UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE
Matriz – Sangolquí

TALLER 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, 16 de Mayo del 2025

Contents

1 DOCUMENTO 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

1.2.1 Objetivo general

Desarrollar un Sistema de Registro Forestal que permita gestionar de manera eficiente la información relacionada con zonas forestales, especies de árboles y actividades de conservación, utilizando una arquitectura N-Capas con Java EE y MySQL.

1.2.2 Objetivos específicos

- Implementar módulos para el registro, consulta y actualización de datos de zonas forestales.
- Desarrollar funcionalidades para gestionar información sobre diferentes especies de árboles.
- Garantizar la integridad y seguridad de la información almacenada.
- Proveer una interfaz amigable para los usuarios que facilite la interacción con el sistema.

1.2.3 Alcance

El sistema de Registro Forestal que se desarrollará permitirá la gestión de información fundamental sobre zonas forestales, especies de árboles y actividades de conservación, facilitando el registro, consulta y actualización de estos datos mediante una interfaz sencilla. El enfoque principal está en apoyar las tareas administrativas y de monitoreo ambiental a nivel local o institucional, sin incluir funcionalidades avanzadas como integración con dispositivos de campo, generación de reportes complejos o sistemas de administración de usuarios avanzados.

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 registrar, consultar y actualizar datos relevantes, facilitando el seguimiento y control de los recursos forestales por parte de los usuarios encargados de su administración. Entre sus funcionalidades principales se encuentra la posibilidad de realizar consultas específicas sobre las zonas registradas, lo que permite un acceso rápido y organizado a la información.

La implementación se realizó utilizando una arquitectura N-Capas, aplicando el patrón de diseño Modelo-Vista-Controlador (MVC), lo cual asegura una separación clara entre la lógica de presentación, lógica de negocio y acceso a datos, favoreciendo el mantenimiento y escalabilidad del sistema. 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.

Este sistema está orientado a ser usado, donde se requiere una herramienta confiable para el registro y seguimiento de información ambiental. Su diseño modular también permite futuras ampliaciones, como la integración de reportes o nuevos módulos funcionales.

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.

1.4.1 Entidades Principales

El sistema está centrado en tres entidades clave:

ZONAS: Representa zonas geográficas donde se realizan actividades de conservación o se registran especies de árboles. Cada zona contiene información relevante como nombre, ubicación, provincia, tipo de bosque (Seco, Húmedo Tropical, Montano, Manglar, Otro), área en hectáreas y descripción.

ESPECIES DE ÁRBOLES: Incluyen información botánica sobre las especies registradas dentro de una zona, como nombre común, nombre científico, familia botánica, estado de conservación (enum), uso principal, altura máxima y zona.

ACTIVIDADES: Reflejan las acciones de manejo, conservación o intervención realizadas dentro de una zona. Esta entidad guarda información como el nombre

de la actividad, fecha, responsable, tipo de actividad (enum), descripción, zona asociada y si está activa.

1.4.2 Actores del Sistema

El actor del sistema representa a cualquier persona que utilice el sistema para registrar, consultar o modificar información sobre zonas forestales, especies y actividades, por lo que todas las funcionalidades están disponibles para el usuario del software.

Funciones del Actor del Sistema:

- Registra y consulta información relacionada con zonas forestales.
- Ingresa especies de árboles identificadas en campo.
- Registra las actividades de conservación realizadas.
- Crea, edita y elimina registros de zonas, especies y actividades.

1.4.3 Procesos del Negocio

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

Registrar Zona Forestal

Descripción: Permite ingresar una nueva zona forestal con sus datos básicos.

Entrada: Nombre de la zona, ubicación, superficie, observaciones.

Salida: Zona almacenada en el sistema con estado activo.

Registrar Especie de Árbol

Descripción: Permite ingresar una especie de árbol identificada en una zona y se la asocia con la misma.

Entrada: Nombre común, nombre científico, características, zona relacionada.

Salida: Especie almacenada en el sistema con estado activo.

Registrar Actividad de Conservación

Descripción: Permite documentar actividades realizadas en una zona forestal.

Entrada: Tipo de actividad, descripción, fecha, zona correspondiente.

Salida: Actividad registrada en el sistema con estado activo.

