INFORME FINAL

SISTEMA DE GESTION DE BIBLIOTECAS

Curso: Programación Orientada a Objetos.

Profesor: Herminio Paucar.

Grupo: G5

INTEGRANTES:

- Pariona Moya ,Douglas 18190095
- Ruberto Yin Lin, Jose Luis 18190317
- Soto Salazar ,Frank Sandro 18190100
- Espinoza Bendezú , Franco Enrique 18190092
- Valenzuela Morales , Pool Anderson 18190320
- Rodriguez Romero, River David 18190098

2019

INTRODUCCION

En la actualidad existe una amplia gama bibliográfica disponible por diferentes medios debido a la necesidad de las personas de conseguir información ya sea por informarse, por deberes, entretenimiento o por realizar estudios a manera de investigaciones por obtener grados profesionales.

Por ello, las instituciones que otorgan servicios educativos están obligadas a contar con fuentes bibliográficas a manera de ayuda al público en general, ofreciéndole las facilidades de poder obtener la información que necesitan de fuentes confiables.

Al ser tan amplio el contenido bibliográfico, requiere de un sistema que permita organizarlo de manera eficiente, que facilite búsqueda de información bibliográfica, así como también el estado, ubicación, etc. Para ello se ha realizado una forma de poder sistematizar toda esta información y que este disponible para las personas. La forma de realizar esto es mediante un sistema de biblioteca que nos facilite desde los datos bibliográficos hasta la información de los usuarios, así como la de los bibliotecarios y su superior el administrador.

Motivación

Como todo centro o institución donde se imparte la educación siempre cuenta con un inventario de material bibliográfico con la cual los estudiantes realizan sus investigaciones.

Puesto que el material que se ofrece tiene una gran variedad nos encontramos motivados a poder administrarlo de manera que conozcamos su ubicación en el establecimiento (biblioteca), código, genero de libro, títulos, estado y disponibilidad, así como también saber que fuentes complementarias son necesarias acordes a los requerimientos de los usuarios en la biblioteca y que estos tengan un fácil acceso a la información para realizar sus deberes.

Problema

A lo que nos enfrentamos es que la forma de administrar toda esta información es que se hace por medio de archivos en físico lo cual es tedioso al momento de seleccionar lo requerid; vuelve poco practicas las acciones de solicitar material, búsqueda, entrega y disponibilidad.

En consecuencia, lo que se obtiene es, desde no encontrar lo requerido, hasta perdidas del material bibliográfico. Para evitar este tipo de inconvenientes se creo el sistema de administración virtual de biblioteca.

Requerimientos del Sistema

El sistema debe cumplir con ciertos estándares que aseguren su eficiencia y que cubra con las necesidades requeridas. Entre estos requisitos tenemos:

Funcionales:

Son las características del sistema que deben cumplirse para que el usuario pueda interactuar sin inconvenientes con el producto , estos deben ser claro incluso para personas que no nociones de programación. Entre estos tenemos:

Proceso:

_El programa nos dirá si el lector cuenta ya con un préstamo o no, además si el libro fue devuelto y si en caso no haya sido así se le aplicará una multa que aumentará por cada día luego del día al que se le propuso al lector devolver.

_ El programa también permitirá al bibliotecario dar a conocer mediante el programa si el libro se encuentra en disponible o no en caso no lo esté se procederá a repararlo para su pronto uso.

_El sistema contara con tres usuarios: Administrador, bibliotecario y lector , además tendrá una jerarquía en la cual el administrador será apto para añadir

y heredar sus funciones, y estos a su vez, serán capaces de añadir a los lectores.

_El sistema muestra una ventana principal con una barra de búsqueda en el que el lector puede buscar el material que desea.

_ El administrador y bibliotecario cuando ingresen ingrese cada uno tendrán sus respectivas funciones.

_Se presentan maneras de búsqueda rápida como titulo,autor,tomo,etc.

_ Una vez el lector reserve el libro, inmediatamente, el sistema debe guardar la información en una base de datos, sea Excel (según se considere más optimo), para ello cabe recalcar el uso de la librería apache.

_Una vez el sistema haya guardado la reserva del libro, este mostrará mediante un panel al bibliotecario las reservas recientes.

_Una vez el bibliotecario haya registrado el préstamo, el sistema guardará la información en la base de datos con una fecha inicial y final de préstamo.

Interfaz Gráfica:

El lector podrá buscar un libro y se le facilitará dandole opciones de búsqueda los cuales serán: Título, Autor, Género, en caso el lector no se acuerde del nombre correcto del libro que desee prestarse el sistema le proporcionará resultados coincidentes por el nombre al que se le ingrese en el campo título.

Legales:

El sistema controlará el acceso y permitirá solamente a los usuarios autorizados.

Seguridad:

El sistema aceptará solamente que el bibliotecario tenga acceso al préstamo del libro. El sistema pedirá el acceso de una contraseña a los bibliotecarios y al lector se le pedirá identificarse con su código.

Técnicas:

Definen las condiciones que el programa debe cumplir, así como las limitaciones que el este posee.

