**Actividad 1.**

**Hilos y Sockets**

**Ivan Nuñez Rodriguez**

2º DAM

Programación de Procesos y Servicios

**Índice**

1. Código.
2. Resultado.
3. Enlace de Github

**Código**

**Clase CLienteBiblioteca**

import java.io.\*;  
import java.net.Socket;  
import java.util.Scanner;  
  
public class ClienteBiblioteca {  
 private static final String *HOST* = "localhost";  
 private static final int *PUERTO* = 12345;  
  
 public static void main(String[] args) {  
 try (Socket cliente = new Socket(*HOST*, *PUERTO*);  
 ObjectOutputStream salida = new ObjectOutputStream(cliente.getOutputStream());  
 ObjectInputStream entrada = new ObjectInputStream(cliente.getInputStream());  
 Scanner scanner = new Scanner(System.*in*)) {  
  
 String opcion;  
 do {  
 System.*out*.println("\n--- Menú de la Biblioteca ---");  
 System.*out*.println("1. Consultar libro por ISBN");  
 System.*out*.println("2. Consultar libro por título");  
 System.*out*.println("3. Consultar libros por autor");  
 System.*out*.println("4. Añadir libro");  
 System.*out*.println("5. Salir de la aplicación");  
 System.*out*.print("Seleccione una opción: ");  
 opcion = scanner.nextLine();  
 salida.writeObject(opcion);  
  
 switch (opcion) {  
 case "1":  
 System.*out*.print("Introduzca el ISBN: ");  
 String isbn = scanner.nextLine();  
 salida.writeObject(isbn);  
 System.*out*.println("Respuesta: " + entrada.readObject());  
 break;  
 case "2":  
 System.*out*.print("Introduzca el título: ");  
 String titulo = scanner.nextLine();  
 salida.writeObject(titulo);  
 System.*out*.println("Respuesta: " + entrada.readObject());  
 break;  
 case "3":  
 System.*out*.print("Introduzca el autor: ");  
 String autor = scanner.nextLine();  
 salida.writeObject(autor);  
 System.*out*.println("Respuesta: " + entrada.readObject());  
 break;  
 case "4":  
 System.*out*.print("ISBN: ");  
 String nuevoISBN = scanner.nextLine();  
 System.*out*.print("Título: ");  
 String nuevoTitulo = scanner.nextLine();  
 System.*out*.print("Autor: ");  
 String nuevoAutor = scanner.nextLine();  
 salida.writeObject(new Libro(nuevoISBN, nuevoTitulo, nuevoAutor));  
 System.*out*.println("Respuesta: " + entrada.readObject());  
 break;  
 case "5":  
 System.*out*.println("Cerrando la sesión...");  
 System.*out*.println("Respuesta del servidor: " + entrada.readObject());  
 break;  
 default:  
 System.*out*.println("Opción no válida.");  
 break;  
 }  
 } while (!opcion.equals("5"));  
  
 } catch (IOException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
}

**clase Libro**

import lombok.\*;  
  
import java.io.Serializable;  
import java.util.List;  
import java.util.stream.Collectors;  
@Getter  
@Setter  
@ToString  
@NoArgsConstructor  
@AllArgsConstructor  
public class Libro implements Serializable {  
 private String ISBN;  
 private String titulo;  
 private String autor;  
  
 // Métodos de consulta  
 public static Libro consultarPorISBN(List<Libro> libros, String isbn) {  
 return libros.stream().filter(libro -> libro.getISBN().equals(isbn)).findFirst().orElse(null);  
 }  
  
 public static Libro consultarPorTitulo(List<Libro> libros, String titulo) {  
 return libros.stream().filter(libro -> libro.getTitulo().equalsIgnoreCase(titulo)).findFirst().orElse(null);  
 }  
  
 public static List<Libro> consultarPorAutor(List<Libro> libros, String autor) {  
 return libros.stream().filter(libro -> libro.getAutor().equalsIgnoreCase(autor)).collect(Collectors.*toList*());  
 }  
}

