nivel de aceptación y apertura de los empleados hacia la implementación de un sistema automatizado para el registro de horas.

Figura 9

Diagrama de respuesta a la pregunta 05.



Fuente: Autoría propia.

Análisis: De 53 encuestados, todos (100%) están de acuerdo en la implementación de un sistema automatizado para el registro de horas laborales. Este consenso muestra una fuerte disposición por parte de los empleados para adoptar nuevas tecnologías que mejoren la eficiencia del registro de sus horas laborales.

Tabla 5

Respuesta a la pregunta 05

¿Estaría de acuerdo con la implementación de un sistema automatizado para el registro de horas laborales?		
Si	53	100%
No	0	0%
Indeciso/a	0	0%
Total, encuestados	53	100%

Nota. Muestras de encuesta de viabilidad del proyecto.

Análisis General de Resultados:

Las respuestas de la encuesta reflejan un claro descontento con el sistema manual actual de registro de horas en OptiSalud. La mayoría de los empleados (98%) consideran que el proceso es ineficiente, lo que se traduce en errores frecuentes (77% mencionan que los errores ocurren muy frecuentemente) que afectan directamente la precisión en el conteo de sus horas laborales (100% lo confirma). Estos errores, que incluyen discrepancias en los horarios registrados y omisiones en la entrada de datos, generan problemas de exactitud en la remuneración de los empleados.

Por otro lado, aunque la mayoría de los encuestados no están familiarizados con los sistemas automatizados de registro (87%), todos los empleados (100%) están dispuestos a adoptar un sistema automatizado para mejorar la precisión y la eficiencia en el registro de sus

horas laborales. Esto indica una apertura significativa hacia el cambio tecnológico, siempre que este ofrezca mejoras tangibles.

Conclusión y Viabilidad del Proyecto

A partir del análisis de los resultados, se puede concluir que existe una necesidad urgente de mejorar el sistema actual de registro de horas en OptiSalud. La frecuencia de errores y la insatisfacción con el sistema manual actual destacan la importancia de implementar una solución automatizada. El alto nivel de aceptación hacia la adopción de un sistema automatizado por parte de los empleados refuerza la viabilidad del proyecto. Además, la implementación de SCESO-Soft no solo resolvería los problemas actuales de eficiencia y precisión, sino que también mejoraría la satisfacción de los empleados y optimizaría la gestión del tiempo laboral en la organización.

Planificación

Fase de Planificación

Actividad 1: Planificación y análisis (8 agosto 2023 - 27 agosto 2023)

Duración: 20 días

Descripción: Identificación de recursos, establecimiento de metas y objetivos, y análisis de requisitos iniciales.

Fase de Diseño

Actividad 2: Definición de requisitos del sistema (28 agosto 2023 - 5 septiembre 2023)

Duración: 10 días

Descripción: Identificación y documentación detallada de los requisitos específicos del sistema.

Actividad 3: Diseño de la interfaz de usuario (6 septiembre 2023 - 20 septiembre 2023)

Duración: 15 días

Descripción: Creación de un diseño detallado para la interfaz de usuario del sistema.

Fase de Desarrollo

Actividad 4: Diseño de la arquitectura de software (21 septiembre 2023 - 30 septiembre 2023)

Duración: 10 días

Descripción: Establecimiento de la estructura general del software y definición de la tecnología a utilizar.

Actividad 5: Desarrollo del software (1 octubre 2023 - 29 diciembre 2023)

Duración: 90 días

Descripción: Implementación práctica del sistema según el diseño establecido.

Fase de Pruebas

Actividad 6: Pruebas unitarias y de integración (30 diciembre 2023 – 8 enero 2023)

Duración: 10 días

Descripción: Realización de pruebas para verificar que cada componente del software funcione correctamente.

Actividad 7: Pruebas de sistema y correcciones (9 enero 2024 - 18 enero 2024)

Duración: 10 días

Descripción: Pruebas del sistema en conjunto y corrección de errores detectados.

Fase de Finalización

Actividad 8: Preparación para el lanzamiento (19 enero 2024 - 28 enero 2024)

Duración: 10 días

Descripción: Preparativos finales para el lanzamiento del sistema y formación de usuarios.

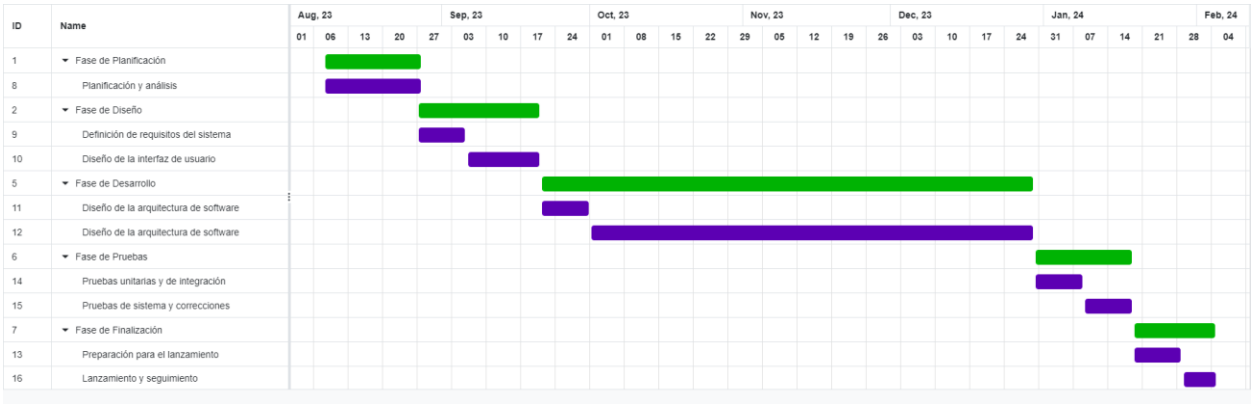
Actividad 9: Lanzamiento y seguimiento (29 enero 2024 - 7 febrero 2024)

Duración: 7 días laborables

Descripción: Lanzamiento del sistema y monitoreo inicial para asegurar la operatividad.

Figura 10

Diagrama de actividades.



Fuente: Autoría propia.

Presupuesto

Tabla 6

Consulta y Metodología

Ítem	Descripción	Costo
Análisis del Problema	Consultoría especializada para analizar a fondo los problemas existentes y proponer soluciones.	400.000
Diseño de Metodología	Desarrollo de una metodología específica adaptada a las necesidades del proyecto.	400.000
Total:		800.000

Nota. Muestras de costo de la consulta y metodología del proyecto.

Tabla 7

Recursos Humanos

Ítem	Descripción	Costo
Desarrollador de Software	Crear la aplicación.	18’000.000
Especialista en Bases de Datos	Crear diseño y gestión de bases de datos del sistema.	3’000.000
Total:		21’000.000

Nota. Costo de los recursos humanos para el proyecto.

Tabla 8

Recursos Técnicos

Ítem	Descripción	Costo
Licencias de Software	Costo de las licencias de software necesarias para el desarrollo y funcionamiento del sistema.	1’200.000

Servidor y Almacenamiento	Adquisición o alquiler de servidor y espacio de almacenamiento para la base de datos.	500.000
Equipamiento de Computadoras	Costo de las computadoras y dispositivos necesarios para el desarrollo y prueba del sistema.	5’000.000
Total:		6’700.000

Nota. Costo de los recursos técnicos para el proyecto.

Tabla 9

Gastos Operativos

Ítem	Descripción	Costo
Consumibles	Gastos en papel, tinta, entre otros, necesarios para la documentación y pruebas del sistema.	200.000
Capacitación y Entrenamiento	Costo de programas de capacitación para el personal que utilizará y mantendrá el sistema.	200.000
Total:		400.000

Nota. Costo de los gastos operativos para el proyecto.

Tabla 10

Gastos adicionales

Ítem	Descripción	Costo
Contingencias	Fondos reservados para imprevistos y contingencias durante el desarrollo del proyecto.	1’500.000
Asesoría Legal	Honorarios legales para la revisión de contratos y garantizar el cumplimiento legal del proyecto.	1’000.000

Total:	2’500.000

Nota. Costo de los gastos operativos para el proyecto.

Tabla 11

Muestra General de Presupuesto

Ítem	Costo
Consulta y Metodología	800.000
Recursos Humanos	21’000.000
Recursos Técnicos	6’700.000
Gastos Operativos	400.000
Gastos adicionales	2’500.000
Total:	31’400.000

Nota. Valor total para la construcción del proyecto.

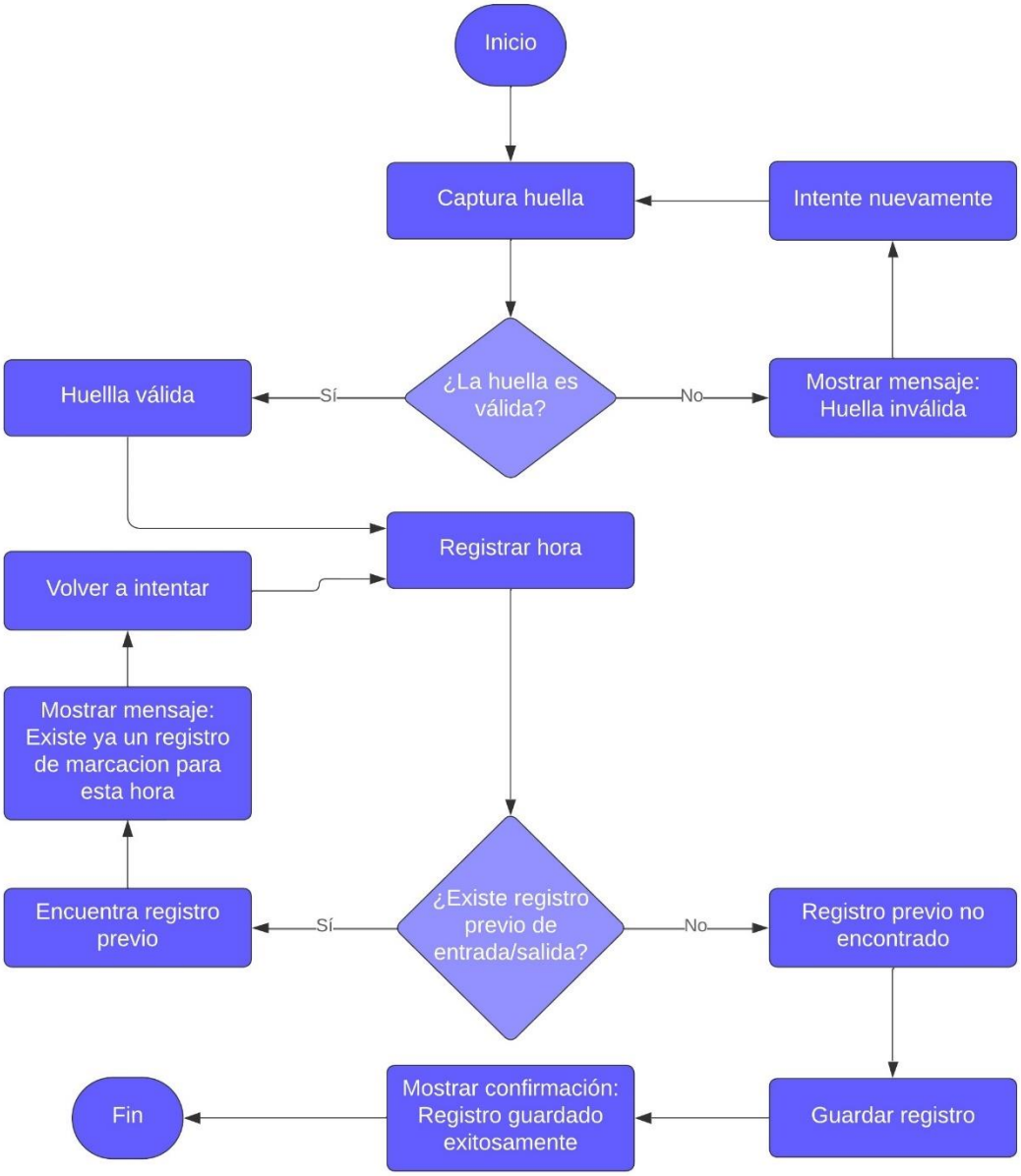
Este presupuesto detallado proporciona una base sólida para la gestión financiera del proyecto, cubriendo todas las áreas clave desde el desarrollo hasta la legalidad y la formación del personal. Como creador y responsable financiero del proyecto, asumo personalmente todos los costos asociados. Este compromiso refleja mi dedicación a la calidad y la eficiencia, asegurando que todas las etapas del proyecto estén adecuadamente financiadas para garantizar su éxito. Este enfoque garantiza la continuidad y la integridad del proyecto, cumpliendo con todos los requisitos documentales necesarios para su realización.

Diseño

Diagrama de Flujos

Figura 11

Diagrama de flujo del proceso de registro de asistencia para empleados.



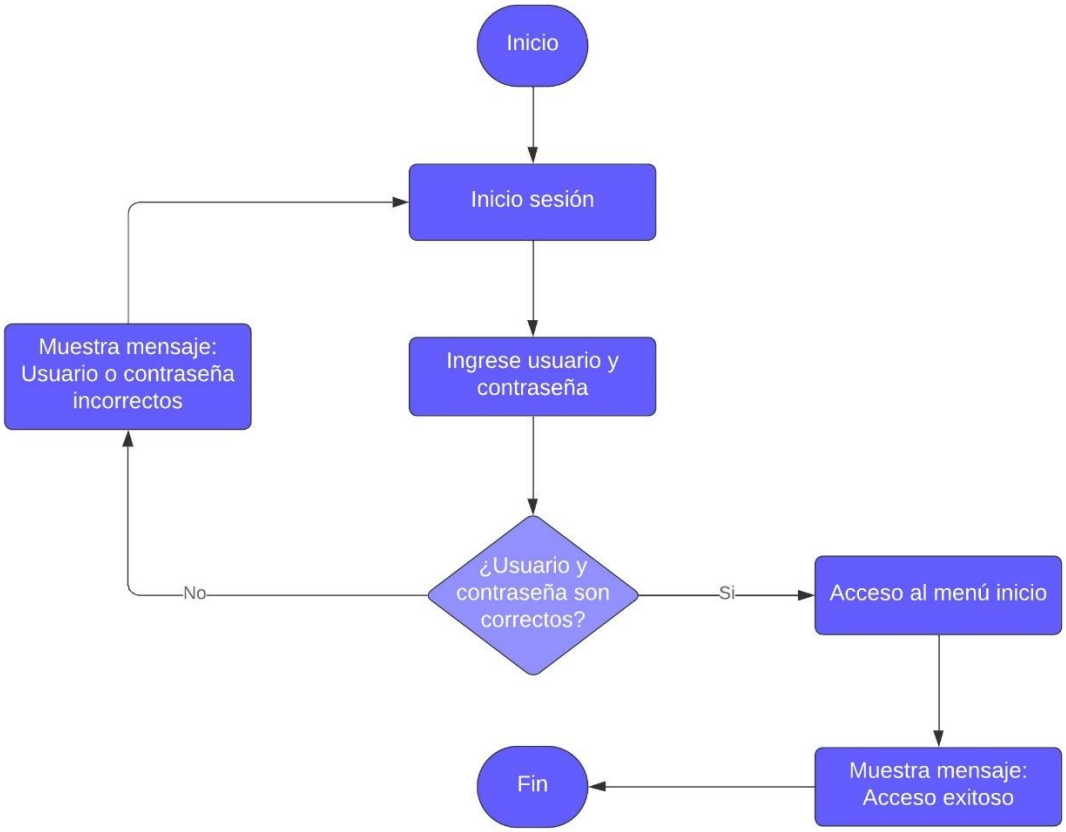
Fuente: Autoría propia.

Descripción del proceso de registro de asistencia para empleados:

Este diagrama de flujo detalla el proceso de registro de asistencia para empleados mediante huella dactilar. El proceso inicia con la captura de la huella, la cual es validada a continuación. Si la huella no es válida, se muestra un mensaje indicando que la huella es inválida y se permite al usuario intentar nuevamente, regresando al paso de captura de huella. Si la huella es válida, se procede al registro de la hora. En caso de que ya exista un registro previo de marcación para ese horario, se muestra un mensaje de aviso y se vuelve a intentar. Si no hay un registro previo, se guarda la nueva entrada o salida. Finalmente, una confirmación del registro exitoso es mostrada al usuario, indicando la conclusión exitosa del proceso.

Figura 12

Diagrama de flujo del proceso de autenticación en el sistema.



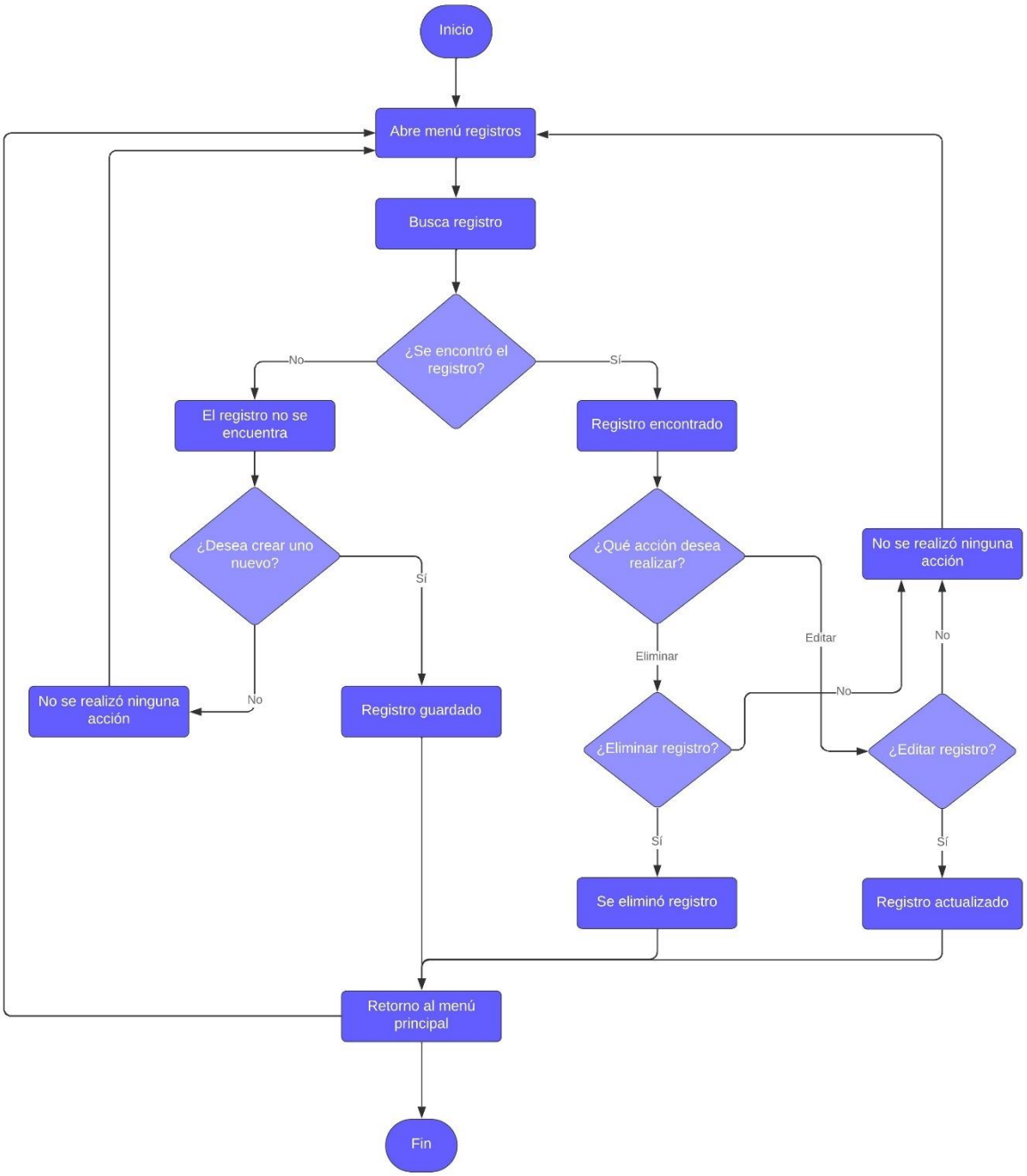
Fuente: Autoría propia.

