EVALUACIÓN PARCIAL – RESOLUCIÓN DE CASOS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| FACULTAD: | Tecnología Informática | | | | |
| CARRERA: | Analista Programador | | | | |
| ALUMNO/A: | **GERARDO TORDOYA** | | | | |
| SEDE: | Buenos Aires | | LOCALIZACIÓN: | Distancia | |
| ASIGNATURA: | Metodología de Desarrollo de Sistemas II. | | | | |
| CURSO: | T4-17-15 2-K-D | | TURNO: | Distancia | |
| PROFESOR: | Leonel Jiménez G. | | FECHA: | 27/05/2023 | |
| TIEMPO DE RESOLUCIÓN: | | 6 dias. | EXAMEN PARCIAL NRO: | | 1 |
| MODALIDAD DE RESOLUCIÓN: | | | Domiciliario – Individual. | | |
| Resultados de aprendizaje:  [Modela]+ [Casos de uso]+ [Para representar los requisitos funcionales] + [Utilizando el estándar UML]  [Modela]+ [Diagramas de clases]+ [Para representar la estructura estática del sistema] + [Aplicando el estándar UML]   * [Modela]+ [Diagramas de interacción]+ [Para representar la interacción dinámica del sistema] + [Aplicando el estándar UML] | | | | | |

Los estudiantes encontrará el examen habilitado en la plataforma ULTRA a partir de la fecha y hora indicada por el profesor.

Criterios de calificación: Para acreditar los saberes deberá obtener, al menos, el 60% de los aspectos conceptuales, además de, al menos, el 60% de los aspectos procedimentales. La calificación final se obtendrá luego de la defensa oral del trabajo presentado.

Criterios de evaluación: Se evaluará la claridad en el planteamiento de los aspectos conceptuales y procedimentales. La evaluación se hará a partir de la siguiente grilla:

|  |  |  |
| --- | --- | --- |
| Criterio | Calificación | Observaciones |
| Instancia oral |  |  |
| Aspectos Conceptuales | | |
| Pregunta 1 |  |  |
| Pregunta 2 |  |  |
| Pregunta 3 |  |  |
| Pregunta 4 |  |  |
| Pregunta 5 |  |  |
| Pregunta 6 |  |  |
| Pregunta 7 |  |  |
| Pregunta 8 |  |  |
| Aspectos procedimentales | | |
| Pregunta 1 |  |  |
| Pregunta 2 |  |  |
| Pregunta 3 |  |  |
| Pregunta 4 |  |  |
| Pregunta 5 |  |  |
|  | | |
| Calificación final |  |  |

Forma de entrega del examen

* Se deberá resolver el examen sobre el mismo archivo del examen original.
* Se deberá subir al ULTRA el examen ya resuelto en un archivo tipo ZIP o RAR.

Aspectos conceptuales

1. ¿Cuál es la diferencia entre análisis, diseño e implementación? [15 Puntos]

El análisis OO pone énfasis en la investigación del problema y los requisitos, en vez de ponerlo en la solución. Es decir, define el problema: En esta etapa se estudia y comprende el problema o la necesidad a resolver. Se recopila información, se identifican los requisitos y se analizan los procesos y las funcionalidades que el sistema debe cumplir. El objetivo es obtener una visión clara de lo que se espera del sistema y de las necesidades de los usuarios.

El diseño pone énfasis en una solución conceptual, que satisface los requisitos, en vez de ponerlo en la implantación. Es decir, propone una solución al problema: En esta etapa se define la estructura y el funcionamiento del sistema a partir de los requisitos establecidos en la etapa de análisis. Se determina cómo se organizarán los componentes del sistema, cómo interactuarán entre sí y cómo se presentará la interfaz de usuario. Además, se definen los algoritmos y las estructuras de datos que se utilizarán en el desarrollo del sistema.

La implantación es la traducción de la solución a un lenguaje de programación determinado. Es decir, construye la solución propuesta: En esta etapa se lleva a cabo la construcción del sistema de acuerdo con el diseño establecido. Se traducen las especificaciones y los diseños en código fuente utilizando un lenguaje de programación. También se realizan pruebas para verificar que el sistema funcione correctamente y se corrigen los errores encontrados. Finalmente, se documenta el código y se prepara el sistema para su despliegue y puesta en marcha.

