Unidad Integrada de Clase

ÍNDICE

FASE 1: COMUNICACIÓN

1. Objetivo del programa

Este programa gestiona una librería, permitiendo a los usuarios buscar libros por palabras clave, añadirlos a una cesta y realizar operaciones básicas cómo eliminar libros. Está dirigido a lectores que quieran realizar sus compras virtuales.

2. Requisitos funcionales (qué debe hacer el sistema)

-Dar de alta libros en el catálogo
-Buscar libros por palabra clave
-Añadir libros a la cesta
-Eliminar libros de la cesta
-Consultar libros en la cesta

3. Requisitos no funcionales

El uso de “JUnit” para realizar test en el código

4. Actores involucrados

-Cesta
-Catálogo
-Libro
-Usuario

5. Tecnologías o herramientas utilizadas

-Lenguaje de programación: Java
-Pruebas: JUnit
-Diagramas: clase y secuencia
-IDE: Visual

6. Metodología de trabajo

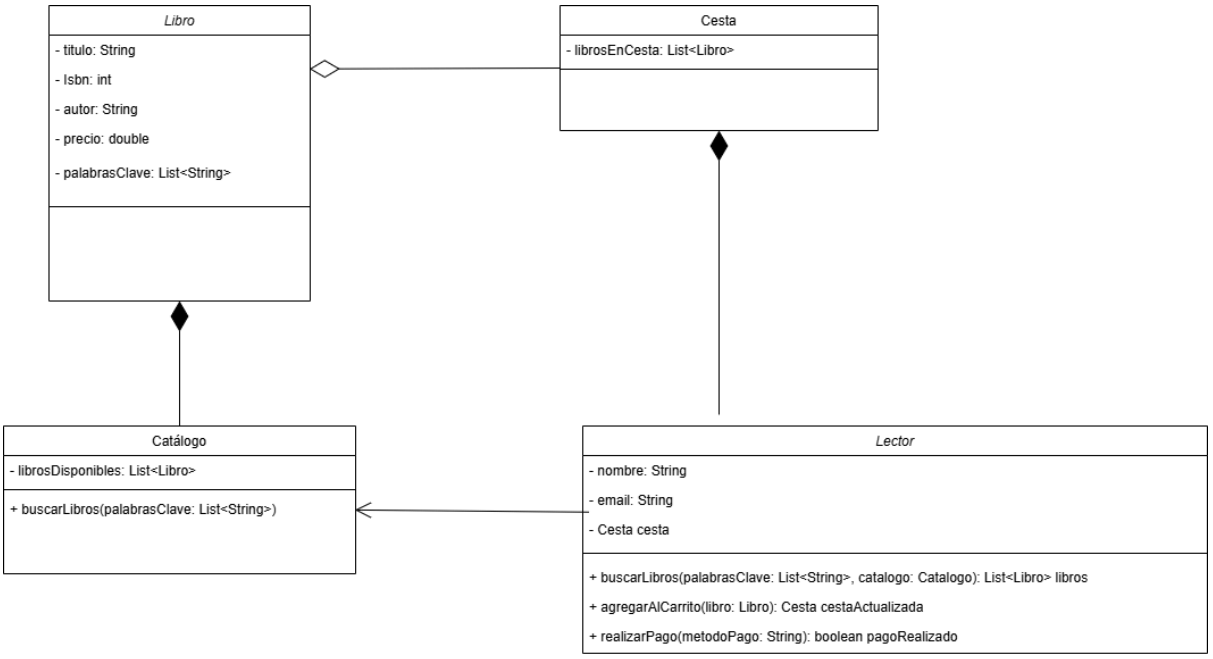
Se desarrolla a lo largo del trabajo en varias fases:
-Comunicación
-Diseño
-Codificación
-Implantación

FASE 2: DISEÑO

En esta fase se tratará todo lo referente al diseño del trabajo, en nuestro caso, los diagramas. Nos encontraremos con dos tipos, los de clase y los de secuencia.

