UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE
Matriz – Sangolquí

PROYECTO UNIDAD 1

Sistema de Registro Forestal

Documentación Completa

Integrantes:
Antonio Adrián Revilla Anchapaxi
Gabriel Omar Murillo Medina
Alexander Freddy Ñacato Peña
Pablo Esteban Zurita

Sangolquí, 1 de Junio del 2025

Índice general

Capítulo 1

Documentación de análisis del sistema

1.1. Introducción

El manejo adecuado y sostenible de los recursos forestales es fundamental para la conservación del medio ambiente y la biodiversidad. En este contexto, el desarrollo de un Sistema de Registro Forestal se vuelve una herramienta esencial para gestionar de manera eficiente la información relacionada con las zonas forestales, las especies de árboles presentes y las actividades de conservación realizadas. Este sistema, basado en una arquitectura N-Capas y desarrollado con tecnologías Java EE y MySQL, permitirá a las organizaciones responsables del cuidado ambiental registrar, consultar y actualizar datos críticos, facilitando la toma de decisiones y el monitoreo continuo de los recursos forestales. Así, se contribuye a la protección ambiental y al cumplimiento de políticas y normativas relacionadas con la gestión forestal.

1.2. Objetivos y alcance

Objetivo general

Analizar y definir los componentes fundamentales del Sistema de Registro Forestal, orientado a la gestión integral de información sobre recursos forestales, garantizando la seguridad, escalabilidad e interoperabilidad mediante una arquitectura N-Capas, integración de servicios web y control de acceso por roles.

Objetivos específicos

- Identificar los requerimientos funcionales y no funcionales del sistema de registro forestal.
- Establecer una arquitectura lógica del sistema basada en N Capas que facilite el mantenimiento, la seguridad y la escalabilidad.
- Definir los mecanismos de seguridad necesarios para controlar el acceso al sistema mediante autenticación y autorización por roles de usuario.
- Determinar las interfaces necesarias para la interoperabilidad con sistemas externos mediante servicios web SOAP.
- Delimitar el alcance funcional del sistema en cuanto a los módulos de gestión de zonas forestales, especies arbóreas y actividades de conservación.

Alcance

El Sistema de Registro Forestal permitirá gestionar de manera centralizada la información relacionada con zonas forestales, especies arbóreas y actividades de conservación, facilitando el registro, consulta y actualización de datos a través de una interfaz web accesible para distintos tipos de usuarios. El sistema incorporará mecanismos de seguridad basados en roles, que permitirán controlar los permisos de acceso según el perfil del usuario, y expondrá servicios web mediante el protocolo SOAP para posibilitar la interoperabilidad con otros sistemas que requieran intercambiar información estructurada.

1.3. Descripción general

El Sistema de Registro Forestal es una aplicación desarrollada con el objetivo de gestionar de manera eficiente la información relacionada con zonas forestales, especies de árboles y actividades de conservación. Este sistema permite realizar operaciones completas de creación, consulta, actualización y eliminación (CRUD) sobre estas tres entidades, facilitando el seguimiento y control de los recursos forestales por parte de los usuarios encargados de su administración.

La implementación se realizó utilizando una arquitectura N-Capas dividida en seis niveles funcionales: Presentación, Controller, Lógica de negocio, Service, DAO (acceso a datos), Modelo y Seguridad. Esta estructura garantiza una separación clara de responsabilidades, facilitando el mantenimiento, la escalabilidad y la seguridad del sistema. En particular, la capa de seguridad se encarga de gestionar roles y permisos, controlando el acceso y las acciones permitidas para cada usuario según su perfil.

Para facilitar la interoperabilidad con otros sistemas, se desarrollaron servicios web basados en el protocolo SOAP, que permiten el intercambio seguro y estructurado de información.

Para el desarrollo del frontend se emplearon tecnologías como HTML, CSS y la biblioteca Bootstrap, lo que permitió una interfaz gráfica amigable y responsiva. El entorno de desarrollo utilizado incluyó el IDE NetBeans y el servidor de bases de datos MySQL gestionado a través de XAMPP, herramientas que facilitaron la implementación y prueba del sistema en un entorno local.

1.4. Análisis del Dominio del Problema

El dominio del problema aborda la gestión de información ambiental relacionada con zonas forestales, especies de árboles y actividades de conservación, con el fin de apoyar procesos de monitoreo, protección y recuperación de áreas naturales. Actualmente, muchas organizaciones ambientales o unidades administrativas no cuentan con una herramienta digital adecuada para gestionar esta información, lo que limita la eficiencia operativa, la toma de decisiones basadas en datos y el cumplimiento de normativas ambientales.

Entidades Principales

El sistema está centrado en seis entidades principales que permiten gestionar integralmente la información ambiental:

- **ZONAS:** Representa zonas geográficas donde se realizan actividades de conservación o se registran especies de árboles. Cada zona contiene información relevante como nombre, tipo de bosque (Seco, Húmedo Tropical, Montano, Manglar, Otro), área en hectáreas y estado activo. Las zonas definen el ámbito espacial para el resto de las entidades.

- **ESPECIES DE ÁRBOLES:** Contienen información botánica y ambiental sobre las especies registradas dentro de una zona, incluyendo nombre común, nombre científico, estado de conservación (que clasifica su nivel de riesgo ambiental), y la relación directa con la zona correspondiente. Esta entidad es fundamental para el seguimiento de la biodiversidad y la evaluación del estado ambiental.
- **ACTIVIDADES DE CONSERVACIÓN:** Reflejan las acciones de manejo, conservación o intervención realizadas dentro de una zona forestal. Cada actividad almacena datos como nombre de la actividad, fecha, responsable, tipo de actividad (reforestación, monitoreo, educación, limpieza), descripción, zona asociada y estado activo, permitiendo el control y documentación de las acciones de preservación ambiental.
- **ESTADO DE CONSERVACIÓN:** Define las categorías de riesgo ambiental para las especies (por ejemplo, Extinto, En Peligro Crítico, Vulnerable), permitiendo clasificar y priorizar las acciones de conservación.
- **TIPO DE ACTIVIDAD:** Clasifica las actividades de conservación según su naturaleza, facilitando la organización y análisis de las intervenciones realizadas.
- **USUARIOS:** Gestiona los usuarios que interactúan con el sistema, almacenando información de acceso y roles (administrador, usuario, invitado), lo que permite controlar permisos y garantizar la seguridad de la información.

Actores del Sistema

Los actores del sistema son los usuarios que interactúan con la aplicación para registrar, consultar y gestionar la información relacionada con zonas forestales, especies y actividades de conservación. Dependiendo del rol asignado (administrador, usuario o invitado), los niveles de acceso y modificación varían para garantizar la seguridad y la integridad de los datos.

1. **Funciones del Administrador:**
   - Registrar, consultar, actualizar y eliminar información relacionada con zonas forestales, especies de árboles y actividades de conservación.
   - Realizar búsquedas y filtrados específicos para facilitar el acceso rápido a la información relevante.
   - Visualizar detalles completos de cada registro para apoyar la toma de decisiones.
2. **Funciones del Usuario Estándar:**
   - Consultar y visualizar información de zonas forestales, especies de árboles y actividades de conservación.
   - No tiene permisos para crear, modificar o eliminar registros.

Procesos del Negocio

Los procesos de negocio del sistema representan las actividades funcionales que permite realizar la aplicación. A continuación se describen:

1. **Registrar Zona Forestal**
   - **Descripción:** Permite ingresar una nueva zona forestal con sus datos básicos.
   - **Entrada:** `nombre` (varchar), `tipo_bosque` (varchar), `area_ha` (decimal).

- **Salida:** Zona almacenada en la tabla `zones` con `activo = 1` y marcas de tiempo automáticas.

2. **Registrar Especie de Árbol**
   - **Descripción:** Permite ingresar una especie de árbol identificada en una zona y asociarla a su estado de conservación.
   - **Entrada:** `nombre_comun` (varchar), `nombre_cientifico` (varchar), `estado_conservacion_id` (int), `zona_id` (int).
   - **Salida:** Especie almacenada en la tabla `tree_species` con `activo = 1` y marcas de tiempo automáticas.
3. **Registrar Actividad de Conservación**
   - **Descripción:** Permite documentar actividades realizadas en una zona forestal.
   - **Entrada:** `nombre_actividad` (varchar), `fecha_actividad` (date), `responsable` (varchar), `tipo_actividad_id` (int), `descripcion` (text), `zona_id` (int).
   - **Salida:** Actividad registrada en la tabla `conservation_activities` con `activo = 1` y marcas de tiempo automáticas.
4. **Editar Zona, Especie o Actividad**
   - **Descripción:** Permite modificar los datos previamente registrados en las tablas `zones`, `tree_species` o `conservation_activities`.
   - **Entrada:** `id` del registro seleccionado, nuevos valores para sus campos correspondientes según la tabla.
   - **Salida:** Registro actualizado con fecha de `actualizado_en` modificada automáticamente.
5. **Borrado Lógico de Registros**
   - **Descripción:** Permite desactivar un registro sin eliminarlo físicamente de la base de datos (ya sea zona, especie o actividad).
   - **Entrada:** `id` del registro a desactivar.
   - **Salida:** Campo `activo` actualizado a `0` para marcar el registro como inactivo.
6. **Consultar Registros**
   - **Descripción:** Permite buscar y filtrar información de zonas, especies y actividades según distintos criterios.
   - **Entrada:** Parámetros de búsqueda que pueden incluir `nombre`, `fecha_actividad`, `zona_id`, `estado_conservacion_id`, etc.
   - **Salida:** Listado de registros que cumplen los criterios, con campos completos y el estado de `activo`.
7. **Visualizar Información de Registros**
   - **Descripción:** Permite al usuario ver en detalle la información almacenada en las tablas `zones`, `tree_species` o `conservation_activities`.
   - **Entrada:** `id` del registro seleccionado.
   - **Salida:** Datos completos del registro consultado, incluyendo todos los campos relevantes.

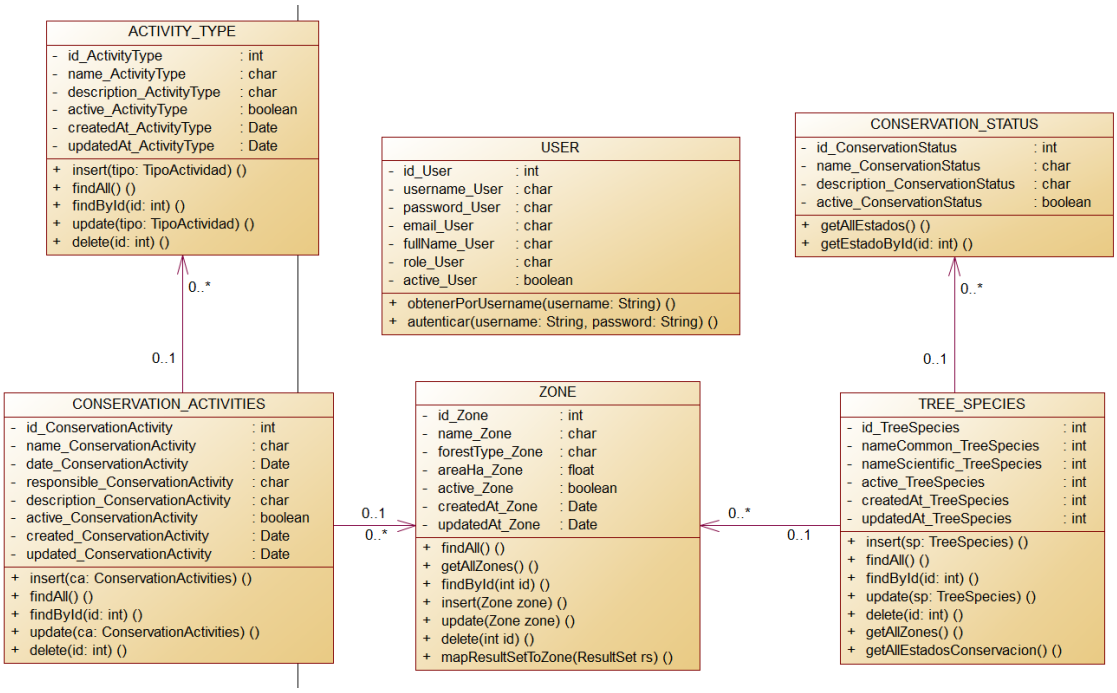


Figura 1.1: Modelo Conceptual del Sistema

1.5. Modelo Conceptual del Sistema

1.6. Posibles Extensiones Futuras

El Sistema de Registro Forestal ha sido diseñado con una arquitectura modular que facilita su expansión y adaptación a nuevas necesidades. A continuación, se describen algunas posibles extensiones que podrían implementarse en el futuro:

- **Módulo de Reportes Avanzados:** Este módulo permitiría la generación de reportes personalizados y dinámicos sobre zonas forestales, especies de árboles y actividades de conservación. Los usuarios podrían exportar estos reportes en diversos formatos, como PDF, Excel y CSV, y se incluirían gráficos y visualizaciones para facilitar el análisis de la información.

