

UTN FRRO

Lenguaje de programación Java



Integrantes:

Cano, Lautaro - Legajo : 44961

Ponce, Tomás Eduardo - Legajo : 44954

Docentes :

Adrian Meca

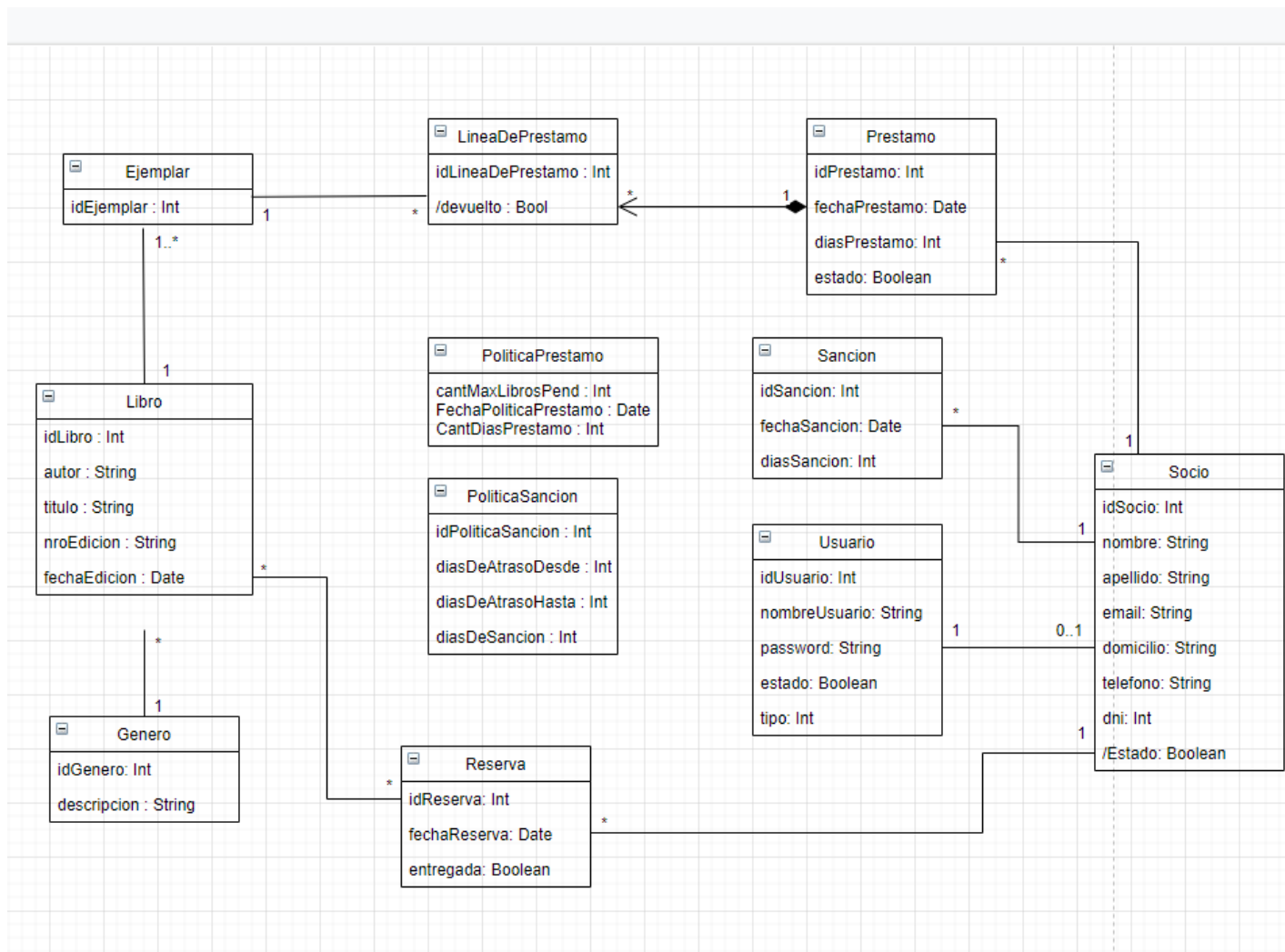
Ricardo Tabacman

Comisión 302, Año 2021

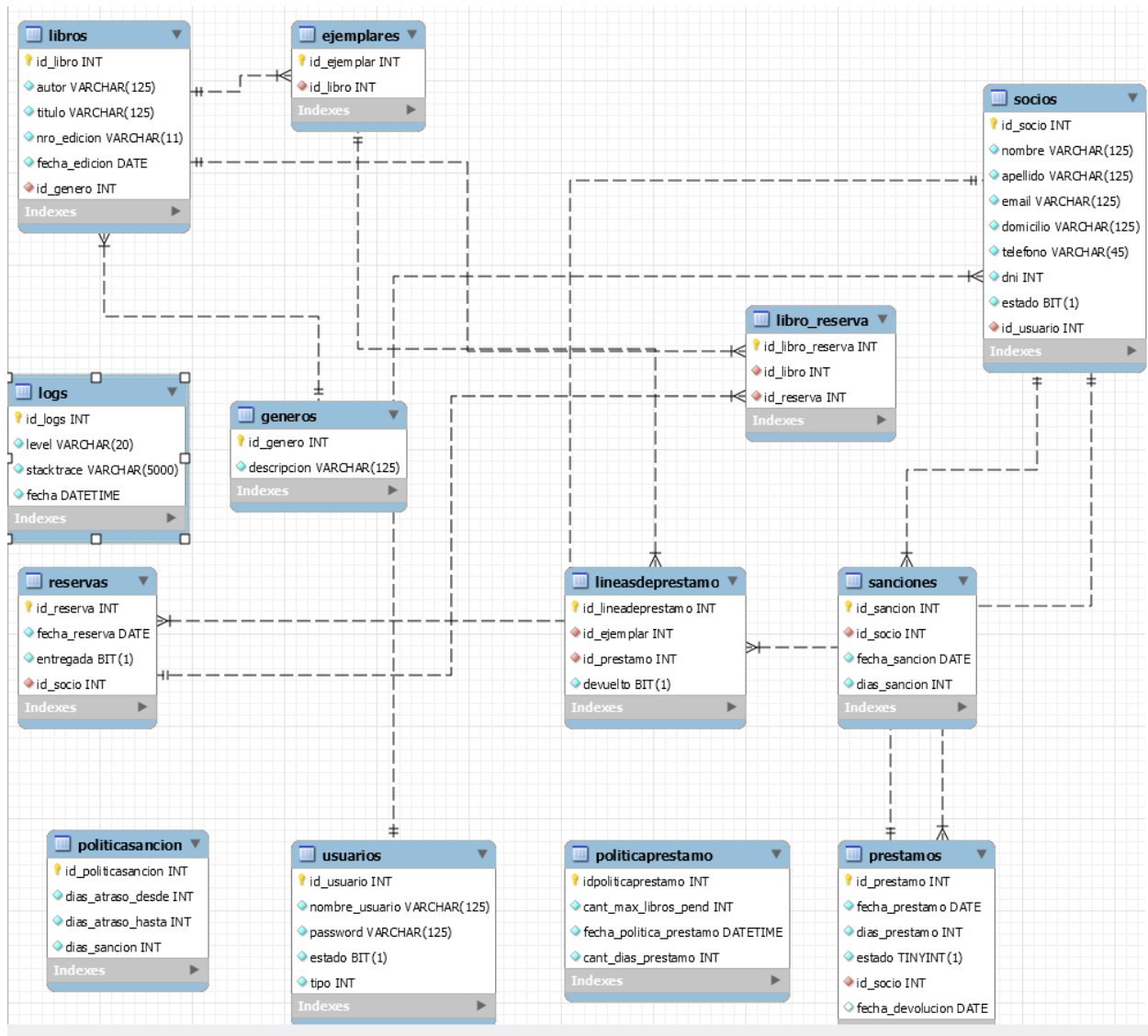
SISTEMA FINAL

El tema elegido fue un sistema de reserva, retiro y devolución de libros de una biblioteca. A continuación, en las diferentes secciones, se describe la documentación correspondiente al sistema.

Diagrama de dominio



Modelo de datos



Listado de casos de uso

- 1) Realizar préstamo (abarca la reserva, retiro y devolución)
- 2) Realizar Reserva (Socio)
- 3) Retirar reserva (Bibliotecario)
- 4) Realizar Devolución (Bibliotecario)
- 5) Ver Reservas (Socio)
- 6) Ver Préstamos (Socio)
- 7) Modificar datos personales (Socio)

Y los casos de uso correspondientes al ABMC de:

-Libro (Administrador)

-Género (Administrador)

-Socio (Administrador)

-Usuario (Administrador)

-Política Préstamo (Administrador)

-Politica Sanción (Administrador)

CU resumen re-estructurado

Código y Nombre del CASO DE USO: CURSRE-Realizar préstamo de libro

Dimensiones de clasificación:

Nivel	Estructura	Alcance	Caja	Instanciación	Interacción
Resumen	Re-estructurado	Sistema	Negra	Real	Semántico

Meta del CASO DE USO:

Actor Primario: Socio, Bibliotecario, Administrador.

PRECONDICIONES :

El socio se encuentra logueado.

El bibliotecario se encuentra logueado.

Todas las categorías de libros deben estar cargadas.

Todos los libros con sus ejemplares deben estar cargados

Todas las políticas de sanción deben estar cargadas.

Todas las políticas de préstamo deben estar cargadas.

Todos los socios deben estar cargados.

Todos los usuarios deben estar cargados

Todos los préstamos con sus líneas de préstamo deben de estar cargadas.

Todas las reservas deben de estar cargadas

Todas las sanciones deben de estar cargadas

DISPARADOR: El socio desea reservar un libro.

CAMINO BÁSICO:

1- El socio reserva libros invocando al CUU-1.1- Reservar libros.

2- El bibliotecario entrega los libros reservados por el socio invocando al CUU-1.2- Realizar préstamo.

3- El bibliotecario registra la devolución de los libros que el socio llevó invocando al CUU-1.3- Registrar devolución de libros

CAMINOS ALTERNATIVOS:

***.1.** El socio desea configurar sus datos personales invocando al CUU-1.4- Configurar datos personales.

***.2.** El administrador desea modificar las políticas de sanción invocando al CUU-1.5-

Modificar políticas de sanción.

***3.** El administrador desea modificar las políticas de préstamo invocando al CUU-1.6- Modificar políticas de préstamo.

1.a. No se puede agregar reserva porque no hay disponibilidad de libros seleccionados.

1.a.1 Sistema informa la situación. Fin del CU.

2.a. No se puede retirar libros porque no hay disponibilidad de ejemplares.

2.a.1 Sistema informa la situación. Fin del CU.

POSTCONDICIONES :

Éxito: El bibliotecario registró la devolución de libros.

Fracaso: No se registró la devolución de libros

Éxito alternativo: El bibliotecario registró la devolución de libros y el socio fue sancionado.

