**Desarrollo de Sistema de información con arquitectura de tres capas:**

**Marco teórico**

Sistema de información: Un sistema de información es un conjunto organizado de elementos que interactúan para recopilar, procesar, almacenar y distribuir datos con el propósito de facilitar la toma de decisiones y el control en una organización. Este sistema puede incluir hardware, software, personal, datos, procedimientos y redes, trabajando en conjunto para gestionar y organizar información relevante. Su objetivo principal es proporcionar a los usuarios internos y externos la información necesaria de manera oportuna y precisa, contribuyendo así al funcionamiento eficiente y efectivo de la entidad o empresa. Los sistemas de información pueden variar en tamaño y complejidad, abarcando desde simples bases de datos hasta intrincados sistemas empresariales que integran múltiples funciones y procesos.

IDE: Un entorno de desarrollo integrado (IDE) es un software que proporciona herramientas y funciones para facilitar la programación y el desarrollo de software. Básicamente, es un ambiente completo que incluye un editor de código, un compilador/intérprete, herramientas de depuración y otras características que ayudan a los desarrolladores a escribir, probar y depurar su código de manera más eficiente.

Base de datos: Una base de datos es un sistema organizado para recopilar, almacenar y gestionar datos de manera estructurada, permitiendo el acceso eficiente, la actualización y la extracción de información según las necesidades específicas. Utiliza tablas para organizar la información en filas y columnas, donde cada fila representa un registro y cada columna un atributo. Este enfoque facilita la búsqueda y manipulación de datos mediante consultas y operaciones, brindando una plataforma eficaz para almacenar grandes cantidades de información de manera coherente y segura. Las bases de datos son fundamentales en numerosas aplicaciones, desde sistemas empresariales hasta sitios web, y desempeñan un papel crucial en la gestión y aprovechamiento de la información en diversos contextos.  
  
Base de datos relacional: Una base de datos relacional es un tipo de sistema de gestión de bases de datos (DBMS) que organiza la información en tablas relacionadas entre sí. Cada tabla consta de filas y columnas, donde cada fila representa un registro y cada columna un atributo específico. La relación entre las tablas se establece mediante claves primarias y claves foráneas, permitiendo la conexión y la referencia cruzada de datos entre diferentes tablas. Este enfoque proporciona integridad referencial y facilita consultas complejas mediante el uso de lenguajes como SQL (Structured Query Language). Las bases de datos relacionales son ampliamente utilizadas en entornos empresariales y aplicaciones donde la estructura y la relación de los datos son fundamentales para la eficiencia y coherencia de la información almacenada.

SQL: SQL, o Structured Query Language, es un lenguaje de programación diseñado para gestionar y manipular bases de datos relacionales. Utilizado en la administración de sistemas de gestión de bases de datos (DBMS), como MySQL, PostgreSQL o Microsoft SQL Server, SQL permite realizar operaciones como la creación y modificación de tablas, la inserción y actualización de datos, así como la realización de consultas complejas para recuperar información específica. Su sintaxis está diseñada para ser legible y fácil de entender, con comandos que se dividen en categorías como DDL (Data Definition Language) para definir la estructura de la base de datos, DML (Data Manipulation Language) para manipular datos, y DQL (Data Query Language) para realizar consultas. SQL es fundamental en el desarrollo y la gestión de sistemas de bases de datos relacionales en una amplia gama de aplicaciones y entornos.

Sistema de gestión de bases de datos: Un Sistema de Gestión de Bases de Datos (DBMS, por sus siglas en inglés) es un software que facilita la creación, gestión y manipulación de bases de datos. Su función principal es permitir a los usuarios definir la estructura de la base de datos, ingresar y actualizar datos, así como realizar consultas de manera eficiente. Los DBMS proporcionan un entorno seguro y organizado para almacenar información, garantizando la integridad de los datos y facilitando el acceso rápido mediante consultas y operaciones. Ejemplos comunes de DBMS incluyen MySQL, Oracle Database y Microsoft SQL Server. Estos sistemas son esenciales en el desarrollo de aplicaciones y en la gestión de grandes volúmenes de datos en entornos empresariales y de software.

