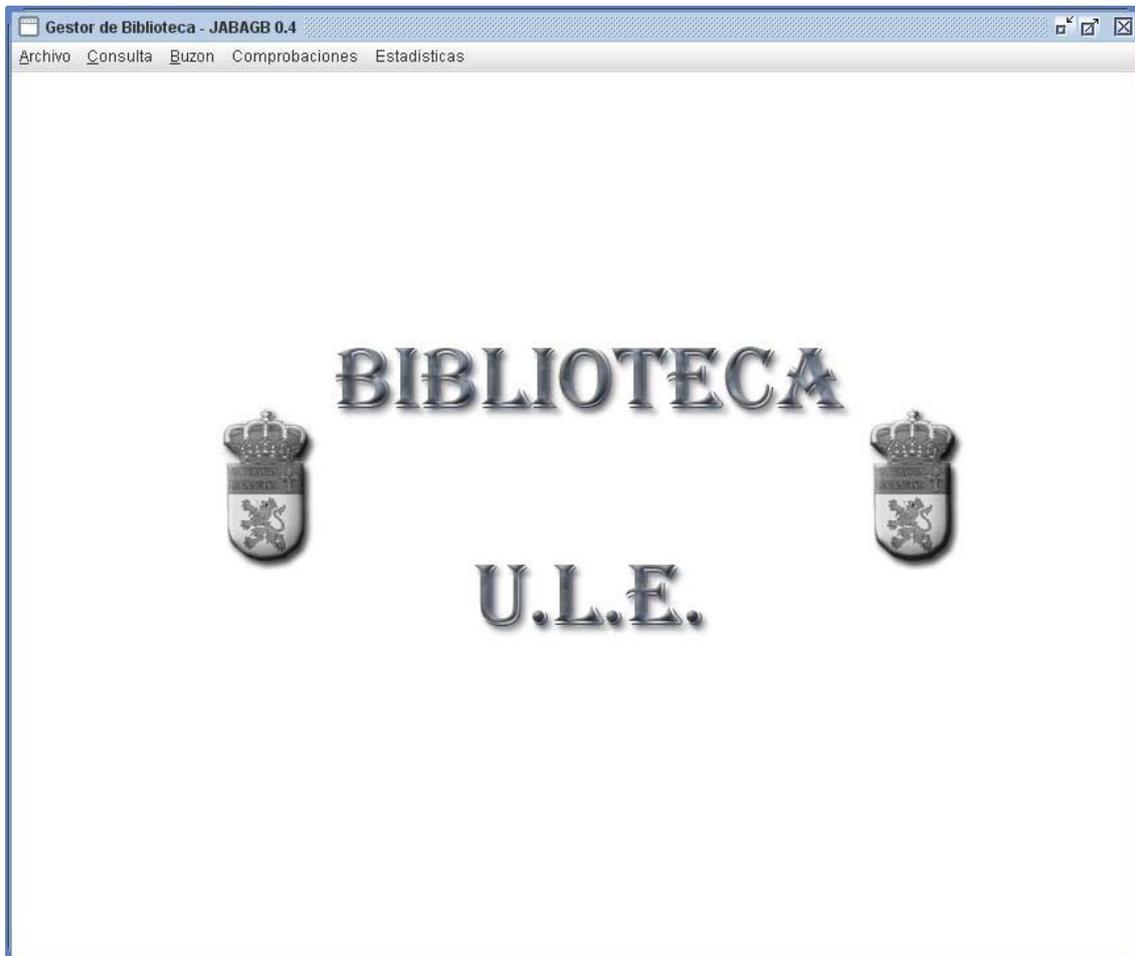


Documentación: Gestor de una Biblioteca (Eclipse, MySQL, JDK 1.6+)

Jose Alberto Benítez Andrades



Metodología y Tecnología de la Programación

2º Ingeniería Informática, NIF:71454586A

Infjab02@estudiantes.unileon.es

ÍNDICE

PRÓLOGO.

1. ESPECIFICACIÓN DE LA PRÁCTICA.

1.1. CLASES DE OBJETOS

2. ARQUITECTURA DE LA APLICACIÓN.

2.1. DISEÑO DE LA BASE DE DATOS.

2.1.1. Tabla Roles.

2.1.2. Tabla Titulaciones.

2.1.3. Tabla Usuarios.

2.1.4. Tabla Libros.

2.1.5. Tabla Revistas.

2.1.6. Tabla Prestamos.

2.1.7. Tabla Reservas.

2.1.8. Tablas Buzon y Buzon2.

2.1.9. Diagrama Entidad-Relación.

3. FUNCIONAMIENTO DEL PROGRAMA.

PRÓLOGO

Esta segunda práctica pretende servir de ejemplo como aplicación de la Programación Orientada a Objetos. Decimos que estamos realizando POO cuando utilizamos clases, objetos y empleamos la herencia.

La herencia es un mecanismo para construir clases derivadas de otras, de forma que esta nueva clase tiene todas las características de la clase padre (atributos y métodos) y permite redefinirlas así como añadir otras nuevas. Con ello conseguimos, entre otras cosas, facilitar la construcción de TAD y permitir la reutilización de código.

A continuación iremos viendo como estos conceptos se aplican en el programa “Gestor de una Biblioteca”.

1. ESPECIFICACIÓN DE LA PRÁCTICA

El segundo programa consistirá en la construcción de una biblioteca orientada a objetos para una biblioteca. La biblioteca se probará con el desarrollo de las simulaciones de préstamos, inserción de usuarios, inserción de fondos...

1.1. CLASES DE OBJETOS

Estas son las clases de objetos que he utilizado para realizar el gestor de la biblioteca:

- ***controlarLongitud*** : Clase mediante la cual controlo la longitud de los ***JTextField*** y si son numéricos o alfanuméricos.
- ***MiLogin*** : En ella creo un objeto ventana que pide al usuario los datos para su autenticación en la biblioteca, Usuario y Contraseña, validándolo mediante búsquedas en la base de datos.
- ***MiMysql*** : Se encuentran todas las consultas SQL que necesito para gestionar la biblioteca. Por ejemplo a la hora de consultar fondos en una biblioteca, aquí están los métodos que nos devuelven la tabla para mostrar en pantalla a los usuarios.

- **ModeloTablaBuzon** : Encargada de devolvernos la tabla con los mensajes que tenga un usuario en su Buzón.
- **RolSuperior** : Es utilizada como puente de acceso entre **MiMysql** y otras clases, es una clase de ayuda auxiliar.
- **VentanaBuzon** : Es la encargada de devolvernos el objeto en el cual veremos nuestro buzón de mensajes.
- **VentanaConsultarFondo** : Muestra la ventana en la cual realizaremos nuestras consultas de Fondos dentro de nuestra biblioteca.
- **VentanaConsultarPrestamo** : Al igual que la anterior, sólo que esta vez para mostrar los Prestamos que queremos buscar.
- **VentanaConsultarUsuario** Igual que las dos anteriores, pero mostrando los Usuarios.
- **VentanaEstadisticas** : Muestra la ventana donde realizaremos nuestro cálculo de estadísticas de la biblioteca.
- **VentanaInsertarFondo** : Interfaz mediante la cual insertaremos **Libros** o **Revistas** en nuestra biblioteca.
- **VentanaInsertarUsuario** : Interfaz mediante la cual insertaremos **Usuarios** en nuestra biblioteca.
- **VentanaMensajeEntrada** : Encargada de mostrar los mensajes que recibimos en nuestra bandeja de entrada.
- **VentanaMensajeSalida** : Muestra la ventana de redacción de un mensaje para enviar a un usuario.
- **VentanaPrincipal**: Muestra la ventana principal del programa con su barra superior de menú correspondiente dependiendo del nivel del usuario.