CU de usuario principales

Código y Nombre del CASO DE USO: 1.1- Reservar libros

Dimensiones de clasificación:

Nivel	Estructura	Alcance	Caja	Instanciación	Interacción
Usuario	Sin estructurar	Sistema	Negra	Real	Semántico

Meta del CASO DE USO: Reservar uno o más libros.

Actor Primario: Socio.

PRECONDICIONES :

El socio se encuentra logueado.

Todas las categorías de libros deben estar cargadas.

Todos los libros con sus ejemplares deben estar cargados

Todas las políticas de sanción deben estar cargadas.

Todas las políticas de préstamo deben estar cargadas.

Todos los socios deben estar cargados.

Todos los usuarios deben estar cargados

Todos los prestamos con sus lineas de préstamo deben de estar cargadas.

Todas las reservas deben de estar cargadas

Todas las sanciones deben de estar cargadas

DISPARADOR: El socio desea reservar un libro.

CAMINO BÁSICO:

1- El sistema muestra los libros disponibles para reservar y socio selecciona un libro.

2- Luego de seleccionar todos los libros a reservar socio ingresa al carrito de reservas y selecciona la opción conjunta (todos los libros deben estar disponibles), el sistema muestra las fechas disponibles para su retiro.

3- Socio selecciona una fecha y realiza la reserva conjunta, sistema registra la reserva.

- Se repite 1 para cuantos libros quiera reservar.
- La/s fecha/s de reserva deben ser como máximo 2 meses posteriores a la actual, y no más, por regla del negocio.

CAMINOS ALTERNATIVOS:

***1.** El socio se desloguea. Fin del CU

1.a El socio no selecciona ningún libro.

1.a.1 Sistema notifica y vuelve al paso 1.

1.b <Previo> Socio selecciona una categoría de libro.

1.b.1 Sistema muestra libros disponibles para esa categoría y continúa en paso 1

2.a. El socio no selecciona ninguna fecha para retirar su reserva.

2.a.1 Sistema notifica y vuelve al paso 2

2.b. El socio selecciona la opción individual

2.b.1 Sistema muestra las fechas disponibles para el retiro individual de cada libro.

2.b.2 Socio selecciona las fechas para cada libro, sistema registra las reservas.

3.a. El socio está sancionado.

3.a.1 Sistema notifica, no se registra la reserva.

3.a.2 Fin del CU.

3.b. El socio tiene préstamos atrasados pendientes de devolución.

3.b.1 Sistema notifica, no se registra la reserva.

3.b.2 Fin del CU.

3.c. Los libros seleccionados ya no están disponibles para la fecha ingresada.

3.c.1 Sistema notifica, no se registra la reserva.

3.c.2 Fin del CU.

POSTCONDICIONES :

Éxito: Se registró la reserva.

Fracaso: No se registró la reserva.

Éxito alternativo: Se registraron las reservas.

Código y Nombre del CASO DE USO: 1.2- Realizar préstamo.

Dimensiones de clasificación:

Nivel	Estructura	Alcance	Caja	Instanciación	Interacción
Usuario	Sin estructurar	Sistema	Negra	Real	Semántico

Meta del CASO DE USO: Retirar libros reservados

Actor Primario: Bibliotecario, Socio.

PRECONDICIONES :

El bibliotecario se encuentra logueado.

Todas las categorías de libros deben estar cargadas.

Todos los libros con sus ejemplares deben estar cargados

Todas las políticas de sanción deben estar cargadas.

Todas las políticas de préstamo deben estar cargadas.

Todos los socios deben estar cargados.

Todos los usuarios deben estar cargados

Todos los préstamos con sus líneas de préstamo deben de estar cargadas.

Todas las reservas deben de estar cargadas

Todas las sanciones deben de estar cargadas

DISPARADOR: El socio desea retirar su reserva.

CAMINO BÁSICO:

1- El Bibliotecario ingresa ID de socio, sistema muestra las reservas realizadas por el socio.

2- Bibliotecaria selecciona la reserva indicada y confirma la realización del préstamo.

Sistema registra el préstamo, la entrega de la reserva y notifica el plazo del préstamo.

CAMINOS ALTERNATIVOS:

*.1. El bibliotecario se desloguea. Fin del CU

1.a La ID del socio no existe.

1.a.1 Sistema indica la situación. Vuelve al paso 1.

1.b Socio no realizó ninguna reserva.

1.b.1 Sistema indica la situación. Fin del CU

2.a. Bibliotecario no selecciona ninguna reserva.

2.a.1 Sistema notifica y vuelve al paso 2

2.b Bibliotecario cancela la reserva.

2.b.1 Sistema notifica. Fin de CU.

2.c. Los libros seleccionados y los ya retirados por el socio superan el límite establecido.

2.c.1 Sistema notifica, no se concreta el préstamo.

2.c.2 Fin del CU.

2.d. El socio está sancionado.

2.d.1 Sistema notifica, no se concreta el préstamo.

3.d.2 Fin del CU.

2.e. El socio tiene préstamos atrasados pendientes de devolución.

2.e.1 Sistema notifica, no se concreta el préstamo.

2.e.2 Fin del CU.

2.e. La fecha de la reserva seleccionada no coincide con la fecha actual.

2.e.1 Sistema notifica, no se concreta el préstamo.

2.e.2 Fin del CU.

2.e. No hay ejemplares disponibles para los libros.

2.e.1 Sistema notifica, no se concreta el préstamo.

2.e.2 Fin del CU.

POSTCONDICIONES :

Éxito: Se registró el préstamo, se actualizó el estado de la reserva a “entregada”.

Fracaso: No se registró ningún préstamo.

Éxito alternativo:

Código y Nombre del CASO DE USO: 1.3- Registrar devolución.

Dimensiones de clasificación:

Nivel	Estructura	Alcance	Caja	Instanciación	Interacción
Usuario	Sin estructurar	Sistema	Negra	Real	Semántico

Meta del CASO DE USO: Devolver libros prestados

Actor Primario: Bibliotecario, Socio.

PRECONDICIONES :

El bibliotecario se encuentra logueado.

Todas las categorías de libros deben estar cargadas.

Todos los libros con sus ejemplares deben estar cargados

Todas las políticas de sanción deben estar cargadas.

Todas las políticas de préstamo deben estar cargadas.

Todos los socios deben estar cargados.

Todos los usuarios deben estar cargados

Todos los préstamos con sus líneas de préstamo deben de estar cargadas.

Todas las reservas deben de estar cargadas

Todas las sanciones deben de estar cargadas

DISPARADOR: El socio desea devolver los libros que tenía en préstamo.

CAMINO BÁSICO:

1- El Bibliotecario ingresa ID de socio, sistema muestra los préstamos realizados por el socio.

2- El Bibliotecario selecciona el préstamo indicado, sistema muestra la información del mismo.

3- Luego de verificar los libros, el bibliotecario confirma la devolución. Sistema registra la devolución y notifica.

CAMINOS ALTERNATIVOS:

***1.** El bibliotecario se desloguea. Fin del CU

1.a La ID del socio no existe.

1.a.1 Sistema indica la situación. Vuelve al paso 1.

1.b Socio no realizó ningún préstamo.

1.b.1 Sistema indica la situación. Fin del CU

3.a El socio no entregó los libros correctos.

3.a.1 Bibliotecario informa de la situación. Fin del CU

3.b El socio excedió los días de préstamo.

3.b.1 Sistema registra la sanción y notifica. Fin del CU.

POSTCONDICIONES :

Éxito: Se actualizó el estado del préstamo a “devuelto”.

Fracaso: No se pudo devolver el préstamo.

Éxito alternativo: Se actualizó el estado del préstamo a “devuelto”, se registró una sanción para el socio.

Código y Nombre del CASO DE USO: 1.4- Configurar datos personales

Dimensiones de clasificación:

Nivel	Estructura	Alcance	Caja	Instanciación	Interacción
Usuario	Sin estructurar	Sistema	Negra	Real	Semántico

Meta del CASO DE USO:

Actor Primario: Bibliotecario, Socio.

PRECONDICIONES :

El Socio se encuentra logueado.

Todos los socios deben estar cargados.

Todos los usuarios deben estar cargados

DISPARADOR: El socio desea modificar sus datos personales.

CAMINO BÁSICO:

1- El socio ingresa los datos a modificar y confirma. Sistema notifica la actualización de los datos

CAMINOS ALTERNATIVOS:

***1.** El Socio se desloguea. Fin del CU

1.a Socio cancela la modificación..

1.a.1 Socio cancela la modificación. Vuelve al paso 1.

1.b Socio deja algún campo en blanco.

1.b.1 Sistema notifica la situación. Vuelve al paso 1.

POSTCONDICIONES :

Éxito: Se registró la modificación de los datos personales del socio

Fracaso: No se pudo realizar la modificación de los datos personales del socio.

Éxito alternativo:

Código y Nombre del CASO DE USO: 1.5- Modificar políticas de sanción

Dimensiones de clasificación:

Nivel	Estructura	Alcance	Caja	Instanciación	Interacción
Usuario	Sin estructurar	Sistema	Negra	Real	Semántico

Meta del CASO DE USO:

Actor Primario: Administrador.

PRECONDICIONES :

El Administrador se encuentra logueado.

Todas las políticas de sanción se encuentran cargadas

DISPARADOR: El administrador desea modificar la política de sanción actual.

CAMINO BÁSICO:

1- El administrador ingresa los datos a modificar y confirma. Sistema notifica la actualización de los datos

CAMINOS ALTERNATIVOS:

*1. El bibliotecario se desloguea. Fin del CU

1.a Bibliotecario cancela la modificación..

1.a.1 Bibliotecario cancela la modificación. Vuelve al paso 1.

1.b Bibliotecario deja algún campo en blanco.

1.b.1 Sistema notifica la situación. Vuelve al paso 1.

1.c Bibliotecario agrega una nueva política de sanción

1.c.1 Administrador ingresa los datos y confirma. Sistema notifica.

POSTCONDICIONES :

Éxito: Se realizó la modificación de la política de sanción

Fracaso: No se pudo realizar la modificación de la política de sanción

Éxito alternativo: Se agregó una nueva política de sanción

Código y Nombre del CASO DE USO: 1.6- Modificar políticas de préstamo

Dimensiones de clasificación:

Nivel	Estructura	Alcance	Caja	Instanciación	Interacción
Usuario	Sin estructurar	Sistema	Negra	Real	Semántico

Meta del CASO DE USO:

Actor Primario: Administrador.

PRECONDICIONES :

El Administrador se encuentra logueado.

Todas las políticas de préstamos se encuentran cargadas.

DISPARADOR: El administrador desea modificar la política de préstamo actual.

CAMINO BÁSICO:

1- El administrador ingresa los datos a modificar y confirma. Sistema notifica la actualización de los datos

CAMINOS ALTERNATIVOS:

***.1.** El bibliotecario se desloguea. Fin del CU

1.a Bibliotecario cancela la modificación..

1.a.1 Bibliotecario cancela la modificación. Vuelve al paso 1.