Accesibilidad:

El administrador solamente puede obtener acceso a todo el sistema en sí. En cuanto al bibliotecario, tendrá acceso a la opción de préstamos de libros y también a agregar libros los cuales se podrá añadir mediante una base de datos. El lector únicamente tiene el acceso a ver que libros cuenta nuestra biblioteca y si se encuentran disponibles.

Encriptación:

Solo el administrador podrá agregar y eliminar a los bibliotecarios ya que estos son importantes para los préstamos que realizará hacia un lector. En cuanto al lector este obtendrá un usuario que el bibliotecario le brindará para que esté pueda solicitar el préstamo de un libro.

Entorno

Se controlará mediante una base de datos las cuales podremos añadir libros desde ahí .Es primordial por que con está podremos expandir nuestra biblioteca.

Entorno:

Se controlará mediante una Excel las cuales podremos añadir libros desde ahí.

Operacionales:

Son las funciones y aseguramientos que nos garantizan que el programa va a continuar funcionando con total normalidad a lo largo del tiempo y de presentar algún tipo de falla, este pueda dar un aviso para poder aplicarse un respectivo mantenimiento y las funciones a largo plazo no se vean afectadas.

Administración del sistema:

La administración del sistema depende de cómo éste utiliza y asigna sus recursos.

Rendimiento del sistema:

Controla el rendimiento del sistema con regularidad para saber cómo se comporta en condiciones normales y en caso de algún error, los administradores podrán notar las irregularidades.

Transicionales:

Son las condiciones del producto que deben implementarse para que pueda ejecutarse exitosamente.

Formación:

Si bien es cierto el orden y separación de cada operación es importante, por ello es necesario implementar un correcto uso de las jerarquías operacionales.

Conversión de datos:

Para este proceso debemos implementar un sistema de interfaz gráfica, de este modo podrá ser viste de manera sencilla por el usuario.

Documentación:

Para cumplir con los objetivos del sistema de biblioteca debemos realizar implementaciones para tener la funcionalidad planteada en los requerimientos, debemos instalar un sistema de información que nos diga que existe conexión con la base de datos y así funcione de una manera óptima.

Código fuente

Usamos la clase Modelo Excel en el cual aquí guardaremos nuestros libros que hemos ingresado para eso utilizamos la librería apache que nos sirve para poder conectar nuestros datos a un archivo de Excel.

Ya explicado esto procedemos a crear una clase con el que podamos exportar nuestros datos a un Excel y que estos datos se almacenen ahí.

Por ejemplo, importamos:

org.apache.poi.EncryptedDocumentException para poder abrir un archivo de Excel y poder descifrar lo que contiene en su interior.

org.apache.poi.ss.usermodel.*

org.apache.poi.openxml4j.exceptions.InvalidFormatException , estos códigos nos ayudan a la exportación de los datos al Excel.

Creamos un código en cual cada dato ingresado en la tabla aparezca en el Excel, y así también para podamos a partir de un Excel podamos importarlo a nuestro JTable.

En el package Controllers crearemos 2 clases una de seguridad en esta clase es exclusiva para comprobar y comparar el texto ingresado con la contaraseña real con un código que nos diga si está contraseña ingreasada es correcta o no a través de un boolean ya que cada administrador cuenta con una contraseña y usuario único .

Si nuestra contraseña es correcta entonces lo redigimos a las ventanas respectivas .

```
if((usuarios[0].equalsIgnoreCase(user)&&usuarios[1].equals(pwd))){
```

ventanaadmin.setVisible(true); //SI LA CONDICION ES CIERTA ABRE LA VENTANA DEL ADMIN

Si en caso pruebe dos veces

```
if(intentos>2){
            JOptionPane.showMessageDialog(null, "3 intentos fallidos, esto se cerrará","Inicio de Sesión",JOptionPane.ERROR_MESSAGE);
      System.exit(0);
```

Con este código podrá solo 3 intentos si en caso no se le cerra .

En el caso de nuestra clase SIBprincipal creamos ponemos un

```
InicioSIB ventana = new InicioSIB();
            ventana.setVisible(true) ;
```

para que se pueda hacer visible .

En nuestros 2 packages donde hay 8 clases , dos en la primera “controllers” y 6 en la segunda “views” hicimos uso de importaciones que nos permitieron realizar el proceso a la hora de la ejecución, vale recordar que todo es importante y comentamos en varias líneas código lo que se avanzo , las importaciones que utilizamos fueron las siguientes.

FileInputSt|ream

Este sirve para obtener bytes de entrada de un archivo en un sistema de archivos estos archivos disponibles dependen del entorno del host. También está diseñado para leer flujos de bytes en bruto, como datos de imagen. Para leer secuencias de caracteres, consideramos el FileReader.

FileOutputStream

Es un flujo de salida para escribir datos en un archivo o en un FileDescriptor. Si un archivo está o no disponible o se puede crear depende de la plataforma subyacente. Debemos recordar que este está diseñado para escribir secuencias de bytes en bruto (eso fue comentado en el programa) , como datos de imagen.

_File archivos lo usamos como fichero , eso también fue comentado.

EventQueue

Es una clase independiente de la plataforma que pone en cola los eventos, tanto de las clases pares subyacentes como de las clases de aplicaciones de confianza. Este tiene un comportamiento particular ya que depende de la implementación.

