UT5_0 – UML. INTRODUCCIÓN

RESULTADOS DE APRENDIZAJE ASOCIADOS
5. Genera diagramas de clases valorando su importancia en el desarrollo de aplicaciones y empleando las herramientas disponibles en el entorno.
CRITERIOS DE EVALUACIÓN
a) Se han identificado los conceptos básicos de la programación orientada a objetos
b) Se ha instalado el módulo del entorno integrado de desarrollo que permite la utilización de diagramas de clases
c) Se han identificado las herramientas para la elaboración de diagramas de clases.
d) Se ha interpretado el significado de diagramas de clases.
e) Se han trazado diagramas de clases a partir de las especificaciones de las mismas.
f) Se ha generado código a partir de un diagrama de clases.
g) Se ha generado un diagrama de clases mediante ingeniería inversa.

UT5_0 – UML. INTRODUCCIÓN

Índice de contenido

1.- Contenidos

En este capítulo se introducen los conceptos básicos que forman el fundamento para la comprensión de la tecnología orientada a objetos. Se hará una introducción al lenguaje UML y sus diagramas.

- Se estudiarán los diagramas de clases
- Se utilizarán herramientas para crear diagramas de clases
- Se generará código a partir de diagramas
- Se generarán diagramas a partir de código JAVA para realizar ingeniería inversa.

2.- Introducción

En el **diseño orientado a objetos** un sistema se entiende como un conjunto de objetos, los tienen propiedades y comportamientos.

Un **objeto** consta de una colección de datos y de una colección de métodos que operan sobre esos datos. Los datos definidos dentro de un objeto son sus atributos. Las operaciones que manipulan esos datos y cambian el valor de sus atributos se definen el comportamiento del objeto y consisten en operaciones. Los objetos comunican unos con otros a través del paso de mensajes.

Una **clase** no es más que una plantilla para la creación de objetos. Cuando se crea un objeto (instanciación) se le dota de una plantilla para que el objeto instantáneo, para que tenga las características del objeto.

Para el análisis y diseño orientado a objetos se utiliza **UML** (*Unified Modeling Language* o *Lenguaje de Modelado Unificado*). Es un lenguaje de modelado basado en diagramas que sirve para expresar modelos (un modelo es una representación de la realidad donde se ignoran los detalles de menor importancia). Se ha convertido en el estándar de facto de las metodologías de desarrollo de sistemas que existen hoy en día.

3.- ¿Qué es UML?

UML, o Lenguaje de Modelado Unificado, es un estándar ampliamente utilizado en la ingeniería de software para visualizar, especificar, construir y documentar sistemas complejos. Proporciona una notación gráfica que permite a los desarrolladores representar visualmente la estructura, el comportamiento, las interacciones y otros aspectos de un sistema.

Interesante video (Youtube - Píldoras informáticas)

https://www.youtube.com/watch?v=KY81igoV8W0

3.1.- Versiones de UML

UML 1.x

UML 1.x (comprende UML 1.1, 1.2, 1.3, 1.4, 1.5) fue la primera serie de versiones UML, lanzada entre 1997 y 2005.

Consistía en **13 tipos de diagramas**, organizados en cuatro categorías:

- Estructura
- Comportamiento
- Interacción
- Caso de uso

UML 1.x tenía como objetivo proporcionar una notación y semántica comunes para modelar diversos aspectos de los sistemas de software, como clases, objetos, componentes, colaboraciones, actividades, estados y escenarios.

UML 1.x fue ampliamente adoptado por desarrolladores de software, arquitectos y analistas, y apoyado por muchas herramientas y plataformas.

UML 2.x

UML 2.x fue la segunda serie de versiones UML, lanzada entre 2003 y 2017.

Introdujo cambios y mejoras significativas en UML 1.x, como:

- Agregar nuevos tipos de diagramas
- Expandir los existentes
- Refinar la sintaxis y la semántica
- Mejorar la alineación con otros estándares y lenguajes.

UML 2.x (de UML 2.0 a UML 2.5) tenía como objetivo abordar las limitaciones y desafíos de UML 1.x, como modelar sistemas complejos y dinámicos, admitir el desarrollo basado en modelos y permitir especificaciones más precisas y rigurosas.