Descripción del proceso de autenticación en el sistema:

El diagrama de flujo "Proceso de Autenticación en el Sistema" describe los pasos que sigue un usuario al iniciar sesión. Primero, el usuario accede a la pantalla de inicio de sesión e ingresa sus credenciales (usuario y contraseña). Luego, el sistema verifica si las credenciales ingresadas son correctas. Si lo son, el usuario accede al menú principal, y se muestra un mensaje de "Acceso exitoso". Si las credenciales son incorrectas, el sistema muestra un mensaje de error y permite que el usuario intente ingresar nuevamente. Este flujo asegura que solo usuarios con credenciales válidas puedan acceder al sistema.

Figura 13

Diagrama de flujo del proceso de gestión de registros.



Fuente: Autoría propia.

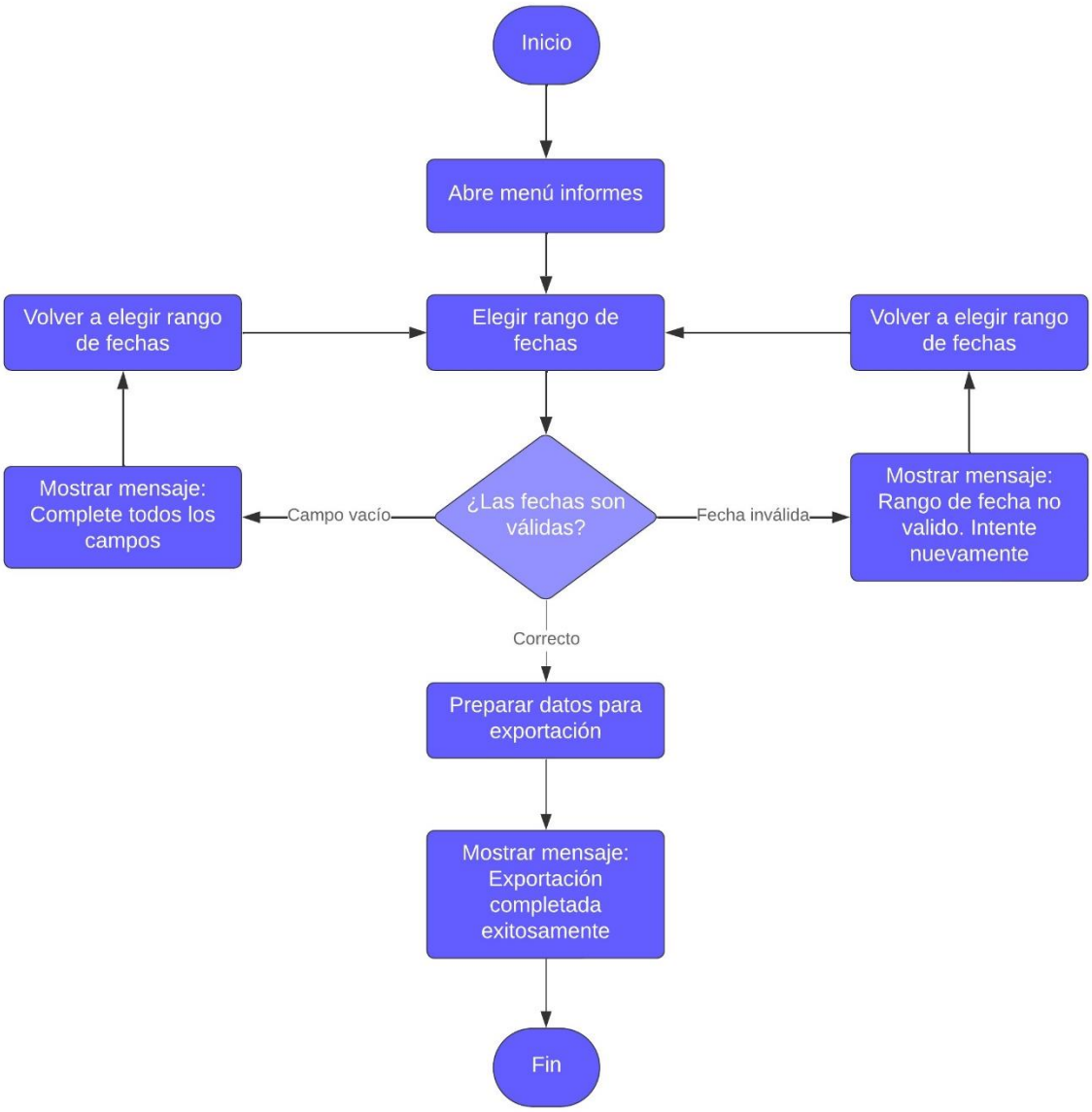
Descripción del proceso de gestión de registros:

Este diagrama de flujo representa el proceso de gestión de registros en el sistema. El usuario inicia en el menú de registros y realiza una búsqueda. Si el registro buscado no se

encuentra, se le da la opción de crear uno nuevo. Si decide crear el registro, este se guarda y el flujo retorna al menú principal. Si el registro existe, el usuario puede elegir entre editarlo o eliminarlo. Si decide eliminar, el registro se borra; si elige editar, el registro se actualiza. En ambos casos, el proceso retorna al menú principal. Además, en cada decisión, si el usuario opta por no realizar ninguna acción, el sistema muestra un mensaje indicando que no se realizó ninguna acción y regresa al menú. Esto asegura una navegación clara y completa para la gestión de registros.

Figura 14

Diagrama de flujo del proceso de generación y exportación de informes.



Fuente: Autoría propia.

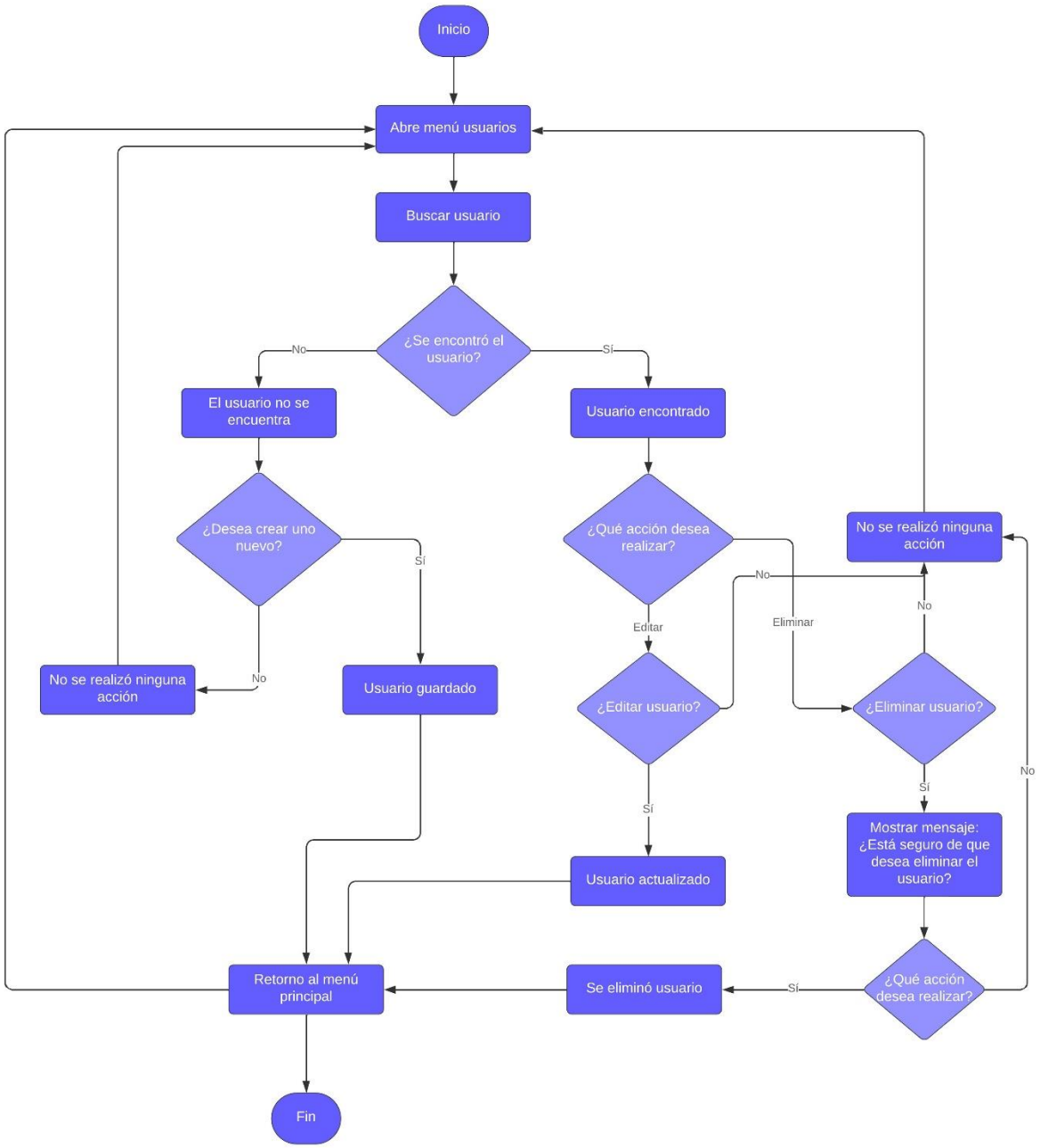
Descripción del proceso de generación y exportación de informes:

Este diagrama de flujo representa el proceso de exportación de informes, comenzando con la apertura del menú de informes y la selección del rango de fechas. Una vez ingresadas las fechas, el sistema valida si los campos están completos y si el rango es válido. En caso de error, muestra mensajes específicos para campos vacíos o rangos no válidos y permite volver a

seleccionar las fechas. Si las fechas son correctas, se preparan los datos para exportación y se completa el proceso mostrando un mensaje de confirmación de exportación exitosa antes de finalizar.

Figura 15

Diagrama de flujo del proceso de gestión de usuarios.



Fuente: Autoría propia.

Descripción del proceso de gestión de usuarios:

Este diagrama de flujo representa el proceso de gestión de usuarios, permitiendo crear, editar o eliminar registros en el sistema. El flujo comienza con la apertura del menú de

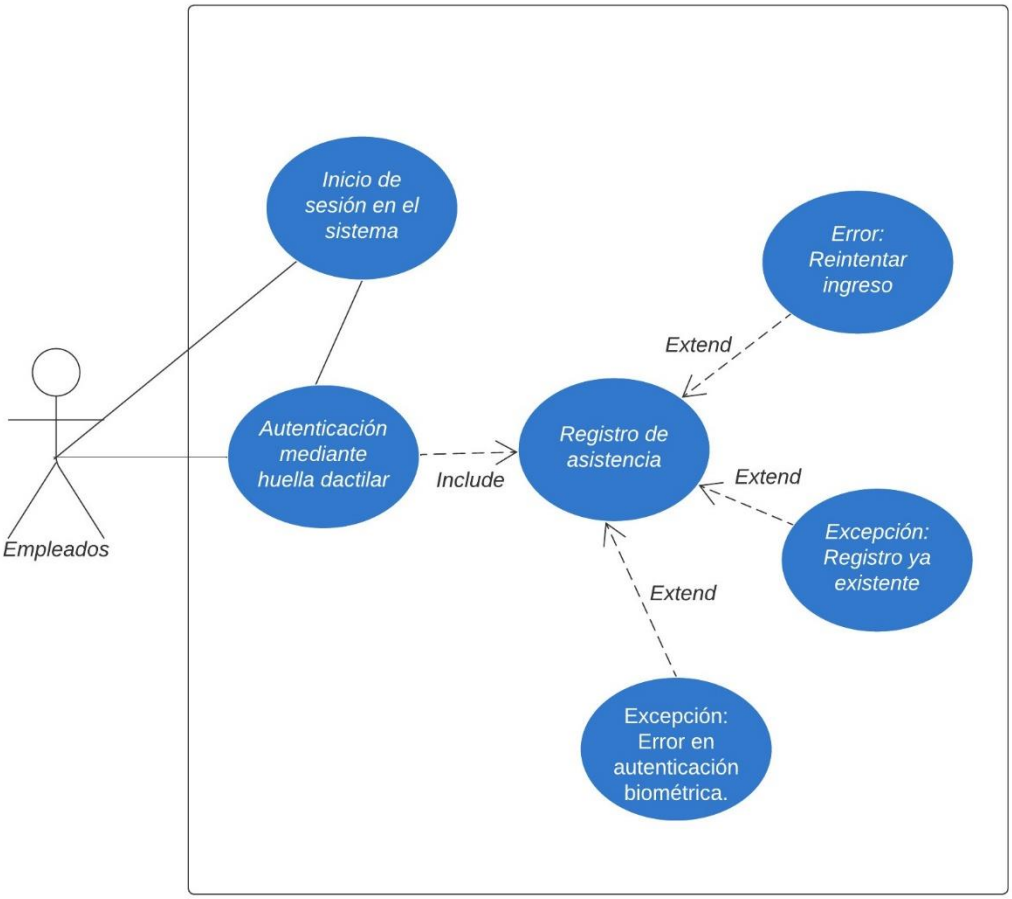
usuarios, seguido de una búsqueda del registro. Si el usuario no se encuentra, se ofrece la opción de crear uno nuevo. Si el usuario existe, se muestra un menú con opciones para editar o eliminar el registro, cada una con su respectiva confirmación de acción. Tras realizar alguna acción (creación, actualización o eliminación), el sistema confirma el cambio y retorna al menú principal, cerrando el proceso de manera ordenada.

Diagrama de casos de usos

Actor: Empleados

Figura 16

Diagrama de casos de uso: registro de Entrada/Salida con Huella Dactilar.



Fuente: Autoría propia.

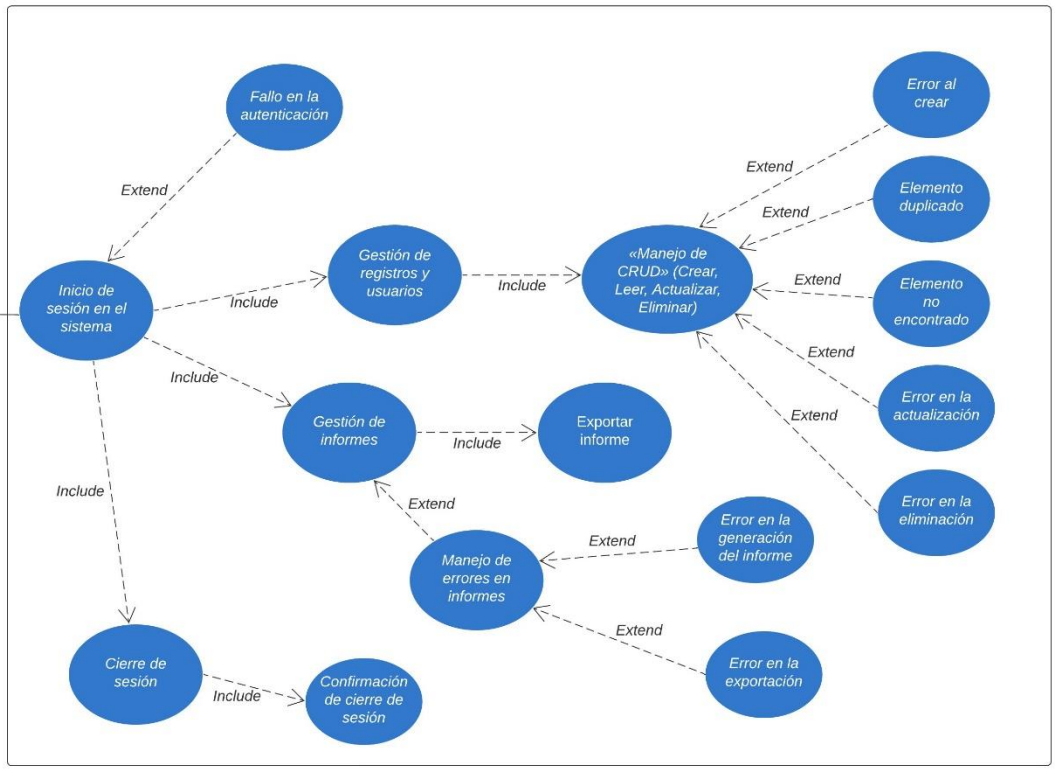
Explicación para el diagrama de casos de uso: registro de Entrada/Salida con Huella Dactilar

Este diagrama de casos de uso representa el flujo de autenticación y registro de asistencia para los empleados en el sistema. Los empleados pueden iniciar sesión en el sistema mediante autenticación con huella dactilar, un paso obligatorio que se incluye dentro del caso de uso de "Registro de asistencia". Existen excepciones para manejar situaciones específicas: si la autenticación falla, se muestra "Error en autenticación biométrica"; si ya existe un registro de asistencia para ese horario, se activa la "Excepción: Registro ya existente"; y en caso de error al ingresar, se da la opción de "Reintentar ingreso". Este flujo permite al sistema gestionar tanto los registros exitosos como los posibles fallos o intentos duplicados de manera clara y estructurada.

Actor: Empleados con privilegios

Figura 17

Diagrama de Casos de Uso: Sistema de gestión para empleados con privilegios.



Fuente: Autoría propia.

Explicación para el diagrama de casos de uso: Sistema de gestión para empleados con privilegios.

Este diagrama de casos de uso representa las funcionalidades principales del sistema de gestión de usuarios e informes, accesibles para empleados con privilegios. El flujo principal comienza con el Inicio de sesión en el sistema, seguido de operaciones como gestión de registros y usuarios (que incluye el manejo de operaciones CRUD: crear, leer, actualizar, eliminar) y gestión de informes. Cada operación CRUD está relacionada con posibles errores específicos, como elemento duplicado o elemento no encontrado, que se activan solo en situaciones excepcionales. El sistema también permite la exportación de informes, que puede generar errores específicos, como error en la generación del informe y error en la exportación. Adicionalmente, el flujo de trabajo contempla el cierre de sesión y su confirmación, mientras

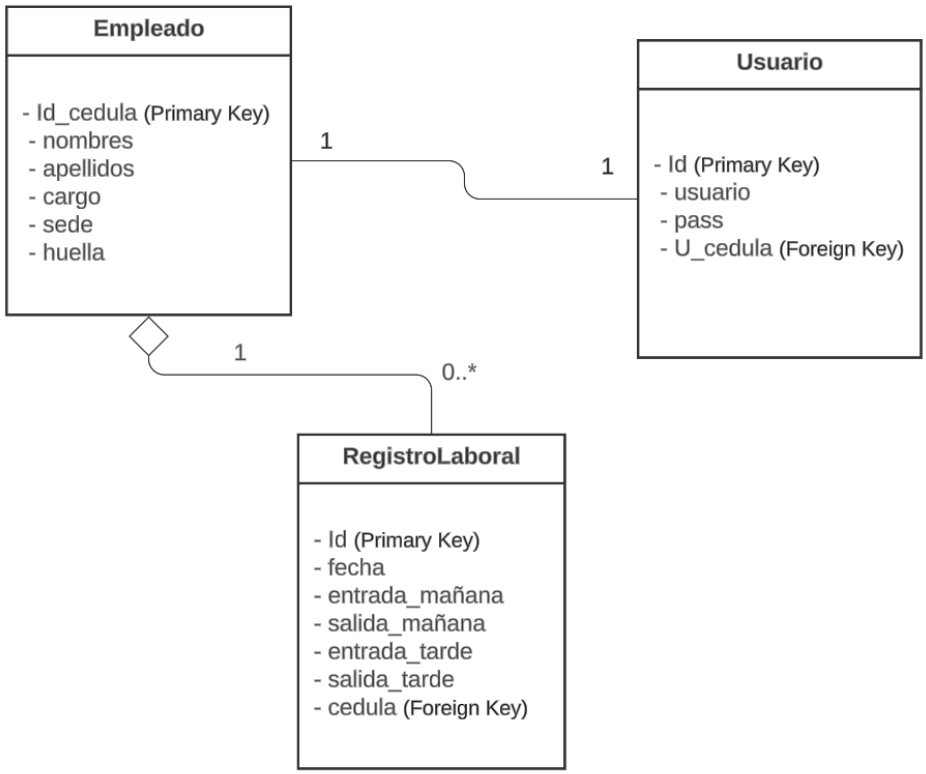
que un posible fallo en la autenticación se gestiona como una excepción en el proceso de inicio de sesión, garantizando una respuesta adecuada ante cualquier error en las funciones críticas del sistema.

