PRUEBA II

BIBLIOTECA

Luis Díaz del Valle

DESCRIPCION:

La aplicación realiza las 6 funciones pedidas a realizar por la aplicación; insertar título, insertar socio, prestar libro, devolver libro, histórico libro, histórico socio.

```
public class Main {
    public static void main(String[] args) throws Exception
    {
        SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");

        Scanner sn = new Scanner(System.in);
        boolean exit = false;
        int opc;
        try ( Dao dao = new OracleDao() )
        {
            while ( !exit )
            {
                System.out.println("---BIBLIOTECA---");
                System.out.println("\t 1. Insertar Libro");
                System.out.println("\t 2. Insertar Socio");
                System.out.println("\t 3. Prestar Libro");
                System.out.println("\t 4. Devolver Libro");
                System.out.println("\t 5. Historico Libro");
                System.out.println("\t 6. Historico Socio");
                System.out.println("\t 7. Salir");

                System.out.println("\t Escribe una de las opciones: ");
                opc = sn.nextInt();
```

```
switch ( opc )
{
    case 1:
        System.out.println("\t Introduce el ISBN del libro");
        String isbn = sn.next();
        System.out.println("\t Introduce el Titulo del libro");
        String titulo = sn.next();
        dao.insertarLibro(isbn, titulo);
        break;
    case 2:
        System.out.println("\t Introduce el DNI del socio");
        String dni = sn.next();
        System.out.println("\t Introduce el Nombre del socio");
        String nombre = sn.next();
        dao.insertarSocio(dni, nombre);
        break;
    case 3:
        System.out.println("\t Introduce el DNI del socio");
        String dniP = sn.next();
        System.out.println("\t Introduce el ISBN del libro");
        String isbnP = sn.next();
        System.out.println("\t Introduce la fecha del prestamo");
        Date fechaPrestamo = sdf.parse(sn.next());
        dao.prestarLibro(isbnP, dniP, fechaPrestamo);
        break;
    case 4:
        System.out.println("\t Introduce el ISBN del libro");
        String isbnD = sn.next();
        System.out.println("\t Introduce la fecha de devolucion");
        Date fechaDevolucion = sdf.parse(sn.next());
        dao.devolverLibro(isbnD, fechaDevolucion);
        break;
    case 5:
        System.out.println("\t Introduce el ISBN del libro");
        String isbnH = sn.next();
        System.out.println(dao.historicoLibro(isbnH));
        //List<Historico> historicos = dao.historicoLibro(isbnH);
        //for(Historico historico: historicos)
        break;
    case 6:
        System.out.println("\t Introduce el Codigo del socio");
        String codigoS = sn.next();
        System.out.println(dao.historicoSocio(codigoS));
        break;
    case 7:
        exit = true;
        break;
    default:
        System.out.println("Solo numeros entre el 1 y 7");
```

TABLAS

```
CREATE TABLE LIBROS
(
    ISBN                VARCHAR2(12),
    TITULO              VARCHAR2(100),
    ESTADO              VARCHAR2(7),
    LOCALIZADOR         VARCHAR2(8),
    CONSTRAINT          "PK_LIBROS"                     PRIMARY KEY (ISBN),
    CONSTRAINT          "UK_LIBROS.LOCALIZADOR"         UNIQUE (LOCALIZADOR),
    CONSTRAINT          "NN_LIBROS.TITULO"              CHECK (TITULO IS NOT NULL),
    CONSTRAINT          "NN_LIBROS.ESTADO"              CHECK (ESTADO IS NOT NULL),
    CONSTRAINT          "CH_LIBROS.ESTADO"              CHECK (ESTADO IN ('LIBRE', 'OCUPADO'))
)


CREATE TABLE SOCIOS
(
    CODIGO              VARCHAR2(6),
    DNI                 VARCHAR2(12),
    NOMBRE              VARCHAR2(100),
    CONSTRAINT          "PK_SOCIOS"                     PRIMARY KEY (CODIGO),
    CONSTRAINT          "UK_SOCIOS"                     UNIQUE (DNI),
    CONSTRAINT          "NN_SOCIOS.DNI"                 CHECK (DNI IS NOT NULL),
    CONSTRAINT          "NN_SOCIOS.NOMBRE"              CHECK (NOMBRE IS NOT NULL)
)

CREATE TABLE PRESTAMOS
(
    LOCALIZADOR       VARCHAR2(8),
    ISBN              VARCHAR2(12),
    DNI               VARCHAR2(12),
    FECHA_PRESTAMO    DATE,
    FECHA_DEVOLUCION  DATE,
    CONSTRAINT        "PK_PRESTAMOS"                PRIMARY KEY (LOCALIZADOR),
    CONSTRAINT        "FK_PRESTAMOS.ISBN"           FOREIGN KEY (ISBN)
                                                    REFERENCES LIBROS(ISBN),
    CONSTRAINT        "FK_PRESTAMOS.DNI"            FOREIGN KEY (DNI)
                                                    REFERENCES SOCIOS(DNI),
    CONSTRAINT        "NN_PRESTAMOS.ISBN"           CHECK (ISBN IS NOT NULL),
    CONSTRAINT        "NN_PRESTAMOS.DNI"            CHECK (DNI IS NOT NULL),
    CONSTRAINT        "NN_PRESTAMOS.FECHA_PRESTAMO" CHECK (FECHA_PRESTAMO IS NOT NULL)
)
```

```
ALTER TABLE LIBROS
 ADD CONSTRAINT "FK_LIBROS.LOCALIZADOR" FOREIGN KEY (LOCALIZADOR)
                                        REFERENCES PRESTAMOS(LOCALIZADOR)
ALTER TABLE PRESTAMOS
    ADD CONSTRAINT "DF_PRESTAMOS.FECHA_DEVOLUCION"
    DEFAULT ('PRESTADO') FOR FECHA_DEVOLUCION

CREATE SEQUENCE SEQ_SOCIOS
    MINVALUE 1
    MAXVALUE 999998
    START WITH 1
    INCREMENT BY 1

CREATE SEQUENCE SEQ_PRESTAMOS
    MINVALUE 1
    MAXVALUE 99999998
    START WITH 1
    INCREMENT BY 1
```

A parte de los CONSTRAINT CHECK IS NOT NULL de los campos que no son UNIQUE O PRIMARY KEY se han añadido:

En la tabla LIBROS se ha dado al Localizador UNIQUE dado que no puede haber dos iguales y CHECK IN 'LIBRE', 'OCUPADO' para el enum Estado. No se le ha dado al Localizador la comprobación IS NOT NULL porque hasta que no se haga un préstamo se inicializara en null.

En la tabla SOCIOS el código representa como PRIMARY KEY y el dni como UNIQUE dado que no puede haber dos PRIMARY KEY en una misma tabla.

En la tabla PRESTAMOS se añade el isbn de la tabla libros y el dni de la tabla socios, para asegurarse de que existen en sus respectivas tablas y de que si se elimina ese libro o socio también se elimina el préstamo se usa el CONSTRAINT FOREIGN KEY (ISBN) REFERENCES LIBROS(ISBN).

Se comprueban todos los campos si son null menos la fecha de préstamo que se inicializa en null hasta que se devuelve el préstamo.

Después de añadir las tablas se altera la tabla LIBROS para hacer el FOREIGN KEY al localizador ahora que se ha añadido la tabla PRESTAMOS, para eso usamos.

```
ALTER TABLE LIBROS
 ADD CONSTRAINT "FK_LIBROS.LOCALIZADOR"    FOREIGN KEY (LOCALIZADOR)
                                           REFERENCES PRESTAMOS(LOCALIZADOR)
```

También he intentado añadir un CONSTRAINT para añadir un DEFAULT que cambie el null por defecto, pero me ha dado error.

```
ALTER  TABLE  PRESTAMOS
   ADD CONSTRAINT  "DF_PRESTAMOS.FECHA_DEVOLUCION"
   DEFAULT  ('PRESTADO')  FOR  FECHA_DEVOLUCION
```

Son necesarias también las sequence para generar de forma automática el código y el localizador.

```
CREATE SEQUENCE SEQ_SOCIOS
        MINVALUE 1
        MAXVALUE 999998
        START WITH 1
        INCREMENT BY 1

CREATE SEQUENCE SEQ_PRESTAMOS
        MINVALUE 1
        MAXVALUE 99999998
        START WITH 1
        INCREMENT BY 1
```

SENTENCIAS SQL

- Insertar Libro:

```
INSERT INTO LIBROS(ISBN, TITULO, ESTADO, LOCALIZADOR)
                VALUES(?,      ?,          ?,            ?                    )
```

- Insertar Socio:

```
INSERT INTO SOCIOS(CODIGO,                    DNI, NOMBRE)
                 VALUES(SEQ_SOCIOS.NEXTVAL,   ?,      ?          )
```

- Prestar Libro:

  Primero compruebo que el isnb y el dni introducidos existen

```
SELECT L.ISBN, S.DNI
        FROM LIBROS L, SOCIOS S
        WHERE L.ISBN = ? AND S.DNI = ?"
```

  Inserto en Prestamos

```
INSERT INTO PRESTAMOS(LOCALIZADOR, ISBN, DNI, FECHA_PRESTAMO,
FECHA_DEVOLUCION)

VALUES(SEQ_PRESTAMOS.NEXTVAL,    ?,    ?,   ?,            ?             )
```

  Actualizo el libro prestado cambiando el estado y añadiéndole su respectivo localizador

```
UPDATE LIBROS

SET ESTADO = 'OCUPADO',  LOCALIZADOR = ?

WHERE ISBN = ?
```

- Devolver Libro:

  Compruebo que el isbn existe tanto la tabla libros como en la tabla prestamos y que el estado esta ocupado.

```
SELECT P.ISBN
FROM LIBROS L, PRESTAMOS P
WHERE P.ISBN = ?  AND  L.ESTADO = 'OCUPADO'
        AND  P.LOCALIZADOR = L.LOCALIZADOR
```

  Actualizo la fecha de préstamo de la tabla prestamos

```
UPDATE PRESTAMOS

SET FECHA_DEVOLUCION = ?

WHERE ISBN = ?
```

  Actualizo los libros devueltos volviendo el estado a libre y el localizador a null

```
UPDATE LIBROS

SET ESTADO = 'LIBRE', LOCALIZADOR = ?

WHERE ISBN = ?
```

- Histórico Libro:
  Añado a una lista el objeto Histórico compuesto del código y nombre del socio y las fechas del préstamo pasandole el isbn y comprobando que existe en las dos tablas diciendo que sus dos dni coinciden. Luego ordeno la lista por fecha de préstamo.
  SELECT S.CODIGO, S.NOMBRE, P.FECHA_PRESTAMO, P.FECHA_DEVOLUCION "
  FROM SOCIOS S, PRESTAMOS
  WHERE P.ISBN = ? AND P.DNI = S.DNI
  ORDER BY FECHA_PRESTAMO
- Histórico Socio:

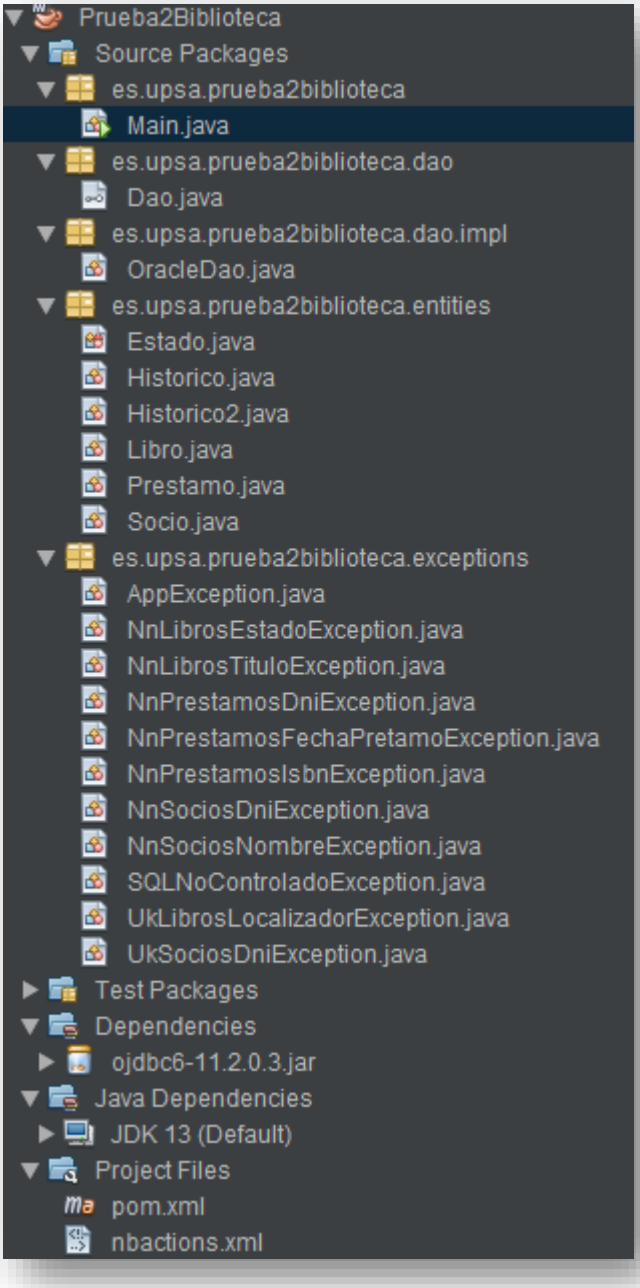
  Añado a una lista el objeto Histórico2 compuesto por el isbn y el titulo del libro y las fechas del préstamo. Se le pasa como filtro el código asique compruebo que tanto los dni de prestamos y socios como los isbn de libros y prestamos coinciden. Se ordena también por fecha de préstamo.

  SELECT L.ISBN, L.TITULO, P.FECHA_PRESTAMO, P.FECHA_DEVOLUCION

  FROM SOCIOS S, PRESTAMOS P, LIBROS L

  WHERE S.CODIGO = ? AND P.DNI = S.DNI AND P.ISBN = L.ISBN

  ORDER BY FECHA_PRESTAMO


Prueba2Biblioteca
Source Packages
es.upsa.prueba2biblioteca
Main.java
es.upsa.prueba2biblioteca.dao
Dao.java
es.upsa.prueba2biblioteca.dao.impl
OracleDao.java
es.upsa.prueba2biblioteca.entities
Estado.java
Historico.java
Historico2.java
Libro.java
Prestamo.java
Socio.java
es.upsa.prueba2biblioteca.exceptions
AppException.java
NnLibrosEstadoException.java
NnLibrosTituloException.java
NnPrestamosDniException.java
NnPrestamosFechaPretamoException.java
NnPrestamosIsbnException.java
NnSociosDniException.java
NnSociosNombreException.java
SQLNoControladoException.java
UkLibrosLocalizadorException.java
UkSociosDniException.java
Test Packages
Dependencies
ojdbc6-11.2.0.3.jar
Java Dependencies
JDK 13 (Default)
Project Files
pom.xml
nbactions.xml

```
3
4   SELECT * FROM LIBROS
5
6
7
```

SELECT * FROM LIBROS ×

Max. rows: 100 Fetched Rows: 3

#	ISBN	TITULO	ESTADO	LOCALIZADOR
1	1-1234-43210	HarryPotter	LIBRE	<NULL>
2	0-1234-43210	Africanus	OCUPADO	141
3	0-4321-01234	Mistborn	OCUPADO	142

```
4   SELECT * FROM SOCIOS
5
6
7
```

SELECT * FROM SOCIOS ×

Max. rows: 100 Fetched Rows:

#	CODIGO	DNI	NOMBRE
1	4	1234567-N	L.Diaz
2	5	7654321-J	A.Fernandez
3	43	1325647-W	J.Calleja
4	41	1237654-D	J.Garcia
5	42	3214567-K	P.Roca

```
4   SELECT * FROM PRESTAMOS
5
6
7
```