1.b Bibliotecario deja algún campo en blanco.

1.b.1 Sistema notifica la situación. Vuelve al paso 1.

1.c Bibliotecario agrega una nueva política de préstamo

1.c.1 Administrador ingresa los datos y confirma. Sistema notifica.

POSTCONDICIONES :

Éxito: Se realizó la modificación de la política de préstamo

Fracaso: No se pudo realizar la modificación de la política de préstamo

Éxito alternativo: Se agregó una nueva política de préstamo

Imágenes del sistema.

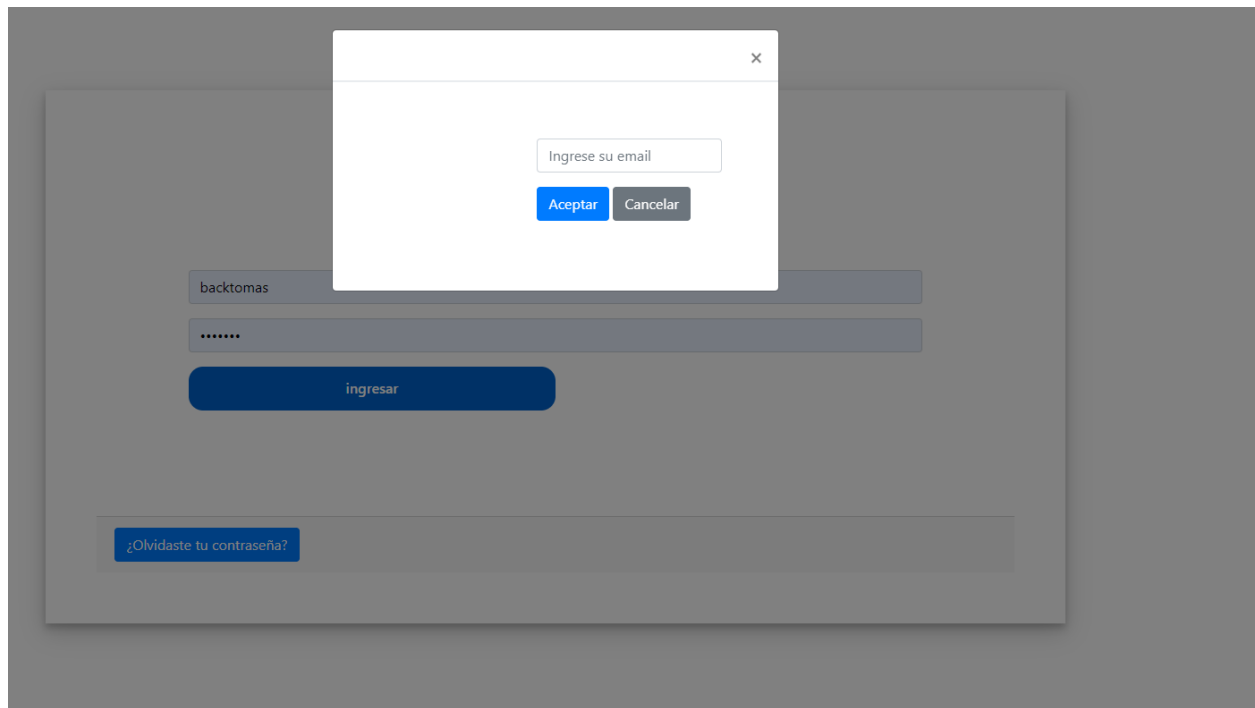
Login

Ingreso



The image shows a login form titled "Sistema de Biblioteca". It contains two input fields: "Nombre de usuario" and "Contraseña". Below these fields is a blue button labeled "ingresar". At the bottom of the form, there is a link that says "¿Olvidaste tu contraseña?".

Recuperación de contraseña



Menu Admin

ABMC Libro

Menú Administrador								Cerrar sesión
ABM Libro	ID Título	Autor	Nro de Edición	Fecha de Edición	Genero	Ejemplares		
	28 El código Da Vinci	Dan Brown	3	2003-03-18	Ciencia ficción	Ver ejemplares	Editar	Eliminar
ABM Género	33 Un mundo feliz	Aldous Huxley	1	1932-04-15	Ciencia ficción	Ver ejemplares	Editar	Eliminar
ABM Socio	34 1984	George Orwell	1	1947-12-11	Ciencia ficción	Ver ejemplares	Editar	Eliminar
ABM Usuario	30 Así habló Zaratustra	Friedrich Nietzsche	1	1883-02-10	Filosofía	Ver ejemplares	Editar	Eliminar
	39 Vigilar y castigar	Michel Foucault	1	1975-02-01	Filosofía	Ver ejemplares	Editar	Eliminar
ABM Política Prestamo	40 La náusea	Jean-Paul Sartre	1	1938-11-11	Filosofía	Ver ejemplares	Editar	Eliminar
ABM Política Sanción	31 El extranjero	Albert Camus	2	1942-06-23	Novela	Ver ejemplares	Editar	Eliminar
Abrir log de errores	32 El principito	Antoine de Saint-Exupéry	1	1943-04-06	Novela	Ver ejemplares	Editar	Eliminar
	35 Ulises	James Joyce	2	1922-02-02	Novela	Ver ejemplares	Editar	Eliminar
	36 Cien años de soledad	Gabriel García Márquez	1	1967-08-10	Novela	Ver ejemplares	Editar	Eliminar
	37 El gran Gatsby	F. Scott Fitzgerald	3	1933-06-13	Novela	Ver ejemplares	Editar	Eliminar
	38 Ficciones	Jorge Luis Borges	2	1944-09-13	Cuento	Ver ejemplares	Aceptar	Cancelar
	Título	Autor	N° de edición	Fecha edición	Ciencia ficción	Cantidad de Ejemp.	Agregar	

ABMC Genero

Menú Administrador				Cerrar sesión
ABM Libro	ID Genero	Descripción		
ABM Género	10	Ciencia ficción		Editar Eliminar
ABM Socio	11	Filosofía		Editar Eliminar
ABM Usuario	12	Novela		Editar Eliminar
ABM Política Prestamo	13	Cuento	Aceptar	Cancelar
ABM Política Sanción		Descripción	Agregar	
ABrir log de errores				

ABMC Socio

Menú Administrador								Cerrar sesión
ABM Libro	ID Nombre	Apellido	DNI	Email	Domicilio	Teléfono	Estado	
ABM Género	9 Lautaro	Cano	41975401	lautarojuancano@gmail.com	Laprida 936	3415350466	true	Editar Eliminar
ABM Socio	10 Tomás	Ponce	41602902	poncetomaseduardo@gmail.	Chaco 1763	3412826061	<input checked="" type="checkbox"/>	Aceptar Cancelar
ABM Usuario	Nombre		Apellido	dni	Email	Domicilio	telefono	Agregar
ABM Política Prestamo								
ABM Política Sanción								
ABrir log de errores								

ABMC Usuario

					Cerrar sesión
ID Tipo	Nombre		Contraseña	Estado	
1 Administrador	admin		admin	true	Editar Eliminar
13 Bibliotecario	bibliotecario		bibliotecario	true	Editar Eliminar
15 Socio	41975401		Cano123	true	Editar Eliminar
16 Socio	41602902	Ponce123	Habilitado	Aceptar	Cancelar
Bibliotecario	Nombre usuario	Contraseña	Habilitado	Agregar	

ABMC Política Prestamo

Cerrar sesión

ID PolíticaPrestamo	CantMax	Fecha	Cantidad Dias Prestamo		
15	5	2021-04-02	7	Editar	Eliminar
16	4			Aceptar	Cancelar
	2021-12-08				
	7				
	Cant. maxima de libros pendientes	Fecha	Cant. dias de prestamo		Agregar

ABMC Política Sanción

Menú Administrador

Cerrar sesión

ID PolíticaSanción	Dias atrasodesde	Dias atrasohasta	Dias sanción		
5	1	5	5	Editar	Eliminar
6	6	15	15	Editar	Eliminar
7	16	365	60	Editar	Eliminar
8	366			Aceptar	Cancelar
	10000000				
	365				
	Días atraso desde	Días atraso hasta	Días de sanción		Agregar

Log de Errores

```

Dec 09, 2021 9:50:10 PM util.Bitacora getBitacora
INFO: Bitacora inicializada
Dec 09, 2021 9:50:10 PM util.Bitacora log
SEVERE: java.lang.NullPointerException
    at servlets.LoginServlet.doGet(LoginServlet.java:69)
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:634)
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:741)
    at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:231)
    at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166)
    at org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:53)
    at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:193)
    at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166)
    at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:202)
    at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:96)
    at org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:541)
    at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:139)
    at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:92)
    at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:74)
    at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:343)
    at org.apache.coyote.http11.Http11Processor.service(Http11Processor.java:367)
    at org.apache.coyote.AbstractProcessorLight.process(AbstractProcessorLight.java:65)
    at org.apache.coyote.AbstractProtocol$ConnectionHandler.process(AbstractProtocol.java:860)
    at org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.doRun(NioEndpoint.java:1598)
    at org.apache.tomcat.util.net.SocketProcessorBase.run(SocketProcessorBase.java:49)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
    at org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61)
    at java.lang.Thread.run(Thread.java:748)

Dec 09, 2021 9:51:17 PM util.Bitacora log
SEVERE: javax.mail.AuthenticationFailedException: 534-5.7.14 <https://accounts.google.com/signin/continue?sarp=1&sc=1&plt=AKgnsbu
534-5.7.14 pCtgkFbr7DQH3wYSXXh7Mf4mqP5nKgHioy6cxbwva06bXja8msClHY2YaI690MhX_rte
534-5.7.14 eESiuJa6_bBVVdPaPGT5F5zyGbUpDp03yzT7ZdYFin9QaROAKVvx5d0HQI4f4QUoe>
534-5.7.14 Please log in via your web browser and then try again.
534-5.7.14 Learn more at
534 5.7.14 https://support.google.com/mail/answer/78754 y6sm610016qtn.23 - gsmtpt

    at com.sun.mail.smtp.SMTPTransport$Authenticator.authenticate(SMTPTransport.java:965)
    at com.sun.mail.smtp.SMTPTransport.authenticate(SMTPTransport.java:876)
    at com.sun.mail.smtp.SMTPTransport.protocolConnect(SMTPTransport.java:780)
    at javax.mail.Service.connect(Service.java:366)
    at javax.mail.Service.connect(Service.java:246)
    at servlets.Servlet.enviarConMail(Servlet.java:182)
    at servlets.FinalizarReservaServlet.doPost(FinalizarReservaServlet.java:153)
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:660)
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:741)
    at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:231)
    at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166)
    at org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:53)
    at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:193)
    at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166)
    at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:202)
    at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:96)
    at org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:541)
    at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:139)
    at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:92)
    at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:74)
    at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:343)
    at org.apache.coyote.http11.Http11Processor.service(Http11Processor.java:367)
    at org.apache.coyote.AbstractProcessorLight.process(AbstractProcessorLight.java:65)
    at org.apache.coyote.AbstractProtocol$ConnectionHandler.process(AbstractProtocol.java:860)