1. ¿Qué tipos de requisitos conoce? Explique y Ejemplifique. [10 Puntos]

**Requisitos Funcionales:** Servicios que el sistema debe proporcionar: Son aquellos que describen las acciones o funciones específicas que el sistema debe ser capaz de realizar. Estos requisitos se centran en qué debe hacer el sistema y cómo debe comportarse en diferentes situaciones. Por ejemplo, un requisito funcional podría ser "el sistema debe permitir a los usuarios iniciar sesión con su nombre de usuario y contraseña".

**Requisitos no funcionales:** Restricciones que afectan al sistema: Son requisitos que se refieren a las características y propiedades del sistema que no están directamente relacionadas con su funcionalidad, sino más bien con aspectos de rendimiento, seguridad, usabilidad, mantenibilidad, entre otros. Algunos ejemplos de requisitos no funcionales son "el sistema debe ser capaz de manejar una carga de 1000 usuarios concurrentes" o "la interfaz de usuario debe ser intuitiva y fácil de usar".

**Requisitos de Usuarios:** Necesidades que los usuarios expresan verbalmente. Es decir, se centran en las necesidades y expectativas de los usuarios finales del sistema. Los requisitos de usuarios, en definitiva, son requisitos que pertenecen a los requisitos tanto funcionales como no funcionales:

* **Requisitos de funcionalidad:** Describen las acciones específicas que los usuarios desean realizar en el sistema. Por ejemplo, "los usuarios deben poder agregar productos al carrito de compras" o "los usuarios deben poder buscar y filtrar los resultados".
* **Requisitos de Usabilidad:** Se refieren a la facilidad de uso y la experiencia del usuario. Estos requisitos están relacionados con la interfaz de usuario, la navegación, la presentación de la información y la interacción con el sistema.
* **Requisitos de Rendimiento:** Se refieren a las características de desempeño que debe tener el sistema, como tiempos de respuesta, capacidad de procesamiento, escalabilidad, etc. Estos requisitos definen cómo debe comportarse el sistema en términos de eficiencia y rendimiento.
* **Requisitos de Seguridad:** Son aquellos que establecen los mecanismos y controles de seguridad que el sistema debe tener para proteger la información y los recursos. Incluyen aspectos como el control de acceso, la encriptación de datos, la gestión de roles y permisos, entre otros.

**Requisitos del Sistema:** Son los componentes que el sistema debe tener para realizar determinadas tareas. A diferencia de los requisitos de usuario (que se centran en las necesidades y expectativas de los usuarios finales) los requisitos de sistema se enfocan en los aspectos técnicos y organizativos del sistema en sí:

* **Requisitos de arquitectura:** Describen la estructura y el diseño general del sistema, incluyendo los componentes, la interacción entre ellos y los estándares tecnológicos a utilizar.
* **Requisitos de rendimiento:** Establecen los objetivos y las métricas de rendimiento que el sistema debe cumplir, como tiempos de respuesta, capacidad de procesamiento, escalabilidad, entre otros.
* **Requisitos de seguridad:** Especifican los mecanismos y controles de seguridad que el sistema debe implementar para proteger la información y los recursos, cumpliendo con los estándares y regulaciones aplicables.
* **Requisitos de integración:** Se refieren a la capacidad del sistema de interoperar con otros sistemas o componentes externos, definiendo los protocolos y formatos de intercambio de datos.
* **Requisitos de mantenibilidad:** Establecen criterios y prácticas que permiten asegurar la facilidad de mantenimiento y evolución del sistema a lo largo del tiempo, como la modularidad del código, la documentación adecuada y las pruebas automatizadas.

OTRA RESPUESTA:

Los requisitos se pueden clasificar en tres tipos: requisitos funcionales, requisitos no funcionales y requisitos de restricciones.

Los **requisitos funcionales** son aquellos que describen las funciones que el sistema debe realizar. Por ejemplo, si se está desarrollando un sistema de reservas de vuelos, un requisito funcional podría ser “el sistema debe permitir a los usuarios buscar vuelos disponibles y reservarlos”.

Los **requisitos no funcionales** son aquellos que describen las características del sistema que no están relacionadas con su funcionalidad. Por ejemplo, si se está desarrollando un sistema de reservas de vuelos, un requisito no funcional podría ser “el sistema debe ser fácil de usar”.

Los **requisitos de restricciones** son aquellos que describen las limitaciones que el sistema debe cumplir. Por ejemplo, si se está desarrollando un sistema de reservas de vuelos, un requisito de restricción podría ser “el sistema debe estar disponible las 24 horas del día”.

