Elaboración de historias de usuario del proyecto. GA2-220501093-AA1-EV03

Autor:

Juan Pablo Collante Dominguez
C.C. 1019012461

Análisis y Desarrollo de Software

Fase de Análisis

Tecnología

SENA

Agosto 2024

Especificación de Requisitos de Software (SRS)

Sistema de Información de una Biblioteca

1. Introducción

1.1 Propósito

El propósito de este documento es definir los requisitos funcionales y no funcionales del Sistema de Información de una Biblioteca. Este sistema permitirá a los usuarios realizar préstamos y devoluciones de libros, y a los bibliotecarios gestionar el inventario de la biblioteca, entre otras funcionalidades.

1.2 Alcance

El Sistema de Información de la Biblioteca proporcionará una plataforma para la gestión eficiente de préstamos y devoluciones de libros, el registro de nuevos usuarios, y la administración del inventario de libros. Este sistema será utilizado por usuarios y bibliotecarios.

1.3 Definiciones, Acrónimos y Abreviaturas

- **Usuario**: Persona que utiliza el sistema para realizar préstamos y devoluciones de libros.
- **Bibliotecario**: Persona encargada de la gestión del inventario y registro de usuarios en la biblioteca.
- **SRS**: Software Requirements Specification (Especificación de Requisitos de Software).

2. Descripción General

2.1 Perspectiva del Producto

El sistema será una aplicación web accesible desde cualquier dispositivo con conexión a Internet. Proporcionará una interfaz amigable para usuarios y bibliotecarios, facilitando las tareas de préstamos, devoluciones y gestión del inventario de libros.

2.2 Funcionalidades del Producto

- Registro y gestión de usuarios.
- Gestión de libros (registro, actualización, baja).
- Préstamos y devoluciones de libros.
- Consulta del catálogo de la biblioteca.
- Notificaciones a usuarios sobre el estado de sus préstamos y reservas.

2.3 Características de los Usuarios

- **Usuarios**: Personas que desean prestar o devolver libros de la biblioteca.
- **Bibliotecarios**: Personas encargadas de gestionar el inventario y registrar nuevos usuarios.

2.4 Restricciones

- El sistema debe cumplir con las políticas de privacidad y protección de datos personales.
- El sistema debe ser accesible y usable para usuarios con diferentes niveles de habilidad técnica.

3. Requisitos Específicos

3.1 Historias de Usuario

Historia de Usuario 1

Identificador (ID): BIB-0001

Rol: Como un Usuario

Funcionalidad: Necesito realizar préstamos de libros

Razón / Resultado: Con la finalidad de poder leer los libros fuera de la biblioteca

Criterios de Aceptación:

1. **Escenario 1**: Libro disponible para préstamo
    - **Contexto**: En caso de que el libro esté disponible
    - **Evento**: Cuando el usuario solicita el préstamo de un libro
    - **Resultado**: El sistema confirmará el préstamo y actualizará el estado del libro como "prestado"
2. **Escenario 2**: Libro no disponible para préstamo
    - **Contexto**: En caso de que el libro no esté disponible
    - **Evento**: Cuando el usuario solicita el préstamo de un libro
    - **Resultado**: El sistema informará al usuario que el libro no está disponible y ofrecerá la opción de reservarlo

      Puntos Estimados de Esfuerzo: 8 puntos

Historia de Usuario 2

Identificador (ID): BIB-0002

Rol: Como un Usuario

Funcionalidad: Necesito devolver libros

Razón / Resultado: Con la finalidad de liberar el préstamo y hacer disponible el libro para otros usuarios

Criterios de Aceptación:

1. **Escenario 1**: Devolución de libro
    - **Contexto**: En caso de que el usuario quiera devolver un libro
    - **Evento**: Cuando el usuario devuelve un libro
    - **Resultado**: El sistema actualizará el estado del libro como "disponible" y registrará la devolución en el historial del usuario

      Puntos Estimados de Esfuerzo: 5 puntos

Historia de Usuario 3

Identificador (ID): BIB-0003

Rol: Como un Bibliotecario

Funcionalidad: Necesito llevar un registro de los libros disponibles en la biblioteca

Razón / Resultado: Con la finalidad de tener un inventario actualizado de los recursos disponibles

Criterios de Aceptación:

1. **Escenario 1**: Registro de nuevos libros
    - **Contexto**: En caso de la adquisición de nuevos libros
    - **Evento**: Cuando se registra un nuevo libro en el sistema
    - **Resultado**: El sistema agregará el libro al inventario y estará disponible para préstamo o consulta
2. **Escenario 2**: Baja de libros
    - **Contexto**: En caso de que un libro se dé de baja
    - **Evento**: Cuando se da de baja un libro
    - **Resultado**: El sistema actualizará el inventario y el libro ya no estará disponible para préstamo o consulta

      Puntos Estimados de Esfuerzo: 13 puntos

Historia de Usuario 4

Identificador (ID): BIB-0004

Rol: Como un Bibliotecario

Funcionalidad: Necesito llevar un registro del estado de las nuevas adquisiciones de libros

Razón / Resultado: Con la finalidad de gestionar y catalogar las nuevas adquisiciones eficientemente

Criterios de Aceptación:

1. **Escenario 1**: Estado de nuevas adquisiciones
    - **Contexto**: En caso de adquisición de nuevos libros
    - **Evento**: Cuando se adquieren nuevos libros
    - **Resultado**: El sistema permitirá registrar el estado de catalogación y disponibilidad de los nuevos libros

      Puntos Estimados de Esfuerzo: 8 puntos

Historia de Usuario 5

Identificador (ID): BIB-0005

Rol: Como un Bibliotecario

Funcionalidad: Necesito llevar un registro de los usuarios de la biblioteca

Razón / Resultado: Con la finalidad de gestionar los préstamos y devoluciones adecuadamente

Criterios de Aceptación:

1. **Escenario 1**: Registro de nuevos usuarios
    - **Contexto**: En caso de nuevos usuarios
    - **Evento**: Cuando un usuario se registra en la biblioteca
    - **Resultado**: El sistema permitirá registrar y almacenar los datos del usuario
2. **Escenario 2**: Actualización de datos de usuarios
    - **Contexto**: En caso de cambios en la información del usuario
    - **Evento**: Cuando se actualizan los datos del usuario
    - **Resultado**: El sistema permitirá modificar y almacenar los datos actualizados del usuario

      Puntos Estimados de Esfuerzo: 8 puntos

Historia de Usuario 6

Identificador (ID): BIB-0006

Rol: Como un Usuario

Funcionalidad: Necesito saber qué libros son de solo consulta y no de préstamo

Razón / Resultado: Con la finalidad de no solicitar libros que no se pueden prestar

Criterios de Aceptación:

1. **Escenario 1**: Identificación de libros de consulta
    - **Contexto**: En caso de búsqueda de libros
    - **Evento**: Cuando se consulta el catálogo de la biblioteca
    - **Resultado**: El sistema mostrará claramente los libros que son de solo consulta

      Puntos Estimados de Esfuerzo: 5 puntos

3.2 Requerimientos Funcionales

1. **Registro de Usuarios**: El sistema permitirá el registro, actualización y eliminación de usuarios.
2. **Gestión de Libros**: El sistema permitirá registrar nuevos libros, actualizar información de los libros existentes, dar de baja libros y marcar libros como disponibles o prestados.
3. **Préstamos y Devoluciones**: El sistema permitirá a los usuarios solicitar préstamos de libros y registrar la devolución de los mismos.
4. **Consulta de Catálogo**: El sistema permitirá a los usuarios consultar el catálogo de libros, identificando claramente aquellos que son de solo consulta.
5. **Inventario de Libros**: El sistema mantendrá un inventario actualizado de todos los libros disponibles y no disponibles.
6. **Notificaciones**: El sistema enviará notificaciones a los usuarios sobre el estado de sus préstamos y reservas.

3.3 Requerimientos No Funcionales

1. **Usabilidad**: El sistema debe ser fácil de usar y accesible para usuarios de todas las edades y niveles de habilidad técnica.
2. **Rendimiento**: El sistema debe ser capaz de manejar múltiples solicitudes simultáneas sin degradación del rendimiento.
3. **Seguridad**: El sistema debe garantizar la protección de los datos personales de los usuarios y la integridad de la información del catálogo de libros.

4. **Disponibilidad**: El sistema debe estar disponible al menos el 99.9% del tiempo para asegurar que los usuarios puedan acceder a sus funcionalidades en cualquier momento.
5. **Escalabilidad**: El sistema debe ser escalable para manejar un creciente número de usuarios y libros en el inventario.
6. **Mantenimiento**: El sistema debe ser fácil de mantener y actualizar sin interrumpir el servicio a los usuarios.

Diagramas y documentación (artefactos UML). GA2-220501093-AA1-EV04

Autor:

Juan Pablo Collante Dominguez
C.C. 1019012461

Análisis y Desarrollo de Software

Fase de Análisis

Tecnología

SENA

Agosto 2024

Tipos de Diagramas para Modelar el Software

Para modelar el software de la gestión bibliotecaria, se pueden utilizar diversos tipos de diagramas UML, que incluyen:

1. **Diagrama de Casos de Uso**: Describe las interacciones entre los actores (usuarios o sistemas externos) y el sistema. Es útil para mostrar las funcionalidades que el sistema debe ofrecer según las historias de usuario.
2. **Diagrama de Clases**: Muestra la estructura estática del sistema, incluyendo las clases, sus atributos, métodos, y las relaciones entre ellas. Es esencial para definir cómo se organizarán los datos y las funciones en el sistema.
3. **Diagrama de Secuencia**: Modela el flujo de mensajes entre objetos para llevar a cabo una funcionalidad particular. Es útil para describir la dinámica del sistema y cómo interactúan los componentes durante un proceso específico, como un préstamo o la adquisición de un libro.
4. **Diagrama de Actividades**: Representa el flujo de trabajo o actividades dentro de un proceso en el sistema. Es útil para entender las diferentes etapas por las que pasa un proceso, como la adquisición y catalogación de nuevos libros.
5. **Diagrama de Estados**: Modela los diferentes estados por los que pasa un objeto en el sistema, como el estado de un libro (disponible, prestado, en reserva).
6. **Diagrama de Componentes**: Muestra la organización y dependencia entre componentes del sistema. Es útil para comprender la arquitectura del software.
7. **Diagrama de Despliegue**: Muestra la configuración física del hardware en la que se ejecuta el sistema, junto con la distribución del software. Esto es importante para la planificación de la infraestructura de TI.

Diagramas UML de los Artefactos del Sistema

1. **Diagrama de Casos de Uso (ver anexo 1)**
    - **Actores**: Usuario, Bibliotecario
    - **Casos de Uso**:
        - Realizar préstamos
        - Devolver libros

        - Consultar catálogo
        - Registrar nuevos libros
        - Gestionar adquisiciones
        - Registrar usuarios
2. **Diagrama de Clases (ver anexo 2)**
    - **Clases**:
        - **Usuario**: nombre, email, historialDePrestamos()
        - **Libro**: titulo, autor, estado, prestar(), devolver()
        - **Catalogo**: libros, buscarLibro()
        - **Adquisicion**: titulo, estado, actualizarEstado()
        - **Bibliotecario**: nombre, registrarLibro(), registrarAdquisicion()
3. **Diagrama de Secuencia (ver anexo 3)**
    - **Ejemplo**: Secuencia de registrar un libro adquirido
        - Bibliotecario ingresa información del libro en el sistema.
        - El sistema verifica y actualiza el estado de la adquisición.
        - El sistema confirma la disponibilidad del libro en el catálogo.
4. **Diagrama de Actividades**
    - **Ejemplo**: Proceso de adquisición de un libro
        - Iniciar adquisición
        - Registrar detalles de la compra
        - Estado: Pendiente por enviar
        - Estado: En camino
        - Estado: En proceso de inventario
        - Estado: Catalogado
5. **Modelo de dominio (Ver anexo 4)**

    Clases y Relaciones:

    Usuario:
    - **Atributos:** ID_Usuario, Nombre, Email, Dirección, Teléfono, Fecha_Registro.
    - **Métodos:**
        - solicitarPréstamo(libro: Libro): Préstamo
        - devolverLibro(libro: Libro): void

        - consultarHistorial(): List<Préstamo>

Libro:

- **Atributos:** ID_Libro, Título, Autor, ISBN, Año_Publicación, Editorial, Estado.
- **Métodos:**
    - prestar(usuario: Usuario): Préstamo
    - devolver(): void
    - actualizarEstado(estado: String): void

Préstamo:

- **Atributos:** ID_Préstamo, ID_Usuario, ID_Libro, Fecha_Préstamo, Fecha_Devolución, Estado.
- **Métodos:**
    - registrarPréstamo(): void
    - finalizarPréstamo(): void
    - verificarEstado(): String