UML 2.5 define **14 tipos de diagramas**, divididos en 3 categorías:

- 7 tipos de diagramas representan la estructura estática de la aplicación o sistema.
- 3 representan tipos generales de comportamiento
- 4 representan diferentes aspectos de las interacciones

1. Diagramas de Estructura (parte estática del modelo):

Representan la parte estática del modelo, detallando los elementos fundamentales que componen el sistema.

Incluyen el:

- diagrama de clases
- diagrama de objetos
- diagrama de componentes
- diagrama de estructura compuesta
- diagrama de paquetes
- diagrama de implementación o despliegue
- diagrama de perfil

Se enfocan en definir los elementos esenciales que deben existir en el sistema modelado.

2. Diagramas de Comportamiento (parte dinámica del modelo):

Representan la parte dinámica del modelo, describiendo cómo se comporta el sistema en diferentes situaciones.

Incluyen el:

- diagrama de casos de uso, empleado durante la recopilación de requisitos en las metodologías orientadas a objetos
- diagrama de actividad y el diagrama de estado.

Se centran en especificar las acciones y eventos que deben ocurrir en el sistema en respuesta a diversas condiciones.

3. Diagramas de Interacción:

Son derivados del diagrama de comportamiento y se centran en el flujo de control y de datos entre los elementos del sistema.

Incluyen el:

- diagrama de secuencia
- diagrama de comunicación
- diagrama de tiempos
- diagrama de vista de interacción

Proporcionan una representación detallada de cómo interactúan los elementos del sistema, facilitando la comprensión del intercambio de información y el control entre ellos.