```

Menú Socio

Libros disponibles para reservar

Prestamos actuales

Devolución atrasada


Préstamo N° 14

Fecha: 2021-12-08


Libros:

- El código Da Vinci

Datos personales

						Cerrar sesión
Nombre	Apellido	DNI	Email	Domicilio	Teléfono	
Lautaro	Cano	41975401	lautarojuancano@gmail.com	Laprida	3415350466	<div>Aceptar Cancelar</div>

Carro de reserva

						Cerrar sesión
Título	Autor	Nro de Edición	Fecha de Edición	Genero		
El código Da Vinci	Dan Brown	3	2003-03-18	Ciencia ficción	<div>Quitar libro</div>	
1984	George Orwell	1	1947-12-11	Ciencia ficción	<div>Quitar libro</div>	
Tipo de reserva:						
<input checked="" type="radio"/> Conjunta						
<input type="radio"/> Individual						
Fecha de retiro						
Finalizar Reservas						

Cerrar sesión

Titulo	Autor	Nro de Edición	Fecha de Edición	Genero	
El código Da Vinci	Dan Brown	3	2003-03-18	Ciencia ficción	Quitar libro
1984	George Orwell	1	1947-12-11	Ciencia ficción	Quitar libro

Tipo de reserva:

☐ Conjunta

☒ Individual

El código Da Vinci

Autor: Dan Brown

Edición: 2003-03-18

Fecha de retiro

1984

Autor: George Orwell

Edición: 1947-12-11

Fecha de retiro

Finalizar Reservas

Menú Bibliotecario

Retiro de Reserva

Cerrar sesión

ID de socio:

id

ID Reserva	Fecha Reserva	Libros	Socio
No hay resultados			

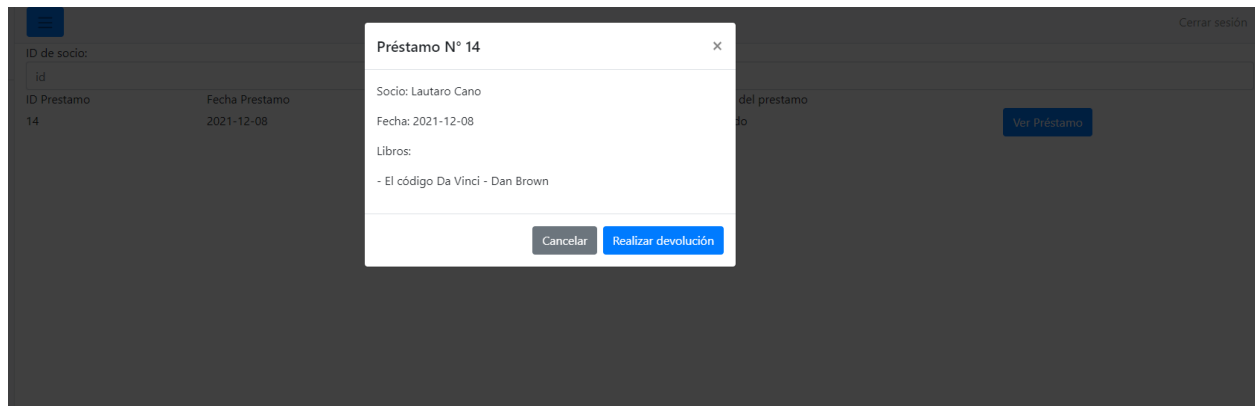
Devolución

Cerrar sesión

ID de socio:

id

ID Prestamo	Fecha Prestamo	Días de prestamo	Estado del prestamo	
14	2021-12-08	1	Atrasado	Ver Préstamo



Fragmentos de código fuente del sistema.

Capa de UI

ReservaServlet

```

@WebServlet("/ReservaServlet")
public class ReservaServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public ReservaServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        if (Servlet.VerificarSessionYUsuario(request, response, Usuario.tipoUsuario.Socio)) {
            // Set standard HTTP/1.1 no-cache headers.
            response.setHeader("Cache-Control", "private, no-store, no-cache, must-revalidate");

            // Set standard HTTP/1.0 no-cache header.
            response.setHeader("Pragma", "no-cache");
            LibroLogic ll = new LibroLogic();
            GeneroLogic gl = new GeneroLogic();
            Genero genero = null;
            ArrayList<Libro> listaLibros = new ArrayList<Libro>();
            try {
                if (request.getParameter("genero")!=null) {
                    try {
                        genero = gl.getOne(Integer.parseInt(request.getParameter("genero")));
                    }
                    catch (NumberFormatException e) {
                        request.setAttribute("mensaje", "No se pudo obtener el genero indicado");
                    }
                }
                if (genero!=null) {
                    if (request.getParameter("libro")!=null) {
                        listaLibros = ll.getAllByTituloAndGenero(request.getParameter("libro"), genero);
                    }
                    else listaLibros = ll.getAllByGenero(genero);
                }
            }
            else if (request.getParameter("libro")!=null) {
                listaLibros = ll.getAllByTitulo(request.getParameter("libro"));
            }
            else listaLibros = ll.getAll();
            request.setAttribute("ListaGeneros", gl.getAll());
            if (request.getSession().getAttribute("libros") != null) {
                @SuppressWarnings("unchecked")
                ArrayList<Libro> librosCarrito=((ArrayList<Libro>)request.getSession().getAttribute("libros"));
                for (Libro l : librosCarrito) {
                    listaLibros.removeIf(libro -> libro.getId() == l.getId());
                }
            }
            request.setAttribute("ListaLibros", listaLibros);
        }
    }
}

```

```

    } catch (SQLException e) {
        Servlet.Log(Level.SEVERE,e, request);
        request.setAttribute("mensaje", "Error en la base de datos.");
        String rootDirectory = request.getSession().getServletContext().getRealPath("/");
        Bitacora.Log(Level.SEVERE, Bitacora.getStackTrace(e), rootDirectory);
    } catch (Exception e) {
        Servlet.Log(Level.SEVERE,e, request);
        request.setAttribute("mensaje", "Ha ocurrido un error durante la ejecución de la operación");
    }
    request.setAttribute("JSP", "Reserva");
    request.getRequestDispatcher("WEB-INF/Socio.jsp").forward(request, response);
}
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
 */
@SuppressWarnings("unchecked")
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    if (Servlet.VerificarSesionYUsuario(request, response, Usuario.tipoUsuario.Socio)) {
        if (request.getParameter("action-type")!=null && request.getParameter("action-type").equals("reservar")) {
            LibroLogic ll=new LibroLogic();
            Libro libro;
            try {
                libro = ll.getOne(Integer.parseInt(request.getParameter("id_libro")));
                if (libro != null) {
                    if(request.getSession().getAttribute("libros") == null) {
                        ArrayList<Libro> libros=new ArrayList<Libro>();
                        request.getSession().setAttribute("libros", libros);
                    }
                    ((ArrayList<Libro>)request.getSession().getAttribute("libros")).add(libro);
                    request.setAttribute("clase-mensaje", "class=\"alert alert-success alert-dismissible fade show\"");
                    request.setAttribute("mensaje", "Libro agregado correctamente");
                }
                else {
                    request.setAttribute("clase-mensaje", "class=\"alert alert-danger alert-dismissible fade show\"");
                    request.setAttribute("mensaje", "Id de libro invalida");
                }
            } catch (NumberFormatException e) {
                request.setAttribute("mensaje", "Error en los datos suministrados.");
            } catch (SQLException e) {
                Servlet.Log(Level.SEVERE,e, request);
                request.setAttribute("mensaje", "Error en la base de datos.");
            } catch (Exception e) {
                Servlet.Log(Level.SEVERE,e, request);
                request.setAttribute("mensaje", "Ha ocurrido un error durante la ejecución de la operación");
            }
        }
        this.doGet(request, response);
    }
}
}

```

ReservaJSP

```

<%@page import="java.util.ArrayList"%>
<%@page import="model.Libro"%>
<%@page import="model.Genero"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<title>Realizar Reservas</title>
</head>
<body>
<%
@SuppressWarnings("unchecked")
ArrayList<Libro> listaLibro=(ArrayList<Libro>)request.getAttribute("ListaLibros");
@SuppressWarnings("unchecked")
ArrayList<Genero> listaGenero=(ArrayList<Genero>)request.getAttribute("ListaGeneros");
String busqueda = "";
String andBusqueda = "";
if (request.getParameter("libro")!=null) {
    busqueda = "?libro="+request.getParameter("libro");
    andBusqueda = "&libro="+request.getParameter("libro");
}
if (listaLibro == null || listaGenero == null || listaLibro.isEmpty()) { %>
<p style="font-size: 16px;">No hay resultados</p>
<%>
else {
%>
<div class="row">

    <div class="col-lg-9">

        <div class="border p-4 mb-4">
<form action="ReservaServlet" method="GET" name="ReservaLibro">
            <%if (request.getParameter("genero")!=null) { %>
<input name="genero" type="hidden" class="form-control" id="genero" value="<%=request.getParameter("genero") %>" required>
            <%> %>
            <div class="input-group">
<%if (request.getParameter("libro")!=null) { %>
                <input name="libro" type="search" class="form-control rounded" placeholder="Búsqueda" aria-label="Search"
                    aria-describedby="search-addon" value="<%=request.getParameter("libro") %>"/>
                <button type="submit" class="btn btn-outline-primary">Buscar</button>
            <%if (request.getParameter("genero")!=null) { %>
<a href="ReservaServlet?genero=<%=request.getParameter("genero") %>" class="btn btn-danger">X</a>
            <%> else { %>|
<a href="ReservaServlet" class="btn btn-danger">X</a>
            <%> %>
            <%> else { %>
                <input name="libro" type="search" class="form-control rounded" placeholder="Búsqueda" aria-label="Search"
                    aria-describedby="search-addon" />
                <button type="submit" class="btn btn-outline-primary">Buscar</button>
            <%> %>
        </div>
    </form>
    </div>

    <div class="row">
        <%for (Libro l : listaLibro) { %>
            <div class="col-lg-4 col-md-6 mb-4">
                <div class="card h-100">
                    <a href="#"></a>

