PRÁCTICA:

“Creando diagramas UML ”

Nombre: Guerrero Carvajal Ximena

Curso Fundamentos del Desarrollo de Software – FDS2 VTG02

Nombre del profesor:

Sebastián Romero Zapata

Introducción

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se utiliza para describir la estructura y el comportamiento de sistemas de software complejos. Fue creado para proporcionar un lenguaje común y semánticamente rico para la arquitectura, el diseño y la implementación de sistemas de software.

UML se utiliza para describir la estructura y el comportamiento de sistemas de software complejos, diseñar sistemas de software y de negocios, también para documentar la estructura y el comportamiento de sistemas de software y de negocios.

Existen diagramas de clases, objetos, secuenciales y de actividades.

A continuación se mostrarán dos diagramas creados en Lucidchartn que es una herramienta en línea que permite crear diagramas UML de manera fácil y rápida. Puedes crear un diagrama UML desde cero o utilizar una plantilla predefinida.

Desarrollo

Diagrama de caso de uso

Para desarrollar este diagrama se utilizo Lucidchart, lo primero que se hizo fui ingresar a la plataforma, posteriormente se creó un documento nuevo en blanco.

Antes de crear el diagrama se reviso el contexto de como se hará el diagrama.

Sistema de gestión de biblioteca

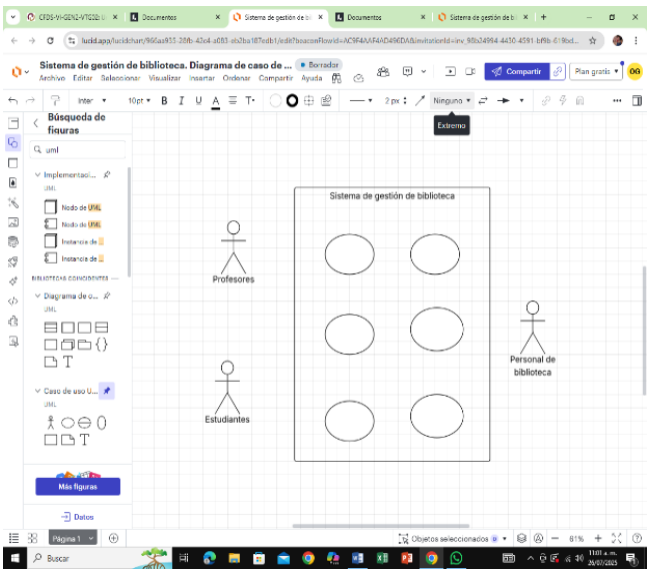
Te proponemos el siguiente escenario: un sistema de gestión de biblioteca diseñado para una universidad. Este sistema tiene como objetivo principal, facilitar el acceso a una amplia variedad de recursos, incluyendo: libros, películas y música para uso de profesores, alumnos y otros miembros de la comunidad universitaria.

El sistema permite a los usuarios buscar recursos utilizando diferentes criterios, como título, autor o género. Una vez que encuentran el recurso deseado, tienen la opción de reservarlo para su uso futuro o solicitar su préstamo inmediato. El sistema gestiona el proceso de préstamo, asegurándose de que los usuarios cumplan con los límites de préstamo establecidos y registrando el historial de préstamos de cada usuario.

Además de la gestión de préstamos, el sistema también ofrece funcionalidades administrativas. El personal de la biblioteca puede agregar, editar o eliminar recursos del catálogo, así como administrar usuarios del sistema. Esto garantiza que el catálogo de la biblioteca esté actualizado y que los usuarios tengan cuentas activas y actualizadas.