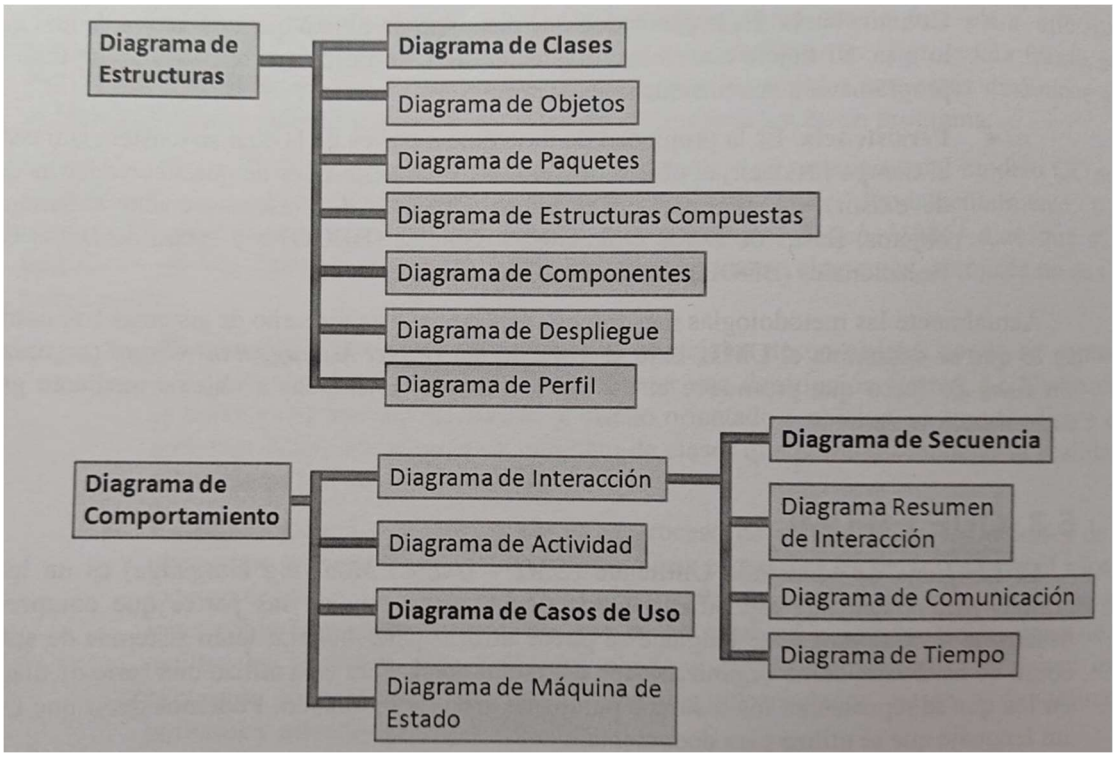


FIGURA 1. DIAGRAMAS UML 2.5. EXTRAÍDA DEL LIBRO DE ALICIA RAMOS MARTÍN

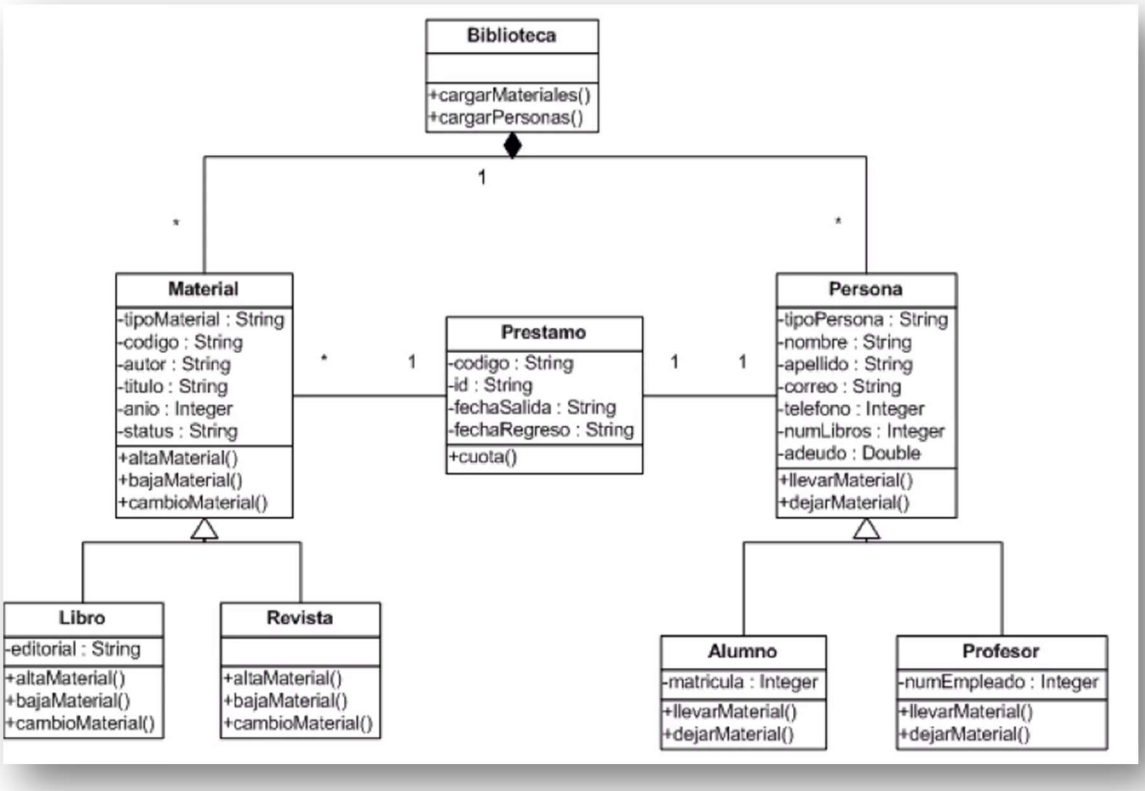
3.2.- Tipos de diagramas

Cada diagrama UML representa alguna parte o punto de vista del sistema. Los diagramas más utilizados son los siguientes:

- **Diagramas de Clase:**

  Definición: Representan la estructura estática del sistema, mostrando las clases del sistema, sus atributos, métodos y las relaciones entre ellas. Son fundamentales para comprender la arquitectura y la organización del sistema.

  Ejemplo: Supongamos que estamos diseñando un sistema de gestión de biblioteca. En el diagrama de clases, tendríamos clases como "Libro", "Usuario", "Bibliotecario", cada una con sus atributos (por ejemplo, título y autor para la clase Libro) y métodos (como "prestar" y "devolver"). Además, mostraríamos las relaciones entre las clases, como la asociación entre Libro y Usuario para representar que un usuario puede tener varios libros prestados.



- **Diagramas de Objeto:**

Definición: Muestran una instancia específica de una clase y sus relaciones con otros objetos en un momento dado. Permiten visualizar cómo interactúan los objetos en tiempo de ejecución.