Diagrama de Dominio

El siguiente diagrama representa la estructura fundamental del sistema que gestiona los empleados, los registros laborales y los usuarios. Se centra en la relación entre estos tres componentes clave: Empleado, RegistroLaboral y Usuario. Cada entidad tiene atributos específicos que describen su propósito y funcionalidad dentro del sistema.

Figura 18

Diagrama de dominio.



Fuente: Autoría propia.

Comportamiento del Sistema Basado en el Diagrama

- Entidades:

Empleado: Representa a los empleados de la organización, con atributos como Id_cedula, nombres, cargo, y la huella dactilar para marcar asistencia.

Relación: Un empleado puede tener varios registros laborales (1:0..*), y está vinculado de forma uno a uno (1:1) con un Usuario.

- Usuario:

Define las credenciales de acceso para el sistema administrativo.

Relación: Cada usuario está asociado a un empleado de forma uno a uno.

- RegistroLaboral:

Contiene los registros de asistencia de cada empleado, con información como la fecha y las horas de entrada y salida.

Relación: Cada registro pertenece a un solo empleado, pero un empleado puede tener múltiples registros.

- Relaciones:

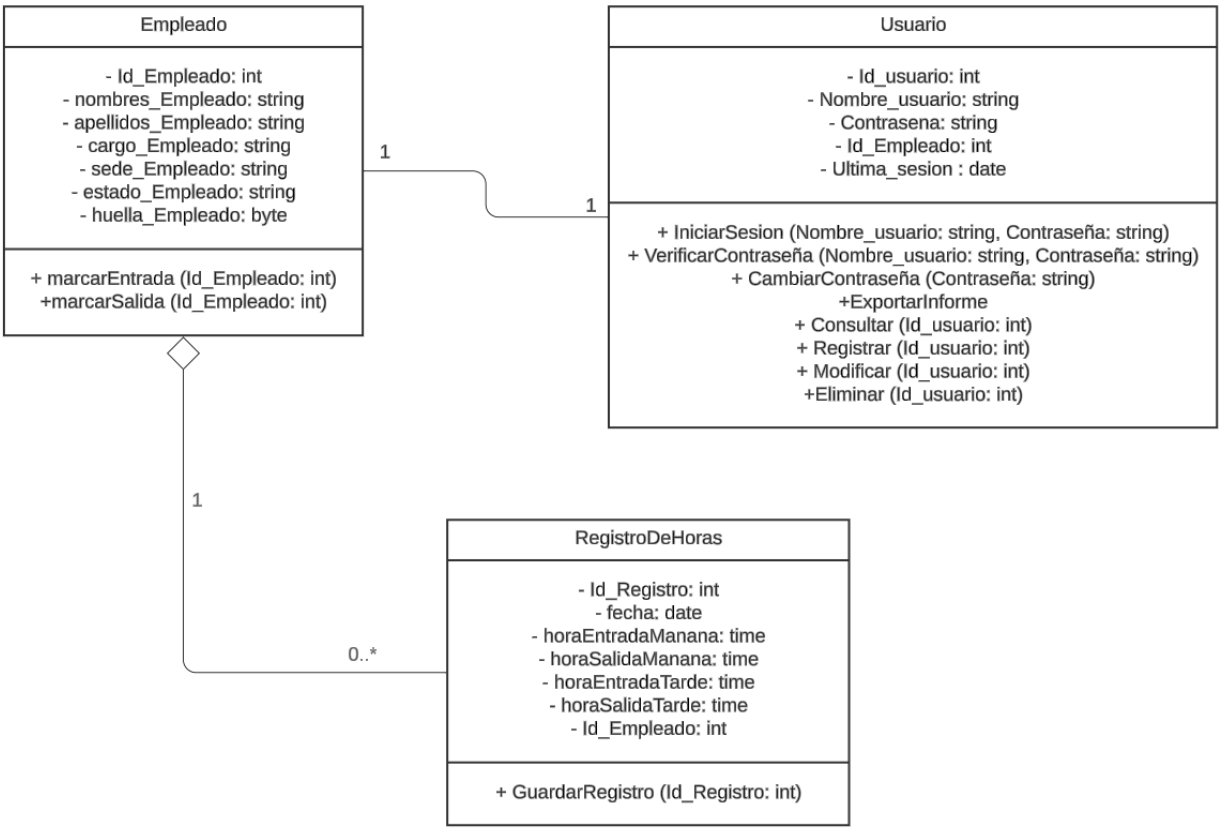
Empleado - Usuario: Relación 1:1. Cada empleado tiene un solo usuario vinculado.

Empleado - RegistroLaboral: Relación 1 a muchos (1:0..*), ya que un empleado puede tener múltiples registros de asistencia.

Diagrama de clases

Figura 19

Diagrama de clases.



Fuente: Autoría Propia.

Este diagrama de clases describe la estructura de un sistema que gestiona empleados, usuarios y registros de horas laborales. A continuación, se explican las clases principales y sus relaciones:

- Clase Empleado:

Contiene atributos privados como Id_Empleado, nombres_Empleado, apellidos_Empleado, cargo_Empleado, sede_Empleado, estado_Empleado y huella_Empleado.

Los métodos públicos marcarEntrada y marcarSalida permiten registrar la entrada y salida del empleado, utilizando su Id_Empleado.

- Clase Usuario:

Representa a los usuarios del sistema, relacionados uno a uno con un empleado a través del Id_Empleado.

Los atributos privados incluyen Id_usuario, Nombre_usuario, Contraseña, Id_Empleado y Ultima_sesion.

Los métodos públicos incluyen la gestión de sesiones y las operaciones CRUD (Consultar, Registrar, Modificar, Eliminar) para los usuarios.

- Clase RegistroDeHoras:

Esta clase almacena los registros de horarios laborales de los empleados.

Contiene atributos como Id_Registro, fecha, horaEntradaManana, horaSalidaManana, horaEntradaTarde, horaSalidaTarde y Id_Empleado.

El método público GuardarRegistro permite guardar el registro de horas laborales.

- Relaciones:

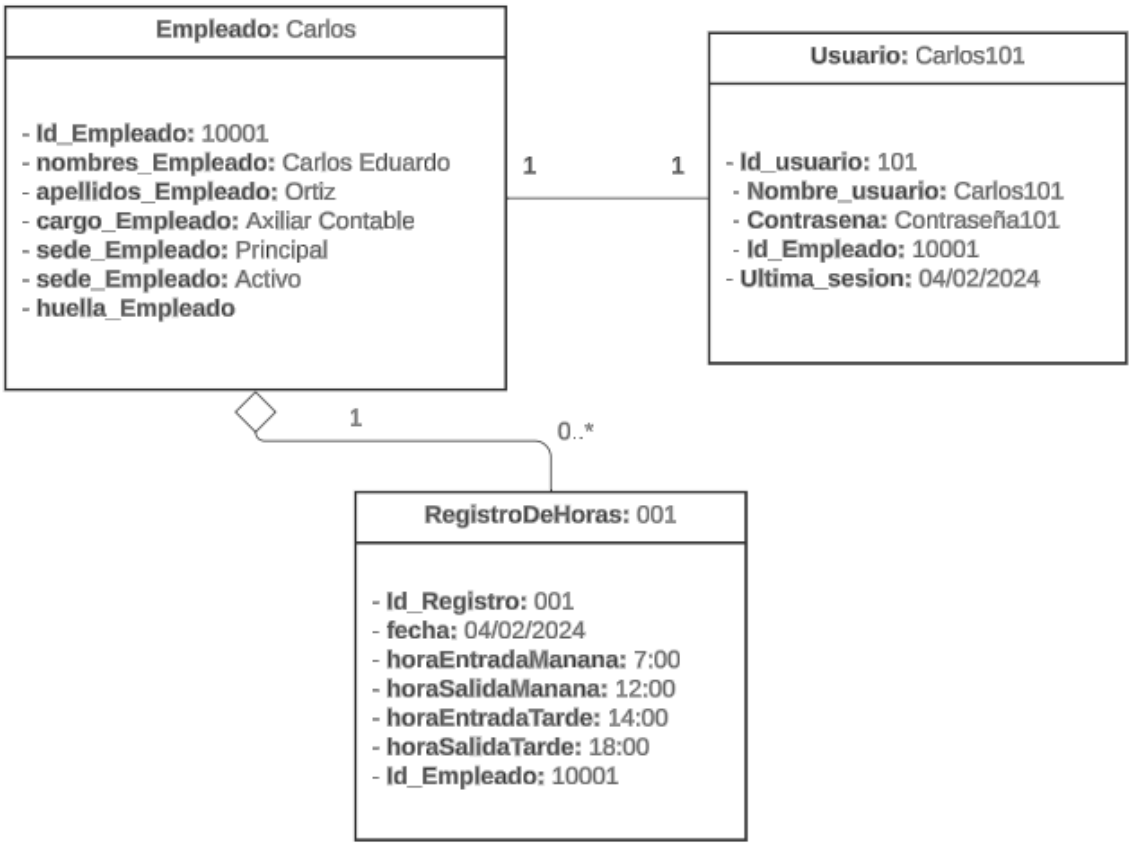
Hay una relación de composición entre Empleado y RegistroDeHoras, indicando que un empleado puede tener múltiples registros de horas, pero los registros dependen del empleado.

Existe una relación uno a uno entre Empleado y Usuario, lo que significa que cada empleado tiene un único usuario asociado en el sistema.

Diagrama de objetos

Figura 20

Diagrama de objetos.



Fuente: Autoría Propia.

Este diagrama de objetos muestra una instancia específica de las clases Empleado, Usuario, y RegistroDeHoras, con datos concretos que ejemplifican cómo se relacionan los objetos en el sistema.

- Objeto Empleado: Carlos:

Este objeto representa al empleado Carlos Eduardo Ortiz, identificado con el Id_Empleado 10001.

Atributos adicionales incluyen su cargo (Auxiliar Contable), sede (Principal) y su huella digital (aunque no se detalla su valor).

Carlos tiene una relación de uno a uno con un usuario en el sistema.

- Objeto Usuario: Carlos101:

El usuario asociado a Carlos es Carlos101, con el Id_usuario 101.

Este usuario tiene atributos como el nombre de usuario (Carlos101), contraseña (Contraseña101), y su última sesión registrada el 04/02/2024.

Está vinculado a su empleado correspondiente mediante el Id_Empleado 10001.

- Objeto RegistroDeHoras: 001:

Este objeto representa un registro laboral específico del empleado Carlos Eduardo Ortiz con el Id_Registro 001.

El registro laboral indica que el 04/02/2024, Carlos tuvo una jornada de trabajo con:

Entrada a las 7:00 am y salida a las 12:00 pm en la mañana.

Entrada a las 14:00 pm y salida a las 18:00 pm en la tarde.

El registro está asociado al Id_Empleado 10001, que corresponde a Carlos.

- Relaciones:

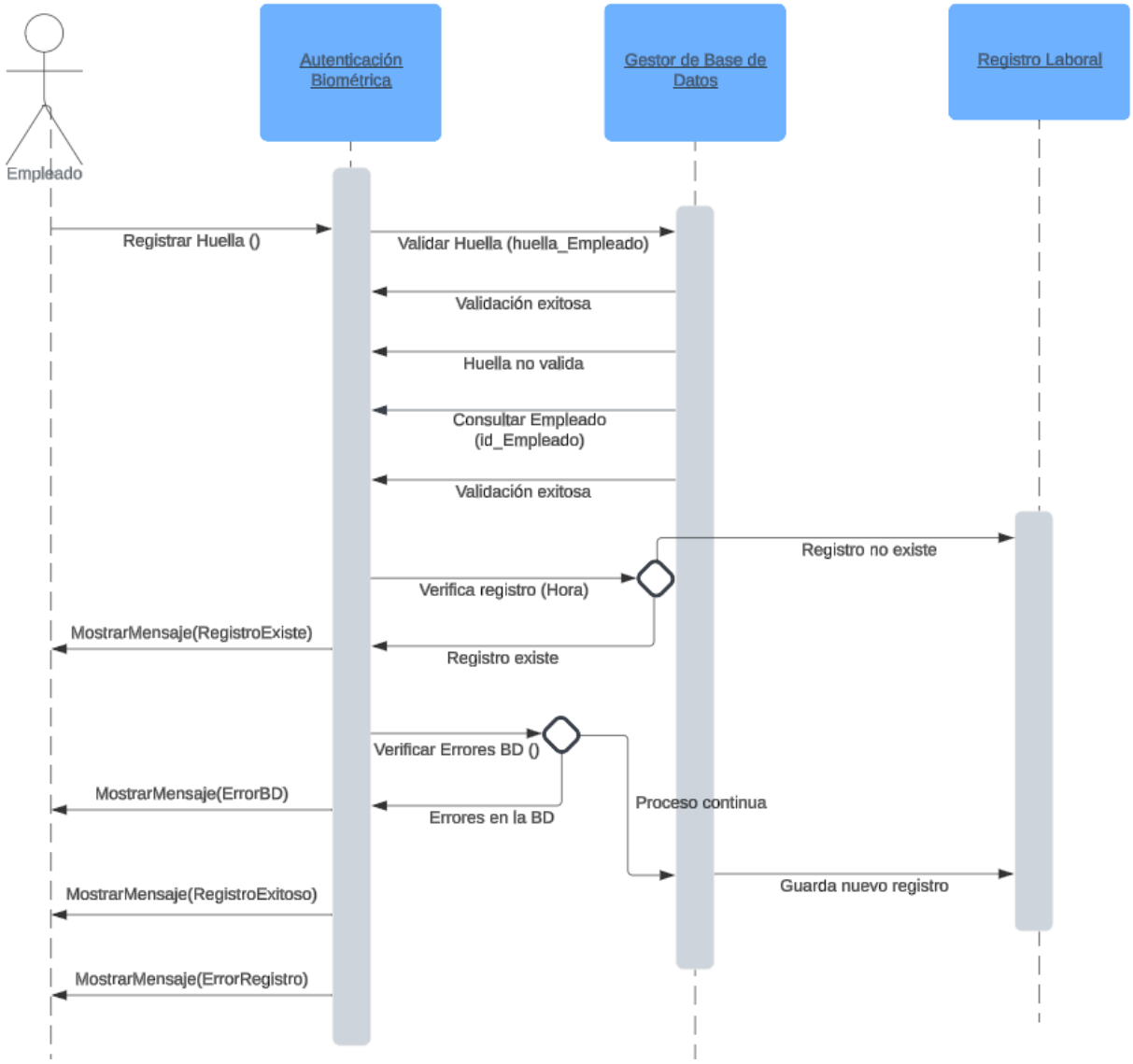
Empleado y Usuario: Existe una relación uno a uno, ya que el empleado Carlos tiene un único usuario asociado, Carlos101.

Empleado y RegistroDeHoras: La relación es de uno a muchos, lo que significa que el empleado Carlos puede tener múltiples registros laborales (en este caso, el registro con Id_Registro 001).

Diagramas de secuencias

Figura 21

Diagrama de Secuencia para Registro de Asistencia Biométrica (Empleado).

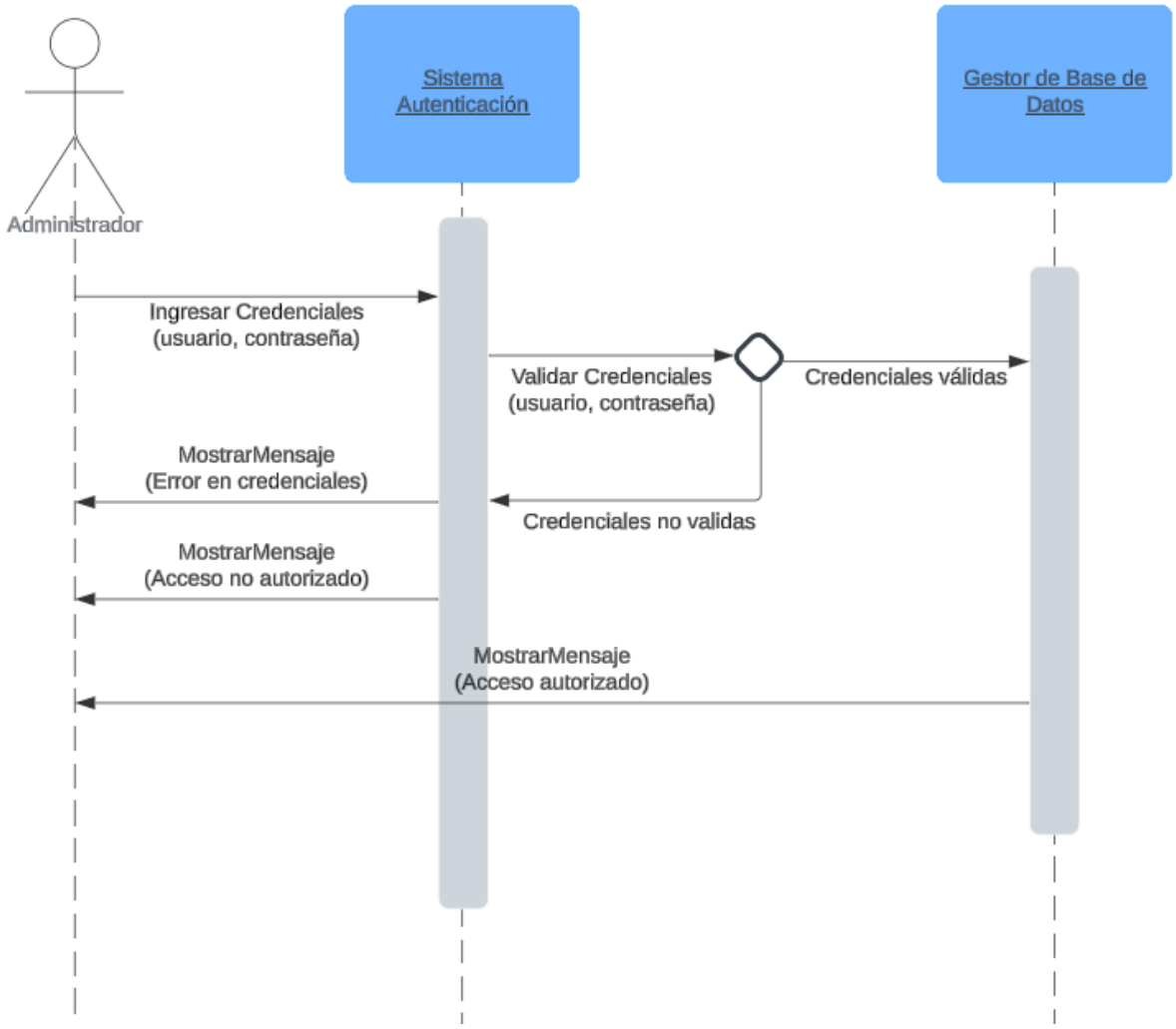


Fuente: Autoría Propia.