Editar Zona, Especie o Actividad

Descripción: Permite modificar los datos previamente registrados de zonas forestales, especies de árboles o actividades de conservación.

Entrada: Registro seleccionado, nuevos datos a actualizar.

Salida: Registro actualizado.

Borrado Lógico de Registros

Descripción: Permite desactivar un registro sin eliminarlo físicamente de la base de datos (zona, especie o actividad).

Entrada: Identificador del registro a desactivar.

Salida: Registro marcado como inactivo o eliminado lógicamente.

Consultar Registros

Descripción: Permite buscar y filtrar información de zonas forestales, especies de árboles y actividades de conservación según distintos criterios.

Entrada: Criterios de búsqueda (nombre, fecha, zona, etc.).

Salida: Lista de registros que cumplen los criterios, incluyendo activos e inactivos si se desea.

Visualizar Información de Registros

Descripción: Permite al usuario ver en detalle la información de cualquier zona, especie o actividad registrada.

Entrada: Selección del registro a visualizar.

Salida: Información completa del registro en pantalla.

1.5 Modelo Conceptual del Sistema

Diagrama del 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.

- **Módulo de Gestión de Usuarios y Roles:** Este módulo se enfocaría en la implementación de un sistema de autenticación y autorización para controlar el acceso al sistema. Se definirían diferentes roles de usuario, cada uno con permisos específicos (por ejemplo, administrador, técnico de campo, investigador), y se registraría la actividad de los usuarios con fines de auditoría.
- **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.

2 DOCUMENTACIÓN DE REQUISITOS FUNCIONES Y NO FUNCIONALES

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**, permitirá a los usuarios autorizados:

- 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 y consulta de datos
- 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.

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.

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 perspectiva general del sistema, sus funciones principales, características de los usuarios y restricciones.
- **Sección 3 - Requisitos específicos:** Detalla los requisitos funcionales y no funcionales del sistema, las interfaces externas y los atributos de calidad.
- **Sección 4 - Apéndices:** Contiene información complementaria relevante para la especificación.

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. Utiliza una arquitectura N-Capas que incluye:

- Capa de presentación (JSF/Servlets).
- Capa de lógica de negocio (EJBs).
- Capa de acceso a datos (JPA o JDBC).
- Base de datos MySQL.

El sistema funcionará de manera independiente, aunque podrá interactuar con sistemas externos a través de interfaces para el intercambio de información geográfica o biológica si fuera necesario en el futuro.

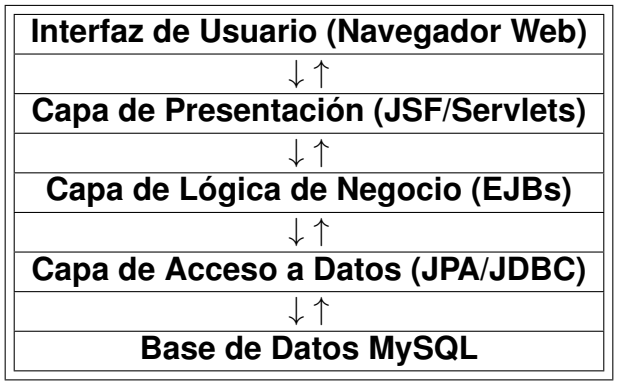

Figure 1: Arquitectura N-Capas del Sistema de Registro Forestal (SRS)

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 registros de zonas forestales con sus características geográficas y ecológicas.
- **Gestión de especies arbóreas:** Registro y mantenimiento del catálogo de especies, asociándolas a las zonas donde se encuentran.
- **Registro de actividades de conservación:** Documentación de intervenciones, monitoreos y acciones realizadas en cada zona forestal.
- **Sistema de búsqueda avanzada:** Filtros por diferentes criterios (geográficos, biológicos, temporales).
- **Generación de reportes:** Exportación de datos en formatos PDF y Excel.

2.2.3 Características de los usuarios (SRS)

El sistema contempla dos tipos principales de usuarios:

- **Técnico:** Profesionales del área forestal y ambiental encargados de registrar y actualizar la información. Tienen permisos para crear y modificar registros. Requieren capacitación en las funcionalidades operativas.
- **Consultor:** Usuarios para consultar información. Pueden ser investigadores, personal de campo o público interesado. Requieren capacitación mínima sobre las funciones de consulta.