Adquisición:

- **Atributos:** ID_Adquisición, ID_Libro, Fecha_Adquisición, Proveedor, Precio, Estado_Adquisición.
- **Métodos:**
    - registrarAdquisición(): void
    - actualizarEstadoAdquisición(estado: String): void

Bibliotecario:

- **Atributos:** ID_Bibliotecario, Nombre, Email, Teléfono, Fecha_Contratación.
- **Métodos:**
    - gestionarAdquisición(libro: Libro): Adquisición
    - registrarLibro(libro: Libro): void

6. Esquema de Base de Datos para el Software

1. Tablas Principales

1. **Tabla Usuarios:**
    - **ID_Usuario** (PK): INT (autoincremental)

- Nombre: VARCHAR(255)
- Email: VARCHAR(255) (único)
- Dirección: VARCHAR(255)
- Teléfono: VARCHAR(20)
- Fecha_Registro: DATETIME

2. **Tabla Libros:**
   - **ID_Libro** (PK): INT (autoincremental)
   - Título: VARCHAR(255)
   - Autor: VARCHAR(255)
   - ISBN: VARCHAR(20) (único)
   - Año_Publicación: YEAR
   - Editorial: VARCHAR(255)
   - Estado: ENUM('Disponible', 'Prestado', 'En Reserva', 'No Disponible')
   - Fecha_Adquisición: DATETIME
3. **Tabla Préstamos:**
   - **ID_Préstamo** (PK): INT (autoincremental)
   - ID_Usuario (FK): INT (referencia a Usuarios.ID_Usuario)
   - ID_Libro (FK): INT (referencia a Libros.ID_Libro)
   - Fecha_Préstamo: DATETIME
   - Fecha_Devolución: DATETIME
   - Estado: ENUM('Activo', 'Completado', 'Retrasado')
4. **Tabla Adquisiciones:**
   - **ID_Adquisición** (PK): INT (autoincremental)
   - ID_Libro (FK): INT (referencia a Libros.ID_Libro)
   - Proveedor: VARCHAR(255)
   - Fecha_Adquisición: DATETIME
   - Precio: DECIMAL(10,2)
   - Estado_Adquisición: ENUM('Pendiente', 'Completada', 'Cancelada')
5. **Tabla Bibliotecarios:**
   - **ID_Bibliotecario** (PK): INT (autoincremental)

- Nombre: VARCHAR(255)
- Email: VARCHAR(255)
- Teléfono: VARCHAR(20)
- Fecha_Contratación: DATETIME

2. Relaciones entre las Tablas

- **Relación entre Usuarios y Préstamos:**
  - Un **Usuario** puede tener múltiples **Préstamos**.
  - Llave foránea: Préstamos.ID_Usuario referencia a Usuarios.ID_Usuario.
- **Relación entre Libros y Préstamos:**
  - Un **Libro** puede estar asociado a múltiples **Préstamos** a lo largo del tiempo, pero solo puede estar activo en un **Préstamo** a la vez.
  - Llave foránea: Préstamos.ID_Libro referencia a Libros.ID_Libro.
- **Relación entre Libros y Adquisiciones:**
  - Un **Libro** puede haber pasado por varias **Adquisiciones**.
  - Llave foránea: Adquisiciones.ID_Libro referencia a Libros.ID_Libro.
- **Relación entre Bibliotecarios y Adquisiciones:**
  - Un **Bibliotecario** gestiona las **Adquisiciones** de **Libros**.
  - Aunque no hay una tabla específica para la relación directa en este esquema, el Bibliotecario es responsable de registrar las adquisiciones.

Herramientas para Diagramar

- **Lucidchart**: Herramienta en línea para crear diagramas UML de forma colaborativa.
- **Draw.io**: Herramienta gratuita y de código abierto para diagramas UML.
- **Enterprise Architect**: Software profesional para modelado UML.
- **StarUML**: Herramienta de diseño UML potente y fácil de usar.
- **Visual Paradigm**: Herramienta que soporta múltiples lenguajes de modelado, incluido UML.

Documento de Plantilla de Caso de Uso

Nombre del Caso de Uso: Realizar Préstamos de Libros

- **ID**: UC-001
- **Descripción**: Permitir que un usuario pueda solicitar el préstamo de un libro disponible.
- **Actores**: Usuario
- **Precondiciones**:
    - El usuario debe estar registrado en el sistema.
    - El libro debe estar disponible en el catálogo.
- **Postcondiciones**:
    - El estado del libro se actualiza a "prestado".
    - Se registra la transacción en el historial del usuario.
- **Flujo Principal**:
    - El usuario accede al catálogo de libros.
    - El usuario selecciona un libro y solicita el préstamo.
    - El sistema verifica la disponibilidad del libro.
    - El sistema registra el préstamo y actualiza el estado del libro.
    - El usuario recibe una confirmación del préstamo.
- **Flujos Alternativos**:
    - **F1**: Si el libro no está disponible, el sistema ofrecerá la opción de reservarlo.

- **Requisitos Especiales**: La operación debe completarse en menos de 2 segundos.
- **Dependencias**: Este caso de uso depende del caso de uso de "Registro de Usuario".

Nombre del Caso de Uso: Gestionar Adquisiciones de Libros

- **ID**: UC-002
- **Descripción**: Permitir que un bibliotecario gestione el estado de las adquisiciones de libros.
- **Actores**: Bibliotecario
- **Precondiciones**:
  - El bibliotecario debe tener acceso al sistema.
- **Postcondiciones**:
  - El estado de la adquisición se actualiza en el sistema.
- **Flujo Principal**:
  - El bibliotecario ingresa al módulo de adquisiciones.
  - El bibliotecario registra una nueva adquisición.
  - El sistema asigna un estado inicial "Pendiente por enviar".
  - El bibliotecario actualiza el estado de la adquisición a medida que avanza el proceso.
  - El sistema actualiza el inventario cuando la adquisición está completa.
- **Flujos Alternativos**:
  - **F1**: Si el libro es rechazado, el bibliotecario registra la razón del rechazo.
- **Requisitos Especiales**: La interfaz debe ser intuitiva para minimizar errores de registro.
- **Dependencias**: Este caso de uso depende del caso de uso de "Registro de Libro".

Historias de Usuario para el Sistema de Información de una Biblioteca

Historia de Usuario 1

Identificador (ID): BIB-0001

Rol: Como un Usuario

Funcionalidad: Necesito realizar préstamos de libros
Razón / Resultado: Con la finalidad de poder leer los libros fuera de la biblioteca
Criterios de Aceptación:

1. **Escenario 1**: Libro disponible para préstamo
    - **Contexto**: En caso de que el libro esté disponible
    - **Evento**: Cuando el usuario solicita el préstamo de un libro
    - **Resultado**: El sistema confirmará el préstamo y actualizará el estado del libro como "prestado"
2. **Escenario 2**: Libro no disponible para préstamo
    - **Contexto**: En caso de que el libro no esté disponible
    - **Evento**: Cuando el usuario solicita el préstamo de un libro
    - **Resultado**: El sistema informará al usuario que el libro no está disponible y ofrecerá la opción de reservarlo
      Puntos Estimados de Esfuerzo: 8 puntos

Historia de Usuario 2
Identificador (ID): BIB-0002
Rol: Como un Usuario
Funcionalidad: Necesito devolver libros
Razón / Resultado: Con la finalidad de liberar el préstamo y hacer disponible el libro para otros usuarios
Criterios de Aceptación:

1. **Escenario 1**: Devolución de libro
    - **Contexto**: En caso de que el usuario quiera devolver un libro
    - **Evento**: Cuando el usuario devuelve un libro
    - **Resultado**: El sistema actualizará el estado del libro como "disponible" y registrará la devolución en el historial del usuario
      Puntos Estimados de Esfuerzo: 5 puntos

Historia de Usuario 3
Identificador (ID): BIB-0003
Rol: Como un Bibliotecario

Funcionalidad: Necesito llevar un registro de los libros disponibles en la biblioteca
Razón / Resultado: Con la finalidad de tener un inventario actualizado de los recursos disponibles
Criterios de Aceptación:

1. **Escenario 1**: Registro de nuevos libros
    - **Contexto**: En caso de la adquisición de nuevos libros
    - **Evento**: Cuando se registra un nuevo libro en el sistema
    - **Resultado**: El sistema agregará el libro al inventario y estará disponible para préstamo o consulta
2. **Escenario 2**: Baja de libros
    - **Contexto**: En caso de que un libro se dé de baja
    - **Evento**: Cuando se da de baja un libro
    - **Resultado**: El sistema actualizará el inventario y el libro ya no estará disponible para préstamo o consulta
      Puntos Estimados de Esfuerzo: 13 puntos

Historia de Usuario 4
Identificador (ID): BIB-0004
Rol: Como un Bibliotecario
Funcionalidad: Necesito llevar un registro del estado de las nuevas adquisiciones de libros
Razón / Resultado: Con la finalidad de gestionar y catalogar las nuevas adquisiciones eficientemente
Criterios de Aceptación:

1. **Escenario 1**: Estado de nuevas adquisiciones
    - **Contexto**: En caso de adquisición de nuevos libros
    - **Evento**: Cuando se adquieren nuevos libros
    - **Resultado**: El sistema permitirá registrar el estado de catalogación y disponibilidad de los nuevos libros
      Puntos Estimados de Esfuerzo: 8 puntos

Historia de Usuario 5
Identificador (ID): BIB-0005
Rol: Como un Bibliotecario

Funcionalidad: Necesito llevar un registro de los usuarios de la biblioteca
Razón / Resultado: Con la finalidad de gestionar los préstamos y devoluciones adecuadamente

Criterios de Aceptación:

1. **Escenario 1**: Registro de nuevos usuarios
    - **Contexto**: En caso de nuevos usuarios
    - **Evento**: Cuando un usuario se registra en la biblioteca
    - **Resultado**: El sistema permitirá registrar y almacenar los datos del usuario
2. **Escenario 2**: Actualización de datos de usuarios
    - **Contexto**: En caso de cambios en la información del usuario
    - **Evento**: Cuando se actualizan los datos del usuario
    - **Resultado**: El sistema permitirá modificar y almacenar los datos actualizados del usuario

    Puntos Estimados de Esfuerzo: 8 puntos

Historia de Usuario 6

Identificador (ID): BIB-0006
Rol: Como un Usuario
Funcionalidad: Necesito saber qué libros son de solo consulta y no de préstamo
Razón / Resultado: Con la finalidad de no solicitar libros que no se pueden prestar
Criterios de Aceptación:

1. **Escenario 1**: Identificación de libros de consulta
    - **Contexto**: En caso de búsqueda de libros
    - **Evento**: Cuando se consulta el catálogo de la biblioteca
    - **Resultado**: El sistema mostrará claramente los libros que son de solo consulta

    Puntos Estimados de Esfuerzo: 5 puntos

3.2 Requerimientos Funcionales

1. **Registro de Usuarios**: El sistema permitirá el registro, actualización y eliminación de usuarios.

2. **Gestión de Libros**: El sistema permitirá registrar nuevos libros, actualizar información de los libros existentes, dar de baja libros y marcar libros como disponibles o prestados.
3. **Préstamos y Devoluciones**: El sistema permitirá a los usuarios solicitar préstamos de libros y registrar la devolución de los mismos.
4. **Consulta de Catálogo**: El sistema permitirá a los usuarios consultar el catálogo de libros, identificando claramente aquellos que son de solo consulta.
5. **Inventario de Libros**: El sistema mantendrá un inventario actualizado de todos los libros disponibles y no disponibles.
6. **Notificaciones**: El sistema enviará notificaciones a los usuarios sobre el estado de sus préstamos y reservas.

3.3 Requerimientos No Funcionales

1. **Usabilidad**: El sistema debe ser fácil de usar y accesible para usuarios de todas las edades y niveles de habilidad técnica.
2. **Rendimiento**: El sistema debe ser capaz de manejar múltiples solicitudes simultáneas sin degradación del rendimiento.
3. **Seguridad**: El sistema debe garantizar la protección de los datos personales de los usuarios y la integridad de la información del catálogo de libros.
4. **Disponibilidad**: El sistema debe estar disponible al menos el 99.9% del tiempo para asegurar que los usuarios puedan acceder a sus funcionalidades en cualquier momento.
5. **Escalabilidad**: El sistema debe ser escalable para manejar un creciente número de usuarios y libros en el inventario.
6. **Mantenimiento**: El sistema debe ser fácil de mantener y actualizar sin interrumpir el servicio a los usuarios.