Este diagrama de secuencia muestra el proceso de registro de asistencia mediante huella dactilar. El empleado registra su huella en el sistema de autenticación biométrica, que valida la huella y consulta en la base de datos si el empleado existe. Si la validación es exitosa, el sistema verifica si ya existe un registro para la hora actual. Si no existe, se guarda un nuevo registro en el registro laboral. En caso de errores en la base de datos o en el registro, el sistema notifica al empleado con mensajes específicos de éxito o error.

Figura 22

Diagrama de secuencia para *inicio de sesión (Administrador).*



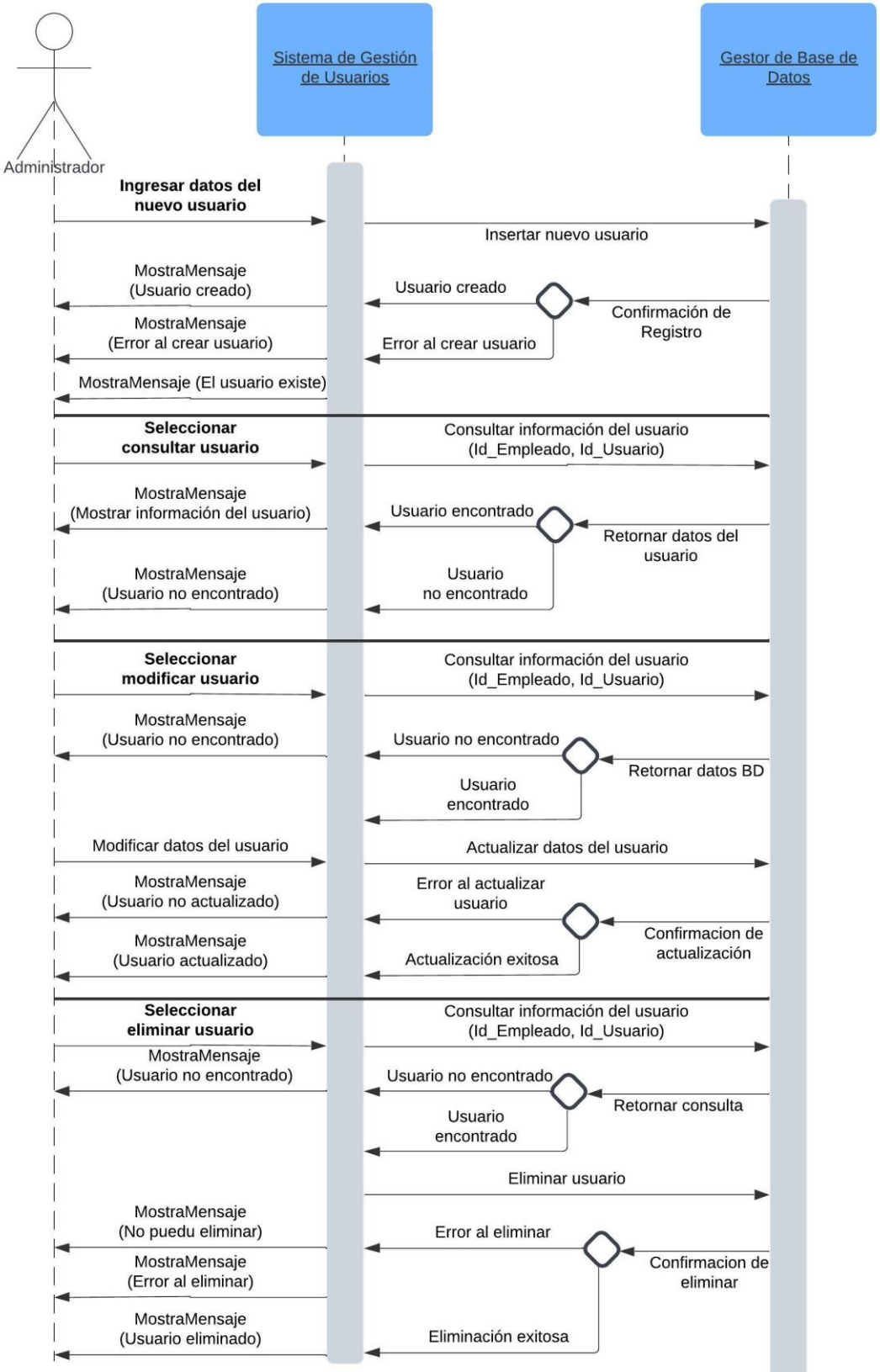
Fuente: Autoría Propia.

Este diagrama de secuencia representa el proceso de inicio de sesión de un administrador. Primero, el administrador ingresa sus credenciales (usuario y contraseña), que son enviadas al sistema de autenticación. El sistema valida las credenciales consultando la base de datos. Si las credenciales son válidas, el sistema muestra un mensaje de "Acceso

Autorizado". Si las credenciales son incorrectas o no válidas, se envía un mensaje de error o de "Acceso No Autorizado", y el proceso finaliza. Este flujo garantiza que solo usuarios con credenciales correctas puedan acceder.

Figura 23

Diagrama de secuencia para *gestión de usuarios (CRUD).*

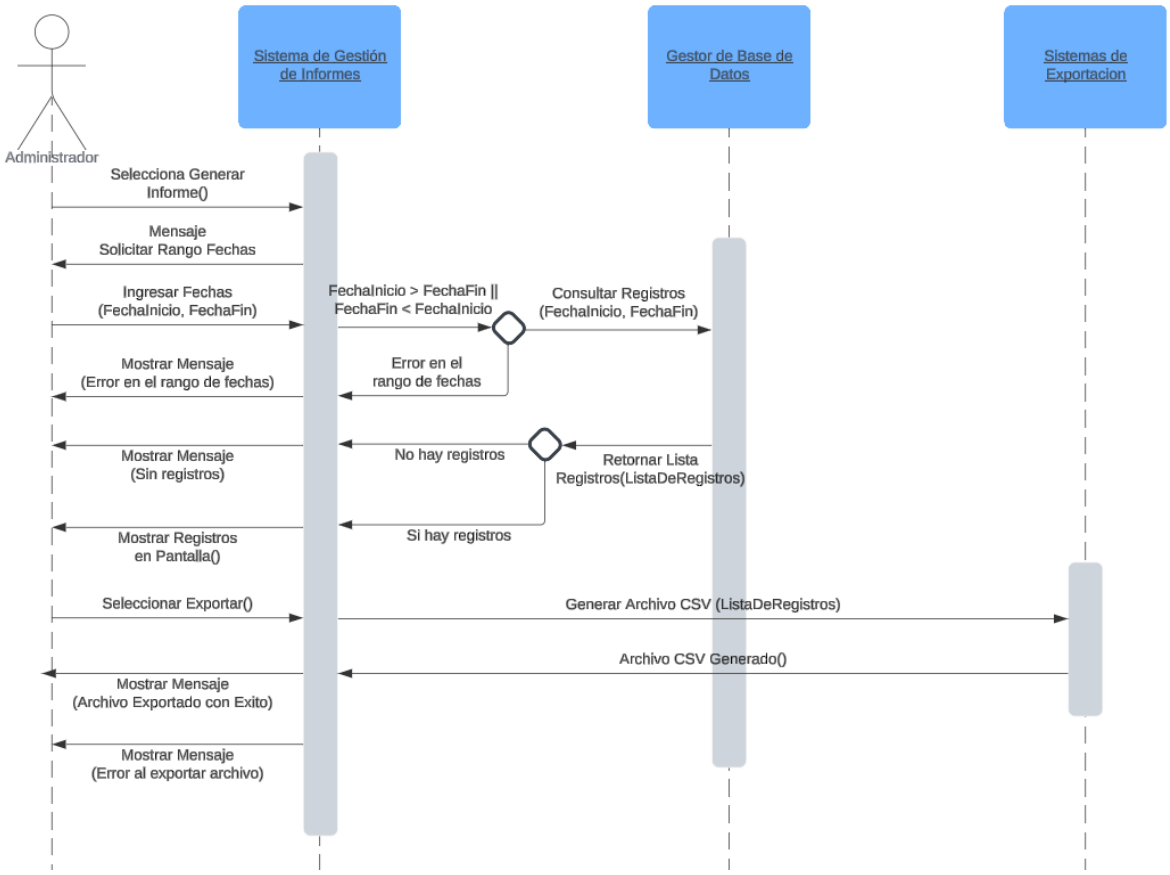

Administrador
Sistema de Gestión de Usuarios
Gestor de Base de Datos
Ingresar datos del nuevo usuario
Insertar nuevo usuario
MostraMensaje (Usuario creado)
Usuario creado
Confirmación de Registro
MostraMensaje (Error al crear usuario)
Error al crear usuario
MostraMensaje (El usuario existe)
Seleccionar consultar usuario
Consultar información del usuario (Id_Empleado, Id_Usuario)
MostraMensaje (Mostrar información del usuario)
Usuario encontrado
Retornar datos del usuario
MostraMensaje (Usuario no encontrado)
Usuario no encontrado
Seleccionar modificar usuario
Consultar información del usuario (Id_Empleado, Id_Usuario)
MostraMensaje (Usuario no encontrado)
Usuario no encontrado
Retornar datos BD
Usuario encontrado
Modificar datos del usuario
Actualizar datos del usuario
MostraMensaje (Usuario no actualizado)
Error al actualizar usuario
Confirmacion de actualización
MostraMensaje (Usuario actualizado)
Actualización exitosa
Seleccionar eliminar usuario
Consultar información del usuario (Id_Empleado, Id_Usuario)
MostraMensaje (Usuario no encontrado)
Usuario no encontrado
Retornar consulta
Usuario encontrado
Eliminar usuario
MostraMensaje (No puedu eliminar)
Error al eliminar
Confirmacion de eliminar
MostraMensaje (Error al eliminar)
MostraMensaje (Usuario eliminado)
Eliminación exitosa

Fuente: Autoría Propia.

Este diagrama de secuencia ilustra las operaciones del CRUD (Crear, Leer, Actualizar, Eliminar) para la gestión de usuarios en un sistema. Inicia con el administrador ingresando los datos del nuevo usuario para su creación, mostrando mensajes según el éxito o fallo en el registro. Luego, detalla el proceso de consulta de usuario, mostrando si el usuario es encontrado o no. El administrador también puede modificar los datos del usuario o eliminarlos, y en ambos casos, el sistema retorna mensajes que reflejan el éxito o error en las acciones, incluyendo errores específicos como "Usuario no encontrado" o "Error al eliminar". Cada operación es gestionada por el sistema de gestión de usuarios y comunicada a la base de datos, que almacena y verifica la información de los usuarios.

Figura 24

Diagrama de secuencia para *exportación de informes.*

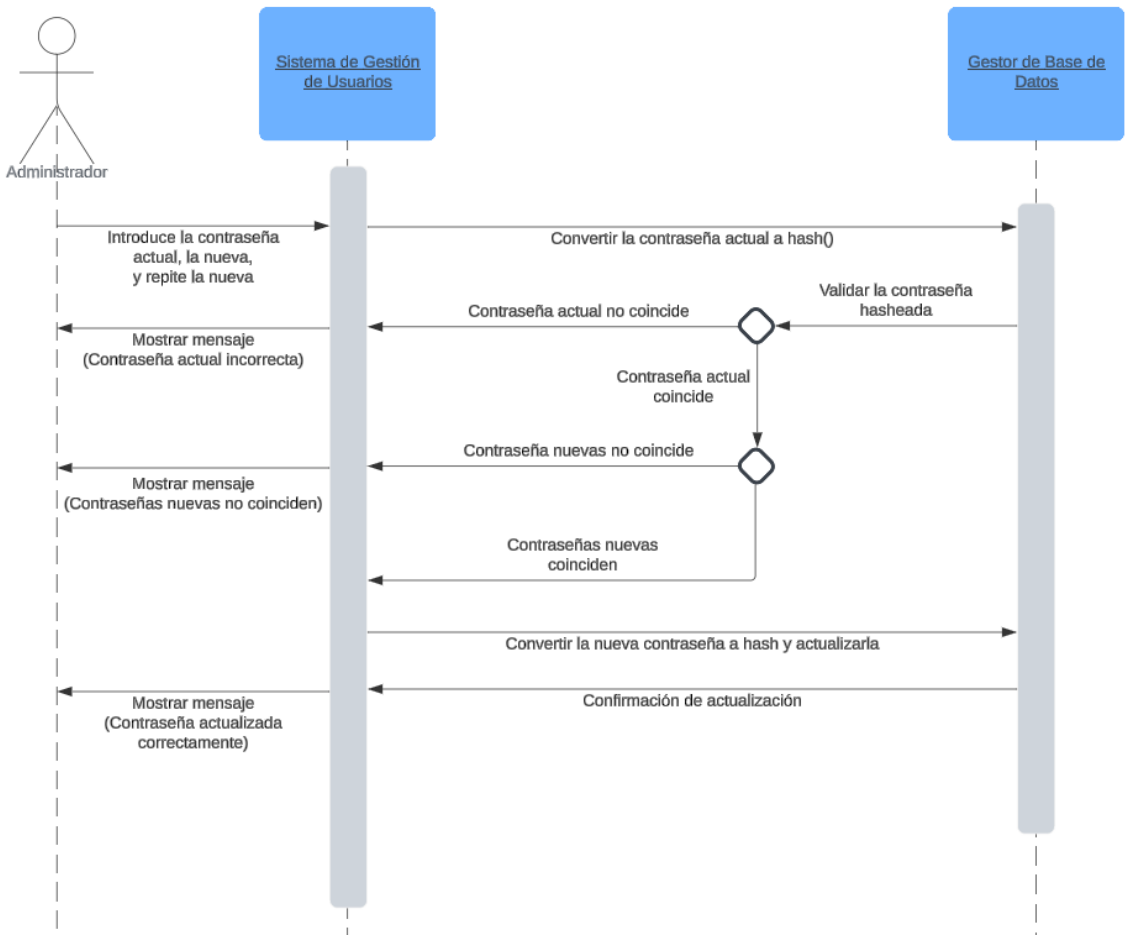


Fuente: Autoría Propia.

Este diagrama de secuencia muestra el proceso de generación y exportación de informes basado en un rango de fechas seleccionado por el administrador. El flujo comienza cuando el administrador selecciona la opción para generar un informe, se le solicita un rango de fechas y se valida si las fechas ingresadas son correctas. Si las fechas son inválidas, se muestra un mensaje de error; si no, el sistema consulta la base de datos en busca de registros dentro del rango indicado. Si no hay registros disponibles, se notifica al administrador; de lo contrario, los registros se muestran en pantalla. Finalmente, el administrador puede optar por exportar los datos, generando un archivo CSV que se guarda o mostrando un mensaje de error si el proceso de exportación falla.

Figura 25

Diagrama de secuencia para cambio de contraseña *(Administrador).*



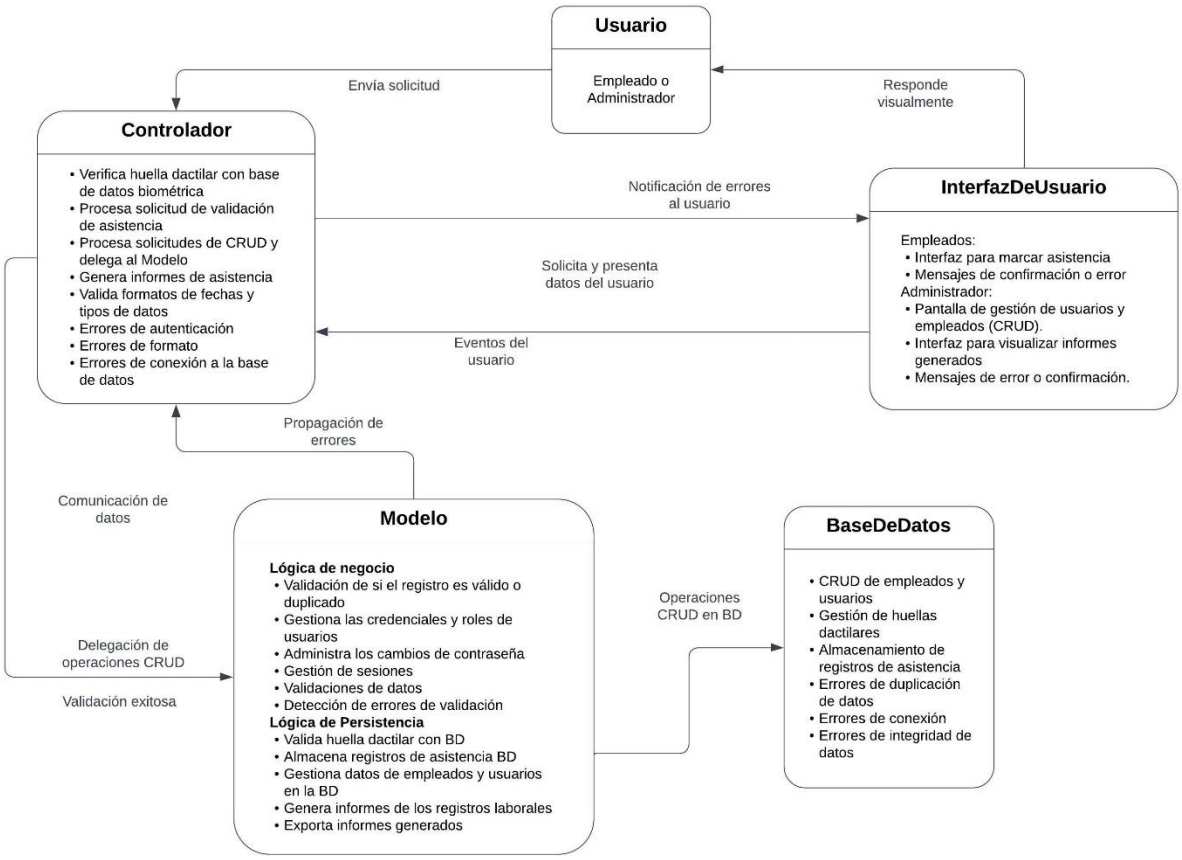
Fuente: Autoría Propia.

El diagrama de secuencia para el cambio de contraseña muestra el proceso paso a paso que sigue un usuario para actualizar su contraseña. Inicia cuando el administrador introduce la contraseña actual, la nueva, y repite la nueva. Luego, el sistema convierte la contraseña actual a un hash y la valida comparándola con la que está almacenada en la base de datos. Si la contraseña no coincide, se muestra un mensaje de error. Si coincide, se verifica si las contraseñas nuevas coinciden entre sí; si no coinciden, se muestra un mensaje de error. Finalmente, si ambas contraseñas coinciden, la nueva contraseña se convierte a hash y se actualiza en la base de datos, confirmando el cambio exitoso al usuario.

Diagrama de componentes de arquitectura MVC

Figura 26

Diagrama de arquitectura MVC.



Fuente: Autoría Propia.

Este diagrama representa la arquitectura MVC (Modelo-Vista-Controlador) del sistema de gestión de asistencia y usuarios.