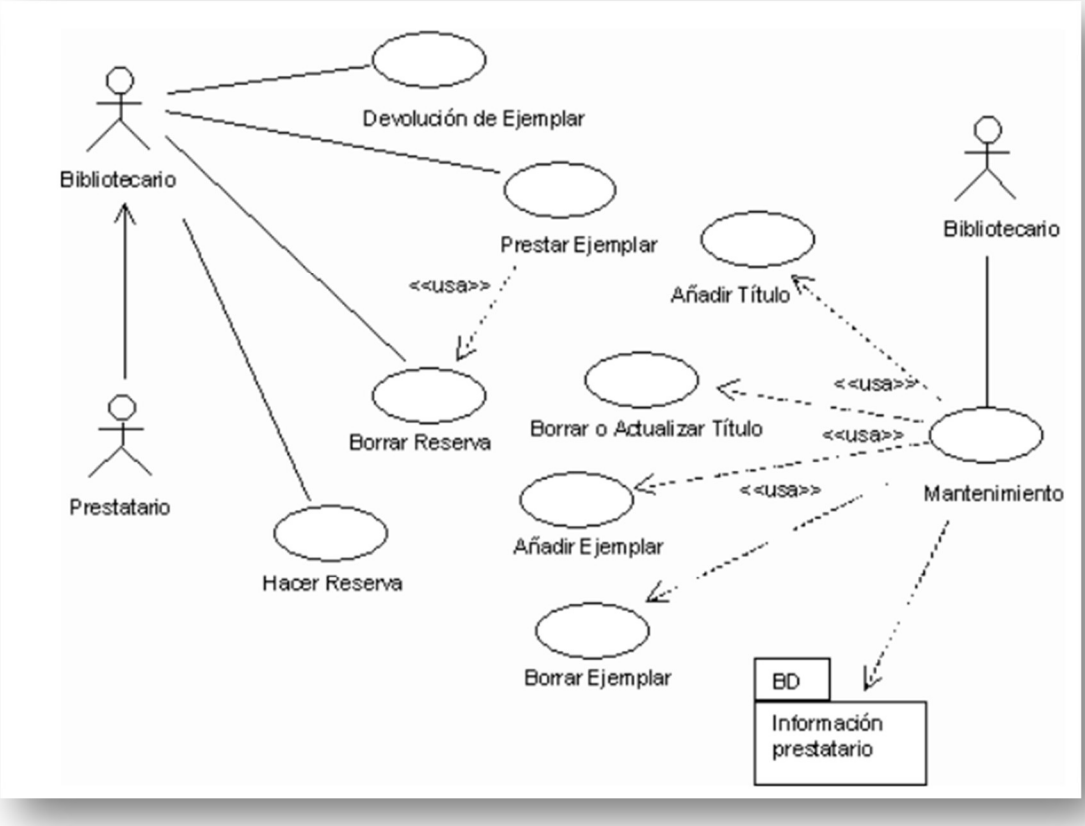
Ejemplo: Continuando con el ejemplo de la biblioteca, podríamos tener un diagrama de objeto que muestra un objeto específico de la clase Libro con sus valores de atributos (como el título y el autor) y su relación con un objeto de la clase Usuario que lo ha prestado.

- **Diagramas de Casos de Uso:**

Definición: Describen las interacciones entre un sistema y sus actores externos, representando los diferentes escenarios de uso del sistema. Son útiles para capturar los requisitos funcionales del sistema desde el punto de vista del usuario.

Ejemplo: Para el sistema de gestión de bibliotecas, podríamos tener casos de uso como "Buscar libro", "Prestar libro", "Devolver libro", cada