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.
  - 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:**
  - Debe cumplir con estándares de protección de datos personales.
  - Las comunicaciones deben ser cifradas mediante HTTPS.

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.

- **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.

2.2.6 Requisitos futuros (SRS)

Los siguientes requisitos podrían considerarse para futuras versiones del sistema:

- Integración con sistemas de información geográfica (GIS).
- Aplicación móvil para trabajo de campo.
- Módulo de análisis estadístico y visualización de datos.
- Funcionalidades de inteligencia artificial para identificación de especies.
- Ampliación para incluir fauna asociada a cada zona forestal.

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.
- La base de datos requiere MySQL 8.0 o superior.

Interfaz de comunicaciones (SRS)

- Se utilizará el protocolo HTTP/HTTPS para todas las comunicaciones.
- Las transferencias de datos utilizarán formato JSON para el intercambio de información.
- Se implementará un mecanismo de manejo de sesiones para mantener la autenticación de los usuarios.

2.3.2 Funciones (SRS)

RF01 – Registro de zonas forestales (SRS)

- El sistema debe permitir registrar zonas indicando: nombre, región, coordenadas geográficas, superficie, tipo de bosque, y estado de conservación.
- Se debe poder adjuntar archivos (imágenes, documentos) a cada registro de zona.
- El registro debe incluir campos para metadatos adicionales (altitud, temperatura promedio, precipitación).

RF02 – Consulta de zonas forestales (SRS)

- El usuario podrá consultar zonas mediante filtros: nombre, región, tipo de bosque o estado.
- Los resultados deben poder ordenarse por diferentes criterios.
- Debe existir una vista de mapa y una vista de lista.
- El sistema debe permitir ver el detalle completo de cada zona.

RF03 – Actualización de zonas forestales (SRS)

- El sistema debe permitir modificar los datos de zonas previamente registradas.
- Los usuarios técnicos pueden realizar actualizaciones.

RF04 – Gestión de especies arbóreas (SRS)

- Se deben registrar especies indicando: nombre común, nombre científico, familia, altura promedio, diámetro promedio, y usos conocidos.
- Las especies se asociarán a una o más zonas.
- Debe permitirse adjuntar imágenes de cada especie.
- El sistema debe facilitar la búsqueda de especies por criterios taxonómicos.

RF05 – Registro y consulta de actividades de conservación (SRS)

- El usuario podrá registrar actividades indicando: nombre, tipo (reforestación, monitoreo, control de plagas, etc.), fecha, responsables, y zona forestal.
- Se podrá consultar por fecha, tipo y zona.
- Las actividades deben poder relacionarse con las especies afectadas.
- Debe existir un sistema de seguimiento del estado de cada actividad.

RF06 – Reportes exportables (SRS)

- El sistema debe permitir generar reportes en formato PDF y Excel de las zonas, especies y actividades registradas.
- Los reportes deben ser configurables, permitiendo seleccionar qué campos incluir.
- Se deben poder generar reportes estadísticos básicos (número de especies por zona, actividades por periodo, etc.)

2.3.3 Requisitos de rendimiento (SRS)

- **RNF01 - Rendimiento:**
  - Las respuestas a las consultas no deben superar los 3 segundos con hasta 1000 registros visibles.
  - El sistema debe soportar al menos 50 usuarios concurrentes sin degradación notable del servicio.
  - La carga inicial de la aplicación no debe superar los 5 segundos en una conexión estándar.
- **Carga de datos:**
  - El sistema debe ser capaz de almacenar información sobre al menos 10,000 zonas forestales.
  - Debe poder manejar un catálogo de 5,000 especies arbóreas.
  - La base de datos debe optimizarse para manejar eficientemente consultas complejas con múltiples criterios de filtrado.

2.3.4 Restricciones de diseño (SRS)

- El desarrollo debe seguir el patrón de arquitectura MVC.
- Se debe implementar el patrón DAO para el acceso a datos.
- El sistema debe utilizar un sistema de inyección de dependencias.
- La capa de presentación debe estar claramente separada de la lógica de negocio.
- Se debe utilizar JPA como tecnología de persistencia.
- El diseño de la base de datos debe seguir al menos la tercera forma normal.

