BEATRIZ GARCÍA

Beatriz García

Hito II Acceso a Datos

1er trimestre

ÍNDICE

[**SISTEMA DE GESTIÓN DE BASE DE DATOS**](#_SISTEMA_DE_GESTIÓN)

**CÓDIGO**

**BASE DE DATOS**

**LOGIN**

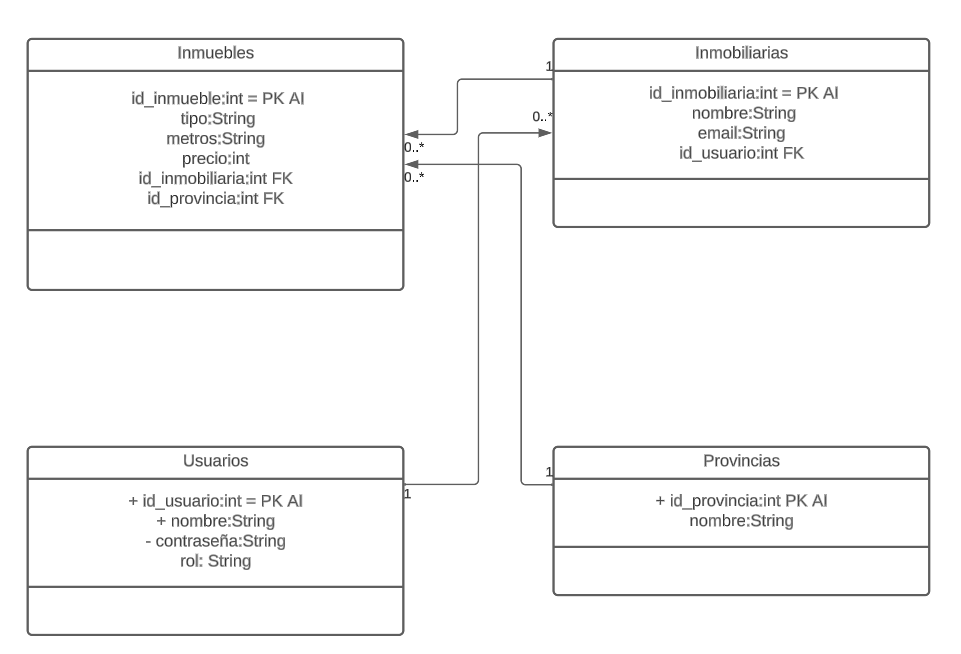
**[EVALUACIÓN](#_EVALUACIÓN_1)**

[**BIBLIOGRAFÍA**](#_Bibliografía)

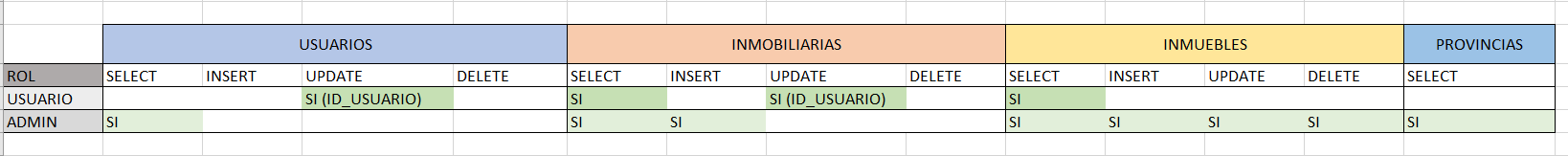
# SISTEMA DE GESTIÓN DE BASE DE DATOS

He realizado el diseño de una base de datos relacional con la información distribuida en distintas tablas utilizando SQL. Código [SQL](#_Base_de_datos)

A continuación, se muestra una imagen con el diagrama de relaciones de la nueva base de datos.



Vamos a ver mediante una tabla los permisos que tiene cada rol de usuario en la base de datos:



Al abrir la aplicación, se solicitarán unos datos para poder realizar el login de usuario (click [aquí](#_Login) para ver esta parte en el código).