```

```

<div class="card-body">
  <h4 class="card-title">
    <a href="#"><%=l.getTitulo() %></a>
  </h4>
  <h5><%=l.getAutor() %></h5>
  <p class="card-text"><%= "Edición N°"+l.getNroEdicion() %></p>
</div>
<div class="card-footer">
  <button type="button" class="btn btn-primary" data-toggle="modal" data-target="#modal<%=l.getId() %>">
    Ver Libro
  </button>
</div>
</div>
<!-- Modal -->
<div class="modal fade" id="modal<%=l.getId() %>" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel<%=l.getId() %>" aria-hidden="true">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLabel<%=l.getId() %>"><%=l.getTitulo() %></h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="container-fluid">
        <div class="row">
          <div class="col-md-6">
            
          </div>
          <div class="col">
            <p>Autor: <%=l.getAutor() %></p>
            <p>Nro de edición: <%=l.getNroEdicion() %> </p>
            <p>Fecha de edición: <%=l.getFechaEdicion() %> </p>
            <p>Género: <%=l.getGenero().getDescripcion() %></p>
          </div>
        </div>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-dismiss="modal">Cancelar</button>
        <form action="ReservaServlet" method="POST" name="ReservaLibro">
          <input name="id_libro" type="hidden" class="form-control" id="id_libro" placeholder="id libro" value="<%=l.getId() %>" required>
          <button type="submit" name="action-type" value="reservar" class="btn btn-primary">Agregar a reserva</button>
        </form>
      </div>
    </div>
  </div>
<!-- Modal -->
<%= %>
</div>
<!-- /.row -->
</div>
<!-- /.col-lg-9 -->
<div class="col">
  <h3 class="my-4">Filtrar:</h3>
  <div class="list-group">
    <%if (request.getParameter("genero")!=null) { %>
      <a href="ReservaServlet?busqueda =<%=busqueda %>" class="list-group-item">Todos</a>
      <%for (Genero gen : listaGenero) {
        if (Integer.parseInt(request.getParameter("genero"))==gen.getId()) {%>
      <a href="#" class="list-group-item active"><%=gen.getDescripcion() %></a>
      <%}
      else {%>
      <a href="ReservaServlet?genero=<%=gen.getId()+andBusqueda %>" class="list-group-item"><%=gen.getDescripcion() %></a>
      <%}
      }%>
    </div>
    <%= %>
    else {%>
      <a href="#" class="list-group-item active">Todos</a>
      <%for (Genero gen : listaGenero) { %>
      <a href="ReservaServlet?genero=<%=gen.getId()+andBusqueda %>" class="list-group-item"><%=gen.getDescripcion() %></a>
      <%}%>
    <%= %>
  </div>
</div>
<!-- /.row -->
<%= %>
</body>
</html>

```

FinalizarReservaServlet

```

package servlets;

import java.io.IOException;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.logging.Level;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import logic.SocioLogic;
import logic.ReservaLogic;
import logic.BusinessLogicException;
import model.Libro;
import model.LibroReserva;
import model.Reserva;
import model.Socio;
import model.Usuario;

/**
 * Servlet implementation class FinalizarReservaServlet
 */
@WebServlet("/FinalizarReservaServlet")
public class FinalizarReservaServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private static final String prefijoMensaje = "mensaje";
    private int idMensaje;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public FinalizarReservaServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    @SuppressWarnings("unchecked")
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        idMensaje = 0;
        if (Servlet.VerificarSesionYUsuario(request, response, Usuario.tipoUsuario.Socio)) {
            if (request.getSession().getAttribute("libros") != null) {
                ArrayList<Libro> libros = ((ArrayList<Libro>) request.getSession().getAttribute("libros"));
                ArrayList<ArrayList<String>> fechasDisponiblesTotal = new ArrayList<ArrayList<String>>();
                ArrayList<String> fechasDisponiblesLibro;
                if (libros.size() > 0) {
                    ReservaLogic rl = new ReservaLogic();
                    try {
                        for (Libro l : libros) {
                            fechasDisponiblesLibro = rl.getFechasDisponible(l, 2);
                            request.setAttribute("availableDays"+l.getId(), fechasDisponiblesLibro);
                            fechasDisponiblesTotal.add(fechasDisponiblesLibro);
                        }
                        request.setAttribute("availableDays", rl.getFechasDisponible(fechasDisponiblesTotal, 2));
                    } catch (SQLException e) {

```

```

        Servlet.log(Level.SEVERE,e, request);
        request.setAttribute(this.getIdMensaje(), "Error en la base de datos, inténtelo nuevamente en unos minutos.");
    } catch (Exception e) {
        Servlet.log(Level.SEVERE,e, request);
        request.setAttribute("mensaje", "Ha ocurrido un error durante la ejecución de la operación");
    }
    request.setAttribute("JSP", "FinalizarReserva");
    request.getRequestDispatcher("WEB-INF/Socio.jsp").forward(request, response);
}
else {
    request.setAttribute(this.getIdMensaje(), "Carrito vacío, por favor agregue al menos un elemento.");
    request.getRequestDispatcher("ReservaServlet").forward(request,response);
}
}
else {
    request.setAttribute(this.getIdMensaje(), "Carrito vacío, por favor agregue al menos un elemento.");
    request.getRequestDispatcher("ReservaServlet").forward(request,response);
}
}
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
 */
@SuppressWarnings("unchecked")
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    idMensaje = 0;
    if (Servlet.VerificarSesionUsuario(request, response, Usuario.tipoUsuario.Socio)) {
        try {
            if (request.getSession().getAttribute("libros")!=null && request.getParameter("action-type")!=null) {
                if (request.getParameter("action-type").equals("borrar")) {
                    try {
                        int id=Integer.parseInt(request.getParameter("id libro"));
                        ((ArrayList<Libro>)request.getSession().getAttribute("libros")).removeIf(l -> l.getId() == id);
                    } catch (NumberFormatException e) {
                        request.setAttribute(this.getIdMensaje(), "Error en los datos suministrados. Id de libro inválida.");
                    }
                    this.doGet(request, response);
                }
            }
            else if(request.getParameter("action-type").equals("finalizar") && request.getParameter("tipo")!=null) {
                ArrayList<Libro> libros=((ArrayList<Libro>)request.getSession().getAttribute("libros"));
                Socio socio = (Socio) request.getSession().getAttribute("socio");
                Sociologic sl = new Sociologic();
                Reserva reserva;
                ArrayList<LibroReserva> librosReservas;
                LibroReserva lr;
                if (libros.size() > 0) {
                    if (request.getParameter("tipo").equals("conjunta")) {
                        reserva=new Reserva();
                        librosReservas = new ArrayList<LibroReserva>();
                        for(Libro l : libros) {
                            lr=new LibroReserva();
                            lr.setLibro(l);
                            librosReservas.add(lr);
                        }
                        reserva.setLibros(librosReservas);
                    }
                    try {
                        reserva.setFechaReserva(java.sql.Date.valueOf(request.getParameter("fecha")));
                        reserva.setSocio(socio);
                    }
                    try {

```

```

        sl.realizaReserva(reserva);
        request.setAttribute("class="+this.getIdMensaje(), "class=\"alert alert-success alert-dismissible fade show\"");
        request.setAttribute(this.getIdMensaje(), "Reserva guardada.");
        Servlet.enviarConMail(socio.getEmail(), "Reserva Biblioteca", "Ha realizado una reserva con éxito", request);
        request.getSession().setAttribute("libros", null);
    } catch (BusinessLogicException ble) {
        request.setAttribute(this.getIdMensaje(), ble.getMessage());
    } catch (SQLException e) {
        Servlet.log(Level.SEVERE,e, request);
        request.setAttribute(this.getIdMensaje(), "Error en la base de datos, su reserva puede no haber sido realizada.");
    }
}
catch (IllegalArgumentException e) {
    e.printStackTrace();
    request.setAttribute(this.getIdMensaje(), "Fecha no disponible.");
}
}
else if (request.getParameter("tipo").equals("individual")) {
    for(Libro l :libros) {
        reserva=new Reserva();
        librosReservas = new ArrayList<LibroReserva>();
        lr=new LibroReserva();
        lr.setLibro(l);
        librosReservas.add(lr);
        reserva.setLibros(librosReservas);
        try {
            reserva.setFechaReserva(java.sql.Date.valueOf(request.getParameter("fecha"+l.getId())));
            reserva.setSocio(socio);
            try {
                sl.realizaReserva(reserva);
                request.setAttribute("class="+this.getIdMensaje(), "class=\"alert alert-success alert-dismissible fade show\"");
                request.setAttribute(this.getIdMensaje(), "Reserva guardada para el libro"+l.getTitulo()+"\n.");
                Servlet.enviarConMail(socio.getEmail(), "Reserva Biblioteca", "Ha realizado una reserva con éxito", request);
                request.getSession().setAttribute("libros", null);
            } catch (BusinessLogicException ble) {
                request.setAttribute(this.getIdMensaje(), ble.getMessage());
            } catch (SQLException e) {
                Servlet.log(Level.SEVERE,e, request);
                request.setAttribute(this.getIdMensaje(), "Reserva de "+l.getTitulo()+" : Error en la base de datos, su reserva puede no haber sido realizada.");
            }
        } catch (IllegalArgumentException e) {
            request.setAttribute(this.getIdMensaje(), "Reserva de "+l.getTitulo()+" : Fecha no disponible.");
        } finally {
            this.IdMensaje++;
        }
    }
}
}
else {
    request.setAttribute(this.getIdMensaje(), "No se pudo agregar una reserva si el carrito está vacío.");
}
request.getRequestDispatcher("ReservaServlet").forward(request,response);
}
else {
    request.setAttribute(this.getIdMensaje(), "Error en los datos suministrados.");
    this.doGet(request, response);
}
}
else {

```

```

        request.setAttribute(this.getIdMensaje(), "Error en los datos suministrados.");
        this.doGet(request, response);
    }
}
else {
    request.setAttribute(this.getIdMensaje(), "Error en los datos suministrados.");
    this.doGet(request, response);
}
} catch (Exception e) {
    Servlet.log(Level.SEVERE,e, request);
    request.setAttribute("mensaje", "Ha ocurrido un error durante la ejecución de la operación");
    this.doGet(request, response);
}
}
}
private String getIdMensaje() {
    if (idMensaje == 0) {
        return prefijoMensaje;
    }
    else return prefijoMensaje + idMensaje;
}
}
}

```

FinalizarReservaJSP


```

<%@page import="java.util.ArrayList"%>
<%@page import="model.Libro"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Finalizar Reserva</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
    integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z" crossorigin="anonymous">
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
    integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
    crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
    integrity="sha384-refTGAw83Elw2RDu2S0VKaIzap3H66LZH81PoYLFhbGU+6B2p6G7n1u7355sk7LN"
    crossorigin="anonymous"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"
    integrity="sha384-B4gt1jrGC7Jh4AgTPSdUt08vf08shuf57BaghfFPLVyxfvL8/KUEfYiJQWVW+V"
    crossorigin="anonymous"></script>

<link rel="stylesheet" type="text/css" href="CSS/table-style.css">

<link rel="stylesheet" href="//code.jquery.com/ui/1.12.1/themes/base/jquery-ui.css">
<link rel="stylesheet" href="/resources/demos/style.css">
<script src="https://code.jquery.com/jquery-1.12.4.js"></script>
<script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>
</script>

var availableDays = <%=request.getAttribute("availableDays") %>;

function disableDays(date) {
    var ymd = $.datepicker.formatDate('yy-mm-dd', date);
    if ($.inArray(ymd, availableDays) != -1) {
        return [true, "", "Disponible"];
    } else {
        return [false, "", "No disponible"];
    }
}

$( function() {
    $( "#datepicker" ).datepicker({
        minDate: 0,
        maxDate: "+2M",
        changeMonth: true,
        changeYear: true,
        dateFormat: "yy-mm-dd",
        beforeShowDay: disableDays
    });
} );
</script>
<%
@SuppressWarnings("unchecked")
ArrayList<Libro> listaLibro=(ArrayList<Libro>)session.getAttribute("libros");
String datepicker;
for (Libro l : listaLibro) {
    datepicker="\n"#datepicker"+l.getId()+"'";

