FORMATO DE REQUERIMIENTOS DEL SISTEMA
Ingeniería de Software I

1. Información general del proyecto

Nombre del proyecto	Préstamo de Libros
Integrantes	Raúl De Jesús Ugalde
Programa académico	Ingeniería de Sistemas y Computación
Fecha de entrega	18/02/2026
Lenguaje de programación	Java
Tipo de aplicación	Consola

2. Descripción general del sistema

- El sistema permite registrar los libros existentes en el sistema de la biblioteca de la Universidad de Cundinamarca de forma correcta.
- El sistema enlista los libros por título, autor, identificador único y muestra el estado actual, ya sea disponible o prestado, así como la opción de devolver el libro.
- Las restricciones que tiene la biblioteca es que un libro prestado al usuario, no se podrá prestar por segunda ocasión al mismo usuario y solo los libros que están disponibles en la biblioteca pueden ser prestados.
- El sistema muestra los resultados y controla de forma segura y eficaz el préstamo de libros de la biblioteca.
- El desarrollo del programa solicitado está desarrollado en el lenguaje de programación: Java, aplicando lo aprendido en Programación Orientada a Objetos.

3. Requerimientos Funcionales (RF)

ID	Nombre	Descripción	Entrada(s)	Proceso	Salida
RF-01	Registrar los libros existentes	Permite registrar los libros de forma correcta en el sistema, evitando extravío de los mismos.	Autor, título e identificador único	Registrar los datos del libro en el sistema de forma correcta	Datos agregados en el sistema

RF-02	Cantidad de libros disponibles	Permite registrar la cantidad de libros disponibles en la biblioteca	Cantidad	Calcula la suma del total de piezas disponibles	Muestra el total de libros disponibles en la Universidad de Cundinamarca
RF-03	Condiciones de la biblioteca	Establece reglas que se deben cumplir para el funcionamiento del sistema	Usuario, libro prestado, libros disponibles	Analiza en la lista si al usuario se le prestó ese libro y a partir de ese momento empiezan a funcionar las condiciones	Dictamina si se le puede prestar el libro a un usuario.
RF-04	Mostrar resultados	Permite ver los datos del sistema en tiempo real	Registrar los libros existentes, Cantidad de libros disponibles, Condiciones de la biblioteca y Mostrar resultados	Analiza, muestra y hace el corte final de los resultados del sistema en base a los procesos anteriores	Mostrar Resultados

4. Requerimientos No Funcionales (RNF)

ID	Tipo	Descripción
RNF-01	Disponibilidad	El sistema debe estar disponible para consultas de inventario el 99.9% del tiempo durante el horario de la biblioteca.
RNF-02	Seguridad	Solo el rol de Administrador podrá modificar los porcentajes de intereses aplicados, mediante autenticación de dos factores.
RNF-03	Mantenibilidad	El código debe seguir el patrón de arquitectura

		por capas para facilitar futuras actualizaciones sin afectar el núcleo del sistema.

5. Relación Requerimiento – POO

ID Requerimiento	Clase	Método	Tipo
RF-01	Libro	rLibro()	Funcional
RF-02	LibroService	cDisponibles()	Funcional
RF-03	PrestamoService	vPrestamo()	Funcional
RF-04	LibroService	mResultados()	Funcional
RNF-01	ValidacionUtil	vCantidad()	No Funcional
RNF-02	ValidacionUtil	vInteres	No Funcional

6. Entidades del Dominio

1. Libro: Es la clase central que contiene los atributos de los libros, tales como: titulo, autor, identificador y estado, ya sea que esté disponible o no.
2. Biblioteca: En esta entidad es donde se gestiona la cantidad total de libros disponibles.
3. Usuario. Es la clase que define la relación entre el usuario y el libro.
4. Prestamo de libros: Se define la relación entre el usuario y el libro, basándose en las condiciones que tiene la biblioteca

7. Relaciones de las entidades

1. Usuario – Libro
2. Biblioteca – Libro
3. Libro – Estado
4. Usuario – Condiciones

8. Diagramas

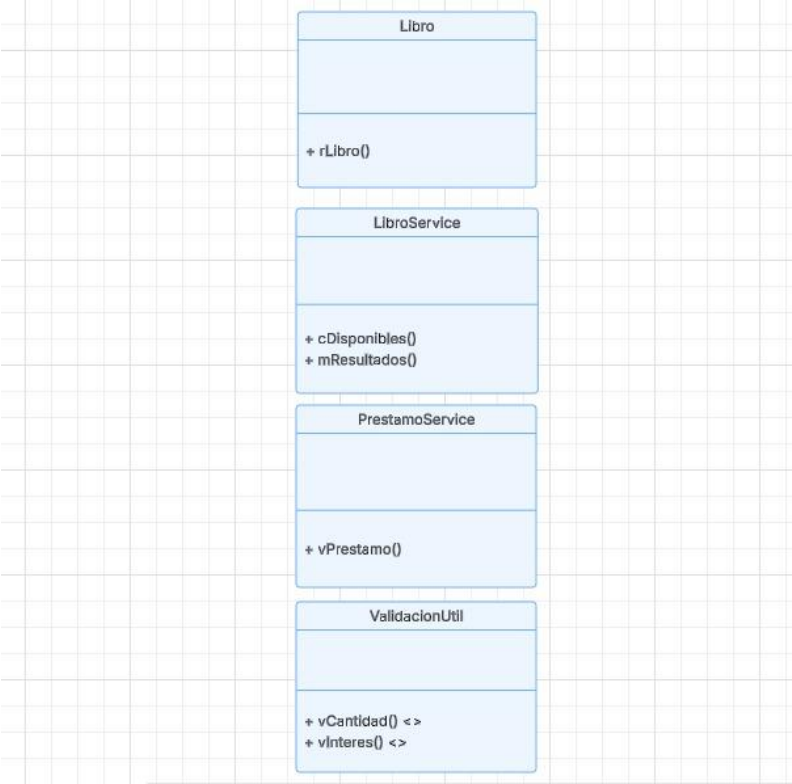


Figura 1.1. Diagrama de clases de fuente propia

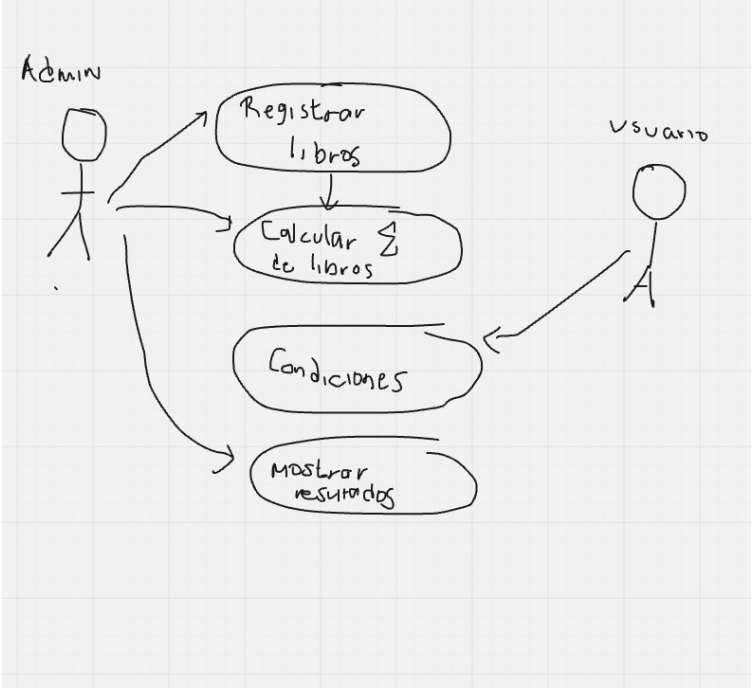


Figura 1.2. Diagrama de casos entre el administrador de la biblioteca y el usuario – Fuente propia

9. Tecnologías de Java a usar

Se usa la arquitectura (RNF-03), además de la modularidad por paquetes, ya que organiza el código mediante módulos lógicos (model, service,util), facilitando la navegación de forma eficiente, además del mantenimiento y la escalabilidad del sistema. Oracle (2025)

Java Collections Framework: Nos permite almacenar la cantidad de libros registrados, para ello se utilizarán estructuras en forma de listas. Oracle (2025)

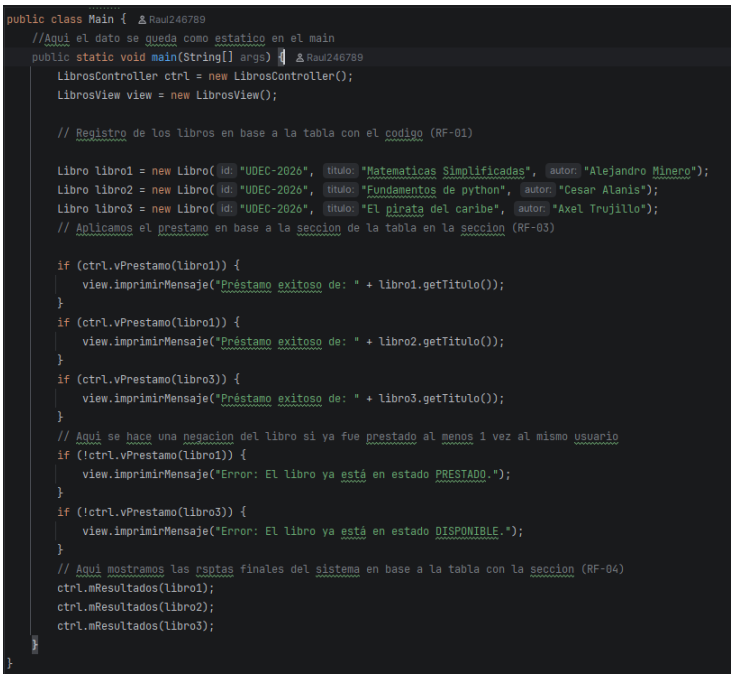
Programación Orientada a Objetos para el encapsulamiento y polimorfismo. Oracle (2025)

10. Diseño

Utilizaremos el modelo por capas para diseñar el sistema para la biblioteca de la Universidad de Cundinamarca, además de definir los paquetes a usar, en este caso usaremos el requerimiento RNF-03.

1. Model: Es la clase principal de los datos (libros)
2. Service: En esta parte va la lógica principal del programa (registros)
3. Util: Se va a usar para las validaciones del sistema (condiciones)

11. Codificación



```
public class Main {  Raul246789
    //Aqui el dato se queda como estatico en el main
    public static void main(String[] args) {  Raul246789
        LibrosController ctrl = new LibrosController();
        LibrosView view = new LibrosView();

        // Registro de los libros en base a la tabla con el codigo (RF-01)

        Libro libro1 = new Libro( id: "UDEC-2026",  titulo: "Matematicas Simplificadas",  autor: "Alejandro Minero");
        Libro libro2 = new Libro( id: "UDEC-2026",  titulo: "Fundamentos de python",  autor: "Cesar Alanis");
        Libro libro3 = new Libro( id: "UDEC-2026",  titulo: "El pirata del caribe",  autor: "Axel Trujillo");
        // Aplicamos el prestamo en base a la seccion de la tabla en la seccion (RF-03)

        if (ctrl.vPrestamo(libro1)) {
            view.imprimirMensaje("Préstamo exitoso de: " + libro1.getTitulo());
        }
        if (ctrl.vPrestamo(libro1)) {
            view.imprimirMensaje("Préstamo exitoso de: " + libro2.getTitulo());
        }
        if (ctrl.vPrestamo(libro3)) {
            view.imprimirMensaje("Préstamo exitoso de: " + libro3.getTitulo());
        }
        // Aqui se hace una negacion del libro si ya fue prestado al menos 1 vez al mismo usuario
        if (!ctrl.vPrestamo(libro1)) {
            view.imprimirMensaje("Error: El libro ya está en estado PRESTADO.");
        }
        if (!ctrl.vPrestamo(libro3)) {
            view.imprimirMensaje("Error: El libro ya está en estado DISPONIBLE.");
        }
        // Aqui mostramos las rsptas finales del sistema en base a la tabla con la seccion (RF-04)
        ctrl.mResultados(libro1);
        ctrl.mResultados(libro2);
        ctrl.mResultados(libro3);
    }
}
```

Carpeta Main.Java

```java
package org.example.LibrosView;

public class LibrosView {  3 usages  Raul246789
    public void imprimirMensaje(String mensaje) {  5 usages  Raul246789
        System.out.println("Estado actual: " + mensaje);
    }
}
```

Carpeta LibrosView.Java

```java
package org.example.LibrosController;

import org.example.LibrosModel.Libro;
import org.example.LibrosModel.EstadoLibro;
import org.example.LibrosService.ILibrosService;

public class LibrosController implements ILibrosService {  3 usages  Raul246789

    @Override  5 usages  Raul246789
    public boolean vPrestamo(Libro libro) {
        // Condi1: Los libros que esten disponibles pueden ser prestados
        if (libro.getEstado() == EstadoLibro.DISPONIBLE) {
            libro.setEstado(EstadoLibro.PRESTADO);
            return true;
        }
        // Condi2: Un libro que ya fueprestado no puede prestarse nuevamente
        return false;
    }

    @Override  3 usages  Raul246789
    public void mResultados(Libro libro) {
        System.out.println("Informacion del sistema UDec");
        System.out.println(libro.toString());
    }
}
```

Carpeta LibrosController.Java

```java
package org.example.LibrosService;

import org.example.LibrosModel.Libro;

public interface ILibrosService {  2 usages  1 implementation  Raul246789
    // RF-03: Método para validar las reglas de préstamo
    boolean vPrestamo(Libro libro);  5 usages  1 implementation  Raul246789

    // RF-04: Método para mostrar resultados
    void mResultados(Libro libro);  3 usages  1 implementation  Raul246789
}
```

Carpeta LibrosService.Java