2.3.5 Atributos del sistema (SRS)

Seguridad (SRS)

- **RNF02 - Seguridad:**
  - Las comunicaciones deben ser cifradas mediante HTTPS.
  - Deben implementarse mecanismos contra ataques de inyección SQL y XSS.
  - Se deben aplicar principios de desarrollo seguro para proteger la integridad de los datos.

Usabilidad (SRS)

- **RNF03 - Usabilidad:**
  - La interfaz debe ser intuitiva y accesible, con formularios validados y mensajes de error claros.
  - Compatible con los navegadores modernos (Chrome, Firefox, Edge).
  - El sistema debe ser accesible según las pautas WCAG 2.1 nivel AA.
  - Se deben proporcionar ayudas contextuales en los formularios complejos.

Mantenibilidad y Portabilidad (SRS)

- **RNF04 - Escalabilidad:**
  - El sistema debe poder escalar horizontalmente para atender múltiples solicitudes simultáneas.
- **RNF05 - Mantenibilidad:**
  - El código debe estar documentado y seguir patrones de diseño MVC y DAO.
  - Se debe mantener un conjunto de pruebas unitarias con al menos 70% de cobertura.
- **RNF06 - Disponibilidad:**
  - El sistema debe estar disponible al menos el 95% del tiempo durante el horario laboral.
- **RNF07 - Compatibilidad:**
  - Compatible con sistemas operativos Windows, Linux y MacOS en su cliente (navegador).
- **RNF08 - Internacionalización:**
  - Aunque inicialmente se usará en español, debe prepararse para futuras traducciones.
- **RNF09 - Respaldo de datos:**
  - Debe realizarse respaldo automático de la base de datos cada 24 horas.

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

Apéndice A: Diagrama de entidades (SRS)

2.4.2 Apéndice B: Mockups de interfaz (SRS)

Se incluirán diseños preliminares de las principales interfaces:

- Panel de control
- Formulario de registro de zona forestal
- Vista de consulta de especies
- Formulario de actividades de conservación

Apéndice B: Mockups de interfaz (SRS)

2.4.3 Apéndice C: Glosario de términos forestales (SRS)

Se incluirá un glosario con terminología específica del ámbito forestal relevante para la comprensión del sistema.

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.

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.

- **Ubicación en el proyecto:** Carpeta `Web Pages/` (archivos `.jsp` y `.html`)
- **Función:** Interfaz con la que el usuario interactúa directamente. Se encarga de mostrar datos y capturar entradas.

- **Tecnologías:** JSP, HTML, CSS, JavaScript (muy poco MODAL)
- `index.jsp`: Página de inicio del sistema.
- `TreeSpecies.jsp`, `TreeSpeciesFrm.jsp`: Interfaces para visualizar y registrar especies de árboles.
- `Zones.jsp`, `ZonesFrm.jsp`: Interfaces para visualizar y registrar zonas geográficas.
- `footer.jsp`, `menu.jsp`, `index.html`: Componentes reutilizables de la interfaz como el menú y pie de página.

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`: Gestiona solicitudes para actividades de conservación.
- `TreeSpeciesController.java`: Gestiona solicitudes para especies de árboles.
- `ZonesController.java`: Gestiona solicitudes para zonas geográficas.

Capa de Lógica de Negocio (Business Logic Layer / Service Layer): Contiene la lógica central de la aplicación, implementa reglas de negocio, válida procesos y orquesta la interacción entre controladores y acceso a datos.

- **Ubicación en el proyecto:** `com.espe.sistemaregistroforestal.service` (Package Service)
- **Función:** Contiene la lógica principal del sistema, como reglas de validación, procesos de negocio y coordinación de acceso a datos.
- **Tecnologías:** Java
- `ConservationActivitiesService.java`: Implementa la lógica relacionada con actividades de conservación.
- `TreeSpeciesService.java`: Contiene la lógica de gestión de especies de árboles.
- `ZonesService.java`: Procesa la lógica relacionada con zonas.

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 con la base de datos.
- `ConservationActivitiesDAO.java`: Acceso a los datos de actividades de conservación.
- `TreeSpeciesDAO.java`: Acceso a los datos de especies de árboles.
- `ZonesDAO.java`: Acceso a los datos de zonas.