SELECT * FROM PRESTAMOS ×

Max. rows: 100 Fetched Rows: 9

#	LOCALIZADOR	ISBN	DNI	FECHA_PRESTAMO	FECHA_DEVOLUCION
1	105	1-1234-43210	1234567-N	2020-01-01 00:00:00.0...	2020-02-01 00:00:00.000
2	104	0-1234-43210	7654321-J	2019-12-30 00:00:00.0...	2020-01-30 00:00:00.000
3	103	0-4321-01234	1234567-N	2019-12-25 00:00:00.0...	2020-01-22 00:00:00.000
4	121	0-4321-01234	7654321-J	2019-11-30 00:00:00.0...	<NULL>
5	122	0-4321-01234	1325647-W	2019-12-06 00:00:00.0...	<NULL>
6	123	0-4321-01234	1237654-D	2020-01-19 00:00:00.0...	<NULL>
7	124	0-4321-01234	3214567-K	2020-01-15 00:00:00.0...	<NULL>
8	141	0-1234-43210	1234567-N	2019-11-03 00:00:00.0...	<NULL>
9	142	0-4321-01234	1234567-N	2019-06-06 00:00:00.0...	<NULL>

```
---BIBLIOTECA---
        1. Insertar Libro
        2. Insertar Socio
        3. Prestar Libro
        4. Devolver Libro
        5. Historico Libro
        6. Historico Socio
        7. Salir
        Escribe una de las opciones:
5
        Introduce el ISBN del libro
0-4321-01234
[Historico Libro {codigo=4, nombre=L.Diaz, fecha_prestamo=2019-06-06, fecha_devolucion=null}
, Historico Libro {codigo=5, nombre=A.Fernandez, fecha_prestamo=2019-11-30, fecha_devolucion=null}
, Historico Libro {codigo=43, nombre=J.Calleja, fecha_prestamo=2019-12-06, fecha_devolucion=null}
, Historico Libro {codigo=4, nombre=L.Diaz, fecha_prestamo=2019-12-25, fecha_devolucion=2020-01-22}
, Historico Libro {codigo=42, nombre=P.Roca, fecha_prestamo=2020-01-15, fecha_devolucion=null}
, Historico Libro {codigo=41, nombre=J.Garcia, fecha_prestamo=2020-01-19, fecha_devolucion=null}
```

```
---BIBLIOTECA---
        1. Insertar Libro
        2. Insertar Socio
        3. Prestar Libro
        4. Devolver Libro
        5. Historico Libro
        6. Historico Socio
        7. Salir
        Escribe una de las opciones:
6
        Introduce el Codigo del socio
5
[Historico Socio {isbn=0-4321-01234, titulo=Mistborn, fecha_prestamo=2019-11-30, fecha_devolucion=null}
, Historico Socio {isbn=0-1234-43210, titulo=Africanus, fecha_prestamo=2019-12-30, fecha_devolucion=2020-01-30}
]
```

EXCEPTIONS

```java
public class UkLibrosLocalizadorException extends AppException
{
    private String localizador;

    public UkLibrosLocalizadorException(String localizador) {
        super("Hay otro libro con el localizador " + localizador);
        this.localizador = localizador;
    }

    public String getLocalizador() {
        return localizador;
    }
}
```

```java
public class NnLibrosTituloException extends AppException
{
    public NnLibrosTituloException() {
        super("Titulo vacio");
    }
}
```

DAO

```java
public interface Dao extends AutoCloseable
{
    Libro insertarLibro(String isbn, String titulo) throws AppException;
    Socio insertarSocio(String dni, String nombre) throws AppException;
    Optional<Prestamo> prestarLibro(String isbn, String dni, Date fecha_prestamo) throws AppException;
    Prestamo insertarPrestamo(String isbn, String dni, Date fecha_prestamo) throws AppException;
    Libro actualizarLibroPrestamo(String isbn, String localizador) throws AppException;
    Optional<Prestamo> devolverLibro(String isbn, Date fecha_devolucion) throws AppException;
    Prestamo actualizarPrestamoDevuelto(String isbn, Date fecha_devolucion) throws AppException;
    Libro actualizarLibroDevuelto(String isbn) throws AppException;
    List<Historico> historicoLibro(String isbn) throws AppException;
    List<Historico2> historicoSocio(String codigo) throws AppException;
}
```