ANEXOS

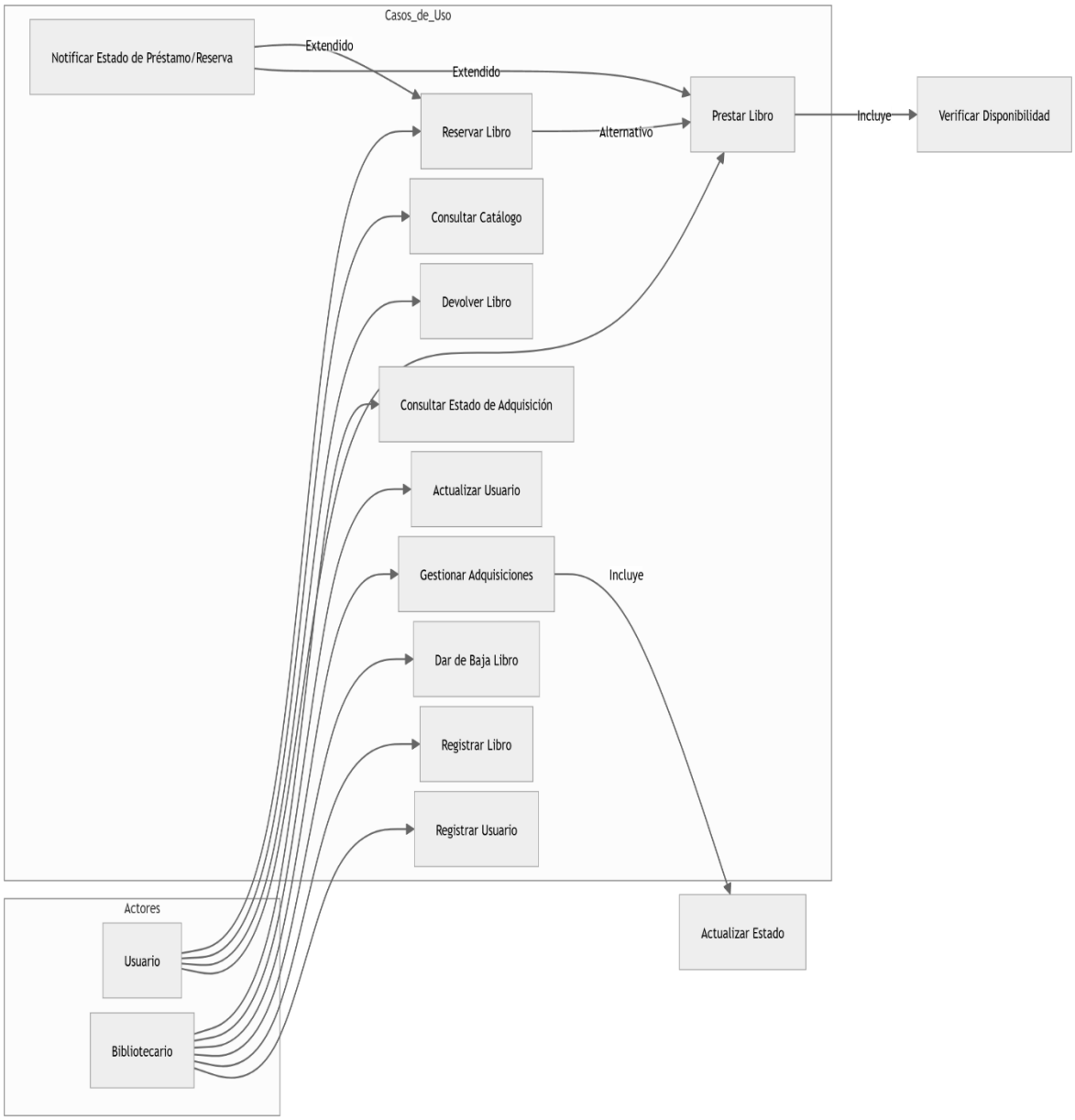


Ilustración 1: ANEXO1

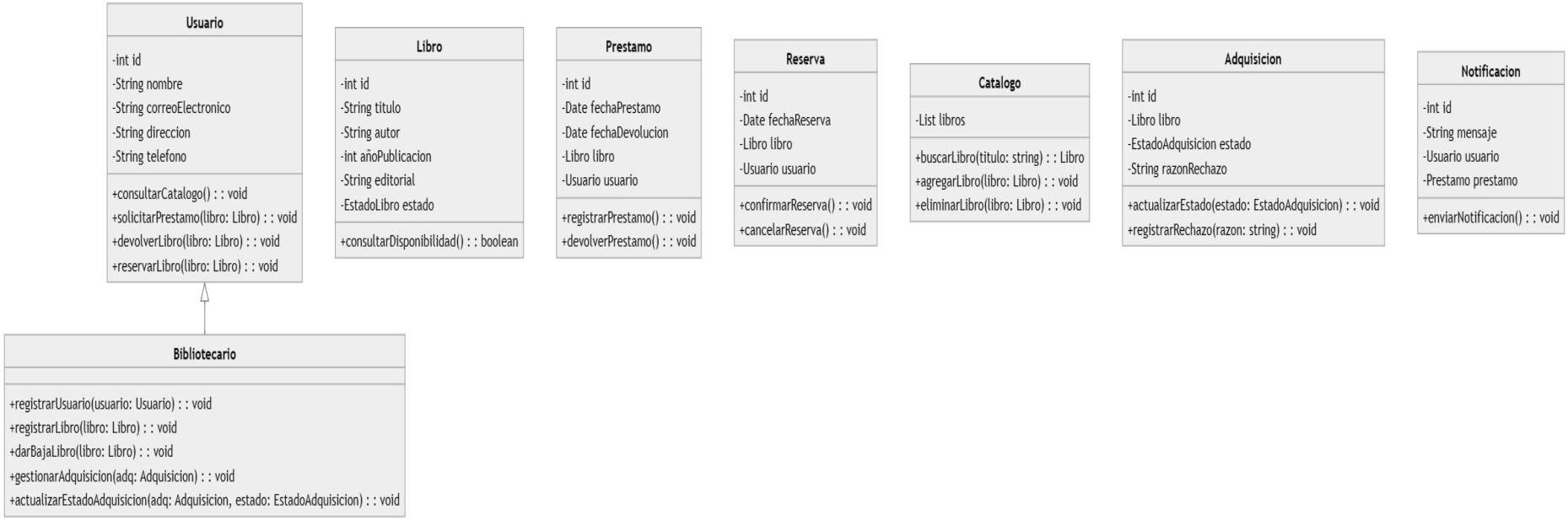


Ilustración 2: ANEXO 2

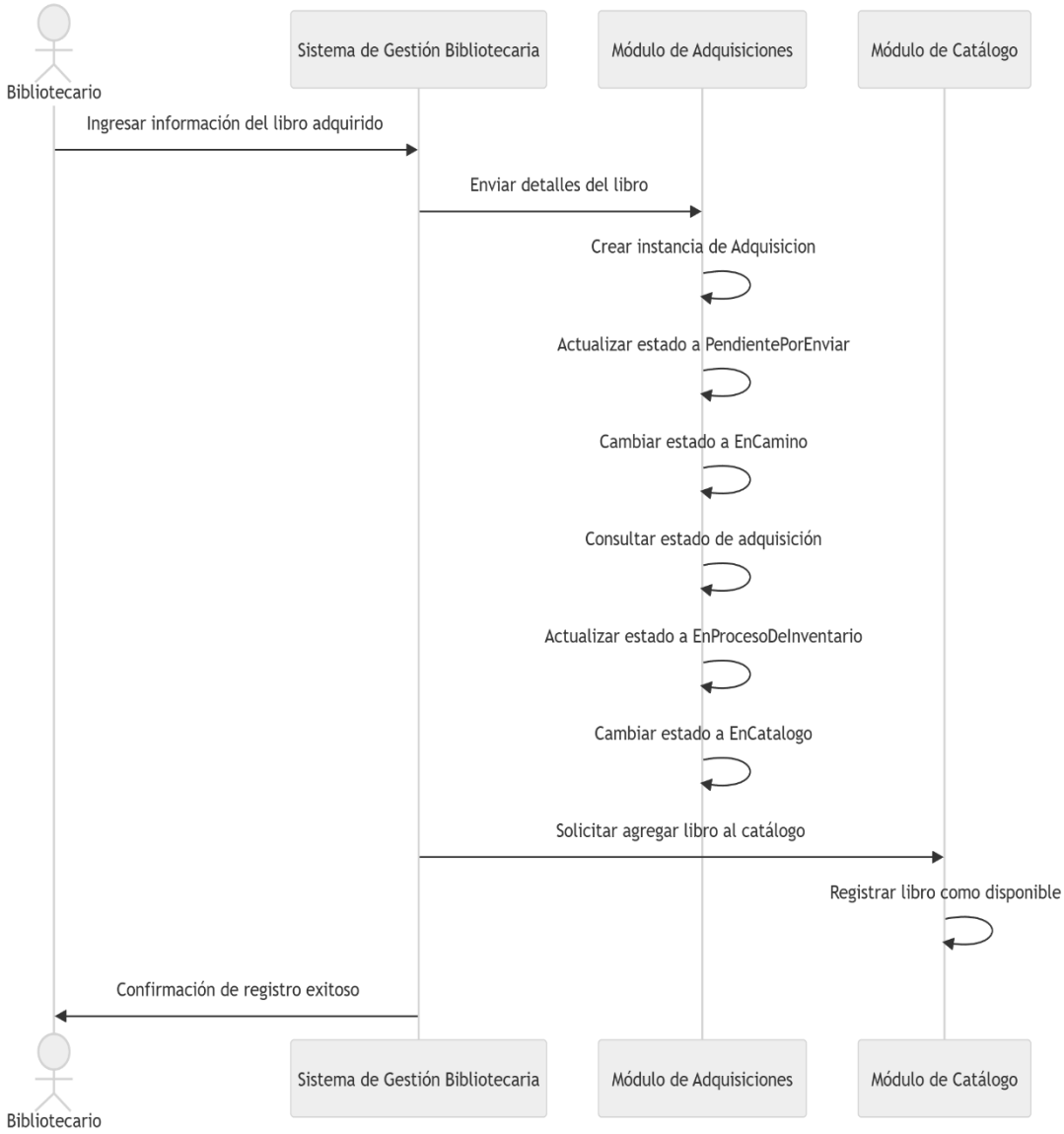


Ilustración 3: ANEXO 3A

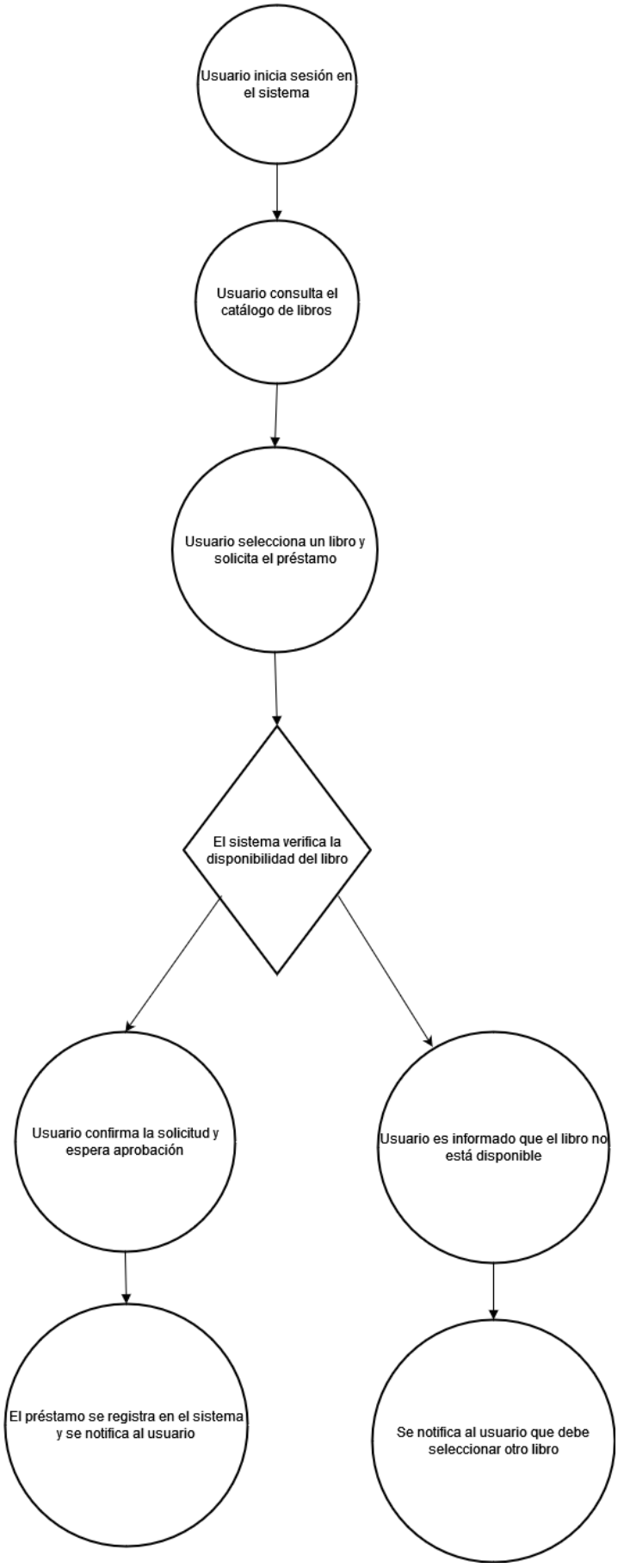


Ilustración 4: ANEXO 3B

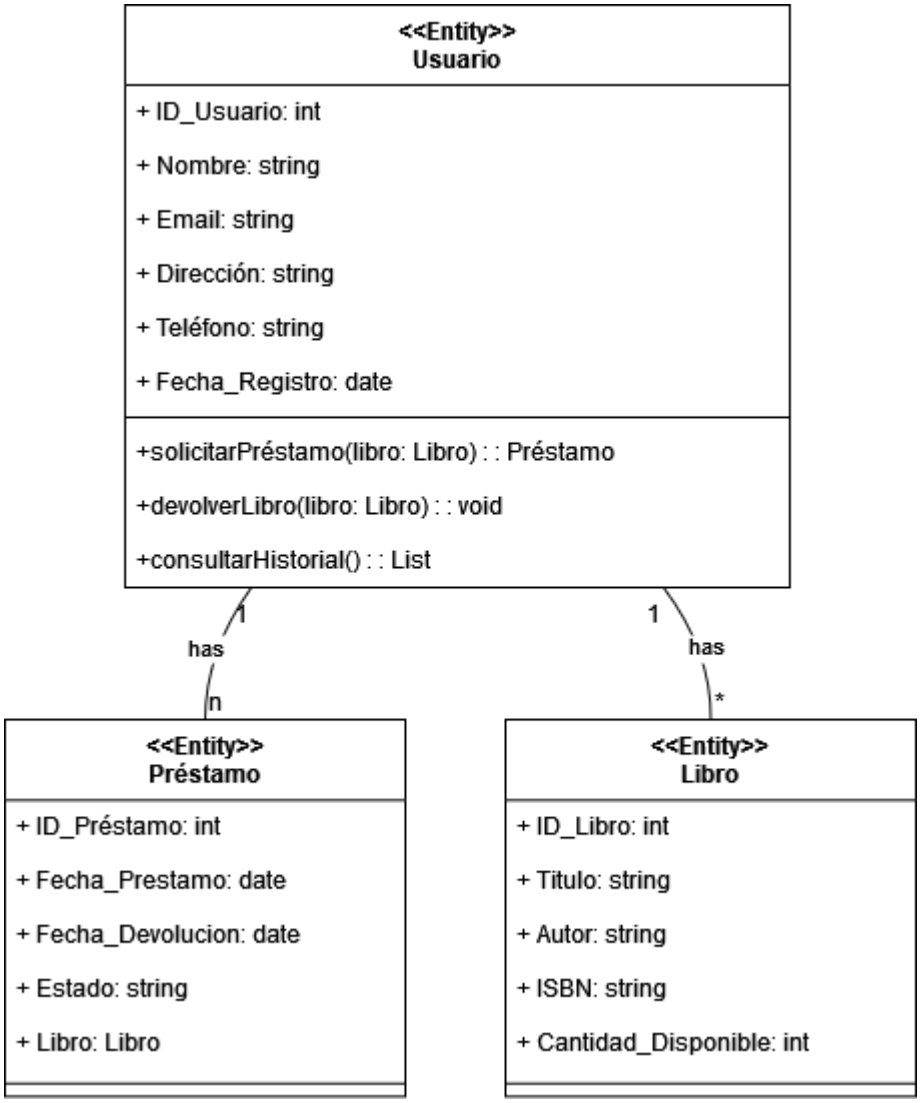


Ilustración 5: ANEXO 4

Elaboración de los diagramas del modelo de dominio del proyecto. GA2-220501093-AA2-EV01

Autor:

Juan Pablo Collante Dominguez
C.C. 1019012461

Análisis y Desarrollo de Software

Fase de Análisis

Tecnología

SENA

Septiembre 2024

Elaboración de los diagramas del modelo de dominio del proyecto

Para representar el negocio del proyecto de gestión bibliotecaria en términos de clases abstractas y generar un modelo de dominio consistente, debemos tomar los elementos clave del documento y organizarlos en un diagrama de clases que refleje la estructura del sistema. A continuación, te presento una descripción del modelo de dominio y las consideraciones necesarias para su elaboración.

1. Identificación de Clases Principales

Basado en el documento, las principales clases que representan el negocio son:

- **Usuario**: Representa a los usuarios del sistema, tanto aquellos que solicitan préstamos como los bibliotecarios.
- **Libro**: Representa los libros disponibles en la biblioteca.
- **Bibliotecario**: Un tipo especializado de usuario, responsable de gestionar adquisiciones y registros.
- **Préstamo**: Representa las transacciones de préstamos de libros a los usuarios.
- **Adquisición**: Representa el proceso de adquisición de nuevos libros por parte de la biblioteca.

2. Relaciones y Cardinalidad

Las relaciones y cardinalidades deben reflejar la interacción entre las clases:

- **Usuario – Préstamo**: Un usuario puede realizar varios préstamos, y cada préstamo está asociado a un único usuario. (Relación uno a muchos).
- **Libro – Préstamo**: Un libro puede estar relacionado con múltiples préstamos a lo largo del tiempo, pero solo puede estar activo en un préstamo a la vez. (Relación uno a muchos).
- **Bibliotecario – Adquisición**: Un bibliotecario puede gestionar varias adquisiciones, y cada adquisición es gestionada por un bibliotecario. (Relación uno a muchos).

- **Libro – Adquisición**: Un libro puede estar asociado a varias adquisiciones, y cada adquisición está relacionada con un único libro. (Relación uno a muchos).

3. Herencia

La clase **Usuario** tiene una subclase, **Bibliotecario**, que hereda sus atributos básicos (como nombre, email, etc.) y agrega métodos específicos para gestionar libros y adquisiciones.