Set

Tambien utilizamos una serie de ”set” después del “ public Inicio SIB” como setBackground setFont (seleccionamos el tipo de letra , color , tamaños) ,setIconImage(donde se ven los iconos) ,setType, setTitle(donde va el titulo como queramos) y setDefaultCloseOperation Además podemos decir que la interfaz Set establece estipulaciones adicionales, más allá de las heredadas de la interfaz, en los contratos de todos los constructores y en los contratos de los métodos add, equals y hashCode. Las declaraciones de otros métodos heredados también se incluyen aquí para mayor comodidad. (Las especificaciones que acompañan estas declaraciones se han adaptado a la interfaz del conjunto, pero no contienen ninguna estipulación adicional).Cabe recalcar que este uso ha sido frencuente en el programa.

JTextArea

Es un área de varias líneas que muestra texto sin formato. Está pensado para ser un componente ligero que brinde compatibilidad de origen con la java.awt.TextAreaclase en la que puede hacerlo razonablemente. Puede encontrar información y ejemplos sobre el uso de todos los componentes de texto en Uso de componentes de texto , una sección en el Tutorial de Java. Este componente tiene capacidades que no se encuentran en la java.awt.TextAreaclase. La superclase debe ser consultada para capacidades adicionales. Las clases de texto multilínea alternativas con más capacidades son JTextPaney JEditorPane.

Font

La Fontclase representa las fuentes, que se utilizan para representar texto de forma visible. Una fuente proporciona la información necesaria para asignar

secuencias de caracteres a secuencias de glifos y para representar secuencias de glifos Graphicsy Componentobjetos.

ActionEvent

Un evento semántico que indica que ocurrió una acción definida por el componente. Este evento de alto nivel es generado por un componente (como a Button) cuando ocurre la acción específica del componente (como ser presionado). El evento se pasa a cada ActionListenerobjeto que se registró para recibir dichos eventos utilizando el addActionListenermétodo del componente .

Interfaz ActionListener

EmptyBorder

Una clase que proporciona un borde vacío y transparente que ocupa espacio pero no hace ningún dibujo.

MatteBorder

Una clase que proporciona un borde mate como un color sólido o un icono en mosaico.

DefaultTableModel

Se trata de una aplicación de TableModel que utiliza una Vector de Vectores para almacenar los objetos de valor célula.

JScrollPane

Esta clase pública proporciona una vista desplazable de un componente ligero. A JScrollPane administra una ventana gráfica, barras de desplazamiento horizontales y verticales opcionales y ventanas gráficas opcionales de encabezado de fila y columna. Puede encontrar documentación orientada a tareas JScrollPane en Cómo utilizar los paneles de desplazamiento , una sección en el Tutorial de Java . Tenga en cuenta que JScrollPane no admite componentes de peso pesado.

JFileChooser

Este proporciona un mecanismo simple para que el usuario elija un archivo. Para obtener información sobre el uso de JFileChooser, consulte cómo usar los selectores de archivos, una sección en el Tutorial de Java.

El nuestro código muestra que funciona como un selector de archivos para el directorio de inicio del usuario.

JTextArea

Es un área de varias líneas que muestra texto sin formato. Está pensado para ser un componente ligero que brinde compatibilidad de origen con la java.awt.TextAreaclase en la que puede hacerlo razonablemente. Puede encontrar información y ejemplos sobre el uso de todos los componentes de texto en Uso de componentes de texto , una sección en el Tutorial de Java.

Este componente tiene capacidades que no se encuentran en la java.awt.TextAreaclase. La superclase debe ser consultada para capacidades adicionales. Las clases de texto multilínea alternativas con más capacidades son JTextPaney JEditorPane.

DefaultComboBoxModel

El modelo predeterminado para cuadros combinados.

LineBorder

Una clase que implementa un borde de línea de grosor arbitrario y de un solo color.

SystemColor

Una clase para encapsular colores simbólicos que representan el color de los objetos GUI nativos en un sistema. Para los sistemas que admiten la actualización dinámica de los colores del sistema (cuando el usuario cambia los colores), los valores RGB reales de estos colores simbólicos también cambiarán dinámicamente. Para poder comparar el valor RGB "actual" de un SystemColor objeto con un objeto de color no simbólico, se getRGB debe usar en lugar de equals.

Tenga en cuenta que la forma en que estos colores del sistema se aplican a los objetos de la GUI puede variar ligeramente de una plataforma a otra, ya que los objetos de la GUI se pueden representar de manera diferente en cada plataforma.

BufferedReader

Lee el texto de un flujo de entrada de caracteres, y almacena en búfer los caracteres para proporcionar una lectura eficiente de los caracteres, matrices y líneas.

Se puede especificar el tamaño del búfer o se puede usar el tamaño predeterminado. El valor predeterminado es lo suficientemente grande para la mayoría de los propósitos.

En general, cada solicitud de lectura hecha de un Reader hace que se realice una solicitud de lectura correspondiente del carácter subyacente o del flujo de bytes. Por lo tanto, es recomendable envolver un BufferedReader alrededor de cualquier Reader cuyas operaciones de lectura () puedan ser costosas, como FileReaders y InputStreamReaders.