El conjunto de operaciones para un **objeto de la clase MiLogin** es:

- **public** MiLogin()
 - Constructor de la clase.
- **protected** JComponent createButtonPanel()
 - Crea el panel de botones para la ventana de MiLogin.
- **private static void** createAndShowGUI()
 - Encargado de mostrar la parte gráfica en pantalla.
- **public static void** main(String[] args) **throws** Exception
 - Función principal del programa.

Para la **clase de objeto MiMysql**:

- **public** MiMysql()
 - Función principal del programa.
- **public static void** ConectaDB() **throws** Exception
 - Encargado de conectar con la base de datos.
- • **public static void** LlamadaDB(String llamada)
 - Encargado de hacer llamadas a la base de datos sin obtener resultados, por ejemplo "UPDATE", "DELETE", "INSERT" ...
- **public static void** EntraDB()
 - Función que conecta con la base de datos **JABAGB**.
- **public static** String Login(String DNI,**boolean** Ejecuta)
 - Se encarga de verificar el usuario y la contraseña del usuario cuando se autentifican.
- **public static** DefaultTableModel ConsultaUsuarioDB(String llamada,String nivel)
 - Mediante esta función recibiremos una tabla con los datos de los usuarios que hemos buscado previamente.
- • **public static** DefaultTableModel ConsultaFondoDB(String llamada,**boolean** libro)

- Con ella, los administradores y técnicos recibirán la tabla de Fondos que han buscado, pudiendo modificar los datos que deseen sobre ella.
- **public static** NonEditableTableModel ConsultaFondoDBLector(String Llamada, **boolean** libro)
 - Su función es igual al procedimiento anterior, sólo que la tabla que devuelve está restringida para que los usuarios no puedan modificar la tabla.
- **public static** ModeloTablaBuzon ConsultaMensajesBuzon(String queBuzon,String Llamada,**boolean** tecnico)
 - Devuelve los mensajes, tanto enviados como recibidos, del usuario que está autenticado en el gestor.
- **public static** DefaultTableModel ConsultaPrestamo(String Llamada)
 - Devuelve la tabla de préstamos de un usuario concreto, o de diversos volúmenes, según haya sido la búsqueda del **Administrador o Técnico**.
- **public static** DefaultTableModel ConsultaPrestamoLector(String Llamada)
 - Función equitativa a la anterior, con la salvedad de que es solamente para que cada lector visualice sus préstamos.
- **public static** String[] RecogeMensaje(String idmensaje)
 - Encargado de devolver un mensaje concreto para poder visualizarse en pantalla.
- **public static** String[] DameTitulaciones()
 - Función encargada de enviar las titulaciones que están en la tabla titulaciones.
- **public int** ultimaID()
 - Devuelve la última id añadida a una tabla de la base de datos.
- **public** String dame_fecha(String actual,**boolean** corto)
 - Función que utilizo para conocer la fecha actual o para conocer la fecha actual sumándole 2 días o 21 días, dependiendo del tipo de préstamo.
- **public int** resta_fecha(String fecha,**boolean** corto)
 - Función auxiliar que devuelve la fecha restada a un intervalo de días definido por el programador.

- **public int** getNumCopiasFondo(String llamada)
 - Función que fue diseñada para obtener el número de copias de un fondo, y que utilizo a lo largo del programa para obtener más números de selecciones que hago en la base de datos.
- **public** String getCadena(String llamada)
 - Función que recibe una cadena en una búsqueda en la base de datos.
- **public int** getNumPrestamosLector(String idusuario)
 - Devuelve el número de préstamos que posee un lector en su poder.
- **public int** devolverConsultas()
 - Encargada de hacer las devoluciones de las consultas al abrirse el programa cada día.
- **public void** comprobarCaducados()
 - Gestor de fondos que caducan encargado de avisar a los usuarios de la finalización de uno de sus préstamos.
- **public void** comprobarReservas()
 - Encargado de borrar las reservas caducadas.
- **public static** JPanel comprobarReservasLector(String id)
 - Avisa al usuario de si tiene reservas disponibles para recoger prestadas.

Para la **clase de objetos RolSuperior**:

- **public** RolSuperior()
 - Constructor de la clase.
- **protected int** setDatosUsuarios(String datos[])
 - Método encargado de insertar un usuario en la base de datos cuando el técnico o administrador lo ordena.
- **protected void** setDatosFondos(String datos[],**boolean** esLibro)
 - Método que se encarga de insertar un fondo (libro o revista) en la base de datos, cuando el técnico o administrador lo ordena.
- **protected** DefaultTableModel getTablaBusquedaUsuario(String busqueda, String criterio,String nivel)
 - Método que devuelve una tabla cuando se buscan usuarios.

- **protected** DefaultTableModel getTablaBusquedaFondo(String busqueda, String criterio, **boolean** esLibro)
 - Método que devuelve el resultado de una búsqueda de libros.
- **protected** NonEditableTableModel getTablaBusquedaFondoLector(String busqueda, String criterio, **boolean** esLibro)
 - Posee la misma función que la anterior, sólo que esta vez el resultado es para un lector en lugar de para un administrador.
- **protected** String[] getDatosMensaje(String idmensaje)
 - Devuelve los datos de un mensaje (remitente, asunto, texto del mensaje..).
- **protected** String[] getTitulaciones()
 - Devuelve las titulaciones que hay en ese momento.
- **static public void** comprobacionesIniciales()
 - Realiza las comprobaciones iniciales de caducados, reservas...

Para la **clase de objetos VentanaBuzon**:

- **public** VentanaBuzon(String NSOCIO, String Nivel)
 - Constructor de la ventana que muestra el buzón de mensajes.
- **class** SharedModelDemo **extends** JPanel
 - Clase interna que devuelve un JPanel para mostrar en pantalla.
 - **protected** JComponent createToolBar()
 - Crea la barra de herramientas superior de nuestro buzón.
 - **protected** Imagen crearImagen(String path)
 - Crea los iconos para los botones que creo, es una función auxiliar que está en más clases.
 - **public void** actionPerformed(ActionEvent e)
 - Desde el actionPerformed controlo todos los eventos de los botones, actualizaciones, modificaciones, etc...

Para la **clase de objetos** *VentanaConsultarFondo*:

- **public** VentanaConsultarFondo(String nivel,String NSocio)
 - Constructor de la clase.
- **class** ConsultorioFondo **extends** JPanel
 - Clase interna que devuelve el panel a mostrar por la ventana de consulta de fondo.

Para la **clase de objetos** *VentanaConsultarPrestamo*:

- **public** VentanaConsultarPrestamo(String nivel,String NSocio)
 - Constructor de la clase.
- **class** ConsultorioPrestamo **extends** JPanel
 - Clase interna que devuelve el panel a mostrar por la ventana de consulta de préstamo.

Para la **clase de objetos** *VentanaConsultarUsuario*:

- **public** VentanaConsultarUsuario(String nivel)
 - Constructor de la clase.
- **class** ConsultorioFondo **extends** JPanel
 - Clase interna que devuelve el panel a mostrar por la ventana de consulta de usuarios.