MySQL: MySQL es un sistema de gestión de bases de datos relacional de código abierto. Desarrollado por Oracle Corporation, MySQL es ampliamente utilizado para almacenar y gestionar datos en aplicaciones web y sistemas empresariales. Ofrece una sólida funcionalidad de base de datos, incluyendo la capacidad de definir estructuras de tablas, realizar consultas complejas utilizando SQL, y garantizar la integridad y la seguridad de los datos. MySQL es conocido por su rendimiento eficiente, escalabilidad y compatibilidad con múltiples plataformas. Además, al ser de código abierto, ha ganado popularidad en la comunidad de desarrolladores, permitiendo a los usuarios adaptar y personalizar el sistema según sus necesidades específicas. Muchas aplicaciones web, como WordPress, utilizan MySQL como su sistema de gestión de bases de datos subyacente.

MariaDB: MariaDB es un sistema de gestión de bases de datos relacional de código abierto, diseñado como un fork de MySQL después de la adquisición de este último por Oracle Corporation. Desarrollado por la comunidad de código abierto y liderado por los creadores originales de MySQL, MariaDB mantiene la compatibilidad con MySQL pero también ofrece características adicionales y mejoras. Al igual que MySQL, MariaDB es conocido por su rendimiento, escalabilidad y facilidad de uso, y es utilizado en una variedad de aplicaciones, desde sitios web hasta sistemas empresariales. MariaDB se ha convertido en una opción popular para aquellos que buscan una alternativa de código abierto y comunitaria a MySQL, y ha ganado aceptación en la industria de la tecnología de la información.  
  
Java: Java es un lenguaje de programación de propósito general que fue desarrollado por Sun Microsystems en la década de 1990. Es conocido por ser un lenguaje orientado a objetos, lo que significa que organiza el código en objetos que interactúan entre sí. Una de las características distintivas de Java es su capacidad de ser "write once, run anywhere" (escribir una vez, ejecutar en cualquier lugar), gracias a la máquina virtual Java (JVM). Esto significa que el código Java puede ejecutarse en cualquier dispositivo que tenga una JVM instalada, lo que lo hace altamente portátil.

Java se utiliza en una variedad de aplicaciones, desde desarrollo web hasta aplicaciones empresariales y dispositivos móviles (Android). Además de ser portátil, Java es conocido por su seguridad, facilidad de mantenimiento y extensibilidad. Se ha convertido en uno de los lenguajes de programación más populares y ampliamente utilizados en el mundo del desarrollo de software.

Apache NetBeans: Apache NetBeans es un entorno de desarrollo integrado (IDE) de código abierto que proporciona herramientas y funciones para facilitar el desarrollo de software en varios lenguajes de programación, como Java, HTML, PHP y otros. Anteriormente conocido como "NetBeans IDE" antes de ser transferido a la Apache Software Foundation, este entorno ofrece un editor de código, depurador, diseñador de interfaz gráfica y otras utilidades para simplificar el proceso de desarrollo. Apache NetBeans es conocido por su versatilidad y compatibilidad con múltiples plataformas, lo que permite a los desarrolladores crear aplicaciones en diferentes entornos. Su participación en la comunidad de código abierto y su integración con Apache han contribuido a su evolución continua y su posición como una herramienta popular en el mundo del desarrollo de software.27/12/2023

UML: UML, o Lenguaje de Modelado Unificado (Unified Modeling Language, por sus siglas en inglés), es un estándar de la industria utilizado en ingeniería de software para visualizar, especificar, construir y documentar sistemas software orientados a objetos. Desarrollado en colaboración por destacados expertos en el campo, UML proporciona un conjunto de notaciones gráficas y semánticas que permiten a los desarrolladores representar visualmente la estructura y el comportamiento de un sistema. Este lenguaje de modelado versátil incluye diagramas como el de clases, de casos de uso, de secuencia y de actividades, ofreciendo una herramienta común para comunicar conceptos y diseños entre los miembros del equipo de desarrollo y facilitando la comprensión y documentación de proyectos complejos de software.

Arquitectura de desarrollo: La arquitectura de desarrollo se refiere a la estructura organizativa y conceptual de un sistema de software, delineando la manera en que sus componentes interactúan y se relacionan. Esta estructura guía el diseño y la implementación del software, definiendo la distribución de responsabilidades y funciones entre los diferentes módulos o capas del sistema. La arquitectura de desarrollo puede incluir decisiones sobre la elección de patrones de diseño, la separación de preocupaciones, la escalabilidad, la seguridad y otros aspectos cruciales para el desarrollo de software. Al adoptar una arquitectura bien planificada, los desarrolladores pueden lograr sistemas más robustos, mantenibles y flexibles. Ejemplos comunes de arquitecturas de desarrollo incluyen la arquitectura de tres capas, la arquitectura orientada a servicios (SOA) y la arquitectura de microservicios. Cada una de estas enfoques tiene sus propias características y beneficios, adaptándose a distintos contextos y requisitos de desarrollo.