**clase SevidorBiblioteca**

import java.io.IOException;  
import java.io.ObjectInputStream;  
import java.io.ObjectOutputStream;  
import java.net.ServerSocket;  
import java.net.Socket;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.concurrent.ExecutorService;  
import java.util.concurrent.Executors;  
  
public class ServidorBiblioteca {  
 private static final int *PUERTO* = 12345;  
 private static List<Libro> *libros* = new ArrayList<>();  
 private static volatile boolean *enEjecucion* = true;  
  
 public static void main(String[] args) {  
 *inicializarLibros*();  
 ExecutorService pool = Executors.*newFixedThreadPool*(10);  
  
 try (ServerSocket servidor = new ServerSocket(*PUERTO*)) {  
 System.*out*.println("Servidor iniciado en el puerto " + *PUERTO*);  
  
 while (*enEjecucion*) {  
 Socket cliente = servidor.accept();  
 pool.execute(() -> *gestionarCliente*(cliente));  
 }  
 pool.shutdownNow();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 System.*out*.println("Servidor cerrado.");  
 }  
  
 private static void inicializarLibros() {  
 *libros*.add(new Libro("1234567890", "El Principito", "Antoine de Saint-Exupéry"));  
 *libros*.add(new Libro("0987654321", "1984", "George Orwell"));  
 *libros*.add(new Libro("5432167890", "Fahrenheit 451", "Ray Bradbury"));  
 *libros*.add(new Libro("6789054321", "La Odisea", "Homero"));  
 *libros*.add(new Libro("1029384756", "Moby Dick", "Herman Melville"));  
 }  
  
 private static void gestionarCliente(Socket cliente) {  
 try (ObjectInputStream entrada = new ObjectInputStream(cliente.getInputStream());  
 ObjectOutputStream salida = new ObjectOutputStream(cliente.getOutputStream())) {  
  
 String opcion;  
 while ((opcion = (String) entrada.readObject()) != null) {  
 switch (opcion) {  
 case "1":  
 String isbn = (String) entrada.readObject();  
 salida.writeObject(Libro.*consultarPorISBN*(*libros*, isbn));  
 break;  
 case "2":  
 String titulo = (String) entrada.readObject();  
 salida.writeObject(Libro.*consultarPorTitulo*(*libros*, titulo));  
 break;  
 case "3":  
 String autor = (String) entrada.readObject();  
 salida.writeObject(Libro.*consultarPorAutor*(*libros*, autor));  
 break;  
 case "4":  
 Libro nuevoLibro = (Libro) entrada.readObject();  
 synchronized (*libros*) {  
 *libros*.add(nuevoLibro);  
 }  
 salida.writeObject("Libro añadido con éxito.");  
 break;  
 case "5":  
 System.*out*.println("Cliente " + cliente.getInetAddress() + " ha cerrado la sesión.");  
 salida.writeObject("Sesión cerrada correctamente.");  
  
 return;  
 default:  
 salida.writeObject("Opción no válida.");  
 break;  
 }  
 }  
 } catch (IOException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
}

**Ejecución**

**Ejecución ServidorBiblioteca**

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

**Ejecución ClienteBiblioteca**

**Texto

Descripción generada automáticamente**

**Ejecución opción 1**

**Texto

Descripción generada automáticamente**

**Ejecución opción 2**

**Texto

Descripción generada automáticamente**

**Ejecución opción 3**

**Texto

Descripción generada automáticamente**

**Ejecución opción 4**

**Forma

Descripción generada automáticamente con confianza media**

**Ejecución comprobación 4**

**Texto

Descripción generada automáticamente**

**Ejecuncion opción 5**

**Texto

Descripción generada automáticamente**

**Ejecución finalización programa parte servidor**

**Texto

Descripción generada automáticamente**