ORACLEDAO

```java
public class OracleDao implements Dao
{
    private Connection connection;

    public OracleDao() throws AppException
    {
        try
        {
            Driver driver = new OracleDriver();
            DriverManager.registerDriver( driver );
            this.connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE",
                                                          "c##luisdiaz",
                                                          "1995");

        } catch (SQLException sqlException)
          {
              throw new SQLNoControladoException(sqlException);
          }
    }
```

```java
//inserta un nuevo libro con estado LIBRE y localizador NULL
@Override
public Libro insertarLibro(String isbn, String titulo) throws AppException
{
    final String SQL = "INSERT INTO LIBROS(ISBN, TITULO, ESTADO, LOCALIZADOR)" +
                       "            VALUES(?,    ?,      ?,      ?           )" ;

    String[] fields = new String[] { "ISBN" };
    String localizador = null;
    Estado estado = Estado.LIBRE;
    try ( PreparedStatement ps = connection.prepareStatement(SQL, fields) )
    {
    ps.setString(1, isbn);
    ps.setString(2, titulo);
    ps.setString(3, estado.name());
    ps.setString(4, localizador);

    ps.executeUpdate();

    return new Libro(isbn, titulo, estado, localizador);


    }catch (SQLException sqlException)
        {
            String message = sqlException.getMessage();
            if      ( message.contains("UK_LIBROS.LOCALIZADOR") ) throw new UkLibrosLocalizadorException(localizador);
            else if ( message.contains("NN_LIBROS.TITULO") )      throw new NnLibrosTituloException();
            else if ( message.contains("NN_LIBROS.ESTADO") )      throw new NnLibrosEstadoException();

            throw new SQLNoControladoException(sqlException);
        }
}
```

```java
//inserta un nuevo socio
@Override
public Socio insertarSocio(String dni, String nombre) throws AppException
{
    final String SQL = "INSERT INTO SOCIOS(CODIGO,                DNI, NOMBRE)" +
                       "            VALUES(SEQ_SOCIOS.NEXTVAL,    ?,   ?     )";

    String[] fields = new String[] { "CODIGO" };

    try ( PreparedStatement ps = connection.prepareStatement(SQL, fields) )
    {
        ps.setString(1, dni);
        ps.setString(2, nombre);
        ps.executeUpdate();
        try( ResultSet rsKeys = ps.getGeneratedKeys() )
        {
        rsKeys.next();
        return new Socio(rsKeys.getString(1), dni, nombre);
        }

    } catch (SQLException sqlException)
        {
            String message = sqlException.getMessage();
            if      ( message.contains("UK_SOCIOS.DNI") )       throw new UkSociosDniException(dni);
            else if ( message.contains("NN_SOCIOS.DNI") )       throw new NnSociosDniException();
            else if ( message.contains("NN_SOCIOS.NOMBRE") )    throw new NnSociosNombreException();
            throw new SQLNoControladoException(sqlException);
        }
}
```

```java
//actualiza un libro prestado cambiando el estado a OCUPADO y el localizador al del prestamo
@Override
public Libro actualizarLibroPrestamo(String isbn, String localizador) throws AppException
{
    final String SQL =  "UPDATE LIBROS                          " +
                        "SET ESTADO = 'OCUPADO', LOCALIZADOR = ? " +
                        "WHERE ISBN = ?                         " ;


    try ( PreparedStatement ps = connection.prepareStatement(SQL) )
    {
        ps.setString(1, localizador);
        ps.setString(2, isbn);

        ps.executeUpdate();

        return null;

    }catch (SQLException sqlException)
        {
            String message = sqlException.getMessage();
            if      ( message.contains("UK_LIBROS.LOCALIZADOR") ) throw new UkLibrosLocalizadorException(localizador);
            else if ( message.contains("NN_LIBROS.ESTADO") )      throw new NnLibrosEstadoException();

            throw new SQLNoControladoException(sqlException);
        }

}
```

```java
//registra el prestamo con la fecha de devolucion = NULL
//llama a la funcion anterior para actualizar el libro prestado
@Override
public Prestamo insertarPrestamo(String isbn, String dni, Date fecha_prestamo) throws AppException
{
    final String SQL = "INSERT INTO PRESTAMOS(LOCALIZADOR,              ISBN, DNI, FECHA_PRESTAMO, FECHA_DEVOLUCION)" +