IIOException

Una clase de excepción utilizada para señalar el fallo en tiempo de ejecución de las operaciones de lectura y escritura.

Además de una cadena de mensajes, se mantiene una referencia a otro Throwable(Error Exception). Esta referencia, si no es null, se refiere al evento que causó esta excepción. Por ejemplo, un IOException tiempo de lectura de un File se almacenaría allí.

ImageIcon

Una implementación de la interfaz de iconos que pinta iconos de imágenes. Las imágenes que se crean a partir de una URL, un nombre de archivo o una matriz

de bytes se cargan previamente usando MediaTracker para monitorear el estado cargado de la imagen.

Toolkit

Esta clase es la superclase abstracta de todas las implementaciones reales del kit de herramientas Abstract Window. Las subclases de la Toolkitclase se utilizan para vincular los diversos componentes a implementaciones de herramientas nativas particulares.

Interfaz SwingConstantes

Color

Se utiliza para encapsular colores en el espacio de color predeterminado o colores en espacios de color arbitrarios identificados por a “ ColorSpace” . Cada color tiene un valor alfa implícito de 1.0 o uno explícito proporcionado en el constructor. El valor alfa define la transparencia de un color y se puede representar mediante un valor flotante en el rango de 0.0 - 1.0 o 0 - 255. Un valor alfa de 1.0 o 255 significa que el color es completamente opaco y un valor alfa de 0 o 0.0 Significa que el color es completamente transparente.