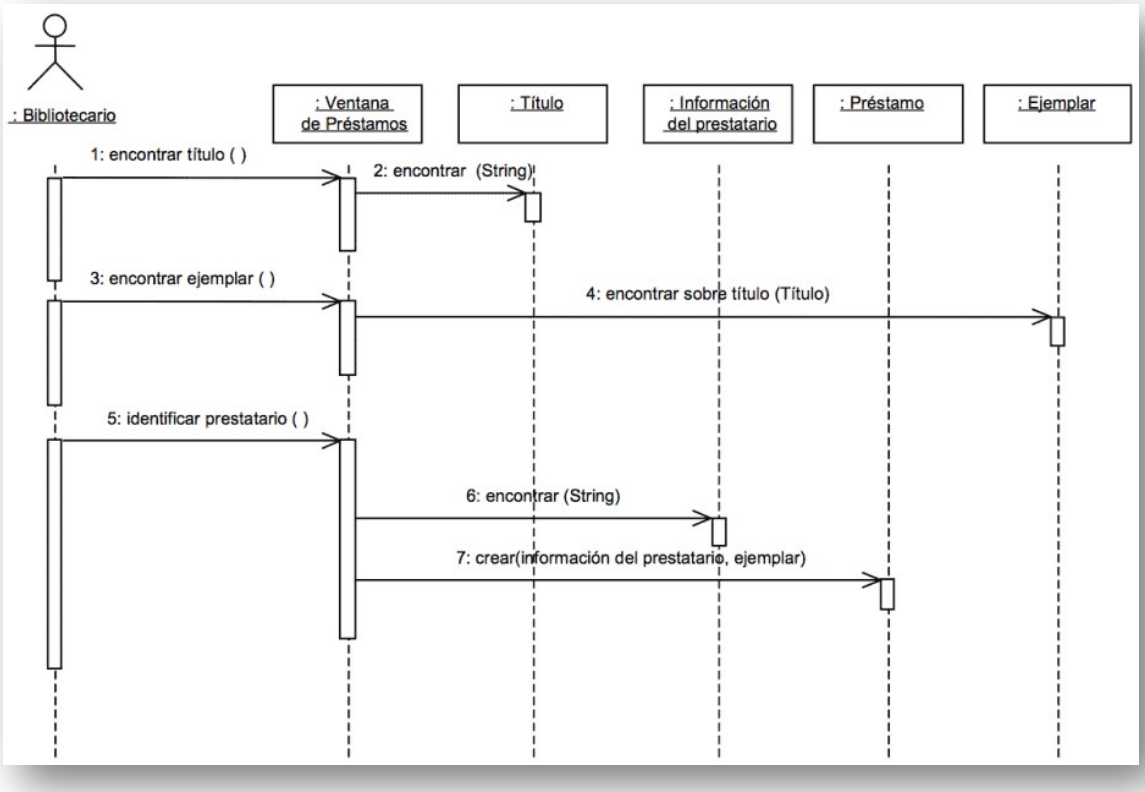
uno representando una tarea que un usuario puede realizar en el sistema. Estos casos de uso estarían conectados con los actores externos, como "Usuario" y "Bibliotecario".



- **Diagramas de Secuencia:**

Definición: Representan la secuencia temporal de interacciones entre objetos en un escenario específico, mostrando el orden en que los mensajes son enviados y recibidos. Son útiles para visualizar cómo los objetos colaboran entre sí para realizar una tarea.

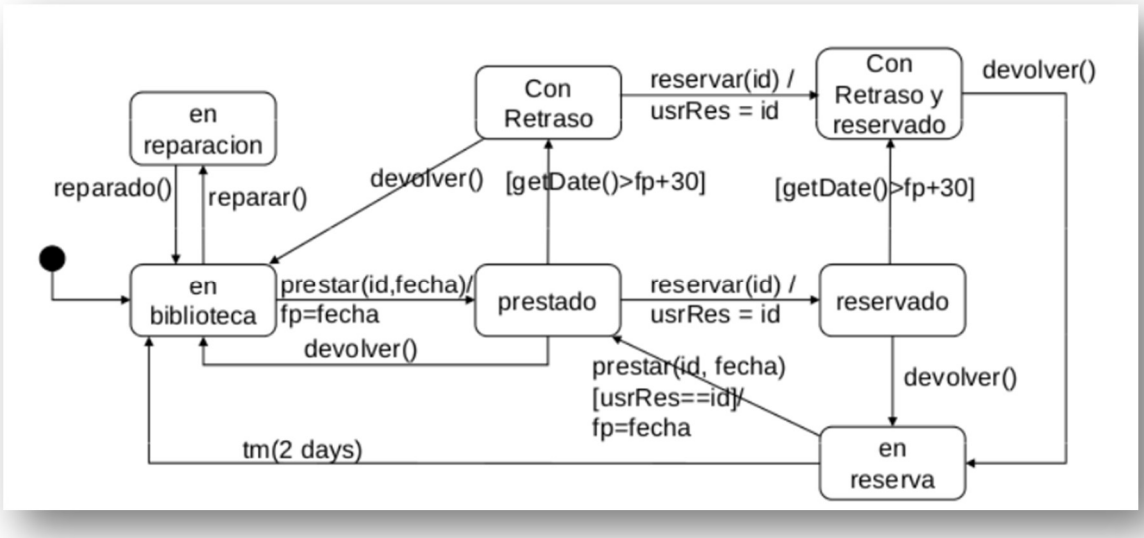
Ejemplo: En el sistema de gestión de bibliotecas, un diagrama de secuencia podría mostrar el proceso de un usuario que busca un libro, lo solicita al bibliotecario, el bibliotecario verifica la disponibilidad del libro y finalmente el usuario lo recibe.