Arquitectura de tres capas: La arquitectura de tres capas es un modelo de diseño de software que divide una aplicación en tres componentes principales o capas, cada una con funciones específicas y responsabilidades bien definidas. Estas capas son:

1. \*\*Capa de Presentación (Interfaz de Usuario):\*\* También conocida como capa de presentación o interfaz de usuario, esta capa se encarga de la presentación y la interacción directa con el usuario. Aquí se ubican elementos como la interfaz gráfica, formularios, botones y otros elementos visuales.

2. \*\*Capa de Lógica de Negocio (Lógica de Aplicación):\*\* Esta capa contiene la lógica de negocio de la aplicación. Aquí se procesan y manejan las reglas y operaciones que definen el comportamiento central de la aplicación. Es donde se lleva a cabo el procesamiento de datos y se toman decisiones basadas en la lógica específica del dominio.

3. \*\*Capa de Acceso a Datos (Persistencia):\*\* En esta capa, se gestionan las operaciones relacionadas con el almacenamiento y la recuperación de datos. Puede incluir la interacción con bases de datos, sistemas de archivos u otros medios de persistencia de datos.

La ventaja de la arquitectura de tres capas radica en la separación clara de responsabilidades, lo que facilita el mantenimiento, la escalabilidad y la reutilización del código. Además, permite cambios en una capa sin afectar directamente a las otras, lo que mejora la modularidad y la flexibilidad del sistema. Este enfoque es comúnmente utilizado en el desarrollo de aplicaciones empresariales y sistemas de software robustos.

XAMPP: XAMPP es un paquete de software gratuito y de código abierto que facilita la creación de un entorno de desarrollo web local. El nombre XAMPP proviene de las iniciales de sus componentes principales: X (cualquier sistema operativo), Apache (servidor web), MySQL (sistema de gestión de bases de datos), PHP (lenguaje de programación) y Perl (otro lenguaje de programación). Este paquete incluye todos estos componentes preconfigurados, lo que simplifica la tarea de configurar un servidor web local para el desarrollo y prueba de aplicaciones web.

XAMPP es especialmente popular entre desarrolladores web porque proporciona un entorno fácil de instalar y usar que simula un servidor web en un entorno de producción. Esto permite a los desarrolladores trabajar en sus proyectos de manera local antes de desplegarlos en un servidor en vivo. Además de los componentes principales, XAMPP también incluye herramientas adicionales y utilidades que son útiles para el desarrollo web.

Es importante destacar que, aunque XAMPP es adecuado para entornos de desarrollo, no se recomienda su uso directo en entornos de producción debido a cuestiones de seguridad y configuración.

Herramientas

Para el desarrollo de este sistema se utilizan las siguientes herramientas:

Apache netbeans: Gestion de codigo fuente e interfaz de usuario.

MySQL: Sistema de gestión de base de datos.

MariaDB: Base de datos.

XAAMP: Control de entornos de red.

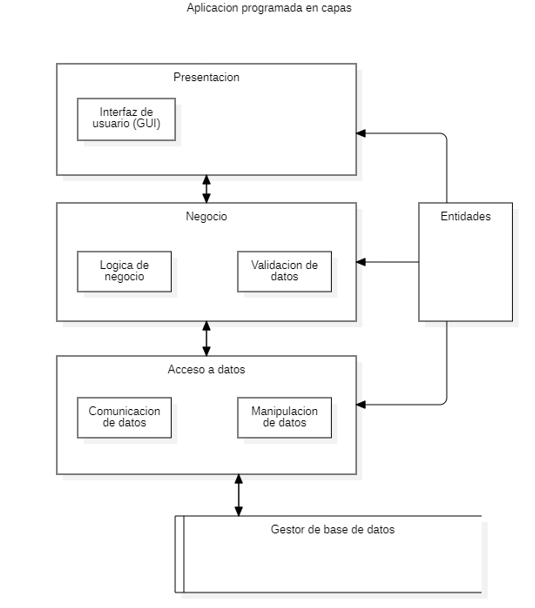
Java: Lenguaje de programación.

Git: Sistema de control de versión y repositorios

GitHub: Sistema de repositorio remoto.

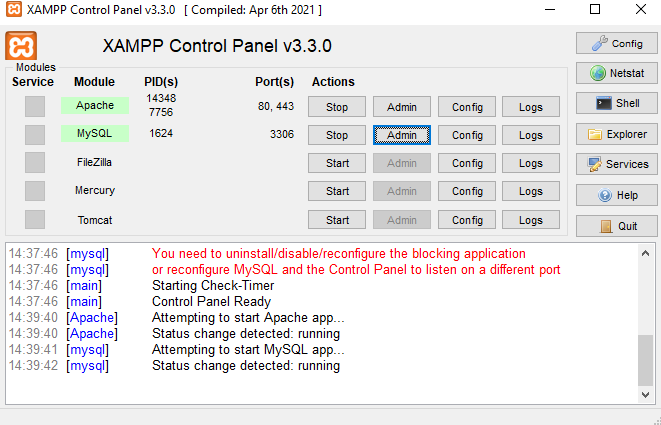
Arquitectura del sistema

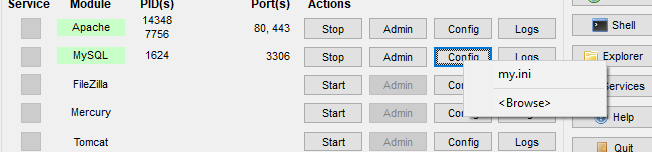
El sistema esta desarrollado usando arquitectura de 3 capas

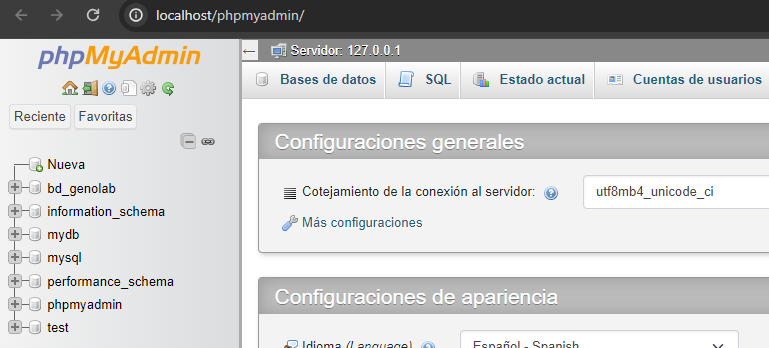


Gestor de base de datos

En este caso el gestor de base de datos será MySQL en combinación con MariaDB para su puesta en producción

La ejecución del servidor de bases de datos es como sigue:  
  
- En el panel de control de XAMPP, ejecutar el modulo apache y MySQL. Para  
  
- El número de puerto se puede configurar en my.ini, por defecto es 3306



- Luego, ejecutar phpmyadmin, que es la interfaz del sistema de gestión de MariaDb / XAMPP

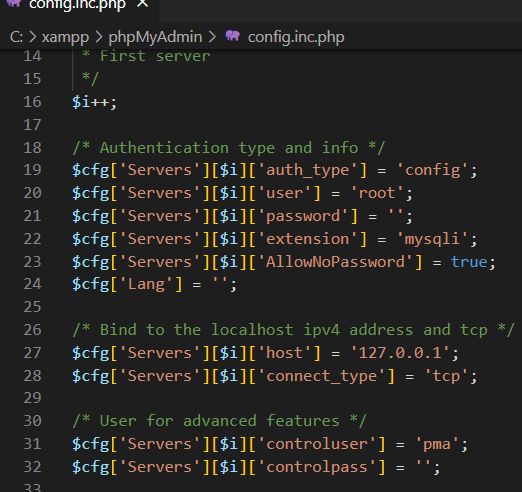
- Adicionalmente tambien se puede configurar el puerto el cual es el root de phpMyAdmin. Este archivo de configuración generalmente esta almacenado en la ruta C:\xampp\phpMyAdmin\config.inc.php

- En la línea 27 se puede configurar el puerto. Por defecto es:

$cfg['Servers'][$i]['host'] = '127.0.0.1';

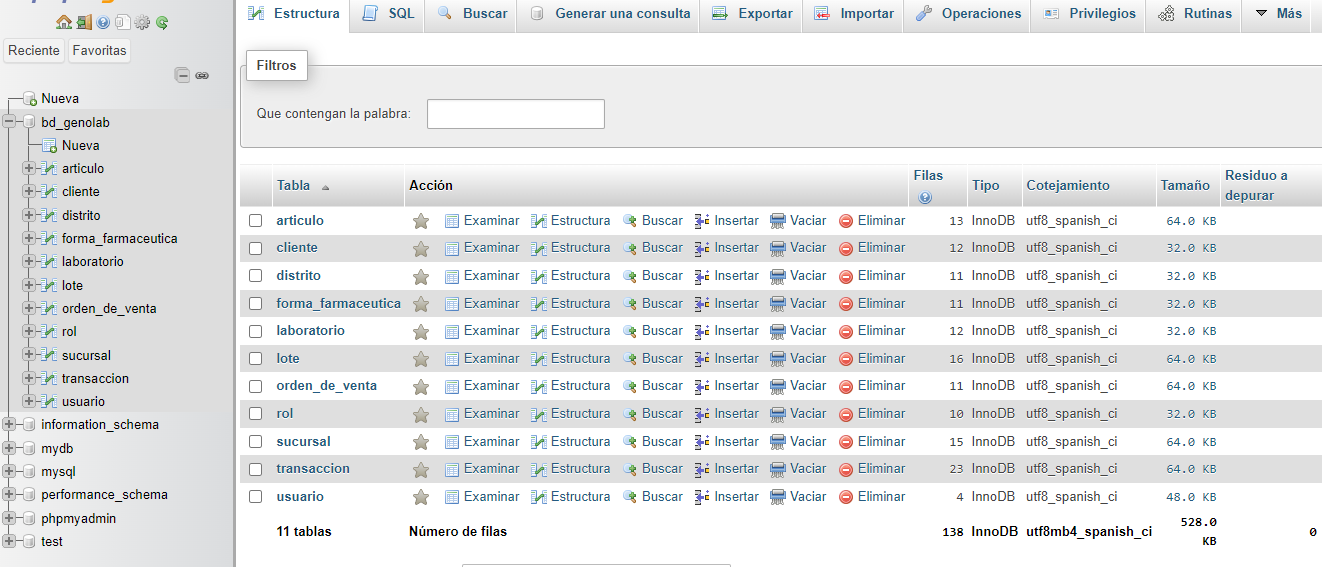
Para conectar con el puerto 3307 por ejemplo se puede configurarlo de esta manera:

$cfg['Servers'][$i]['host'] = '127.0.0.1:3307';



- Como ejemplo de la base de datos, ver el archivo bd\_genolab.sql, que es el backup de la base de datos de este proyecto.

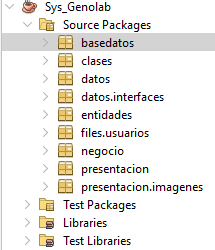
- Una vez disponible la base de datos se puede proceder con el desarrollo de la aplicación, en este caso, el cotejamiento de la base de datos es utf8\_spanish\_ci, esta configuración se puede asignar en Mysql server y Phpmyadmin al crear la base de datos.



Desarrollo de aplicación

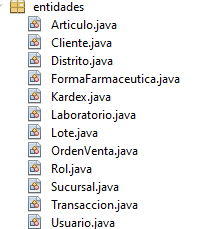
La aplicación se desarrolla en Apache Netbeans, con JDK adoptium

La plantilla del proyecto es un proyecto Java ant

En este caso la aplicación tiene los siguientes paquetes:  
  


Capa de acceso a datos

Entidades



El paquete entidades contiene las clases que representan las entidades de la aplicación.

Las entidades en este paquete se utilizan para representar los datos en la base de datos.

Las clases en este paquete se utilizan para transferir datos entre la aplicación.

En esta explicación, se referirá con Entidades para generalizar cualquier clase dentro de este paquete, por fines explicativos

Una clase de entidad (por ejemplo Rol.java, Usuario.java) representa una entidad u objeto único en la aplicación. Es una clase Java simple que contiene propiedades y sus métodos de obtención y establecimiento (get y set). Estas propiedades corresponden a las columnas de una tabla en la base de datos. Una clase de Entidad se utiliza para mapear los datos de la tabla a objetos Java y viceversa. Este mapeo permite que la aplicación interactúe con la base de datos de manera orientada a objetos.

Este es un desglose general del código:

private int id; - Esta línea declara una variable entera privada llamada id. Esta variable representa el identificador único (clave primaria) para la entidad en la base de datos.

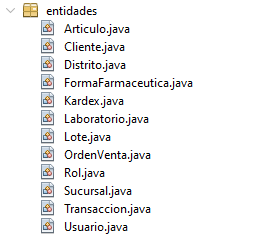
private String name; - Esta línea declara una variable String privada llamada name. Esta variable representa un atributo de nombre de la entidad.

private boolean active; - Esta línea declara una variable booleana privada llamada active. Esta variable representa el estado activo de la entidad.

public Entidad() {} - Este es el constructor por defecto para la clase Entidad. No toma ningún parámetro y no realiza ninguna acción. Este constructor es necesario para que la clase pueda ser instanciada.