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

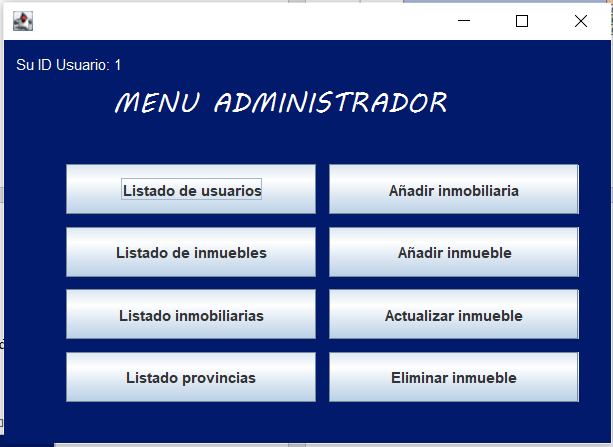
En esta parte se han controlado los posibles errores a la hora de introducir los datos para la comprobación de usuario. Errores de conexión y errores de consulta de base de datos.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Si no hay ningún problema con la conexión y el nombre y contraseña son correctos, el sistema comprueba el rol del usuario que tiene el usuario que ha iniciado sesión y abre un menú diferente para cada rol, en este caso, un menú para los encargados de la gestión de Found Home con el rol de “administrador” en la base de datos, y otro menú para los que tienen el rol de “usuario” que serán las inmobiliarias y los clientes.

Cada menú ofrece opciones distintas según los permisos otorgados a cada usuario. Vamos a ver la ejecución de ambos en la aplicación:



Se ha añadido la información del ID de usuario que ha iniciado sesión en la parte superior izquierda a cada uno de los menús para tenerlo presente por si fuera necesario para realizar alguna de las operaciones (click [aquí](#_Información_ID_usuario) para ver esta parte en el código).

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

El método que recoge los datos del usuario que ha iniciado sesión, así como los métodos que comprueban el nombre y contraseña para iniciar sesión se encuentran en la clase “MenuPrincipal”.

Antes de continuar vamos a ver la disposición del código en el proyecto para poder orientarnos mejor.

El proyecto tiene siete paquetes distintos para organizar las clases y los métodos en el código:

Texto

Descripción generada automáticamente con confianza baja

* Conexión: este paquete contiene una clase con el método para establecer conexión con la base de datos, y una clase que gestiona todas las consultas que se realizan a la base de datos.
* Gui: menús principales de la aplicación.
* Inmobiliarias: contiene las clases con las distintas ventanas para la navegación referente a las inmobiliarias.
* Inmuebles: contiene las clases con las ventanas para la navegación referente a los inmuebles.
* Main: contiene la clase con el método principal main():
* Provincias: contiene la clase con la ventana que muestra el listado de provincias.
* Usuarios: contiene las clases con las disntintas ventanas para la navegación referente a usuarios y la clase Usuario con su constructor y métodos getter y setter.

Vamos a ver cómo trabajar con la aplicación viendo algunos supuestos.

**Proceso de añadir una inmobiliaria con rol de administrador.**

Iniciamos sesión en la aplicación con el rol de administrador:

Nombre: admin

Contraseña: 1234

A continuación, hacemos click en “Añadir inmobiliaria”. Se abrirán dos ventanas: una ventada con el listado de usuarios para poder comprobar el id\_usuario que queremos relacionar con la inmobiliaria y otra ventana con el formulario para añadir la inmobiliaria a la base de datos. Esta es una de las ventajas de este sistema, que puedes acceder a información de otra tabla al momento y siempre que lo necesites configurando el código según las necesidades del usuario.

Interfaz de usuario gráfica

Descripción generada automáticamente

El administrador tiene acceso a poder agregar inmobiliarias sin limitaciones por su id\_usuario, es decir, puede agregar tantas inmobiliarias como personas hayan contactado con la empresa para publicar un anuncio.

Esta parte de código también tiene manejo de errores: no se podrán introducir inmobiliarias con un id\_usuario que no conste en la base de datos. Tampoco se podrán crear dos inmobiliarias con el mismo nombre. De esta manera evitamos duplicados y se protege la integridad de los datos. Cuando una consulta de update o insert se ha ejecutado correctamente la aplicación también muestra un mensaje informativo a través de la interfaz para informar al usuario.

**Proceso de añadir inmueble rol administrador**

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

En este caso al hacer click en añadir inmueble, se abre una ventana con el listado de las inmobiliarias para poder rellenar correctamente el campo id\_inmobiliaria en el formulario. De esta manera se tiene acceso al momento a toda la información sin perder tiempo en otras consultas. Esta parte mantiene la integridad referencial con control de errores.