1. Explique el concepto de escenario en el contexto de los casos de uso. Relacione este concepto con el uso de los diagramas de secuencia. [10 Puntos]

Presumo que se refiere a la especificación de casos de uso. Y teniendo en mente que un caso de uso es, ante todo, requisitos funcionales, entonces un escenario se refiere a una secuencia específica de acciones y eventos que ocurren durante la ejecución de un caso de uso. Los casos de uso se utilizan para capturar los requisitos funcionales del sistema, describiendo las interacciones entre los actores (usuarios, sistemas externos) y el sistema en sí. Un caso de uso puede tener varios escenarios, y cada escenario representa una instancia particular de la ejecución del caso de uso. Cada escenario se compone de una secuencia de pasos y acciones que describen cómo interactúan los actores con el sistema para lograr un objetivo específico.

Los diagramas de secuencia, por otro lado, son diagramas UML que representan visualmente la interacción entre objetos en un sistema. Estos diagramas muestran la secuencia de mensajes que se intercambian entre los objetos a lo largo del tiempo para cumplir con un escenario o un caso de uso en particular.

La relación entre los escenarios y los diagramas de secuencia radica en que los escenarios proporcionan la base para crear los diagramas de secuencia. Los pasos y las acciones descritas en los escenarios se traducen en los mensajes intercambiados entre los objetos representados en el diagrama de secuencia. Así, un escenario **describe** la secuencia de acciones en lenguaje natural, mientras que el diagrama de secuencia **representa** (proporciona una representación visual) de esa secuencia utilizando objetos y mensajes. El diagrama de secuencia ayuda a visualizar el flujo de interacción entre los objetos involucrados en un caso de uso y proporciona una visión más detallada de cómo se lleva a cabo cada paso en el escenario.

En resumen, escenarios y diagramas de secuencia están estrechamente relacionados ya que los escenarios proporcionan la base para construir los diagramas de secuencia (los cuales ayudan a visualizar la interacción entre objetos durante la ejecución de un caso de uso específico).

1. ¿Cuál es la diferencia entre mensajes, operaciones y método? Ejemplifique [10 Puntos]

**Mensajes:** son solicitudes para que se lleve a cabo la operación indicada y se produzca el resultado. En un diagrama de secuencia, los mensajes representan la comunicación o interacción entre los objetos participantes en un escenario o flujo de ejecución. Un mensaje en un diagrama de secuencia indica que un objeto envía un estímulo a otro objeto, solicitando que realice una acción o ejecute una operación específica.

**Operaciones:** son funciones o transformaciones que se aplican a un objeto de una clase . En un diagrama de clases, las operaciones se representan en forma de métodos en las clases. Una operación es una acción o comportamiento definido en la interfaz de una clase y se representa mediante el nombre de la operación junto con los parámetros y el tipo de retorno si corresponde. Las operaciones indican las funcionalidades disponibles en una clase, pero no especifican la implementación detallada de dichas funcionalidades.

**Métodos:** son las implementaciones de una operación. En un diagrama de clases, los métodos se representan dentro de las clases como los bloques de construcción de la implementación concreta de una operación. Los métodos se indican mediante su nombre, parámetros, tipo de retorno y posiblemente otras características como la visibilidad y los modificadores. Los métodos proporcionan detalles sobre cómo se implementa una operación en una clase específica.

1. ¿Cuáles son las diferencias entre clase, clase abstracta e interfaz? [15 Puntos]

**Clase:** Una clase es una plantilla o molde para crear objetos. Define las propiedades y comportamientos que tienen en común los objetos de ese tipo. Una clase describe los atributos (variables) y los métodos (comportamientos) que los objetos de esa clase pueden tener. Las clases se utilizan para crear instancias de objetos, es decir, son los planos a partir de los cuales se crean los objetos reales.

**Clase abstracta:** Una clase abstracta es una clase que no puede ser instanciada directamente, sino que se utiliza como base para otras clases derivadas. Proporciona una abstracción común para un grupo de clases relacionadas. Una clase abstracta puede contener métodos con implementación y métodos abstractos. Los métodos abstractos son declarados en la clase abstracta pero no tienen una implementación definida. Las clases derivadas que heredan de una clase abstracta deben implementar los métodos abstractos definidos por esa clase. Las clases abstractas se utilizan para crear jerarquías de clases donde se comparten características comunes.

**Interfaz:** Una interfaz define un conjunto de métodos que una clase debe implementar. Es un contrato que una clase acepta cumplir al implementar los métodos definidos en la interfaz. Una interfaz proporciona una especificación de los métodos que una clase debe tener sin especificar su implementación. Una clase puede implementar múltiples interfaces, lo que permite la implementación de múltiples conjuntos de comportamientos. Las interfaces se utilizan para lograr la abstracción y el polimorfismo, permitiendo que los objetos se traten de manera genérica a través de interfaces comunes.