                       "              VALUES(SEQ_PRESTAMOS.NEXTVAL,     ?,    ?,   ?,              ?               )" ;

    String[] fields = new String[] { "LOCALIZADOR" };
    String localizador = null;
    java.sql.Date fecha_devolucion = null;
    try ( PreparedStatement ps = connection.prepareStatement(SQL, fields) )
    {
        ps.setString(1, isbn);
        ps.setString(2, dni);
        java.sql.Date sqlDate = new java.sql.Date( fecha_prestamo.getTime() );
        ps.setDate(3, sqlDate);
        ps.setDate(4, fecha_devolucion);;
        ps.executeUpdate();

        try ( ResultSet rsKeys = ps.getGeneratedKeys() )
        {
            rsKeys.next();
            localizador = rsKeys.getString(1);
        }
        actualizarLibroPrestamo(isbn, localizador);
        Prestamo prestamo = new Prestamo();
        prestamo.setLocalizador(localizador);
        prestamo.setIsbn(isbn);
        prestamo.setDni(dni);
        prestamo.setFecha_prestamo(fecha_prestamo);
        prestamo.setFecha_devolucion(fecha_devolucion);
        return prestamo;


    } catch (SQLException sqlException)
        {
            String message = sqlException.getMessage();
            if ( message.contains("NN_PRESTAMOS.ISBN") )           throw new NnPrestamosIsbnException();
            else if ( message.contains("NN_PRESTAMOS.DNI") )            throw new NnPrestamosDniException();
            else if ( message.contains("NN_PRESTAMOS.FECHA_PRESTAMO") ) throw new NnPrestamosFechaPretamoException();
```

```java
//comprueba que el dni y el isbn existen para registrar un prestamo
//llama a la funcion anterior para registrar el prestamo
@Override
public Optional<Prestamo> prestarLibro(String isbn, String dni, Date fecha_prestamo) throws AppException
{
    final String SQL = "SELECT L.ISBN, S.DNI           " +
                       "FROM LIBROS L, SOCIOS S        " +
                       "WHERE L.ISBN = ? AND S.DNI = ?" ;
    try ( PreparedStatement ps = connection.prepareStatement(SQL) )
    {
        ps.setString(1, isbn);
        ps.setString(2, dni);
        try ( ResultSet rs = ps.executeQuery() )
        {
            if ( rs.next() )
            {
                return Optional.of(insertarPrestamo(isbn, dni, fecha_prestamo));
            }
            else
            {
                return Optional.empty();
            }
        }

    } catch (SQLException sqlException)
        {
            throw new SQLNoControladoException(sqlException);
        }

}
```

```java
//actualiza el estado a libre y el localizador a null de un libro devuelto
@Override
public Libro actualizarLibroDevuelto(String isbn) throws AppException
{
    final String SQL =  "UPDATE LIBROS                              " +
                        "SET ESTADO = 'LIBRE', LOCALIZADOR = ? " +
                        "WHERE ISBN = ?                             " ;


    try ( PreparedStatement ps = connection.prepareStatement(SQL) )
    {
        String localizador = null;
        ps.setString(1, localizador);
        ps.setString(2, isbn);

        ps.executeUpdate();

        Libro libro = new Libro();
        libro.setEstado(Estado.LIBRE);
        libro.setLocalizador(localizador);
        return libro;

    }catch (SQLException sqlException)
        {
            String message = sqlException.getMessage();
            if ( message.contains("NN_LIBROS.ESTADO") )      throw new NnLibrosEstadoException();

            throw new SQLNoControladoException(sqlException);
        }

}
```

```java
//actualiza la fecha de devolucion de un prestamo devuelto
@Override
public Prestamo actualizarPrestamoDevuelto(String isbn, Date fecha_devolucion) throws AppException
{
    final String SQL =  "UPDATE PRESTAMOS           " +
                        "SET FECHA_DEVOLUCION = ?   " +
                        "WHERE ISBN = ?             " ;


    try ( PreparedStatement ps = connection.prepareStatement(SQL) )
    {
        java.sql.Date sqlDate = new java.sql.Date( fecha_devolucion.getTime() );
        ps.setDate(1, sqlDate);
        ps.setString(2, isbn);

        ps.executeUpdate();

        Prestamo prestamo = new Prestamo();
        prestamo.setFecha_devolucion(fecha_devolucion);
        return prestamo;

    }catch (SQLException sqlException)
        {
            throw new SQLNoControladoException(sqlException);
        }

}
```

```java