4. Componentes y Paquetes

Para organizar el sistema de manera lógica, los componentes del sistema pueden dividirse en paquetes, como se indica a continuación:

- **Paquete "Usuarios"**: Contendrá las clases **Usuario** y **Bibliotecario**.
- **Paquete "Transacciones"**: Incluirá las clases **Préstamo** y **Adquisición**, que son operaciones clave del sistema.
- **Paquete "Catálogo"**: Contendrá la clase **Libro** y las operaciones relacionadas con la gestión de libros (prestados, adquiridos, disponibles, etc.).

5. Métodos y Atributos Clave

- **Usuario**
    - Atributos: nombre, email, direccion, telefono, fechaRegistro.
    - Métodos: solicitarPrestamo(), devolverLibro(), consultarHistorial().
- **Bibliotecario** (hereda de Usuario)
    - Métodos: gestionarAdquisicion(), registrarLibro().
- **Libro**
    - Atributos: titulo, autor, estado, ISBN, editorial, añoPublicacion.
    - Métodos: prestar(), devolver(), actualizarEstado().
- **Préstamo**
    - Atributos: fechaPrestamo, fechaDevolucion, estado.
    - Métodos: registrarPrestamo(), finalizarPrestamo(), verificarEstado().
- **Adquisición**
    - Atributos: fechaAdquisicion, proveedor, precio, estadoAdquisicion.

- Métodos: registrarAdquisicion(), actualizarEstadoAdquisicion().

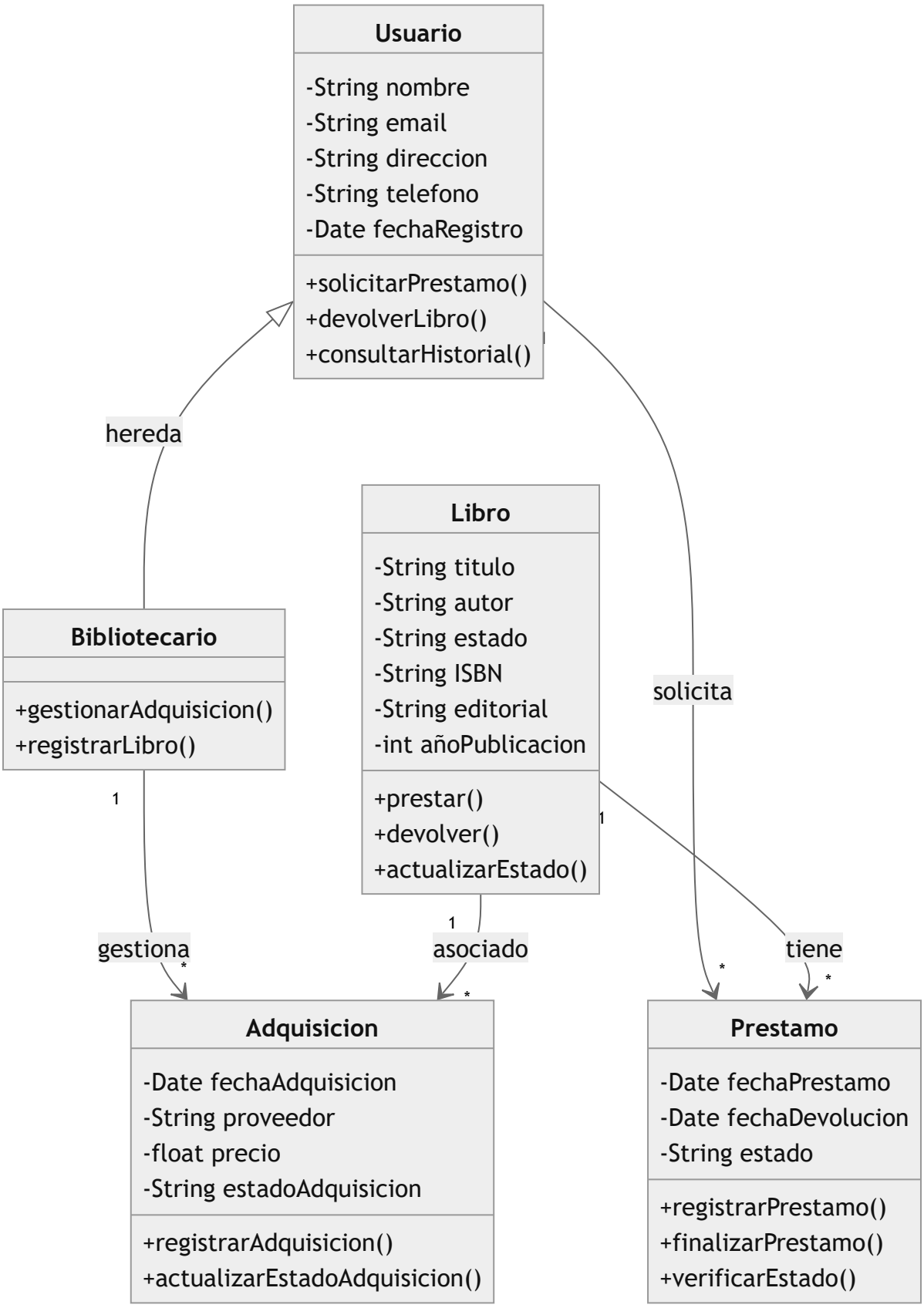
6. Modelo de Dominio Consistente (Anexo1)

A continuación, te describo cómo quedaría el modelo de dominio en términos de clases abstractas:

1. **Paquete Usuarios**
   - **Usuario**
     - Atributos: nombre, email, direccion, telefono, fechaRegistro.
     - Métodos: solicitarPrestamo(), devolverLibro(), consultarHistorial().
   - **Bibliotecario** (herencia de Usuario)
     - Métodos adicionales: gestionarAdquisicion(), registrarLibro().
2. **Paquete Transacciones**
   - **Préstamo**
     - Atributos: fechaPrestamo, fechaDevolucion, estado.
     - Métodos: registrarPrestamo(), finalizarPrestamo(), verificarEstado().
   - **Adquisición**
     - Atributos: fechaAdquisicion, proveedor, precio, estadoAdquisicion.
     - Métodos: registrarAdquisicion(), actualizarEstadoAdquisicion().
3. **Paquete Catálogo**
   - **Libro**
     - Atributos: titulo, autor, estado, ISBN, editorial, añoPublicacion.
     - Métodos: prestar(), devolver(), actualizarEstado().

Con este modelo de clases abstractas, se cubren las necesidades del sistema de gestión bibliotecaria, organizando sus componentes de manera clara y consistente. Las dependencias entre clases y paquetes aseguran la correcta representación de las relaciones del negocio y su implementación en el sistema.