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 para actividades de conservación.
- `TreeSpecies.java`: Modelo para especies de árboles.
- `Zones.java`: Modelo para zonas geográficas.
- `TipoActividad.java`, `TipoBosque.java`: Representan catálogos o tipos relacionados con las actividades o los tipos de bosques.

3.3 Diagrama de Arquitectura

Diagrama de Arquitectura del Sistema

3.4 Estrategia de Implementación

3.4.1 Desarrollo Modular y por Capas

Implementación separada de capas: Cada capa (Presentación, Control, Lógica de Negocio, Acceso a Datos, Dominio) se desarrollará de forma independiente para garantizar una clara separación de responsabilidades, facilitar el mantenimiento y permitir la reutilización de componentes. Interfaz bien definida entre capas: Se utilizarán

contratos claros (interfaces o APIs) entre capas para permitir cambios internos en una capa sin afectar 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 “Controlador” 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.

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.

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, 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.

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.

Modelo Entidad-Relación (ER) de la Base de Datos

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).

zones: Contiene información de zonas forestales protegidas.

Campo	Tipo	Descripción
id	int(11)	Identificador único de la zona (PK, autoincremental).

Campo	Tipo	Descripción
nombre	varchar(100)	Nombre de la zona protegida.
ubicacion	varchar(200)	Descripción de ubicación o coordenadas geográficas.
provincia	varchar(100)	Provincia donde se encuentra la zona.
tipo_bosque	enum	Tipo de bosque: Seco, Húmedo Tropical, Montano, Manglar, Otro
area_ha	decimal(10,2)	Área total de la zona en hectáreas.
descripcion	text	Descripción adicional de la zona.
fecha_registro	date	Fecha en que fue registrado el área.

tree_species: Contiene información de especies arbóreas con sus características.

Campo	Tipo	Descripción
id	int(11)	Identificador único de la especie (PK, autoincremental).
nombre_comun	varchar(100)	Nombre común de la especie.
nombre_cientifico	varchar(150)	Nombre científico.
familia_botanica	varchar(100)	Familia botánica a la que pertenece.
estado_conservacion	enum	Estado de conservación (ej: Vulnerable, En Peligro, etc.).
uso_principal	varchar(100)	Uso principal de la especie (madera, medicina, ornamental, etc.)
altura_maxima_m	decimal(5,2)	Altura máxima aproximada en metros.
zona_id	int(11)	Identificador de la zona donde se encuentra (FK).

conservation_zona: Tabla intermedia que relaciona actividades con múltiples zonas (relación muchos a muchos).

Campo	Tipo	Descripción
conservation_id	int(11)	ID de la actividad (FK).
zona_id	int(11)	ID de la zona (FK).

zona_especie: Tabla intermedia que relaciona zonas con especies (relación muchos a muchos).

Campo	Tipo	Descripción
especie_id	int(11)	ID de la especie (FK).
zona_id	int(11)	ID de la zona (FK).

Relaciones:

- Una zona puede tener muchas actividades de conservación (1 a muchos).
- Una actividad de conservación puede estar asociada a varias zonas a través de `conservation_zona` (muchos a muchos).

- Una zona puede contener muchas especies (muchos a muchos) mediante la tabla `zona_especie`.
- Cada especie está asociada a una única zona en la tabla `tree_species` pero además está relacionada con zonas en `zona_especie` para fines más específicos.

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) AUTO_INCREMENT PRIMARY KEY,
    nombre_actividad VARCHAR(150) NOT NULL,
    fecha_actividad DATE NOT NULL,
    responsable VARCHAR(150) NOT NULL,
    tipo_actividad ENUM('Reforestación', 'Monitoreo',
                        'Control', 'Otro') NOT NULL,
    descripcion TEXT,
    zona_id INT(11),
    activo TINYINT(1) DEFAULT 1,
    FOREIGN KEY (zona_id) REFERENCES zones(id)
);
```

Descripción: Esta tabla almacena las actividades de conservación forestal realizadas en las diferentes zonas protegidas, incluyendo detalles como tipo, fecha y responsables.

Tabla: zones