En las opciones actualizar y eliminar inmueble del menú de administrador, también se abrirá la ventana con el listado de los inmuebles para tener acceso rápido a los id de los inmuebles y evitar cometer errores.

**Actualizar datos del perfil de usuario rol de usuario**

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

El usuario sólo podrá acceder a modificar su nombre y su contraseña. Esta parte tiene manejo de errores (click [aquí](#_Actualizar_Perfil_Usuario) para ver esta parte en el código).

**Actualizar datos de contacto inmobiliaria rol usuario**

Interfaz de usuario gráfica

Descripción generada automáticamente

Al solicitar la actualización de datos de contacto de la inmobiliaria se abrirá una ventana con el listado de inmobiliarias ya que un mismo usuario, que podría ser un comercial freelance, puede trabajar para varias inmobiliarias. Ingresará el id de la inmobiliaria que desea actualizar y los nuevos datos. Sólo podrá modificar los datos de contacto de las inmobiliarias que estén asociadas a su id de usuario. De esta manera se mantiene la integridad de los datos y se refuerza la seguridad (click [aquí](#_Actualizar_datos_de) para ver el código).

# CÓDIGO

## Base de datos

CREATE TABLE provincias (

id INT AUTO\_INCREMENT PRIMARY KEY,

nombre VARCHAR(100)

);

INSERT INTO provincias (nombre) VALUES

('Álava'),

('Albacete'),

('Alicante'),

('Almería'),

('Asturias'),

('Ávila'),

('Badajoz'),

('Barcelona'),

('Burgos'),

('Cáceres'),

('Cádiz'),

('Cantabria'),

('Castellón'),

('Ciudad Real'),

('Córdoba'),

('Cuenca'),

('Gerona'),

('Granada'),

('Guadalajara'),

('Guipúzcoa'),

('Huelva'),

('Huesca'),

('Islas Baleares'),

('Jaén'),

('La Coruña'),

('La Rioja'),

('Las Palmas'),

('León'),

('Lérida'),

('Lugo'),

('Madrid'),

('Málaga'),

('Murcia'),

('Navarra'),

('Orense'),

('Palencia'),

('Pontevedra'),

('Salamanca'),

('Santa Cruz de Tenerife'),

('Segovia'),

('Sevilla'),

('Soria'),

('Tarragona'),

('Teruel'),

('Toledo'),

('Valencia'),

('Valladolid'),

('Vizcaya'),

('Zamora'),

('Zaragoza');

CREATE TABLE usuarios (

id\_usuario INT AUTO\_INCREMENT PRIMARY KEY,

nombre VARCHAR(50),

contraseña VARCHAR(20),

rol VARCHAR(50)

);

-- Insertar un administrador

INSERT INTO usuarios (nombre, contraseña, rol)

VALUES ('Admin', '1234', 'administrador'),

('Admin2', '1234', 'administrador');

-- Insertar usuarios con rol "usuario" y contraseña "1234"

INSERT INTO usuarios (nombre, contraseña, rol)

VALUES ('User1', '1234', 'usuario'),

('User2', '1234', 'usuario'),

('User3', '1234', 'usuario'),

('User4', '1234', 'usuario'),

('User5', '1234', 'usuario'),

('User6', '1234', 'usuario'),

('User7', '1234', 'usuario'),

('User8', '1234', 'usuario'),

('User9', '1234', 'usuario'),

('User10', '1234', 'usuario'),

('User11', '1234', 'usuario'),

('User12', '1234', 'usuario'),

('User13', '1234', 'usuario'),

('User14', '1234', 'usuario'),

('User15', '1234', 'usuario'),

('User16', '1234', 'usuario'),

('User17', '1234', 'usuario'),

('User18', '1234', 'usuario'),

('User19', '1234', 'usuario'),

('User20', '1234', 'usuario');

CREATE TABLE inmobiliarias (

id\_inmobiliaria INT AUTO\_INCREMENT PRIMARY KEY,

nombre VARCHAR(50),

email VARCHAR(100),

id\_usuario INT,

FOREIGN KEY (id\_usuario) REFERENCES usuarios(id\_usuario)

);

-- Insertar inmobiliarias con id\_usuario correspondiente

INSERT INTO inmobiliarias (nombre, email, id\_usuario)