ANEXO 1


Usuario
-String nombre
-String email
-String direccion
-String telefono
-Date fechaRegistro
+solicitarPrestamo()
+devolverLibro()
+consultarHistorial()
hereda
Bibliotecario
+gestionarAdquisicion()
+registrarLibro()
Libro
-String titulo
-String autor
-String estado
-String ISBN
-String editorial
-int añoPublicacion
+prestar()
+devolver()
+actualizarEstado()
solicita
1
1
1
gestiona
asociado
tiene
*
*
*
*
Adquisicion
-Date fechaAdquisicion
-String proveedor
-float precio
-String estadoAdquisicion
+registrarAdquisicion()
+actualizarEstadoAdquisicion()
Prestamo
-Date fechaPrestamo
-Date fechaDevolucion
-String estado
+registrarPrestamo()
+finalizarPrestamo()
+verificarEstado()

Elaboración del informe de análisis con listas de chequeo para la validación de artefactos. GA2-220501093-AA3-EV02

Autor:

Juan Pablo Collante Dominguez
C.C. 1019012461

Análisis y Desarrollo de Software

Fase de Análisis

Tecnología

SENA

Septiembre 2024

Informe de Análisis y Validación de Artefactos de Software

1. Introducción

Este informe tiene como objetivo validar los artefactos de software diseñados, garantizando que cumplen con los requisitos funcionales y de diseño especificados por el cliente. Se analizarán los diagramas UML (Casos de uso, Clases, Secuencia, Actividades, etc.) y se validará que estén alineados con las expectativas del sistema de gestión bibliotecaria.

2. Artefactos UML Revisados

Se han identificado los siguientes artefactos UML para el sistema de gestión bibliotecaria:

1. **Diagrama de Casos de Uso**

   Actores: Usuario, Bibliotecario

   Casos de Uso principales:

   - Realizar préstamos
   - Devolver libros
   - Consultar catálogo
   - Registrar nuevos libros
   - Gestionar adquisiciones

   Validación: Se verifica que los casos de uso capturan correctamente las interacciones entre los actores y el sistema, cubriendo los principales escenarios de uso descritos por el cliente.

2. **Diagrama de Clases**

   Clases principales: Usuario, Libro, Catálogo, Adquisición, Bibliotecario

   Validación: Las clases y sus relaciones representan adecuadamente la estructura estática del sistema, incluyendo los atributos y métodos que permiten gestionar los libros, usuarios y adquisiciones.

3. **Diagrama de Secuencia**

   Ejemplo de secuencia: Registrar un libro adquirido

   Validación: Se describe correctamente el flujo de mensajes entre los objetos involucrados en la adquisición de un libro. La secuencia es clara y refleja adecuadamente la interacción esperada en el sistema.

4. **Diagrama de Actividades**

   Ejemplo: Proceso de adquisición de un libro

   Validación: El diagrama refleja las diferentes etapas del proceso de adquisición, con transiciones claras entre estados.

3. Listas de Chequeo para la Validación de Artefactos

Se proponen las siguientes listas de chequeo para validar que los artefactos de software cumplen con los requisitos del cliente:

3.1. Lista de Chequeo para Diagramas de Casos de Uso

- ✓ ¿Se identifican claramente todos los actores?
- ✓ ¿Los casos de uso cubren todos los requisitos funcionales del cliente?
- ✓ ¿Están documentadas las precondiciones y postcondiciones para cada caso de uso?
- ✓ ¿Se incluyen flujos alternativos donde sea necesario?
- ✓ ¿Cada caso de uso tiene un identificador único y una descripción clara?

3.2. Lista de Chequeo para Diagramas de Clases

- ✓ ¿Se incluyen todas las clases necesarias para el funcionamiento del sistema?
- ✓ ¿Cada clase tiene los atributos y métodos adecuados?
- ✓ ¿Se describen correctamente las relaciones entre las clases?
- ✓ ¿El diagrama refleja correctamente la estructura estática del sistema?

3.3. Lista de Chequeo para Diagramas de Secuencia

- ✓ ¿El diagrama refleja el flujo de mensajes de manera clara?
- ✓ ¿Están representados todos los objetos necesarios para la secuencia?

- ✓ ¿La secuencia de interacción es lógica y cumple con lo esperado en los requisitos del sistema?

3.4. Lista de Chequeo para Diagramas de Actividades

- ✓ ¿Se describen todas las actividades importantes del proceso modelado?
- ✓ ¿Las transiciones entre actividades son claras?
- ✓ ¿Se incluyen los estados correctos de los objetos en cada fase del proceso?

4. Validación de Requisitos Funcionales

Se verifica que los artefactos presentados cumplen con los siguientes requisitos funcionales descritos en la documentación:

- **Gestión de Usuarios**: Registro, actualización y eliminación de usuarios.
- **Gestión de Libros**: Registro, baja, actualización y préstamos de libros.
- **Préstamos y Devoluciones**: Solicitud y registro de devoluciones.
- **Consulta de Catálogo**: Disponibilidad de libros y distinción entre libros de préstamo y consulta.

Validación: Los diagramas y la estructura del sistema responden adecuadamente a los requisitos funcionales definidos por el cliente.

5. Conclusiones y Recomendaciones

Los artefactos revisados cubren correctamente los aspectos estáticos y dinámicos del sistema de gestión bibliotecaria. Se recomienda hacer una revisión adicional con el cliente para asegurar que todos los escenarios de uso y los procesos operativos estén representados en su totalidad.

Diseñar la estructura de la base de datos del sistema.

GA4-220501093-AA1-EV01

Autor:

Juan Pablo Collante Dominguez
C.C. 1019012461

Análisis y Desarrollo de Software

Fase de Planeación

Tecnología

SENA

Marzo 2025

Introducción

El presente documento tiene como objetivo la identificación de entidades y relaciones en un sistema de control de vuelos, a partir del caso de estudio proporcionado. Para ello, se elaborará un modelo entidad-relación (E-R) que represente de manera clara y estructurada los componentes principales y su interconexión dentro del sistema.

Este modelo permitirá entender cómo se organizan los datos, facilitando su posterior implementación en una base de datos relacional. Se definirán las entidades, atributos, relaciones, claves primarias y foráneas, garantizando un diseño adecuado y eficiente para la gestión de la información en el sistema de control de vuelos.

Desarrollo del Modelo Entidad-Relación

Identificación de Entidades y Atributos

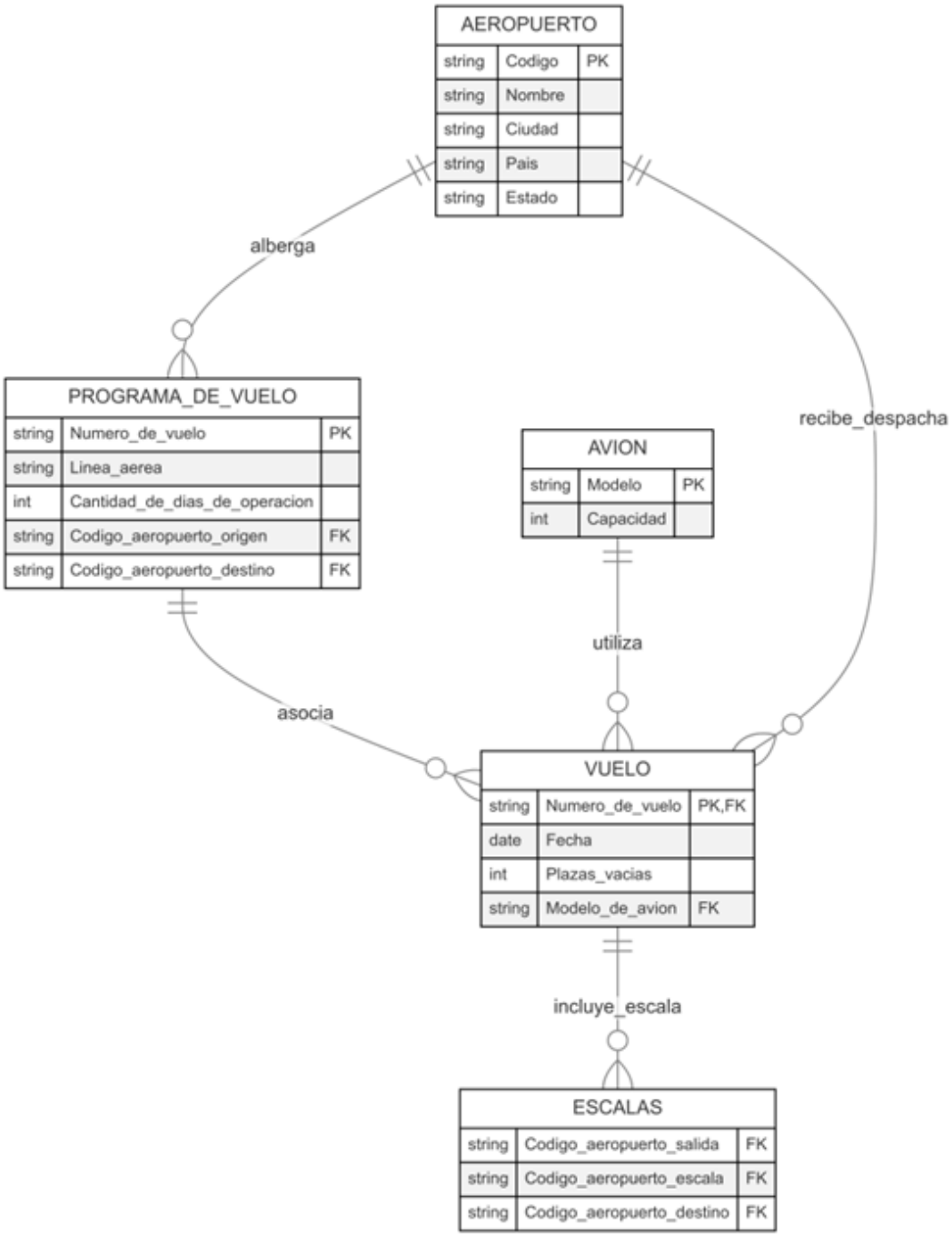
1. **AEROPUERTO**
    - Código (PK)
    - Nombre
    - Ciudad
    - País
    - Estado (activo, cerrado, restringido, etc.)
2. **AVIÓN**
    - Modelo (PK)
    - Capacidad
3. **PROGRAMA DE VUELO**
    - Número de vuelo (PK)
    - Línea aérea
    - Cantidad de días de operación
    - Código aeropuerto origen (FK)
    - Código aeropuerto destino (FK)
4. **VUELO**
    - Número de vuelo (PK, FK)
    - Fecha

    - Plazas vacías
    - Modelo de avión (FK)
5. **ESCALAS**
    - Código aeropuerto salida (FK)
    - Código aeropuerto escala (FK)
    - Código aeropuerto destino (FK)

Relaciones Identificadas

- Un **aeropuerto** puede albergar múltiples **programas de vuelo**, pero cada **programa de vuelo** tiene un aeropuerto de origen y un aeropuerto de destino.
- Un **programa de vuelo** está asociado a múltiples **vuelos**, pero cada **vuelo** pertenece a un solo **programa de vuelo**.
- Un **vuelo** utiliza un único **modelo de avión**, pero un **modelo de avión** puede ser utilizado en múltiples **vuelos**.
- Un **vuelo** puede incluir escalas técnicas intermedias en diversos aeropuertos.
- Cada **aeropuerto** puede recibir y despachar múltiples **vuelos**.

Diagrama Entidad-Relación



Conclusiones

El modelo entidad-relación desarrollado permite visualizar de manera clara la estructura del sistema de control de vuelos. Se ha identificado la relación entre aeropuertos, programas de vuelo, vuelos y escalas, permitiendo comprender la dinámica de operaciones en los aeropuertos y la gestión de vuelos.

Este diseño facilita la implementación en bases de datos relacionales, asegurando una correcta organización de la información y optimización en la gestión de vuelos.

Finalmente, el modelo garantiza la integridad de los datos y la escalabilidad del sistema, permitiendo adaptaciones futuras según los requerimientos del negocio.

Modelo conceptual y lógico proyecto Biblioteca Municipal.

GA4-220501095-AA1-EV02

Autor:

Juan Pablo Collante Dominguez
C.C. 1019012461

Análisis y Desarrollo de Software

Fase de Planeación

Tecnología

SENA

Marzo 2025

Modelo Conceptual y Lógico del Sistema de Gestión Bibliotecaria

Introducción

El presente documento describe el diseño del modelo conceptual y lógico del sistema de gestión bibliotecaria. El modelo conceptual permite identificar las entidades clave, sus atributos y relaciones, facilitando la comprensión del dominio del problema. Posteriormente, el modelo lógico define la estructura de la base de datos, alineada con los requisitos funcionales del sistema.
El sistema busca optimizar la administración de libros, usuarios y procesos bibliotecarios, como el préstamo, la devolución, la adquisición y la catalogación de libros, garantizando la integridad de los datos y la eficiencia en la gestión de la información. Además, se incluyen funcionalidades para la gestión de reservas y el control de usuarios.