DIAGRAMA DE CLASE

Un usuario accede a la app de “La Casa del Libro” para buscar libros disponibles. Puede ingresar palabras clave o filtros para encontrar uno o varios libros. Una vez que encuentra los libros deseados, los añade al carrito de compras. Luego, el usuario procede a realizar el pago seleccionando un método disponible.



He creado una aplicación para poder gestionar la compra de libros online a través de la app de “La casa del Libro”. En esta aplicación tenemos un diagrama de clases, esquematizando en nuestro caso cuatro.

Clase: Libro
.Descripción: Representa un libro individual disponible en el sistema.

.Atributos:
+titulo: String → El título del libro.
+isbn: int → Número ISBN del libro, identificador.
+autor: String → Nombre del autor del libro.
+precio: double → Precio del libro.
+palabrasClave: List<String> → Lista de palabras clave asociadas al libro, útiles para búsquedas, en este caso.

.Función: Esta clase almacena la información básica de cada libro, y se usa tanto en el catálogo como en la cesta de compra.

Clase: Catálogo
.Descripción: Representa el conjunto de libros disponibles en el sistema para ser consultados o comprados.

.Atributos:
+librosDisponibles: List<Libro> → Lista de libros disponibles actualmente.

.Métodos:
+buscarLibros(palabrasClave: List<String>): List<Libro> → Devuelve una lista de libros que coinciden con las palabras clave proporcionadas.

.Función: El catálogo permite buscar libros disponibles mediante palabras clave y gestionar la disponibilidad del inventario.

Clase: Cesta
.Descripción: Contiene los libros que un lector ha añadido como parte de su carrito de compras.

.Atributos:
+librosEnCesta: List<Libro> → Lista de libros que el lector ha añadido a su cesta.

.Función: La cesta representa el carrito de compras del lector, donde se almacenan los libros antes de realizar el pago.

Clase: Lector
.Descripción: Representa a un usuario del sistema que busca libros y realiza compras.

.Atributos:
+nombre: String → Nombre del lector.
+email: String → Correo electrónico del lector.
+cesta: Cesta → Cesta personal del lector para agregar libros.

.Métodos:
+buscarLibros(palabrasClave: List<String>, catalogo: Catalogo): List<Libro> → Permite al lector buscar libros en el catálogo mediante palabras clave.
+agregarAlCarrito(libro: Libro): Cesta → Añade un libro a la cesta del lector y devuelve la cesta actualizada.
+realizarPago(metodoPago: String): boolean → Realiza el pago por los libros en la cesta, usando el método de pago especificado. Devuelve true si el pago fue exitoso, false en caso contrario.

.Función: El lector puede buscar libros, agregarlos a su cesta y proceder a pagar. Representa al usuario final de la aplicación.

En cuanto a los tipos de relaciones tenemos:
.Catálogo - Libro: Composición

Justificación: El Catálogo posee y gestiona la colección de Libros. Si el Catálogo desaparece, los Libros ya no tienen sentido en este contexto.
.Usuario - Catálogo: Asociación

Justificación: El Usuario consulta el Catálogo para buscar libros. No lo posee ni lo modifica directamente.

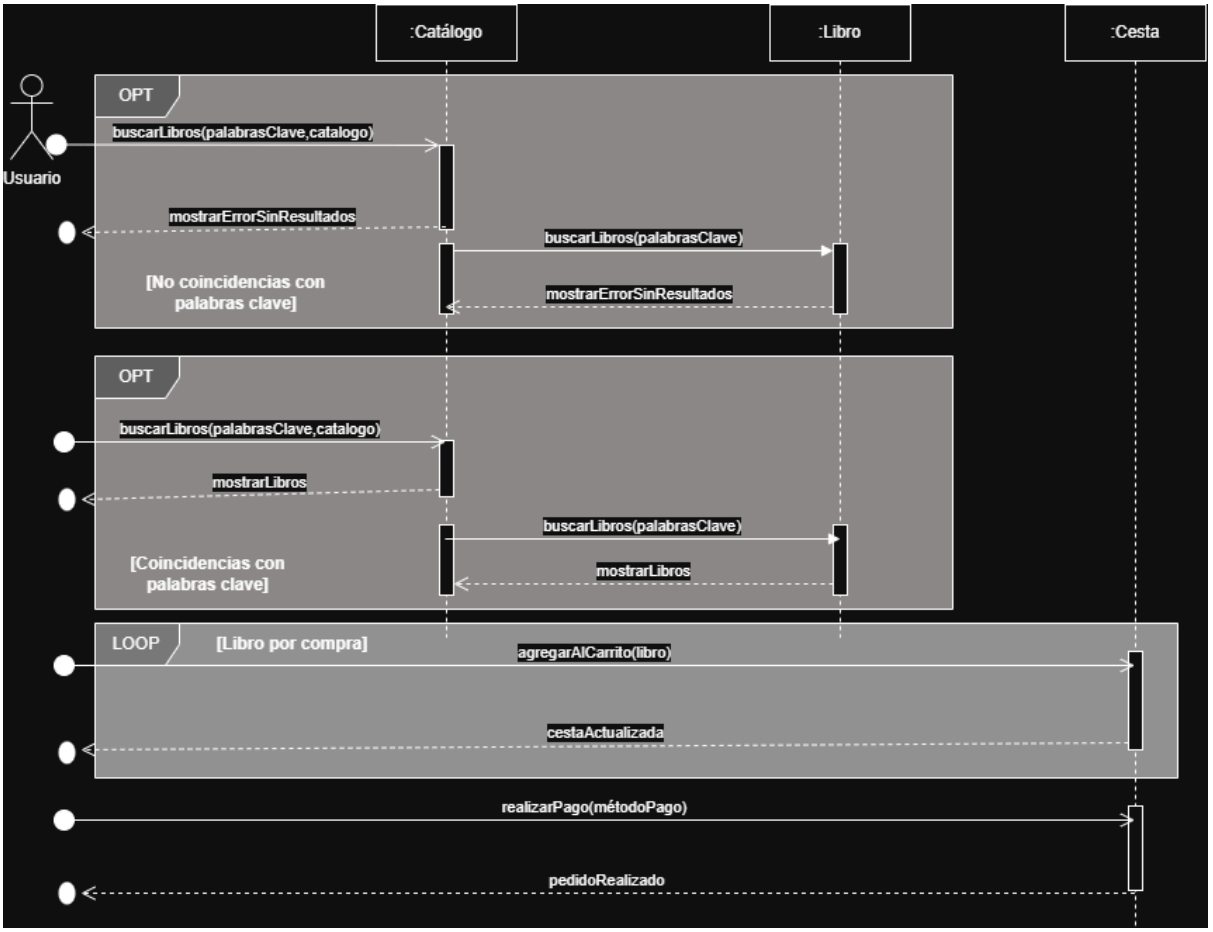
.Usuario - Cesta: Composición

Justificación: Cada Usuario tiene su propia Cesta. Si el Usuario deja de existir, su Cesta también.

.Cesta - Libro: Agregación

Justificación: La Cesta contiene referencias a Libros, pero no los crea ni los destruye. Los libros existen independientemente.

DIAGRAMA DE SECUENCIA



.Descripción breve:
El diagrama muestra cómo un usuario (lector) interactúa con el sistema para buscar libros usando palabras clave, revisar los resultados, agregar libros al carrito (cesta) y finalmente realizar el pago.

.Clases:
-Usuario: El actor que inicia la búsqueda de los libros y realiza la compra.
-Catálogo: Componente encargado de gestionar y buscar los libros disponibles.
-Libro: Clase sobre la que se ejecuta la búsqueda.
-Cesta: Carrito de compra del usuario donde se agregan los libros seleccionados.

1. Búsqueda de libros (OPT sin resultados)
-Usuario solicita buscarLibros(palabrasClave, catalogo).
-El mensaje se envía a la clase Catálogo.
-El catálogo llama a buscarLibros(palabrasClave) en la clase Libro.

Si no hay coincidencias, el sistema invoca “mostrarErrorSinResultados” para notificar al usuario.

2. Búsqueda de libros (OPT con resultados)
-El mecanismo comienza casi igual que el anterior: el usuario envía buscarLibros(palabrasClave, catalogo).
-Esta vez, el resultado es positivo: hay coincidencias.
-Se ejecuta mostrarLibros para presentar los libros encontrados.

3. Agregar libros a la cesta
-El usuario selecciona libros para comprar.
-Por cada libro seleccionado (representado con un bucle LOOP: [Libro por compra]):
-Se llama a agregarAlCarrito(libro) desde el usuario al objeto :Cesta.
-Se recibe la respuesta cestaActualizada.

4. Realizar pago
-Después de agregar libros, el usuario llama a realizarPago(metodoPago).
-El sistema procesa el pago y responde con pedidoRealizado, indicando que la compra fue exitosa.

FASE 3: CODIFICACIÓN

En base a lo relacionado anteriormente crearemos el código en Java para poder implementarlo y luego hacer test de ello. Disponemos de las clases:

.Cesta

Esta clase Cesta permite gestionar una colección de libros, añadiéndolos o eliminándolos de forma segura.

+Mantiene una lista de objetos Libro en la variable libros, que representa la colección de libros añadidos a la cesta.

+Permite obtener la lista completa de libros mediante el método getLibros().

+Permite añadir libros a la cesta con el método incluirLibro(), siempre que no sea null ni esté ya en la lista.

+Permite eliminar libros de la cesta con el método eliminarLibro() si no es null.

.Catálogo

La clase Catalogo representa un catálogo de libros, es decir, una colección central de libros disponibles que podrían estar a la venta, para préstamo o simplemente listados en una biblioteca digital.

.Atributo principal:

-librosDisponibles: una lista que contiene todos los libros registrados en el catálogo.

.Métodos principales:

+getLibrosDisponibles(): Devuelve la lista completa de libros disponibles en el catálogo.
+altaLibro(Libro libro): Añade un libro al catálogo si no es null y no está ya incluido. Evita duplicados en el catálogo.
+buscarLibros(String palabraClave): Busca libros que contengan una palabra clave específica en su lista de palabras clave. Devuelve una lista con los libros que coincidan.

.Usuario

La clase Usuario representa a un usuario dentro del contexto de una aplicación de librería, como podría ser una tienda de libros en línea o un sistema de gestión bibliotecaria.

.Atributos principales:

-nombre: el nombre del usuario.
-email: el correo electrónico del usuario.
-cesta: una instancia de la clase Cesta, que representa la colección de libros que el usuario ha añadido (como un carrito de compras).

.Constructor:

+Inicializa el nombre y el email del usuario.
+Crea una nueva Cesta vacía para el usuario automáticamente.
+Getters y Setters: Permiten acceder y modificar el nombre, el email y la cesta asociada al usuario.

.Libro
La clase Libro representa una estructura para almacenar información relacionada con un libro. A nivel general, esta clase define los atributos básicos de un libro y proporciona métodos para acceder y modificar estos atributos.

.Atributos:
-título: el título del libro.
-autor: el nombre del autor.
-isbn: el número ISBN (identificador único).
-precio: el precio del libro.
-palabrasClave: una lista de palabras clave asociadas al libro, útil por ejemplo para búsquedas o categorización.

-Tiene un constructor que permite crear un objeto Libro inicializando todos sus atributos.

+Incluye métodos get y set (getters y setters) para cada atributo, lo cual permite: Obtener el valor actual de un atributo. Modificar el valor de un atributo después de que el objeto ha sido creado.