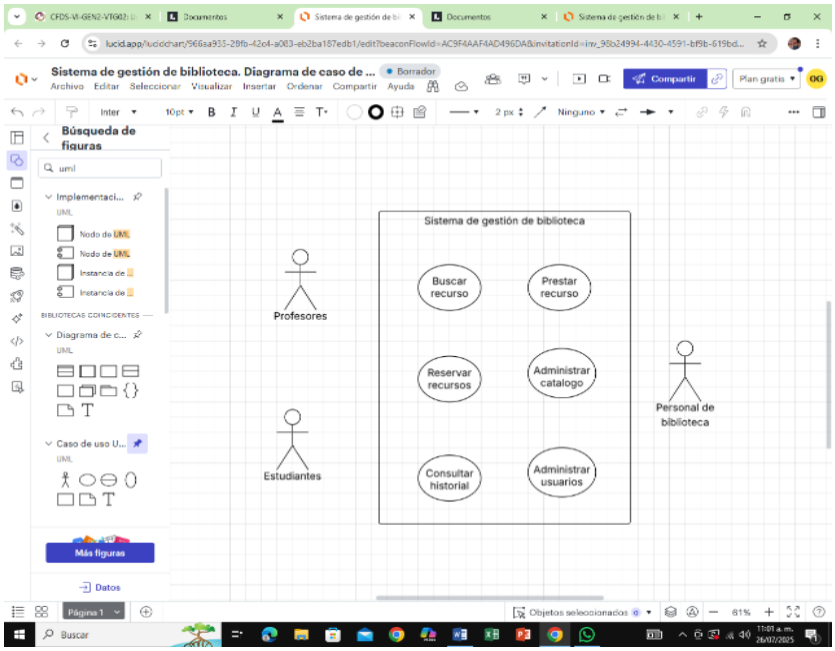
En resumen, el Sistema de Gestión de Biblioteca proporciona una plataforma centralizada para la gestión eficiente de recursos bibliográficos y multimedia, mejorando así el acceso y la experiencia de los usuarios en el entorno universitario.

Se le coloco el nombre de Sistema de gestión se biblioteca. Diagrama de casos de uso.



En el buscador de figuras se busco UML caso de uso, y se seleccionaron tres actores y un contenedor de rectángulo, a los actores se les dio el nombre de, estudiante, profesor y personal de biblioteca.

Luego se colocaron 6 figuras de tipo Caso de uso dentro del contenedor del sistema y se le coloco una acción descriptiva, la cual debe iniciar con un verbo infinito y el objeto de la acción.



Finalmente se ponen conectores de las acciones de deben hacer cada uno de los actores.