//devuelve un libro comprobando que el isbn existe, que su estado es ocupado y que el localizador
//coincide en la tabla LIBROS Y PRESTAMOS
//llama a las dos funciones anteriores para actualizar libRo y prestamo
@Override
public Optional<Prestamo> devolverLibro(String isbn, Date fecha_devolucion) throws AppException
{
    final String SQL = "SELECT P.ISBN                                                                  " +
                       "FROM LIBROS L, PRESTAMOS P                                                     " +
                       "WHERE P.ISBN = ? AND L.ESTADO = 'OCUPADO' AND P.LOCALIZADOR = L.LOCALIZADOR  " ;
    try ( PreparedStatement ps = connection.prepareStatement(SQL) )
    {
        ps.setString(1, isbn);
        try ( ResultSet rs = ps.executeQuery() )
        {
            if ( rs.next() )
            {
                actualizarLibroDevuelto(isbn);
                java.sql.Date sqlDate = new java.sql.Date( fecha_devolucion.getTime() );

                return Optional.of(actualizarPrestamoDevuelto(isbn, sqlDate));
            }
            else
            {
                return Optional.empty();
            }
        }

    } catch (SQLException sqlException)
        {
            String message = sqlException.getMessage();
            if ( message.contains("NN_LIBROS.ESTADO") )      throw new NnLibrosEstadoException();
            throw new SQLNoControladoException(sqlException);
        }
```

```java
@Override
public List<Historico> historicoLibro(String isbn) throws AppException
{
    final String SQL = "SELECT S.CODIGO, S.NOMBRE, P.FECHA_PRESTAMO, P.FECHA_DEVOLUCION " +
                       "FROM SOCIOS S, PRESTAMOS P                                       " +
                       " WHERE P.ISBN = ? AND P.DNI = S.DNI                              " +
                       "ORDER BY FECHA_PRESTAMO                                          " ;

    List<Historico> historicos = new LinkedList<>();

    try (PreparedStatement ps = connection.prepareStatement(SQL);
        )
    {
        ps.setString(1, isbn);
        try ( ResultSet rs = ps.executeQuery() )
        {
            if ( rs.next() )
            {
                do
                {
                    Historico historico = new Historico();
                    historico.setCodigo(rs.getString(1));
                    historico.setNombre(rs.getString(2));
                    historico.setFecha_prestamo(rs.getDate(3) );
                    historico.setFecha_devolucion(rs.getDate(4) );
                    historicos.add(historico);

                } while ( rs.next() );
            }
        }
    }
    catch (SQLException sqlException)
      {
          String message = sqlException.getMessage();
          if ( message.contains("NN_PRESTAMOS.ISBN") )      throw new NnPrestamosIsbnException();
          throw new SQLNoControladoException(sqlException);
      }
    return historicos;
}
```

```
public List<Historico2> historicoSocio(String codigo) throws AppException
{
    final String SQL = "SELECT L.ISBN, L.TITULO, P.FECHA_PRESTAMO, P.FECHA_DEVOLUCION " +
                       "FROM SOCIOS S, PRESTAMOS P, LIBROS L                          " +
                       " WHERE S.CODIGO = ? AND P.DNI = S.DNI AND P.ISBN = L.ISBN     " +
                       "ORDER BY FECHA_PRESTAMO                                       " ;

    List<Historico2> historicos = new LinkedList<>();

    try (PreparedStatement ps = connection.prepareStatement(SQL);
        )
    {
        ps.setString(1, codigo);
        try ( ResultSet rs = ps.executeQuery() )
        {
            if ( rs.next() )
            {
                do
                {
                    Historico2 historico = new Historico2();
                    historico.setIsbn(rs.getString(1));
                    historico.setTitulo(rs.getString(2));
                    historico.setFecha_prestamo(rs.getDate(3) );
                    historico.setFecha_devolucion(rs.getDate(4) );
                    historicos.add(historico);

                } while ( rs.next() );
            }
        }
    }
    catch (SQLException sqlException)
      {
          String message = sqlException.getMessage();
          if ( message.contains("NN_PRESTAMOS.ISBN") )      throw new NnPrestamosIsbnException();
          throw new SQLNoControladoException(sqlException);
      }
    return historicos;
}
```