La clase Entity típicamente también incluye métodos de obtención y establecimiento para cada una de sus propiedades. Estos métodos permiten que otras partes de tu aplicación accedan y modifiquen las propiedades de los objetos Entity.

Recuerda que los nombres reales y los tipos de las propiedades dependerán de la entidad específica que la clase esté representando. Los nombres y tipos deben coincidir con las columnas y sus tipos en la tabla de base de datos correspondiente.



Base datos:



En este paquete se pertenece a la capa de acceso a datos

La clase Conexion representa una conexión a una base de datos. Proporciona métodos para conectarse y desconectarse de la base de datos, gestionar transacciones y acceder a metadatos.

La clase Conexion representa una instancia única de conexión a la base de datos. Garantiza que solo se cree una instancia de la conexión y proporciona un método sincronizado para acceder a la instancia. (ver [método getInstancia](../3er%20semestre/FABIOCAMACHOPROYECTOFINAL/Proyecto/src/basedatos/Conexion.java))

Datos

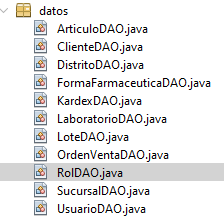
Interfaces



El código proporcionado es una interfaz de Java llamada CrudSimpleInterface. Esta interfaz está diseñada para establecer un contrato para operaciones CRUD básicas (Crear, Leer, Actualizar, Eliminar) en un tipo genérico T.

La interfaz CrudSimpleInterface se declara con un parámetro de tipo <T>. Esto significa que se puede utilizar con cualquier clase o tipo. El tipo T es un marcador de posición que se reemplazará por un tipo real cuando se cree un objeto de CrudSimpleInterface.

(ver interface CrudSimpleInterface)

Objetos de acceso a datos (DAO)  
  
Las clases Objetos de Acceso a Datos son clases que utilizan la API JDBC para interactuar con la base de datos SQL  
  
En este ejemplo, el código pertenece a una clase llamada RolDAO, que significa "Rol Data Access Object" (Objeto de Acceso a Datos de Rol). El patrón de diseño DAO es comúnmente utilizado para aislar la capa de aplicación/negocio de la capa de persistencia (generalmente una base de datos relacional, pero podría ser cualquier otro tipo de fuente de datos).

La clase RolDAO implementa CrudSimpleInterface<Rol>. Esto sugiere que la clase proporciona métodos para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en objetos Rol. La clase Rol probablemente sea un objeto Java simple que representa una fila en la tabla 'rol' en la base de datos.

La clase tiene varias variables de instancia:

CON es una instancia de la clase Conexion, que es responsable de gestionar la conexión a la base de datos.

ps es un objeto PreparedStatement, que se puede utilizar para ejecutar consultas SQL parametrizadas.

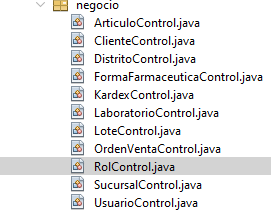
rs es un objeto ResultSet, que puede contener los resultados de una consulta SQL.

respuesta es un booleano que podría usarse para indicar el éxito o el fracaso de una operación.

La clase RolDAO tiene un constructor (public RolDAO()) que actualmente está vacío. Este constructor podría utilizarse para inicializar la variable CON u realizar otras tareas de configuración.

(ver clase RolDAO)

Negocio



Las clases en el paquete negocio forman parte de la capa de lógica empresarial (o capa de servicios) en la aplicación. Estas clases son responsables de controlar las operaciones relacionadas con las clases de Entidad en el paquete entidades. Interactuan con las clases DAO para realizar operaciones CRUD (en las) con las clases de Entidad.