Proyecto en eclipse

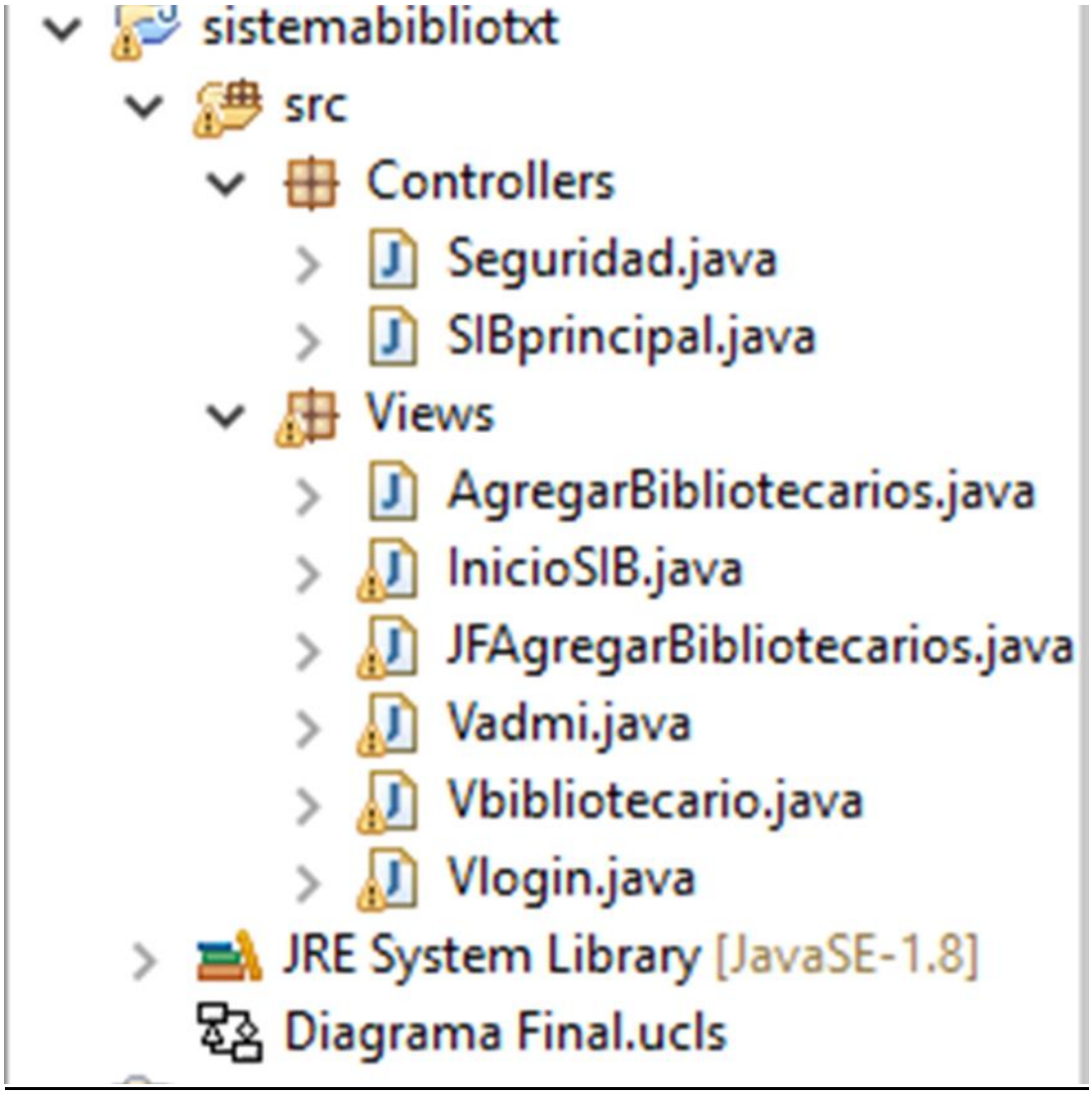
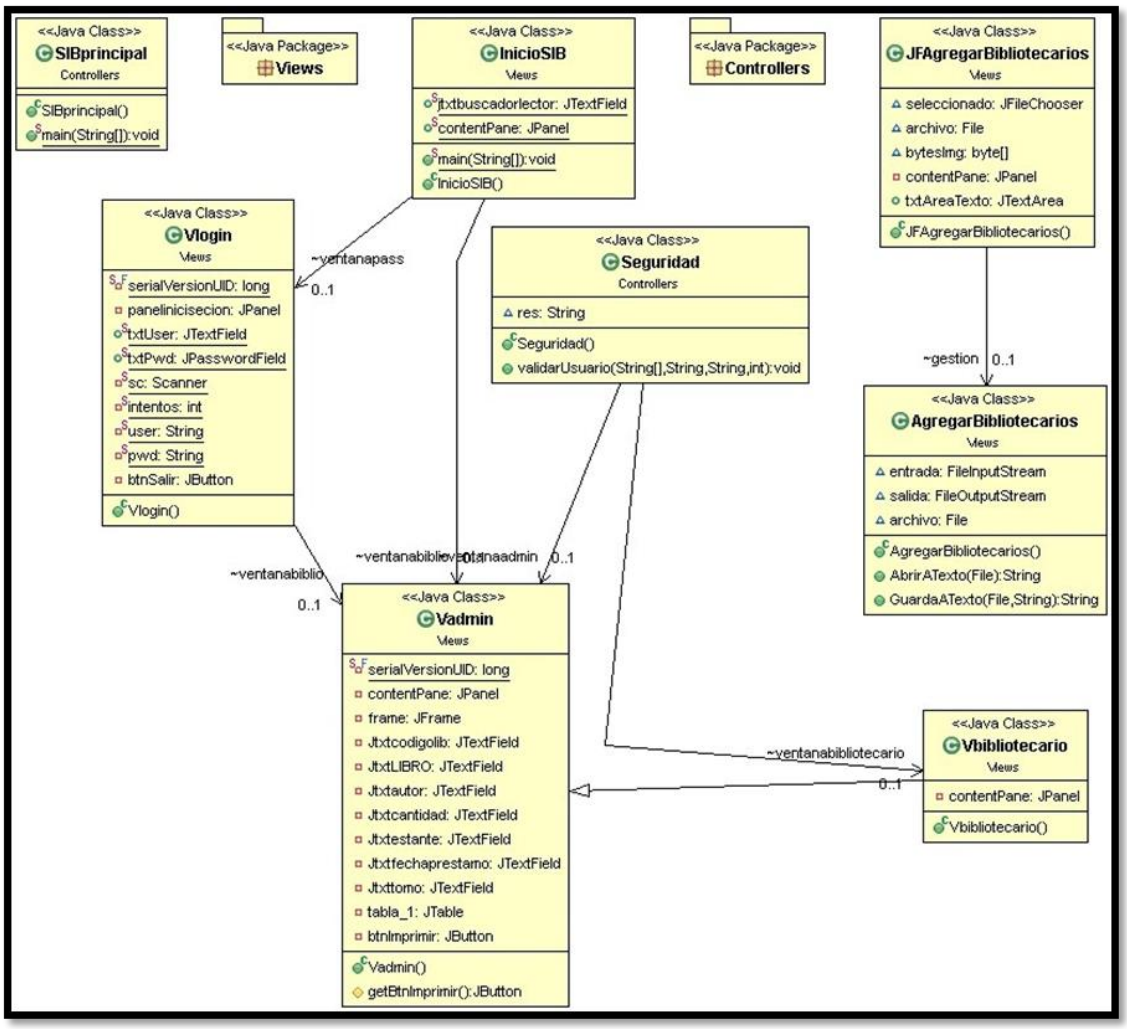


Diagrama de clases



Siguiendo el paradigma de la Programación Orientada a objetos (POO), en el proyecto de Sistema de Gestión de bibliotecas, aplicamos los conceptos de Modularidad, Encapsulamiento, alta cohesión y bajo acoplamiento, polimorfismo y herencia. Este último se puede observar en el diagrama de clases, donde la clase Bibliotecario (Vbibliotecario) hereda los atributos y métodos de la clase Administrador (Vadmin); sin embargo, no hereda el método de Gestionar Bibliotecarios (Agregar bibliotecarios, específicamente), de esta manera tendríamos una clase con mayor jerarquía que la otra, restringiendo algunos métodos que pueden tener. Así mismo, se dividió el proyecto en dos paquetes:

Controllers y Views, este último con todas las interfaz gráficas necesarias; y el primero, siguiendo la lógica de la aplicación. También se encapsulo algunos métodos con el modificador de visibilidad protected, de manera que la subclase Bibliotecario, herede los métodos de la superclase Administrador, excepto un método, que fue encapsulada con el modificador de visibilidad private para que pueda ser modificada solamente por su propia clase. En el proceso de la creación del programa se pudo ir implementando más concepto de Programación Orientada a Objetos; sin embargo, por el tiempo requerido se obvio algunos de estos, ya que buscábamos una aplicación, lo más sencilla e interactiva posible, de manera tal que la gestión de préstamo de libros sea mucho más eficiente y poco engorrosa.
El Proyecto ha sido separado en dos packages para seguir un adecuado modelamiento, y también tenemos un package secundario para los íconos de nuestros botones y bordes.