```

```

%>
<script>
var availableDays<%=l.getId() %> = <%=request.getAttribute("availableDays"+l.getId()) %>;

function disableDays<%=l.getId() %>(date) {
    var ymd = $.datepicker.formatDate('yy-mm-dd', date);
    if(eval("<%= $.isArray(ymd,availableDays"+l.getId()+") != -1" %>)) {
        return [true,"","Disponible"];
    } else {
        return [false, "", "No disponible"];
    }
}

$( function() {
    $( "<%=datepicker%> " ).datepicker({
        minDate: 0,
        maxDate: "+2M",
        changeMonth: true,
        changeYear: true,
        dateFormat: "yy-mm-dd",
        beforeShowDay: eval('disableDays'+<%=l.getId() %>)
    });
} );

</script>
<%> %>

<script>
$(document).ready(function() {
    $("input[name='tipo']").click(function() {
        var test = $(this).val();

        $("div.desc").hide();
        $(eval(test)).show();
    });
});
</script>

</head>
<body>
<div class="table">
    <div class="thead">
        <div class="tr">
            <span class="td">Titulo</span>
            <span class="td">Autor</span>
            <span class="td">Nro de Edición</span>
            <span class="td">Fecha de Edición</span>
            <span class="td">Genero</span>
            <span class="td"></span>
        </div>
    </div>
    <div class="tbody">
        <%
        for (Libro l : listaLibro) {
        %>
        <form class="tr" action="FinalizarReservaServlet" method="POST" name="FinalizarReserva">

            <input name="id_libro" type="hidden" class="form-control" id="id_libro" placeholder="id_libro" value="<%=l.getId() %>" required>
            <span class="td">

```

```

        <span class="td">
            <%=l.getTitulo() %>
        </span>
        <span class="td">
            <%=l.getAutor() %>
        </span>
        <span class="td">
            <%=l.getNroEdicion() %>
        </span>
        <span class="td">
            <%=l.getFechaEdicion() %>
        </span>
        <span class="td">
            <%=l.getGenero().getDescripcion() %>
        </span>

        <span class="td"><button type="submit" name="action-type" value="borrar" class="btn btn-danger btn-block" >Quitar libro</button> </span>
    </form>
    <%
    }
    %>

</div>

</div>
<form action="FinalizarReservaServlet" method="POST" name="FinalizarReserva">
<%
if (listalibro.size() > 1) {
%>
<p>Tipo de reserva:</p>
<div>
<input type="radio" id="conjunta" name="tipo" value="conjunta" checked>
<label for="conjunta">Conjunta</label>
</div>
<div>
<input type="radio" id="individual" name="tipo" value="individual">
<label for="individual">Individual</label>
</div>
<div id="individual" class="desc card-deck" style="display: none">
<%
for (Libro l : listalibro) {
%>
<div class="card text-white bg-success mb-3" style="max-width: 18rem;">
<div class="card-body">
<h5 class="card-title"><%=l.getTitulo() %></h5>
<p class="card-text">Autor: <%=l.getAutor() %></p>
<p class="card-text">Edición: <%=l.getFechaEdicion() %></p>
</div>
<div class="card-footer">
<input name="fecha"<%=l.getId() %>" type="text" class="form-control" id="datepicker"<%=l.getId() %>" placeholder="Fecha de retiro">
</div>
</div>
<% %>
</div>
<%>
else {%>

<input type="hidden" id="conjunta" name="tipo" value="conjunta">
<% %>
<div id="conjunta" class="desc">
<input name="fecha" type="text" class="form-control" id="datepicker" placeholder="Fecha de retiro" >
</div>
<span><button type="submit" name="action-type" value="finalizar" class="btn btn-primary btn-block" >Finalizar Reservas</button> </span>
</form>
</body>
</html>

```

Capa de Lógica

LibroLogic

```

package logic;

import java.sql.SQLException;
import java.util.ArrayList;

import data.LibroDAO;
import model.Genero;
import model.Libro;

public class LibroLogic {

    private LibroDAO _LibroDAO;

    public LibroLogic() {
        this._LibroDAO = new LibroDAO();
    }

    public ArrayList<Libro> getAll() throws SQLException {
        try {
            return this._LibroDAO.getAll();
        }
        catch (SQLException exception) {
            throw exception;
        }
    }

    public Libro getOne(int id) throws SQLException {
        try {
            return this._LibroDAO.getOne(id);
        }
        catch (SQLException exception) {
            throw exception;
        }
    }

    public ArrayList<Libro> getAllByGenero(Genero genero) throws SQLException {
        try {
            return this._LibroDAO.getAllByGenero(genero);
        }
        catch (SQLException exception) {
            throw exception;
        }
    }

    public ArrayList<Libro> getAllByTitulo(String titulo) throws SQLException {
        try {
            return this._LibroDAO.getAllByTitulo(titulo);
        }
        catch (SQLException exception) {
            throw exception;
        }
    }

    public ArrayList<Libro> getAllByTituloAndGenero(String titulo, Genero genero) throws SQLException {
        try {
            return this._LibroDAO.getAllByTituloAndGenero(titulo, genero);
        }
        catch (SQLException exception) {
            throw exception;
        }
    }
}

```

```

    }

    public void insert(Libro lib) throws SQLException {
        try {
            this._LibroDAO.insert(lib);
        }
        catch (SQLException exception) {
            throw exception;
        }
    }

    public void update(Libro lib) throws SQLException {
        try {
            this._LibroDAO.update(lib);
        }
        catch (SQLException exception) {
            throw exception;
        }
    }

    public void delete(Libro lib) throws SQLException {
        try {
            this._LibroDAO.delete(lib);
        }
        catch (SQLException exception) {
            throw exception;
        }
    }
}

```

ReservaLogic

```
package logic;

import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.Date;

import data.ReservaDAO;
import model.Libro;
import model.LibroReserva;
import model.Reserva;

public class ReservaLogic {
    private ReservaDAO _ReservaDAO;

    public ReservaLogic() {
        this._ReservaDAO = new ReservaDAO();
    }

    public ArrayList<Reserva> getAll() throws SQLException {
        try {
            return this._ReservaDAO.getAll();
        }
        catch (SQLException exception) {
            throw exception;
        }
    }

    public ArrayList<Reserva> getAllPendientes() throws SQLException {
        try {
            return this._ReservaDAO.getAllPendientes();
        }
        catch (SQLException exception) {
            throw exception;
        }
    }

    public ArrayList<Reserva> getAllBySocio(int id) throws SQLException {
        try {
            return this._ReservaDAO.getAllBySocio(id);
        }
        catch (SQLException exception) {
            throw exception;
        }
    }

    public ArrayList<Reserva> getAllPendientesBySocio(int id) throws SQLException {
        try {
            return this._ReservaDAO.getAllPendientesBySocio(id);
        }
        catch (SQLException exception) {
            throw exception;
        }
    }

    public Reserva getOne(int id) throws SQLException {
        try {
            return this._ReservaDAO.getOne(id);
        }
    }
}
```

```

        catch (SQLException exception) {
            throw exception;
        }
    }

    public void delete(Reserva res) throws SQLException {
        try {
            this._ReservaDAO.delete(res);
        }
        catch (SQLException exception) {
            throw exception;
        }
    }

    public void entregarReserva(Reserva res) throws SQLException, BusinessException {
        try {
            SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMdd");
            Date date = new Date();
            if (sdf.format(res.getFechaReserva()).equals(sdf.format(date))) {
                this._ReservaDAO.entregarReserva(res);
            }
            else throw new BusinessException("Las reservas se retiran únicamente el día de la fecha pactada. Sin excepción");
        }
        catch (SQLException exception) {
            throw exception;
        }
    }

    public ArrayList<Reserva> getReservasFuturas(int idLibro) throws SQLException {
        try {
            return this._ReservaDAO.getReservasFuturas(idLibro);
        }
        catch (SQLException exception) {
            throw exception;
        }
    }

    public ArrayList<String> getFechasDisponibles(Libro lib, int cantMeses) throws SQLException {
        try {
            return this._ReservaDAO.getFechasDisponibles(lib, cantMeses);
        }
        catch (SQLException exception) {
            throw exception;
        }
    }

    public ArrayList<String> getFechasDisponibles(ArrayList<ArrayList<String>> fechasPorLibro, int cantMeses) {
        LocalDate date = LocalDate.now();
        ArrayList<String> fechas = new ArrayList<String>();
        ArrayList<String> fechasAux = new ArrayList<String>();
        while (date.isBefore(LocalDate.now().plusMonths(cantMeses))) {
            fechas.add("'" + date.toString() + "'");
            date = date.plusDays(1);
        }
        for (ArrayList<String> fechasLibro : fechasPorLibro) {
            for (String fecha : fechas) {
                System.out.print(fecha);
                if (!fechasLibro.contains(fecha)) {
                    fechasAux.add(fecha);
                    System.out.print("No disponible");
                }
            }
        }
    }

```

```

    }
    System.out.println("Fin");
}
fechas.removeAll(fechasAux);
return fechas;
}

public int getDiasMaximoPrestamo(Libro lib, int diasPoliticaPrestamo, Reserva reserva) throws SQLException {
    try {
        return this._ReservaDAO.getDiasMaximoPrestamo(lib, diasPoliticaPrestamo, reserva);
    }
    catch (SQLException exception) {
        throw exception;
    }
}

public int getDiasMaximoPrestamo(Reserva reserva) throws SQLException {
    int diasMaximoLibro;
    int diasMaximoPrestamo;
    int diasPoliticaPrestamo;
    try {
        PoliticaPrestamoLogic ppl = new PoliticaPrestamoLogic();
        diasPoliticaPrestamo = ppl.getActual().getDiasPrestamo();
        diasMaximoPrestamo = diasPoliticaPrestamo;
        for (LibroReserva lr : reserva.getLibros()) {
            diasMaximoLibro = this._ReservaDAO.getDiasMaximoPrestamo(lr.getLibro(), diasMaximoPrestamo, reserva);
            if (diasMaximoLibro < diasMaximoPrestamo) {
                diasMaximoPrestamo = diasMaximoLibro;
            }
        }
        return diasMaximoPrestamo;
    }
    catch (SQLException exception) {
        throw exception;
    }
}
}
}

```

Capa de Datos

DbConnector


```

public class DbConnector {

    private static DbConnector instancia;

    /*
    private static String driver = "com.mysql.jdbc.Driver";
    private static String url = "jdbc:mysql://localhost:3306/biblioteca";
    private static String user = "root";
    private static String password = "1111";
    private static String serverTimezone = "UTC";
    */

    private static String driver = "com.mysql.jdbc.Driver";
    private static String url = "jdbc:mysql://lyn7gfoxo996yjjco.cbetxkdyhwsb.us-east-1.rds.amazonaws.com:3306/r5wpcujxui2fbpn";
    private static String user = "h1xsm87yosp5n4sr";
    private static String password = "ca8ay3d3dl6ef4qm";
    private static String serverTimezone = "UTC";
    private int conectados=0;
    private Connection conn=null;

    private DbConnector() throws ClassNotFoundException {
        try {
            Class.forName(driver);
        } catch (ClassNotFoundException e) {
            throw e;
        }
    }

    public static DbConnector getInstancia() throws ClassNotFoundException {
        if (instancia == null) {
            instancia = new DbConnector();
        }
        return instancia;
    }

    public Connection getConn() throws SQLException {
        try {
            if(conn==null || conn.isClosed()) {
                Properties props = new Properties();
                props.put("user", user);
                props.put("password", password);
                props.put("serverTimezone", serverTimezone);
                conn = DriverManager.getConnection(url,props);
                conectados=0;
            }
        } catch (SQLException e) {
            throw e;
        }
        conectados++;
        return conn;
    }

    public void releaseConn() throws SQLException {
        conectados--;
        try {
            if (conectados<=0) {
                conn.close();
            }
        } catch (SQLException e) {
            throw e;
        }
    }
}

```

BaseDAO

```

package data;

import java.sql.Connection;

public abstract class BaseDAO {
    protected static Connection conn;

    protected void openConnection() throws SQLException {
        try {
            conn = DbConnector.getInstancia().getConn();
        }
        catch (SQLException e) {
            throw e;
        }
        catch (ClassNotFoundException e) {
            throw new SQLException("Error en la base de datos. No se encontró el driver");
        }
    }

    protected void closeConnection(PreparedStatement pst, ResultSet rs) {
        try {
            if(rs!=null) rs.close();
            if(pst!=null) pst.close();
            if(conn!=null) {
                conn = null;
                DbConnector.getInstancia().releaseConn();
            }
        }
        catch (SQLException e) {
            e.printStackTrace();
        }
        catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }

    protected void closeConnection(Statement stm, ResultSet rs) {
        try {
            if(rs!=null) rs.close();
            if(stm!=null) stm.close();
            if(conn!=null) conn.close();
        }
        catch (SQLException e) {
            e.printStackTrace();
        }
    }

    protected void closeConnection(PreparedStatement pst) {
        try {
            if(pst!=null) pst.close();
            if(conn!=null) conn.close();
        }
        catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

```

ReservaDAO

```

package data;

import java.sql.Date;

public class ReservaDAO extends BaseDAO implements IBaseDAO<Reserva> {

    public Reserva mapearReserva(ResultSet rs) throws SQLException {
        Reserva r = new Reserva();
        r.setId(rs.getInt("id_reserva"));
        r.setFechaReserva(rs.getDate("fecha_reserva"));
        SocioDAO sDAO = new SocioDAO();
        r.setSocio(sDAO.mapearSocio(rs));
        return r;
    }

    public void insert(Reserva res) throws SQLException {
        PreparedStatement pst = null;
        ResultSet rs = null;
        try {
            this.openConnection();
            pst = conn.prepareStatement("INSERT INTO reservas(fecha_reserva, "
                + "entregada, id_socio) VALUES(?,?,?)", Statement.RETURN_GENERATED_KEYS);
            pst.setDate(1, (Date)res.getFechaReserva());
            pst.setBoolean(2, false);
            pst.setInt(3, res.getSocio().getId());
            pst.executeUpdate();
            rs = pst.getGeneratedKeys();
            if (rs.next()) {
                res.setId(rs.getInt(1));
            }
            rs.close();
            for (LibroReserva l: res.getLibros()) {
                this.insertLibroReserva(l, res, pst);
            }
        } catch (SQLException e){
            throw e;
        } finally {
            this.closeConnection(pst, rs);
        }
    }

    public void update(Reserva res) throws SQLException {
        PreparedStatement pst = null;
        try {
            this.openConnection();
            pst = conn.prepareStatement("UPDATE reservas SET fecha_reserva = ?, "
                + "entregada = ?, id_socio = ? WHERE id_reserva = ?");
            pst.setDate(1, (Date)res.getFechaReserva());
            pst.setBoolean(2, res.isEntregada());
            pst.setInt(3, res.getSocio().getId());
            pst.setInt(4, res.getId());
            pst.executeUpdate();
            for (LibroReserva l: res.getLibros()) {
                if (l.getId() == 0) this.insertLibroReserva(l, res, pst);
                else if (l.getLibro() == null) this.deleteLibroReserva(l, pst);
                else this.updateLibroReserva(l, res, pst);
            }
        }
    }
}

```

```

public void delete(Reserva res) throws SQLException {
    PreparedStatement pst = null;
    try {
        this.openConnection();
        pst = conn.prepareStatement("DELETE FROM libro_reserva WHERE id_reserva = ?");
        pst.setInt(1, res.getId());
        pst.executeUpdate();
        pst.close();
        pst = conn.prepareStatement("DELETE FROM reservas WHERE id_reserva = ?");
        pst.setInt(1, res.getId());
        pst.executeUpdate();
    }
    catch (SQLException e){
        throw e;
    }
    finally {
        this.closeConnection(pst);
    }
}

public ArrayList<Reserva> getAll() throws SQLException {
    ArrayList<Reserva> reservas = new ArrayList<Reserva>();
    PreparedStatement pst = null;
    ResultSet rs = null;
    try {
        this.openConnection();
        pst = conn.prepareStatement("SELECT r.id_reserva, r.fecha_reserva, r.entregada, s.*, u.nombre_usuario, u.password, u.tipo, u.estado "
            + "FROM reservas r INNER JOIN socios s ON r.id_socio = s.id_socio "
            + "INNER JOIN usuarios u ON s.id_usuario = u.id_usuario");
        rs = pst.executeQuery();
        while (rs.next()) {
            reservas.add(this.mapearReserva(rs));
        }
        rs.close();
        pst.close();
        for (Reserva res: reservas) {
            res.setLibros(this.getAllLibroReserva(res, pst, rs));
        }
    }
    catch (SQLException e) {
        throw e;
    }
    finally {
        this.closeConnection(pst, rs);
    }
    return reservas;
}

```

```

public ArrayList<Reserva> getAllPendientes() throws SQLException {
    ArrayList<Reserva> reservas = new ArrayList<Reserva>();
    PreparedStatement pst = null;
    ResultSet rs = null;
    try {
        this.openConnection();
        pst = conn.prepareStatement("SELECT r.id_reserva, r.fecha_reserva, r.entregada, s.*, u.nombre_usuario, u.password, u.tipo, u.estado "
            + "FROM reservas r INNER JOIN socios s ON r.id_socio = s.id_socio "
            + "INNER JOIN usuarios u ON s.id_usuario = u.id_usuario "
            + "WHERE r.entregada = 0");
        rs = pst.executeQuery();
        while (rs.next()) {
            reservas.add(this.mapearReserva(rs));
        }
        rs.close();
        pst.close();
        for (Reserva res: reservas) {
            res.setLibros(this.getAllLibroReserva(res, pst, rs));
        }
    }
    catch (SQLException e) {
        throw e;
    }
    finally {
        this.closeConnection(pst, rs);
    }
    return reservas;
}

public ArrayList<Reserva> getAllBySocio(int id) throws SQLException {
    ArrayList<Reserva> reservas = new ArrayList<Reserva>();
    PreparedStatement pst = null;
    ResultSet rs = null;
    try {
        this.openConnection();
        pst = conn.prepareStatement("SELECT r.id_reserva, r.fecha_reserva, r.entregada, s.*, u.nombre_usuario, u.password, u.tipo, u.estado "
            + "FROM reservas r INNER JOIN socios s ON r.id_socio = s.id_socio "
            + "INNER JOIN usuarios u ON s.id_usuario = u.id_usuario "
            + "WHERE r.id_socio = ?");
        pst.setInt(1, id);
        rs=pst.executeQuery();
        while (rs.next()) {
            reservas.add(this.mapearReserva(rs));
        }
        rs.close();
        for (Reserva res: reservas) {
            res.setLibros(this.getAllLibroReserva(res, pst, rs));
        }
    }
    catch (SQLException e) {
        throw e;
    }
    finally {
        this.closeConnection(pst, rs);
    }
    return reservas;
}

```

```

public ArrayList<Reserva> getAllPendientesBySocio(int id) throws SQLException {
    ArrayList<Reserva> reservas = new ArrayList<Reserva>();
    PreparedStatement pst = null;
    ResultSet rs = null;
    try {
        this.openConnection();
        pst = conn.prepareStatement("SELECT r.id_reserva, r.fecha_reserva, r.entregada, s.*, u.nombre_usuario, u.password, u.tipo, u.estado "
            + "FROM reservas r INNER JOIN socios s ON r.id_socio = s.id_socio "
            + "INNER JOIN usuarios u ON s.id_usuario = u.id_usuario "
            + "WHERE r.id_socio = ? AND r.entregada = 0");
        pst.setInt(1, id);
        rs=pst.executeQuery();
        while (rs.next()) {
            reservas.add(this.mapearReserva(rs));
        }
        rs.close();
        for (Reserva res: reservas) {
            res.setlibros(this.getAllLibroReserva(res, pst, rs));
        }
    }
    catch (SQLException e) {
        throw e;
    }
    finally {
        this.closeConnection(pst, rs);
    }
    return reservas;
}

public Reserva getOne(int id) throws SQLException {
    Reserva res = null;
    PreparedStatement pst = null;
    ResultSet rs = null;
    try {
        this.openConnection();
        pst = conn.prepareStatement("SELECT r.id_reserva, r.fecha_reserva, r.entregada, s.*, u.nombre_usuario, u.password, u.tipo, u.estado "
            + "FROM reservas r INNER JOIN socios s ON r.id_socio = s.id_socio "
            + "INNER JOIN usuarios u ON s.id_usuario = u.id_usuario "
            + "WHERE id_reserva = ?");
        pst.setInt(1, id);
        rs = pst.executeQuery();
        if (rs.next()) {
            res = this.mapearReserva(rs);
            res.setlibros(this.getAllLibroReserva(res, pst, rs));
        }
        rs.close();
    }
    catch (SQLException e){
        throw e;
    }
    finally {
        this.closeConnection(pst, rs);
    }
    return res;
}

```

```

public ArrayList<Reserva> getReservasFuturas(int idLibro) throws SQLException {
    ArrayList<Reserva> reservas = new ArrayList<Reserva>();
    PreparedStatement pst = null;
    ResultSet rs = null;
    try {
        this.openConnection();
        pst = conn.prepareStatement("SELECT r.id_reserva, r.fecha_reserva, r.entregada, s.*, u.nombre_usuario, u.password, u.tipo, u.estado "
            + "FROM libro_reserva lr "
            + "INNER JOIN reservas r ON r.id_reserva= lr.id_reserva "
            + "INNER JOIN socios s ON r.id_socio = s.id_socio "
            + "INNER JOIN usuarios u ON s.id_usuario = u.id_usuario "
            + "WHERE lr.id_libro=? AND r.fecha_reserva >= curdate()");
        pst.setInt(1, idLibro);
        rs = pst.executeQuery();
        if (rs.next()) {
            reservas.add(this.mapearReserva(rs));
        }
        rs.close();
    }
    catch (SQLException e){
        throw e;
    }
    finally {
        this.closeConnection(pst, rs);
    }
    return reservas;
}

public void entregarReserva(Reserva res) throws SQLException {
    PreparedStatement pst = null;
    try {
        this.openConnection();
        pst = conn.prepareStatement("UPDATE reservas SET entregada = ? WHERE id_reserva = ?");
        pst.setBoolean(1, true);
        pst.setInt(2, res.getId());
        pst.executeUpdate();
    }
    catch (SQLException e){
        throw e;
    }
    finally {
        this.closeConnection(pst);
    }
}

```

```

public ArrayList<LibroReserva> getAllLibroReserva(Reserva res, PreparedStatement pst, ResultSet rs) throws SQLException {
    ArrayList<LibroReserva> librosRes = new ArrayList<LibroReserva>();
    if (conn == null || conn.isClosed()) this.openConnection();
    if (!pst.isClosed()) pst.close();
    if (!rs.isClosed()) rs.close();
    pst = conn.prepareStatement("SELECT lr.id_libro_reserva, lr.id_reserva,l.*,g.* FROM libro_reserva lr "
        + "INNER JOIN libros l ON lr.id_libro = l.id_libro "
        + "INNER JOIN generos g ON g.id_genero = l.id_genero "
        + "WHERE lr.id_reserva = ?");
    pst.setInt(1, res.getId());
    rs = pst.executeQuery();
    LibroDAO lDAO = new LibroDAO();
    while (rs.next()) {
        LibroReserva libRes = new LibroReserva();
        libRes.setId(rs.getInt("id_libro_reserva"));
        libRes.setLibro(lDAO.mapearLibro(rs));
        librosRes.add(libRes);
    }
    rs.close();
    pst.close();
    return librosRes;
}

public void insertLibroReserva(LibroReserva l, Reserva res, PreparedStatement pst) throws SQLException {
    if (conn == null || conn.isClosed()) this.openConnection();
    if (!pst.isClosed()) pst.close();
    pst = conn.prepareStatement("INSERT INTO libro_reserva(id_libro, "
        + "id_reserva) VALUES (?,?)");
    pst.setInt(1, l.getLibro().getId());
    pst.setInt(2, res.getId());
    pst.executeUpdate();
}

public void updateLibroReserva(LibroReserva libRes, Reserva res, PreparedStatement pst) throws SQLException {
    if (conn == null || conn.isClosed()) this.openConnection();
    if (!pst.isClosed()) pst.close();
    pst = conn.prepareStatement("UPDATE libro_reserva SET id_libro = ?, "
        + "id_reserva = ? WHERE id_libro_reserva = ?");
    pst.setInt(1, libRes.getLibro().getId());
    pst.setInt(2, res.getId());
    pst.setInt(3, libRes.getId());
    pst.executeUpdate();
}

public void deleteLibroReserva(LibroReserva libRes, PreparedStatement pst) throws SQLException {
    if (conn == null || conn.isClosed()) this.openConnection();
    if (!pst.isClosed()) pst.close();
    pst = conn.prepareStatement("DELETE FROM libro_reserva "
        + "WHERE id_libro_reserva = ?");
    pst.setInt(1, libRes.getId());
    pst.executeUpdate();
}

```



```

public ArrayList<String> getFechasDisponibles(Libro lib, int cantMeses) throws SQLException {
    LocalDate date = LocalDate.now();
    ArrayList<String> fechas = new ArrayList<String>();
    PreparedStatement pst = null;
    ResultSet rs = null;
    int ejemplaresDisponibles = 0;
    try {
        this.openConnection();
        while (date.isBefore(LocalDate.now().plusMonths(cantMeses))) {

            pst = conn.prepareStatement("SELECT e.id_ejemplar FROM prestamos p "
                + "INNER JOIN lineasdeprestamo lp ON p.id_prestamo = lp.id_prestamo "
                + "RIGHT JOIN ejemplares e ON lp.id_ejemplar = e.id_ejemplar "
                + "WHERE e.id_libro = ? "
                + "GROUP BY e.id_ejemplar "
                + "HAVING (sum(if(((date_add(ifnull(p.fecha_prestamo, now()), INTERVAL ifnull(p.dias_prestamo, 0) DAY) >= ?) OR "
                + "[date_add(ifnull(p.fecha_prestamo, now()), INTERVAL ifnull(p.dias_prestamo, 0) DAY) < now() ])) AND (ifnull(lp.devuelto, 1)=0), 1, 0))= 0)");

            pst.setInt(1, lib.getId());
            pst.setObject(2, date);
            rs=pst.executeQuery();
            while (rs.next())
            {
                ejemplaresDisponibles += 1;
            }
            rs.close();
            pst.close();
            pst = conn.prepareStatement("SELECT count(lr.id_libro) AS cantreservas FROM reservas r "
                + "INNER JOIN libro_reserva lr ON lr.id_reserva = r.id_reserva "
                + "WHERE r.entregada = FALSE AND lr.id_libro = ? AND ? BETWEEN r.fecha_reserva "
                + "AND date_add(r.fecha_reserva, INTERVAL (select cant_dias_prestamo from politicaprestamo where fecha_politica_prestamo in "
                + "( select max(fecha_politica_prestamo) from politicaprestamo where fecha_politica_prestamo <= ?)) DAY) ");

            pst.setInt(1, lib.getId());
            pst.setObject(2, date);
            pst.setObject(3, date);
            rs=pst.executeQuery();
            if (rs.next())
                ejemplaresDisponibles -= rs.getInt("cantreservas");
            if (ejemplaresDisponibles > 0) {
                fechas.add(date.toString()+"");
            }
            ejemplaresDisponibles = 0;
            date=date.plusDays(1);
        }
    }
    catch (SQLException e) {
        throw e;
    }
    finally {
        this.closeConnection(pst, rs);
    }
    return fechas;
}

```

```

public int getDiasMaximoPrestamo(Libro lib, int diasPoliticaPrestamo, Reserva reserva) throws SQLException {
    LocalDate date = LocalDate.now().plusDays(1);
    PreparedStatement pst = null;
    ResultSet rs = null;
    int ejemplaresDisponibles = 0;
    int dias = 0;
    try {
        this.openConnection();
        while (date.isBefore(LocalDate.now().plusDays(diasPoliticaPrestamo+1))) {
            pst = conn.prepareStatement("SELECT e.id_ejemplar FROM ejemplares e WHERE e.id_libro = ? "
                + "AND NOT EXISTS ( SELECT * FROM lineasdeprestamo lp "
                + "INNER JOIN prestamos p ON p.id_prestamo = lp.id_prestamo "
                + "WHERE lp.id_ejemplar = e.id_ejemplar AND lp.devuelto = 0 AND ("
                + "date_add(p.fecha_prestamo, INTERVAL p.dias_prestamo DAY) >= ? "
                + "OR date_add(p.fecha_prestamo, INTERVAL p.dias_prestamo DAY) < now())");
            pst.setInt(1, lib.getId());
            pst.setDate(2, date);
            rs=pst.executeQuery();
            while (rs.next())
            {
                ejemplaresDisponibles += 1;
            }
            rs.close();
            pst.close();
            pst = conn.prepareStatement("SELECT count(lr.id_libro) AS cantreservas FROM reservas r "
                + "INNER JOIN libro_reserva lr ON lr.id_reserva = r.id_reserva "
                + "WHERE r.id_reserva <> ? AND r.entregada = FALSE AND r.fecha_reserva >= ? AND lr.id_libro = ? AND ? "
                + "BETWEEN r.fecha_reserva AND date_add(r.fecha_reserva, INTERVAL (select cant_dias_prestamo from politicaprestamo where fecha_politica_prestamo in"
                + " ( select max(fecha_politica_prestamo) from politicaprestamo where fecha_politica_prestamo <= ?)) DAY) ");
            pst.setInt(1, reserva.getId());
            pst.setDate(2, (Date)reserva.getFechaReserva());
            pst.setInt(3, lib.getId());
            pst.setObject(4, date);
            pst.setObject(5, date);
            rs=pst.executeQuery();
            if (rs.next())
                ejemplaresDisponibles -= rs.getInt("cantreservas");
            if (ejemplaresDisponibles > 0) {
                dias++;
            } else break;
            ejemplaresDisponibles = 0;
            date=date.plusDays(1);
        }
    }
    catch (SQLException e) {
        throw e;
    }
    finally {
        this.closeConnection(pst, rs);
    }
    return dias;
}
}

```

Model
Libro

```

package model;

import java.time.LocalDate;

public class Libro extends Entidad {
    private String _autor;
    private String _titulo;
    private String _nroEdicion;
    private LocalDate _fechaEdicion;
    private Genero _genero;
    private int _cantidadEjemplares;

    public String getAutor() {
        return _autor;
    }
    public void setAutor(String autor) {
        this._autor = autor;
    }
    public String getTitulo() {
        return _titulo;
    }
    public void setTitulo(String titulo) {
        this._titulo = titulo;
    }
    public String getNroEdicion() {
        return _nroEdicion;
    }
    public void setNroEdicion(String nroEdicion) {
        this._nroEdicion = nroEdicion;
    }
    public LocalDate getFechaEdicion() {
        return _fechaEdicion;
    }
    public void setFechaEdicion(LocalDate fechaEdicion) {
        this._fechaEdicion = fechaEdicion;
    }
    public Genero getGenero() {
        return _genero;
    }
    public void setGenero(Genero genero) {
        this._genero = genero;
    }
    public int getCantEjemplares() {
        return _cantidadEjemplares;
    }
    public void setCantEjemplares(int cantidadEjemplares) {
        this._cantidadEjemplares = cantidadEjemplares;
    }
}

```

Reserva

```

package model;

import java.util.ArrayList;

public class Reserva extends Entidad {
    private Date _fechaReserva;
    private boolean _entregada;
    private Socio _socio;
    private ArrayList <LibroReserva> _libros;

    public Date getFechaReserva() {
        return _fechaReserva;
    }
    public void setFechaReserva(Date fechaReserva) {
        this._fechaReserva = fechaReserva;
    }
    public boolean isEntregada() {
        return _entregada;
    }
    public void setEntregada(boolean entregada) {
        this._entregada = entregada;
    }
    public Socio getSocio() {
        return _socio;
    }
    public void setSocio(Socio socio) {
        this._socio = socio;
    }
    public ArrayList <LibroReserva> getLibros() {
        return _libros;
    }
    public void setLibros(ArrayList <LibroReserva> libros) {
        this._libros = libros;
    }
}

```