```
CREATE TABLE zones (
    id INT(11) AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    ubicacion VARCHAR(200),
    provincia VARCHAR(100),
    tipo_bosque ENUM('Seco', 'Húmedo Tropical', 'Montano',
                     'Manglar', 'Otro') NOT NULL,
    area_ha DECIMAL(10,2),
    descripcion TEXT,
    fecha_registro DATE
);
```

Descripción: Define las zonas o áreas protegidas donde se ejecutan actividades de conservación y habitan especies forestales.

Tabla: tree_species

```
CREATE TABLE tree_species (
    id INT(11) AUTO_INCREMENT PRIMARY KEY,
    nombre_comun VARCHAR(100) NOT NULL,
    nombre_cientifico VARCHAR(150),
    familia_botanica VARCHAR(100),
    estado_conservacion ENUM('Vulnerable', 'En Peligro',
                            'Extinto', 'Sin Riesgo') NOT NULL,
    uso_principal VARCHAR(100),
    altura_maxima_m DECIMAL(5,2),
    zona_id INT(11),
    FOREIGN KEY (zona_id) REFERENCES zones(id)
);
```

Descripción: Contiene información detallada sobre las especies arbóreas que habitan en las zonas protegidas.

Tabla: conservation_zona

```
CREATE TABLE conservation_zona (
    conservation_id INT(11),
    zona_id INT(11),
    PRIMARY KEY (conservation_id, zona_id),
    FOREIGN KEY (conservation_id)
        REFERENCES conservation_activities(id),
    FOREIGN KEY (zona_id) REFERENCES zones(id)
);
```

Descripción: Tabla intermedia que relaciona actividades de conservación con múltiples zonas, estableciendo una relación de muchos a muchos.

Tabla: zona_especie

