ASPECTOS CONCEPTUALES

1. La diferencia entre análisis, diseño e implementación es que el análisis pone énfasis en la investigación del problema y los requisitos; el diseño pone énfasis en una solución conceptual que satisface los requisitos; y la implementación es la traducción de la solución a un lenguaje de programación determinado.

Por último, durante el análisis se intenta encontrar y describir los objetos del dominio del problema y durante el diseño se presta atención a la definición de los objetos software y en como colaboran para satisfacer los requisitos.

1. Los tipos de requisitos son:

* Requisitos Funcionales: Describen las funcionalidades y el comportamiento específico que debe tener el software. Estos requisitos indican qué tareas o acciones debe ser capaz de realizar el sistema. Un ejemplo podría ser “El sistema debe ser capaz de generar informes mensuales de ventas”.
* Requisitos No Funcionales: Se refieren a características del software que no están directamente relacionados con su funcionalidad, sino con aspectos como el rendimiento, la seguridad, etc. Un ejemplo podría ser “El sistema debe responder en menos de 2 segundos a las solicitudes de los usuarios”.
* Requisitos de Sistema: Son los componentes que el sistema debe tener para realizar determinadas tareas. Un ejemplo podría ser “El sistema debe ser capaz de procesar y almacenar al menos 10,000 registros de clientes”.
* Requisitos de Usuario: Se centran en las necesidades y expectativas que los usuarios expresan verbalmente. Un ejemplo podría ser “El sistema debe tener una interfaz intuitiva y fácil de usar, de manera que los usuarios puedan realizar tareas sin dificultad”.

1. Un Escenario en el contexto de los casos de usos es una secuencia específica de acciones entre los actores y el sistema (instancia de un caso de uso). Por ejemplo, en el caso de uso Contratar Empleado podría tener variantes como: contratar externos, traslado dentro de la misma empresa ó contratar extranjeros. Cada uno se puede expresar en una secuencia diferente.

Éste concepto lo podemos relacionar con el concepto de diagrama de secuencia ya que un escenario de caso de uso puede ser representado en un DSS mediante la visualización de la secuencia de mensajes y acciones que ocurren entre los objetos involucrados en el escenario.

1. La diferencia entre mensajes, operaciones y métodos es que un mensaje es una solicitud que un objeto envía a otro objeto para que realice una determinada acción; las operaciones son las acciones que pueden ser invocadas o ejecutadas en respuesta a un mensaje y definen qué puede hacer un objeto y cómo se debe realizar una acción; y por último, los métodos es una función asociado a un objeto/clase que realiza una acción específica.

Ejemplo de mensajes, operaciones y métodos en JavaScript:

Texto

Descripción generada automáticamente

1. Las diferencias entre clase, clase abstracta e interfaz son: una Clase es una plantilla que define las propiedades y comportamientos de un objeto. Puede contener atributos y métodos que describen el estado y el comportamiento de los objetos que se crean a partir de ella. Además, pueden heredar propiedades y comportamientos de otras clases y pueden ser instanciadas para crear objetos concretos.

Una Clase Abstracta es una clase que no se la puede instanciar directamente, sino que se utiliza como clase base para otras clases. Contiene métodos abstractos (sin implementación) así métodos concretos. Las clases que hereden de una clase abstracta deberán implementar los métodos abstractos definidos en la clase abstracta.

Por último, una interfaz es una colección de métodos abstractos que se utilizan para definir un contrato que una clase debe implementar. Define qué métodos deben estar presentes en una clase, pero no especifica cómo se deben implementar.

1. El Encapsulamiento es un mecanismo para reunir datos y métodos dentro de una estructura ocultando la implementación del objeto. El objetivo es ocultar los detalles internos y proteger los datos dentro del objeto, y además proporciona un conjunto de métodos para manipular a esos datos.

Se implementa a partir de modificadores de acceso como public, private y protected que permiten controlar la visibilidad y el acceso a los miembros de un objeto. También, a partir de métodos de acceso como getters y setters que son métodos que se utilizan para acceder y modificar los atributos privados de un objeto.

Ejemplo de encapsulamiento en c#:

Texto

Descripción generada automáticamente

1. Herencia: Es un mecanismo mediante el cual una clase puede heredar propiedades y comportamientos a partir de otra clase existente. La herencia permite que las clases derivadas reutilicen y extiendan el código de la clase base, lo que promueve la reutilización y la modularidad en el diseño del software.

Generalización: Es un proceso mediante el cual se identifican características comunes entre varias clases y se crea una más general para representar esas características comunes. La clase general se la conoce como superclase. Por ejemplo, si tenemos las clases “Perro”, “Gato” y “Pájaro” podemos identificar las características comunes y crear una nueva superclase llamada “Animal” que posea todos los atributos y comportamientos compartidos por todas las especies animales.