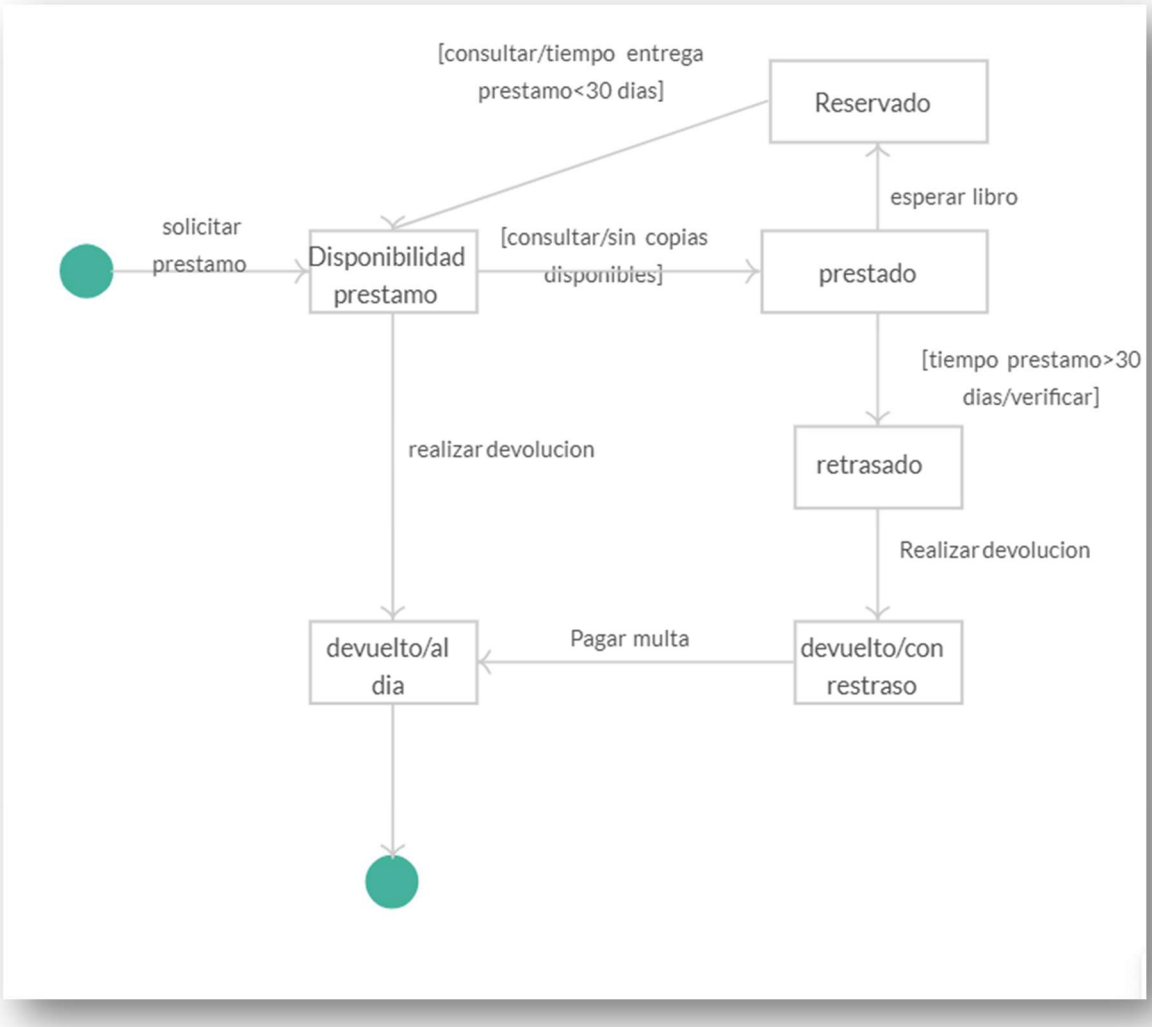


- **Diagramas de Estado:**

Definición: Modelan el comportamiento de un objeto en términos de sus estados, transiciones entre estados y eventos que desencadenan estas transiciones. Son útiles para representar el ciclo de vida de un objeto y cómo su estado cambia en respuesta a eventos externos.