VALUES ('Inmobiliaria1', 'info1@inmobiliaria.com', 1),

('Inmobiliaria2', 'info2@inmobiliaria.com', 2),

('Inmobiliaria3', 'info3@inmobiliaria.com', 3),

('Inmobiliaria4', 'info4@inmobiliaria.com', 4),

('Inmobiliaria5', 'info5@inmobiliaria.com', 5),

('Inmobiliaria6', 'info6@inmobiliaria.com', 6),

('Inmobiliaria7', 'info7@inmobiliaria.com', 7),

('Inmobiliaria8', 'info8@inmobiliaria.com', 8),

('Inmobiliaria9', 'info9@inmobiliaria.com', 9),

('Inmobiliaria10', 'info10@inmobiliaria.com', 10),

('Inmobiliaria11', 'info11@inmobiliaria.com', 11),

('Inmobiliaria12', 'info12@inmobiliaria.com', 12),

('Inmobiliaria13', 'info13@inmobiliaria.com', 13),

('Inmobiliaria14', 'info14@inmobiliaria.com', 14),

('Inmobiliaria15', 'info15@inmobiliaria.com', 15),

('Inmobiliaria16', 'info16@inmobiliaria.com', 16),

('Inmobiliaria17', 'info17@inmobiliaria.com', 17),

('Inmobiliaria18', 'info18@inmobiliaria.com', 18),

('Inmobiliaria19', 'info19@inmobiliaria.com', 19),

('Inmobiliaria20', 'info20@inmobiliaria.com', 20),

('Inmobiliaria21', 'info21@inmobiliaria.com', 1),

('Inmobiliaria22', 'info22@inmobiliaria.com', 2),

('Inmobiliaria23', 'info23@inmobiliaria.com', 3),

('Inmobiliaria24', 'info24@inmobiliaria.com', 4),

('Inmobiliaria25', 'info25@inmobiliaria.com', 5),

('Inmobiliaria26', 'info26@inmobiliaria.com', 6),

('Inmobiliaria27', 'info27@inmobiliaria.com', 7),

('Inmobiliaria28', 'info28@inmobiliaria.com', 8),

('Inmobiliaria29', 'info29@inmobiliaria.com', 9),

('Inmobiliaria30', 'info30@inmobiliaria.com', 10);

CREATE TABLE inmuebles (

id\_inmueble INT AUTO\_INCREMENT PRIMARY KEY,

tipo VARCHAR(50),

metros VARCHAR(20),

precio INT,

id\_inmobiliaria INT,

id\_provincia INT,

FOREIGN KEY (id\_inmobiliaria) REFERENCES inmobiliarias(id\_inmobiliaria),

FOREIGN KEY (id\_provincia) REFERENCES provincias(id)

);

INSERT INTO inmuebles (tipo, metros, precio, id\_inmobiliaria, id\_provincia)

VALUES

('Casa baja', '200', 250000, 1, 1),

('Piso', '80', 120000, 2, 2),

('Chalet', '150', 180000, 3, 3),

('Piso', '100', 135000, 4, 4),

('Ático-dúplex', '180', 210000, 5, 5),

('Oficina', '75', 110000, 6, 6),

('Casa', '220', 280000, 7, 7),

('Piso', '90', 125000, 8, 8),

('Chalet', '160', 190000, 9, 9),

('Dúplex', '85', 130000, 10, 10),

('Casa baja', '210', 260000, 11, 11),

('Piso', '95', 140000, 12, 12),

('Chalet', '170', 200000, 13, 13),

('Piso', '88', 128000, 14, 14),

('Ático-dúplex', '190', 220000, 15, 15),

('Oficina', '78', 115000, 16, 16),

('Casa baja', '230', 290000, 17, 17),

('Piso', '92', 132000, 18, 18),

('Chalet', '175', 205000, 19, 19),

('Piso', '82', 118000, 20, 20),

('Casa baja', '205', 255000, 1, 1),

('Piso', '87', 126000, 2, 2),

('Chalet', '155', 185000, 3, 3),

('Piso', '105', 138000, 4, 4),

('Ático-dúplex', '183', 213000, 5, 5),

('Oficina', '79', 112000, 6, 6),

('Casa baja', '225', 283000, 7, 7),

('Piso', '91', 127000, 8, 8),

('Chalet', '163', 193000, 9, 9),

('Piso', '83', 131000, 10, 10),

('Casa baja', '215', 263000, 11, 11),

('Piso', '97', 142000, 12, 12),

('Chalet', '173', 203000, 13, 13),

('Piso', '90', 130000, 14, 14),

('Ático-dúplex', '195', 223000, 15, 15),

('Oficina', '80', 117000, 16, 16),

('Casa baja', '235', 293000, 17, 17),

('Piso', '95', 134000, 18, 18),

('Chalet', '180', 207000, 19, 19),

('Piso', '85', 120000, 20, 20);