Usuario (Empleado o Administrador):

El usuario puede ser un empleado o administrador, quienes inician solicitudes hacia el sistema. Estas solicitudes incluyen acciones como el registro de asistencia o la gestión de usuarios y empleados. El sistema responde de manera visual en la Interfaz de Usuario.

Interfaz de Usuario (Vista):

La interfaz de usuario es el punto de interacción visual para los empleados y administradores. Los empleados utilizan la interfaz para marcar su asistencia, recibir mensajes de error o confirmación, y visualizar el estado de sus acciones. Por otro lado, los administradores gestionan usuarios y empleados mediante operaciones CRUD, visualizan informes generados y reciben retroalimentación en caso de errores o confirmaciones de las acciones.

Controlador:

El Controlador actúa como intermediario entre la Interfaz de Usuario y el Modelo. Procesa las solicitudes enviadas por el usuario y realiza varias funciones:

- Validación de huellas dactilares con la base de datos biométrica.
- Procesa las solicitudes de validación de asistencia (asegurándose de que las huellas sean válidas).
- Delegación de operaciones CRUD al modelo para la gestión de empleados y usuarios.
- Validación de formatos de fechas y tipos de datos antes de pasar al modelo.
- Manejo de errores y excepciones: si ocurre un error, el controlador propaga los errores a la interfaz, notificando al usuario directamente.

Modelo:

El modelo contiene toda la lógica de negocio y persistencia de datos del sistema:

Lógica de negocio:

- Validación de si los registros de asistencia son válidos o duplicados.
- Gestión de credenciales y roles de usuarios.
- Administración de los cambios de contraseña y gestión de sesiones.
- Detecta los errores de validación, pero no los maneja directamente.

Lógica de persistencia:

- Comunicación con la base de datos para validar huellas dactilares.
- Almacenamiento de registros de asistencia.
- Gestión de los datos de empleados y usuarios en la base de datos.
- Generación y exportación de informes de los registros laborales.

Base de Datos:

El modelo interactúa con la base de datos, que maneja:

- Operaciones CRUD de empleados y usuarios.
- Gestión de huellas dactilares y almacenamiento de registros de asistencia.
- Manejo de errores de duplicación de datos, errores de conexión, y errores de integridad de datos.

Flujos de Datos:

- El Usuario envía solicitudes mediante la interfaz de usuario.
- El controlador procesa las solicitudes, valida la información y delega operaciones al modelo si todo es correcto.
- El modelo realiza las operaciones en la base de datos y devuelve los resultados o errores al controlador.
- Si ocurren errores en las validaciones o procesos, el controlador los notifica a la interfaz de usuario, mostrando mensajes de error o confirmación.

Este diagrama refleja de manera clara cómo cada componente juega su papel en la arquitectura MVC, garantizando que el flujo de datos, las validaciones, y las interacciones con la base de datos se manejen correctamente, mientras se asegura una retroalimentación adecuada hacia el usuario.

Modelo entidad – relación.

El esquema o base de datos llamada RegistroLaboral se ha diseñado para facilitar la gestión y el seguimiento de los empleados, sus credenciales de usuario, y su asistencia dentro de la organización OptiSalud. Este modelo entidad-relación se compone de tres entidades principales: Empleados, Usuarios, y RegistroLaboral, las cuales se describen a continuación, junto con las relaciones que las interconectan.

Entidades y Atributos

- Empleados

Esta representa a los empleados de OptiSalud, almacenando información esencial sobre cada uno de ellos.

Atributos:

Id_cedula (clave primaria): Identificador único de cada empleado, de tipo INT. Es un campo obligatorio que garantiza la identificación inequívoca de los empleados.

nombres: Nombre(s) del empleado, de tipo VARCHAR(50).

apellidos: Apellido(s) del empleado, de tipo VARCHAR(50).

cargo: Cargo que el empleado ocupa dentro de la organización, de tipo VARCHAR(50).

sede: Ubicación o sede donde el empleado realiza sus funciones, de tipo VARCHAR(50).

huella: Huella digital del empleado, almacenada como un objeto binario largo (LONGBLOB). Este atributo es opcional y permite almacenar datos binarios de gran tamaño.

- Usuarios

Gestiona las cuentas de usuario que permiten el acceso al sistema, vinculando cada cuenta a un empleado específico.

Atributos:

Id (clave primaria): Identificador único y auto incremental para cada cuenta de usuario, de tipo INT.

usuario: Nombre de usuario para el inicio de sesión, de tipo VARCHAR(50).

pass: Contraseña para el inicio de sesión, almacenada de forma segura, de tipo VARCHAR(255).

Rol: Define el rol o permisos asociados a la cuenta de usuario, de tipo VARCHAR(50).

U_cedula (clave foránea): Referencia a Id_cedula en la entidad Empleados, estableciendo una relación directa entre la cuenta de usuario y el empleado correspondiente.

- RegistroLaboral

Esta registra la asistencia de los empleados, detallando las horas de entrada y salida durante la jornada laboral.

Atributos:

Id (clave primaria): Identificador único y auto incremental para cada registro de asistencia, de tipo INT.

fecha: Fecha del registro de asistencia, de tipo DATE.

in_m, out_m: Hora de entrada y salida en la mañana, respectivamente, de tipo DATETIME.

in_t, out_t: Hora de entrada y salida en la tarde, respectivamente, de tipo DATETIME.

cedula (clave foránea): Referencia a Id_cedula en la entidad Empleados, vinculando cada registro de asistencia a un empleado específico.

Relaciones

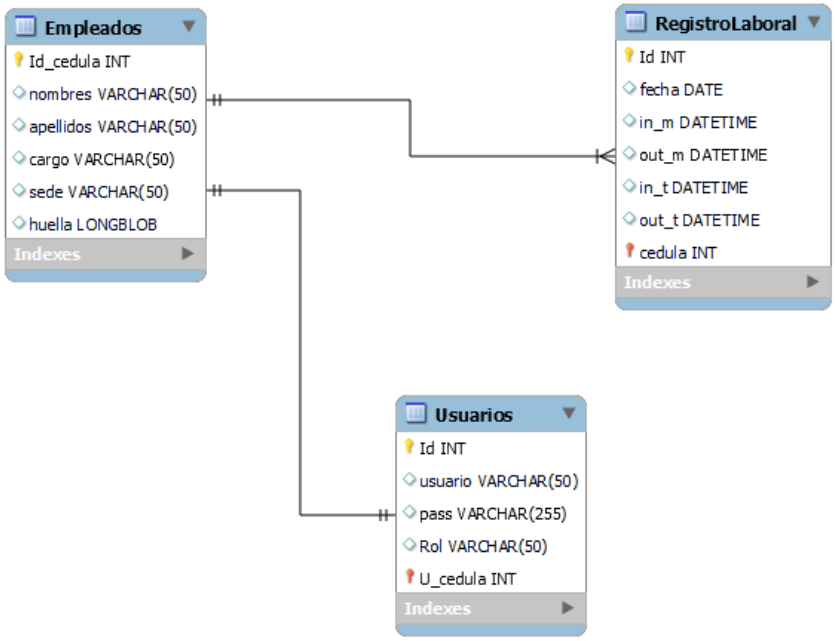
El modelo establece relaciones clave foráneas que integran estas entidades. Estas relaciones son fundamentales para mantener la integridad referencial y asegurar que los datos estén consistentemente relacionados entre sí:

Usuarios - Empleados: La relación entre Usuarios y Empleados se establece a través de la clave foránea U_cedula, permitiendo que cada cuenta de usuario esté directamente relacionada con un empleado.

RegistroLaboral - Empleados: La relación entre RegistroLaboral y Empleados se define por la clave foránea cedula, asegurando que cada registro de asistencia se asocie con el empleado correspondiente.

Figura 27

Modelo entidad – relación.



Fuente: Autoría Propia.

Procedimientos almacenados

- Actualizar Empleado

Este procedimiento es especialmente útil para mantener actualizada la información de los empleados sin necesidad de escribir consultas UPDATE manualmente cada vez.

Figura 28

Código MySql, actualiza empleado.

```
DELIMITER $$

CREATE PROCEDURE `ActualizarEmpleado` (
    IN `p_Id_cedula` INT,
    IN `p_nombres` VARCHAR(50),
    IN `p_apellidos` VARCHAR(50),
    IN `p_cargo` VARCHAR(50),
    IN `p_sede` VARCHAR(50)
)
BEGIN
    UPDATE `empleados`
    SET `nombres` = p_nombres,
        `apellidos` = p_apellidos,
        `cargo` = p_cargo,
        `sede` = p_sede
    WHERE `Id_cedula` = p_Id_cedula;
END$$

DELIMITER ;
```

Fuente: Autoría propia

- Actualizar Empleado Huella

Este procedimiento se utiliza para actualizar la información de un empleado, incluyendo su huella digital, en la base de datos.

Figura 29

Código MySql, actualiza empleado huella.

```
DELIMITER $$

CREATE PROCEDURE `ActualizarEmpleadoHuella` (
    IN `p_Id_cedula` INT,
    IN `p_nombres` VARCHAR(50),
    IN `p_apellidos` VARCHAR(50),
    IN `p_cargo` VARCHAR(50),
    IN `p_sede` VARCHAR(50),
    IN `p_huella` LONGBLOB
)
BEGIN
    UPDATE empleados
    SET nombres = p_nombres,
        apellidos = p_apellidos,
        cargo = p_cargo,
        sede = p_sede,
        huella = p_huella
    WHERE Id_cedula = p_Id_cedula;
END$$

DELIMITER ;
```

Fuente: Autoría propia

- Insertar Empleado

Este procedimiento permite insertar un nuevo empleado en la base de datos.

Figura 30

Código MySql, insertar empleado.

```
DELIMITER $$

CREATE PROCEDURE `InsertarEmpleado` (
    IN `pId_cedula` INT,
    IN `pNombres` VARCHAR(50),
    IN `pApellidos` VARCHAR(50),
    IN `pCargo` VARCHAR(50),
    IN `pSede` VARCHAR(50),
    IN `pHuella` LONGBLOB
)
BEGIN
    INSERT INTO empleados (Id_cedula, nombres, apellidos, cargo, sede, huella)
    VALUES (pId_cedula, pNombres, pApellidos, pCargo, pSede, pHuella);
END$$

DELIMITER ;
```

Fuente: autoría propia.

- Muestra Empleado

Este procedimiento recupera y muestra la información de todos los empleados.

Figura 31

Código MySql, muestra empleado.

```
DELIMITER $$

CREATE PROCEDURE `MuestraEmpleado`()
BEGIN
    SELECT * FROM empleados;
END$$

DELIMITER ;
```

Fuente: autoría propia.

- Obtener Empleado

A diferencia de Muestra Empleado, este procedimiento está diseñado para seleccionar y mostrar información específica (excluyendo la huella digital) de todos los empleados.

Figura 32

Código MySql, obtener empleado.

```
DELIMITER $$

CREATE PROCEDURE `ObtenerEmpleado`()
BEGIN
    SELECT Id_cedula, nombres, apellidos, cargo, sede FROM empleados;
END$$

DELIMITER ;
```

Fuente: autoría propia.

- Obtener Usuarios

Este procedimiento recupera y muestra información básica de los usuarios, excluyendo detalles sensibles como las contraseñas.

Figura 33

Código MySql, obtener usuarios.

```
DELIMITER $$

CREATE PROCEDURE `ObtenerUsuarios`()
BEGIN
    SELECT Id, usuario, rol, U_cedula FROM Usuarios;
END$$

DELIMITER ;
```

Fuente: autoría propia.

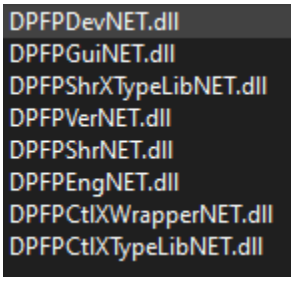
Desarrollo y función de la aplicación

Para el desarrollo de la aplicación, hemos seleccionado el lector de huellas dactilares Digital Persona U.are.U 4450 como componente clave de nuestro sistema de autenticación. Este dispositivo destaca por su precisión y confiabilidad, características esenciales para garantizar la seguridad y eficiencia de nuestra solución.

Para integrar de manera efectiva las funciones del Digital Persona U.are.U 4450 en nuestra aplicación, hemos incorporado las dependencias necesarias dentro del entorno de desarrollo. Esta etapa es crucial para asegurar que el lector de huellas se comunique sin problemas con nuestra aplicación, asegurando un funcionamiento óptimo.

Figura 34

Muestra de dependencias utilizadas.

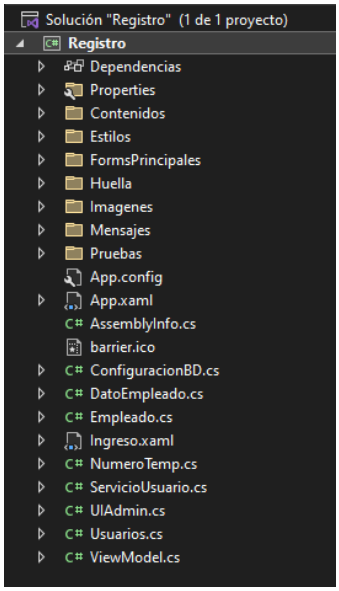


Fuente: autoría propia.

El proyecto ha sido estructurado en Microsoft Visual Studio, siguiendo las mejores prácticas de organización y codificación. La estructura del proyecto está diseñada para facilitar la navegación entre los diferentes componentes y módulos, lo que optimiza el desarrollo y el mantenimiento futuro del código. A continuación, se describe la organización del proyecto en Visual Studio:

Figura 35

Estructura de datos de la aplicación.



Fuente: autoría propia.

Este enfoque estructurado, basado en el patrón de diseño Modelo-Vista-Controlador (MVC), no solo mejora la eficiencia del desarrollo, sino que también contribuye a la escalabilidad y sostenibilidad de la aplicación a largo plazo. El MVC permite separar la lógica de negocio, la interfaz de usuario y el control de flujo, lo que facilita la gestión de cada componente de manera independiente.

La separación de preocupaciones significa que cada módulo o componente del sistema se encarga de una única funcionalidad específica, evitando que las responsabilidades se mezclen. Esto asegura que las actualizaciones, mejoras o correcciones puedan realizarse sin afectar otras partes del sistema, lo que reduce el riesgo de errores y facilita el mantenimiento.

Además, al mantener una clara separación de las dependencias requeridas para el lector de huellas dactilares y otros módulos, estamos sentando las bases para una aplicación robusta, confiable y segura. Este enfoque también mejora la seguridad y permite una mayor flexibilidad para futuras expansiones o integraciones con nuevas tecnologías.

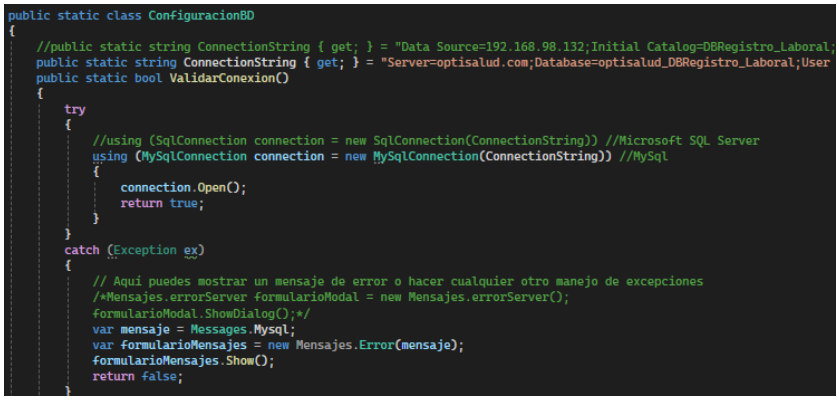
Clases

ConfiguracionBD

Esta clase estática sirve como configuración centralizada para la conexión con la base de datos. Contiene una propiedad llamada ConnectionString que almacena la cadena de conexión necesaria para establecer la comunicación con la base de datos. Además, incluye un método ValidarConexion() que intenta abrir una conexión con la base de datos para verificar si la cadena de conexión es válida; si la conexión es exitosa, retorna true, y en caso de fallar, muestra un mensaje de error y retorna false.

Figura 36

Muestra de código C# de la clase, conexión base de datos.



```
public static class ConfiguracionBD
{
    //public static string ConnectionString { get; } = "Data Source=192.168.98.132;Initial Catalog=DBRegistro_Laboral;
    public static string ConnectionString { get; } = "Server=optisalud.com;Database=optisalud_DBRegistro_Laboral;User
    public static bool ValidarConexion()
    {
        try
        {
            //using (SqlConnection connection = new SqlConnection(ConnectionString)) //Microsoft SQL Server
            using (MySqlConnection connection = new MySqlConnection(ConnectionString)) //MySql
            {
                connection.Open();
                return true;
            }
        }
        catch (Exception ex)
        {
            // Aquí puedes mostrar un mensaje de error o hacer cualquier otro manejo de excepciones
            /*Mensajes.errorServer formularioModal = new Mensajes.errorServer();
            formularioModal.ShowDialog();*/
            var mensaje = Messages.Mysql;
            var formularioMensajes = new Mensajes.Error(mensaje);
            formularioMensajes.Show();
            return false;
        }
```

Fuente: autoría propia.

DatoEmpleado

DatoEmpleado es una clase estática que proporciona métodos para interactuar con la base de datos y realizar operaciones como la carga de empleados o la obtención de registros laborales. Estos registros pueden incluir información de todos los empleados o estar filtrados por un rango de fechas o una fecha específica. La clase utiliza procedimientos almacenados (como MuestraEmpleado y ObtenerEmpleado) para recuperar los datos y mapearlos a objetos

Empleado. Además, DatoEmpleado encapsula toda la lógica necesaria para la comunicación con la base de datos, lo que simplifica el manejo de los datos en otras partes de la aplicación.

Figura 37

Muestra de código C# de la clase, dato empleado.

```
public class DatoEmpleado
{
    public static List<Empleado> MuestraEmpleados()
    {
        List<Empleado> listaEmpleados = new List<Empleado>();
        try
        {
            using (MySqlConnection connection = new MySqlConnection(ConfiguracionBD.ConnectionString))
            {
                connection.Open();
                using (MySqlCommand command = connection.CreateCommand())
                {
                    command.CommandType = CommandType.StoredProcedure;
                    command.CommandText = "MuestraEmpleado";

                    using (MySqlDataReader dr = command.ExecuteReader())
                    {
                        if (dr.HasRows)
                        {
                            while (dr.Read())
                            {
```

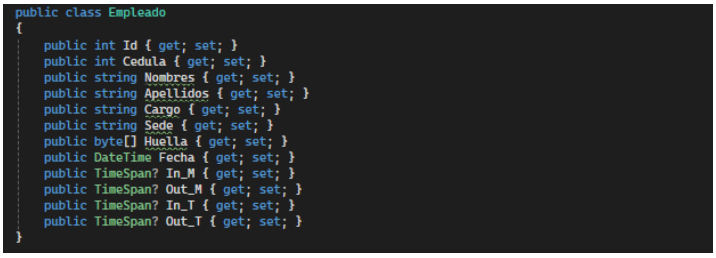
Fuente: autoría propia.

Empleado

Esta clase modela a un empleado dentro del sistema, conteniendo información personal y profesional, como nombres, apellidos, cargo, sede, y datos biométricos (huella digital). También incluye propiedades para registrar tiempos de entrada y salida (mañana y tarde), y una fecha que podría representar, por ejemplo, el día de trabajo al que corresponden esos registros. Es una representación de los datos de un empleado que se almacenarán o se han almacenado en la base de datos.