El primero será **Controllers** donde encontraremos las clases de **seguridad** y **SIBprincipal**:

- **CLASE SEGURIDAD**: será la clase exclusiva para comprobar y comparar el texto ingresado con la contraseña real con código que nos diga si esta contraseña ingresada es correcta o no a través de un boolean; debido a que cada bibliotecario y el administrador tiene diferente usuario y contraseña.

- **CLASE DE SIBPRINCIPAL:** Esta clase es la que se encargará de la inicialización del proyecto.

Como segundo package tenemos **VIEWS;** en este podemos encontrar las clases donde encontraremos las diferentes funcionalidades del sistema de biblioteca.

- **CLASE INICIOSIB:** en esta clase encontraremos JtextField y el JPanel, que es la primera interfaz del sistema de biblioteca y podamos acceder a las opciones que nos brinda la aplicación.

- **CLASE JFAGREGARBIBLIOTECARIOS:** Encontraremos los códigos donde solo el administrador podrá tener acceso para después pasar a la clase agregar bibliotecarios.

- **CLASE AGREGAR BIBLIOTECARIOS:** Esta clase fue implementada para buscar y guardar archivos con los cambios generados, en este caso cuando agreguemos bibliotecarios con sus respectivos usuarios y contraseña.

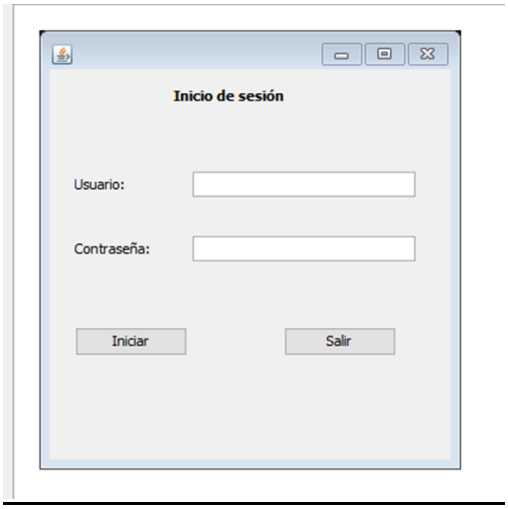
- **CLASE VLOGIN**: El login va servir para que iniciar sesión el administrador y bibliotecarios, con sus respectivos usuarios y contraseñas, que ya habían sido guardadas en un archivo txt.

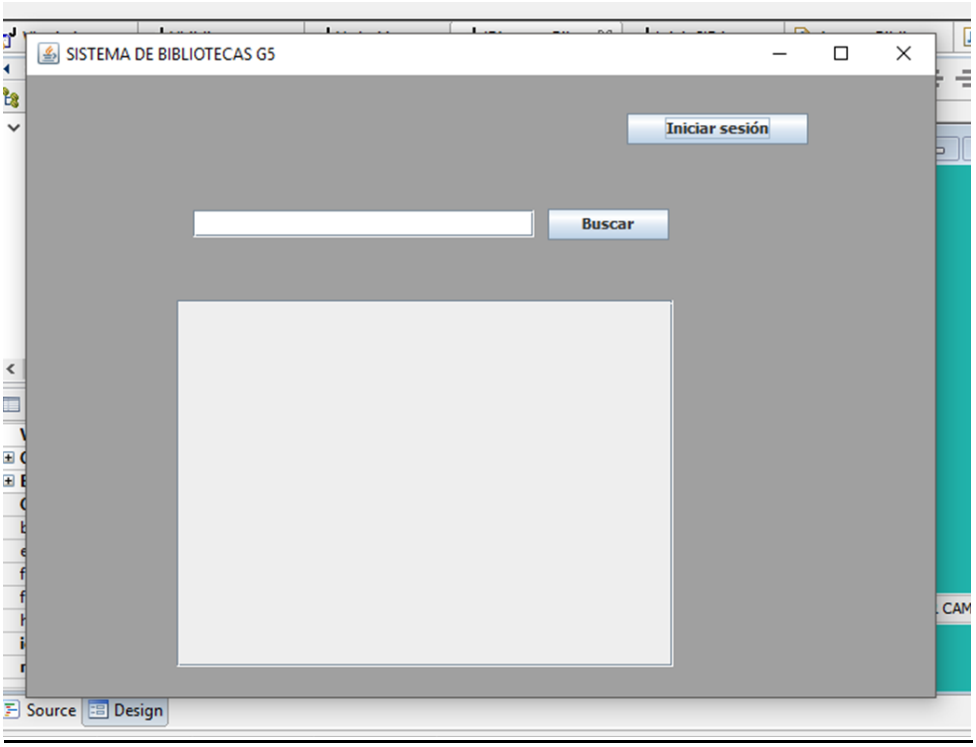
- **CLASEVADMIN:** Aquí encontramos los métodos que nos harán posible la realización de los métodos que contiene este, para la agregación de libros, gestión de bibliotecarios,etc.
  En esta clase encontraremos la línea de códigos donde estarán las interfacez de botones, tabla, y texto añadido. Interfacez usadas en esta clase son: JPanel, JFrame, JTextField, JTable, JButton.