1. ¿Qué es el encapsulamiento? Explique cómo se implementa. Ejemplifique. [10 Puntos]

El encapsulamiento **es** uno de los principios fundamentales de la programación orientada a objetos. Consiste en ocultar los detalles internos de un objeto y exponer solo una interfaz[[1]](#footnote-1) que permita interactuar con él. El encapsulamiento combina datos y métodos relacionados en una sola entidad, lo que proporciona una capa de protección y facilita el mantenimiento y la modificación del código.

El encapsulamiento **se implementa** mediante el uso de modificadores de acceso en la definición de clases y miembros de clase. Estos modificadores de acceso, como "***public***", "***private***" y "***protected***", controlan la visibilidad de los miembros de una clase (es decir, quién puede acceder a ellos):

* **Public:** Un miembro público es accesible desde cualquier parte del programa. Puede ser accedido directamente tanto desde dentro de la clase como desde otras clases.
* **Private:** Un miembro privado es accesible solo desde dentro de la clase en la que se define. No se puede acceder a él directamente desde fuera de la clase.
* **Protected:** Un miembro protegido es accesible desde dentro de la clase en la que se define y desde sus subclases (clases que heredan de ella).

NOTA: Al definir una clase, se utilizan los modificadores de acceso para controlar la visibilidad de los atributos y métodos de la clase. Los atributos se suelen declarar como privados y se accede a ellos mediante métodos públicos, conocidos como "métodos de acceso" (***getters*** y ***setters***), que proporcionan una interfaz controlada para leer y modificar los valores de los atributos.

Ejemplo de encapsulamiento:

public class CuentaBancaria {

    private double saldo;

    public void depositar(double cantidad) {

        // Lógica para realizar el depósito

        saldo += cantidad;

    }

    public void retirar(double cantidad) {

        // Lógica para realizar el retiro

        if (cantidad <= saldo) {

            saldo -= cantidad;

        } else {

            System.out.println("Saldo insuficiente");

        }

    }

    public double getSaldo() {

        return saldo;

    }

}

En este ejemplo, la clase **CuentaBancaria** encapsula el saldo como un atributo privado. Los métodos depositar y retirar proporcionan la interfaz pública para interactuar con el objeto y modificar el saldo de manera controlada. El método **getSaldo** es un método de acceso que permite obtener el valor del saldo, pero no permite modificarlo directamente. De esta manera, el encapsulamiento asegura que el saldo solo pueda ser modificado a través de métodos específicos y protege la integridad de los datos de la clase.

1. Explique y relacione los siguientes conceptos: Herencia, Generalización, Especialización. [15 Puntos]

La herencia permite que una clase herede características y comportamientos de otra clase, estableciendo una relación de "es-un" (el común denominador entre ellas). La generalización implica la creación de una clase más general a partir de clases más específicas, mientras que la especialización consiste en crear clases más específicas a partir de una clase base más general. Estos conceptos son fundamentales para establecer jerarquías de clases y permiten una mayor reutilización y extensibilidad del código.

**Herencia:** La herencia es un mecanismo mediante el cual una clase (llamada clase derivada o subclase) puede heredar características y comportamientos de otra clase (llamada clase base o superclase). La clase derivada tiene acceso a los atributos y métodos de la clase base y puede extenderlos o modificarlos según sea necesario. La herencia permite la reutilización de código y establece una relación de "es-un" entre las clases.

**Generalización:** La generalización es un concepto relacionado con la herencia y se refiere a la creación de una clase más general a partir de clases más específicas. En la generalización, se identifican características y comportamientos comunes en varias clases y se los extrae para formar una clase base más abstracta. Las clases específicas se convierten en subclases de la clase generalizada, heredando sus propiedades y comportamientos comunes. La generalización ayuda a establecer jerarquías de clases y a capturar la relación de "es-un" entre ellas.

**Especialización:** La especialización es el proceso opuesto a la generalización. Consiste en crear clases más específicas a partir de una clase base más general. Las clases especializadas (subclases) heredan las características y comportamientos de la clase base, pero también pueden tener sus propias características adicionales o modificar los comportamientos heredados. La especialización permite crear clases más concretas y específicas que amplían o adaptan la funcionalidad de la clase base.