Aquí hay un desglose del código:

1. **package negocio;** - Esta línea declara el paquete al que pertenece la clase RolControl. En Java, los paquetes se utilizan para agrupar clases e interfaces relacionadas. Es probable que el paquete negocio contenga otras clases que manejen la lógica empresarial de tu aplicación.
2. **import datos.RolDAO; import entidades.Rol;** - Estas líneas importan las clases RolDAO y Rol, que se utilizan más adelante en la clase RolControl. La clase RolDAO es responsable de interactuar con la base de datos, mientras que la clase Rol representa una entidad 'rol'.
3. **private final RolDAO DATOS; private Rol obj; private DefaultTableModel modeloTabla;** - Estas líneas declaran tres variables de instancia privadas. DATOS es un objeto final de RolDAO, lo que significa que una vez inicializado, no puede cambiarse. obj es un objeto Rol, y modeloTabla es un objeto DefaultTableModel, que forma parte de la biblioteca Swing y se utiliza para contener datos de una tabla en una interfaz gráfica de usuario (GUI).
4. **public int registrosMostrados;** - Esta línea declara una variable pública de tipo entero llamada registrosMostrados. Esta variable probablemente lleva un registro del número de registros 'rol' que se están mostrando actualmente en la GUI.
5. **public String insertar:** Inserta una nueva Entidad con nombre dado. Si ya existe una entidad con el mismo nombre, retorna una cadena que dice “El registro ya existe”  
   Sino, se inserta la nueva Entidad y retorna una cadena “Ok” si hay éxito con la inserción, sino devuelve un mensaje de error.
6. **public String actualizar:** Actualiza la entidad Rol con el idRol, nombre y nombreAnterior dados.

Si el nombre es igual a nombreAnterior, actualiza la entidad Rol con el

idRol y nombre dados.

Si el nombre es diferente de nombreAnterior, verifica si ya existe una entidad Rol

con el mismo nombre.

Si existe, devuelve "El registro ya existe".

De lo contrario, actualiza la entidad Rol con el idRol y nombre dados, y

devuelve "OK" si tiene éxito,

o "Error en la actualización" en caso contrario.

@param idRol El ID de la entidad Rol que se va a actualizar.

@param nombre El nuevo nombre de la entidad Rol.

@param nombreAnterior El nombre anterior de la entidad Rol.

@return Una cadena que indica el resultado de la operación de actualización.

1. El bloque de comentario describe el constructor de la clase RolControl. Este constructor inicializa el campo DATOS con una nueva instancia de RolDAO, el campo obj con una nueva instancia de Rol y el campo registrosMostrados con 0. Sin embargo, el código real del constructor no se muestra en el código proporcionado.

Este código forma parte de un proyecto más grande que utiliza Ant para compilar el código Java. Ant es una biblioteca y herramienta de línea de comandos de Java que ayuda en la ejecución de procesos descritos en archivos de compilación como objetivos y puntos de extensión dependientes entre sí. El uso principal conocido de Ant es la compilación de aplicaciones Java.

Paquete de clases



Este paquete aloja clases complementarias que almacenan datos para su uso durante la aplicación, y para validaciones generales en los formularios

Variables

The Variables.java file is a Java class that contains a set of public static variables. These variables are accessible from any other class in your application without needing to instantiate the Variables class.

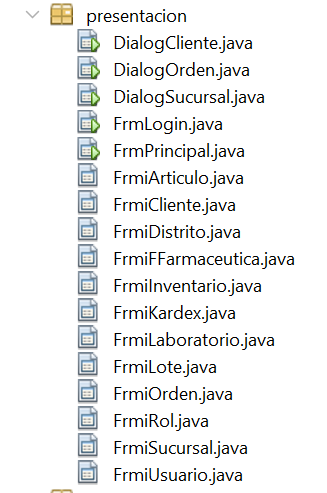
Here's a breakdown of the code:

La clase Variables.java contiene un conjunto de pariables publicas estáticas. Estas variables son accesibles desde cualquier otra clase en la aplicación sin necesidad de intanciar la clase Variables

These variables are likely used to store information about the current user and their role in your application. The values of these variables can be set and accessed from any other class in your application.

Estas variables son usadas para guardar información acerca del usuario actual y su rol en la aplicación. Estos valores pueden ser cambiados y accedidos desde cualquier otra clase de la aplicación.

Paquete de presentacion:



The presentacion package contains classes related to the presentation layer of the application.  
  
**Formulario de acceso al sistema**

The provided code is part of a Java Swing application, specifically a class named FrmLogin which extends javax.swing.JFrame. This class is likely a login form for your application.

The FrmLogin class is part of the presentacion package, as indicated by the package declaration at the top of the file. It also imports the JOptionPane class from the javax.swing package and the UsuarioControl class from the negocio package.

The FrmLogin class has a constructor FrmLogin(), which is called when a new instance of the class is created. Inside this constructor, it calls the initComponents() method and sets the title and location of the JFrame.