Especialización: Es el proceso opuesto a la generalización donde se crea una clase más especifica que hereda de una clase general. La especialización agrega atributos y comportamientos específicos adicionales a la clase base y redefine los atributos/métodos heredados.

Entonces, la herencia permite que una clase herede propiedades y comportamientos de otra clase. La generalización y la especialización son procesos complementarios que permiten la organización jerárquica y la reutilización de código.

1. El Polimorfismo es un concepto que se refiere a la capacidad de un objeto de tomar o comportarse de diferentes formas en función del contexto en el que se utilice. Además, permite que un objeto se comporte como si fuera de diferentes tipos, proporcionando flexibilidad y extensibilidad.

Ejemplo de polimorfismo en c#:

Texto

Descripción generada automáticamente

ASPECTOS PROCEDIMENTALES

1. RFN1. Gestión de Préstamos y Reservas. La gestión de préstamos se efectúa cuando los socios que llegan físicamente al establecimiento, realizan el debido préstamo.

Esta gestión permite la obtención de préstamos de la biblioteca; el establecimiento de una fecha límite de devolución; la reservación de ejemplares; y la asignación del estado de un préstamo.

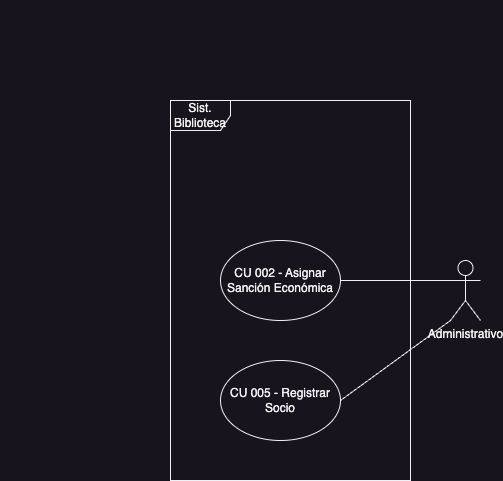
RFN2. Gestión de Socios. La gestión de socios se realizará por parte de la Casa Matriz de la Biblioteca de manera presencial. Esta funcionalidad debe proporcionar: registrar la información de los usuarios; y la asignación de sanciones económicas a los socios con fecha y monto determinado.

1. RFN1. DIAGRAMA DE CASO DE USO

Diagrama

Descripción generada automáticamente

RFN2. DIAGRAMA DE CASO DE USO



|  |
| --- |
| **ID Y NOMBRE:** CU 001 – Gestionar Préstamo |
| **DESCRIPCIÓN:** Permite la gestión por sistema de los préstamos efectuados por los socios. |
| **ACTOR PRINCIPAL:** Bibliotecario |
| **ACTORES SECUNDARIOS:** Socio |
| **PRECONDICIONES:** El socio ya debe estar registrado en la biblioteca y no poseer sanciones. El bibliotecario debe estar logueado en el sistema. |
| **ESCENARIO PRINCIPAL:**   1. El bibliotecario solicita los datos personales del socio para gestionar el préstamo. 2. El bibliotecario solicita ‘Gestionar Préstamo’ en el sistema 3. El sistema solicita ingresar el número de socio y el apellido del socio. 4. El bibliotecario ingresa el número de socio y el apellido del socio. 5. El sistema comprueba que el socio se encuentre cargado en la base de datos de la biblioteca y que no posea ninguna sanción. 6. El sistema despliega la lista de ejemplares disponibles para ser prestados. 7. El bibliotecario selecciona la lista de ejemplares disponibles que el socio eligió. INCLUYE CU 004 – SELECCIONAR EJEMPLARES. 8. El sistema despliega una ventana de confirmación con la lista de ejemplares a prestar. 9. El bibliotecario confirma que los títulos a prestar son los correctos. 10. El sistema solicita ingresar una fecha de devolución. 11. El bibliotecario ingresa la fecha de devolución de los ejemplares. 12. El sistema solicita la confirmación del préstamo para su generación. 13. El bibliotecario confirma la creación del préstamo. 14. El sistema actualiza el estado del préstamo. 15. El sistema muestra la factura generada. 16. El sistema envía comprobante del préstamo al mail registrado del socio. |
| **FLUJOS ALTERNATIVOS:**  5.1. En el caso que el sistema no encuentre ningún socio registrado que coincida con los datos ingresados por el administrativo, se le solicitará por pantalla al bibliotecario que ingrese los datos nuevamente. Retorna al paso 3.  11.1 En el caso de que el bibliotecario no ingrese una fecha de devolución, el sistema ingresará por defecto 10 días a partir de la fecha del préstamo generado. |
| **POSTCONDICIONES:** Se generó correctamente el préstamo al socio. |

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente