



**INSTITUTO POLITECNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO**

“Practica 8-JDBC”

-Tinoco Videgaray Sergio Ernesto

Grupo: 3BV1

Materia: Paradigmas de programación

INSTITUTO POLITÉCNICO NACIONAL



ESCOM®

01/12/21

En esta práctica se va a desarrollar una GUI que va a permitir al usuario realizar consultas a una base de datos local, así como realizar altas, bajas y modificaciones por medio de la interfaz.

Primeramente se va a implementar la clase Libro para la gestión de los datos capturados, la cual va a funcionar como intermediario entre la base de datos y la interfaz de usuario.

```
private String isbn;  
private String titulo;  
private String autor;  
private String editorial;  
private String imagen;  
private double precio;
```

Se declaran las variables dentro de la clase Libro las cuales corresponden con los atributos del libro.

```
public Libro(String isbn, String titulo,  
             String autor, String edit,  
             String imagen, double precio)  
{  
    this.isbn = isbn;  
    this.titulo = titulo;  
    this.autor = autor;  
    this.editorial=edit;  
    this.imagen = imagen;  
    this.precio = precio;  
}
```

Constructor de la clase libro

```

public String getIsbn()
{
    return isbn;
}

public String getTitulo()
{
    return titulo;
}

public String getAutor()
{
    return autor;
}

public String getEditorial()
{
    return editorial;
}

public String getImagen()
{
    return "imagenes/" + imagen;
}

public double getPrecio()
{
    return precio;
}

```

Métodos para recuperar los atributos del libro consultado.

```

public void setIsbn(String isbn)
{
    this.isbn = isbn;
}

public void setTitulo(String titulo)
{
    this.titulo = titulo;
}

public void setAutor(String autor)
{
    this.autor = autor;
}

public void setEditorial(String editorial)
{
    this.editorial = editorial;
}

public void setPrecio(double precio)
{
    this.precio = precio;
}

public void setImagen(String imagen)
{
    this.imagen = imagen;
}

```

Métodos para establecer los atributos del libro que se va a agregar a la base de datos.

```
public void muestraDatos()
{
    System.out.println("ISBN : " + isbn);
    System.out.println("Titulo: " + titulo);
    System.out.println("Autor : " + autor);
    System.out.println("Precio: " + precio);
}
```

Método que muestra los datos del libro actual en la consola.

```
public static Libro getLibroFromDB(String isbnConsulta, Properties prop)
{
    Libro libro = new Libro(); // Nuevo libro en blanco

    try
    {
        String driver = prop.getProperty("dbdriver");
        String host = prop.getProperty("dbhost");
        String user = prop.getProperty("dbuser");
        String password = prop.getProperty("dbpassword");
        String name = prop.getProperty("dbname");
        String url = host + name + "?user=" + user + "&password=" + password;
        System.out.println("Conexion a la BD: " + url);
    }
}
```

Se declara un método que va a realizar la consulta a la base de datos y va a almacenar los valores en un objeto de la clase libro.

Se obtienen los parámetros del archivo properties para realizar la conexión a la base de datos.

Se realiza la conexión a la base de datos por medio del driver.

Posteriormente se declara un objeto de tipo PreparedStatement que va a realizar la consulta por medio de la sentencia Select * from libros, mandándole como parámetro el ISBN del libro.

```
Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD

PreparedStatement ps = con.prepareStatement("SELECT * FROM LIBROS WHERE ISBN = ?");
ps.setString(1, isbnConsulta);
System.out.println(ps.toString());
ps.executeQuery();
ResultSet rs = ps.getResultSet();
```

```

if(rs!=null && rs.next())
{
    String isbn    = rs.getString("isbn");
    String titulo = rs.getString("titulo");
    String autor  = rs.getString("autor");
    String editorial = rs.getString("editorial");
    String imagen = rs.getString("imagen");
    double precio = rs.getDouble("precio");

    libro.setIsbn(isbn);
    libro.setTitulo(titulo);
    libro.setAutor(autor);
    libro.setEditorial(editorial);
    libro.setImagen(imagen);
    libro.setPrecio(precio);
    con.close();
    return libro;
}

```

Se evalúa si la consulta existe y en dado caso se almacenan los valores de cada columna en una variable.

Posteriormente se almacenan los valores de las variables que contienen los atributos del libro en el objeto de la clase libro declarado previamente.

En caso de que la consulta no se ejecute correctamente se manda un mensaje de error en la consola.

```

catch (Exception ex)
{
    ex.printStackTrace();
}
return null;

```

De manera análoga se declara el método cambiar, que va a realizar una actualización de los datos almacenados. Por lo que vuelven a declarar las variables para obtener los parámetros del archivo properties y se cargan al controlador para realizar la conexión.

```

public boolean cambiar(Properties prop)
{
    boolean exito = false;

    try
    {
        String driver = prop.getProperty("dbdriver");
        String host    = prop.getProperty("dbhost");
        String user    = prop.getProperty("dbuser");
        String password = prop.getProperty("dbpassword");
        String name     = prop.getProperty("dbname");
        String url = host + name + "?user=" + user + "&password=" + password;
        System.out.println("Conexion a la BD: " + url);

        Class.forName(driver); // Carga el driver
    }
}

```

Se realiza la conexión a la base de datos por medio del controlador y se crea un objeto con la sentencia requerida para actualizar los datos de la BD, en este caso se utiliza la sentencia Update Libros para modificar los valores de cada columna. Mandándole como parámetro las variables del objeto libro y el ISBN para identificar el registro correspondiente.

Posteriormente se cierra la conexión a la BD.

```
Connection con = DriverManager.getConnection(url); // Crea una conexión a la BD

PreparedStatement ps = con.prepareStatement("UPDATE LIBROS SET TITULO = ?,Autor = ?,"
    + "Editorial = ?,Imagen = ?,Precio = ? WHERE ISBN = ?");

ps.setString(1, this.titulo); // El título que llega de la Vista
ps.setString(2, this.autor);
ps.setString(3, this.editorial);
ps.setString(4, this.imagen);
ps.setString(5, String.valueOf(this.precio));
ps.setString(6, String.valueOf(this.isbn));

System.out.println(ps.toString());
exito = ps.executeUpdate() > 0;
con.close();
```

En caso de que la consulta no se ejecute correctamente se manda un mensaje de error en la consola.

```
catch (Exception ex)
{
    ex.printStackTrace();
}
return null;
```

De igual forma se declara el método borrar, que va a eliminar un registro de la base de datos. Por lo que vuelven a declarar las variables para obtener los parámetros del archivo properties y se cargan al controlador para realizar la conexión.

```
public boolean borrar(Properties prop)
{
    boolean exito = false;

    try
    {
        String driver = prop.getProperty("dbdriver");
        String host    = prop.getProperty("dbhost");
        String user    = prop.getProperty("dbuser");
        String password = prop.getProperty("dbpassword");
        String name     = prop.getProperty("dbname");
        String url = host + name + "?user=" + user + "&password=" + password;
        System.out.println("Conexion a la BD: " + url);

        Class.forName(driver);        // Carga el driver
    }
}
```

En este caso se va a utilizar únicamente el ISBN para ubicar el registro del libro que se quiere eliminar.

```
Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD

PreparedStatement ps = con.prepareStatement("DELETE FROM LIBROS WHERE ISBN = ?");
ps.setString(1, this.isbn);
System.out.println(ps.toString());
exito = ps.executeUpdate() > 0;
con.close();
```

En caso de que la conexión no se realice correctamente se muestra un mensaje de excepción en consola.

```
catch (Exception ex)
{
    ex.printStackTrace();
}
return exito;
```

Por último se declara el método alta, para crear un nuevo registro en la base de datos.

De igual manera se van a cargar los valores del archivo properties en un controlador que va a realizar la conexión con la base de datos.

```
public boolean alta(Properties prop)
{
    boolean exito = false;

    try
    {
        String driver = prop.getProperty("dbdriver");
        String host    = prop.getProperty("dbhost");
        String user    = prop.getProperty("dbuser");
        String password = prop.getProperty("dbpassword");
        String name     = prop.getProperty("dbname");
        String url = host + name + "?user=" + user + "&password=" + password;
        System.out.println("Conexion a la BD: " + url);

        Class.forName(driver);        // Carga el driver
    }
}
```

Para este caso se va a utilizar la sentencia "Insert into libros, values".

Por lo que se van a mandar como parámetros los valores del objeto libro en el orden que se muestra a continuación:

```
PreparedStatement ps = con.prepareStatement("INSERT INTO LIBROS (ISBN, TITULO,AUTOR,EDITORIAL,IMAGEN,PRECIO) "
                                           + "VALUES (?, ?, ?, ?, ?, ?)");

ps.setString(1, this.isbn);
ps.setString(2, this.titulo);
ps.setString(3, this.autor);
ps.setString(4, this.editorial);
ps.setString(5, this.imagen);
ps.setString(6, String.valueOf(this.precio));
System.out.println(ps.toString());
exito = ps.executeUpdate() > 0;
con.close();
```

En caso de que la conexión no se realice correctamente se muestra un mensaje de excepción en consola.

```
catch (Exception ex)
{
    ex.printStackTrace();
}
return exito;
```


Clase Vista:

Se define la clase Vista que hereda de la clase JFrame.

Se declaran los componentes necesarios para generar la interfaz de usuario.

```
public class Vista extends JFrame
{
    private JTextField tIsbn = new JTextField();
    private JTextField tTitulo= new JTextField();
    private JTextField tAutor = new JTextField();
    private JTextField tEditorial= new JTextField();
    private JTextField tPrecio = new JTextField();
    private JTextField tNombreImagen= new JTextField();

    private JButton btBuscar = new JButton("Buscar");
    private JButton btInsertar = new JButton("Agregar");
    private JButton btBorrar = new JButton("Eliminar");
    private JButton btCambiar = new JButton("Cambiar");
    private JButton btLimpiar = new JButton("Limpiar");
}
```

Se declara el constructor de la clase vista que recibe como parámetro un objeto de tipo properties y va a generar el nombre de la ventana, las coordenadas y los tamaños del Frame.

Se cargan los valores del archivo properties en la variable prop.

En caso de que se genere una excepción se muestra un error en consola.

```
public Vista(Properties prop)
{
    this.prop = prop;
    initComponents();
    this.setTitle("Librerías de Cristal (CRUD del Catálogo)");
    this.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
    this.setLayout(null);
    this.setBounds(10,10,700,400);

    // Carga las propiedades desde el archivo
    try
    {
        prop.load(new FileInputStream("config.properties"));
    }
    catch (Exception ex)
    {
        ex.printStackTrace();
    }
}
```

Método initComponents:

Se declaran objetos de tipo JLabel para las etiquetas de la interfaz.

```
JLabel et1      = new JLabel("ISBN:");  
JLabel et2      = new JLabel("Título:");  
JLabel et3      = new JLabel("Autor(es):");  
JLabel et4      = new JLabel("Editorial:");  
JLabel et5      = new JLabel("Precio:");  
JLabel et6      = new JLabel("Archivo de imagen:");
```

Se generan los objetos para la barra de menú del frame.

```
// Diseña el menu  
  
JMenuBar barraMenus = new JMenuBar();  
JMenu menuArchivo   = new JMenu("Archivo");  
JMenuItem opSalir   = new JMenuItem("Salir");  
this.setJMenuBar(barraMenus);  
  
barraMenus.add(menuArchivo);  
menuArchivo.add(opSalir);
```

Se establecen los tamaños y las coordenadas de las etiquetas.

```
et1.setBounds(10, 30, 100, 25);  
et2.setBounds(10, 70, 100, 25);  
et3.setBounds(10, 110, 100, 25);  
et4.setBounds(10, 150, 100, 25);  
et5.setBounds(10, 190, 100, 25);  
et6.setBounds(10, 230, 175, 25);
```

Se establecen los tamaños y las coordenadas de las cajas de texto.

```
imagen.setBounds(530, 30, 125, 170);  
tIsbn.setBounds(80, 30, 100, 25);  
tTitulo.setBounds(80, 70, 400, 25);  
tAutor.setBounds(80, 110, 300, 25);  
tEditorial.setBounds(80, 150, 200, 25);  
tPrecio.setBounds(80, 190, 150, 25);  
tNombreImagen.setBounds(150, 230, 150, 25);
```

Se establecen los tamaños y las coordenadas de los botones.

```
btBuscar.setBounds(200,30,80,25);  
btInsertar.setBounds(80,280,80,25);  
btBorrar.setBounds(200,280,80,25);  
btCambiar.setBounds(320,280,85,25);  
btLimpiar.setBounds(440,280,80,25);
```

```
add(et1);  
add(et2);  
add(et3);  
add(et4);  
add(et5);  
add(et6);  
  
add(tIsbn);  
add(tTitulo);  
add(tAutor);  
add(tEditorial);  
add(tPrecio);  
add(tNombreImagen);  
  
add(btBuscar);  
add(btInsertar);  
add(btBorrar);  
add(btCambiar);  
add(btLimpiar);  
add(imagen);
```

Se añaden los componentes previamente establecidos al frame actual.

Se añaden los listeners correspondientes a los botones del frame.

```
opSalir.addActionListener(evt -> gestionaSalir(evt));  
btBuscar.addActionListener(evt -> gestionaBuscar(evt));  
btInsertar.addActionListener(evt -> gestionaInsertar(evt));  
btBorrar.addActionListener(evt -> gestionaBorrar(evt));  
btCambiar.addActionListener(evt -> gestionaCambiar(evt));  
btLimpiar.addActionListener(evt -> gestionaLimpiar(evt));
```

Métodos para el botón salir, en caso de que el usuario presione el botón salir se va a mostrar en pantalla un mensaje de aviso preguntándole al usuario si desea salir del programa, en caso de ser así se termina la ejecución del programa y de la máquina virtual de java.

```
public void gestionaSalir(java.awt.event.ActionEvent evt)
{
    exit();
}

public void exit()
{
    int respuesta = JOptionPane.showConfirmDialog(rootPane, "Desea salir?", "Aviso", JOptionPane.YES_NO_OPTION);
    if(respuesta==JOptionPane.YES_OPTION) System.exit(0);
}
```

Método para buscar un registro dentro de la base de datos por medio del ISBN.

En caso de que la caja de texto del ISBN este vacía, se mostrara un mensaje de advertencia al usuario ya que no se puede realizar la consulta sin un ISBN.

```
public void gestionaBuscar(java.awt.event.ActionEvent evt)
{
    if(tIsbn.getText().isBlank())
    {
        JOptionPane.showMessageDialog(this, "Para localizar un libro se requiere el ISBN", "Aviso!", JOptionPane.ERROR_MESSAGE);
    }
}
```

En caso contrario, se realiza la consulta por medio de un objeto tipo Libro.

Una vez se carguen los datos del libro en el objeto libro se van a mostrar en las cajas de texto correspondientes así como la imagen del libro.

```
else
{
    Libro newBook = Libro.getLibroFromDB(tIsbn.getText(),prop); // Método estático para obtener un libro desde la BD
    if(newBook != null) // Si hubo éxito
    {
        tTitulo.setText(newBook.getTitulo());
        tAutor.setText(newBook.getAutor());
        tEditorial.setText(newBook.getEditorial());
        tPrecio.setText(String.valueOf(newBook.getPrecio()));
        tNombreImagen.setText((newBook.getImagen()).substring(9));

        String nombreArchivoImagen = newBook.getImagen();
        imagen.setIcon(new ImageIcon(nombreArchivoImagen));
    }
    else JOptionPane.showMessageDialog(this, "El libro con el ISBN indicado no fue localizado", "Aviso!", JOptionPane.ERROR_MESSAGE);
}
```

Método para actualizar los datos del libro previamente consultado por medio del ISBN.

En caso de que la caja de texto del ISBN este vacía, se mostrara un mensaje de advertencia al usuario ya que no se puede realizar la actualización sin un ISBN.

```
public void gestionaCambiar(java.awt.event.ActionEvent evt)
{
    if(tIsbn.getText().isBlank())
    {
        JOptionPane.showMessageDialog(this, "Para localizar el libro que se va a actualizar se requiere el ISBN", "Aviso!", JOptionPane.ERROR_MESSAGE);
    }
    else
    {

```

En caso contrario, se evalua si las casillas tienen valores capturados y de ser el caso se realiza la consulta. En caso de que se presente un error de consulta se mostrara una mensaje de error en pantalla.

```
if(!invalido()) // Se intenta realizar el UPDATE solo si no hay error de captura
{
    Libro newBook = Libro.getLibroFromDB(tISBN.getText(),prop); // Método estático para obtener un libro desde la BD
    if(newBook != null)
    {
        newBook.setTitulo(tTitulo.getText()); // Actualiza el titulo del objeto libro
        newBook.setAutor(tAutor.getText()); // Actualiza el autor del objeto libro
        newBook.setEditorial(tEditorial.getText()); // Actualiza la editorial del objeto libro
        newBook.setPrecio(Double.parseDouble(tPrecio.getText())); // Actualiza el precio del objeto libro
        newBook.setImagen(tNombreImagen.getText()); // Actualiza el nombre del archivo imagen del objeto libro

        if(newBook.cambiar(prop)) // Si hubo éxito
            JOptionPane.showMessageDialog(this, "Registro actualizado: " + tISBN.getText(), "Aviso!",JOptionPane.INFORMATION_MESSAGE);
        else
            JOptionPane.showMessageDialog(this, "Acción no realizada!!", "Aviso!",JOptionPane.ERROR_MESSAGE);
    }
    else JOptionPane.showMessageDialog(this, "El libro con el ISBN indicado no fue localizado", "Aviso!", JOptionPane.ERROR_MESSAGE);
}
else JOptionPane.showMessageDialog(this, mensajeError, "Aviso!", JOptionPane.ERROR_MESSAGE);
```

Se define un método para realizar una alta en la base de datos.

Se evalua si los campos de los datos estan llenos y de ser asi se instancia un objeto de tipo libro , y se comprueba que el ISBN del libro no este ya registrado en la BD.

Posteriormente se almacenan los datos en el obtejo de la clase libro.

```
public void gestionaInsertar(java.awt.event.ActionEvent evt)
{
    if(!invalido()) // Se intenta realizar el INSERT solo si no hay error de captura
    {
        // Primero investigamos si no hay otro registro con el mismo ISBN
        Libro newBook = Libro.getLibroFromDB(tISBN.getText(),prop);

        if(newBook == null) // Solo si el ISBN no está registrado
        {
            // Adquirimos los datos de la vista
            newBook = new Libro();
            newBook.setIsbn(tISBN.getText());
            newBook.setTitulo(tTitulo.getText());
            newBook.setAutor(tAutor.getText());
            newBook.setEditorial(tEditorial.getText());
            newBook.setPrecio(Double.parseDouble(tPrecio.getText()));
            newBook.setImagen(tNombreImagen.getText());
        }
    }
}
```

Se invoca al método “alta” de la clase libro y se realiza la consulta correspondiente, en caso de que se presente algún error de consulta se mostrara un mensaje de error.

```
if(newBook.alta(prop)) // Si la alta fue exitosa
    JOptionPane.showMessageDialog(this, "Registro agregado: " + tISBN.getText(), "Aviso!",JOptionPane.INFORMATION_MESSAGE);
else
    JOptionPane.showMessageDialog(this, "Acción no realizada!!", "Aviso!",JOptionPane.ERROR_MESSAGE);
}
else JOptionPane.showMessageDialog(this, "El ISBN ya está registrado", "Aviso!", JOptionPane.ERROR_MESSAGE);
}
else JOptionPane.showMessageDialog(this, mensajeError, "Aviso!", JOptionPane.ERROR_MESSAGE);
```

Método para borrar un registro de la BD.

Se comprueba si el campo del ISBN tiene texto y de no ser así se mostrara un mensaje de error al usuario.

```
public void gestionaBorrar(java.awt.event.ActionEvent evt)
{
    if(tIsbn.getText().isBlank())
    {
        JOptionPane.showMessageDialog(this, "Para localizar el libro que se va a eliminar se requiere el ISBN", "Aviso!",
    }
    else
    {

```

En caso contrario se mostrara un mensaje de advertencia al usuario para preguntarle si desea eliminar el registro indicado.

En caso de que el usuario elija la opcion "YES" se va a instanciar un objeto de la clase libro.

Se evalúa si el registro existe y en dado caso se manda a llamar al método borrar de la clase libro, si la sentencia se ejecutó correctamente se muestra un mensaje de Registro eliminado y se invoca al método limpiarCampos.

En caso de que se produzca algún error al realizar la baja se mostrara un mensaje de error al usuario.

```
// Solicitamos confirmacion
int respuesta = JOptionPane.showConfirmDialog(this, "Desea borrar este registro?", "Atención!!!", JOptionPane.YES_NO_OPTION,

if(respuesta==JOptionPane.YES_OPTION) // Si el usuario confirma
{
    Libro newBook = Libro.getLibroFromDB(tIsbn.getText(),prop); // Trata de recuperar el libro de la BD

    if(newBook != null) // Si lo encuentra
    {
        // Intenta eliminar el registro
        if(newBook.borrar(prop)) // Si hubo éxito
        {
            JOptionPane.showMessageDialog(this, "Registro eliminado: " + tIsbn.getText(), "Aviso!",JOptionPane.WARNING_MESSAGE);
            limpiarCampos();
        }
        else JOptionPane.showMessageDialog(this, "Acción no realizada!!", "Aviso!",JOptionPane.ERROR_MESSAGE);
    }
    else JOptionPane.showMessageDialog(this, "El libro con el ISBN indicado no fue localizado", "Aviso!", JOptionPane.ERROR_M
}
}
```

Método para vaciar las cajas de texto y la imagen del libro.

```
private void limpiarCampos()
{
    tIsbn.setText("");
    tTitulo.setText("");
    tAutor.setText("");
    tEditorial.setText("");
    tPrecio.setText("");
    tNombreImagen.setText("");
    imagen.setIcon(null);
}
```

Método para comprobar si todas las cajas de texto están llenas, en caso de no ser así mostrara un mensaje de error al usuario.

```
// Validación de datos
private boolean invalido()
{
    boolean hayError = false;
    mensajeError = "";

    if(tISBN.getText().isBlank())
    {
        hayError = true;
        mensajeError = mensajeError.concat("No debe dejar el ISBN en blanco\n");
    }

    if(tTitulo.getText().isBlank())
    {
        hayError = true;
        mensajeError = mensajeError.concat("No debe dejar el título en blanco\n");
    }
}
```

Clase Main:

Se declara un objeto de tipo Properties para recuperar los parámetros del archivo config.properties el cual almacena los valores necesarios para realizar la conexión a la base de datos libro.

En caso de que se presente alguna excepción se va a mostrar un error en la ventana del usuario y se imprime en consola el código del error.

De otro modo se cargan los valores recuperados a las variables correspondientes para el driver.

```
public static void main(String[] args)
{
    // Testing Database

    boolean todoBien = true;
    Properties prop = new Properties(); // Para guardar la conf de la base de datos

    // Carga las propiedades desde el archivo
    try
    {
        prop.load(new FileInputStream("config.properties"));
    }
    catch (Exception ex)
    {
        todoBien = false;
        JOptionPane.showMessageDialog(null, "Problema con el archivo de propiedades", "Aviso!", JOptionPane.ERROR_MESSAGE);
        ex.printStackTrace();
    }

    if(todoBien)
    {
        String driver = prop.getProperty("dbdriver");
        String host = prop.getProperty("dbhost");
        String user = prop.getProperty("dbuser");
        String password = prop.getProperty("dbpassword");
        String name = prop.getProperty("dbname");
        String url = host + name + "?user=" + user + "&password=" + password;
        System.out.println("Conexion a la BD: " + url);
    }
}
```