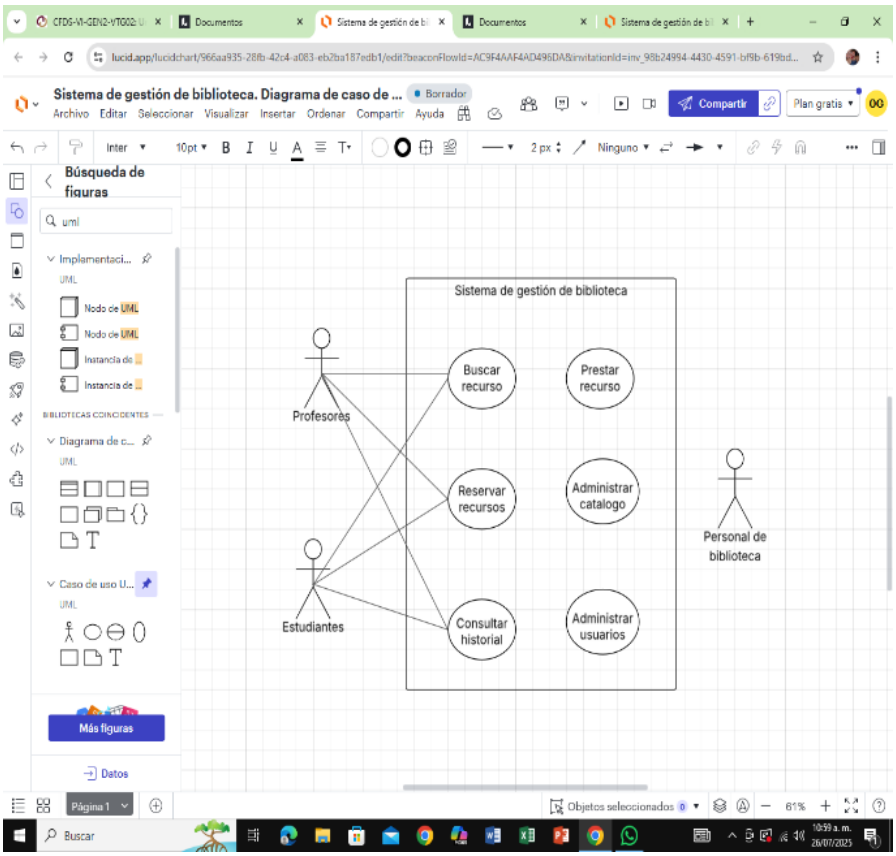


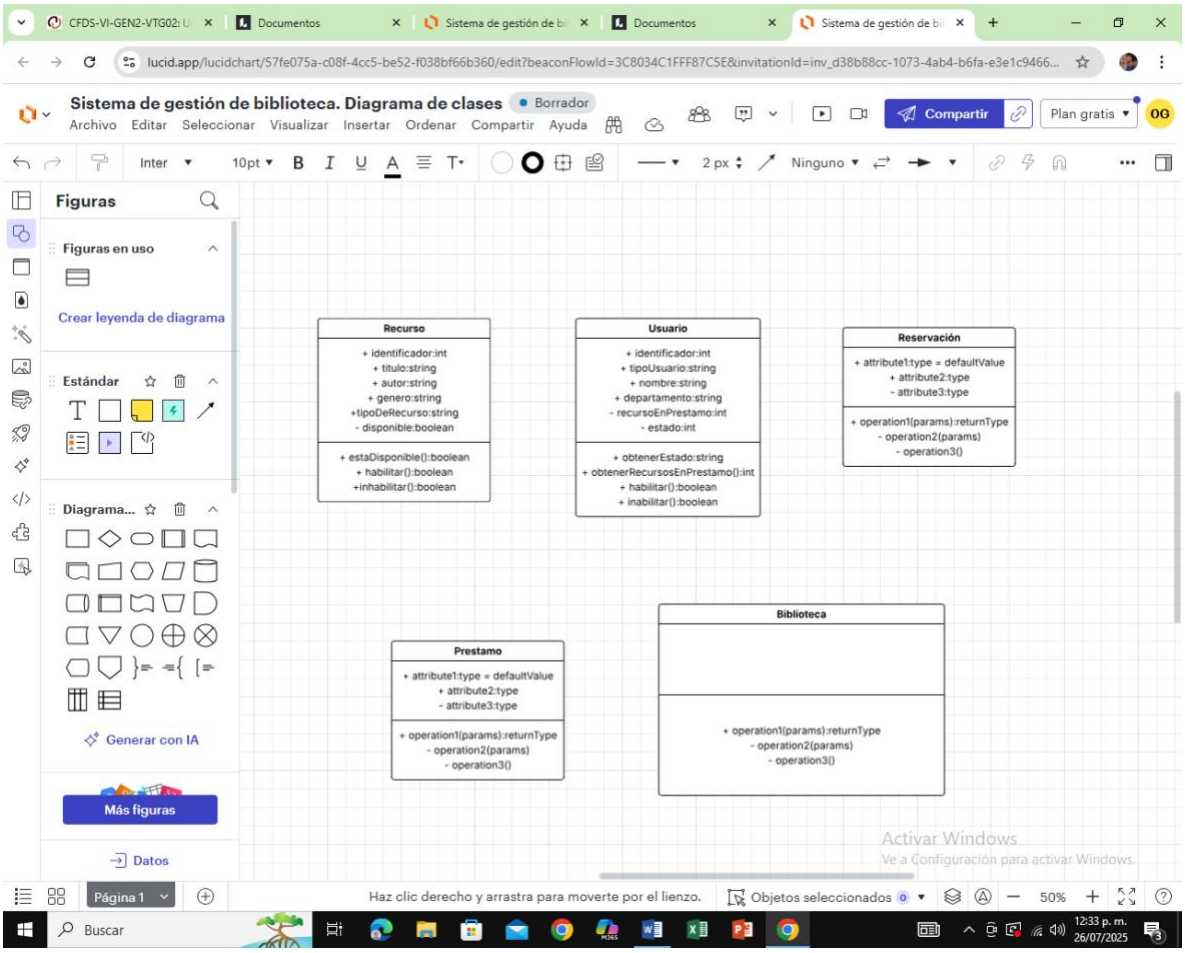
Diagrama de clases

Se desarrollo un diagrama se clase el cual es una herramienta visual que permite visualizar la estructura estática de un sistema de software, incluyendo las clases, atributos, métodos y relaciones entre ellas. Esto ayuda a comprender mejor la organización y las interacciones entre los componentes del sistema, lo que a su vez facilita el análisis y diseño de soluciones efectivas.