- **CLASE VBIBLIOTECARIO.** El bibliotecario hereda los métodos del administrador, excepto el método de gestionar bibliotecario, que es exclusivo del administrador.
  Es la línea de código que tendrá acceso el bibliotecario, donde solo podrá agregar libros y ver la situación del préstamo de libros.

Funcionamiento

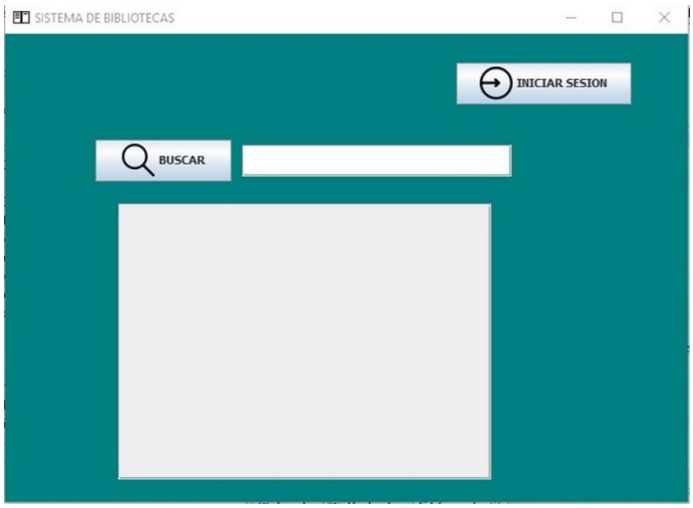
En sus inicios antes de las modificaciones.

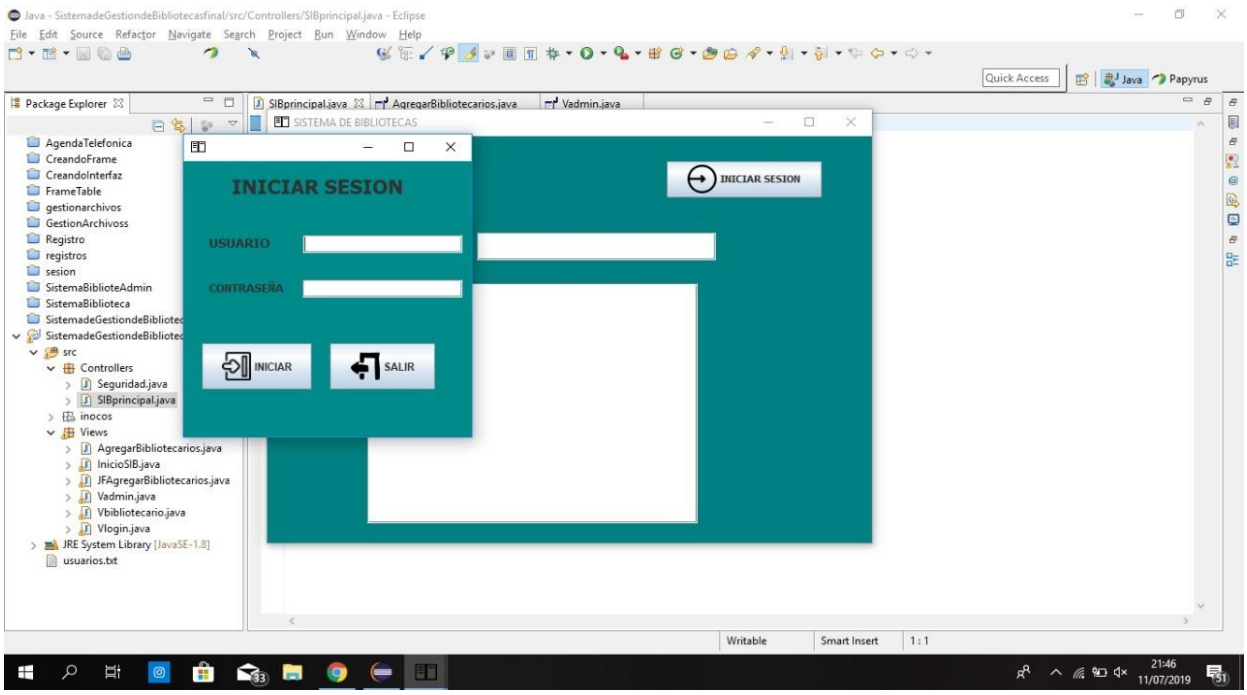




_A medida que se iba avanzando el trabajo se realizo ciertas modificaciones que perfeccionaron el sistema.

_En este caso tenemos el panel de sistema de biblioteca donde ya se puede iniciar sesión y hacer uso del buscador gracias también a JButton.