```
CREATE TABLE zona_especie (
    zona_id INT(11),
    especie_id INT(11),
    PRIMARY KEY (zona_id, especie_id),
    FOREIGN KEY (zona_id) REFERENCES zones(id),
    FOREIGN KEY (especie_id) REFERENCES tree_species(id)
);
```

Descripción: Tabla intermedia que relaciona zonas con especies arbóreas, permitiendo una relación muchos a muchos.

4.3.3 Esquema Simplificado de Tablas

5 DOCUMENTACIÓN DE PRUEBAS

Sistema: SistemaRegistroForestal
Motor de BD: MySQL
Objetivo: Validar que las operaciones CRUD y las relaciones entre tablas funcionen correctamente a nivel de base de datos.

ID Caso	Descripción	Datos de Entrada / Pasos	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

6 MANUAL DE DESARROLLADOR

6.1 Introducción

Este manual está diseñado para facilitar a los desarrolladores la comprensión, mantenimiento, mejora y ampliación del Sistema de Registro Forestal. El sistema está construido bajo una arquitectura de tipo N-Capas, lo que permite separar las responsabilidades y hacer el código más mantenible y escalable.

El backend se desarrolla con Java EE, utilizando Servlets para la lógica del servidor y acceso a datos mediante JDBC. La base de datos relacional empleada es MySQL, gestionada localmente mediante XAMPP. El frontend se implementa con tecnologías web clásicas como JSP, HTML, CSS, y JavaScript, usando Bootstrap para asegurar una interfaz adaptativa y amigable en distintos dispositivos.

Este documento describe la estructura del proyecto, flujo general del sistema, recomendaciones para el desarrollo y posibles extensiones futuras.

6.2 Tecnologías Utilizadas

Backend:

- Java EE para la capa de negocio y controladores.
- Servlets para el manejo de peticiones HTTP.
- JDBC para la conexión y operaciones con la base de datos MySQL.

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, que facilita la administración del servidor de bases de datos en entornos de desarrollo.

Entorno de Desarrollo:

- NetBeans IDE, plataforma recomendada por su integración con Java EE y soporte para JSP.

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.

- `index.jsp` — Página principal, punto de entrada del sistema.
- `Zones.jsp` — Página para la gestión y visualización de zonas forestales.
- `TreeSpecies.jsp` — Página para administrar las especies de árboles registradas.
- `ConservationActivities.jsp` — Página para registrar y consultar actividades de conservación forestal.
- `menu.jsp` — Componente común que contiene el menú de navegación para todo el sistema.
- `footer.jsp` — Componente común que incluye el pie de página con información general.

`/com/espe/sistemaregistroforestal/controller/` **(Package Controller)** Paquete encargado de manejar las peticiones de los usuarios, Controladores que gestionan las solicitudes del cliente y comunican con la capa de servicios.

- `ZonesController.java` — Controlador para gestionar acciones relacionadas con zonas forestales (crear, actualizar, listar, eliminar).
- `TreeSpeciesController.java` — Controlador para manejar la administración de especies de árboles.
- `ConservationActivitiesController.java` — Controlador que gestiona el registro y mantenimiento de actividades de conservación.

`/com/espe/sistemaregistroforestal/service/` **(Package Service)** Aquí se implementa la lógica de negocio. La capa de servicio procesa la información recibida del controlador, ejecuta las reglas y llama a la capa DAO para interactuar con la base de datos.

- `ZonesService.java` — Implementa las reglas y operaciones relacionadas con las zonas forestales.
- `TreeSpeciesService.java` — Procesa la información y lógica para el manejo de especies arbóreas.
- `ConservationActivitiesService.java` — Maneja la lógica de las actividades de conservación, incluyendo validaciones específicas.

`/com/espe/sistemaregistroforestal/dao/` **(Package DAO)** La capa DAO (Data Access Object) es responsable de realizar todas las operaciones de persistencia con la base de datos, usando JDBC.

- `ZonesDAO.java` — Clase que ejecuta las operaciones SQL para la tabla de zonas.

- `TreeSpeciesDAO.java` — Encargada de las consultas y modificaciones sobre la tabla de especies de árboles.
- `ConservationActivitiesDAO.java` — Implementa las operaciones de persistencia para las actividades de conservación.
- `ConnectionBdd.java` — Clase utilitaria que gestiona la apertura y cierre de conexiones a la base de datos, garantizando el correcto manejo de recursos y evitando fugas.

`/com/espe/sistemaregistroforestal/model/` **(Package Model)** Este paquete contiene las clases que representan las entidades del negocio, conocidas como modelos o POJOs. Estas clases definen los atributos de cada entidad y métodos de acceso para manipular sus datos. No incluyen lógica de negocio ni acceso a la base de datos.

- `ConservationActivities.java` — Modelo que representa una actividad de conservación forestal.
- `TreeSpecies.java` — Modelo para las especies de árboles registradas.
- `Zones.java` — Modelo para las zonas o áreas geográficas de interés.
- `TipoActividad.java` — Enumeración que define los tipos de actividades disponibles.
- `TipoBosque.java` — Enumeración que clasifica los tipos de bosques.

6.4 Flujo General de Operación

6.4.1 Interacción del usuario con la interfaz (Frontend)

El usuario accede a la aplicación mediante 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 `menu.jsp` y `footer.jsp`.
- Validaciones iniciales pueden realizarse en el navegador usando JavaScript para mejorar la experiencia.

6.4.2 Recepción de la petición por el Controlador (Controller)

Cuando el usuario envía una solicitud (ejemplo: agregar una nueva zona forestal), esta se envía al servlet correspondiente dentro del paquete `/controller`.

- El controlador (`ZonesController.java`, `TreeSpeciesController.java`, etc.) recibe la petición HTTP, procesa parámetros y realiza validaciones preliminares para asegurar datos correctos.
- Una vez validada, el controlador invoca el servicio correspondiente, pasando la información necesaria para la operación.

6.4.3 Ejecución de la lógica de negocio en la capa Servicio (Service)

La capa de servicios (`ZonesService.java`, `TreeSpeciesService.java`, etc.) recibe la petición desde el controlador.

- Aquí se ejecutan las reglas de negocio, validaciones complejas y procesos específicos, como validación de formatos, cálculos o decisiones condicionales.
- Luego, el servicio llama a la capa DAO para realizar la persistencia o recuperación de datos desde la base de datos.

6.4.4 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.

6.4.5 Respuesta hacia el usuario

La información procesada y resultado de la operación (éxito, error, datos solicitados) se transmite desde DAO hacia la capa servicio y de allí al controlador.

- 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.

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

- **Gestión de usuarios y roles:** Implementar un módulo de autenticación y autorización para controlar el acceso según perfiles de usuario.
- **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.