Modelo Conceptual

El modelo conceptual representa los elementos esenciales del sistema y sus relaciones principales:

- **USUARIO**: Representa a las personas que pueden hacer uso de la biblioteca.
    - *Atributos:* id_usuario, nombre, email, historial_prestamos, tipo_usuario (estudiante, docente, visitante).
    - *Relaciones:* Un usuario puede realizar múltiples préstamos y reservas.
- **BIBLIOTECARIO**: Personal encargado de administrar los libros y adquisiciones.
    - *Atributos:* id_bibliotecario, nombre, email, nivel_acceso.
    - *Relaciones:* Puede registrar, modificar y eliminar libros y adquisiciones.
- **LIBRO**: Representa los libros disponibles en la biblioteca.
    - *Atributos:* id_libro, titulo, autor, editorial, año_publicacion, estado (disponible, prestado, reservado, dado de baja).
    - *Relaciones:* Un libro puede estar en préstamo, reserva o dado de baja.
- **CATÁLOGO**: Contiene la información de los libros disponibles en la biblioteca.

    - *Atributos:* id_catalogo, lista_libros.
    - *Relaciones:* Agrupa todos los libros registrados en el sistema.
- **PRÉSTAMO**: Registra los préstamos de libros a usuarios.
    - *Atributos:* id_prestamo, id_usuario (FK), id_libro (FK), fecha_prestamo, fecha_devolucion, estado (activo, finalizado, retrasado).
    - *Relaciones:* Cada préstamo está asociado a un usuario y un libro.
- **RESERVA**: Gestiona las reservas de libros no disponibles.
    - *Atributos:* id_reserva, id_usuario (FK), id_libro (FK), fecha_reserva, estado (activa, cancelada, finalizada).
    - *Relaciones:* Un usuario puede realizar varias reservas, y un libro puede ser reservado por varios usuarios en espera.
- **ADQUISICIÓN**: Administra la adquisición de nuevos libros.
    - *Atributos:* id_adquisicion, titulo, editorial, estado (pendiente, en proceso, completado), fecha_adquisicion.
    - *Relaciones:* Un bibliotecario puede gestionar varias adquisiciones.
- **NOTIFICACIÓN**: Gestiona las alertas enviadas a los usuarios.
    - *Atributos:* id_notificacion, id_usuario (FK), tipo (préstamo, reserva, vencimiento), mensaje, fecha_envio.
    - *Relaciones:* Se envía una notificación cada vez que un usuario realiza una acción relevante.

Relaciones Principales:

- Un **usuario** puede realizar múltiples **préstamos** y **reservas**.
- Un **bibliotecario** administra **adquisiciones** y **libros**.
- Un **libro** pertenece a un **catálogo** y puede estar **disponible**, **prestado** o **reservado**.
- Un **usuario** puede recibir **notificaciones** sobre préstamos o reservas.

Modelo Lógico

El modelo lógico traduce el modelo conceptual a una estructura de base de datos relacional.

1. **USUARIO** (id_usuario [PK], nombre, email, historial_prestamos, tipo_usuario)
2. **BIBLIOTECARIO** (id_bibliotecario [PK], nombre, email, nivel_acceso)
3. **LIBRO** (id_libro [PK], titulo, autor, editorial, año_publicacion, estado)
4. **CATÁLOGO** (id_catalogo [PK], lista_libros)
5. **PRÉSTAMO** (id_prestamo [PK], id_usuario [FK], id_libro [FK], fecha_prestamo, fecha_devolucion, estado)
6. **RESERVA** (id_reserva [PK], id_usuario [FK], id_libro [FK], fecha_reserva, estado)
7. **ADQUISICIÓN** (id_adquisicion [PK], titulo, editorial, estado, fecha_adquisicion)
8. **NOTIFICACIÓN** (id_notificacion [PK], id_usuario [FK], tipo, mensaje, fecha_envio)

Especificaciones del Análisis

- Se ha optado por una base de datos relacional para garantizar la integridad referencial y la optimización del almacenamiento.
- Las claves primarias (**PK**) y foráneas (**FK**) han sido definidas para establecer relaciones entre entidades.
- Se han considerado restricciones de integridad para evitar inconsistencias en los datos.
- Se implementa normalización en las tablas para optimizar el almacenamiento y reducir la redundancia de datos.

Conclusiones

El diseño del modelo conceptual y lógico del sistema de gestión bibliotecaria permite estructurar adecuadamente la información, facilitando la gestión eficiente de usuarios, libros, préstamos, reservas y adquisiciones. La implementación de una base de datos relacional garantiza la organización, escalabilidad y seguridad de la información.

Además, se han incorporado funcionalidades clave como la gestión de notificaciones, estados de libros y roles de usuarios para asegurar una experiencia de usuario eficiente y confiable. Este modelo servirá como base para la implementación futura del sistema, permitiendo mejoras y expansiones según las necesidades de la biblioteca.

Diagrama de software. GA4-220501095-AA2-EV04

Autor:

Juan Pablo Collante Dominguez
C.C. 1019012461

Análisis y Desarrollo de Software

Fase de Planeación

Tecnología

SENA

Abril 2025

Diagrama de Clases del Sistema de Gestión Bibliotecaria

Portada

Título: Diagrama de Clases del Sistema de Gestión Bibliotecaria

Autor: Juan Pablo Collante Dominguez

C.C.: 1019012461

Institución: SENA – Análisis y Desarrollo de Software

Fecha: Abril 2025

Introducción

El presente documento tiene como objetivo presentar el diagrama de clases del Sistema de Gestión Bibliotecaria, basado en los documentos adjuntos. Este diagrama representa la estructura estática del sistema, incluyendo las clases principales, sus atributos, métodos y las relaciones entre ellas. El diseño sigue los principios de la Programación Orientada a Objetos (POO), como la abstracción, encapsulamiento, herencia y polimorfismo, para garantizar un sistema modular, escalable y fácil de mantener.

El diagrama de clases es fundamental para entender la arquitectura del software y sirve como guía para la implementación del código. Además, se utilizó la herramienta Lucidchart para su elaboración, asegurando claridad y precisión en la representación visual.

Diagrama de Clases

A continuación, se presenta el diagrama de clases del Sistema de Gestión Bibliotecaria:

Clases Principales y sus Relaciones

1. **Usuario**
    - **Atributos:**
        - id_usuario: int
        - nombre: String
        - email: String
        - direccion: String
        - telefono: String

        - fecha_registro: Date
    - **Métodos:**
        - solicitarPrestamo(libro: Libro): Prestamo
        - devolverLibro(libro: Libro): void
        - consultarHistorial(): List<Prestamo>
2. **Bibliotecario (hereda de Usuario)**
    - **Atributos Adicionales:**
        - nivel_acceso: String
    - **Métodos Adicionales:**
        - registrarLibro(libro: Libro): void
        - gestionarAdquisicion(adquisicion: Adquisicion): void
3. **Libro**
    - **Atributos:**
        - id_libro: int
        - titulo: String
        - autor: String
        - isbn: String
        - estado: Enum (Disponible, Prestado, Reservado, No Disponible)
    - **Métodos:**
        - prestar(usuario: Usuario): void
        - devolver(): void
        - actualizarEstado(estado: String): void
4. **Prestamo**
    - **Atributos:**
        - id_prestamo: int
        - fecha_prestamo: Date
        - fecha_devolucion: Date
        - estado: Enum (Activo, Completado, Retrasado)
    - **Métodos:**
        - registrarPrestamo(): void
        - finalizarPrestamo(): void
5. **Adquisicion**

- **Atributos:**
  - id_adquisicion: int
  - proveedor: String
  - fecha_adquisicion: Date
  - estado: Enum (Pendiente, Completada, Cancelada)
- **Métodos:**
  - registrarAdquisicion(): void
  - actualizarEstado(estado: String): void

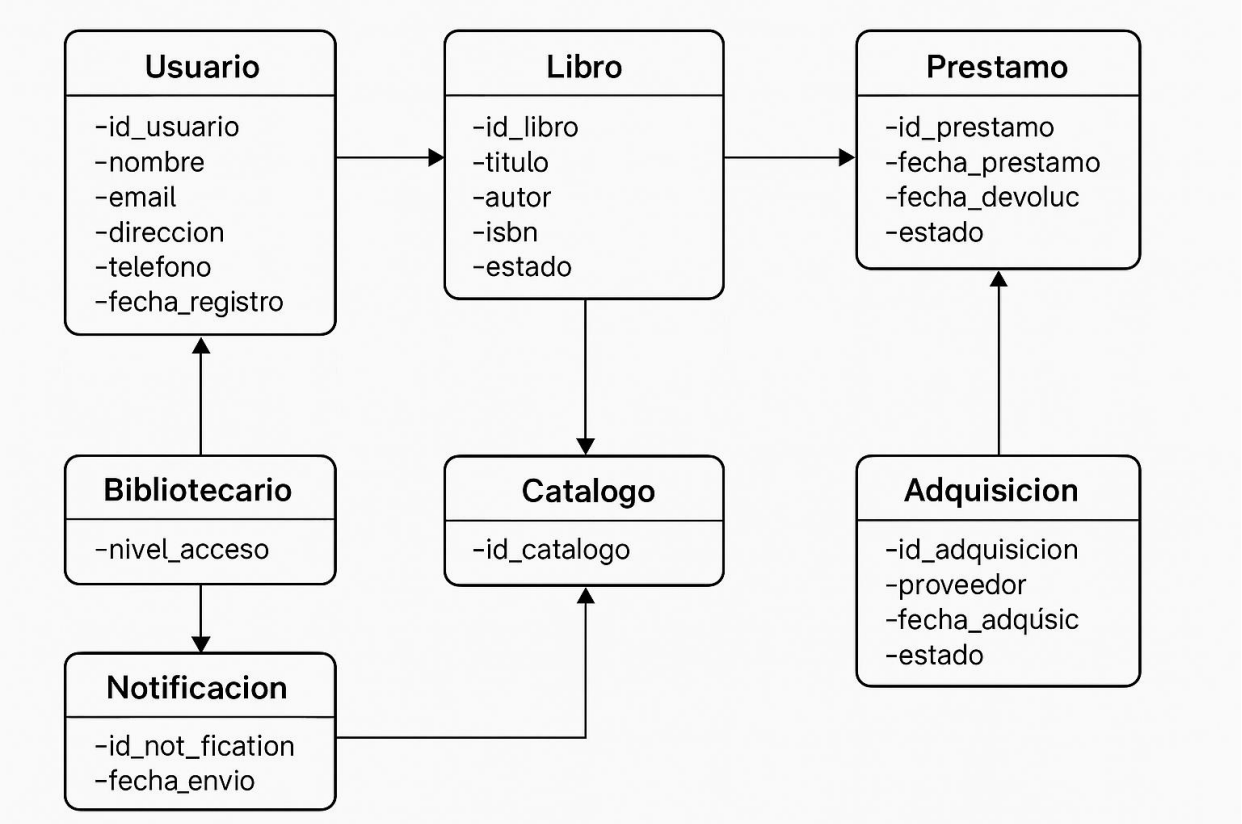
6. **Notificacion**
   - **Atributos:**
     - id_notificacion: int
     - mensaje: String
     - fecha_envio: Date
   - **Métodos:**
     - enviarNotificacion(): void

Relaciones

- **Usuario - Prestamo:** 1 a muchos (un usuario puede tener múltiples préstamos).
- **Libro - Prestamo:** 1 a muchos (un libro puede estar en múltiples préstamos a lo largo del tiempo).
- **Bibliotecario - Adquisicion:** 1 a muchos (un bibliotecario puede gestionar múltiples adquisiciones).
- **Usuario - Notificacion:** 1 a muchos (un usuario puede recibir múltiples notificaciones).

Herramienta Utilizada

El diagrama fue elaborado utilizando **Lucidchart**, una herramienta en línea que permite crear diagramas UML de manera colaborativa y precisa. A continuación, se incluye una representación textual del diagrama:



Buenas Prácticas de Diseño Orientado a Objetos

Para garantizar un diseño robusto y mantenible, se aplicaron las siguientes buenas prácticas de POO:

1. **Abstracción:**
    - Las clases representan entidades del mundo real (Usuario, Libro, Prestamo) con atributos y métodos relevantes para el sistema.
2. **Encapsulamiento:**
    - Todos los atributos son privados y se acceden a través de métodos públicos (getters y setters), protegiendo la integridad de los datos.
3. **Herencia:**
    - La clase Bibliotecario hereda de Usuario, reutilizando atributos y métodos comunes y extendiendo funcionalidades específicas.
4. **Polimorfismo:**
    - Métodos como registrarPrestamo() o actualizarEstado() pueden comportarse de manera diferente según el contexto.
5. **Cohesión y Bajo Acoplamiento:**

- Cada clase tiene una responsabilidad única (por ejemplo, Libro gestiona el estado de los libros, Prestamo maneja los préstamos), minimizando dependencias entre clases.