Para la **clase de objetos** *VentanaEstadísticas*:

- **public** VentanaEstadísticas(String nivel,String NSocio)
 - Constructor de la clase.
- **class** ConsultorioFondo **extends** JPanel
 - Clase interna que devuelve el panel a mostrar por la ventana de consulta de estadísticas, con el resultado hallado.

Para la **clase de objetos** *VentanaInsertarFondo*:

- **public** VentanaConsultarFondo()
 - Constructor de la clase.
- **class** FormularioInsertarFondo **extends** JPanel
 - Clase interna que devuelve el panel a mostrar por la ventana de inserción de un fondo.

Para la **clase de objetos** *VentanaInsertarUsuario*:

- **public** VentanaConsultarUsuario ()
 - Constructor de la clase.
- **class** FormularioInsertarUsuario **extends** JPanel
 - Clase interna que devuelve el panel a mostrar por la ventana de inserción de un usuario.

Para la **clase de objetos** *VentanaMensajeEntrada*:

- **public** VentanaMensajeEntrada (String idmensaje,**boolean** esprestamo)
 - Constructor de la clase.
- **class** MensajeUsuario **extends** JPanel
 - Clase interna que devuelve el panel a mostrar por la ventana de mensaje de entrada.

Para la **clase de objetos** *VentanaMensajeSalida*:

- **public** VentanaMensajeSalida()
 - Constructor de la clase.
- **class** MensajeUsuario **extends** JPanel
 - Clase interna que devuelve el panel a mostrar por la ventana de mensaje de salida.

Para la **clase de objetos** *VentanaPrincipal*:

- **public class** *VentanaPrincipal* **extends** *JFrame*
 - Constructor de la clase.
- **class** *MainDesktopPane* **extends** *JDesktopPane*
 - Clase interna mediante la cual creo un *DesktopPane* con un fondo asignado por mí.

2.ARQUITECTURA DE LA APLICACIÓN.

La elaboración de una práctica mediana conlleva una buena fase de análisis del problema y diseño del programa, ya que si no seguimos bien las especificaciones del programa, podemos crear una aplicación que no hace lo que el cliente nos pide.

Otro problema muy frecuente, es la mala elección de unos tipos de datos y una mala codificación del programa, ya que eso conllevaría a un programa ineficiencia, ya sea porque posee muchas líneas de código para hacer algo sencillo, o simplemente porque a la hora de la ejecución, el programa funciona muy lentamente y con errores.

En este apartado voy a explicar el diseño de la base de datos que he utilizado para hacer el programa, y también desglosaré el funcionamiento de las unidades y del programa principal.

2.1. DISEÑO DE LA BASE DE DATOS.

Según las especificaciones del programa, debemos almacenar los siguientes datos en la base de datos: ***Usuarios, Libros, Revistas, Prestamos y Reservas***. También necesitaremos almacenar mensajes en alguna parte de la base de datos, así que las tablas que he necesitado para realizar el gestor son las siguientes: ***Roles, Titulaciones, Usuarios, Libros, Revistas, Prestamos, Reservas, Buzon y Buzon2***.

2.1.1. Tabla Roles.

La tabla *Roles* es en la que almacenaremos los niveles que existen entre usuarios, que en principio son 4, “Administrador”, “Tecnico”, “Lector-Socio” y “Lector-Basico”.

Campos:

- **ID:** Almacenaremos un id primario para cada ROL existente.
- **ROL:** Nombre para cada nivel.

2.1.2. Tabla Titulaciones.

La tabla *Titulaciones* es en la que almacenaremos las posibles titulaciones que pueden tener los usuarios. Campos:

- **Titulo:** Almacenamos el nombre de la carrera que estudié el lector.

2.1.3. Tabla Usuarios.

La tabla *Usuarios* es en la que almacenaremos los usuarios y sus respectivos datos, está formada por los siguientes campos:

- **Nombre:** En él almacenaremos el nombre de cada usuario.
- **Apellidos:** Almacenaremos los apellidos del usuario.
- **Telefono:** Se guardará el teléfono de cada usuario.
- **Dirección:** Almacenaremos también la dirección dónde residen nuestros usuarios.
- **Titulacion:** Guardaremos también la información de qué carrera estudia nuestro lector.
- **NSocio:** Id por la cual identificaremos a cada usuario en nuestro gestor, es la clave primaria de la tabla Usuarios.
- **ROL:** Almacenaremos el nivel que posee el usuario, "*Administrador*", "*Técnico*", "*Lector-Socio*" o "*Lector-Basico*".
- **DNI:** El DNI del lector también se almacenará.
- **Password:** Y una contraseña para identificarse en el gestor.

2.1.4. Tabla Libros.

La tabla *Libros* es en la que almacenaremos todos los libros de la biblioteca. Los campos son los siguientes:

- **ID:** Una id primaria para cada libro.
- **Título:** Título de cada libro.
- **Autores:** Autores de cada libro.
- **Editorial:** Qué empresa editó ese libro.
- **Edición:** Qué edición llevan ya del libro.
- **Año:** Año de publicación de cada libro.
- **Páginas:** Cuántas páginas posee.
- **Signatura:** A qué signatura pertenece.
- **ISBN:** El código ISBN que posee.
- **CopiasSala:** Número de copias que hay para consultas.
- **CopiasPrestamo:** Número de copias que hay para préstamo.
- **PrestadasSala:** Número de libros prestados actualmente para consultas.
- **PresadasPrestamo:** Número de libros prestados para préstamo actualmente.
- **CortoPlazo:** Almacena si es de corto plazo (2 días) o no (21 días).

2.1.5. Tabla Revistas.

La tabla *Libros* es en la que almacenaremos todos los libros de la biblioteca. Los campos son los siguientes:

- **ID:** Una id primaria para cada revista.
- **Título:** Título de cada revista.
- **Autores:** Autores de cada revista.
- **Editorial:** Qué empresa editó ese revista.
- **Volumen:** Nombre de volumen asignado a esa revista.
- **VolumenCreado:** Variable booleana que indica si esa revista está asignada algún volumen o está suelta.
- **VolumenReal:** Nombre del volumen real de ese libro, si está asignado a alguno..
- **Número:** Año de publicación de cada revista.
- **Año:** Año de publicación de cada revista.
- **Páginas:** Cuántas páginas posee.

- **Signatura:** A qué signatura pertenece.
- **CopiasSala:** Número de copias que hay para consultas.
- **CopiasPrestamo:** Número de copias que hay para préstamo.
- **PrestadasSala:** Número de revista prestados actualmente para consultas.
- **PresadasPrestamo:** Número de l revista prestados para préstamo actualmente.

2.1.6. Tabla Prestamos.

La tabla *Prestamos* es en la que almacenaremos todos los préstamos de la biblioteca.

Los campos son los siguientes:

- **Id_prestamo:** Una id primaria para cada préstamo.
- **Id_revista:** Si el préstamo es de una revista, almacenamos la id de la revista, sino el campo será null.
- **Id_libro:** Si el préstamo es de un libro, almacenamos la id del libro, sino, almacenaremos null.
- **NSocio:** Id del socio que realiza ese préstamo.
- **EsLibro :** Booleano que indica si es un libro o una revista lo que prestamos.
- **Fecha_inicio:** Fecha en la que se realiza el préstamo.
- **Fecha_fin:** Fecha en la que deben devolver el préstamo.
- **Fecha_devuelto:** Fecha en la que devuelven el préstamo.
- **Devuelto:** Booleano que indica si el libro ya está devuelto.
- **Consulta:** Booleano que indica si el libro es de Sala o de Prestamo.
- **AvisoEnviado:** Booleano que nos indica si ya le hemos enviado un mensaje al usuario indicando el vencimiento de su libro.