The initComponents() method is typically auto-generated by a GUI builder tool and is used to initialize the components of the form. The setTitle("Acceso al sistema") method sets the title of the JFrame to "Acceso al sistema". The setLocationRelativeTo(null) method centers the JFrame on the screen.

The FrmLogin class also has three private methods for displaying different types of messages to the user: mensajeError(String mensaje), mensajeInformacion(String mensaje), and mensajeAlerta(String mensaje). These methods use the JOptionPane.showMessageDialog() method to display a dialog box with a message. The type of message (error, information, alert) determines the icon that is displayed in the dialog box. The methods are private, meaning they can only be called within the FrmLogin class.

The provided code is a method named btnIngresarActionPerformed. This method is an event handler that gets called when the "Ingresar" button is clicked in your Java Swing application. Here's a breakdown of what this method does:

if (txtEmail.getText().isEmpty() || txtPassword.getText().isEmpty()): This line checks if the email or password fields are empty. If either of them is empty, it displays an error message, sets the focus back to the email field, and returns from the method.

UsuarioControl control = new UsuarioControl();: This line creates a new instance of the UsuarioControl class. This class is likely responsible for user-related operations, such as logging in.

String resp = control.login(txtEmail.getText(), txtPassword.getText());: This line calls the login method of the UsuarioControl class, passing the text from the email and password fields as arguments. The login method likely checks these credentials against a database or other data source and returns a response.

The following if and else if statements check the response from the login method. If the response is "1", it disposes of the current form, creates a new instance of the FrmPrincipal class (likely the main form of your application), brings it to the front, and makes it visible. If the response is "2", it displays an alert message saying that the user is inactive. If the response is anything else, it displays an error message saying that the login credentials are incorrect.

The mensajeError and mensajeAlerta methods are used to display error and alert messages to the user. They use the JOptionPane.showMessageDialog method to display a dialog box with a message and an icon. The mensajeError method displays an error message with an error icon, and the mensajeAlerta method displays an alert message with a warning icon.

(ver FrmLogin)

**Formulario principal**

The provided code is part of a Java Swing application, specifically a class named FrmPrincipal which extends javax.swing.JFrame. This class is likely the main window or form of your application.

The FrmPrincipal class is part of the presentacion package, as indicated by the package declaration at the top of the file. It also imports the Variables class from the clases package and the JOptionPane class from the javax.swing package.

The FrmPrincipal class has a constructor FrmPrincipal(), which is called when a new instance of the class is created. Inside this constructor, it calls three methods:

initComponents(): This method is typically auto-generated by a GUI builder tool and is used to initialize the components of the form.

setExtendedState(MAXIMIZED\_BOTH): This method sets the form to be maximized in both dimensions (height and width) when it is displayed.

cargarOpcionesMenu(): This method is used to load the menu options based on the user's role. The implementation of this method is not shown in the provided code, but the comment above the method declaration provides some insight into its functionality. If the user's role is null (i.e., clases.Variables.rolNombre == null), it displays an alert message and exits the system. It also enables or disables menu options based on the user's role and sets labels with the user's name, last name, and role.

This code demonstrates a common way of handling role-based access control (RBAC) in a GUI application, where different types of users have access to different parts of the application.

(ver FrmPrincipal)

Formularios de dialogo (dialogo cliente)  
The provided code is part of a Java Swing application, specifically a class named `DialogCliente` which extends `javax.swing.JDialog`. This class is likely a dialog form for managing client-related operations in your application.

The `DialogCliente` class is part of the `presentacion` package, as indicated by the package declaration at the top of the file. It also imports several other classes: `OrdenVentaControl` from the `negocio` package, `Cliente` from the `entidades` package, and `TableRowSorter` from the `javax.swing.table` package.

The `DialogCliente` class has two private instance variables: `control` of type `OrdenVentaControl` and `cliente` of type `Cliente`.

The `DialogCliente` constructor takes two parameters: a `java.awt.Frame` object and a boolean value. Inside this constructor, it calls the `super(parent, modal)` method to create a new `JDialog` with the specified parent frame and modality. It then calls the `initComponents()` method to initialize the components of the dialog. The `control` variable is assigned a new instance of `OrdenVentaControl`, the `listado("")` method is called (though its implementation is not shown in the provided code), and the `cliente` variable is assigned a new instance of `Cliente`.

The `getCliente()` and `setCliente(Cliente cliente)` methods are getter and setter methods for the `cliente` instance variable. The getter method returns the current `Cliente` object, and the setter method assigns a new `Cliente` object to the `cliente` variable.

**(ver dialogo cliente)**