1. ¿Qué es el polimorfismo? Explique y ejemplifique. [15 Puntos]

El polimorfismo es un principio de la programación orientada a objetos que permite que objetos de diferentes clases se comporten de manera similar en función del contexto en el que se utilicen. El polimorfismo permite tratar objetos de diferentes clases de manera genérica, lo que facilita la flexibilidad y la reutilización del código. El polimorfismo se basa en la capacidad de las clases derivadas de una clase base para proporcionar su propia implementación de los métodos heredados de la clase base. Esto significa que un método de la clase base puede tener diferentes comportamientos en las diferentes clases derivadas.

Un ejemplo común de polimorfismo es el uso de un método común, como "dibujar()", en diferentes clases que representan formas geométricas. Cada clase derivada de la clase base (por ejemplo, "Círculo", "Cuadrado" y "Triángulo") puede implementar su propia versión del método "dibujar()" de acuerdo con su forma específica:

public abstract class Forma {

    public abstract void dibujar();

}

public class Circulo extends Forma {

    @Override

    public void dibujar() {

        System.out.println("Dibujando un círculo");

    }

}

public class Cuadrado extends Forma {

    @Override

    public void dibujar() {

        System.out.println("Dibujando un cuadrado");

    }

}

public class Triangulo extends Forma {

    @Override

    public void dibujar() {

        System.out.println("Dibujando un triángulo");

    }

}

public class Main {

    public static void main(String[] args) {

        Forma forma1 = new Circulo();

        Forma forma2 = new Cuadrado();

        Forma forma3 = new Triangulo();

        forma1.dibujar(); // Salida: Dibujando un círculo

        forma2.dibujar(); // Salida: Dibujando un cuadrado

        forma3.dibujar(); // Salida: Dibujando un triángulo

    }

}

En este ejemplo, la clase abstracta **Forma** define el método abstracto dibujar(). Las clases derivadas **Circulo**, **Cuadrado** y **Triangulo** implementan este método de manera específica para dibujar cada forma en particular. Al tratar los objetos de estas clases como objetos de tipo **Forma**, podemos llamar al método **dibujar()** de manera genérica y el polimorfismo se encargará de ejecutar la implementación específica en función del tipo de objeto en tiempo de ejecución.

Aspectos procedimentales

Del siguiente enunciado que se indica al final, determinar:

1. Los Requerimientos Funcionales planteados. [10 Puntos]
2. Modelo de casos de uso de todo el sistema. [10 Puntos]
3. Especificar el Caso de Uso (Planilla de Especificación del Caso de Uso) más relevante. El CU principal desde el punto de vista de su análisis. [20 Puntos]
4. Elaborar el diagrama de secuencia para el caso de uso del punto anterior. [30 Puntos]
5. Elaborar el diagrama de clases de toda la solución. [30 Puntos]

Usted puede sumar los Atributos de Dato que considere sean necesarios para robuztecer su diseño. También puede asumir alguna otra específicacion y/o funcionalidad que considere conveniente.

**ENUNCIADO (BIBLIOTECA)**

Una biblioteca contiene ejemplares de libros y películas en VHS. Cada ejemplar tiene un código que es distinto para todos y una descripción de su estado; además puede ser prestado a los socios de la misma. Los libros tienen un título, ***n*** autores, una editorial, un número de edición y una fecha de publicación. De las películas se conoce la editorial, la fecha de publicación y nombre y el tiempo en minutos. En cambios están clasificados por temas (ciencia - ficción, historia, etc.). Además, existe un conjunto de socios de los que se guarda un número de socio, nombre, primer y segundo apellido, dirección, localidad, provincia y número de teléfono. Los socios pueden recibir sanciones económicas por un motivo dado, en una determinada fecha y con un determinado monto. Los socios pueden obtener préstamos de la biblioteca. Cada préstamo tiene como entrada la fecha del día como límite por defecto 10 días a contar a partir de esa fecha. Se mantendrá una lista de ejemplares que los socios piden pero que bien todavía no han sido adquiridos, o bien no hay suficientes ejemplares para satisfacer la demanda (ejemplares agotados) en la que se guardará una referencia al título demandado y la fecha en que se pidió. Aquellas referencias correspondientes a títulos de los que todavía no se han adquirido ejemplares, también serán apuntadas, por lo que se deberá de distinguir entre aquéllos de los que, si existen ejemplares, de los que no existen. Existe la posibilidad de hacer reservas de ejemplares que estén siendo utilizados por otros socios, a tal efecto se le facilitará al socio que va a hacer la reserva la fecha más próxima en la que se va a devolver un ejemplar del título deseado, quedando el ejemplar bloqueado para préstamo durante dos días, a la espera de que lo vaya a retirar el socio que lo ha reservado.