Figura 38

Muestra de código C# de la clase, empleados.



```
public class Empleado
{
    public int Id { get; set; }
    public int Cedula { get; set; }
    public string Nombres { get; set; }
    public string Apellidos { get; set; }
    public string Cargo { get; set; }
    public string Sede { get; set; }
    public byte[] Huella { get; set; }
    public DateTime Fecha { get; set; }
    public TimeSpan? In_M { get; set; }
    public TimeSpan? Out_M { get; set; }
    public TimeSpan? In_T { get; set; }
    public TimeSpan? Out_T { get; set; }
}
```

Fuente: autoría propia.

NumeroTemp

Es una clase simple que se utiliza para manejar información temporal sobre un usuario. Tiene tres propiedades: Cedula, Nombre, y Rol, que almacenan respectivamente la cédula, el nombre, y el rol de un usuario. Esta clase podría usarse para transferir datos dentro de la aplicación sin necesidad de interactuar con la base de datos.

Figura 39

Muestra de código C# de la clase, numero temporal.

```
public class NumeroTemp
{
    public string Cedula { get; set; }
    public string Nombre { get; set; }
    public string Rol { get; set;}
}
```

Fuente: autoría propia.

ServicioUsuario

ServicioUsuario es una clase que contiene un método estático CargarUsuarios() para cargar usuarios de la base de datos. Este método se conecta a la base de datos usando la cadena de conexión definida en ConfiguracionBD, ejecuta un procedimiento almacenado llamado ObtenerUsuarios, y lee los resultados para construir una lista de objetos Usuarios. En caso de error durante este proceso, se captura la excepción y se muestra un mensaje de error.

Figura 40

Muestra de código C# de la clase, servicios con usuarios.

```
public class ServicioUsuario
{
    public static List<Usuarios> CargarUsuarios()
    {
        List<Usuarios> listaUsuarios = new List<Usuarios>();
        using (MySqlConnection connection = new MySqlConnection(ConfiguracionBD.ConnectionString))
        {
            try
            {
                connection.Open();
                MySqlCommand cmd = new MySqlCommand("ObtenerUsuarios", connection);
                cmd.CommandType = CommandType.StoredProcedure;

                using (MySqlDataReader reader = cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        listaUsuarios.Add(new Usuarios
                        {
                            Id = reader.GetInt32("Id"),
```

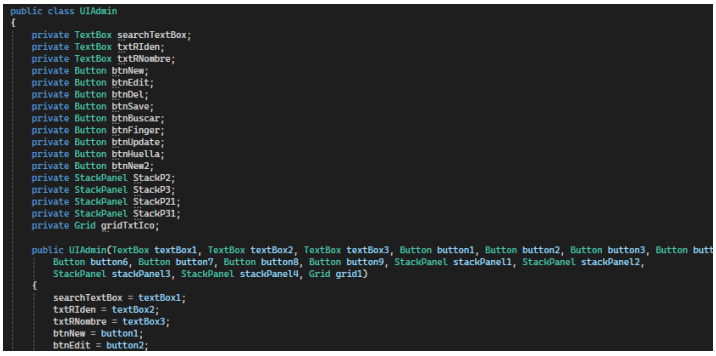
Fuente: autoría propia.

UIAdmin

La clase UIAdmin gestiona elementos de la interfaz de usuario (UI) para la administración, como campos de texto, botones y paneles. Los constructores y métodos de esta clase manejan la visibilidad y el comportamiento de estos elementos en respuesta a acciones del usuario, como crear un nuevo usuario, buscar, editar, entre otros. Por ejemplo, el método BotonNew() prepara la UI para ingresar un nuevo usuario, haciendo visibles ciertos botones y campos de texto, mientras oculta otros.

Figura 41

Muestra de código C# de la clase, UI Admin.



```
public class UIAdmin
{
    private TextBox searchTextBox;
    private TextBox txtRIden;
    private TextBox txtRNombre;
    private Button btnNew;
    private Button btnEdit;
    private Button btnDel;
    private Button btnSave;
    private Button btnBuscar;
    private Button btnFinger;
    private Button btnUpdate;
    private Button btnHuella;
    private Button btnNew2;
    private StackPanel StackP2;
    private StackPanel StackP3;
    private StackPanel StackP21;
    private StackPanel StackP31;
    private Grid gridTxtIco;

    public UIAdmin(TextBox textBox1, TextBox textBox2, TextBox textBox3, Button button1, Button button2, Button button3, Button butt
        Button button6, Button button7, Button button8, Button button9, StackPanel stackPanel1, StackPanel stackPanel2,
        StackPanel stackPanel3, StackPanel stackPanel4, Grid grid1)
    {
        searchTextBox = textBox1;
        txtRIden = textBox2;
        txtRNombre = textBox3;
        btnNew = button1;
        btnEdit = button2;
```

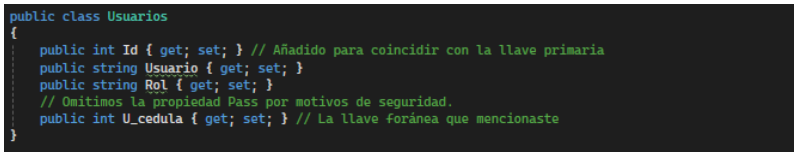
Fuente: autoría propia.

Usuarios

Esta clase representa la entidad Usuarios en la base de datos, diseñada para almacenar y manejar información sobre los usuarios del sistema. Incluye las propiedades básicas como Id, Usuario, Rol, y U_cedula, que se relaciona con la cédula del empleado asociado a este usuario. La propiedad Pass está omitida por razones de seguridad, lo cual es una práctica común para proteger datos sensibles.

Figura 42

Muestra de código C# de la clase, usuarios.



```
public class Usuarios
{
    public int Id { get; set; } // Añadido para coincidir con la llave primaria
    public string Usuario { get; set; }
    public string Rol { get; set; }
    // Omitimos la propiedad Pass por motivos de seguridad.
    public int U_cedula { get; set; } // La llave foránea que mencionaste
}
```

Fuente: autoría propia.

ViewModel

Esta clase implementa la interfaz INotifyPropertyChanged y se utiliza para crear un modelo de vista que soporte el enlace de datos (data binding) reactivos en aplicaciones que utilizan WPF. Utiliza un DispatcherTimer para actualizar periódicamente el color de un elemento de la UI mediante interpolación entre dos colores.

Figura 43

Muestra de código C# de la clase, view model.

```
public class ViewModel : INotifyPropertyChanged
{
    private Brush foregroundBrush;
    private readonly DispatcherTimer timer;

    private readonly Color startColor = (Color)ColorConverter.ConvertFromString("#D5CFF5"); // Color hexadecimal para #D5CFF5
    private readonly Color endColor = (Color)ColorConverter.ConvertFromString("#127369");
    //private readonly Color endColor = Colors.LightCyan; // Color predefinido LightCyan
    private double interpolationFactor = 0.0;
    private bool increasing = true; // Dirección de la interpolación
    private readonly double step = 0.01; // Velocidad de cambio

    public ViewModel()
    {
        foregroundBrush = new SolidColorBrush(startColor);

        timer = new DispatcherTimer();
        timer.Interval = TimeSpan.FromMilliseconds(10); // Ajusta para controlar la velocidad del bucle
        timer.Tick += Timer_Tick;
        timer.Start();
    }

    public Brush ForegroundBrush
    {
```

Fuente: autoría propia.

Estilos y recursos

En el proyecto se implementaron estilos y recursos con el objetivo de estandarizar y mejorar la interfaz de usuario (UI), garantizando una experiencia visual coherente y profesional en toda la aplicación. Estos estilos permiten que elementos como botones, cuadros de texto e iconos mantengan una apariencia uniforme, lo que facilita su reutilización y asegura que el diseño siga principios de consistencia y accesibilidad.

El uso de recursos centralizados no solo mejora la consistencia visual, sino que también simplifica el mantenimiento del código. Al aplicar un enfoque global para gestionar la apariencia de los componentes, es posible realizar cambios de diseño de manera eficiente sin la necesidad de ajustar cada elemento individualmente. Esto optimiza el proceso de desarrollo, permitiendo un ajuste rápido y flexible del diseño, y proporciona una base sólida para la escalabilidad futura del proyecto.

A continuación, se muestran algunos ejemplos de estilos aplicados en la aplicación, que demuestran cómo se gestionan los elementos visuales:

```
        <ResourceDictionary
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:fa="http://schemas.awesome.incremented/wpf/xaml/fontawesome.sharp">

            <!--Estilo para el botón del menú-->
            <Style x:Key="menuButton" TargetType="RadioButton">
                <Setter Property="Height" Value="50"/>
                <Setter Property="Background" Value="Transparent"/>
                <Setter Property="Foreground" Value="{StaticResource plainTextColor3}"/>
                <Setter Property="BorderBrush" Value="Transparent"/>
                <Setter Property="Template">
                    <Setter.Value>
                        <ControlTemplate TargetType="RadioButton">
                            <Border Background="{TemplateBinding Background}"
                                    BorderBrush="{TemplateBinding BorderBrush}">
                                <ContentPresenter HorizontalAlignment="Left"
VerticalAlignment="Center"/>
                            </Border>
                        </ControlTemplate>
                    </Setter.Value>
                </Setter>
                <Style.Triggers>
                    <!-- Estilo cuando el mouse está sobre el botón -->
                    <Trigger Property="IsMouseOver" Value="True">
                        <Setter Property="Background" Value="{StaticResource panelOverColor}"/>
                        <Setter Property="Foreground" Value="{StaticResource titleColor3}"/>
                    </Trigger>
                    <!-- Estilo cuando el botón está activado -->
                    <Trigger Property="IsChecked" Value="True">
                        <Setter Property="Background" Value="{StaticResource panelActiveColor}"/>
```

```
                    <Setter Property="Foreground" Value="{Binding Path=Tag,
RelativeSource={RelativeSource Self}}"/>
                </Trigger>
            </Style.Triggers>
        </Style>

        <!-- Estilo para los iconos del menú -->
        <Style x:Key="menuButtonIcon" TargetType="fa:IconImage">
            <Setter Property="Foreground" Value="{Binding Path=Tag,
RelativeSource={RelativeSource AncestorType=RadioButton}}"/>
            <Setter Property="Width" Value="22"/>
            <Setter Property="Height" Value="22"/>
        </Style>

        <!-- Estilo para el texto del menú -->
        <Style x:Key="menuButtonText" TargetType="TextBlock">
            <Setter Property="Foreground" Value="{Binding Path=Foreground,
RelativeSource={RelativeSource AncestorType=RadioButton}}"/>
            <Setter Property="FontFamily" Value="Montserrat"/>
            <Setter Property="FontWeight" Value="Medium"/>
            <Setter Property="FontSize" Value="13.5"/>
            <Setter Property="VerticalAlignment" Value="Center"/>
        </Style>
    </ResourceDictionary>
```

Estos estilos definen la apariencia de los botones del menú, los iconos y los textos, estandarizando la UI y facilitando su modificación en el futuro. Por ejemplo, el estilo "menuButton" gestiona el aspecto visual de los botones del menú y cómo cambian cuando se pasa el cursor por encima o cuando están seleccionados, mientras que el estilo "menuButtonIcon" controla el tamaño y color de los iconos en función de la interacción del usuario.

Figura 44

Muestra de código XMAL, estilos y recursos.

Fuente: autoría propia.

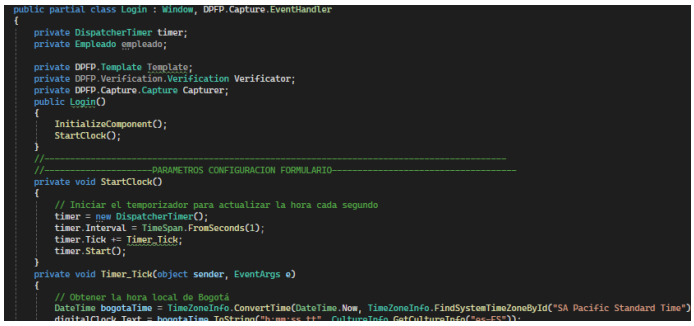
Formularios

Login

El formulario Login está diseñado para autenticar a los usuarios mediante la verificación de huellas dactilares, facilitando así un método seguro y eficiente de acceso. Utiliza la librería DPFP.Verification para integrar el hardware de captura biométrica y comparar las muestras obtenidas con registros preexistentes en una base de datos MySQL, asegurando una gestión precisa de la identidad del usuario.

Figura 45

Muestra de código C# de módulo, Login.



```
public partial class Login : Window, DPFP.Capture.EventHandler
{
    private DispatcherTimer timer;
    private Empleado empleado;

    private DPFP.Template Template;
    private DPFP.Verification.Verification Verificator;
    private DPFP.Capture.Capture Capturer;
    public Login()
    {
        InitializeComponent();
        StartClock();
    }
    //------------------------------------------------------------------------------
    //--------------------PARAMETROS CONFIGURACION FORMULARIO-----------------------
    private void StartClock()
    {
        // Iniciar el temporizador para actualizar la hora cada segundo
        timer = new DispatcherTimer();
        timer.Interval = TimeSpan.FromSeconds(1);
        timer.Tick += Timer_Tick;
        timer.Start();
    }
    private void Timer_Tick(object sender, EventArgs e)
    {
        // Obtener la hora local de Bogotá
        DateTime bogotaTime = TimeZoneInfo.ConvertTime(DateTime.Now, TimeZoneInfo.FindSystemTimeZoneById("SA Pacific Standard Time"));
```

Fuente: autoría propia.

Figura 46

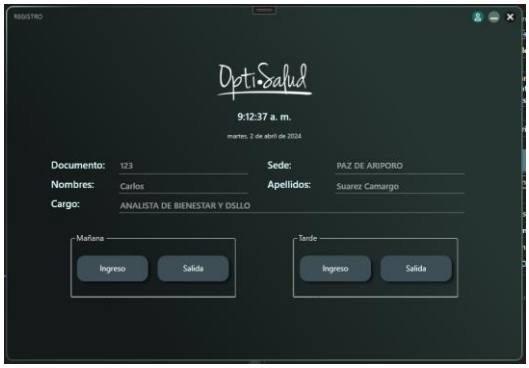
Módulo Login con huella dactilar.

Fuente: autoría propia.

Además, este formulario mejora la interacción del usuario mediante una interfaz dinámica construida con WPF, la cual muestra información relevante como la hora y fecha actual ajustadas a la zona horaria de Bogotá y adapta su diseño según el contexto de uso mediante métodos para ajustar el tamaño y posición de la ventana principal. A través de la implementación de eventos específicos del lector de huellas y la conexión con la base de datos para operaciones como la inserción y actualización de registros, el formulario no solo ofrece una solución de autenticación avanzada, sino que también contribuye a la eficiencia operativa y seguridad de la gestión de accesos en entornos laborales.

Figura 47

Módulo de registro de horario de empleados.



Fuente: autoría propia.

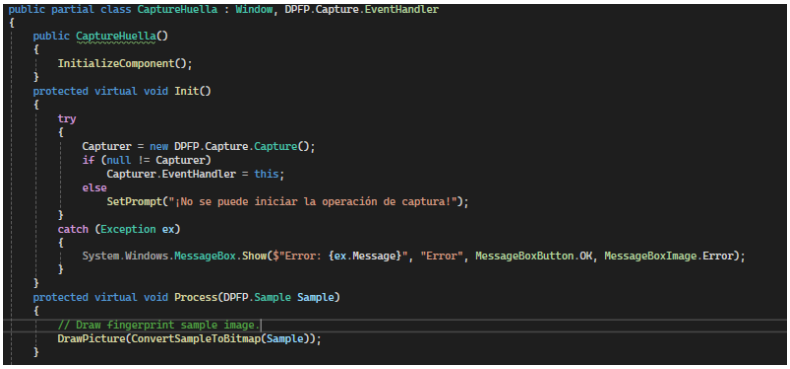
CaptureHuella

El formulario CaptureHuella es una ventana para la captura y procesamiento de huellas dactilares, formando parte esencial de un sistema de autenticación o registro basado en biometría. Implementando la interfaz DPFP.Capture.EventHandler de la librería DigitalPersona, este formulario inicializa el dispositivo de captura de huellas dactilares al cargar la ventana y

maneja eventos críticos como la conexión y desconexión del lector, así como la captura efectiva de la huella.

Figura 48

Muestra de código C# de módulo, captura de huella.



```
public partial class CaptureHuella : Window, DPFP.Capture.EventHandler
{
    public CaptureHuella()
    {
        InitializeComponent();
    }
    protected virtual void Init()
    {
        try
        {
            Capturer = new DPFP.Capture.Capture();
            if (null != Capturer)
                Capturer.EventHandler = this;
            else
                SetPrompt("¡No se puede iniciar la operación de captura!");
        }
        catch (Exception ex)
        {
            System.Windows.MessageBox.Show($"Error: {ex.Message}", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }
    protected virtual void Process(DPFP.Sample Sample)
    {
        // Draw fingerprint sample image.
        DrawPicture(ConvertSampleToBitmap(Sample));
    }
```

Fuente: autoría propia.

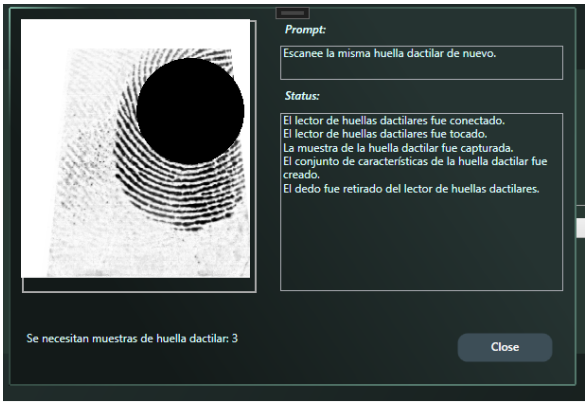
Al capturar una muestra de huella dactilar, el sistema procesa y convierte esta captura en una imagen visible en la interfaz, proporcionando retroalimentación inmediata al usuario sobre la calidad de la huella capturada y el estado del proceso de captura. Además, permite detener el proceso de captura al cerrar la ventana, asegurando una gestión adecuada de los recursos del sistema.

La interacción con el usuario se facilita a través de mensajes y actualizaciones de estado que informan sobre el progreso de la captura. Por ejemplo, la UI muestra mensajes claros para indicar si la captura de la huella dactilar fue exitosa o si hubo un error durante el proceso, como una muestra dactilar de baja calidad o un fallo en la lectura. Estos mensajes proporcionan al usuario una experiencia fluida y comprensible, asegurando que cualquier acción o error sea rápidamente identificado.

Este enfoque no solo refuerza la seguridad a través de la verificación biométrica, sino que también mejora la experiencia del usuario al ofrecer una interfaz clara y responsiva para la interacción con el sistema de captura de huellas dactilares.

Figura 49

Módulo para captura de huella dactilar.



Fuente: autoría propia.

EnrollmentWindow

El formulario EnrollmentWindow, heredando de CaptureHuella, se especializa en el registro y enrolamiento de huellas dactilares en el sistema, extendiendo las capacidades básicas de captura de huellas para incluir el procesamiento y almacenamiento de características únicas de huellas dactilares necesarias para crear un template de huella dactilar completo y utilizable.

