

<b>EVALUACION</b>	Obligatorio	<b>GRUPO</b>	TODOS	<b>FECHA</b>	16/09/24
<b>MATERIA</b>	<b>Algoritmos y Estructuras de Datos 1</b>				
<b>CARRERA</b>	Analista Programador / Analista en Tecnologías de la información				
<b>CONDICIONES</b>	<p>Obligatorio 1</p> <ul style="list-style-type: none"> <li>- <b>Puntaje máximo:</b> 20 puntos    - <b>Puntaje mínimo:</b> 0 puntos</li> <li>- <b>Fecha de entrega:</b> 14/10/2024 hasta las 21:00 horas</li> </ul> <p>Obligatorio 2</p> <ul style="list-style-type: none"> <li>- <b>Puntaje máximo:</b> 35 puntos    - <b>Puntaje mínimo:</b> 0 puntos</li> <li>- <b>Fecha de entrega:</b> 21/11/2024 hasta las 21:00 horas</li> </ul> <p>Entregas se realizan en <a href="http://gestion.ort.edu.uy">gestion.ort.edu.uy</a> (max. 40Mb en formato zip, rar)</p> <p><b>Uso de material de apoyo y/o consulta</b></p> <p><u>Inteligencia Artificial Generativa</u></p> <ul style="list-style-type: none"> <li>- Seguir las pautas de los docentes: Se deben seguir las instrucciones específicas de los docentes sobre cómo utilizar la IA en cada curso.</li> <li>- Citar correctamente las fuentes y usos de IA: Siempre que se utilice una herramienta de IA para generar contenido, se debe citar adecuadamente la fuente y la forma en que se utilizó.</li> <li>- Verificar el contenido generado por la IA: No todo el contenido generado por la IA es correcto o preciso. Es esencial que los estudiantes verifiquen la información antes de usarla.</li> <li>- Ser responsables con el uso de la IA: Conocer los riesgos y desafíos, como la creación de "alucinaciones", los peligros para la privacidad, las cuestiones de propiedad intelectual, los sesgos inherentes y la producción de contenido falso</li> <li>- En caso de existir dudas sobre la autoría, plagio o uso no atribuido de IAG, el docente tendrá la opción de convocar al equipo de obligatorio a una defensa específica e individual sobre el tema</li> </ul> <p><b>IMPORTANTE:</b></p> <ul style="list-style-type: none"> <li>- Previo a la semana de cada entrega, quedarán disponible un juego de pruebas para validar todas las operaciones solicitadas. Adicionalmente, el grupo debe entregar su conjunto de pruebas (utilizando Junit4 y la clase Retorno provista) que evidencien el testing realizado.</li> <li>- Inscribirse.</li> <li>- Formar grupos de hasta 2 personas del mismo dictado.</li> <li>- Subir el trabajo a Gestión antes de la hora indicada (ver hoja al final del documento: "RECORDATORIO")</li> </ul> <p>Aquellos de ustedes que presenten alguna dificultad con su inscripción o tengan inconvenientes técnicos, por favor contactarse con el Coordinador o Coordinación adjunta <b>antes de las 20:00hs.</b> del día de la entrega, a través de los mails <a href="mailto:alamon@ort.edu.uy">alamon@ort.edu.uy</a> y <a href="mailto:fernandez_ma@ort.edu.uy">fernandez_ma@ort.edu.uy</a>, o telefónicamente al 29021505 - int 1156 u 1138</p>				

### **Información importante**

- La selección adecuada de las estructuras para modelar el problema y la eficiencia en cada una de las operaciones es muy importante.
- Se deben implementar todos los TADs utilizados, no se permite el uso de estructuras nativas de JAVA (colecciones, pilas, colas, etc., por ejemplo, List, ArrayList, HashMap).
- Los obligatorios se realizan por equipos de hasta 2 estudiantes.
- Se deberán respetar los tipos de retorno dados.
- El sistema no debe requerir ningún tipo de interacción con el usuario por consola.
- Es obligación del estudiante mantenerse al tanto de las aclaraciones que se realicen en clase o a través del foro de aulas.
- Deberá aplicar la metodología vista en el curso.
- El proyecto será implementado en lenguaje JAVA sobre una interfaz que se publicará en el sitio de la materia en [aulas.ort.edu.uy](http://aulas.ort.edu.uy) (el uso de esta interfaz es obligatorio).

## Obligatorio:

### Contenido

1. Introducción - Sistema de Gestión de Bibliotecas.....	4
2. Administración de Estudiantes y Libros .....	5
2.1. Crear Sistema de Gestión .....	5
2.2. Agregar Estudiante .....	5
2.3. Obtener Estudiante .....	5
2.4. Eliminar Estudiante .....	5
2.5. Agregar Libro .....	6
2.6. Prestar Libro .....	6
2.7. Reservar Libro .....	6
2.8. Devolver Libro .....	7
2.9. Eliminar Libro .....	7
3. Listados y reportes .....	7
3.1. Listar Estudiantes .....	7
3.2. Listar Libros .....	8
3.3. Listar Libros por categoría.....	8
3.4. Listar prestamos de un estudiante.....	8
3.5. Libros más prestados.....	9
3.6. Deshacer las últimas n eliminaciones de libros.....	9
3.7. Cantidad prestamos activos .....	9
3.8. Ranking de categorías .....	10

## 1. Introducción - Sistema de Gestión de Bibliotecas

Se desea implementar un sistema que permita gestionar la biblioteca de una institución educativa administrando las solicitudes de préstamos y devoluciones de libros por parte de los estudiantes. El sistema debe poder gestionar estudiantes y libros y también administrar de manera eficiente el préstamo y devolución de estos. Además, debe ofrecer funcionalidades de reporte sobre los préstamos realizados, la disponibilidad de libros y un historial de préstamos para cada estudiante. Los distintos módulos y procesos deben ser resueltos a nivel de arquitectura y estructuras del sistema de la forma más eficiente posible.

Se proveen los siguientes tipos de datos que deberán ser respetados.

Sistema	<pre>public class Sistema{      /*Aquí introduzca la información que estime conveniente*/  }</pre>
Retorno	<pre>public class Retorno{      public enum Resultado{OK,ERROR_1,ERROR_2,ERROR_3,ERROR_4,     ERROR_5, ERROR_6, NO_IMPLEMENTADA};      boolean valorBooleano      int valorEntero;      String valorString;      Resultado resultado;  }</pre>

Pueden definirse tipos de datos (clases) auxiliares.

## 2. Administración de Estudiantes y Libros

### 2.1. Crear Sistema de Gestión

**Firma:** Retorno `crearSistemaDeGestion()`;

**Descripción:** Crea el sistema de gestión inicializando las estructuras a utilizar.

Retornos posibles	
OK	Si pudo inicializar el sistema correctamente.
ERROR	
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

### 2.2. Agregar Estudiante

**Firma:** Retorno `agregarEstudiante(String nombre, String apellido, int numero)`;

**Descripción:** Registra un nuevo estudiante en el sistema. De cada estudiante se conoce su nombre, su apellido y su número (identificador único). El número debe ser mayor a cero y menor o igual a 500000.

Retornos posibles	
OK	Si pudo registrar el estudiante.
ERROR	<ol style="list-style-type: none"><li>1. Si el nombre o el apellido son vacíos o null.</li><li>2. Si el número es menor o igual a cero o mayor a 500000.</li><li>3. Si ya existe un estudiante con ese número.</li></ol>
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

### 2.3. Obtener Estudiante

**Firma:** Retorno `obtenerEstudiante(int numero)`;

**Descripción:** Dado un número, obtiene los datos del estudiante que tiene dicho número y los carga en el valor String del Retorno con el siguiente formato: nombre#apellido#numero. Ejemplo: Ana#Perez#123123

Retornos posibles	
OK	Si pudo encontrar al estudiante con ese número.
ERROR	<ol style="list-style-type: none"><li>1. Si el número es menor o igual a cero o mayor a 500000.</li><li>2. Si no existe un estudiante con ese número.</li></ol>
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

### 2.4. Eliminar Estudiante

**Firma:** Retorno `eliminarEstudiante(int numero)`;

**Descripción:** Elimina el estudiante que tenga el número indicado. Solo se puede eliminar si nunca realizó préstamos o reservas.

Retornos posibles	
OK	Si pudo eliminar el estudiante.
ERROR	<ol style="list-style-type: none"><li>1. Si el número es menor o igual a cero o mayor a 500000.</li><li>2. Si no existe un estudiante con ese número.</li><li>3. Si tiene préstamos o reservas.</li></ol>
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

## 2.5. Agregar Libro

**Firma:** Retorno agregarLibro(String nombre, String ISBN, String categoría, int cantidad);

**Descripción:** Registra un libro en el sistema. De cada uno se conoce: su nombre, ISBN (único), su categoría y la cantidad de libros en stock. La cantidad debe ser un número mayor que cero.

Retornos posibles	
OK	Si se pudo registrar el libro
ERROR	<ol style="list-style-type: none"><li>1. Si alguno de los parámetros es vacío o null.</li><li>2. Si ya existe un libro con ese ISBN.</li><li>3. Si la cantidad es menor o igual a cero.</li></ol>
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

## 2.6. Prestar Libro

**Firma:** Retorno prestarLibro(String ISBN, int numero);

**Descripción:** Registra el préstamo de un ejemplar de un libro a un estudiante. Cada estudiante solo puede tener en préstamo un ejemplar del mismo libro al mismo tiempo (préstamo activo) y como máximo, 8 préstamos simultáneos (activos). En caso de realizarse el préstamo correctamente, se debe decrementar el stock del libro en 1, al crear el préstamo su estado es "activo". Solo se puede realizar el préstamo si quedan ejemplares disponibles en stock.

Retornos posibles	
OK	Si se pudo prestar el libro.
ERROR	<ol style="list-style-type: none"><li>1. Si el ISBN es vacío o null.</li><li>2. Si no existe un libro con ese ISBN.</li><li>3. Si el número es menor o igual a cero o mayor a 500000.</li><li>4. Si no existe un estudiante con ese número.</li><li>5. Si el stock de ese libro es cero.</li><li>6. Si ya existe un préstamo activo de ese libro para ese estudiante o si ya tiene 8 préstamos activos.</li></ol>
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

## 2.7. Reservar Libro

**Firma:** Retorno reservarLibro(String ISBN, int numero);

**Descripción:** Registra una reserva de un ejemplar de un libro para un estudiante. Solo se puede realizar una reserva si no existe stock disponible en el momento para realizar el préstamo.

Retornos posibles	
OK	Si se pudo reservar el libro.
ERROR	<ol style="list-style-type: none"><li>1. Si el ISBN es vacío o null.</li><li>2. Si no existe un libro con ese ISBN.</li><li>3. Si el número es menor o igual a cero o mayor a 500000.</li><li>4. Si no existe un estudiante con ese número.</li><li>5. Si el stock de ese libro es mayor que cero.</li></ol>
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

## 2.8. Devolver Libro

**Firma:** `Retorno devolverLibro(String ISBN, int numero);`

**Descripción:** Registra la devolución de un libro cambiando el estado de un préstamo de “activo” a “finalizado” e incrementando el stock de dicho libro en caso de que corresponda. Si existen reservas para ese libro en lugar de incrementar el stock debe crear un préstamo para el primer estudiante que haya realizado la reserva (el que solicitó antes recibirá el libro antes).

Retornos posibles	
<b>OK</b>	Si se pudo realizar la devolución.
<b>ERROR</b>	<ol style="list-style-type: none"> <li>1. Si el ISBN es vacío o null.</li> <li>2. Si no existe un libro con ese ISBN.</li> <li>3. Si el número es menor o igual a cero o mayor a 500000.</li> <li>4. Si no existe un estudiante con ese número.</li> <li>5. Si no existe un préstamo activo de ese libro para ese estudiante.</li> </ol>
<b>NO_IMPLEMENTADA</b>	Cuando aún no se implementó. Es el tipo de retorno por defecto.

## 2.9. Eliminar Libro

**Firma:** `Retorno eliminarLibro(String ISBN);`

**Descripción:** Elimina el libro del sistema. Solo se pueden eliminar libros si nunca se realizaron préstamos de dicho libro.

Retornos posibles	
<b>OK</b>	Si pudo eliminar el libro.
<b>ERROR</b>	<ol style="list-style-type: none"> <li>1.- Si el ISBN es vacío o null.</li> <li>2.- Si se realizaron préstamos de dicho libro.</li> </ol>
<b>NO_IMPLEMENTADA</b>	Cuando aún no se implementó. Es el tipo de retorno por defecto.

## 3. Listados y reportes

Los listados y reportes deben cargarse en el valor string del resultado Retorno, separando cada dato por un “#” y cada conjunto de datos por un “|” (Ver ejemplos).

### 3.1. Listar Estudiantes

**Firma:** `Retorno listarEstudiantes();`

**Descripción:** Se listan los datos de los estudiantes ordenados creciente por número, cargando el resultado en el valor String del retorno.

Retornos posibles	
<b>OK</b>	Si se retorna el listado correctamente.
<b>ERROR</b>	No hay errores
<b>NO_IMPLEMENTADA</b>	Cuando aún no se implementó. Es el tipo de retorno por defecto.

**NOTA:** Esta operación debe realizarse recorriendo una sola vez la estructura escogida.

**EJEMPLO DE CARGA DE RESULTADO:**

Formato: `nombre1#apellido1#numero1|nombre2#apellido2#numero2`      `numero1 < numero2`

Ejemplo: “Ana#Santos#111111|Pedro#Perez#215481”

### 3.2. Listar Libros

**Firma:** Retorno `listarLibros()`;

**Descripción:** Se listan los datos de todos los libros de la biblioteca ordenados creciente por ISBN cargando el resultado en el valor String del retorno.

Retornos posibles	
<b>OK</b>	Si se muestran todos los libros de dicha
<b>ERROR</b>	No hay errores
<b>NO_IMPLEMENTADA</b>	Cuando aún no se implementó. Es el tipo de retorno No por defecto.

Formato: `nombre1#ISBN1#categoria1|nombre2#ISBN2#categoria2`      `ISBN1 < ISBN2`

Ejemplo:

Diseño y Análisis#978-0132316810#Ciencia de la Computación|Estructuras de Datos en Java#978-0134093413#Programación

### 3.3. Listar Libros por categoría

**Firma:** Retorno `listarLibros(String categoria)`;

**Descripción:** Se listan los datos de los libros de la categoría ordenados creciente por ISBN, cargando el resultado en el valor string del retorno.

Retornos posibles	
<b>OK</b>	Si se muestran todos los libros de dicha
<b>ERROR</b>	1. Si la categoría es vacía o null.
<b>NO_IMPLEMENTADA</b>	Cuando aún no se implementó. Es el tipo de retorno No por defecto.

**NOTA:** Esta operación debe ser realizada en forma recursiva.

Formato: `nombre1#ISBN1#categoria1|nombre2#ISBN2#categoria1`

Ejemplo:

Análisis y diseño de orientación a objetos#978-1134294411#Programación|Estructuras de datos en Java#999-0134093413#Programación

### 3.4. Listar prestamos de un estudiante

**Firma:** Retorno `listarPrestamos(int numero)`;

**Descripción:** Lista todos los préstamos del estudiante ordenados cronológicamente (primero el más reciente) cargando el resultado en el valor String del retorno. De cada préstamo se debe indicar los datos del libro y el estado del préstamo.

Retornos posibles	
<b>OK</b>	Si listan todos los préstamos del estudiante
<b>ERROR</b>	1. Si el número es menor o igual a cero o mayor a 500000. 2. Si no existe un estudiante con ese número.
<b>NO_IMPLEMENTADA</b>	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Formato: `nombreLibro1#ISBN1#categoria1#estado|nombreLibro2#ISBN2#categoria2#estado`

Ejemplo: Orientación a objetos#999-1134294411#Programación#activo|Diseño y Análisis#978-0132316810#Ciencia de la Computación#finalizado



### 3.5. Libros más prestados

**Firma:** `Retorno librosMásPrestados();`

**Descripción:** Se deben listar el/los libros más prestados ordenado por ISBN, cargando el resultado en el valor String del retorno. En caso de existir libros con la misma cantidad "máxima de préstamos", se deberán mostrar todos.

Retornos posibles	
<b>OK</b>	Si listan los libros más prestados
<b>ERROR</b>	
<b>NO_IMPLEMENTADA</b>	Cuando aún no se implementó. Es el tipo de retorno por defecto.

**Formato:** `nombreLibro1#ISBN1#categoría1#cantidad | nombreLibro2#ISBN2# categoría2#cantidad`

Ejemplo: ISBN1 < ISBN2

Diseño y Análisis#966-1134294411#Programación#5 | Orientación a objetos#978-0132316810#Ciencia de la Computación#5

### 3.6. Deshacer las últimas n eliminaciones de libros

**Firma:** `Retorno deshacerEliminaciones(int n);`

**Descripción:** Se deben deshacer las ultimas n eliminaciones de libros realizadas en el sistema. Si existen menos de n eliminaciones, se deshacen todas las realizadas. Se deberá cargar el resultado de los libros recuperados en el valor String del retorno, según el orden de recuperación (primer mostrar el último cancelado y recuperado)

Retornos posibles	
<b>OK</b>	Si se deshacen las eliminaciones indicadas
<b>ERROR</b>	1. Si n es menor o igual a cero.
<b>NO_IMPLEMENTADA</b>	Cuando aún no se implementó. Es el tipo de retorno por defecto.

**Formato:** `nombreLibro1#ISBN1#categoría1 | nombreLibro2#ISBN2#categoría2`

Ejemplo para n=2:

Orientación a objetos#999-1134294411#Programación | Diseño y Análisis#978-0132316810#Programación

### 3.7. Cantidad prestamos activos

**Firma:** `Retorno cantidadPrestamosActivos(String ISBN);`

**Descripción:** Se debe retornar en valor entero la cantidad de préstamos activos del libro indicado.

Retornos posibles	
<b>OK</b>	Si se retorna la cantidad de préstamos activos.
<b>ERROR</b>	1.- Si el ISBN es vacío o null.
<b>NO_IMPLEMENTADA</b>	Cuando aún no se implementó. Es el tipo de retorno por defecto.

### 3.8. Ranking de categorías

**Firma:** Retorno prestamosXCategoría();

**Descripción:** Se deberán listar la cantidad de libros prestados (activos y finalizados) por cada categoría en orden alfabético, cargando el resultado de los libros en el valor string del retorno.

Retornos posibles	
OK	Si se muestran las cantidades de prestamos por categoría.
ERROR	
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

**Formato:** categoría1#cantidad1|categoría2#cantidad2

Ejemplo:

Análisis matemático#3|Programación#18

## PRIMERA ENTREGA

Para la primera entrega se solicita:

- 1) Representación gráfica de las estructuras seleccionadas para resolver TODO el problema planteado (ver ejemplo disponible en Aulas). Justificar las decisiones de diseño adoptadas (las mismas deben estar fundamentadas a partir de los requerimientos del problema).
- 2) Especificación de pre y post condiciones de TODAS las operaciones 2.X que conforman la interfaz “Obligatorio”
- 3) Implementar las operaciones 2.1, 2.2, 2.3, 2.4, 2.5, 3.1, 3.2, 3.3
- 4) Realizar y entregar juego de pruebas que evidencien el correcto funcionamiento de dichas operaciones. Se deberá adjuntar una hoja (.pdf) con el resultado, detallando – en caso de existir - las funcionalidades que no cumplen con los requisitos planteados.