**SINTAGMAS NOMINALES**

Una biblioteca, Ejemplares de libros y películas en VHS, Cada ejemplar, Un código, Una descripción de su estado, Los socios de la misma, Los libros, Un título, N autores, Una editorial, Un número de edición, Una fecha de publicación, Las películas, La editorial, La fecha de publicación, El nombre, El tiempo en minutos, Los cambios, Los temas (ciencia-ficción, historia, etc.), Un conjunto de socios, Un número de socio, Un nombre, Un primer apellido, Un segundo apellido, Una dirección, Una localidad, Una provincia, Un número de teléfono, Sanciones económicas, Un motivo dado, Una determinada fecha, Un determinado monto, Los préstamos, La fecha del día, El límite por defecto de 10 días, Una lista de ejemplares, La demanda, Los ejemplares agotados, Una referencia al título demandado, La fecha en que se pidió, Aquellas referencias correspondientes a títulos, Los ejemplares, Aquéllos de los que si existen ejemplares, Los que no existen, La posibilidad de hacer reservas, Los socios que van a hacer la reserva, La fecha más próxima, Un ejemplar del título deseado, El socio que lo ha reservado.

**SINTAGMAS VERBALES**

Contiene, Tienen, Es, Puede ser prestado, Se conoce, Están clasificados, Se guarda, Pueden recibir, Se mantendrá, Piden, Han sido adquiridos, Serán apuntadas, Se deberá distinguir, Existen, Existe, Hacer, Reservas, Están siendo utilizados, Se facilitará, Se va a devolver, Quedando, Bloqueado, Retirar, Ha reservado.

* Biblioteca:
* Ejemplar:
* Libro:
* Película:
* Socio:
* Sanción:
* Préstamo:

**REQUERIMIENTOS FUNCIONALES**

1. Gestión de ejemplares:

* Registrar ejemplares de libros y películas en VHS.
* Asignar un código único a cada ejemplar.
* Mantener una descripción del estado de cada ejemplar.
* Permitir el préstamo de ejemplares a los socios.

1. Gestión de libros:

* Registrar información de libros, como título, autores, editorial, número de edición y fecha de publicación.

1. Gestión de películas:

* Registrar información de películas, como editorial, fecha de publicación, nombre y duración en minutos.
* Clasificar películas por temas (ciencia-ficción, historia, etc.).

1. Gestión de socios:

* Mantener un registro de socios con información como número de socio, nombre, apellidos, dirección, localidad, provincia y número de teléfono.
* Aplicar sanciones económicas a los socios por motivos específicos, con una fecha y un monto determinados.
* Permitir que los socios realicen préstamos de la biblioteca.

1. Gestión de préstamos:

* Registrar préstamos realizados por los socios.
* Establecer una fecha límite de devolución por defecto de 10 días a partir de la fecha del préstamo.
* Mantener una lista de ejemplares solicitados pero aún no adquiridos o agotados.
* Registrar la referencia al título demandado y la fecha de solicitud.

1. Reservas de ejemplares:

* Permitir reservas de ejemplares que están siendo utilizados por otros socios.
* Asignar una fecha próxima para la devolución del ejemplar reservado.
* Bloquear el ejemplar para préstamo durante dos días, esperando que el socio que lo ha reservado lo retire.

**CASOS DE USO DE TODO EL SISTEMA**

**Actores principales:**

* Bibliotecario
* Socio

**Casos de uso:**

1. Gestión de ejemplares:

* Registrar ejemplar
* Actualizar información de ejemplar
* Eliminar ejemplar
* Consultar ejemplar

1. Gestión de libros:

* Registrar libro
* Actualizar información de libro
* Eliminar libro
* Consultar libro

1. Gestión de películas:

* Registrar película
* Actualizar información de película
* Eliminar película
* Consultar película

1. Gestión de socios:

* Registrar socio
* Actualizar información de socio
* Eliminar socio
* Consultar socio
* Asignar sanción económica a socio
* Eliminar sanción económica de socio

1. Gestión de préstamos:

* Realizar préstamo
* Devolver ejemplar
* Renovar préstamo
* Consultar préstamos por socio
* Consultar préstamos por ejemplar

1. Reservas de ejemplares:

* Realizar reserva de ejemplar
* Cancelar reserva de ejemplar
* Consultar reservas por socio
* Consultar reservas por ejemplar

1. Consultas generales:

* Consultar catálogo de libros
* Consultar catálogo de películas
* Consultar ejemplares disponibles
* Consultar socios con préstamos activos

**CASO DE USO MÁS RELEVANTE**

Dentro del texto proporcionado, el caso de uso más relevante podría ser el **“Realizar Préstamo”**. Este caso de uso es fundamental para el funcionamiento del sistema, ya que permite a los socios de la biblioteca solicitar y obtener préstamos de ejemplares. Al realizar un préstamo, se establece una relación entre un socio y un ejemplar específico, y se registra la fecha de préstamo y la fecha límite de devolución.