2.1.7. Tabla Reservas.

La tabla *Prestamos* es en la que almacenaremos todos los préstamos de la biblioteca.

Los campos son los siguientes:

- **Id_reserva:** Una id primaria para cada reserva.
- **Id_revista:** Si el préstamo es de una revista, almacenamos la id de la revista, sino el campo será null.
- **Id_libro:** Si el préstamo es de un libro, almacenamos la id del libro, sino, almacenaremos null.
- **NSocio:** Id del socio que realiza ese préstamo.
- **EsLibro :** Booleano que indica si es un libro o una revista lo que prestamos.
- **Puede:** Booleano que nos dice si la reserva ya está disponible para convertirse en préstamo.
- **Fecha_reserva:** Fecha en la que se realiza el préstamo.
- **Fecha_puede:** Fecha en la que se puede coger prestado ya ese libro de esa reserva.

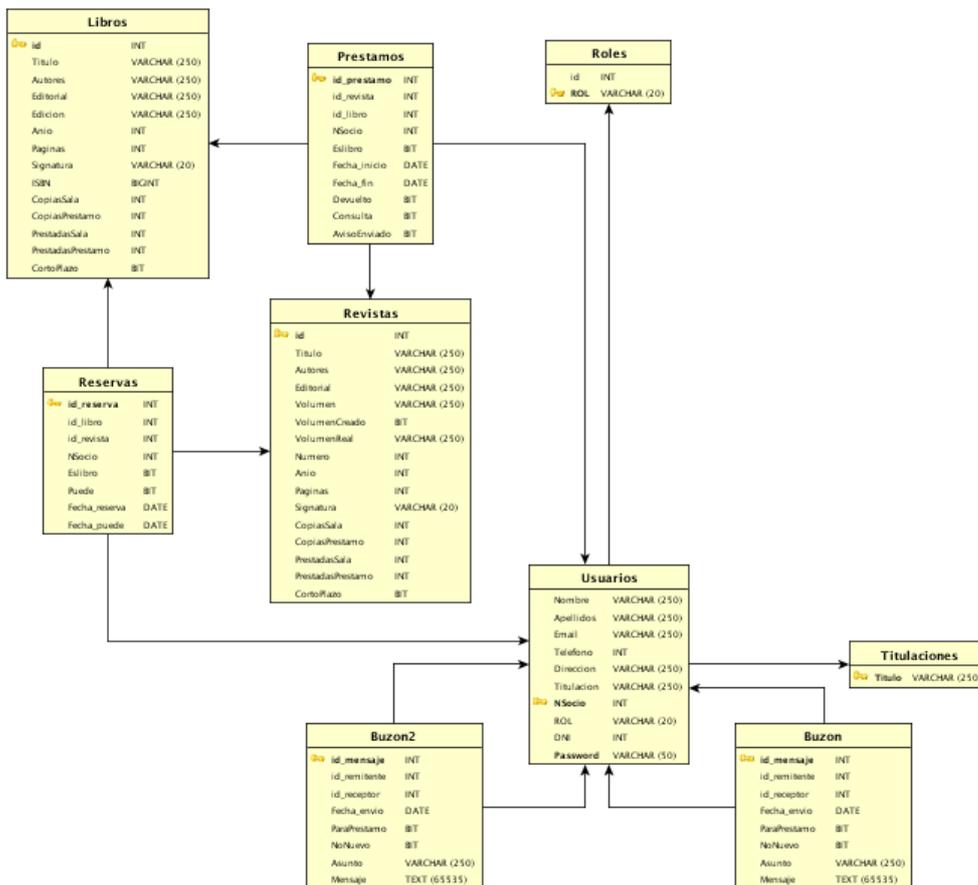
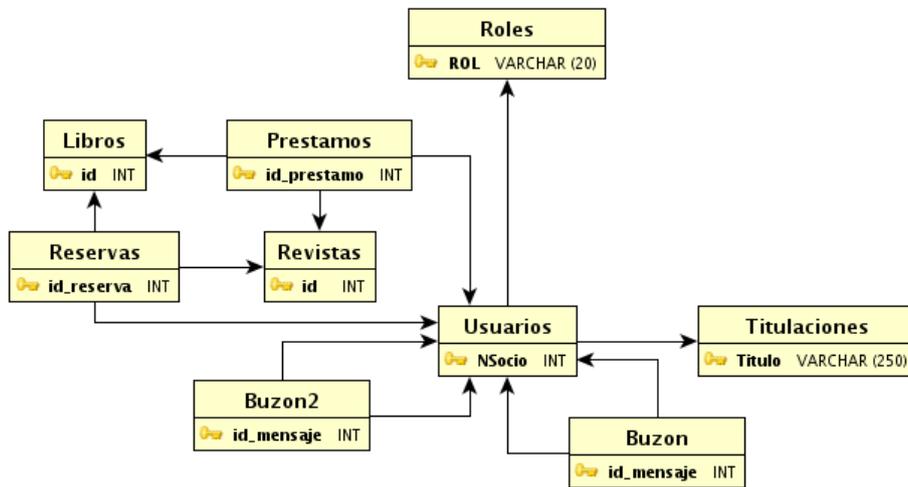
2.1.8. Tablas Buzon y Buzon2

Las tablas *Buzon* y *Buzon2* almacenan los mensajes de los usuarios, usamos 2 porque a la hora de borrar un mensaje, si un usuario borra un mensaje de entrada, el usuario que envió ese mensaje también tendría borrado su mensaje de la carpeta enviados, para que no haya conflictos se crean estas dos tablas que poseen los mismos campos:

- **Id_mensaje:** id primaria para cada mensaje enviado:
- **Id_remitente:** id del remitente del mensaje.
- **Id_receptor:** id del receptor del mensaje.
- **Fecha_envio:** fecha en la que fue enviado el mensaje.
- **ParaPrestamo:** Booleano que indica si el mensaje es un préstamo o es un mensaje normal.
- **NoNuevo:** Booleano que indica si el mensaje es nuevo o no.
- **Asunto:** Almacena el asunto del mensaje.
- **Mensaje:** Almacena el texto del mensaje enviado.

2.1.9. DIAGRAMA ENTIDAD-RELACION.

Para que se vean bien las referencias, voy a poner una foto en la que sólo se muestran las claves primarias de las tablas y los nombres de las mismas, y debajo la detallada:



3 .FUNCIONAMIENTO DEL PROGRAMA.

El programa comienza con una ventana de autenticación donde el usuario debe introducir su usuario y su contraseña, es decir, su número de socio y su contraseña.