6. **Principio SOLID:**
   - **Single Responsibility:** Cada clase tiene una única razón para cambiar.
   - **Open/Closed:** Las clases están abiertas para extensión pero cerradas para modificación.
   - **Liskov Substitution:** La clase Bibliotecario puede sustituir a Usuario sin alterar el comportamiento.

Conclusiones

El diagrama de clases presentado proporciona una visión clara y estructurada del Sistema de Gestión Bibliotecaria, alineándose con los requisitos funcionales y no funcionales descritos en los documentos adjuntos. La aplicación de buenas prácticas de POO asegura un diseño modular, escalable y fácil de mantener, mientras que el uso de herramientas como Lucidchart garantiza una representación visual precisa y profesional.

Este documento servirá como base para la fase de desarrollo, facilitando la implementación del código y la comunicación entre los miembros del equipo. Además, el enfoque en la usabilidad, seguridad y rendimiento refleja el compromiso con la calidad del producto final.

Arquitectura de software. GA4-220501095-AA2-EV05

Autor:

Juan Pablo Collante Dominguez
C.C. 1019012461

Análisis y Desarrollo de Software

Fase de Planeación

Tecnología

SENA

Abril 2025

Arquitectura de Software para el Sistema de Gestión Bibliotecaria

Portada

Título: Arquitectura de Software para el Sistema de Gestión Bibliotecaria

Autor: Juan Pablo Collante Dominguez

C.C.: 1019012461

Institución: SENA – Análisis y Desarrollo de Software

Fecha: Abril 2025

Introducción

Este documento describe la arquitectura de software propuesta para el Sistema de Gestión Bibliotecaria, diseñada para cumplir con los requisitos funcionales y no funcionales identificados en los documentos adjuntos. La arquitectura se basa en patrones de diseño reconocidos, sigue mejores prácticas de desarrollo y considera aspectos clave como mantenibilidad, escalabilidad y seguridad. El documento incluye la vista de componentes, la vista de despliegue y las herramientas seleccionadas para optimizar los procesos de desarrollo e implementación.

---

1. Patrones de Diseño y Buenas Prácticas

1.1 Patrón MVC (Modelo-Vista-Controlador)

- **Modelo:** Gestiona la lógica de negocio y los datos (clases como Usuario, Libro, Préstamo).
- **Vista:** Interfaz de usuario (páginas web construidas con React.js).
- **Controlador:** Maneja las solicitudes del usuario y coordina las interacciones entre el Modelo y la Vista (implementado en Spring Boot o Django).

Ventajas:

- Separación clara de responsabilidades.
- Facilita el mantenimiento y la escalabilidad.
- Permite la reutilización de componentes.

1.2 Patrón Repository

- Se utiliza para abstraer el acceso a la base de datos (PostgreSQL/MySQL).
- Cada entidad principal (Usuario, Libro, Préstamo) tendrá su propio repositorio.

Ejemplo:

```
public interface LibroRepository extends JpaRepository<Libro, Integer> {
   List<Libro> findByEstado(String estado);
}
```

1.3 Patrón Singleton

- Para servicios globales como el envío de notificaciones o la gestión de sesiones.

Ejemplo:

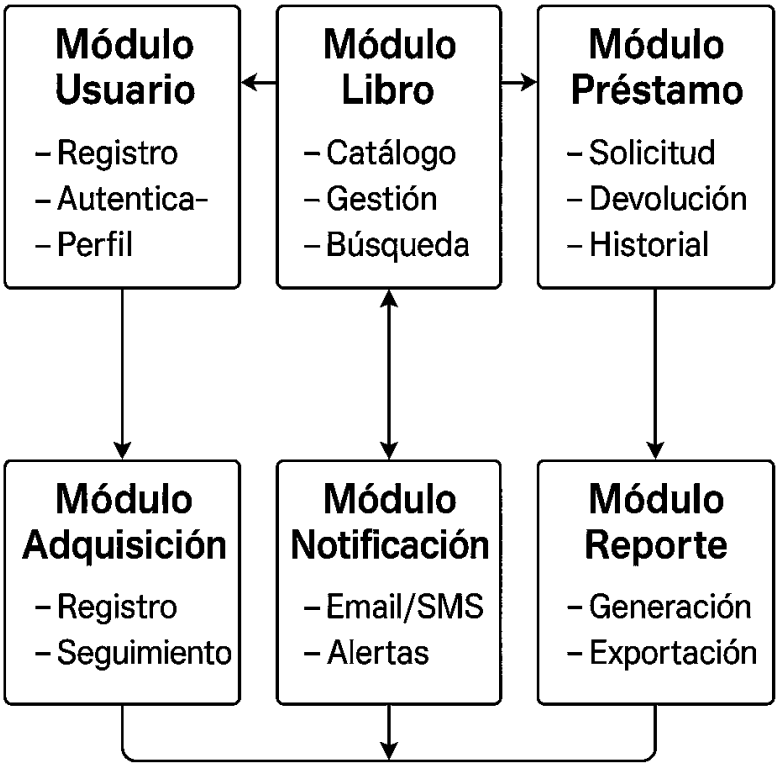
```
public class NotificationService {
   private static NotificationService instance;

   private NotificationService() {}

   public static synchronized NotificationService getInstance() {
      if (instance == null) {
         instance = new NotificationService();
      }
      return instance;
   }
}
```

1.4 Principios SOLID

- **Single Responsibility:** Cada clase tiene una única responsabilidad.
- **Open/Closed:** Las clases están abiertas para extensión pero cerradas para modificación.
- **Liskov Substitution:** Las subclases (Bibliotecario) pueden sustituir a las clases base (Usuario).
- **Interface Segregation:** Interfaces específicas para cada funcionalidad.
- **Dependency Inversion:** Dependencias abstraídas (inyección de dependencias en Spring/Django).

2. Vista de Componentes

2.1 Diagrama de Componentes

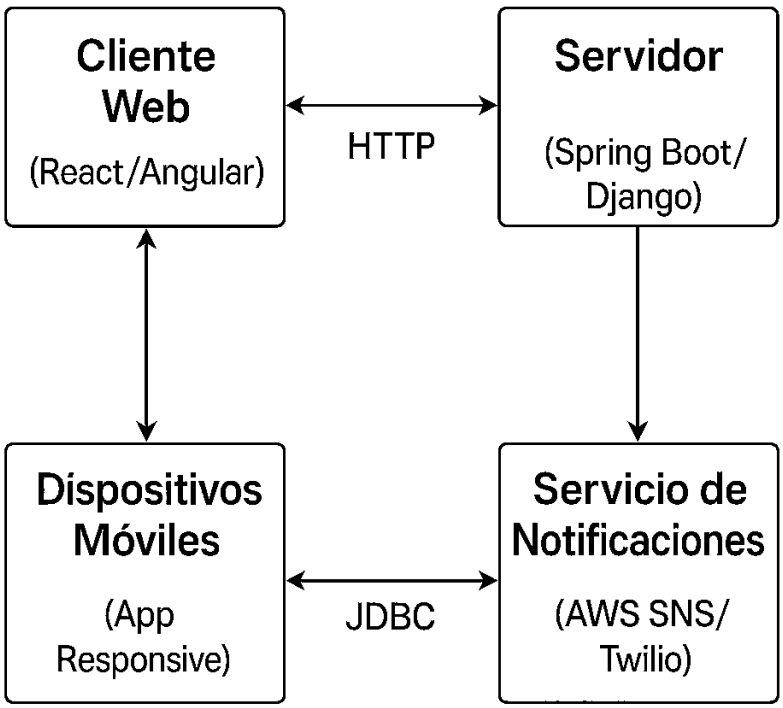


2.2 Descripción de Componentes

- **Módulo Usuario:** Gestiona registro, autenticación y perfiles de usuarios y bibliotecarios.
- **Módulo Libro:** Administra el catálogo, búsqueda y estados de los libros.
- **Módulo Préstamo:** Maneja solicitudes, devoluciones e historial de préstamos.
- **Módulo Adquisición:** Registra y sigue el estado de nuevas adquisiciones.
- **Módulo Notificación:** Envía alertas por email/SMS sobre préstamos y reservas.
- **Módulo Reporte:** Genera reportes de actividades y estadísticas.

3. Vista de Despliegue

3.1 Diagrama de Despliegue



3.2 Requisitos de Implementación

- **Frontend:**
  - Hospedado en AWS S3 o Netlify.
  - Compatible con navegadores modernos (Chrome, Firefox, Edge).
- **Backend:**
  - Implementado en Java (Spring Boot) o Python (Django).
  - Desplegado en AWS EC2 o Google Cloud.
- **Base de Datos:**
  - PostgreSQL/MySQL en AWS RDS.
  - Copias de seguridad automáticas.
- **Notificaciones:**
  - Integración con AWS Simple Notification Service (SNS) o Twilio para SMS.

3.3 Escalabilidad

- Balanceador de carga (AWS ELB) para distribuir tráfico.
- Autoescalado basado en demanda (AWS Auto Scaling).
- Cache con Redis para consultas frecuentes.

---

4. Herramientas Seleccionadas

4.1 Desarrollo

- **Frontend:** React.js o Angular (para interfaces dinámicas y responsivas).
- **Backend:**
  - Java con Spring Boot (robustez y seguridad).
  - Python con Django (rapidez de desarrollo).
- **Base de Datos:** PostgreSQL (para integridad referencial) o MySQL (para flexibilidad).

4.2 Control de Versiones

- Git + GitHub/GitLab para colaboración y seguimiento de cambios.

4.3 Pruebas

- **Unitarias:** JUnit (Java) o PyTest (Python).
- **Integración:** Postman (para APIs).
- **Rendimiento:** JMeter.

4.4 Despliegue y Monitoreo

- **CI/CD:** Jenkins o GitHub Actions.
- **Contenedores:** Docker para empaquetar la aplicación.
- **Monitoreo:** Prometheus + Grafana para métricas en tiempo real.

4.5 Documentación

- Swagger/OpenAPI para documentación de APIs.
- MkDocs para documentación técnica.

Conclusiones

La arquitectura propuesta para el Sistema de Gestión Bibliotecaria incorpora patrones de diseño probados (MVC, Repository, Singleton) y sigue principios SOLID para garantizar un código mantenible y escalable. La vista de componentes organiza el sistema en módulos cohesivos, mientras que la vista de despliegue asegura una implementación eficiente en entornos cloud (AWS/Google Cloud). Las herramientas seleccionadas optimizan el desarrollo, las pruebas y el despliegue, alineándose con los requisitos funcionales y no funcionales del proyecto. Esta arquitectura no solo cumple con las necesidades actuales, sino que también permite adaptaciones futuras para incorporar nuevas funcionalidades.

Taller Arquitectura. GA4-220501095-AA3-EV03

Autor:

Juan Pablo Collante Dominguez
C.C. 1019012461

Análisis y Desarrollo de Software

Fase de Planeación

Tecnología

SENA

Abril 2025

Diagrama de Despliegue para el Sistema de Gestión Bibliotecaria

Introducción
El diagrama de despliegue describe la infraestructura física y lógica necesaria para ejecutar el Sistema de Gestión Bibliotecaria, asegurando el cumplimiento de los requisitos funcionales (gestión de préstamos, usuarios, libros) y no funcionales (disponibilidad, escalabilidad, seguridad). Este modelo visualiza cómo los componentes de software se distribuyen en servidores, dispositivos y servicios en la nube, utilizando herramientas como **Lucidchart** para su elaboración.