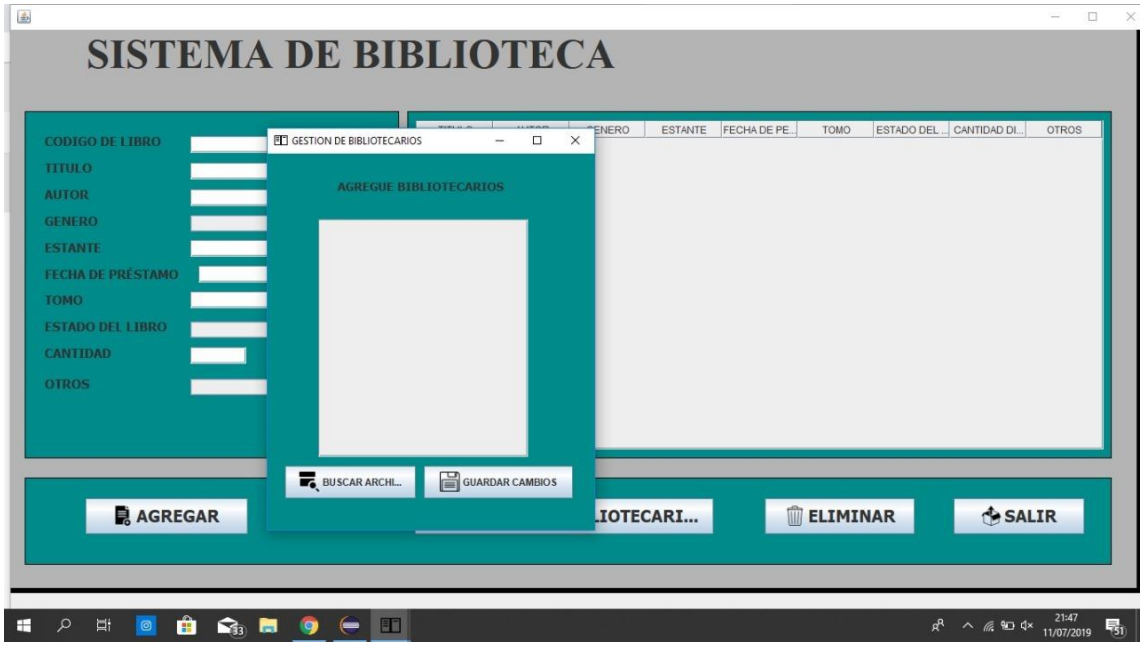
_Acá se indica el nombre de usuario y la contraseña programado con la cual se puede tener acceso al sistema , en nuestro programa tanto el usuario , jefe administrador y bibliotecario están registrados diferentes y cada uno tiene diferente funciones en el sistema.

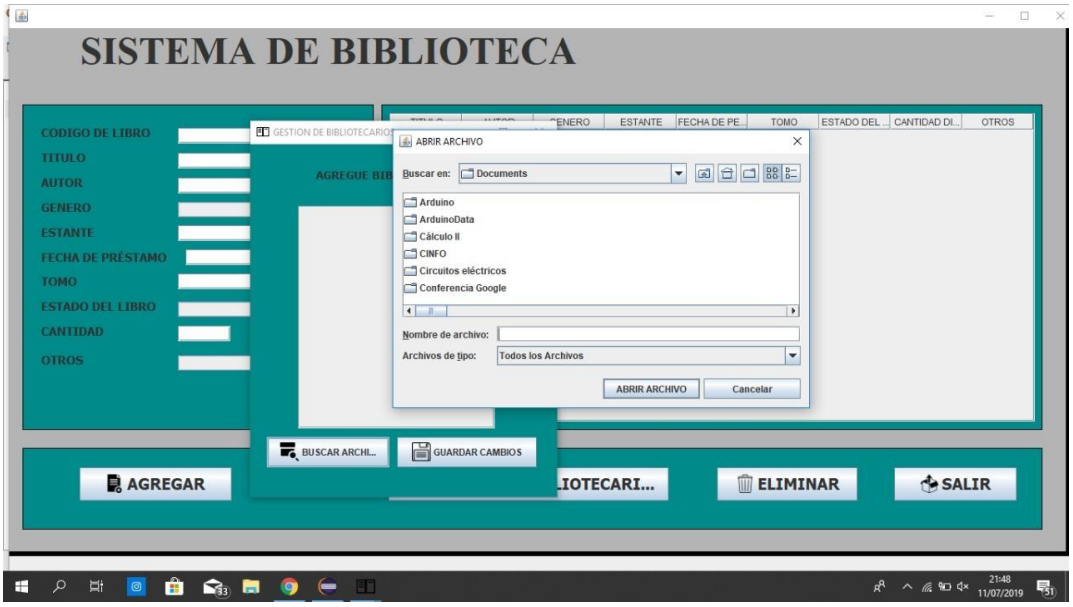
_De esta manera debe quedar cuando se haya iniciado sesión , claro que depende de quien ingrese habrán unos pequeños cambios en los botones de “gestionar bibliotecario”, etc.

Uso normal

Uso para el administrador.

Gestión de bibliotecarios.





Uso para un usuario normal.