Finalmente se realiza la conexión a la BD y se crea la vista del usuario.

```
try
{
    Class.forName(driver); // Carga el driver
    Connection con = DriverManager.getConnection(url); // Crea una conexión a la BD
    con.close();
}
catch (ClassNotFoundException ex)
{
    todoBien = false;
    JOptionPane.showMessageDialog(null, "Problema al cargar el driver", "Aviso!!!", JOptionPane.ERROR_MESSAGE);
}
catch (SQLException ex)
{
    todoBien = false;
    JOptionPane.showMessageDialog(null, "Problema al tratar de hacer la conexión", "Aviso!!!", JOptionPane.ERROR_MESSAGE);
}

if(todoBien) // Solo si todo está bien con la BD
{
    Vista v = new Vista(prop);
    v.setVisible(true);
}
}
```

Pruebas de funcionamiento:

Librerías de Cristal (CRUD del Catálogo)

Archivo

ISBN: 9780201433289


Título: JDBC(TM) API Tutorial and Reference

Autor(es): White, Set

Editorial: Sun Microsystems

Precio: 250.0

Archivo de imagen: 0010.jpg



ISBN: 00141

Título:

Autor(es):

Editorial:

Precio:

Archivo de imagen:

Aviso!
El libro con el ISBN indicado no fue localizado

ISBN: 00141

Título: Java para niños

Autor(es): Nadia Ameziane Garcia

Editorial: Kindle

Precio: 312.45

Archivo de imagen: 0011.jpg

Aviso!
Registro agregado: 00141

ISBN: 00141


Título: Java para niños

Autor(es): Nadia Ameziane Garcia

Editorial: Kindle

Precio: 312.0

Archivo de imagen: 0011.jpg



ISBN: 00141

Título: Java para niños

Autor(es): Nadia Ameziane Garcia


Editorial: Kindle

Precio: 300.0

Archivo de imagen: 0011.jpg

Aviso!

Registro actualizado: 00141



ISBN: 00141

Título: Java para niños

Autor(es): Nadia Ameziane Garcia


Editorial: Kindle

Precio: 300.0

Archivo de imagen: 0011.jpg

Aviso!

Registro eliminado: 00141



ISBN: 00141

Título:

Autor(es):

Editorial:

Precio:

Aviso!

El libro con el ISBN indicado no fue localizado

ISBN: 00142

Título: SQL. Los fundamentos del lenguaje

Autor(es): Anne-Christine Bisson

Editorial: Eni

Precio: 794

Archivo de imagen: 0013.jpg

Aviso!

Registro agregado: 00142

ISBN: 00142

Título: SQL. Los fundamentos del lenguaje

Autor(es): Anne-Christine Bisson-Eric Godoc

Editorial: Eni

Precio: 794.0

Archivo de imagen: 0013.jpg



ISBN: 123456

Título: Redes Neuronales

Autor(es): Roselph Rosse

Editorial:

Precio: 399

Archivo de imagen: 0012.jpg

Aviso!

No debe dejar la editorial en blanco

ISBN: 123456

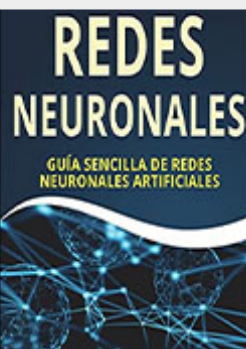
Título: Redes Neuronales

Autor(es): Roselph Rosse

Editorial: CPNE

Precio: 399.0

Archivo de imagen: 0012.jpg



Observaciones y conclusiones:

En el desarrollo de la práctica tuve complicaciones a la hora de establecer la conexión a la base de datos debido a que el archivo properties no estaba bien editado, ya que el programa hace uso de dicho archivo para obtener los valores necesarios para el conector.

De igual forma en esta práctica resalta el uso de una clase tipo libro para ayudar a gestionar los datos recuperados de la base de datos y mostrarlos en la interfaz, por lo que el uso de clases y variables resulta ser un aspecto clave para almacenar y gestionar los datos en un programa.

Por lo anterior puedo concluir que una base de datos permite almacenar información en una unidad de almacenamiento no volátil por lo que los datos quedan almacenados incluso cuando el dispositivo está apagado y cuando se requiera acceder a ellos es necesario utilizar un programa que por medio de un adaptador o “driver” establezca una conexión y los muestre en una interfaz grafica.