Figura 50

Muestra de código C# de módulo, de alojamiento de huella.

```
public partial class EnrollmentWindow : CaptureHuella
{
    public delegate void OnTemplateEventHandler(DPFP.Template template);

    public event OnTemplateEventHandler OnTemplate;
    protected override void Init()
    {
        base.Init();
        base.Title = "Registro de Huella Dactilar";
        Enroller = new DPFP.Processing.Enrollment();
        UpdateStatus();
    }

    protected override void Process(DPFP.Sample Sample)
    {
        base.Process(Sample);

        DPFP.FeatureSet features = ExtractFeatures(Sample, DPFP.Processing.DataPurpose.Enrollment);

        if (features != null)
        {
            try
            {
                MakeReport("El conjunto de características de la huella dactilar fue creado.");
                Enroller.AddFeatures(features);
```

Fuente: autoría propia.

Al iniciar, este formulario establece su propósito específico mediante la inicialización de un objeto DPFP.Processing.Enrollment, que se utiliza para recopilar suficientes muestras válidas de una huella dactilar con el fin de generar un template completo. El template es un objeto que almacena temporalmente las muestras de huellas dactilares en el aplicativo, necesarias para completar el proceso de verificación posterior.

Durante el proceso de captura, cada muestra de huella dactilar se procesa no solo para su visualización, sino también para evaluar si cumple con los criterios requeridos para el enrolamiento. Las características válidas de cada muestra se agregan al Enroller (enrolador). A medida que se recopilan las características, el formulario proporciona retroalimentación continua sobre el número de muestras restantes que se necesitan para completar el template.

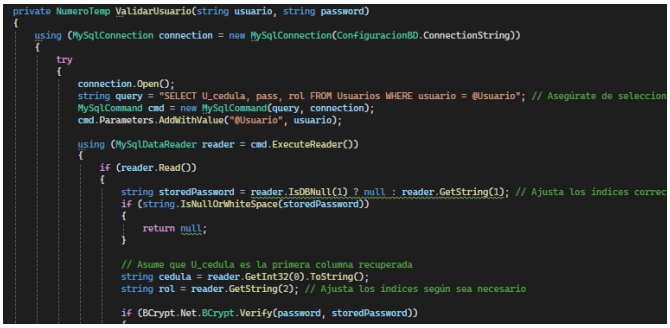
Una vez que se han recopilado suficientes datos y el template de la huella dactilar está completo, se dispara un evento (OnTemplate) para notificar a otros componentes del sistema que el enrolamiento ha sido exitoso. Esto permite que se realicen procedimientos posteriores, como la verificación de la huella. Este enfoque detallado garantiza un registro preciso y eficiente de los usuarios en sistemas que requieren autenticación biométrica, destacando la importancia de recopilar datos biométricos de alta calidad para asegurar la seguridad y funcionalidad del sistema.

InicioSesion

El formulario InicioSesion es la interfaz principal para la autenticación de usuarios en el panel administrativo de la aplicación, diseñado para ofrecer un método seguro y eficiente de inicio de sesión mediante la introducción de credenciales. Al lanzarse, el formulario coloca el foco automáticamente en el campo de usuario para agilizar el proceso de entrada. Este proceso de autenticación no se realiza diariamente por todos los empleados, sino que está reservado para el acceso exclusivo del área de Recursos Humanos, quienes gestionan los aspectos administrativos del sistema.

Figura 51

Muestra de código C# de módulo, inicio de sesión.



```
private NumeroTemp ValidarUsuario(string usuario, string password)
{
    using (MySqlConnection connection = new MySqlConnection(ConfiguracionBD.ConnectionString))
    {
        try
        {
            connection.Open();
            string query = "SELECT U_cedula, pass, rol FROM Usuarios WHERE usuario = @Usuario"; // Asegúrate de seleccion
            MySqlCommand cmd = new MySqlCommand(query, connection);
            cmd.Parameters.AddWithValue("@Usuario", usuario);

            using (MySqlDataReader reader = cmd.ExecuteReader())
            {
                if (reader.Read())
                {
                    string storedPassword = reader.IsDBNull(1) ? null : reader.GetString(1); // Ajusta los indices correc
                    if (string.IsNullOrWhiteSpace(storedPassword))
                    {
                        return null;
                    }

                    // Asume que U_cedula es la primera columna recuperada
                    string cedula = reader.GetInt32(0).ToString();
                    string rol = reader.GetString(2); // Ajusta los indices según sea necesario

                    if (BCrypt.Net.BCrypt.Verify(password, storedPassword))
```

Fuente: autoría propia.

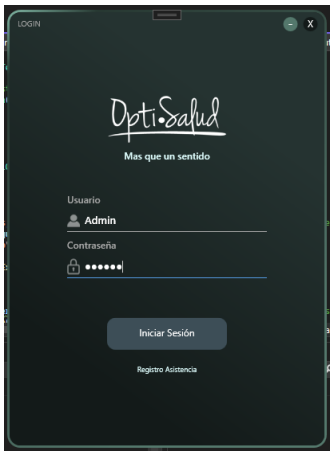
Una característica notable es la implementación de un evento de tecla para el campo de contraseña, permitiendo que el usuario presione "Enter" para iniciar sesión, emulando la acción del botón de ingreso y mejorando la experiencia del usuario. Además, el formulario permite arrastrar la ventana desde cualquier punto de su superficie, aumentando la usabilidad. Los botones presentes permiten minimizar la ventana, cerrar la aplicación completamente o proceder con el inicio de sesión tras validar las credenciales contra registros en una base de datos MySQL, utilizando hashing seguro para comparar las contraseñas.

En caso de una validación exitosa, se accede al panel administrativo de la aplicación, específicamente diseñado para el área de Recursos Humanos. Dependiendo del rol del usuario (administrador o usuario estándar), el sistema mostrará las opciones y formularios correspondientes. Por ejemplo, los administradores tienen acceso a funcionalidades avanzadas, como la gestión de usuarios o la configuración del sistema, mientras que los usuarios estándar pueden acceder únicamente a las funciones necesarias para gestionar registros laborales y reportes de asistencia. En caso de un intento fallido de inicio de sesión, se notifica al usuario mediante un mensaje de error.

Este enfoque de autenticación no solo asegura la protección de datos sensibles, sino que también proporciona una transición fluida hacia las funcionalidades administrativas de la aplicación, permitiendo que solo los usuarios autorizados accedan a los formularios adecuados según su rol.

Figura 52

Módulo de inicio de sesión.



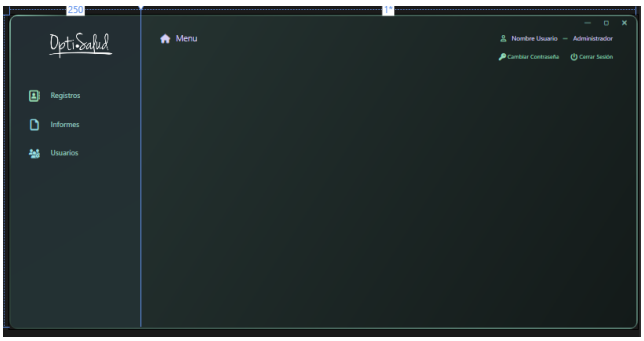
Fuente: autoría propia.

Administración

El formulario Administracion es una interfaz de gestión centralizada para la aplicación, diseñada para ofrecer a los usuarios autenticados acceso a diferentes áreas de funcionalidad basadas en su rol y permisos. Al inicializarse, establece la interfaz predeterminada al área de "Registro" y muestra la información del usuario autenticado, como su nombre, rol, y cédula, proporcionando un contexto claro de quién está utilizando la aplicación.

Figura 53

Modulo administración.



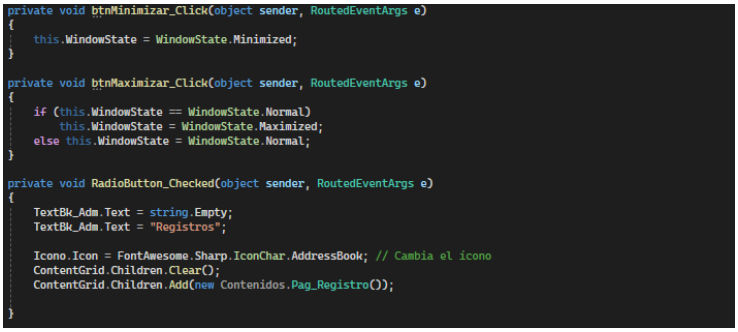
Fuente: autoría propia.

Esta ventana maneja la interacción del usuario con elementos de navegación como botones y radio buttons para alternar entre diferentes vistas de contenido como registros, informes, y gestión de usuarios, cargando dinámicamente el contenido relevante dentro de un área designada de la ventana. Los eventos de interacción visual, como el cambio de color de botones al pasar el ratón, mejoran la experiencia del usuario, haciendo la interfaz intuitiva y amigable. Funcionalidades adicionales incluyen la capacidad de cerrar sesión, regresando al formulario de inicio de sesión para cambiar de usuario, y la opción de cambiar la contraseña, lo que subraya la importancia de la seguridad y personalización en la experiencia del usuario. Mediante el uso de controles personalizados y la integración de librerías externas para íconos, Administracion se presenta como el núcleo desde el cual los usuarios pueden acceder y

gestionar eficientemente las diversas capacidades del sistema, asegurando una administración efectiva y segura de la información y los procesos dentro de la aplicación.

Figura 54

Muestra de código C# de módulo, administración.



```
private void btnMinimizar_Click(object sender, RoutedEventArgs e)
{
    this.WindowState = WindowState.Minimized;
}

private void btnMaximizar_Click(object sender, RoutedEventArgs e)
{
    if (this.WindowState == WindowState.Normal)
        this.WindowState = WindowState.Maximized;
    else this.WindowState = WindowState.Normal;
}

private void RadioButton_Checked(object sender, RoutedEventArgs e)
{
    TextBk_Adm.Text = string.Empty;
    TextBk_Adm.Text = "Registros";

    Icono.Icon = FontAwesome.Sharp.IconChar.AddressBook; // Cambia el icono
    ContentGrid.Children.Clear();
    ContentGrid.Children.Add(new Contenidos.Pag_Registro());

}
```

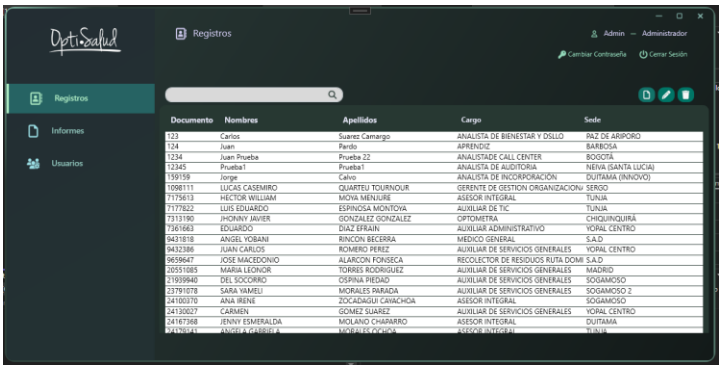
Fuente: autoría propia.

Pag_Registro

El formulario Pag_Registro es un componente crucial dentro de una aplicación de administración, dedicado a la gestión de registros de empleados, mostrando una lista actualizable de empleados y proporcionando funcionalidades para agregar, editar, o eliminar registros. Inicializado con una interfaz intuitiva que carga automáticamente los datos de los empleados desde una base de datos utilizando MySQL y los presenta en un DataGrid, este formulario permite a los usuarios buscar específicamente empleados por nombre, apellido, cédula, cargo, o sede mediante un filtro de búsqueda dinámico.

Figura 55

Modulo registro de empleado.

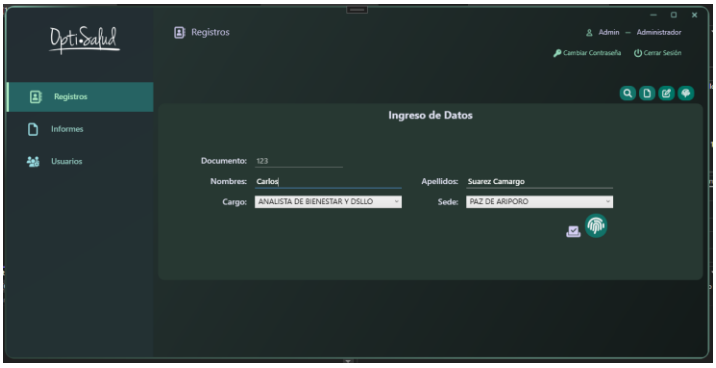


Fuente: autoría propia.

Además, la aplicación integra la capacidad de interactuar con el sistema de captura de huellas dactilares para asociar o verificar las huellas de los empleados, reforzando la seguridad y la unicidad de cada registro. Los administradores tienen la opción de modificar los datos de los empleados, incluyendo su información personal y biométrica, a través de interfaces claras que permiten realizar acciones específicas como nuevo registro, edición y eliminación de datos, así como la actualización de contraseñas y la gestión de accesos según el rol y las credenciales de cada empleado.

Figura 56

Módulo de creación, edición, de empleado.



Fuente: autoría propia.

Este enfoque modular y seguro para la gestión de empleados no solo mejora la eficiencia operativa, sino que también asegura la integridad y confidencialidad de los datos dentro del sistema.

Figura 57

Muestra de código C# de módulo, registro de empleado.

```
public partial class Pag_Registro : UserControl
{
    private UIAdmin uiAdmin;
    private ObservableCollection<Empleado> listaEmpleado = new ObservableCollection<Empleado>();
    private ICollectionView vistaEmpleado;
    public Pag_Registro()
    {
        InitializeComponent();
        CargarEmpleadoDG();
        vistaEmpleado = CollectionViewSource.GetDefaultView(listaEmpleado);
        datagridEmpleados.ItemsSource = listaEmpleado;
        CargarCargos();
        CargarSedes();
        datagridEmpleados.HeadersVisibility = DataGridHeadersVisibility.None;
        datagridEmpleados.SelectionUnit = DataGridSelectionUnit.FullRow;
        datagridEmpleados.IsReadOnly = true;
        datagridEmpleados.ColumnHeaderStyle = null;
        //Funciones de los botones
        uiAdmin = new UIAdmin(searchTextBox, txtRIden, txtRNombre, btnNew, btnEdit, btnDel, btnSave, btnBuscar, btnFinger, btnUpdate,
            StackP2, StackP3, StackP21, StackP31, gridTxtIco);


    }
    private void CargarEmpleadoDG()
    {
        listaEmpleado.Clear();
        var empleados = DatoEmpleado.CargarEmpleadoDG(); // Asume que este método devuelve List<Usuarios>
        foreach (var empleado in empleados)
```

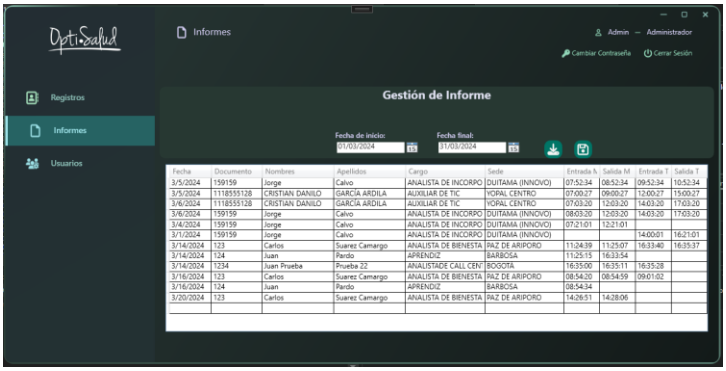
Fuente: autoría propia.

Informe

El formulario Informe es una herramienta de reporte dentro de una aplicación de gestión de empleados, diseñada para visualizar y exportar registros laborales basados en criterios de fecha específicos.

Figura 58

Módulo informe.



Fuente: autoría propia.

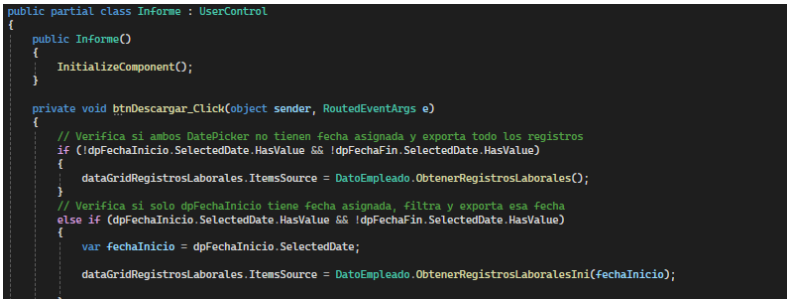
Al inicializarse el formulario, se presenta una interfaz de usuario simple pero funcional que permite seleccionar un rango de fechas mediante controles de tipo DatePicker. El botón btnDescargar filtra y muestra en un DataGrid los registros laborales que coinciden con las fechas especificadas, brindando la opción de exportar toda la información disponible o ajustar la visualización a un intervalo de fechas específico. Si las fechas no se eligen correctamente o el rango no es válido, se muestra un mensaje de error claro y directo.

El formulario también incluye un botón btnGuardar, que permite exportar los datos visualizados a un archivo CSV para una posterior revisión o análisis fuera de la aplicación. Esto facilita la supervisión operativa y el análisis de la productividad laboral, asegurando que los datos sean fácilmente accesibles y manejables.

Este proceso de generación de informes mejora la funcionalidad de la aplicación, proporcionando un soporte valioso para la gestión eficiente de datos en entornos empresariales.

Figura 59

Muestra de código C# de módulo, servicios con usuarios.



```
public partial class Informe : UserControl
{
    public Informe()
    {
        InitializeComponent();
    }

    private void btnDescargar_Click(object sender, RoutedEventArgs e)
    {
        // Verifica si ambos DatePicker no tienen fecha asignada y exporta todo los registros
        if (!dpFechaInicio.SelectedDate.HasValue && !dpFechaFin.SelectedDate.HasValue)
        {
            dataGridRegistrosLaborales.ItemsSource = DatoEmpleado.ObtenerRegistrosLaborales();
        }
        // Verifica si solo dpFechaInicio tiene fecha asignada, filtra y exporta esa fecha
        else if (dpFechaInicio.SelectedDate.HasValue && !dpFechaFin.SelectedDate.HasValue)
        {
            var fechaInicio = dpFechaInicio.SelectedDate;

            dataGridRegistrosLaborales.ItemsSource = DatoEmpleado.ObtenerRegistrosLaboralesIni(fechaInicio);
```

Fuente: autoría propia.

Users

El formulario Users es un componente diseñado para la gestión de usuarios dentro de la aplicación, permitiendo visualizar, crear y eliminar usuarios a través de una interfaz clara y efectiva. Al inicializarse, muestra una lista de usuarios en un DataGrid, con la opción de realizar búsquedas dinámicas por nombre de usuario, rol o documento, utilizando un sistema de filtrado en el campo de búsqueda.

Este formulario permite agregar nuevos usuarios o eliminar los existentes, integrando campos de entrada para capturar información relevante, como documento, nombre de usuario, contraseña y rol. Antes de guardar los datos, se realizan validaciones para asegurar que toda la información necesaria esté completa y correcta.

Figura 60

Módulo de registro de usuarios.

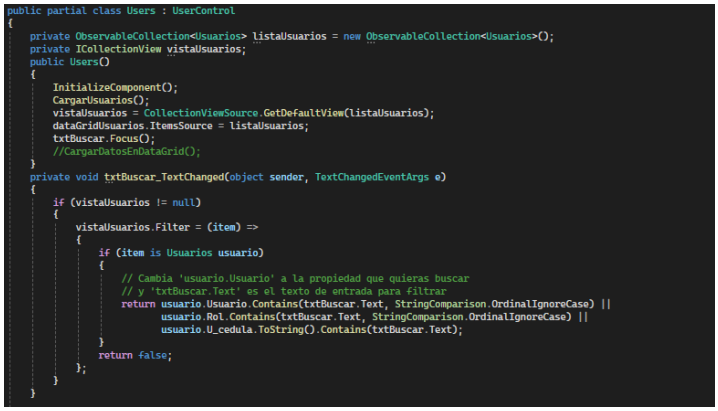
Fuente: autoría propia.