Este caso de uso implica varias acciones importantes, como verificar la disponibilidad del ejemplar, actualizar el estado del ejemplar a "prestado", registrar la información del préstamo y establecer una fecha límite para su devolución. Además, el caso de uso podría involucrar validaciones adicionales, como verificar si el socio tiene alguna sanción económica pendiente o si ha alcanzado el límite máximo de préstamos permitidos.

El caso de uso **“Realizar Préstamo”** es crucial para el sistema, ya que permite el acceso y uso de los recursos de la biblioteca por parte de los socios, promoviendo la circulación y disponibilidad de los ejemplares.

**DIAGRAMA DE SECUENCIA PARA EL CASO DE USO “REALIZAR PRÉSTAMO”**

Secuencia: Realizar Préstamo

Actor: Socio

Sistema: Biblioteca

1. El Socio solicita realizar un préstamo de un ejemplar específico.

2. El Sistema verifica la disponibilidad del ejemplar.

3. El Sistema verifica si el Socio tiene alguna sanción económica pendiente.

4. El Sistema verifica si el Socio ha alcanzado el límite máximo de préstamos permitidos.

5. El Sistema marca el ejemplar como "prestado" y lo asocia al Socio.

6. El Sistema registra la fecha de préstamo y calcula la fecha límite de devolución.

7. El Sistema confirma al Socio que el préstamo se ha realizado exitosamente.

8. Fin de la secuencia.

En este diagrama de secuencia, el Socio interactúa con el Sistema de la Biblioteca para solicitar un préstamo. El Sistema realiza varias verificaciones, como disponibilidad del ejemplar, sanciones económicas y límite de préstamos, antes de marcar el ejemplar como prestado y registrar la información del préstamo. Finalmente, el Sistema confirma al Socio que el préstamo se ha realizado exitosamente.

Cabe mencionar que este diagrama de secuencia es una representación simplificada y se pueden agregar más detalles o interacciones dependiendo de los requisitos específicos del sistema.

OTRO PLANTEO:

Diagrama de secuencia para el escenario "REALIZAR PRÉSTAMO" en la biblioteca:

* El socio inicia sesión en el sistema de la biblioteca.
* El sistema muestra al socio las opciones disponibles, incluyendo la opción de realizar un préstamo.
* El socio selecciona la opción de realizar un préstamo.
* El sistema muestra al socio un formulario para ingresar los detalles del préstamo, como el número de socio y el código del ejemplar a prestar.
* El socio completa los campos requeridos en el formulario.
* El sistema verifica si el socio y el ejemplar son válidos y si el ejemplar está disponible para préstamo.
* Si el socio o el ejemplar no son válidos, el sistema muestra un mensaje de error y vuelve al paso 4.
* Si el ejemplar no está disponible, el sistema muestra al socio la opción de hacer una reserva.
* El socio decide si hacer una reserva o cancelar el préstamo.
* Si el socio decide hacer una reserva, el sistema muestra al socio la fecha más próxima en la que el ejemplar estará disponible y bloqueado para préstamo.
* El socio confirma la reserva.
* El sistema registra la reserva y muestra al socio un mensaje de confirmación.
* Si el socio decide cancelar el préstamo, el sistema muestra un mensaje de cancelación.
* Si el socio decide continuar con el préstamo, el sistema registra el préstamo con la fecha actual y establece la fecha límite de devolución (10 días después).
* El sistema actualiza el estado del ejemplar a prestado y lo asocia al socio.
* El sistema muestra al socio un mensaje de confirmación del préstamo.
* El socio finaliza la transacción y cierra sesión en el sistema.

Este diagrama de secuencia muestra la interacción entre el socio y el sistema de la biblioteca para realizar un préstamo.

**DIAGRAMA DE CLASES DE TODA LA SOLUCIÓN**

Este diagrama de clases representa las entidades principales del escenario: Biblioteca, Ejemplar, Libro, Película, Socio, Sanción, Prestamo y Reserva. Las relaciones entre las clases se definen mediante atributos y métodos para representar la interacción y las características de cada entidad en el sistema de la biblioteca.