## Login

Texto

Descripción generada automáticamente

Análisis por líneas y contenido:

Línea 22: Declaración de la clase MenuPrincipal que extiende JFrame

para crear la ventana de interfaz gráfica.

26-36. Declaración de elementos necesarios para la interfaz gráfica y de el objeto Usuario que tendrá los atributos del usuario que inicie sesión.

38-91. Elementos para la interfaz gráfica de la clase MenuPrincipal.

Texto

Descripción generada automáticamente

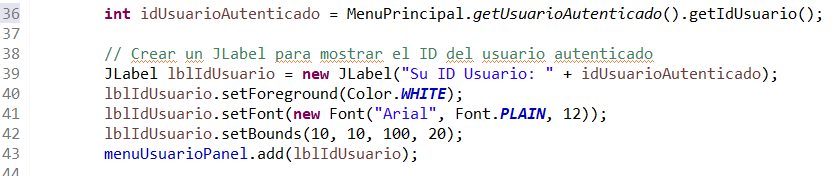
99-100. Se recogen los valores ingresados por el usuario como nombre y contraseña en dos variables.

102-105. Se comprueba si ese usuario existe en la base de datos y que rol tiene dicho usuario. Se almacenan los datos de este usuario en el objeto Usuario que instanciamos anteriormente.

107-127. Se comprueba el rol del usuario que ha iniciado sesión. Si tiene rol de administrador se cierra la ventana de login y se muestra el menú de administrador y si tiene rol de usuario se cierra la ventana de login y se abre el menú de usuario. Se crean instancias de cada una de las clases que contiene dicha interfaz con el menú dependiendo del rol del usuario. Se manejan los errores a la hora de autentificar el usuario.

El resto de métodos validarUsuario(), obtenerRolUsuario() y obtenerUsuario() se encuentran en la misma clase MenuPrincipal() pero no voy a extenderme con su explicación, pero también tiene manejo y control de errores.

## Información ID usuario



Llama a la clase MenuPrincipal y luego al método para autenticar el usuario, luego recoge el id del usuario que ha inciado sesión y lo almacena en una variable cuyo valor se mostrará en el JLabel lablIdUsuario.

## Actualizar Perfil Usuario

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

El método actualizarDatos que es llamado al hacer click en actualizar en ka ventana de actualizar usuario, recibe como parámetros el id de usuario que ha iniciado sesión, y los nuevos valores que quieren subirse a la base de datos.

## Actualizar datos de contacto inmobiliaria

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

# 

# EVALUACIÓN

Una vez analizado y realizado todas las pruebas necesarias, se puede concluir en que la aplicación cumple con los requisitos establecidos por el cliente para un mejor manejo de los datos.

El proceso de normalización ha sido todo un éxito ya que mantiene la integridad referencial, evita duplicados y con el control y el manejo de errores implementados optimizamos su rendimiento.

La base de datos actual puede incrementarse en cualquier momento sin que esto afecte de forma negativa al sistema diseñado como base.

La aplicación puede ampliarse en un futuro si quisiera ofrecer otro tipo de consultas mediante nuevas clases con interfaces gráficas y métodos que puedan realizar y ejecutar dichas consultas.

Se programarán copias de seguridad periódicas para garantizar una mayor seguridad de los datos.

# BIBLIOGRAFÍA

En el siguiente enlace podemos encontrar la api de java en la cual encontraremos información acerca de los paquetes que debemos importar en nuestro proyecto en Eclipse así como el uso de sus métodos:

<https://docs.oracle.com/javase/8/docs/api/>

Enlaces con información acerca de los modelos de bases de datos y normalización:

<https://www.lucidchart.com/pages/es/que-es-un-modelo-de-base-de-datos>

<https://ayudaleyprotecciondatos.es/bases-de-datos/modelos/>

<https://learn.microsoft.com/es-es/office/troubleshoot/access/database-normalization-description>