- **Funcionalidad de Alertas y Notificaciones:** Este módulo permitiría a los usuarios configurar alertas automáticas para eventos importantes, como la detección de incendios o cambios significativos en el estado de conservación de una especie. El sistema podría enviar notificaciones por correo electrónico o a través de la interfaz de la aplicación, manteniendo a los usuarios informados sobre situaciones críticas que requieren atención.

- **Integración con Sistemas de Información Geográfica (SIG):** Agregar la capacidad de integrar mapas interactivos y datos espaciales avanzados para la visualización detallada de las zonas forestales y la ubicación de especies o actividades. Esto permitiría análisis geoespaciales más precisos y soporte para la toma de decisiones basadas en localización.

- **Incorporación de Inteligencia Artificial para Análisis Predictivo:** Aplicar técnicas de machine learning para predecir riesgos ambientales, evaluar tendencias de conservación y optimizar la planificación de actividades, mejorando la capacidad de prevención y gestión proactiva.

- **Implementación de un Módulo Móvil:** Desarrollar una aplicación móvil complementaria que permita a los usuarios registrar y consultar información desde dispositivos móviles en campo, incluso con funcionalidad offline y sincronización posterior, para mejorar la captura de datos en zonas remotas.

Capítulo 2

Documentación de Requisitos Funcionales y No Funcionales (SRS)

2.1. Introducción (SRS)

2.1.1. Propósito (SRS)

Este documento tiene como objetivo definir los requisitos funcionales y no funcionales del Sistema de Registro Forestal, una aplicación destinada a registrar, consultar y actualizar información relativa a zonas forestales, especies de árboles y actividades de conservación. Será desarrollado en Java EE utilizando MySQL como sistema de gestión de base de datos, implementando una arquitectura N-Capas para asegurar la separación de responsabilidades y una alta mantenibilidad.

El documento está dirigido a los desarrolladores del sistema, stakeholders y usuarios finales del producto.

2.1.2. Ámbito del sistema (SRS)

El sistema, denominado **Sistema de Registro Forestal**, está diseñado para facilitar la gestión integral de información relacionada con zonas forestales, especies arbóreas y actividades de conservación.
Permitirá a los usuarios autenticados y autorizados realizar las siguientes acciones, de acuerdo con sus permisos y roles:

- Registrar nuevas zonas forestales.
- Registrar especies de árboles.
- Registrar y consultar actividades de conservación.
- Consultar y actualizar la información almacenada.

Este sistema está destinado a ser utilizado por entidades gubernamentales, ONGs, y organizaciones encargadas de la gestión ambiental y forestal.

Los principales beneficios que aportará el sistema son:

- Centralización de la información forestal.
- Facilidad de acceso, consulta y registro de datos por medio de una interfaz web intuitiva.
- Seguimiento de actividades de conservación.
- Mejora en la toma de decisiones relacionadas con la gestión forestal.

2.1.3. Definiciones, Acrónimos y Abreviaturas (SRS)

- **Java EE:** Java Platform, Enterprise Edition.
- **MySQL:** Sistema de gestión de bases de datos relacional.
- **N-Capas:** Arquitectura de software que divide la aplicación en capas lógicas (presentación, lógica de negocio, acceso a datos).
- **SRS:** Software Requirements Specification (Especificación de Requisitos del Software).
- **GUI:** Interfaz gráfica de usuario.
- **HTTPS:** Protocolo seguro de transferencia de hipertexto.
- **MVC:** Modelo-Vista-Controlador, patrón de arquitectura de software.
- **DAO:** Data Access Object, patrón de diseño para acceso a datos.
- **SOAP:** Simple Object Access Protocol, protocolo para intercambio estructurado de información en servicios web.
- **WSDL:** Web Services Description Language, lenguaje basado en XML para describir servicios web.
- **JAX-WS:** Java API for XML Web Services, API de Java para crear servicios web basados en SOAP.
- **SHA-256:** Algoritmo criptográfico de hash seguro de 256 bits.
- **HTTP Session:** Sesión de comunicación entre cliente y servidor en aplicaciones web, usada para mantener la autenticación del usuario.
- **Filtro Servlet:** Componente que intercepta peticiones en aplicaciones Java EE para aplicar lógica de seguridad o procesamiento adicional.
- **Autenticación:** Proceso de verificar la identidad de un usuario.
- **Autorización:** Proceso de controlar el acceso a recursos según permisos del usuario.
- **Rol:** Conjunto de permisos asignados a un usuario para definir su nivel de acceso dentro del sistema (por ejemplo, administrador, usuario).

2.1.4. Referencias (SRS)

- IEEE Std 830-1998: Especificaciones de los requisitos del software.
- Java EE 8 API Documentation.
- MySQL 8.0 Reference Manual.
- Documentación de patrones de diseño MVC y DAO.
- W3C. (2007). *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition).*
- Oracle. (n.d.). *JAX-WS: Java API for XML Web Services*. Documentación oficial de Oracle.

2.1.5. Visión general del documento (SRS)

El resto de este documento está organizado de la siguiente manera:

- **Sección 2 - Descripción general:** Proporciona una visión del sistema, incluyendo sus principales funcionalidades, tipos de usuarios (roles), restricciones y dependencias tecnológicas como el uso de servicios SOAP.
- **Sección 3 - Requisitos específicos:** Detalla los requisitos funcionales y no funcionales, incluyendo aspectos de autenticación, control de acceso por roles, seguridad, y las interfaces externas del sistema.
- **Sección 4 - Apéndices:** Contiene información complementaria relevante, como glosario de términos, diagramas arquitectónicos y referencias técnicas.

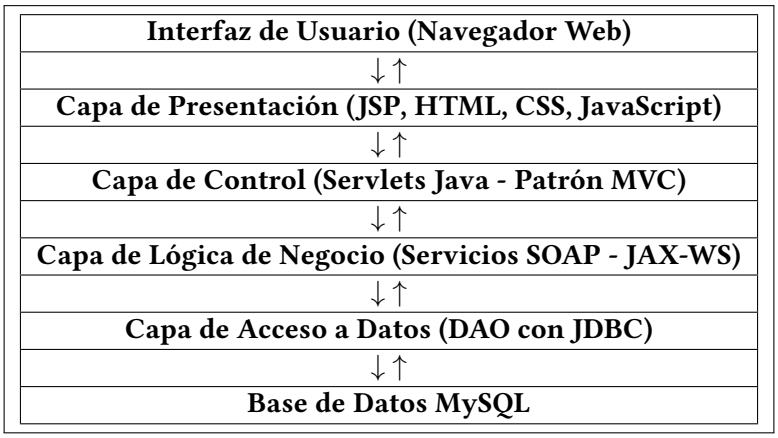
2.2. Descripción general del Sistema (SRS)

2.2.1. Perspectiva del producto (SRS)

El Sistema de Registro Forestal es una aplicación web distribuida basada en Java EE que sigue una arquitectura en N-Capas con la siguiente estructura:

- **Capa de Presentación:** Proporciona la interfaz con la que los usuarios interactúan directamente. Está implementada mediante páginas JSP, HTML estático, CSS y JavaScript (incluyendo modales para formularios). Esta capa maneja la presentación de datos, la captura de entradas y el control de acceso a través de menús dinámicos según roles.
- **Capa de Control:** Consiste en controladores Java (Servlets) que reciben y procesan las solicitudes de la capa de presentación, validan entradas, gestionan sesiones y controlan el flujo hacia la lógica de negocio, respetando roles y permisos de usuario. Esta capa implementa el patrón MVC para separar presentación y lógica.
- **Capa de Lógica de Negocio:** Implementada mediante servicios web SOAP desarrollados con JAX-WS, que exponen operaciones para la gestión del dominio, tales como manejo de especies, zonas, actividades y usuarios. Esta capa facilita la interoperabilidad y la separación clara entre cliente y servidor.
- **Capa de Acceso a Datos:** Gestiona la persistencia mediante objetos DAO que utilizan JDBC para comunicarse con la base de datos MySQL. Esta capa abstrae las operaciones CRUD y garantiza la integridad y consistencia de los datos.
- **Capa de Dominio / Entidades:** Define las clases modelo que representan las entidades del negocio, tales como especies de árboles, zonas, actividades, tipos y usuarios, que sirven como medio de transferencia de información entre capas.
- **Capa de Seguridad:** Es una capa transversal que asegura la confidencialidad, integridad y disponibilidad del sistema mediante autenticación, autorización basada en roles, cifrado de contraseñas (SHA-256) y protección de recursos mediante filtros y gestión de sesiones HTTP.

El sistema está diseñado para operar de manera autónoma, aunque cuenta con la capacidad para integrarse con otros sistemas externos mediante servicios web en caso de ser necesario, facilitando la interoperabilidad para el intercambio de información ambiental o geográfica.



Capa de Seguridad: transversal a todas las capas anteriores, controla autenticación, autorización y protección de datos.

Figura 2.1: Arquitectura N-Capas con capa de seguridad transversal

2.2.2. Funciones del producto (SRS)

El Sistema de Registro Forestal proporcionará las siguientes funcionalidades principales:

- **Registro y gestión de zonas forestales:** Creación, consulta, actualización y eliminación de zonas con sus características.
- **Gestión de especies arbóreas:** Registro y mantenimiento del catálogo de especies y su asociación con zonas.
- **Registro de actividades de conservación:** Documentación de las intervenciones y monitoreos realizados.
- **Búsqueda avanzada:** Filtros para facilitar la consulta por diferentes criterios.
- **Generación de reportes:** Exportación de información en formatos PDF y Excel.
- **Seguridad básica:** Control de acceso mediante autenticación y autorización para proteger la información según el rol del usuario.
- **Servicios SOAP:** Exposición de funciones clave a través de servicios SOAP para permitir la integración con otros sistemas.

2.2.3. Características de los usuarios (SRS)

El sistema contempla dos perfiles principales:

- **Administrador:** Tiene permisos para crear, leer, actualizar y eliminar toda la información del sistema. Responsable de gestionar y mantener los datos.
- **Consultor:** Solo puede visualizar la información sin permisos para modificar o eliminar datos.

2.2.4. Restricciones (SRS)

Las siguientes restricciones afectan el desarrollo del sistema:

- **Tecnológicas:**
  - El sistema debe ejecutarse en un servidor compatible con Java EE (GlassFish, WildFly).
  - La base de datos debe ser MySQL 8.0 o superior.
  - Los servicios de la capa de lógica de negocio se implementan como servicios web SOAP para ser consumidos por clientes externos.
  - Debe ser accesible a través de los navegadores web modernos (Chrome, Firefox, Edge).
- **Operativas:**
  - La interfaz debe estar en español.
  - El sistema debe poder operar en redes con ancho de banda limitado.
- **Seguridad:**
  - La seguridad es adecuada para protección básica, sin mecanismos avanzados ni robustos.
  - Debe cumplir con estándares básicos de protección de datos personales.
  - Las comunicaciones externas deben ser cifradas mediante HTTPS.
  - El sistema incluye autenticación y control de acceso por roles simples.

2.2.5. Suposiciones y dependencias (SRS)

- **Suposiciones:**
  - Los usuarios dispondrán de acceso a internet.
  - Los usuarios tendrán conocimientos básicos de informática.
  - La información geográfica estará disponible en formatos estándar.
  - El sistema contará con una configuración básica de seguridad para autenticación y autorización por roles.
- **Dependencias:**
  - Disponibilidad de un servidor con Java EE configurado correctamente.
  - Disponibilidad de un servidor MySQL accesible desde el servidor de aplicaciones.
  - Conexión a internet estable para el acceso remoto.
  - Configuración adecuada del entorno para soportar servicios web SOAP y mecanismos de seguridad básicos.

2.2.6. Requisitos futuros (SRS)

Los siguientes requisitos podrían considerarse para futuras versiones del sistema:

- Sistema de reportes avanzados: generación de informes con gráficos, filtros personalizables y exportación a formatos como PDF o Excel.
- Alertas y notificaciones automáticas: envío de notificaciones por correo electrónico o mensajes internos ante eventos relevantes, como registros críticos o alertas de conservación.