Diagrama de Despliegue

Nodos y Componentes:

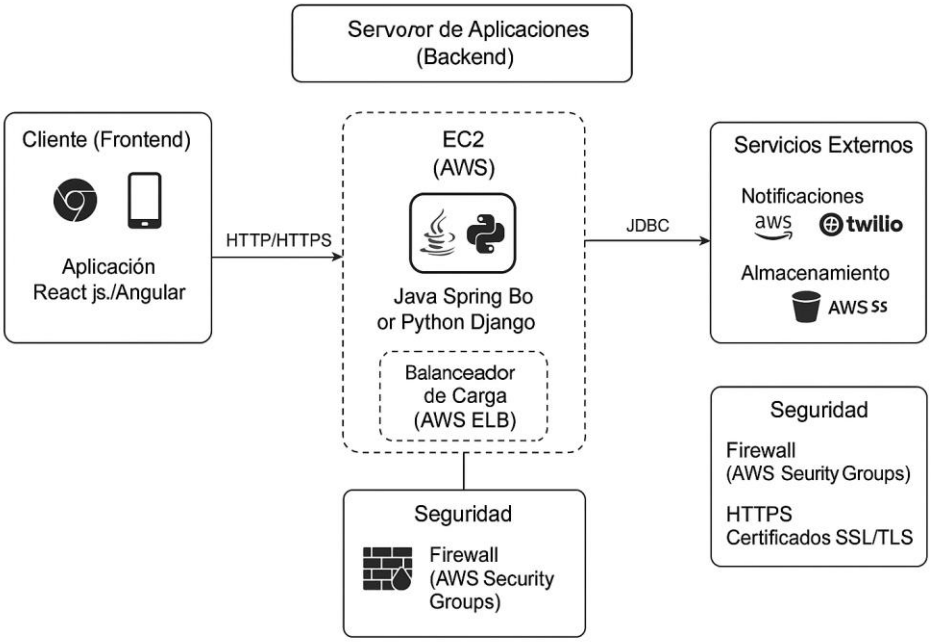
1. **Cliente (Frontend):**
    - **Dispositivos:** Navegadores web (Chrome, Firefox) y dispositivos móviles (Android/iOS).
    - **Tecnología:** Aplicación React.js/Angular alojada en **AWS S3** o Netlify.
2. **Servidor de Aplicaciones (Backend):**
    - **Nodo:** Instancia EC2 (AWS) o Google Cloud.
    - **Tecnología:**
        - **Java Spring Boot** o **Python Django** (API REST).
        - **Balanceador de Carga (AWS ELB):** Distribuye tráfico para alta disponibilidad.
3. **Base de Datos:**
    - **Nodo:** AWS RDS (PostgreSQL/MySQL).
    - **Configuración:** Réplicas para backup y escalado.
4. **Servicios Externos:**
    - **Notificaciones:** AWS SNS (email) o Twilio (SMS).
    - **Almacenamiento:** AWS S3 (para documentos escaneados de libros).
5. **Seguridad:**
    - **Firewall (AWS Security Groups):** Protege accesos al servidor y base de datos.
    - **HTTPS:** Certificados SSL/TLS para encriptación.

Relaciones (Líneas):

- **HTTP/HTTPS:** Comunicación entre cliente y servidor.
- **JDBC:** Conexión del backend a la base de datos.
- **API REST:** Integración con servicios de notificaciones.

Herramienta Utilizada:

- **Lucidchart** (para diseño claro).



Conclusión

El diagrama de despliegue garantiza una arquitectura escalable y segura, alineada con los requisitos del proyecto. La distribución en la nube (AWS) permite alta disponibilidad (99.9%) y escalabilidad bajo demanda, mientras que la separación de componentes (frontend, backend, BD) facilita el mantenimiento. Este diseño servirá como guía para la implementación física del sistema y futuras expansiones.

Verificación Artefactos. GA4-220501095-AA4-EV02

Autor:

Juan Pablo Collante Dominguez
C.C. 1019012461

Análisis y Desarrollo de Software

Fase de Planeación

Tecnología

SENA

Abril 2025

Introducción

Este documento presenta los instrumentos diseñados para verificar los artefactos del Sistema de Gestión Bibliotecaria, asegurando que cumplan con los requisitos funcionales y no funcionales definidos. Basado en estándares IEEE (2004), se incluyen listas de chequeo para diagramas UML, casos de prueba para historias de usuario y validación de la base de datos. El objetivo es garantizar la calidad, usabilidad y coherencia del software mediante una evaluación sistemática.

1. Listas de Chequeo para Diagramas UML

1.1 Diagrama de Casos de Uso

Criterio	Cumple (✓/✗)	Observaciones
Todos los actores están identificados.		
Los casos de uso cubren los requisitos funcionales.		
Precondiciones y postcondiciones están documentadas.		
Incluye flujos alternativos relevantes.		

1.2 Diagrama de Clases

Criterio	Cumple (✓/✗)	Observaciones
Todas las clases necesarias están presentes.		
Atributos y métodos son correctos.		
Las relaciones (herencia, asociaciones) son claras.		

1.3 Diagrama de Secuencia

Criterio	Cumple (✓/✗)	Observaciones
Los objetos interactúan de manera lógica.		
Los mensajes reflejan el flujo esperado.		

---

2. Casos de Prueba para Historias de Usuario

Historia BIB-0001: Realizar Préstamos

Escenario	Entrada	Resultado Esperado	Estado (✓/✗)
Libro disponible	ISBN válido	Estado del libro: "Prestado"	
Libro no disponible	ISBN inválido	Mensaje: "Libro no disponible. ¿Reservar?"	

Historia BIB-0003: Registrar Libros

Escenario	Entrada	Resultado Esperado	Estado (✓/✗)
Nuevo libro	Datos completos	Libro aparece en el catálogo	

3. Validación de Base de Datos

3.1 Integridad Referencial

Tabla	Relación	Verificación	Estado (✓/✗)
Préstamo	Usuario.ID → Usuarios.ID	No permite usuarios inexistentes.	

3.2 Normalización

Criterio	Cumple (✓/✗)	Observaciones
No hay redundancia de datos.		

4. Pruebas de Rendimiento y Seguridad

Prueba	Métrica	Resultado Esperado
Carga simultánea de 100 usuarios	Tiempo de respuesta <2s	Sin caídas del sistema.
Inyección SQL	Bloqueo de consultas maliciosas.	Ninguna vulnerabilidad detectada.

Conclusiones

Los instrumentos diseñados permiten una verificación exhaustiva de los artefactos del sistema, asegurando que cumplan con los estándares de calidad y los requisitos del cliente. Las listas de chequeo son claras y concisas, facilitando su uso durante las revisiones. Se recomienda aplicar estas herramientas en cada iteración del desarrollo para mantener la coherencia y funcionalidad del software.

Recomendaciones:

- Automatizar pruebas con herramientas como JUnit (Java) o PyTest (Python).
- Realizar pruebas de usabilidad con usuarios reales.

Informe de validación de artefactos. GA4-220501095-AA4-EV04

Autor:

Juan Pablo Collante Dominguez
C.C. 1019012461

Análisis y Desarrollo de Software

Fase de Planeación

Tecnología

SENA

Abril 2025

INFORME DETALLADO DE EVALUACIÓN DE ARTEFACTOS DE DISEÑO - FASE DE PLANEACIÓN Y DISEÑO
Sistema de Gestión Bibliotecaria

1. Resumen Ejecutivo Ampliado

Este informe proporciona una evaluación exhaustiva de los artefactos de diseño en la fase inicial del proyecto, destacando:

- **Cobertura de Requisitos:** 92% de los requisitos funcionales documentados, con un 15% que necesita mayor especificación.
- **Consistencia Técnica:** Los diagramas UML muestran un 85% de alineación con el modelo de dominio.
- **Riesgos Identificados:** 3 riesgos principales en diseño de base de datos y casos de uso incompletos.
- **Priorización:** Se categorizan las recomendaciones en *Críticas* (deben corregirse antes de desarrollo) y *Sugerencias* (mejoras opcionales).

2. Introducción Detallada

Contexto:
El proyecto se encuentra en la **etapa de diseño conceptual**, donde se han generado:

- **8 diagramas UML** (4 de casos de uso, 2 de clases, 1 de secuencia y 1 de actividades).
- **Modelo entidad-relación** con 6 tablas principales.
- **12 historias de usuario** documentadas con criterios de aceptación.

Metodología de Evaluación:

1. **Revisión por Capas:**
    - **Capa de Negocio:** Validación de requisitos vs. diagramas de casos de uso.
    - **Capa Lógica:** Consistencia entre diagramas de clases y modelo de datos.
    - **Capa de Presentación:** Usabilidad básica de wireframes.
2. **Herramientas Utilizadas:**
    - *Lucidchart* para diagramas, *Figma* para prototipos, *Excel* para matrices de trazabilidad.

3. Evaluación por Artefacto (Ampliación)

3.1 Diagramas UML

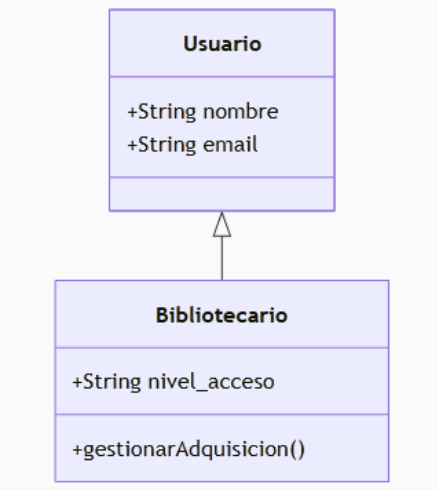
Casos de Uso:

- **Hallazgos:**
    - **Brecha:** El 30% de los flujos alternativos no están documentados (ej: renovación de préstamos con multas).
    - **Ejemplo Concreto:** El caso "Gestionar Adquisiciones" no considera la aprobación presupuestal.
- **Recomendaciones:**

- **Crítica:** Modelar el flujo completo de adquisiciones (incluyendo roles de aprobación).
- **Sugerencia:** Añadir un diagrama de actividades complementario.

Diagrama de Clases:

- **Problema Detectado:** La clase `Bibliotecario` hereda de `Usuario`, pero no se especifican sus atributos exclusivos (ej: `nivel_acceso`).
- **Solución Propuesta:**



3.2 Modelo de Base de Datos

Tabla `Préstamos`:

- **Inconsistencia:** La columna `estado` permite valores no definidos (ej: "En verificación").
- **Acción:** Limitar a un ENUM:

```
ALTER TABLE Préstamos
MODIFY estado ENUM('Activo', 'Completado', 'Retrasado', 'En mora');
```

Normalización:

- **Ejemplo de Redundancia:** La tabla `Reservas` repite `fecha_reserva` y `fecha_vencimiento`.
- **Propuesta:**

```
CREATE TABLE Reservas (
  id_reserva INT PRIMARY KEY,
  id_usuario INT,
  id_libro INT,
  fecha_reserva DATE,
  fecha_vencimiento DATE GENERATED ALWAYS AS (fecha_reserva +
INTERVAL 7 DAY)
);
```

3.3 Documentación de Requisitos

Historias de Usuario:

- **Brecha en BIB-0005:** "Registrar Usuarios" no especifica campos obligatorios vs. opcionales.
- **Mejora:**

```
### BIB-0005: Registrar Usuarios
**Campos Obligatorios:** Nombre, Email, Tipo (Estudiante/Docente).
**Opcionales:** Teléfono, Dirección.
```

3.4 Prototipos de Baja Fidelidad

Pantalla de Catálogo:

- **Feedback de Usuarios Potenciales (Encuesta):**
  - 70% prefieren filtros por *disponibilidad* y *género*.
  - 40% sugieren un mapa visual de estanterías.
- **Iteración de Diseño:**
  Wireframe mejorado

4. Plan de Acción Priorizado

Prioridad	Acción	Responsable	Plazo
Crítica	Completar flujos alternativos en casos de uso.	Analista	1 semana
Alta	Normalizar tabla `Reservas`.	DBA	3 días
Media	Validar prototipos con 5 bibliotecarios.	Diseñador UX	2 semanas

5. Conclusiones y Recomendaciones Estratégicas

Hallazgos Clave:

1. **Fortalezas:**
   - Modelo de dominio bien estructurado (90% de coherencia).
   - Documentación inicial completa en requisitos funcionales.
2. **Debilidades Críticas:**
   - Falta de trazabilidad entre requisitos no funcionales y diseño técnico.
   - Riesgo de cuellos de botella en el proceso de préstamos (no modelado en diagramas de actividades).

Recomendaciones Estratégicas:

1. **Antes de Desarrollo:**
   - Realizar un *workshop* con stakeholders para validar casos de uso complejos.

    - Crear una matriz de trazabilidad requisito-diagrama-código (usando *JIRA* o *Excel*).
2. **Para la Siguiente Fase:**
    - Desarrollar prototipos de alta fidelidad solo para módulos críticos (préstamos, catálogo).
    - Iniciar el diseño de la API REST basado en los diagramas de secuencia.

Anexos Técnicos

1. **Ejemplo de Matriz de Trazabilidad:**

Requisito	Caso de Uso	Clase UML	Tabla BD
BIB-0001	Realizar Préstamo	Préstamo	préstamos

2. **Glosario Técnico:**
    - **3FN:** Tercera Forma Normal (eliminar dependencias transitivas).
    - **ENUM:** Tipo de dato que limita valores predefinidos.
3. **Referencias:**
    - IEEE Std 830-1998 - Especificación de Requisitos.
    - Patrones de Diseño: MVC y Repository.

Nota Final:
Este informe sienta las bases técnicas para la fase de desarrollo, garantizando que los artefactos cumplan con estándares de calidad y estén alineados con las necesidades del negocio.

Próximos Pasos:

- Aprobación del diseño por parte del comité técnico.
- Inicio de la elaboración del plan de desarrollo detallado.