Se deberá entregar un .ZIP con el proyecto (implementación, pre y post condiciones y pruebas) y un archivo .PDF con la representación de la arquitectura escogida y el resultado de las pruebas. Se valorará la eficiencia y adecuación de esta al contexto planteado.

**Nota: previo a la semana se entrega, se compartirá un juego de pruebas reducido.**

## SEGUNDA ENTREGA

Para la segunda entrega se solicita:

- 1) Representación gráfica - ajustada - final.
- 2) Implementar de TODAS las operaciones de la interfaz “Obligatorio”
- 3) Realizar y entregar juego de pruebas definitivo que evidencie el correcto funcionamiento de TODAS las operaciones.


Se deberá entregar un .ZIP con el proyecto (implementación, pre y post condiciones y pruebas) y un archivo .PDF con la representación de la arquitectura final escogida. Se valorará la eficiencia y adecuación de esta al contexto planteado.

## RECORDATORIO: IMPORTANTE PARA LA ENTREGA

- **Obligatorios**

La entrega de los obligatorios será en formato digital online, a excepción de algunas materias que se entregarán en Bedelía y en ese caso recibirá información específica en el dictado de la misma.

Los principales aspectos a destacar sobre la **entrega online de obligatorios** son:

1. Ingresá al sistema de Gestión.
2. En el menú, seleccioná el ítem “Evaluaciones” y la instancia de evaluación correspondiente, que figura bajo el título “Inscripto”.
3. Para iniciar la entrega hacé clic en el ícono: 
4. Ingresá el número de estudiante de cada uno de los integrantes y hacé clic en “Agregar”. El sistema confirmará que los integrantes estén inscriptos al obligatorio y, de ser así, mostrará el nombre y la fotografía de cada uno de ellos. Una vez agregados todos los integrantes, hacé clic en “Crear equipo”.

**Cualquier integrante podrá:**

- **Modificar la integración del equipo.**
  - **Subir el archivo de la entrega.**
5. Seleccioná el archivo que deseás entregar. Verificá el nombre del archivo que aparecerá en la pantalla y hacé clic en “Subir” para iniciar la entrega. Cada equipo (hasta 2 estudiantes) debe entregar **un único archivo en formato zip o rar** (los documentos de texto deben ser pdf, y deben ir dentro del zip o rar). El archivo a subir debe tener **un tamaño máximo de 40mb**

Cuando el archivo quede subido, se mostrará el nombre generado por el sistema (1), el tamaño y la fecha en que fue subido.

6. El sistema enviará un e-mail a todos los integrantes del equipo informando los detalles del archivo entregado y confirmando que la entrega fue realizada correctamente.
7. Podés cerrar la pestaña de entrega y continuar utilizando Gestión o salir del sistema.
8. La **hora tope para subir el archivo será las 21:00** del día fijado para la entrega.
9. La entrega se podrá realizar desde cualquier lugar (ej. hogar del estudiante, laboratorios de la Universidad, etc).
10. Aquellos de ustedes que presenten alguna dificultad con su inscripción o tengan inconvenientes técnicos, por favor contactarse con la Coordinadora o Coordinación adjunta antes de las 20:00hs. del día de la entrega, a través de los mails, [alamon@ort.edu.uy](mailto:alamon@ort.edu.uy) y [fernandez\\_ma@ort.edu.uy](mailto:fernandez_ma@ort.edu.uy), o telefónicamente al 29021505 - int 1156 u 1138.