- Integración con Sistemas de Información Geográfica (SIG): visualización de zonas forestales, actividades de conservación y ubicaciones de especies en mapas interactivos mediante librerías como OpenLayers o Leaflet.
- Módulo de auditoría y logs detallados: registro detallado de acciones realizadas por usuarios, incluyendo quién, cuándo y desde dónde (IP), para mejorar la trazabilidad y seguridad del sistema.
- Aplicación móvil para trabajo de campo, facilitando la captura y consulta de datos en terreno.
- Funcionalidades de inteligencia artificial para identificación automática de especies arbóreas.

2.3. Requisitos específicos (SRS)

2.3.1. Interfaces externas (SRS)

Interfaz de usuario (SRS)

- El sistema proporcionará una interfaz web responsiva, compatible con resoluciones de pantalla de 1024x768 píxeles y superiores.
- La interfaz seguirá principios de diseño adaptativo para funcionar correctamente en diferentes dispositivos.
- Se utilizarán componentes estándar de HTML5, CSS3 y JavaScript.
- Los formularios mostrarán validaciones instantáneas y mensajes de error claros.

Interfaz de hardware (SRS)

- El sistema no requiere hardware especializado para su funcionamiento básico.
- Para funcionalidades de campo, se podría requerir integración con GPS para la versión móvil futura.

Interfaz de software (SRS)

- El sistema debe ser compatible con los navegadores web: Chrome (v80+), Firefox (v75+), Edge (v80+).
- El servidor requiere Java EE 8 o superior y un servidor de aplicaciones compatible (GlassFish, WildFly).
- La base de datos requiere MySQL 8.0 o superior.
- Los servicios de la capa de lógica de negocio se implementan como servicios web SOAP para consumo por clientes externos.
- La capa de seguridad incluye autenticación y autorización basadas en roles simples para controlar el acceso a funcionalidades.

Interfaz de comunicaciones (SRS)

- Se utilizará el protocolo HTTP/HTTPS para todas las comunicaciones, garantizando cifrado en las conexiones externas.
- Las transferencias de datos para la interfaz web se realizarán preferentemente en formato JSON.
- Los servicios web SOAP utilizarán mensajes XML para la comunicación.

- Se implementará un mecanismo de manejo de sesiones para mantener la autenticación y autorización de los usuarios en la interfaz web.

2.3.2. Funciones (SRS)

Identificación	FR-01
Nombre	Gestión de usuarios
Descripción	El sistema debe permitir gestionar usuarios con atributos: nombre de usuario, contraseña cifrada, email, nombre completo y rol (admin, usuario, invitado). Debe incluir creación, edición, eliminación lógica y listado de usuarios activos. La gestión solo estará accesible para usuarios con rol administrador.

Cuadro 2.1: Requisito Funcional FR-01

Identificación	FR-02
Nombre	Gestión de zonas forestales
Descripción	El sistema debe permitir la gestión de zonas con atributos: nombre, tipo de bosque, área en hectáreas y estado activo. Se deben permitir crear, modificar, listar y eliminar zonas.

Cuadro 2.2: Requisito Funcional FR-02

Identificación	FR-03
Nombre	Gestión de especies de árboles
Descripción	El sistema debe permitir la gestión de especies de árboles con atributos: nombre común, nombre científico, estado de conservación y zona asociada. Debe soportar creación, edición, listado y eliminación lógica de especies.

Cuadro 2.3: Requisito Funcional FR-03

Identificación	FR-04
Nombre	Gestión de estados de conservación
Descripción	El sistema debe mantener un catálogo de estados de conservación con nombre, descripción y estado activo. Estos estados se usarán para clasificar las especies de árboles.

Cuadro 2.4: Requisito Funcional FR-04

Identificación	FR-05
Nombre	Gestión de tipos de actividad
Descripción	El sistema debe mantener un catálogo de tipos de actividades de conservación con nombre, descripción y estado activo. Estas actividades se asociarán a las actividades registradas.

Cuadro 2.5: Requisito Funcional FR-05

Identificación	FR-06
Nombre	Gestión de actividades de conservación
Descripción	El sistema debe permitir la gestión de actividades de conservación con atributos: nombre de actividad, fecha, responsable, tipo de actividad, descripción, zona asociada, y estado activo. Se deben poder crear, listar, editar y eliminar actividades.

Cuadro 2.6: Requisito Funcional FR-06

Identificación	FR-07
Nombre	Exposición de servicios SOAP
Descripción	El sistema debe implementar servicios SOAP que expongan las operaciones CRUD de las entidades principales (usuarios, zonas, especies, actividades, estados y tipos). Estos servicios serán consumidos de forma externa por aplicaciones desarrolladas en Python para interoperabilidad y automatización.

Cuadro 2.7: Requisito Funcional FR-07

Identificación	FR-08
Nombre	Autenticación y control de acceso por roles
Descripción	El sistema debe implementar un mecanismo de autenticación de usuarios que permita identificar a cada usuario mediante credenciales seguras. Además, debe controlar el acceso a las funcionalidades y datos del sistema según el rol asignado al usuario (por ejemplo, administrador, usuario, invitado), restringiendo o habilitando operaciones específicas.

Cuadro 2.8: Requisito Funcional FR-08

2.3.3. Requisitos de rendimiento (SRS)

Identificación	PR-01
Nombre	Tiempo de respuesta en consultas
Descripción	Las respuestas a consultas con hasta 1,000 registros visibles deben ser menores a 3 segundos. Para consultas con múltiples filtros o más de 1,000 registros, el tiempo de respuesta no debe superar los 5 segundos.

Cuadro 2.9: Requisitos De Rendimiento PR-01

Identificación	PR-02
Nombre	Capacidad de usuarios concurrentes
Descripción	El sistema debe soportar al menos 50 usuarios concurrentes realizando operaciones (consultas, registros, actualizaciones) sin degradación notable del servicio.

Cuadro 2.10: Requisitos De Rendimiento PR-02

Identificación	PR-03
Nombre	Tiempo de carga inicial
Descripción	La carga inicial de la aplicación debe completarse en menos de 5 segundos bajo una conexión estándar (ej. 10 Mbps).

Cuadro 2.11: Requisitos De Rendimiento PR-03

Identificación	PR-04
Nombre	Capacidad de almacenamiento
Descripción	El sistema debe almacenar al menos 10,000 registros de zonas forestales. Debe manejar un catálogo de al menos 5,000 especies arbóreas.

Cuadro 2.12: Requisitos De Rendimiento PR-04

Identificación	PR-05
Nombre	Optimización de consultas
Descripción	La base de datos y las consultas deben estar optimizadas para manejar eficientemente búsquedas y filtros complejos con múltiples criterios sin afectar el rendimiento.

Cuadro 2.13: Requisitos De Rendimiento PR-05

2.3.4. Restricciones de diseño (SRS)

- El desarrollo debe seguir el patrón de arquitectura Modelo-Vista-Controlador (MVC).
- Se debe implementar el patrón Data Access Object (DAO) para la gestión del acceso a datos.
- El sistema debe utilizar un mecanismo de inyección de dependencias para facilitar la gestión de componentes y promover la modularidad.
- La capa de presentación debe estar claramente separada de la lógica de negocio, garantizando la independencia entre ambas.
- La persistencia de datos se realizará mediante Java Persistence API (JPA).

- El diseño de la base de datos debe cumplir, al menos, con la tercera forma normal para garantizar la integridad y evitar redundancias.
- La capa de seguridad debe garantizar la autenticación y autorización de usuarios, controlando el acceso a las funcionalidades según los roles asignados.
- La lógica de negocio debe exponerse a través de servicios SOAP que serán consumidos por clientes externos (por ejemplo, aplicaciones en Python).

2.3.5. Atributos del sistema (SRS)

Identificación	RNF-01
Nombre	Comunicación segura
Descripción	El sistema debe usar HTTPS para cifrar las comunicaciones.
Factor de Calidad	Seguridad

Cuadro 2.14: Requisito No Funcional RNF-01

Identificación	RNF-02
Nombre	Protección de datos
Descripción	Se deben prevenir ataques como inyección SQL y XSS.
Factor de Calidad	Seguridad

Cuadro 2.15: Requisito No Funcional RNF-02

Identificación	RNF-03
Nombre	Control de acceso
Descripción	Solo usuarios con rol autorizado pueden acceder a cada funcionalidad.
Factor de Calidad	Seguridad

Cuadro 2.16: Requisito No Funcional RNF-03

Identificación	RNF-04
Nombre	Interfaz intuitiva
Descripción	La interfaz debe ser fácil de usar, con formularios claros y mensajes útiles.
Factor de Calidad	Usabilidad

Cuadro 2.17: Requisito No Funcional RNF-04

Identificación	RNF-05
Nombre	Navegadores compatibles
Descripción	Debe funcionar correctamente en Chrome, Firefox y Edge.
Factor de Calidad	Usabilidad

Cuadro 2.18: Requisito No Funcional RNF-05

Identificación	RNF-06
Nombre	Tiempo de respuesta
Descripción	Las consultas con hasta 1,000 registros deben responder en menos de 3 segundos.
Factor de Calidad	Rendimiento

Cuadro 2.19: Requisito No Funcional RNF-06

Identificación	RNF-07
Nombre	Capacidad de datos
Descripción	Debe manejar al menos 10,000 zonas y 5,000 especies.
Factor de Calidad	Rendimiento

Cuadro 2.20: Requisito No Funcional RNF-07

Identificación	RNF-08
Nombre	Código mantenible
Descripción	El sistema debe seguir los patrones MVC y DAO con código documentado.
Factor de Calidad	Mantenibilidad

Cuadro 2.21: Requisito No Funcional RNF-08

Identificación	RNF-09
Nombre	Multiplataforma
Descripción	Debe funcionar en navegadores de Windows, Linux y MacOS.
Factor de Calidad	Compatibilidad

Cuadro 2.22: Requisito No Funcional RNF-09

Identificación	RNF-10
Nombre	Servicios interoperables
Descripción	Los servicios SOAP deben poder ser consumidos desde otras aplicaciones.
Factor de Calidad	Interoperabilidad

Cuadro 2.23: Requisito No Funcional RNF-10

2.3.6. Otros requisitos (SRS)

- **Cumplimiento legal:** El sistema debe cumplir con las regulaciones locales de protección de datos.
- **Documentación:** Debe incluirse un manual de usuario completo y documentación técnica.
- **Capacitación:** Se debe proporcionar un plan de capacitación para los diferentes tipos de usuarios.

2.4. Apéndices (SRS)

2.4.1. Apéndice A: Diagrama de entidades (SRS)

Se incluirá un diagrama entidad-relación con las principales entidades:

- Zona Forestal
- Especie Arbórea
- Actividad de Conservación
- Tipo de Actividad
- Estado de Conservación
- Usuario

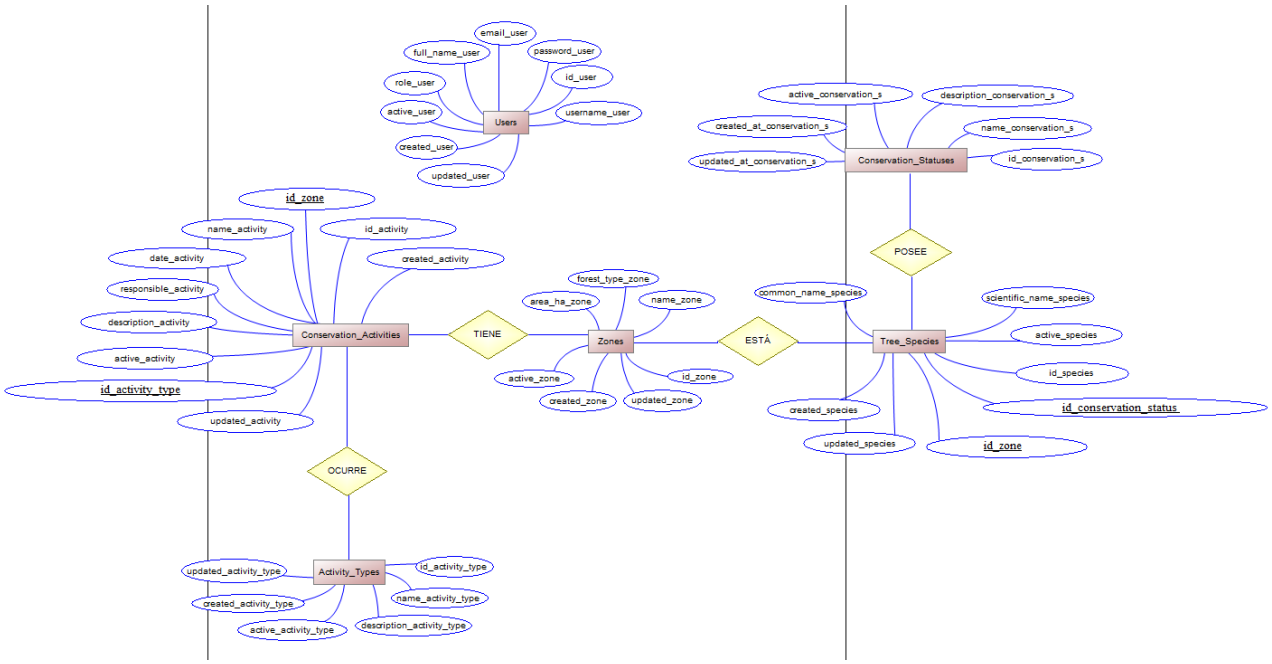


Figura 2.2: Modelo Conceptual del Sistema

2.4.2. Apéndice B: Casos de Uso (SRS)

En este apéndice se presentan los principales casos de uso del sistema de registro forestal, los cuales representan las interacciones entre los actores (usuarios) y el sistema. A continuación, se listan y describen brevemente los casos de uso más relevantes:

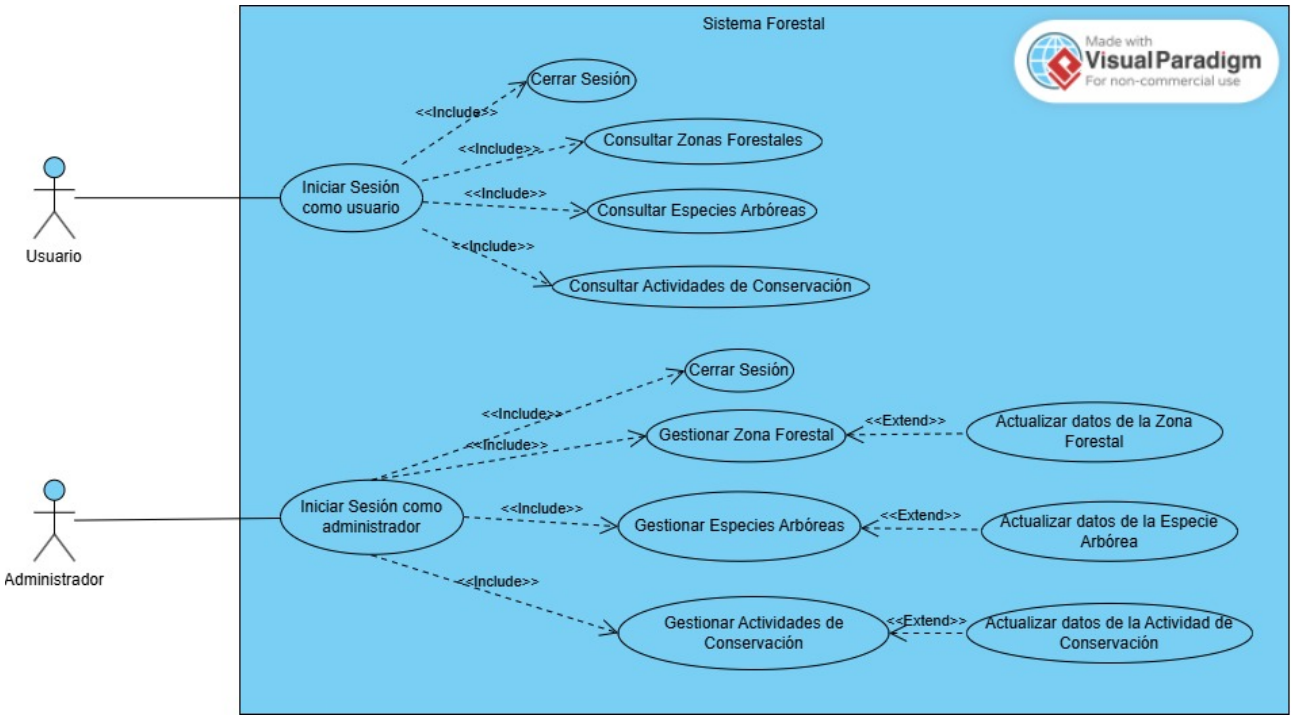


Figura 2.3: Diagrama de Casos de Uso

Capítulo 3

Documentación de Arquitectura del Sistema

3.1. Introducción

El presente documento tiene como objetivo describir la arquitectura del sistema "SistemaRegistroForestal". Este sistema está diseñado para registrar, consultar y actualizar información sobre zonas forestales. La elección de una arquitectura de N-Capas (o multicapa) busca proporcionar una estructura robusta, modular, escalable y mantenible, permitiendo una clara separación de responsabilidades entre los diferentes componentes del sistema. Esta arquitectura facilita el desarrollo colaborativo, la reutilización de código y la adaptación a futuros cambios en los requisitos del negocio.

3.2. Visión General de la Arquitectura y Componentes del Sistema

La comunicación entre capas sigue un flujo jerárquico y unidireccional, comenzando desde la capa de presentación hasta llegar a la base de datos, promoviendo el bajo acoplamiento y la alta cohesión entre componentes.

Principios Arquitectónicos Aplicados:

- **Separación de Responsabilidades:** Cada capa se enfoca en un conjunto específico de tareas, lo cual simplifica el desarrollo y mejora la comprensión del sistema.
- **Reusabilidad:** Los componentes de capas inferiores pueden ser reutilizados por múltiples controladores o servicios, e incluso por otros sistemas.
- **Mantenibilidad:** Al estar las responsabilidades aisladas, los cambios en una capa no afectan significativamente a las demás, lo que reduce riesgos y tiempos de mantenimiento.
- **Escalabilidad:** La arquitectura permite escalar horizontal o verticalmente componentes específicos (como servicios o acceso a datos) sin modificar el resto del sistema.
- **Flexibilidad:** Es posible incorporar nuevas tecnologías en capas individuales (como migrar a frameworks modernos en la presentación o utilizar un ORM en datos) sin afectar drásticamente el sistema completo.
- **Seguridad:** La arquitectura incorpora mecanismos para proteger la integridad, confidencialidad y disponibilidad de la información, mediante controles de acceso, autenticación y autorización distribuidos adecuadamente en las capas correspondientes.

Capas Identificadas en el Sistema:

- **Capa de Presentación (Presentation Layer):** Proporciona la interfaz de usuario y es la encargada de capturar las entradas del usuario y presentar resultados. Implementada mediante páginas JSP y HTML estático, con apoyo de CSS y JavaScript.
  - **Función:** Interfaz con la que el usuario interactúa directamente. Se encarga de mostrar datos y capturar entradas.
  - **Tecnologías:** JSP, HTML, CSS, JavaScript (MODAL)
  - `ConservationActivities.jsp`: Lista de actividades de conservación con control de acceso y modales para formularios.
  - `ConservationActivitiesForm.jsp`: Formulario modal para crear o editar actividades de conservación con validación y selección de tipos y zonas.
  - `index.jsp`: Página principal con secciones informativas, menú incluido, estilos CSS y formulario de contacto.
  - `TipoActividadFrm.jsp`: Modal para crear o editar tipos de actividad con validación y estado activo.
  - `TreeSpecies.jsp`: Lista de especies de árbol con control de acceso para agregar, editar y eliminar solo por administradores, mostrando modales para formularios y tabla dinámica con DataTables.
  - `TreeSpeciesForm.jsp`: Modal con formulario para crear/editar especies, con campos básicos y validación.
  - `Zones.jsp`: Lista de zonas con control de acceso para administración y modales para crear o editar zonas.
  - `ZonesFrm.jsp`: Modal para crear o editar zonas, con formulario validado y selección dinámica de tipos de bosque.
  - `index.html`: Página inicial que redirige automáticamente al login.jsp para iniciar sesión.
  - `indexAdmin.jsp`: Página principal del panel de administración que ofrece un dashboard visual con tarjetas para acceso rápido.
  - `login.jsp`: Página para iniciar sesión con formulario y mensaje de error.
  - `menudinamico.jsp`: Barra de navegación con enlaces principales y estilos Bootstrap.
  - `menuRoles.jsp`: Menú de navegación dinámico según rol (admin o usuario).
- **Capa de Control (Controller Layer):** Actúa como intermediario entre la interfaz de usuario y la lógica de negocio. Procesa solicitudes, válida entradas y dirige el flujo hacia los servicios correspondientes.
  - **Ubicación en el proyecto:** `com.espe.sistemaregistroforestal.controller` (Package Controller)
  - **Función:** Recibe las peticiones de la capa de presentación, gestiona la entrada del usuario y coordina la lógica de negocio.
  - **Tecnologías:** Java (clases controladoras, tipo servlet o clases simples), arquitectura estilo MVC.
  - `ConservationActivitiesController.java`: Controlador Servlet para CRUD de actividades de conservación con control de acceso por rol y manejo de formularios.
  - `LoginServlet.java`: Servlet para autenticación de usuarios con redirección según rol y manejo básico de sesión.

  - `TipoActividadController.java`: Controlador CRUD para tipos de actividad con control de acceso y gestión de formularios.
  - `TreeSpeciesController.java`: Controlador servlet que gestiona CRUD de especies arbóreas con control de acceso por rol y carga de datos relacionados.
  - `ZoneController.java`: Servlet controlador para CRUD de zonas con control de acceso y manejo de tipos de bosque.

- **Capa de Lógica de Negocio (Business Logic Layer / Service Layer):** Esta capa implementa la lógica central del sistema a través de servicios SOAP que exponen operaciones para la gestión del negocio, facilitando la comunicación y orquestación entre clientes y la capa de acceso a datos.

  - **Ubicación en el proyecto:** `com.espe.sistemaregistroforestal.service` (Package Service)
  - **Función:** Proveer métodos web SOAP para ejecutar reglas de negocio y procesos, permitiendo interoperabilidad y separación clara entre cliente y servidor.
  - **Tecnologías:** Java, JAX-WS (Java API for XML Web Services), SOAP, WSDL.
  - `CrudSpeciesPublication.java`: Clase que publica el servicio SOAP para operaciones CRUD sobre especies forestales, exponiendo métodos para crear, consultar, actualizar y eliminar especies, y manteniendo el servicio activo en una URL específica.
  - `CrudSpeciesService.java`: Servicio SOAP que implementa operaciones CRUD para gestionar especies forestales, incluyendo creación, consulta, actualización y eliminación, además de métodos auxiliares para obtener zonas y estados de conservación.
  - `CrudZonesPublication.java`: Clase encargada de publicar el servicio SOAP CrudZonesService en una URL específica para que esté disponible y accesible para clientes.
  - `CrudZonesService.java`: Servicio SOAP que expone operaciones CRUD para la gestión de zonas forestales, incluyendo creación, consulta, actualización y eliminación lógica de zonas. Implementa validaciones de negocio y usa un DAO para acceso a datos, asegurando integridad y registro de actividades mediante logs.
  - `ConservationActivitiesService.java`: clase que gestiona las operaciones CRUD para actividades de conservación, validando datos antes de crear o actualizar, y usando un DAO para acceder a la base de datos.
  - `EstadoConservacionService.java`: Clase que usa TreeSpeciesDAO para obtener la lista de todos los estados de conservación.
  - `TipoActividadService.java`: Servicio que gestiona tipos de actividad: lista, busca, crea, actualiza y elimina con validación básica.
  - `TreeSpeciesService.java`: Servicio que maneja especies de árboles: lista, busca, crea, actualiza y elimina (lógico) con validaciones y control de nombres duplicados.
  - `UsuarioService.java`: Servicio que autentica usuarios verificando su nombre y contraseña hasheada con SHA-256.
  - `ZoneService.java`: Servicio que maneja operaciones CRUD y validaciones básicas para zonas forestales, incluyendo creación, actualización, listado y borrado lógico.

- **Capa de Acceso a Datos (Data Access Layer / DAO):** Gestiona la persistencia de la información en la base de datos mediante operaciones CRUD. Aísla la lógica de datos del resto del sistema utilizando clases DAO y JDBC.

  - **Ubicación en el proyecto:** `com.espe.sistemaregistroforestal.dao` (Package dao)

- **Función:** Interactúa directamente con la base de datos mediante operaciones CRUD.
- **Tecnologías:** Java + MySQL
- `ConnectionBdd.java`: Gestiona la conexión JDBC con la base de datos MySQL.
- `ConservationActivitiesDAO.java`: Acceso a datos para actividades de conservación con métodos CRUD y eliminación lógica.
- `EstadoConservacionDAO.java`: Acceso a datos para estados de conservación con métodos para listar y buscar por ID, manejando conexiones internas y externas.
- `TipoActividadDAO.java`: Acceso a datos para tipos de actividad con CRUD y eliminación lógica, gestionando fechas y estado activo.
- `TreeSpeciesDAO.java`: DAO para operaciones CRUD de especies y obtención de zonas y estados activos.
- `UsuarioDAO.java`: DAO para obtener y autenticar usuarios activos por nombre y contraseña.
- `ZoneDAO.java`: DAO para CRUD y consultas de zonas con eliminación lógica.

- **Capa de Dominio / Entidades (Domain / Entities Layer):** Define las estructuras de datos utilizadas a lo largo del sistema. Son objetos de negocio que representan entidades como zonas, especies o actividades, utilizados como medio de transferencia de información entre capas.

  - **Ubicación en el proyecto:** `com.espe.sistemaregistroforestal.model` (Package Model)
  - **Función:** Define las clases que representan las entidades del negocio. Solo contiene atributos y métodos de acceso (getters y setters).
  - **Tecnologías:** Java (POJOs)
  - `ConservationActivities.java`: Modelo que representa una actividad de conservación con sus atributos básicos.
  - `EstadoConservacion.java`: Modelo para el estado de conservación con nombre, descripción y estado activo.
  - `TipoActividad.java`: Modelo que representa un tipo de actividad con atributos básicos y marcas temporales.
  - `TipoBosque.java`: Enum que define tipos de bosque con nombre desplegable y método para obtener por texto.
  - `TreeSpecies.java`: Modelo de especie de árbol con atributos básicos y metadatos de auditoría.
  - `Usuario.java`: Modelo de usuario con datos de autenticación y roles.
  - `Zone.java`: Modelo para zonas con atributos de nombre, tipo de bosque y área.

- **Capa de Seguridad (Security Layer):** Esta capa actúa de forma transversal a través de las demás capas para garantizar la confidencialidad, integridad y disponibilidad del sistema, protegiendo los recursos y datos mediante mecanismos de autenticación y autorización.

  - **Ubicación en el proyecto:** Paquete `com.espe.sistemaregistroforestal.security`.
  - **Función:** El paquete de seguridad gestiona la autenticación y autorización de usuarios, controlando el acceso a recursos y funciones según roles, para proteger el sistema contra accesos no autorizados.
  - **Tecnologías:** Java EE, filtros Servlet, sesiones HTTP.

- `AuthorizationFilter.java`: Filtro que controla la autenticación y autorización de usuarios para proteger recursos del sistema, permitiendo solo accesos autorizados según roles y redirigiendo usuarios no autenticados al login.
- `LogoutServlet.java`: Servlet que gestiona el cierre de sesión de usuarios invalidando la sesión HTTP y redirigiendo al usuario a la página de login para garantizar la seguridad al salir del sistema.
- `PasswordUtil.java`: Clase utilitaria que proporciona un método para cifrar contraseñas usando el algoritmo SHA-256, asegurando el almacenamiento seguro de las credenciales de usuario.

3.3. Diagrama de Arquitectura

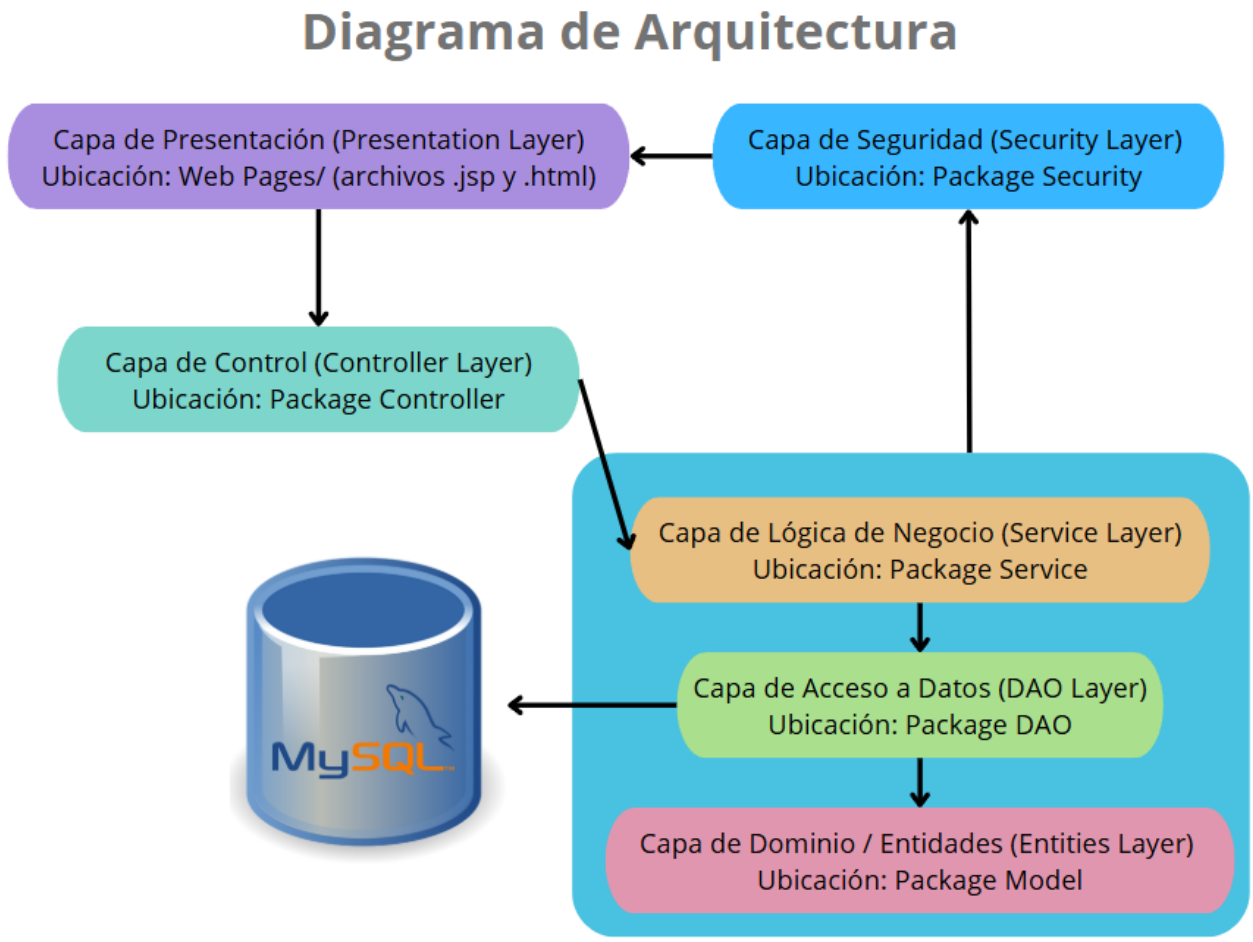


Figura 3.1: Diagrama de Arquitectura del Sistema

3.4. Estrategia de Implementación

3.4.1. Desarrollo Modular y por Capas

El sistema se implementará siguiendo una arquitectura modular basada en capas claramente definidas para asegurar una adecuada separación de responsabilidades, facilitar el mantenimiento y promover la reutilización de componentes. Las capas principales son: la Capa de Presentación, que provee la interfaz con el

usuario mediante páginas JSP, HTML, CSS y JavaScript; la Capa de Control, implementada con servlets que gestionan las solicitudes y validan entradas, funcionando como intermediarios entre la interfaz y la lógica de negocio; la Capa de Lógica de Negocio, que contiene los servicios SOAP encargados de implementar las reglas de negocio y exponer operaciones CRUD para facilitar la interoperabilidad; la Capa de Acceso a Datos, que abstrae la persistencia utilizando DAO y JDBC para interactuar con la base de datos; la Capa de Dominio o Entidades, que define los modelos que representan las estructuras de datos del sistema; y finalmente la Capa de Seguridad, que garantiza la autenticación, autorización y protección de los recursos mediante filtros, gestión de sesiones y encriptación de contraseñas.

Para asegurar la independencia y flexibilidad del sistema, se establecerán interfaces y contratos bien definidos entre las capas, lo que permitirá modificar la implementación interna de una capa sin afectar a las demás.

3.4.2. Uso de Patrones de Diseño

MVC (Modelo-Vista-Controlador):

- La capa de presentación será la "Vista", encargada de la interacción con el usuario.
- La capa de control actuará como Çontrolador "que recibe y valida las solicitudes.
- La capa de dominio y lógica de negocio conforman el "Modelo", encapsulando los datos y reglas de negocio.

DAO (Data Access Object):

- La capa de acceso a datos implementará DAOs para abstraer y centralizar toda la interacción con la base de datos, facilitando futuros cambios en la tecnología de persistencia.

Seguridad:

- Se implementan patrones y mecanismos de seguridad a nivel transversal, incluyendo autenticación, autorización y gestión segura de sesiones, para proteger los recursos y datos del sistema.
- Se utiliza un filtro de autorización para validar accesos y roles, así como cifrado de contraseñas con algoritmos seguros (SHA-256), garantizando confidencialidad e integridad.

3.4.3. Gestión de la Conexión a la Base de Datos

Conexión centralizada:

- Se utilizará una clase única (`ConnectionBdd.java`) para administrar la conexión con la base de datos MySQL, aplicando principios de reutilización y evitando múltiples conexiones dispersas.

Capítulo 4

Documentación de Base de Datos

4.1. Introducción

Este documento describe la estructura, diseño y detalles de la base de datos utilizada en el sistema **SistemaRegistroForestal**. La base de datos almacena la información relacionada con zonas forestales, especies de árboles, actividades de conservación, usuarios y otros datos relevantes para la gestión y consulta de información forestal.

El objetivo principal es garantizar una gestión eficiente, segura y coherente de la información para soportar las funcionalidades del sistema, incluyendo la administración y control de acceso de los usuarios según sus roles.

4.2. Modelo de Datos

4.2.1. Modelo Entidad-Relación (ER)

Se presenta el diagrama ER que representa las entidades principales, sus atributos y las relaciones entre ellas.

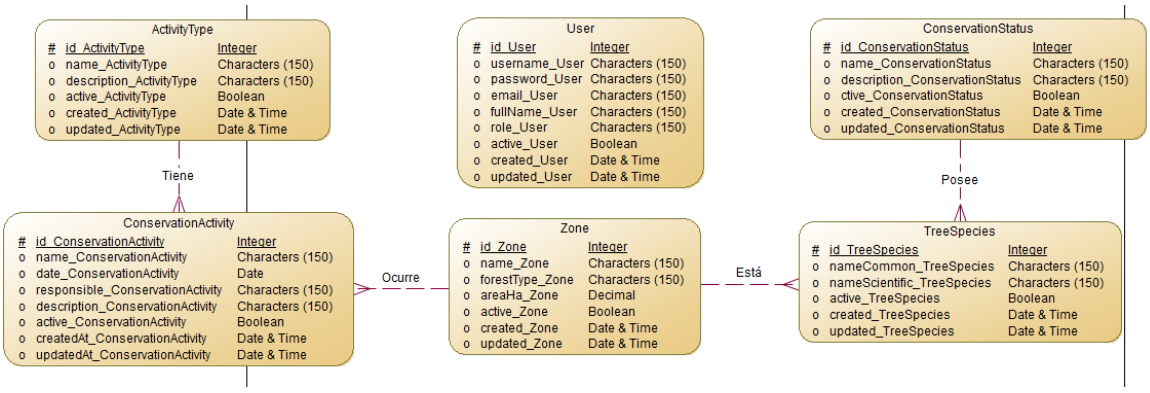


Figura 4.1: Modelo Entidad-Relación (ER)

4.2.2. Descripción de Entidades y Atributos

conservation_activities: Contiene información de actividades de conservación.

Campo	Tipo	Descripción
id	int(11)	Identificador único de la actividad (PK, autoincremental).
nombre_actividad	varchar(150)	Nombre descriptivo de la actividad.
fecha_actividad	date	Fecha en la que se realizó la actividad.
responsable	varchar(150)	Nombre del responsable o entidad a cargo.
tipo_actividad	enum	Tipo de actividad: Reforestación, Monitoreo, Control, etc.
descripcion	text	Descripción detallada de la actividad.
zona_id	int(11)	Identificador de la zona relacionada (FK).
activo	boolean	Indica si la actividad está activa (1) o inactiva (0).
creado_en	timestamp	Fecha y hora de creación del registro, por defecto la fecha actual.
actualizado_en	timestamp	Fecha y hora de la última actualización, se actualiza automáticamente.

estado_conservacion: Contiene información sobre los estados de conservación.

Campo	Tipo	Descripción
id	int(11)	Identificador único del estado de conservación (PK).
nombre	varchar(50)	Nombre del estado de conservación.
descripcion	text	Descripción opcional del estado.
activo	tinyint(1)	Indica si el estado está activo (1) o inactivo (0).
creado_en	timestamp	Fecha y hora de creación del registro, por defecto la fecha actual.
actualizado_en	timestamp	Fecha y hora de la última actualización, se actualiza automáticamente.

tipo_actividad: Contiene información sobre los tipos de actividad.

Campo	Tipo	Descripción
id	int(11)	Identificador único del tipo de actividad (PK).
nombre	varchar(100)	Nombre del tipo de actividad.
descripcion	text	Descripción opcional del tipo de actividad.
activo	tinyint(1)	Indica si el tipo está activo (1) o inactivo (0).
creado_en	timestamp	Fecha y hora de creación del registro, por defecto la fecha actual.
actualizado_en	timestamp	Fecha y hora de la última actualización, se actualiza automáticamente.

tree_species: Contiene información de especies arbóreas con sus características principales.

Campo	Tipo	Descripción
id	int(11)	Identificador único de la especie (PK).
nombre_comun	varchar(100)	Nombre común de la especie (no nulo).
nombre_cientifico	varchar(150)	Nombre científico de la especie (opcional).

Campo	Tipo	Descripción
estado_conservacion_id	int(11)	Referencia al estado de conservación (FK, opcional).
zona_id	int(11)	Referencia a la zona geográfica (FK, opcional).
activo	tinyint(1)	Indica si la especie está activa (1) o inactiva (0).
creado_en	timestamp	Fecha y hora de creación, valor por defecto la fecha actual.
actualizado_en	timestamp	Fecha y hora de última actualización, se actualiza automáticamente.

usuarios: Contiene información de los usuarios del sistema.

Campo	Tipo	Descripción
id	int(11)	Identificador único del usuario (PK).
username	varchar(50)	Nombre de usuario para autenticación (no nulo).
password	varchar(255)	Contraseña hasheada del usuario (no nulo).
email	varchar(100)	Correo electrónico del usuario (no nulo).
nombre_completo	varchar(100)	Nombre completo del usuario (no nulo).
rol	enum('admin','usuario','invitado')	Rol asignado al usuario, por defecto 'usuario'.
activo	tinyint(1)	Indica si el usuario está activo (1) o inactivo (0).
creado_en	timestamp	Fecha y hora de creación, valor por defecto la fecha actual.
actualizado_en	timestamp	Fecha y hora de última actualización, se actualiza automáticamente.

zones: Contiene información de zonas forestales protegidas.

Campo	Tipo	Descripción
id	int(11)	Identificador único de la zona (PK).
nombre	varchar(100)	Nombre de la zona protegida.
tipo_bosque	varchar(100)	Tipo de bosque (puede ser nulo).
area_ha	decimal(10,2)	Área total de la zona en hectáreas (puede ser nulo).
activo	tinyint(1)	Indica si la zona está activa (1) o no (0).
creado_en	timestamp	Fecha y hora de creación del registro, por defecto la fecha actual.
actualizado_en	timestamp	Fecha y hora de la última actualización, se actualiza automáticamente.

Relaciones:

- Cada actividad de conservación (acciones para proteger el ambiente) tiene un tipo específico (categoría de actividad).(Relación 1 a muchos)
- Cada actividad se realiza en una zona geográfica. Una zona puede tener muchas actividades. (Relación 1 a muchos)
- Cada especie de árbol tiene un estado de conservación (riesgo ambiental). Un estado puede aplicarse a muchas especies. (Relación 1 a muchos)

- Cada especie está asociada a una zona donde se encuentra o predomina. Una zona puede tener muchas especies. (Relación 1 a muchos)

Otras tablas sin relaciones:

- **Usuarios:** Representa a los usuarios del sistema con sus roles y credenciales, sin relaciones directas a otras tablas.

4.3. Diseño Físico

4.3.1. Motor de Base de Datos

Para el sistema SistemaRegistroForestal, se eligió el motor de base de datos MySQL por su robustez, amplia adopción, compatibilidad con sistemas web, y soporte para operaciones transaccionales. Además, MySQL permite la definición clara de tipos de datos, relaciones, y restricciones que garantizan la integridad de la información almacenada.

4.3.2. Estructura de Tablas

Tabla: conservation_activities

```
CREATE TABLE `conservation_activities` (
  `id` int(11) NOT NULL,
  `nombre_actividad` varchar(150) NOT NULL,
  `fecha_actividad` date NOT NULL,
  `responsable` varchar(100) DEFAULT NULL,
  `tipo_actividad_id` int(11) DEFAULT NULL,
  `descripcion` text DEFAULT NULL,
  `zona_id` int(11) DEFAULT NULL,
  `activo` tinyint(1) NOT NULL DEFAULT 1,
  `creado_en` timestamp NOT NULL DEFAULT current_timestamp(),
  `actualizado_en` timestamp NOT NULL DEFAULT current_timestamp()
  ON UPDATE current_timestamp()
);
```

Descripción: Tabla que almacena las actividades de conservación realizadas, incluyendo su tipo, fecha, responsable y la zona donde se llevan a cabo.

Tabla: estado_conservacion

```
CREATE TABLE `estado_conservacion` (
  `id` int(11) NOT NULL,
  `nombre` varchar(50) NOT NULL,
  `descripcion` text DEFAULT NULL,
  `activo` tinyint(1) NOT NULL DEFAULT 1,
  `creado_en` timestamp NOT NULL DEFAULT current_timestamp(),
  `actualizado_en` timestamp NOT NULL DEFAULT current_timestamp()
  ON UPDATE current_timestamp()
);
```

Descripción: Tabla que contiene los diferentes estados de conservación para clasificar las especies, con su nombre, descripción y estado activo.

Tabla: tipo_actividad

```
CREATE TABLE `tipo_actividad` (
  `id` int(11) NOT NULL,
  `nombre` varchar(100) NOT NULL,
  `descripcion` text DEFAULT NULL,
  `activo` tinyint(1) NOT NULL DEFAULT 1,
  `creado_en` timestamp NOT NULL DEFAULT current_timestamp(),
  `actualizado_en` timestamp NOT NULL DEFAULT current_timestamp()
  ON UPDATE current_timestamp()
);
```

Descripción: Tabla que almacena los tipos de actividades de conservación, con nombre, descripción y estado activo para su gestión.

Tabla: tree_species

```
CREATE TABLE `tree_species` (
  `id` int(11) NOT NULL,
  `nombre_comun` varchar(100) NOT NULL,
  `nombre_cientifico` varchar(150) DEFAULT NULL,
  `estado_conservacion_id` int(11) DEFAULT NULL,
  `zona_id` int(11) DEFAULT NULL,
  `activo` tinyint(1) NOT NULL DEFAULT 1,
  `creado_en` timestamp NOT NULL DEFAULT current_timestamp(),
  `actualizado_en` timestamp NOT NULL DEFAULT current_timestamp()
  ON UPDATE current_timestamp()
);
```

Descripción: Tabla que almacena especies de árboles con su nombre común, nombre científico, estado de conservación y la zona geográfica asociada.

Tabla: usuarios

```
CREATE TABLE ‘usuarios‘ (
  ‘id‘ int(11) NOT NULL,
  ‘username‘ varchar(50) NOT NULL,
  ‘password‘ varchar(255) NOT NULL,
  ‘email‘ varchar(100) NOT NULL,
  ‘nombre_completo‘ varchar(100) NOT NULL,
  ‘rol‘ enum('admin','usuario','invitado') DEFAULT 'usuario',
  ‘activo‘ tinyint(1) NOT NULL DEFAULT 1,
  ‘creado_en‘ timestamp NOT NULL DEFAULT current_timestamp(),
  ‘actualizado_en‘ timestamp NOT NULL DEFAULT current_timestamp()
  ON UPDATE current_timestamp()
);
```

Descripción: Tabla que almacena los usuarios del sistema con sus credenciales, correo, nombre completo y rol asignado.

Tabla: zones

```
CREATE TABLE ‘zones‘ (
  ‘id‘ int(11) NOT NULL,
  ‘nombre‘ varchar(100) NOT NULL,
  ‘tipo_bosque‘ varchar(100) DEFAULT NULL,
  ‘area_ha‘ decimal(10,2) DEFAULT NULL,
  ‘activo‘ tinyint(1) NOT NULL DEFAULT 1,
  ‘creado_en‘ timestamp NOT NULL DEFAULT current_timestamp(),
  ‘actualizado_en‘ timestamp NOT NULL DEFAULT current_timestamp()
  ON UPDATE current_timestamp()
);
```

Descripción: Tabla que almacena las zonas geográficas, incluyendo nombre, tipo de bosque y área en hectáreas.

4.3.3. Diagrama del Diseño Físico

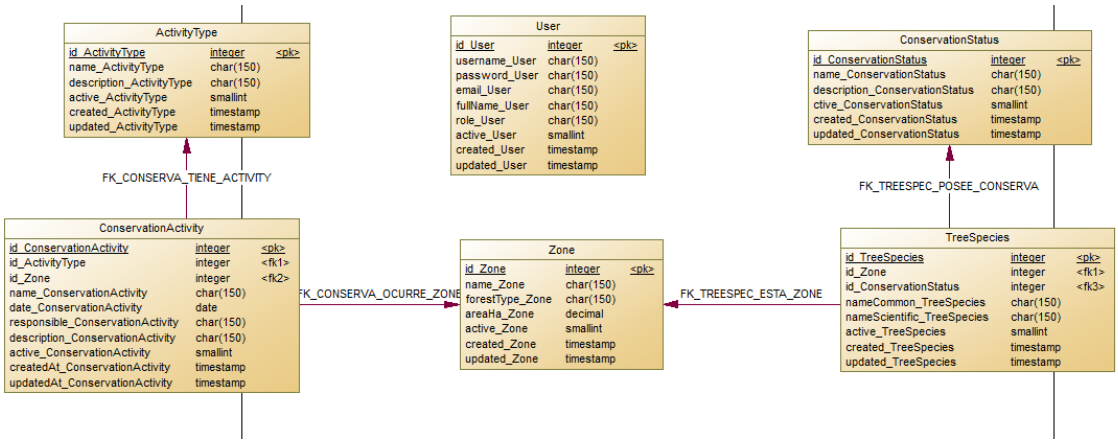


Figura 4.2: Diagrama del Diseño Físico

Capítulo 5

Documentación de Pruebas

5.1. Introducción

Este documento detalla los casos de prueba funcionales para verificar el correcto funcionamiento del sistema de registro forestal desarrollado en arquitectura N-capas. Las pruebas cubren la interacción del usuario con la interfaz gráfica, el control de acceso y autenticación, así como la correcta recepción y procesamiento de peticiones por parte de los controladores.

Sistema: SistemaRegistroForestal
Motor de BD: MySQL

5.2. Alcance de las Pruebas

Se evaluará:

- La interacción del usuario con la interfaz (páginas JSP).
- El proceso de autenticación de usuarios y control de acceso.
- El correcto flujo de datos desde los formularios hacia los controladores.
- La validación de entradas, redirecciones según rol y acceso a recursos protegidos.
- Integración entre capas: presentación, seguridad, controlador, servicio y DAO.

5.3. Casos de Prueba

Cuadro 5.1: Casos de prueba con todos los bordes

ID Caso	Descripción	Entrada	Resultado Esperado
TCA-01	Insertar actividad válida en tabla conservation_activities	nombre_actividad = 'Control Incendios', fecha_actividad = '2025-04-15', tipo_actividad = 'Control'	Registro exitoso
TCA-02	Insertar actividad sin nombre (campo obligatorio)	fecha_actividad = '2025-04-15', tipo_actividad = 'Control'	Error: nombre_actividad es obligatorio
TCA-03	Insertar actividad con tipo_actividad inválido	tipo_actividad = 'Exploración'	Error: valor no permitido en ENUM
TZ-01	Insertar zona válida en tabla zones	nombre = 'Reserva Norte', tipo_bosque = 'Seco', area_ha = 150.50	Zona registrada correctamente
TZ-02	Insertar zona sin tipo_bosque	nombre = 'Reserva Norte', area_ha = 150.50	Error: tipo_bosque es obligatorio
TZ-03	Insertar zona con tipo_bosque inválido	tipo_bosque = 'Nevado'	Error: valor no permitido en ENUM
TT-01	Insertar especie válida en tabla tree_species	nombre_comun = 'Guayacán', estado_conservacion = 'Vulnerable'	Especie registrada
TT-02	Insertar especie sin nombre común	estado_conservacion = 'Vulnerable'	Error: nombre_comun es obligatorio
TT-03	Insertar especie con estado_conservacion inválido	estado_conservacion = 'Crítico'	Error: valor no permitido en ENUM
SAF-01	Acceso a recurso protegido sin iniciar sesión	Ingresar URL directa a /Zones.jsp sin sesión iniciada	Redirección a la página de login
SAF-02	Iniciar sesión con credenciales válidas	usuario = admin, clave = admin123	Inicio de sesión exitoso, redirección al panel principal
SAF-03	Iniciar sesión con credenciales inválidas	usuario = admin, clave = incorrecta	Mensaje de error: credenciales incorrectas
SAF-04	Acceso con rol no autorizado a página restringida	Usuario con rol "Usuario" intenta acceder a módulo de administración	Acceso denegado o redirección a página de error
SAF-05	Cierre de sesión correcto	Click en "Cerrar sesión" en menú	Sesión finalizada, redirección al login

Continúa en la siguiente página

ID Caso	Descripción	Entrada	Resultado Esperado
SOAP-01	Llamar servicio SOAP para insertar especie válida	Método `createTreeSpecies(nombre = 'Caoba', estado = 'Vulnerable')`	Respuesta: `OK - Especie registrada`
SOAP-02	Llamar servicio SOAP con datos inválidos	Método `createTreeSpecies(nombre = '', estado = 'Inexistente')`	Respuesta: `Error - Campos inválidos`
SOAP-03	Consultar especie por ID vía SOAP	Método `getTreeSpeciesById(id = 3)`	Devuelve objeto `TreeSpecies` con datos esperados
SOAP-04	Eliminar lógicamente una zona vía SOAP	Método `deleteZone(id = 5)`	Zona marcada como inactiva en BD, sin eliminar físicamente
SOAP-05	Error al consumir servicio SOAP sin conexión	Se simula pérdida de conexión al host SOAP	Excepción de comunicación capturada, mensaje de error mostrado

5.4. Casos de Prueba para Validación de Funcionalidades

5.4.1. Pruebas de Inserción

- **TCA-01:** Verificar que se puede insertar correctamente una actividad de conservación con todos los campos válidos.
- **TZ-01:** Confirmar que una zona forestal puede ser registrada con información completa y tipos de bosque válidos.
- **TT-01:** Validar el registro de especies arbóreas con datos botánicos correctos.

5.4.2. Pruebas de Validación de Restricciones

- **TCA-02, TT-02:** Verificar que los campos obligatorios son requeridos para el registro exitoso.
- **TCA-03, TZ-03, TT-03:** Confirmar que los valores ENUM solo aceptan opciones predefinidas.

5.4.3. Pruebas de Relaciones entre Tablas

- Verificar que las claves foráneas mantienen la integridad referencial.
- Confirmar que las relaciones muchos a muchos funcionan correctamente a través de tablas intermedias.
- Validar que las eliminaciones en cascada o restricciones funcionan según el diseño.

5.4.4. Pruebas de Seguridad

- **SAF-01:** Validar que los recursos protegidos redirigen al login si no hay sesión activa.
- **SAF-02:** Confirmar que usuarios con credenciales válidas pueden iniciar sesión.
- **SAF-03:** Validar que se detectan credenciales incorrectas.

- **SAF-04:** Comprobar que los usuarios solo acceden a recursos autorizados según su rol.
- **SAF-05:** Verificar el correcto cierre de sesión y eliminación de la sesión activa.

5.4.5. Pruebas de Servicios SOAP

- **SOAP-01:** Validar que se pueden insertar especies vía servicio SOAP correctamente.
- **SOAP-02:** Verificar la validación de entradas en el servicio SOAP.
- **SOAP-03:** Comprobar que la consulta por ID retorna la especie correcta.
- **SOAP-04:** Validar la eliminación lógica desde el servicio.
- **SOAP-05:** Verificar el manejo de errores de conectividad al invocar el servicio.

5.5. Resultados de Pruebas

Las pruebas ejecutadas permitieron validar que el sistema cumple con los requerimientos funcionales y no funcionales establecidos. En particular, se observaron los siguientes resultados:

- Las operaciones CRUD básicas para las distintas entidades (actividades de conservación, zonas, especies) se ejecutan correctamente, garantizando la creación, lectura, actualización y eliminación de registros.
- Las validaciones implementadas para los campos obligatorios y los valores de tipo ENUM funcionan según lo esperado, previniendo la inserción de datos inválidos o incompletos.
- Se mantiene la integridad referencial entre las tablas relacionadas, asegurando la consistencia y coherencia de los datos en el sistema.
- Las consultas que involucran múltiples tablas, incluyendo búsquedas y filtros complejos, se ejecutan con eficiencia, ofreciendo tiempos de respuesta adecuados para la aplicación.
- La capa de seguridad ha demostrado su efectividad al proteger los recursos restringidos, obligando a los usuarios a autenticarse antes de acceder a módulos protegidos. Además, el control de roles permite limitar el acceso según el perfil del usuario, evitando accesos no autorizados.
- Las funcionalidades de cierre de sesión operan correctamente, invalidando las sesiones activas y redirigiendo a la página de login para garantizar la confidencialidad.
- Los servicios web SOAP utilizados para la gestión de especies y zonas cumplen con los requisitos de disponibilidad y manejo de errores. Las operaciones CRUD expuestas por SOAP funcionan adecuadamente con datos válidos, mientras que las validaciones impiden la inserción de datos erróneos. También se manejan correctamente las excepciones en caso de pérdida de conexión o errores en la comunicación.

5.6. Conclusiones

El sistema desarrollado para el Registro Forestal cumple satisfactoriamente con los requisitos funcionales y no funcionales establecidos inicialmente. La estructura de la base de datos y las restricciones aplicadas garantizan la integridad, coherencia y confiabilidad de los datos almacenados.

Las pruebas realizadas demostraron que las operaciones CRUD sobre las distintas entidades funcionan correctamente y que las validaciones de campos obligatorios y tipos enumerados impiden la inserción de datos

inválidos, manteniendo la calidad de la información.

Asimismo, la capa de seguridad implementada asegura el control adecuado de acceso a los recursos del sistema mediante mecanismos de autenticación y autorización basados en roles, protegiendo la confidencialidad y evitando accesos no autorizados.

Por último, los servicios web SOAP integrados ofrecen una interfaz estable y confiable para la manipulación remota de datos, gestionando correctamente las operaciones y manejando de forma adecuada los errores de comunicación o validación, lo que contribuye a la interoperabilidad y escalabilidad del sistema.

En conjunto, el sistema se presenta como una solución robusta y confiable para la gestión y registro de información forestal.

Capítulo 6

Manual del Desarrollador

6.1. Introducción

Este manual está orientado a proveer a los desarrolladores toda la información necesaria para comprender, mantener, mejorar y ampliar el Sistema de Registro Forestal. El sistema ha sido diseñado bajo una arquitectura de tipo N-Capas, la cual facilita la separación clara de responsabilidades, lo que contribuye a que el código sea más mantenible, escalable y seguro.

La aplicación gestiona información fundamental para la conservación ambiental, incluyendo zonas forestales, especies de árboles y actividades de conservación, apoyando la toma de decisiones y el monitoreo sostenible de los recursos naturales.

Desde el punto de vista tecnológico, el backend está desarrollado en Java EE, utilizando Servlets para la lógica de negocio y JDBC para el acceso a datos en una base MySQL administrada localmente con XAMPP. El frontend se implementa con tecnologías web estándar como JSP, HTML, CSS y JavaScript, apoyado con Bootstrap para garantizar una interfaz responsiva y amigable en diferentes dispositivos.

El sistema se estructura en seis capas funcionales: Presentación, Controladores, Lógica de Negocio, Servicios, Acceso a Datos (DAO), Modelo y Seguridad. Esta organización modular no solo mejora la mantenibilidad, sino que también asegura que la seguridad, el control de acceso por roles y la interoperabilidad a través de servicios web SOAP se gestionen de forma eficiente.

Este manual detalla la estructura del proyecto, el flujo general del sistema, los componentes principales, así como las buenas prácticas y recomendaciones para el desarrollo futuro. Además, se proponen posibles extensiones para ampliar la funcionalidad del sistema, adaptándolo a nuevas necesidades del manejo forestal y conservación ambiental.

6.2. Tecnologías Utilizadas

Backend:

- Java EE para la implementación de la lógica de negocio y controladores.
- Servlets para el manejo y procesamiento de peticiones HTTP en la capa de control.
- JAX-WS (Java API for XML Web Services) para la implementación y publicación de servicios SOAP.
- JDBC para la conexión y operaciones con la base de datos MySQL.

- Mecanismos de seguridad basados en filtros Servlet, autenticación, autorización y cifrado de contraseñas (SHA-256).

Frontend:

- JSP para la generación dinámica de páginas web.
- HTML5 y CSS3 para la estructura y estilos visuales.
- Bootstrap para diseño responsivo y componentes UI.
- JavaScript para validaciones y dinámicas en el cliente.

Base de Datos:

- MySQL gestionado localmente con XAMPP, facilitando la administración del servidor en entornos de desarrollo.

Entorno de Desarrollo:

- NetBeans IDE, que ofrece integración con Java EE, JSP y soporte para servicios web.

6.3. Estructura del Proyecto

La organización del código sigue el patrón N-Capas para separar las responsabilidades en distintas carpetas y paquetes:

`/Web Pages/` Contiene las páginas JSP que forman la interfaz visible para el usuario.

- `ConservationActivities.jsp`: Lista de actividades de conservación con control de acceso y modales para formularios.
- `ConservationActivitiesForm.jsp`: Formulario modal para crear o editar actividades de conservación con validación y selección de tipos y zonas.
- `index.jsp`: Página principal con secciones informativas, menú incluido, estilos CSS y formulario de contacto.
- `TipoActividadFrm.jsp`: Modal para crear o editar tipos de actividad con validación y estado activo.
- `TreeSpecies.jsp`: Lista de especies de árbol con control de acceso para agregar, editar y eliminar solo por administradores, mostrando modales para formularios y tabla dinámica con DataTables.
- `TreeSpeciesForm.jsp`: Modal con formulario para crear/editar especies, con campos básicos y validación.
- `Zones.jsp`: Lista de zonas con control de acceso para administración y modales para crear o editar zonas.
- `ZonesFrm.jsp`: Modal para crear o editar zonas, con formulario validado y selección dinámica de tipos de bosque.
- `index.html`: Página inicial que redirige automáticamente al login.jsp para iniciar sesión.
- `indexAdmin.jsp`: Página principal del panel de administración que ofrece un dashboard visual con tarjetas para acceso rápido.
- `login.jsp`: Página para iniciar sesión con formulario y mensaje de error.

- `menudinamico.jsp`: Barra de navegación con enlaces principales y estilos Bootstrap.
- `menuRoles.jsp`: Menú de navegación dinámico según rol (admin o usuario).

`/com/espe/sistemaregistroforestal/controller/` **(Package Controller)** Paquete encargado de manejar las peticiones de los usuarios. Los controladores gestionan las solicitudes del cliente y las encaminan hacia la capa de servicios.

- `ConservationActivitiesController.java`: Controlador Servlet para CRUD de actividades de conservación con control de acceso por rol y manejo de formularios.
- `LoginServlet.java`: Servlet para autenticación de usuarios con redirección según rol y manejo básico de sesión.
- `TipoActividadController.java`: Controlador CRUD para tipos de actividad con control de acceso y gestión de formularios.
- `TreeSpeciesController.java`: Controlador servlet que gestiona CRUD de especies arbóreas con control de acceso por rol y carga de datos relacionados.
- `ZoneController.java`: Servlet controlador para CRUD de zonas con control de acceso y manejo de tipos de bosque.

`/com/espe/sistemaregistroforestal/service/` **(Package Service)** Esta capa implementa la lógica de negocio a través de servicios SOAP. Los servicios procesan la información recibida del controlador, ejecutan las reglas de negocio y acceden a la capa DAO para persistencia.

- `CrudSpeciesPublication.java`: Clase que publica el servicio SOAP para operaciones CRUD sobre especies forestales, exponiendo métodos para crear, consultar, actualizar y eliminar especies, y manteniendo el servicio activo en una URL específica.
- `CrudSpeciesService.java`: Servicio SOAP que implementa operaciones CRUD para gestionar especies forestales, incluyendo creación, consulta, actualización y eliminación, además de métodos auxiliares para obtener zonas y estados de conservación.
- `CrudZonesPublication.java`: Clase encargada de publicar el servicio SOAP CrudZonesService en una URL específica para que esté disponible y accesible para clientes.
- `CrudZonesService.java`: Servicio SOAP que expone operaciones CRUD para la gestión de zonas forestales, incluyendo creación, consulta, actualización y eliminación lógica de zonas. Implementa validaciones de negocio y usa un DAO para acceso a datos, asegurando integridad y registro de actividades mediante logs.
- `ConservationActivitiesService.java`: clase que gestiona las operaciones CRUD para actividades de conservación, validando datos antes de crear o actualizar, y usando un DAO para acceder a la base de datos.
- `EstadoConservacionService.java`: Clase que usa TreeSpeciesDAO para obtener la lista de todos los estados de conservación.
- `TipoActividadService.java`: Servicio que gestiona tipos de actividad: lista, busca, crea, actualiza y elimina con validación básica.
- `TreeSpeciesService.java`: Servicio que maneja especies de árboles: lista, busca, crea, actualiza y elimina (lógico) con validaciones y control de nombres duplicados.
- `UsuarioService.java`: Servicio que autentica usuarios verificando su nombre y contraseña hasheada con SHA-256.
- `ZoneService.java`: Servicio que maneja operaciones CRUD y validaciones básicas para zonas forestales, incluyendo creación, actualización, listado y borrado lógico.

`/com/espe/sistemaregistroforestal/dao/` **(Package DAO)** Capa encargada de la persistencia y acceso a datos, utilizando JDBC para conectarse a la base de datos MySQL.

- `ConnectionBdd.java`: Gestiona la conexión JDBC con la base de datos MySQL.
- `ConservationActivitiesDAO.java`: Acceso a datos para actividades de conservación con métodos CRUD y eliminación lógica.
- `EstadoConservacionDAO.java`: Acceso a datos para estados de conservación con métodos para listar y buscar por ID, manejando conexiones internas y externas.
- `TipoActividadDAO.java`: Acceso a datos para tipos de actividad con CRUD y eliminación lógica, gestionando fechas y estado activo.
- `TreeSpeciesDAO.java`: DAO para operaciones CRUD de especies y obtención de zonas y estados activos.
- `UsuarioDAO.java`: DAO para obtener y autenticar usuarios activos por nombre y contraseña.
- `ZoneDAO.java`: DAO para CRUD y consultas de zonas con eliminación lógica.

`/com/espe/sistemaregistroforestal/model/` **(Package Model)** Contiene las clases que representan las entidades del negocio, conocidas como modelos o POJOs, que definen atributos y métodos de acceso sin lógica de negocio.

- `ConservationActivities.java`: Modelo que representa una actividad de conservación con sus atributos básicos.
- `EstadoConservacion.java`: Modelo para el estado de conservación con nombre, descripción y estado activo.
- `TipoActividad.java`: Modelo que representa un tipo de actividad con atributos básicos y marcas temporales.
- `TipoBosque.java`: Enum que define tipos de bosque con nombre desplegable y método para obtener por texto.
- `TreeSpecies.java`: Modelo de especie de árbol con atributos básicos y metadatos de auditoría.
- `Usuario.java`: Modelo de usuario con datos de autenticación y roles.
- `Zone.java`: Modelo para zonas con atributos de nombre, tipo de bosque y área.

`/com/espe/sistemaregistroforestal/security/` **(Package Security)** Paquete transversal que garantiza la seguridad de la aplicación, controlando autenticación, autorización y manejo de sesiones.

- `AuthorizationFilter.java`: Filtro que controla la autenticación y autorización de usuarios para proteger recursos del sistema, permitiendo solo accesos autorizados según roles y redirigiendo usuarios no autenticados al login.
- `LogoutServlet.java`: Servlet que gestiona el cierre de sesión de usuarios invalidando la sesión HTTP y redirigiendo al usuario a la página de login para garantizar la seguridad al salir del sistema.
- `PasswordUtil.java`: Clase utilitaria que proporciona un método para cifrar contraseñas usando el algoritmo SHA-256, asegurando el almacenamiento seguro de las credenciales de usuario.

6.4. Flujo General de Operación

El flujo de operación del sistema se ha robustecido con la integración de una capa de seguridad y la exposición de servicios SOAP, lo que impacta la interacción con los usuarios y la comunicación interna entre capas.

6.4.1. Interacción del Usuario con la Interfaz (Frontend)

El usuario inicia la interacción con la aplicación a través de las páginas JSP, que conforman la interfaz gráfica.

- Páginas como `index.jsp`, `Zones.jsp`, `TreeSpecies.jsp` o `ConservationActivities.jsp` presentan formularios, listados y opciones para gestionar datos forestales.
- Las páginas incluyen componentes comunes para navegación y presentación, como `menudinamico.jsp` y `menuRoles.jsp`, que adaptan la interfaz según el rol del usuario.
- Validaciones iniciales pueden realizarse en el navegador usando JavaScript para mejorar la experiencia de usuario y reducir la carga en el servidor.
- El acceso inicial al sistema se realiza a través de `index.html`, que redirige a `login.jsp` para el proceso de autenticación.

6.4.2. Control de Acceso y Autenticación (Capa de Seguridad)

Antes de que cualquier solicitud del usuario llegue a los controladores de negocio, la capa de seguridad interviene para validar la identidad y los permisos.

- El usuario accede a `login.jsp` e introduce sus credenciales.
- `LoginServlet.java` procesa estas credenciales. Internamente, utiliza `UsuarioService.java` y `UsuarioDAO.java` para verificar el usuario y su contraseña (hasheada con SHA-256 a través de `PasswordUtil.java`).
- Tras una autenticación exitosa, el sistema establece la sesión del usuario y redirige según su rol (ej., a `indexAdmin.jsp` para administradores).
- El `AuthorizationFilter.java` actúa como un punto de control central para proteger los recursos. Intercepta todas las peticiones a URLs protegidas, verificando la autenticación y los permisos del usuario antes de permitir el acceso al controlador correspondiente. Si el usuario no está autenticado o no tiene los permisos necesarios, el filtro redirige a `login.jsp` o a una página de error, garantizando así la integridad y confidencialidad del sistema.
- Para el cierre de sesión, `LogoutServlet.java` invalida la sesión HTTP, redirigiendo al usuario a la página de login.

6.4.3. Recepción de la Petición por el Controlador (Controller)

Una vez que la petición del usuario ha pasado los filtros de seguridad, esta es recibida por el servlet correspondiente dentro del paquete `/controller`.

- El controlador (`ZonesController.java`, `TreeSpeciesController.java`, etc.) recibe la petición HTTP (ya sea desde el frontend o, en el caso de ser un servicio, a través del mecanismo SOAP), procesa parámetros y realiza validaciones preliminares para asegurar datos correctos.
- Aquí se define si la lógica de negocio se procesa directamente a través de un servicio local (para la UI del frontend) o si se invoca a un servicio SOAP expuesto.

6.4.4. Ejecución de la Lógica de Negocio en la Capa Servicio (Service) o a través de Servicios SOAP

La capa de servicios (`ZonesService.java`, `TreeSpeciesService.java`, etc.) recibe la petición desde el controlador o directamente como una solicitud de un cliente SOAP.

- **Para solicitudes internas (desde el frontend):** Los servicios implementan la lógica de negocio, ejecutan validaciones complejas y procesos específicos (validación de formatos, cálculos, decisiones condicionales).
- **Para solicitudes externas (a través de SOAP):** Clases como `CrudSpeciesPublication.java` y `CrudZonesPublication.java` exponen los servicios `CrudSpeciesService.java` y `CrudZonesService.java` respectivamente. Estos servicios SOAP implementan las operaciones CRUD y la lógica de negocio para entidades específicas, permitiendo que otras aplicaciones puedan interactuar con el sistema.
- Independientemente de si la petición proviene de un controlador interno o de un cliente SOAP, el servicio llama a la capa DAO para realizar la persistencia o recuperación de datos desde la base de datos.

6.4.5. Persistencia de Datos en la Base de Datos mediante DAO

La capa DAO (`ZonesDAO.java`, `TreeSpeciesDAO.java`, etc.) contiene el código que interactúa con MySQL a través de JDBC.

- Utiliza la clase `ConnectionBdd.java` para abrir y cerrar conexiones de forma segura y eficiente.
- Ejecuta sentencias SQL para insertar, actualizar, eliminar o consultar datos.
- Devuelve los resultados o confirma la ejecución exitosa al servicio.

6.4.6. Respuesta hacia el Usuario o Cliente SOAP

La información procesada y el resultado de la operación (éxito, error, datos solicitados) se transmiten desde el DAO hacia la capa de servicio y de allí al controlador o se devuelve directamente al cliente SOAP.

- **Para solicitudes internas (frontend):** El controlador prepara la respuesta para el usuario, estableciendo atributos o redirigiendo a la página JSP adecuada. Finalmente, la JSP muestra los resultados o mensajes correspondientes en la interfaz, completando el ciclo de la petición.
- **Para solicitudes externas (SOAP):** El servicio SOAP devuelve la respuesta estructurada (generalmente XML) al cliente que realizó la petición, permitiendo la interoperabilidad con otros sistemas.

6.5. Recomendaciones para el Desarrollo

- **Separación de responsabilidades:** Evitar mezclar código de presentación (JSP) con lógica de negocio o acceso a datos. Esto se consigue respetando el patrón MVC (Modelo-Vista-Controlador).
- **Patrón MVC:** Mantener el flujo de datos y control bien definido para facilitar futuras modificaciones y la incorporación de nuevas funcionalidades.
- **Gestión de conexiones:** Utilizar la clase `ConnectionBdd.java` para abrir y cerrar conexiones a la base de datos correctamente, evitando fugas que puedan afectar el rendimiento o causar errores.
- **Documentación:** Comentar adecuadamente las funciones y métodos más importantes para facilitar la comprensión de otros desarrolladores y la extensión del sistema.

6.6. Buenas prácticas utilizadas

- **Nomenclatura consistente:** En Java se recomienda usar camelCase para variables y métodos (ej. `nombreActividad`, `obtenerPorId`), y PascalCase para nombres de clases y enums (ej. `ConservationActivitiesController`, `TipoActividad`). Esto mejora la legibilidad y mantiene coherencia con las convenciones oficiales.
- **Separación clara entre capas:** El controlador debe limitarse a recibir las peticiones, invocar la lógica de negocio en los servicios y gestionar las respuestas. Evita incluir lógica de negocio o manipulación directa de datos en el controlador o en las vistas.
- **Validaciones robustas:** Siempre validar parámetros recibidos, especialmente cuando se convierten a tipos numéricos o fechas, para evitar excepciones inesperadas. Implementa validaciones tanto en frontend como en backend.
- **Documentación clara:** Añade comentarios breves y precisos en métodos, indicando qué hace cada bloque de código, especialmente en el controlador y en la capa de servicio.
- **Evitar repetición de código:** Si ciertas operaciones se repiten (como la obtención de parámetros o la validación), considera abstraerlas en métodos auxiliares privados para mejorar la mantenibilidad.

6.7. Extensiones Futuras Sugeridas

- **Sistema de reportes avanzados:** Generar informes con gráficos, filtros y exportación a formatos como PDF o Excel.
- **Alertas y notificaciones automáticas:** Incorporar notificaciones por email o mensajes internos ante eventos relevantes del sistema, como registros críticos o alertas de conservación.
- **Integración con Sistemas de Información Geográfica (SIG):** Permite visualizar las zonas forestales, actividades de conservación y ubicaciones de especies en un mapa interactivo. Esto podría lograrse integrando librerías de mapeo (como OpenLayers o Leaflet en el frontend) y almacenando coordenadas geográficas en la base de datos.
- **Módulo de Auditoría y Logs Detallados:** Más allá de los logs básicos, implementar un sistema que registre detalladamente quién realizó qué acción, cuándo y desde dónde (IP). Esto es crucial para la trazabilidad y la seguridad, especialmente en un sistema que gestiona datos críticos de conservación.