La creación de un nuevo usuario implica la generación de un hash seguro para la contraseña, garantizando un alto nivel de seguridad en la información de autenticación. Cuando se desea eliminar un usuario, se presenta una confirmación para evitar borrados accidentales, preservando la integridad de los datos de la aplicación. Además, el formulario incluye funciones para limpiar los campos de entrada y validar que no se creen usuarios duplicados en función de

su documento, mostrando mensajes informativos o de error según sea necesario para guiar al usuario durante el proceso de gestión.

Figura 61

Muestra de código C# de módulo, usuarios.



```
public partial class Users : UserControl
{
    private ObservableCollection<Usuarios> listaUsuarios = new ObservableCollection<Usuarios>();
    private ICollectionView vistaUsuarios;
    public Users()
    {
        InitializeComponent();
        CargarUsuarios();
        vistaUsuarios = CollectionViewSource.GetDefaultView(listaUsuarios);
        dataGridUsuarios.ItemsSource = listaUsuarios;
        txtBuscar.Focus();
        //CargarDatosEnDataGrid();
    }
    private void txtBuscar_TextChanged(object sender, TextChangedEventArgs e)
    {
        if (vistaUsuarios != null)
        {
            vistaUsuarios.Filter = (item) =>
            {
                if (item is Usuarios usuario)
                {
                    // Cambia 'usuario.Usuario' a la propiedad que quieras buscar
                    // y 'txtBuscar.Text' es el texto de entrada para filtrar
                    return usuario.Usuario.Contains(txtBuscar.Text, StringComparison.OrdinalIgnoreCase) ||
                           usuario.Rol.Contains(txtBuscar.Text, StringComparison.OrdinalIgnoreCase) ||
                           usuario.U_cedula.ToString().Contains(txtBuscar.Text);
                }
                return false;
            };
        }
    }
```

Fuente: autoría propia.

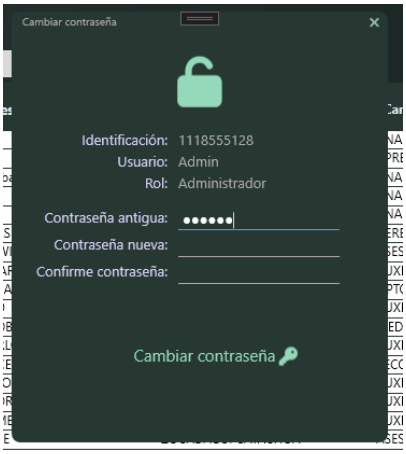
Esta implementación refleja un diseño centrado en la interfaz de usuario para la administración de cuentas en sistemas complejos, facilitando el manejo eficiente de las credenciales de acceso y roles. Esto es fundamental para asegurar la protección de datos y mantener la organización del sistema de manera efectiva.

CambioPass

El formulario CambioPass está diseñado para permitir la actualización segura de la contraseña dentro de la aplicación. Al abrirse, solicita ingresar la contraseña actual y la nueva que se desea establecer, garantizando que el cambio de contraseña sea consciente y esté bajo control del usuario.

Figura 62

Módulo de cambio de contraseña.



Fuente: autoría propia.

El funcionamiento del formulario incluye varios pasos críticos para garantizar la seguridad y la integridad del sistema:

- Verificación de Usuario:

Antes de permitir el cambio de contraseña, el sistema verifica la identidad del usuario comprobando que la contraseña antigua introducida corresponda con la almacenada en la base de datos. Esto se logra mediante una consulta a la base de datos filtrando por la cédula del usuario y utilizando funciones de hash seguras para comparar la contraseña.

- Confirmación de la Nueva Contraseña:

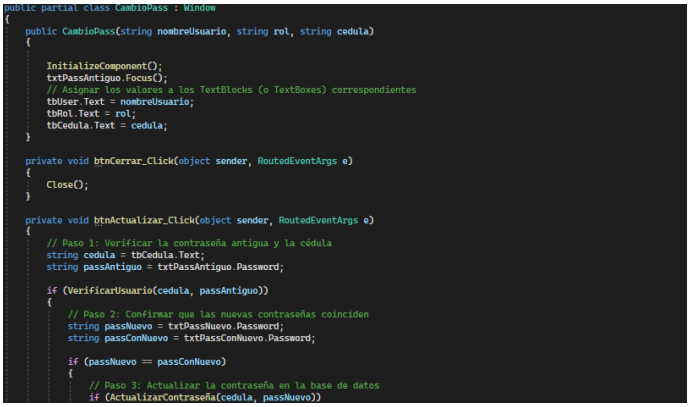
El sistema asegura que la nueva contraseña introducida y su confirmación coincidan, evitando errores de tipeo o malentendidos por parte del usuario al establecer su nueva contraseña.

- Actualización de la Contraseña:

Una vez verificada la identidad del usuario y confirmada la nueva contraseña, el sistema procede a actualizar la contraseña en la base de datos, reemplazando la antigua por la nueva contraseña hash.

Figura 63

Muestra de código C# de módulo, cambiar contraseña.



```
public partial class CambioPass : Window
{
    public CambioPass(string nombreUsuario, string rol, string cedula)
    {

        InitializeComponent();
        txtPassAntiguo.Focus();
        // Asignar los valores a los TextBlocks (o TextBoxes) correspondientes
        tblUser.Text = nombreUsuario;
        tbRol.Text = rol;
        tbCedula.Text = cedula;
    }

    private void btnCerrar_Click(object sender, RoutedEventArgs e)
    {
        Close();
    }

    private void btnActualizar_Click(object sender, RoutedEventArgs e)
    {
        // Paso 1: Verificar la contraseña antigua y la cédula
        string cedula = tbCedula.Text;
        string passAntiguo = txtPassAntiguo.Password;

        if (VerificarUsuario(cedula, passAntiguo))
        {
            // Paso 2: Confirmar que las nuevas contraseñas coinciden
            string passNuevo = txtPassNuevo.Password;
            string passConNuevo = txtPassConNuevo.Password;

            if (passNuevo == passConNuevo)
            {
                // Paso 3: Actualizar la contraseña en la base de datos
                if (ActualizarContraseña(cedula, passNuevo))
```

Fuente: autoría propia.

Este proceso no solo mejora la experiencia del usuario al ofrecer una manera intuitiva y segura de gestionar sus credenciales de acceso, sino que también refuerza la seguridad del sistema asegurando que solo el propietario legítimo de la cuenta pueda realizar cambios en sus credenciales de acceso. Al cerrar el formulario, se deja al usuario en el contexto original de la aplicación, permitiendo continuar con su sesión de manera ininterrumpida y con la seguridad de que su contraseña ha sido actualizada correctamente.

Árbol de carpetas del proyecto

```
Registro/
├─ Dependencias/
│      ├─ Analizadores
│      ├─ Ensamblados/
│      │      ├─ DPFPCLTypelibNET
│      │      ├─ DPFPCTLWrapperNET
│      │      ├─ DPFPCtlXTypeLibNET
│      │      ├─ DPFPDevNET
│      │      ├─ DPFPEngrNET
│      │      ├─ DPFPGuiNET
│      │      ├─ DPFPShrNET
│      │      ├─ DPFPShrXTypeLibNET
│      │      ├─ DPFPVerNET
│      ├─ Marcos de trabajo
│      ├─ Paquetes/
│      │      ├─ BCrypt.Net-Next (4.0.3)
│      │      ├─ FontAwesome.Sharp (6.3.0)
```

```
│   │   ├── MySql.Data (8.3.0)
│   │   ├── System.Data.SqlClient (4.8.6)
│   │   └── System.Drawing.Common (8.0.2)
├── Properties
├── Contenidos/
│   ├── CambioPass.xaml
│   ├── Informe.xaml
│   ├── Pag_New.xaml
│   ├── Pag_Registro.xaml
│   └── Users.xaml
├── Estilos/
│   ├── BotonEstilos.xaml
│   ├── ComboBoxStyles.xaml
│   ├── EstilosBotonBarra.xaml
│   └── UIColor.xaml
├── FormsPrincipales/
│   ├── Administracion.xaml
│   └── InicioSesion.xaml
├── Huella/
│   ├── CaptureHuella.xaml
│   ├── EnrollmentH.xaml
│   └── Login.xaml
├── Imagenes/
│   ├── Ico_B.png
│   ├── IcoCand.png
│   ├── icoError.png
│   ├── IconUs.png
│   ├── IconUser.png
│   ├── Inform.png
│   ├── LogoOptisalud.png
│   └── Logop.png
├── Mensajes/
│   ├── Advertencia.xaml
│   ├── Error.xaml
│   ├── Informacion.xaml
│   └── Messages.resx
├── Pruebas/
│   ├── Anima.xaml
│   └── Ver.xaml
├── App.config
├── App.xaml
├── AssemblyInfo.cs
├── Barrier.ico
├── ConfiguracionBD.cs
├── DatoEmpleado.cs
├── Empleado.cs
├── Ingreso.xaml
├── NumeroTemp.cs
├── ServicioUsuario.cs
├── UIAdmin.cs
```

```
│─ Usuarios.cs
│─ ViewModel.cs
```

Descripción de Carpetas y Archivos

Dependencias

Contiene bibliotecas y componentes externos utilizados en el proyecto, incluyendo controladores de dispositivos para el lector de huellas y otros ensamblados necesarios.

Marcos de trabajo y Paquetes

Alberga los paquetes NuGet utilizados en el proyecto, que proporcionan funcionalidades adicionales como encriptación, manejo de iconos, conexión a bases de datos y operaciones gráficas.

Properties

Incluye archivos de configuración y metadatos del proyecto, como la información de ensamblado y las configuraciones de recursos y ajustes.

Contenidos

Esta carpeta contiene subformularios que se utilizan dentro del formulario principal de Administración. Cada subformulario está diseñado para manejar diferentes aspectos de la gestión administrativa, permitiendo una navegación fluida y organizada dentro de la aplicación.

Estilos

Almacena archivos de estilo CSS que definen la apariencia de los botones, fuentes y otros elementos de la interfaz de usuario. Estos estilos aseguran una experiencia visual coherente y atractiva a lo largo de toda la aplicación.

FormsPrincipales

Contiene los formularios principales que constituyen la interfaz de usuario central de la aplicación. Estos formularios son esenciales para la interacción del usuario y están diseñados para facilitar el acceso a las funcionalidades clave del sistema.

Huella

Incluye los formularios y los componentes necesarios para la captura y verificación de huellas digitales, así como la interfaz de inicio de sesión mediante autenticación biométrica. Esta carpeta es crucial para la seguridad y el control de acceso dentro de la aplicación.

Imágenes

Contiene todas las imágenes utilizadas en el proyecto, incluyendo iconos, logos y gráficos. Estas imágenes son fundamentales para la estética y la identidad visual de la aplicación.

Mensajes

Alberga los archivos que gestionan los mensajes emergentes y las alertas que se muestran en función de diferentes situaciones dentro de la aplicación. También incluye un registro de todos los mensajes estándar utilizados, facilitando la modificación y el mantenimiento de las comunicaciones en la aplicación.

Pruebas

Esta carpeta contiene los scripts y los componentes utilizados para realizar pruebas de la aplicación, incluyendo pruebas unitarias y de integración. Estos elementos son esenciales para garantizar la estabilidad y el funcionamiento correcto del software antes de su despliegue final.

App.config

Archivo de configuración global que define parámetros operativos del sistema, como conexiones a bases de datos y configuraciones específicas de entorno.

App.xaml

Define los recursos globales de la aplicación WPF, como estilos y templates, que se utilizan a lo largo de toda la interfaz gráfica.

ConfiguracionBD.cs

Clase que gestiona la configuración de las conexiones a la base de datos, incluyendo los parámetros de conexión y las cadenas de conexión.

DatoEmpleado.cs

Clase que modela la información relacionada con los empleados, utilizada para operaciones de gestión de datos del personal.

Empleado.cs

Define la estructura y comportamiento de los objetos empleado dentro de la aplicación, incluyendo propiedades como nombre, ID, y Sede.

Ingreso.xaml

Formulario XAML que gestiona la interfaz de ingreso de usuarios, incluyendo autenticación y entrada al sistema.

NumeroTemp.cs

Clase utilizada para manejar números temporales o datos volátiles que requieran almacenamiento durante una sesión.

ServicioUsuario.cs

Clase que proporciona servicios relacionados con la gestión de usuarios, como la creación, modificación y eliminación de perfiles de usuario.

UIAdmin.cs

Clase que maneja la interfaz de administración, incluyendo la lógica de interacción y la visualización de formularios relacionados.

Usuarios.cs

Clase encargada de gestionar los datos y la lógica asociada a los usuarios de la aplicación.

ViewModel.cs

Clase que funciona como el modelo de vista para la aplicación, facilitando la comunicación entre la UI y la lógica.

Repositorio del proyecto

Acceso a proyecto

Abrir Acceso

Conclusiones

El sistema SCESO-Soft se presenta como una herramienta con el potencial de mejorar la precisión y eficiencia en el registro de tiempos laborales. A través de la automatización del proceso, se espera minimizar los errores humanos que eran comunes en el sistema manual, lo que debería reducir las discrepancias en los registros de tiempo a medida que se implemente y utilice de forma continua.

La implementación del sistema ayudara a reducir prácticas indeseadas, como la falsificación o manipulación de registros laborales. La tecnología de validación biométrica ha sido clave para asegurar que los registros sean auténticos, evitando situaciones en las que empleados puedan registrar horas que no corresponden con su tiempo real de trabajo. Este enfoque ha permitido un control más riguroso y preciso de las horas trabajadas, garantizando la integridad de los datos.

El sistema tiene el potencial de facilitar la generación de informes precisos y oportunos, lo que podría mejorar la toma de decisiones en OptiSalud. Aunque actualmente no se cuenta con informes que evidencien esta afirmación, se espera que, a medida que el sistema sea utilizado de manera continua, puedan disponer de datos confiables para evaluar el rendimiento laboral, planificar recursos y tomar decisiones fundamentadas sobre la gestión de los empleados.

La encuesta de viabilidad mostró una amplia aceptación del sistema entre los empleados. Todos los participantes expresaron su acuerdo con la implementación del sistema automatizado, lo que indica una valoración positiva de sus beneficios y una buena disposición hacia el cambio.

A pesar de la disposición positiva hacia el sistema, la encuesta reveló una falta de familiaridad con los sistemas automatizados, lo que resalta la necesidad de programas de

capacitación continuos. Esto garantizará que todos los empleados puedan aprovechar al máximo las funcionalidades ofrecidas y que la solución siga siendo eficaz y adaptada a las necesidades organizacionales a medida que estas evolucionan.

En resumen, el proyecto SCESO-Soft ha establecido una base sólida para la gestión avanzada del tiempo laboral en OptiSalud, mejorando significativamente la eficiencia, seguridad y precisión de los registros laborales. La implementación del sistema no solo ha abordado las ineficiencias previas, sino que también ha puesto la infraestructura necesaria para futuras mejoras y expansiones. Esto posiciona a OptiSalud favorablemente para futuros desafíos y oportunidades en la gestión de recursos humanos.

Lista de referencias

Canós, J. H., Letelier, P., & Penadés, M. C. (2003). Metodologías ágiles en el desarrollo de software. Universidad Politécnica de Valencia, Valencia, 1-8. https://www.academia.edu/download/34546906/XP_Agil.pdf

Duarte, A. O., & Rojas, M. (2008). Las metodologías de desarrollo ágil como una oportunidad para la ingeniería del software educativo. Revista Avances en Sistemas e Informática, 5(2), 159-171. https://www.redalyc.org/pdf/1331/133115027022.pdf

Euroinnova Business School. (2023, January 9). ¿Qué son las aplicaciones de escritorio? Euroinnova Business School. https://www.euroinnova.edu.es/blog/aplicaciones-de-escritorio

Escofet, C. M. (2002). El lenguaje SQL. UOC, la universidad virtual. https://pdfcursos.com/pdf/0018-lenguaje-sql.pdf

Hidglobal.com. Recuperado el 6 de mayo de 2024, de https://www.hidglobal.com/products/4500-fingerprint-reader

Ivan Bravo Mendoza. (2005). Comandos y métodos acerca Digital Persona 4500. https://github.com/IvanBM1/DigitalPersona/blob/master/DigitalPersona/LIB%20-%20SDK%20Digital%20Persona/Samples/Visual%20Studio%202005/CSharp/Enrollment/CaptureForm.cs

Microsoft. (2023). ¿Qué es el IDE de Visual Studio?. https://learn.microsoft.com/es-es/visualstudio/get-started/visual-studio-ide?view=vs-2022

Microsoft. (2022). WPF Samples Repository. https://github.com/microsoft/WPF-Samples

.NET Platform. (2023) Windows Presentation Foundation (WPF). https://github.com/dotnet/wpf

Llerena Ocaña, L. A., Viscaino Naranjo, F. A., & Culque Toapanta, W. V. (2022). Desarrollo de software con Net Core. Revista Universidad y Sociedad, 14(2), 85-89. http://scielo.sld.cu/scielo.php?pid=S2218-36202022000200085&script=sci_arttext

Luis del Valle Hernández. (2016). ¿Qué es WPF ? un repaso por sus características. Programarfacil Arduino y Home Assistant. https://programarfacil.com/blog/programacion-net-blog/que-es-wpf/

Ojeda, J. R. (2019). DigitalPersona: Comandos y métodos acerca de digitalPersona 4500 Reader. https://stackoverflow.com/questions/45493824/fingerprint-scanning-with-digitalpersona-4500-reader-how-to-get-captured-image

Optisalud SAS. (2023). Brochure Corporativo. https://optisalud.com/wp-content/uploads/2023/11/Brochure-Corporativo_20231024_130812_0000.pdf

Velasco, M. V. E., Villacis, J. A. N., Chávez, P. R. S., & Cuchipe, W. C. C. (2021). Revisión sistemática de la metodología SCRUM para el desarrollo de Software. Dominio de las Ciencias, 7(4), 54. https://dialnet.unirioja.es/servlet/articulo?codigo=8384028

Anexos

Encuesta de Viabilidad del Sistema de Registro Automático en OptiSalud.

Esta encuesta tiene como objetivo evaluar la viabilidad de implementar un sistema automático de registro de entrada y salida en OptiSalud. Buscamos entender las percepciones y expectativas de los empleados sobre la transición a una solución digital que promete mejorar la precisión y eficiencia de los procesos actuales de registro de tiempo. Tu opinión es fundamental para garantizar que el sistema propuesto cumpla con las necesidades del personal y contribuya a una gestión laboral más efectiva.

1. Nombres y apellidos
2. Documento
3. ¿Cómo calificaría la eficiencia del proceso actual de registrar sus horas de trabajo de manera manual?

Muy ineficiente

Ineficiente

Neutral

Eficiente

Muy eficiente

4. ¿Con qué frecuencia encuentra errores o discrepancias en los registros manuales de tiempo laboral?

Muy Frecuentemente

Frecuentemente

Ocasionalmente

Raramente

Nunca

5. ¿Cree que los errores en el registro manual de horas afectan la exactitud del conteo de sus horas laborales?

Si

No

6. ¿Conoce sistemas automatizados para el registro de llegada y salida de horarios laborales?

Si

No

7. ¿Estaría de acuerdo con la implementación de un sistema automatizado para el registro de horas laborales?

Si

No

Indeciso/a

Enlace de encuesta.