.Programa (Main)
-Se crean dos objetos Libro con distintos atributos (título, autor, ISBN, precio y palabras clave).
-Se crea un objeto “Catálogo” y agrega los libros a este catálogo usando el método altaLibro().
-Se crea un objeto “Usuario” llamado Juan Pérez con una dirección de correo electrónico.
-Busca libros en el catálogo que contengan la palabra clave "aventura" y los imprime.
-Agrega un libro a la cesta del usuario mediante “usuario.getCesta().incluirLibro(libro1);”
-Muestra los libros que están actualmente en la cesta del usuario imprimiendo sus títulos

La salida esperada del programa por ende sería:
+Libros con la palabra clave 'aventura':
- El Quijote

+Libros en la cesta de Juan Pérez:
- El Quijote

FASE 4: IMPLANTACIÓN

Se realizan varias pruebas con Junit para verificar que el código es correcto. De las clases que se han realizado los test son las siguientes:
-Cesta
-Catálogo
-Usuario

Desglosando y explicando brevemente las pruebas desarrolladas y ejecutadas con éxito, tenemos lo siguiente:

.Usuario
.test_creacionUsuario_cestaVacia() → Crea un nuevo objeto Usuario con un nombre y un email, además, verifica que al crear un usuario, su cesta de libros esté vacía por defecto. Es true, es decir, que no hay libros en la cesta al momento de creación.

.Catálogo
. test_altaLibro_libroNuevo() → Añade dos libros diferentes (LIBRO_1 y LIBRO_2) al catálogo. Se espera que haya 2 libros en el catálogo después de agregarlos.

.test_altaLibro_libroRepetido() → Intenta agregar el mismo libro (LIBRO_1) dos veces. Prueba que el catálogo no permite duplicados y se espera que haya solo 1 libro guardado, incluso tras dos intentos.

.test_buscarLibros_conPalabraClave() → Agrega dos libros y luego realiza búsquedas por distintas palabras clave (etiquetas). Prueba que la búsqueda por palabras clave (etiquetas como "aventura", "realismo", "antiguo") funciona correctamente.

Valida lo siguiente:
"aventura" → debe devolver solo LIBRO_1.
"realismo" → debe devolver solo LIBRO_2.
"antiguo" → ambos libros, ya que ambos tienen esa etiqueta.

.test_buscarLibros_noHayPalabraClave() → Agrega dos libros y busca una palabra clave que no está presente ("juvenil"). Prueba que la búsqueda vacía se maneja bien y no da falsos positivos, se espera una lista vacía.

.Cesta
.test_incluirLibro_libroNuevo() →Agrega dos libros distintos (LIBRO_1 y LIBRO_2) a la cesta. Esto prueba que la cesta puede contener varios libros diferentes y se espera que el número total de libros sea 2.

.test_incluirLibro_libroRepetido() →Intenta añadir el mismo libro dos veces. Esto prueba que la cesta no duplica libros cuando se intenta añadir uno repetido. Espera que haya solo 1 libro en la cesta, a pesar de los dos intentos.

.test_eliminarLibro_libroExistente() → Agrega dos libros y elimina uno. Esto prueba que se puede eliminar correctamente un libro que sí está en la cesta. Espera que quede 1 libro tras la eliminación.

.test_eliminarLibro_libroNoExistente() → Agrega un libro (LIBRO_1) y trata de eliminar otro que no está en la cesta (LIBRO_2). Esto prueba que eliminar un libro que no existe no modifica el contenido de la cesta. La cesta debe seguir teniendo 1 libro.

.test_buscarLibros_conPalabraClave() →Busca libros por palabra clave después de añadirlos al catálogo. Esto prueba que la búsqueda funciona por etiquetas como "aventura", "realismo" y "antiguo".

.test_buscarLibros_noHayPalabraClave() → Busca libros con una etiqueta que no está presente. Esto prueba que la búsqueda devuelve cero resultados si no hay coincidencias.