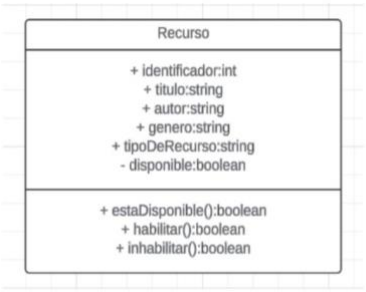
A este diagramas se le coloco en nombre de *Sistema de gestión de biblioteca. Diagrama de clases.*

En el buscador de figuras se busco diagrama UML y se selecciono diagrama de clases UML.

Se arrastraron 5 Figuras de tipo clase y se les asignará a cada uno un nombre en la parte superior como: recurso, usuario, reservación, préstamo y biblioteca.



Se agregaron las siguientes clases con sus atributos y métodos:



a. Usuario

i. Atributo público identificador de tipo int
ii. Atributo público tipoUsuario de tipo string
iii. Atributo público nombre de tipo string
iv. Atributo público departamento de tipo string
v Atributo privado recursosEnPrestamo de tipo int
vi. Atributo privado estado de tipo int
vii. Método público obtenerEstado que retorna string
viii. Método público obtenerRecursosEnPrestamo que regresa int
ix. Método público habilitar que regresa boolean
x. Método público inhabilitar que regresa boolean

b. Reservación

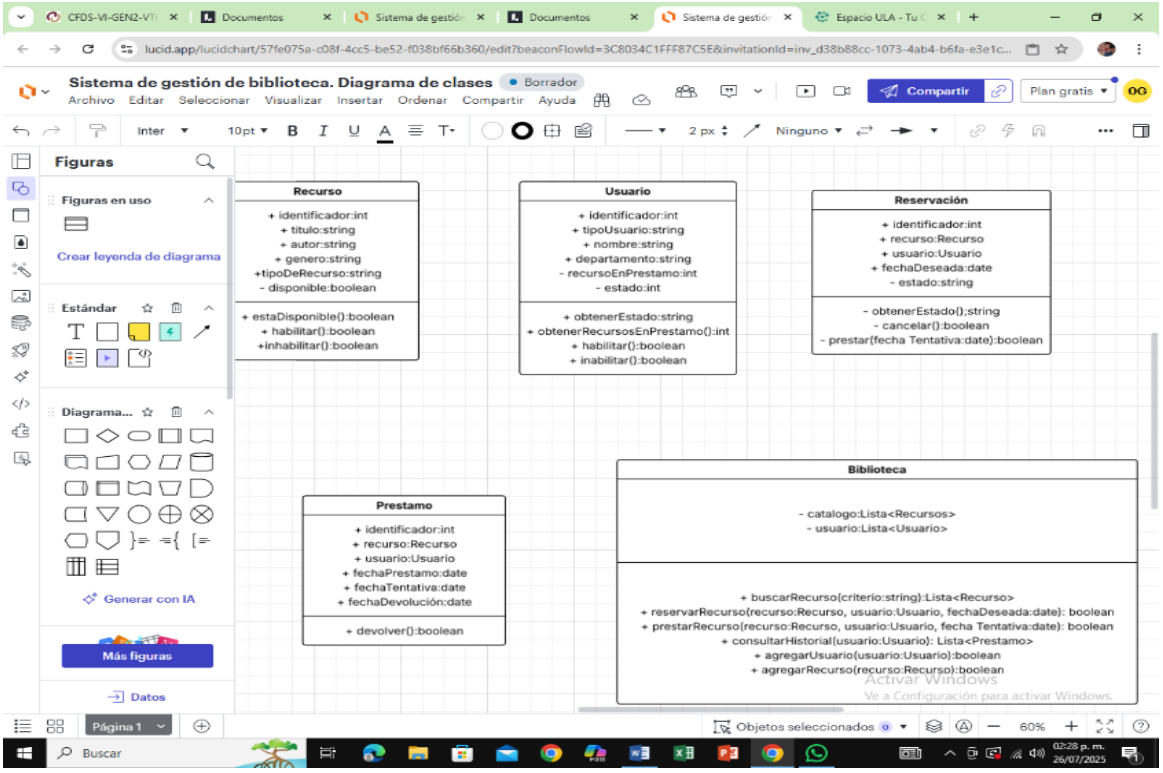
i. Atributo público identificador de tipo int
ii. Atributo público recurso de tipo Recurso
iii. Atributo público usuario de tipo Usuario
iv. Atributo público fechaDeseada de tipo date
v Atributo privado estado de tipo string
vi. Método público obtenerEstado que regresa string
vii. Método público cancelar que regresa boolean
viii. Método público prestar que recibe fechaTentativa de tipo date y regresa boolean

c. Préstamo

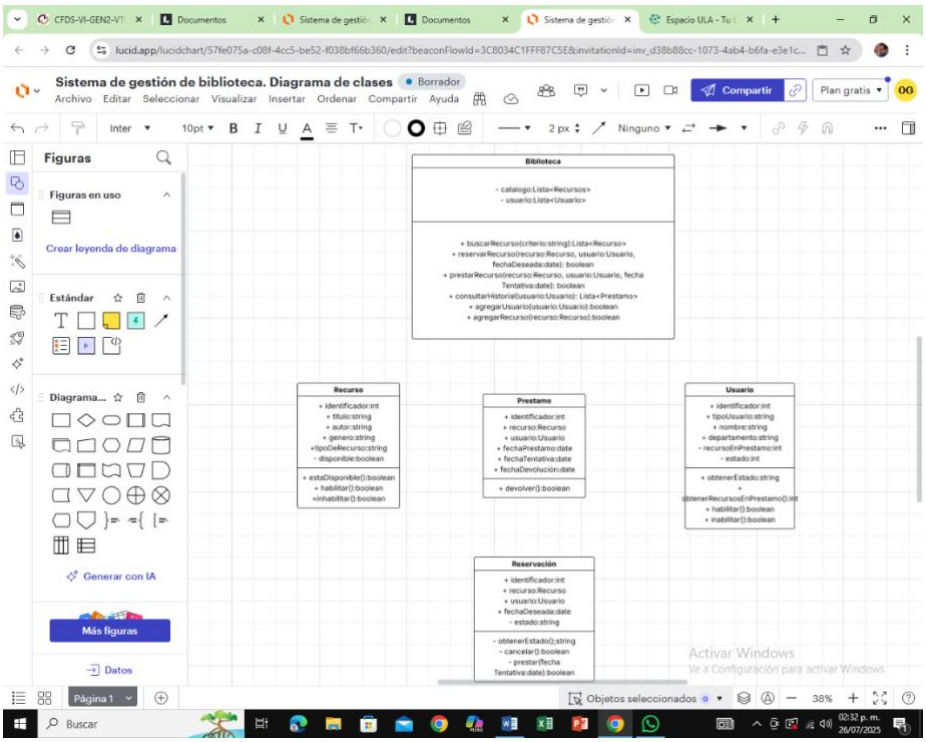
i. Atributo público identificador de tipo int
ii. Atributo público recurso de tipo Recurso
iii. Atributo público usuario de tipo Usuario
iv. Atributo público fechaPrestamo de tipo date
v Atributo público fechaTentativa de tipo date
vi. Atributo público fechaDevolucion de tipo date
vii. Método público devolver que regresa boolean

d. Biblioteca

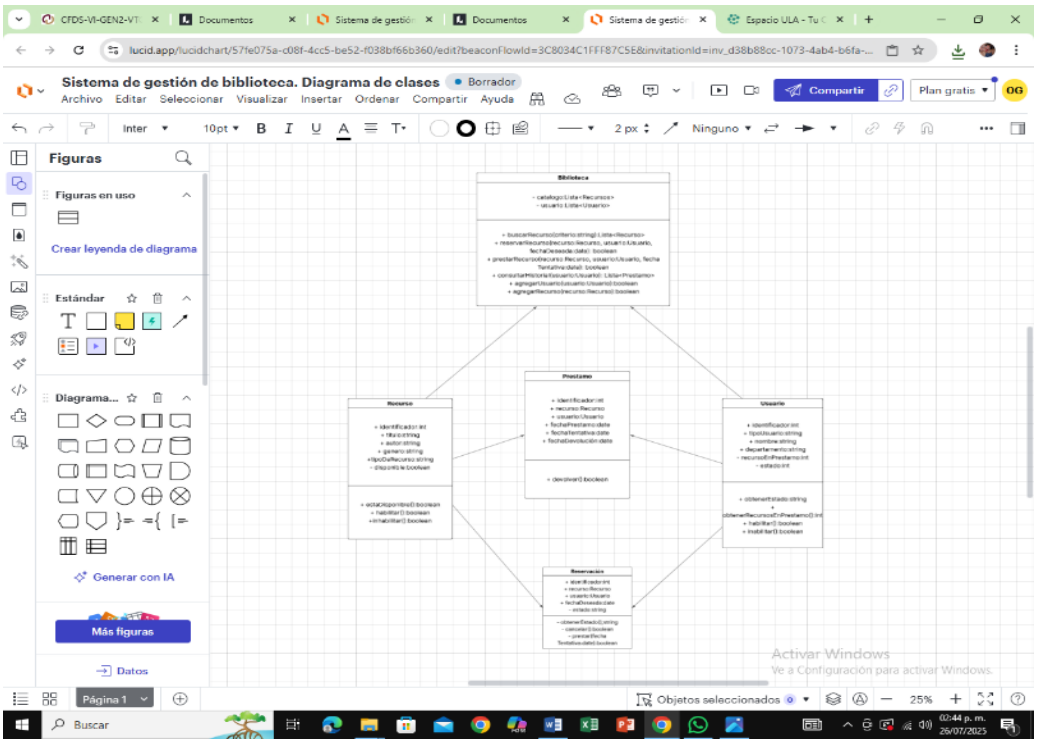
i. Atributo privado catalogo de tipo Lista<Recurso>
ii. Atributo privado usuarios de tipo Lista<Usuario>
iii. Método público buscarRecurso que recibe criterio de tipo string y regresa Lista<Recurso>
iv. Método público reservarRecurso que recibe recurso de tipo Recurso, usuario de tipo Usuario y fechaDeseada de tipo date y regresa boolean
v. Método público prestarRecurso que recibe recurso de tipo Recurso, usuario de tipo Usuario y fechaTentativa de tipo date y regresa boolean
vi. Método público consultarHistorial que recibe usuario de tipo Usuario y regresa Lista<Prestamo>
vii. Método público agregarUsuario que recibe usuario de tipo Usuario y regresa boolean
viii. Método público agregarRecurso que recibe recurso de tipo Recurso y regresa boolean



Se acomodaron las clases para poder poner los conectores



Por último se conectaron las clases para representar la clase entre ellas.



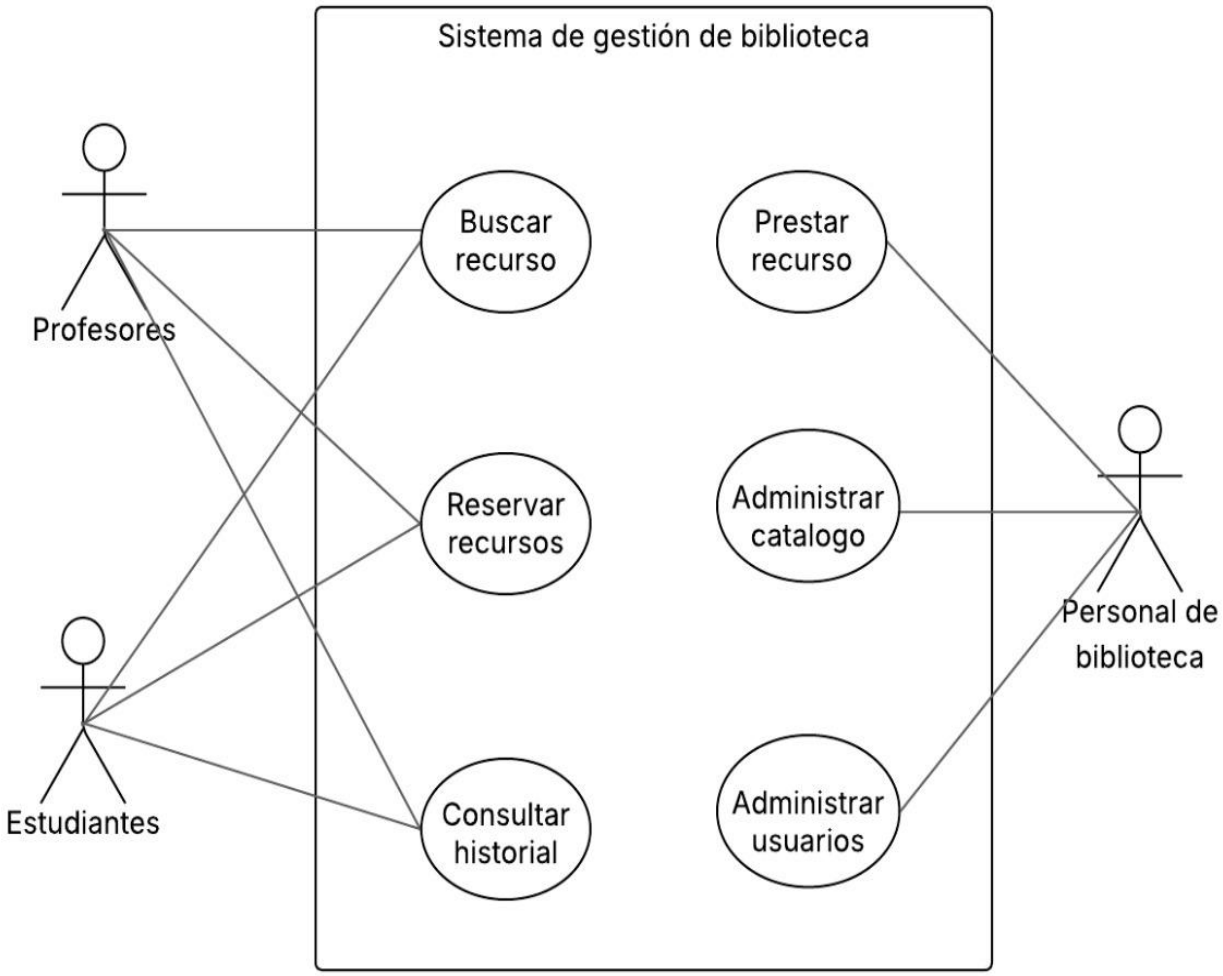
Conclusiones

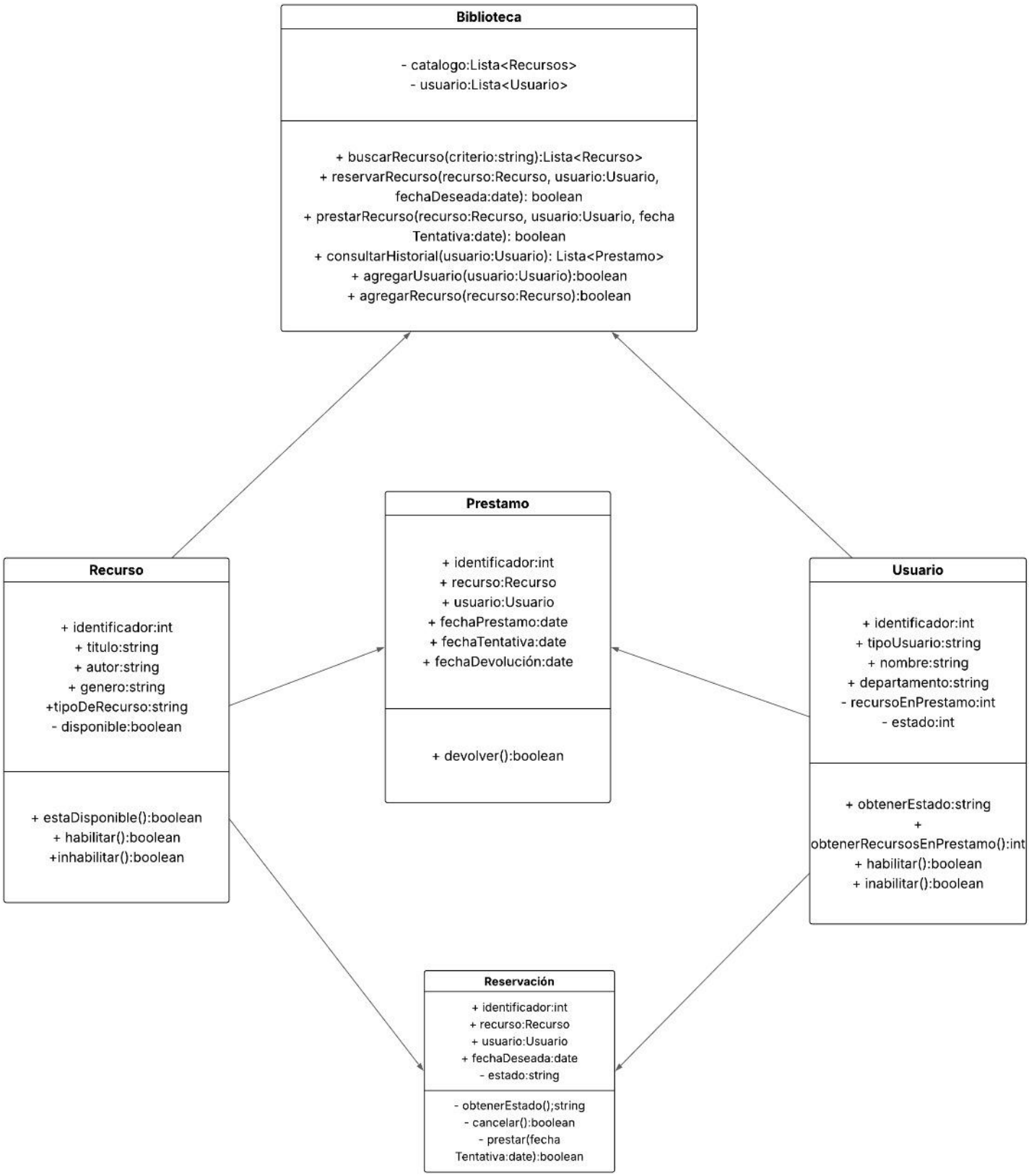
Me pareció muy interesante y útil el trabajo anterior sobre UML y la creación de diagramas con Lucidchart, los ejemplos y diagramas creados fueron muy ilustrativos y ayudaron a visualizar mejor los conceptos.

La utilización de UML permitido una mejor comprensión de la organización y las interacciones entre los componentes del sistema de gestión de biblioteca. Los diagramas UML han facilitado el análisis y diseño de soluciones efectivas para el sistema.

Es fácil de poder desarrollar los diagramas, UML y Lucidchart se complementan muy bien.

Evidencia




Biblioteca
- catalogo:Lista<Recursos>
- usuario:Lista<Usuario>
+ buscarRecurso(criterio:string):Lista<Recurso>
+ reservarRecurso(recurso:Recurso, usuario:Usuario, fechaDeseada:date): boolean
+ prestarRecurso(recurso:Recurso, usuario:Usuario, fecha Tentativa:date): boolean
+ consultarHistorial(usuario:Usuario): Lista<Prestamo>
+ agregarUsuario(usuario:Usuario):boolean
+ agregarRecurso(recurso:Recurso):boolean
Prestamo
+ identificador:int
+ recurso:Recurso
+ usuario:Usuario
+ fechaPrestamo:date
+ fechaTentativa:date
+ fechaDevolución:date
+ devolver():boolean
Recurso
+ identificador:int
+ titulo:string
+ autor:string
+ genero:string
+tipoDeRecurso:string
- disponible:boolean
+ estaDisponible():boolean
+ habilitar():boolean
+inhabilitar():boolean
Usuario
+ identificador:int
+ tipoUsuario:string
+ nombre:string
+ departamento:string
- recursoEnPrestamo:int
- estado:int
+ obtenerEstado:string
+ obtenerRecursosEnPrestamo():int
+ habilitar():boolean
+ inabilitar():boolean
Reservación
+ identificador:int
+ recurso:Recurso
+ usuario:Usuario
+ fechaDeseada:date
- estado:string
- obtenerEstado();string
- cancelar():boolean
- prestar(fecha Tentativa:date):boolean