+----------------------------------+

| Biblioteca |

+----------------------------------+

| - ejemplares: List<Ejemplar> |

| - socios: List<Socio> |

| - reservas: List<Reserva> |

+----------------------------------+

| + agregarEjemplar(ejemplar: Ejemplar): void |

| + agregarSocio(socio: Socio): void |

| + realizarReserva(socio: Socio, titulo: String): void |

+----------------------------------+

+----------------------------------+

| Ejemplar |

+----------------------------------+

| - codigo: String |

| - descripcionEstado: String |

| - prestado: boolean |

+----------------------------------+

| + Ejemplar(codigo: String) |

| + getCodigo(): String |

| + getDescripcionEstado(): String |

| + isPrestado(): boolean |

| + prestar(): void |

| + devolver(): void |

+----------------------------------+

+----------------------------------+

| Libro |

+----------------------------------+

| - titulo: String |

| - autores: List<String> |

| - editorial: String |

| - numeroEdicion: int |

| - fechaPublicacion: Date |

+----------------------------------+

| + Libro(titulo: String, autores: List<String>, editorial: String, numeroEdicion: int, fechaPublicacion: Date) |

| + getTitulo(): String |

| + getAutores(): List<String> |

| + getEditorial(): String |

| + getNumeroEdicion(): int |

| + getFechaPublicacion(): Date |

+----------------------------------+

+----------------------------------+

| Pelicula |

+----------------------------------+

| - titulo: String |

| - editorial: String |

| - fechaPublicacion: Date |

| - nombre: String |

| - duracion: int |

| - tema: String |

+----------------------------------+

| + Pelicula(titulo: String, editorial: String, fechaPublicacion: Date, nombre: String, duracion: int, tema: String) |

| + getTitulo(): String |

| + getEditorial(): String |

| + getFechaPublicacion(): Date |

| + getNombre(): String |

| + getDuracion(): int |

| + getTema(): String |

+----------------------------------+

+----------------------------------+

| Socio |

+----------------------------------+

| - numeroSocio: int |

| - nombre: String |

| - apellido1: String |

| - apellido2: String |

| - direccion: String |

| - localidad: String |

| - provincia: String |

| - telefono: String |

| - sanciones: List<Sancion> |

| - prestamos: List<Prestamo> |

+----------------------------------+

| + Socio(numeroSocio: int, nombre: String, apellido1: String, apellido2: String, direccion: String, localidad: String, provincia: String, telefono: String) |

| + getNumeroSocio(): int |

| + getNombre(): String |

| + getApellido1(): String |

| + getApellido2(): String |

| + getDireccion(): String |

| + getLocalidad(): String |

| + getProvincia(): String |

| + getTelefono(): String |

| + getSanciones(): List<Sancion> |

| + getPrestamos(): List<Prestamo> |

| + solicitarPrestamo(ejemplar: Ejemplar): boolean |

| + devolverPrestamo(ejemplar: Ejemplar): boolean |

| + realizarReserva(titulo: String, fecha: Date): boolean |

+----------------------------------+

+----------------------------------+

| Sancion |

+----------------------------------+

| - motivo: String |

| - fecha: Date |

| - monto: float |

+----------------------------------+

| + Sancion(motivo: String, fecha: Date, monto: float) |

| + getMotivo(): String |

| + getFecha(): Date |

| + getMonto(): float |

+----------------------------------+

+----------------------------------+

| Prestamo |

+----------------------------------+

| - socio: Socio |

| - ejemplar: Ejemplar |

| - fechaLimite: Date |

+----------------------------------+

| + Prestamo(socio: Socio, ejemplar: Ejemplar, fechaLimite: Date) |

| + getSocio(): Socio |

| + getEjemplar(): Ejemplar |

| + getFechaLimite(): Date |

| + isVencido(): boolean |

+----------------------------------+

+----------------------------------+

| Reserva |

+----------------------------------+

| - socio: Socio |

| - titulo: String |

| - fechaPeticion: Date |

| - disponible: boolean |

+----------------------------------+

| + Reserva(socio: Socio, titulo: String, fechaPeticion: Date) |

| + getSocio(): Socio |

| + getTitulo(): String |

| + getFechaPeticion(): Date |

| + isDisponible(): boolean |

| + setDisponible(disponible: boolean): void |

+----------------------------------+

1. xxx [↑](#footnote-ref-1)