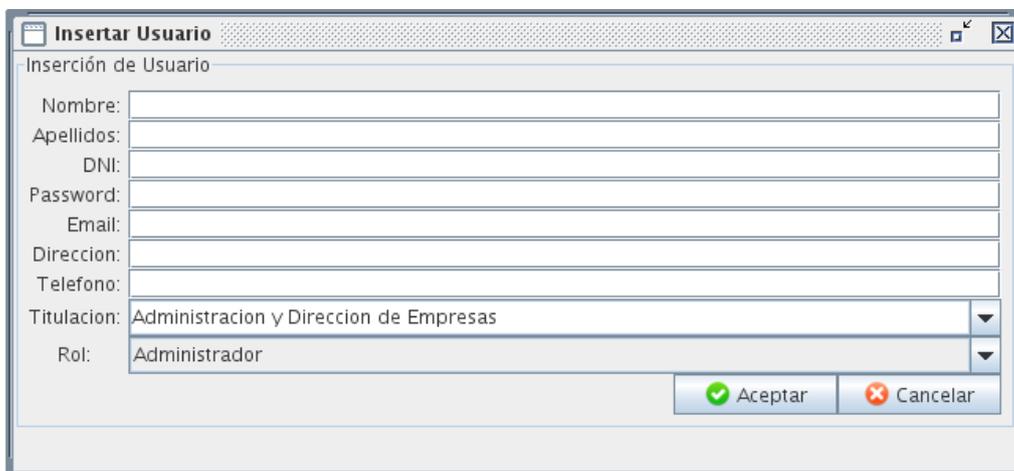


Una vez autenticado el usuario, se abre una ventana principal con todas las opciones en una barra de herramientas superior, esta barra de herramientas es distinta dependiendo el nivel que posea el usuario, es decir, si es **Administrador** o **Tecnico** tiene una serie de opciones que los **Lectores-Socios** y **Lectores-Basicos** no tienen.



Foto: Pantalla principal del gestor.

Una vez autenticado, si somos **Administradores** o **Técnicos** podremos insertar usuarios desde la siguiente interfaz de inserción:



Interfaz de inserción de usuario con los siguientes campos:

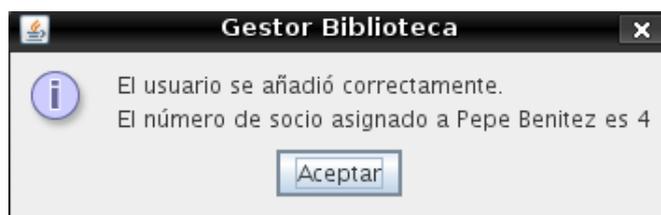
- Nombre:
- Apellidos:
- DNI:
- Password:
- Email:
- Dirección:
- Teléfono:
- Titulación: Administración y Dirección de Empresas
- Rol: Administrador

Botones:

Si falta algún dato por especificar nos dará un error al intentar insertar ese usuario:



De lo contrario, si se inserta bien el usuario, obtendremos un mensaje como el siguiente:



También pueden insertar fondos, y el mensaje de aceptación sería:



También siendo **Administradores o Tecnicos** tenemos la posibilidad de consultar usuarios, pudiendo modificar los datos que ya hemos insertado a la hora de introducirlos en la base de datos. La interfaz gráfica de consulta de usuarios es la siguiente:

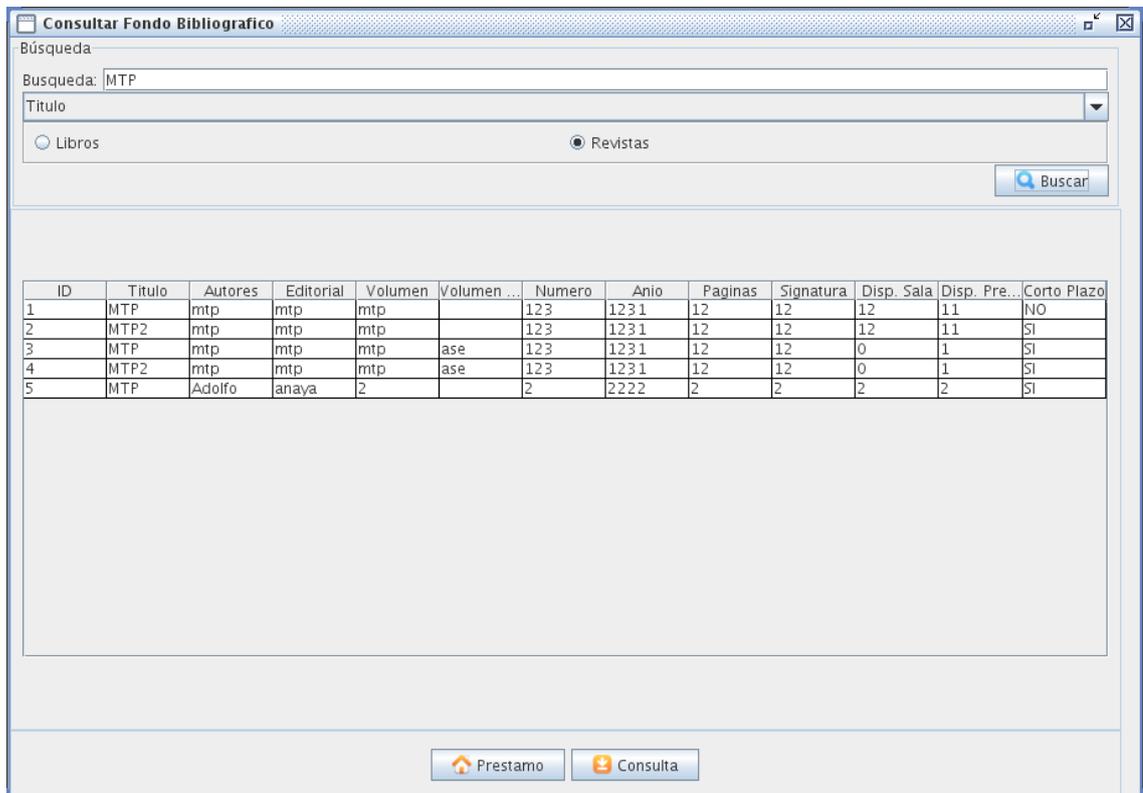
Num Socio	Nombre	Apellidos	DNI	Direccion	Email	Password	Telefono	Rol	Titulacion
1	Jose Alberto	Benítez Andra...	71454586	C/Moisés de L...	infjab02@est...	stop88	606733265	Administrador	Ingeniería Info...

Y la interfaz para consultas de libros para **Administradores y Tecnicos** es:

ID	Titulo	Autores	Editorial	Volumen	Volume...	Numero	Anio	Paginas	Signatura	Disp. Sala	Disp. Pr...	Prestad...	Prestad...	Corto Pl...
1	MTP	mtp	mtp	mtp		123	1231	12	12	12	11	0	0	NO
2	MTP2	mtp	mtp	mtp		123	1231	12	12	12	11	0	0	SI
3	MTP	mtp	mtp	mtp	ase	123	1231	12	12	0	1	0	0	SI
4	MTP2	mtp	mtp	mtp	ase	123	1231	12	12	0	1	0	0	SI
5	MTP	Adolfo	anaya	2		2	2222	2	2	2	2	0	0	SI

En ella se pueden modificar los datos de los libros/revistas y a su vez crear volúmenes de revistas nuevos.