Ejemplo: Para el sistema de gestión de bibliotecas, podríamos tener un diagrama de estado para un libro, mostrando los estados como "Disponible", "Prestado", "Reservado" y las transiciones entre ellos, como "Prestar libro" y "Devolver libro".

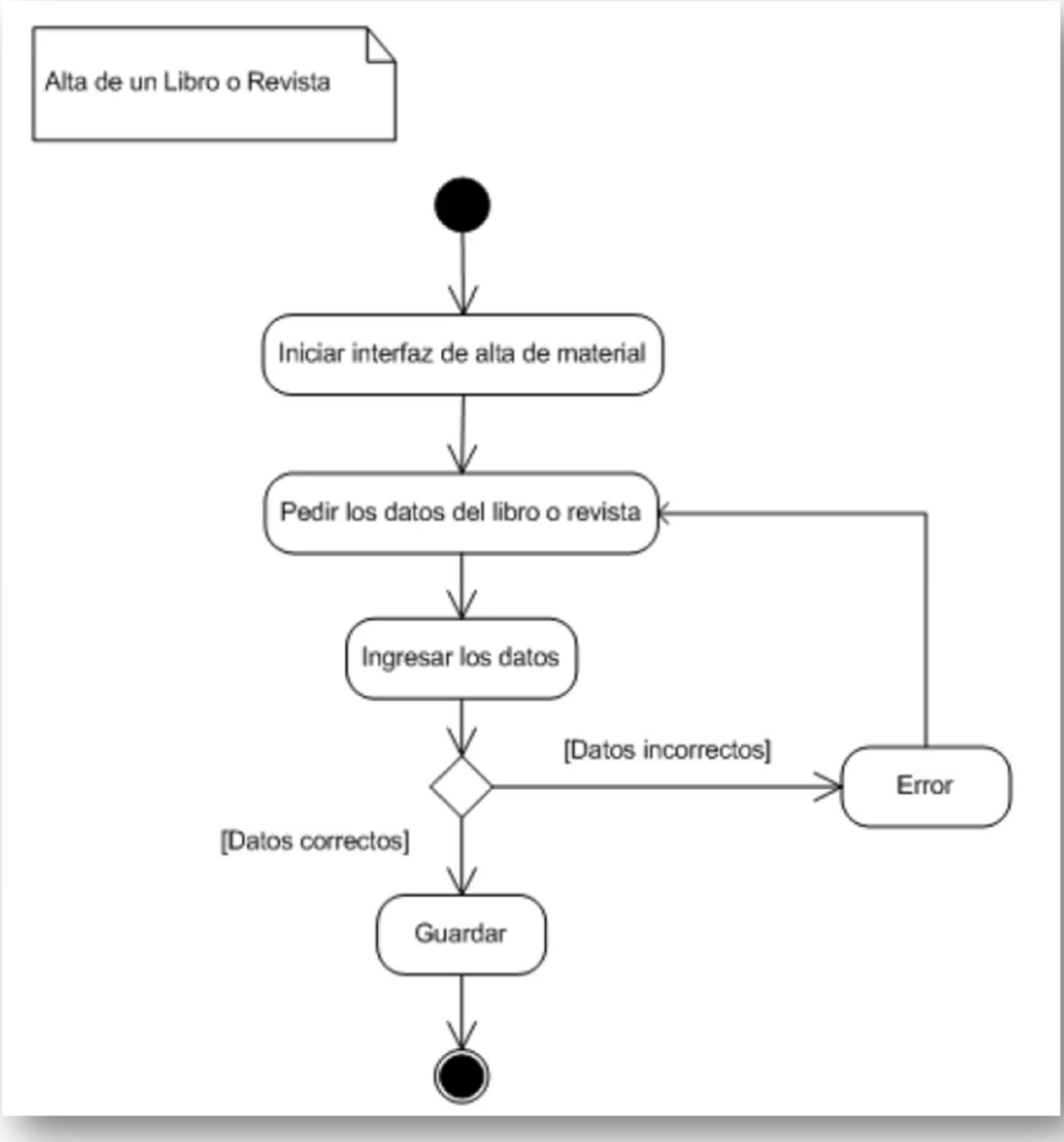

en reparacion
reparado()
reparar()
Con Retraso
reservar(id) / usrRes = id
Con Retraso y reservado
devolver()
devolver()
[getDate()>fp+30]
[getDate()>fp+30]
en biblioteca
prestar(id,fecha)/ fp=fecha
devolver()
prestado
reservar(id) / usrRes = id
reservado
prestar(id, fecha) [usrRes==id]/ fp=fecha
devolver()
tm(2 days)
en reserva


[consultar/tiempo entrega prestamo<30 dias]
Reservado
esperar libro
solicitar prestamo
Disponibilidad prestamo
[consultar/sin copias disponibles]
prestado
[tiempo prestamo>30 dias/verificar]
realizar devolucion
retrasado
Realizar devolucion
devuelto/al dia
Pagar multa
devuelto/con restraso

- **Diagramas de Actividad:**

Definición: Representan el flujo de actividades o procesos en el sistema, mostrando acciones, decisiones, bifurcaciones y uniones. Son útiles para modelar el comportamiento dinámico del sistema y los procesos de negocio.