Para los usuarios **Lectores-Basicos** y **Lectores-Socios** la interfaz de consulta de fondos es la siguiente:



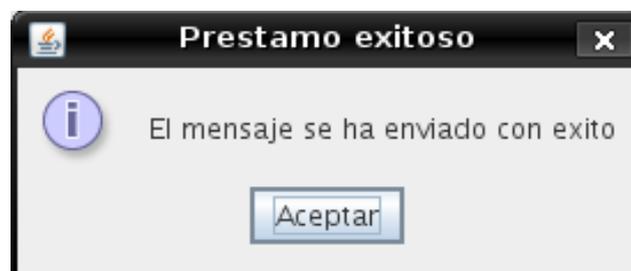
The screenshot shows a window titled "Consultar Fondo Bibliografico". It features a search bar with the text "MTP" and a dropdown menu for "Titulo". Below the search bar are radio buttons for "Libros" and "Revistas", with "Revistas" selected. A "Buscar" button is located to the right. Below the search interface is a table with the following data:

ID	Titulo	Autores	Editorial	Volumen	Volumen ...	Numero	Anio	Paginas	Signatura	Disp. Sala	Disp. Pre...	Corto Plazo
1	MTP	mtp	mtp	mtp		123	1231	12	12	12	11	NO
2	MTP2	mtp	mtp	mtp		123	1231	12	12	12	11	SI
3	MTP	mtp	mtp	mtp	ase	123	1231	12	12	0	1	SI
4	MTP2	mtp	mtp	mtp	ase	123	1231	12	12	0	1	SI
5	MTP	Adolfo	anaya	2		2	2222	2	2	2	2	SI

At the bottom of the window, there are two buttons: "Prestamo" and "Consulta".

Pueden realizar un préstamo pulsando el botón *Prestamo* o realizar una consulta pulsando *Sala*.

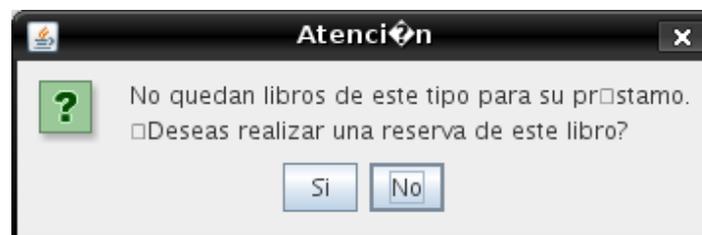
Cuando realizan un préstamo con éxito la ventana que aparece es la siguiente:



Si tuviese libros a devolver o fondos caducados, mostraría el siguiente dialogo:



Si el libro no se pudiese coger prestado en ese momento, mostraría el siguiente mensaje:



Permitiendo al usuario realizar una reserva de ese libro o revista. Si el usuario acepta la reserva, aparecería un mensaje similar al siguiente:



También tenemos un interfaz de consultas de **Préstamos** que es distinto para usuarios **Administradores o Técnicos** que para usuarios **Lectores**. Para los primeros, el interfaz es así:

Consultar Préstamos

Búsqueda:

Busqueda:

Todos

Todos Id Usuario A devolver

Buscar

Resultados

ID Prestamo	ID Libro	ID Revista	N Socio	Fecha inicio	Fecha fin	Devuelto	Consulta
1		2	6	2008-06-28	2008-06-30	SI	NO
2	1		6	2008-06-28	2008-06-30	SI	NO
3		2	6	2008-06-29	2008-07-01	SI	NO
4	1		6	2008-06-29	2008-07-01	SI	NO
5	1		6	2008-06-29	2008-07-01	SI	NO
6	1		6	2008-06-29	2008-07-01	SI	NO
7		1	6	2008-06-29	2008-07-20	SI	NO
8		2	6	2008-06-29	2008-07-01	SI	NO
9		1	6	2008-06-29	2008-07-20	SI	NO
10		2	6	2008-06-29	2008-07-01	SI	NO
11		2	6	2008-06-29	2008-07-01	SI	NO
12		2	6	2008-06-29	2008-07-01	SI	NO
13		2	6	2008-06-29	2008-07-01	SI	NO
14		2	6	2008-06-29	2008-07-01	SI	NO
15		2	6	2008-06-29	2008-07-01	SI	NO
16		2	6	2008-06-29	2008-07-01	SI	NO

Modificar Devolver

Para los Lectores, el interfaz es el siguiente:

Consultar Préstamos

Préstamos

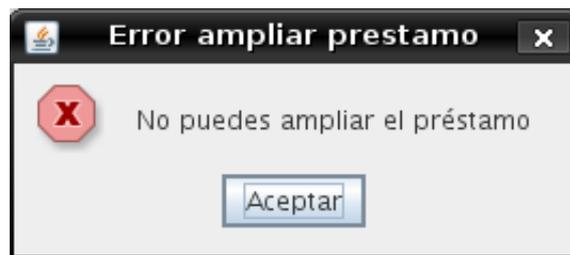
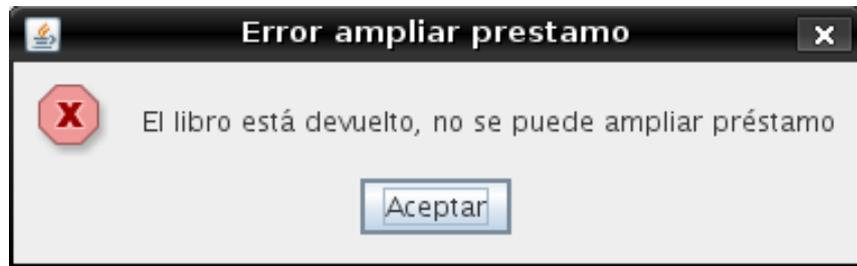
Devueltos Pendientes Todos

Resultados

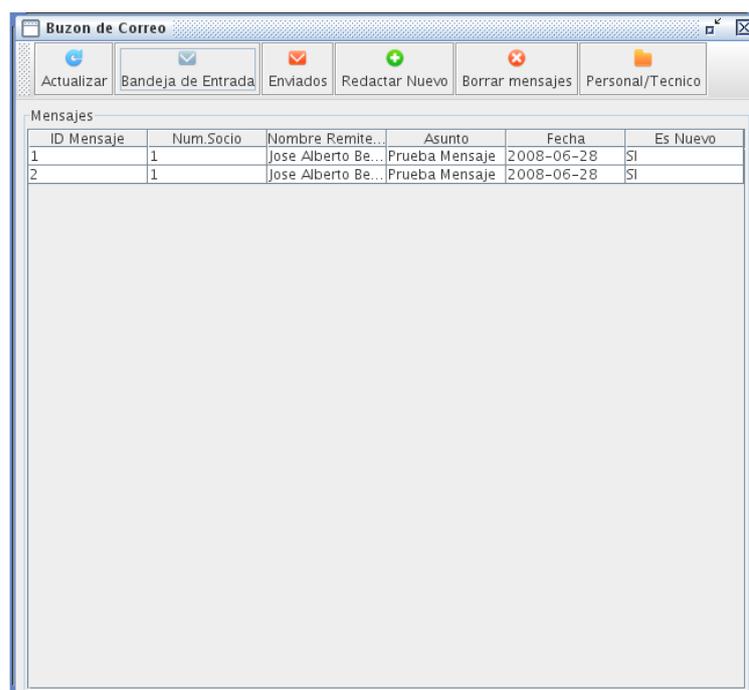
ID Prestamo	ID Libro	ID Revista	Fecha inicio	Fecha fin	Devuelto
1		2	2008-06-28	2008-06-30	SI
2	1		2008-06-28	2008-06-30	SI
3		2	2008-06-29	2008-07-01	SI
4	1		2008-06-29	2008-07-01	SI
5	1		2008-06-29	2008-07-01	SI
6	1		2008-06-29	2008-07-01	SI
7		1	2008-06-29	2008-07-20	SI
8		2	2008-06-29	2008-07-01	SI
9		1	2008-06-29	2008-07-20	SI
10		2	2008-06-29	2008-07-01	SI
11		2	2008-06-29	2008-07-01	SI
12		2	2008-06-29	2008-07-01	SI
13		2	2008-06-29	2008-07-01	SI
14		2	2008-06-29	2008-07-01	SI
15		2	2008-06-29	2008-07-01	SI
16		2	2008-06-29	2008-07-01	SI