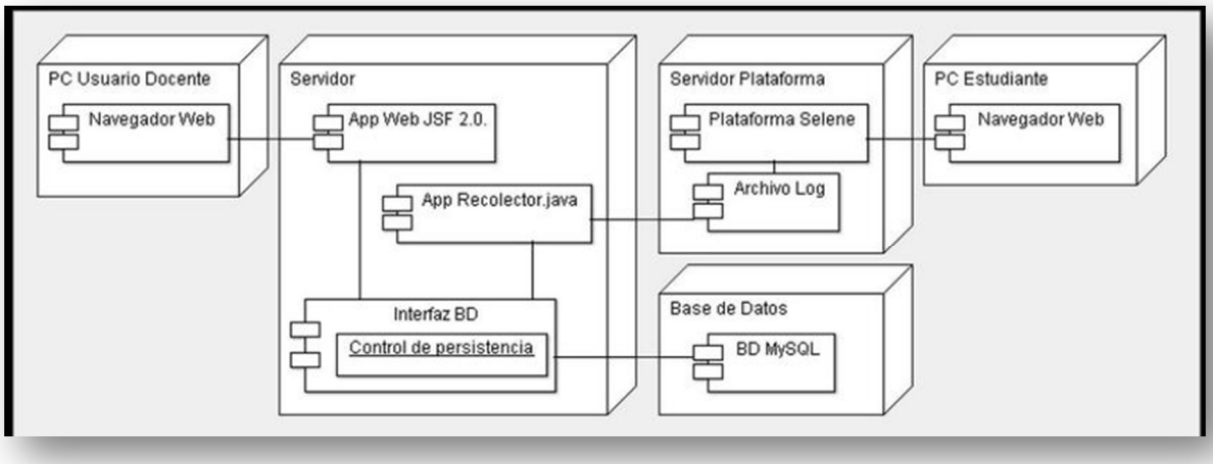
Ejemplo: Un diagrama de actividad para el sistema de gestión de bibliotecas podría mostrar el proceso de préstamo de un libro, incluyendo acciones como "Verificar disponibilidad", "Registrar préstamo" y "Actualizar inventario".



- **Diagramas de Despliegue:**

Definición: Muestran la disposición física de los componentes del sistema en el entorno de implementación, incluyendo nodos de hardware y software y las conexiones entre ellos. Son útiles para planificar la infraestructura técnica necesaria para implementar el sistema.

Ejemplo: En el contexto del sistema de gestión de bibliotecas, un diagrama de despliegue podría mostrar cómo se distribuyen los servidores, bases de datos y aplicaciones cliente en la infraestructura de la biblioteca.



4.- Referencias bibliográficas

- Moreno Pérez, J.C. *Entornos de desarrollo.* Editorial Síntesis.
- Ramos Martín, A. & Ramos Martín, M.J. *Entornos de desarrollo*. Grupo editorial Garceta.