Ampliar préstamo

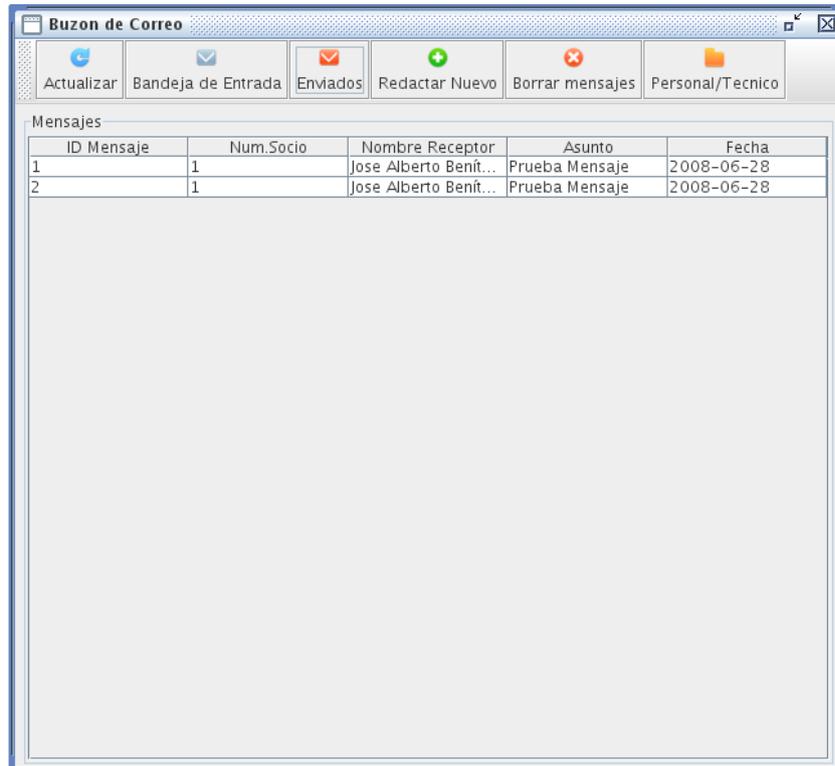
Pueden pedir una ampliación de préstamo, que a su vez puede estar denegada o aceptada:



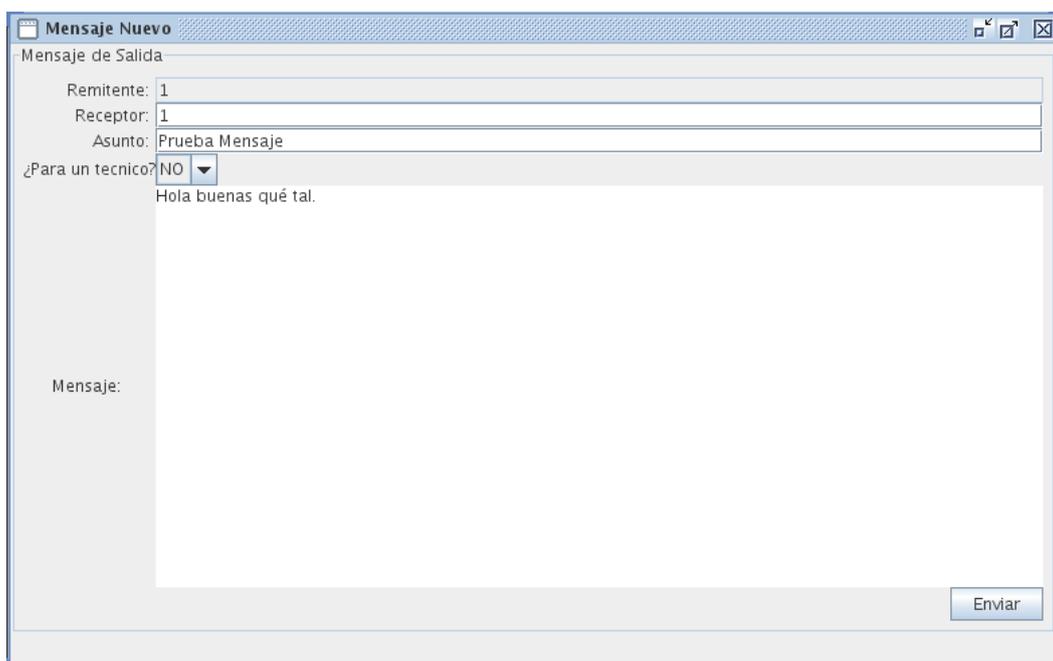
Por otra parte, tenemos la opción de mirar nuestro **Buzón** de mensajes:



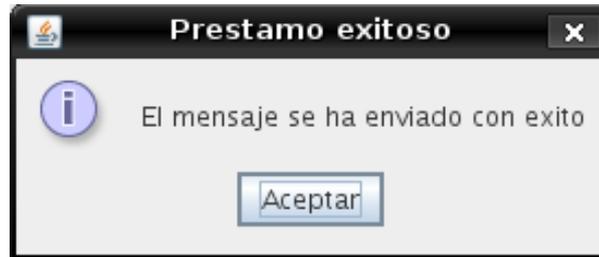
Y una bandeja de enviados tal que así:



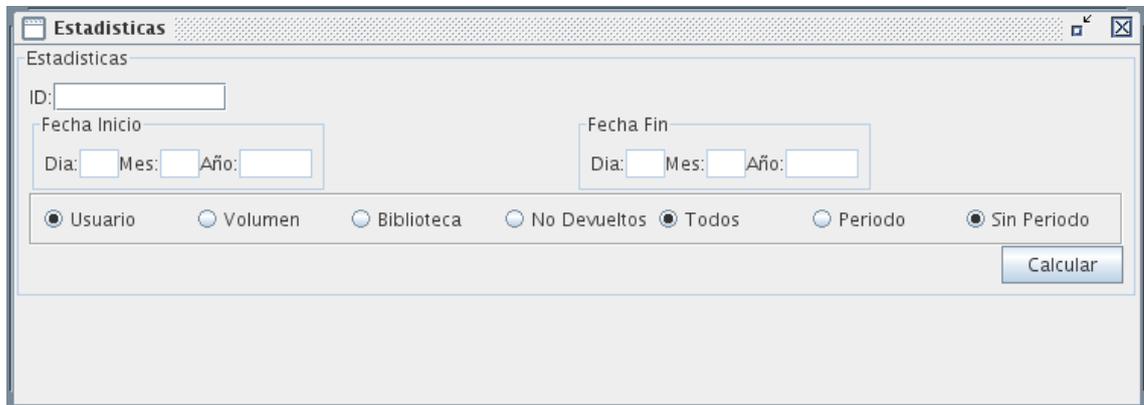
Si queremos **leer** o **enviar** un mensaje, tendremos unas ventanas como las siguientes:



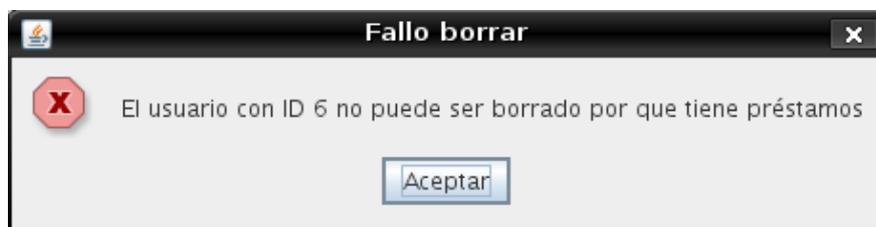
Si el mensaje se envía correctamente, obtendremos el siguiente mensaje:



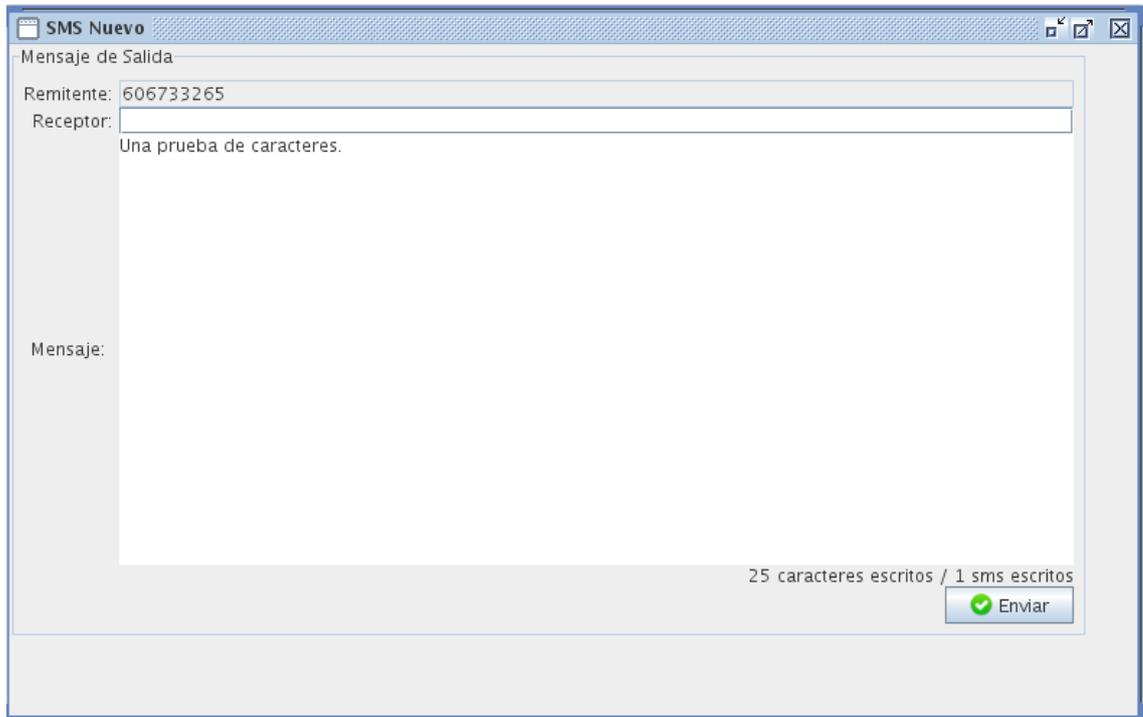
La ventana de estadísticas es la siguiente:



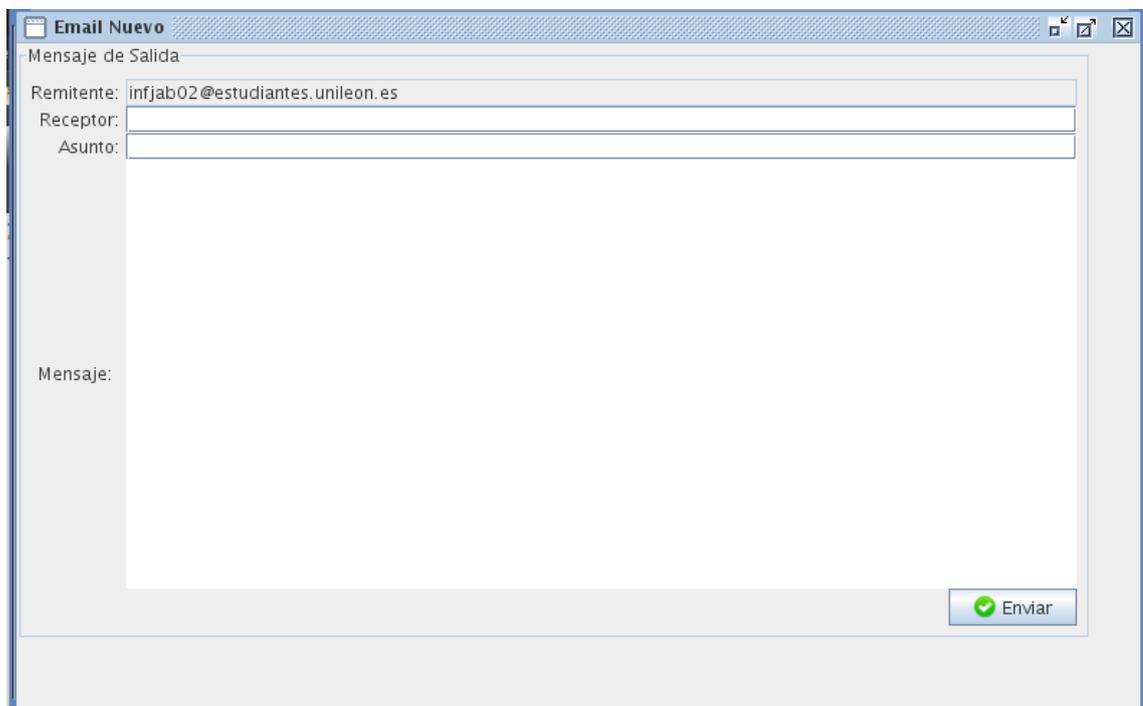
En relación al borrado de usuarios, si un usuario tuviese un préstamo en su poder, y lo intentásemos borrar, el programa mostraría el siguiente error:



Y ya por último, muestro dos imágenes de lo que sería la interfaz para enviar **SMS** y **Correos electrónicos** a los usuarios, pero que no está implementado.



The screenshot shows a window titled "SMS Nuevo" with a "Mensaje de Salida" header. It contains a "Remitente:" field with the value "606733265", an empty "Receptor:" field, and a "Mensaje:" area with the text "Una prueba de caracteres." At the bottom right, it displays "25 caracteres escritos / 1 sms escritos" and an "Enviar" button with a green checkmark icon.



The screenshot shows a window titled "Email Nuevo" with a "Mensaje de Salida" header. It contains a "Remitente:" field with the value "infjab02@estudiantes.unileon.es", empty "Receptor:" and "Asunto:" fields, and a "Mensaje:" area. At the bottom right, it displays an "Enviar" button with a green checkmark icon.