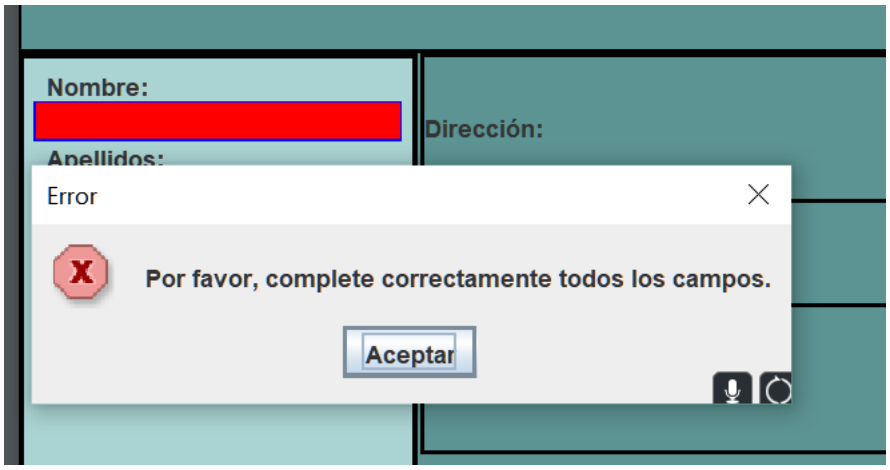
10 REGLAS HEURISTICAS

1. Visibilidad del Estado del Sistema:

- **Implementación:** Uso etiquetas y mensajes de texto para informar al usuario sobre el estado del sistema, como mensajes de confirmación y advertencias.
- **Mejora sugerida:** Podría agregar elementos visuales adicionales, como barras de progreso o iconos.

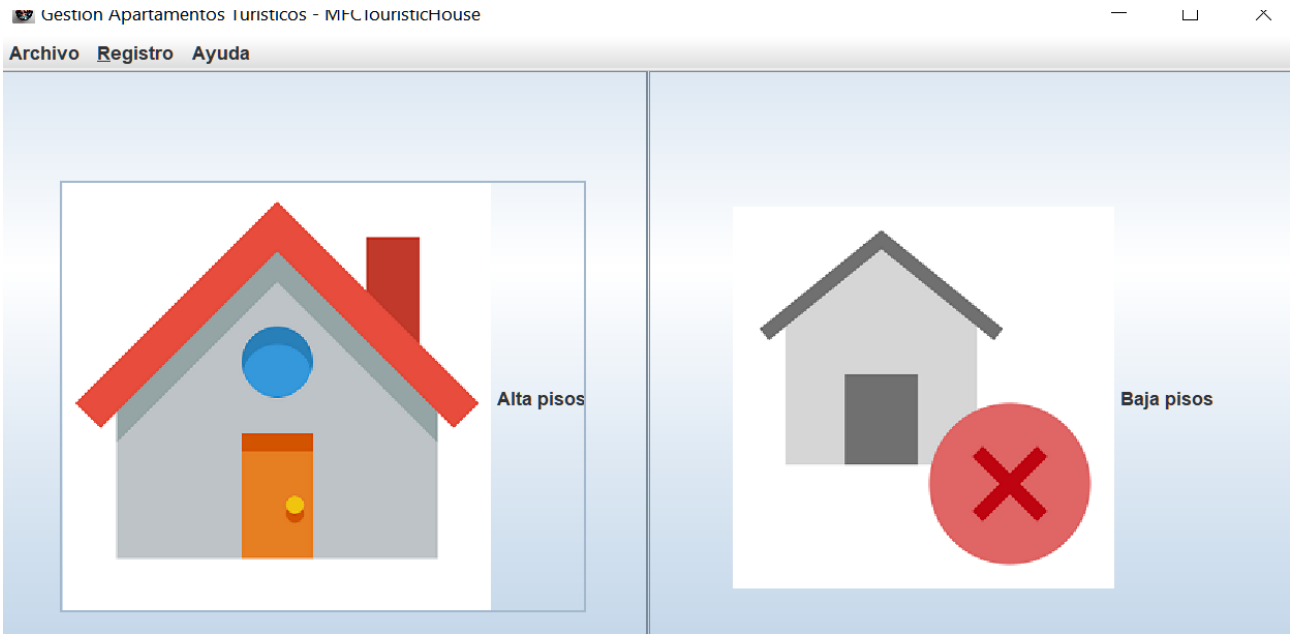
```
nuevo.addActionListener(new ActionListener() {

    @Override
    public void actionPerformed(ActionEvent e) {
        nuevoRegistro();
        JOptionPane.showMessageDialog(VentanaDialogo.this, "Datos vaciados", "Informacion",
                JOptionPane.INFORMATION_MESSAGE);

    }
});

JButton guardar = new JButton("Guardar");
```



2. Coincidencia entre el Sistema y el Mundo Real:

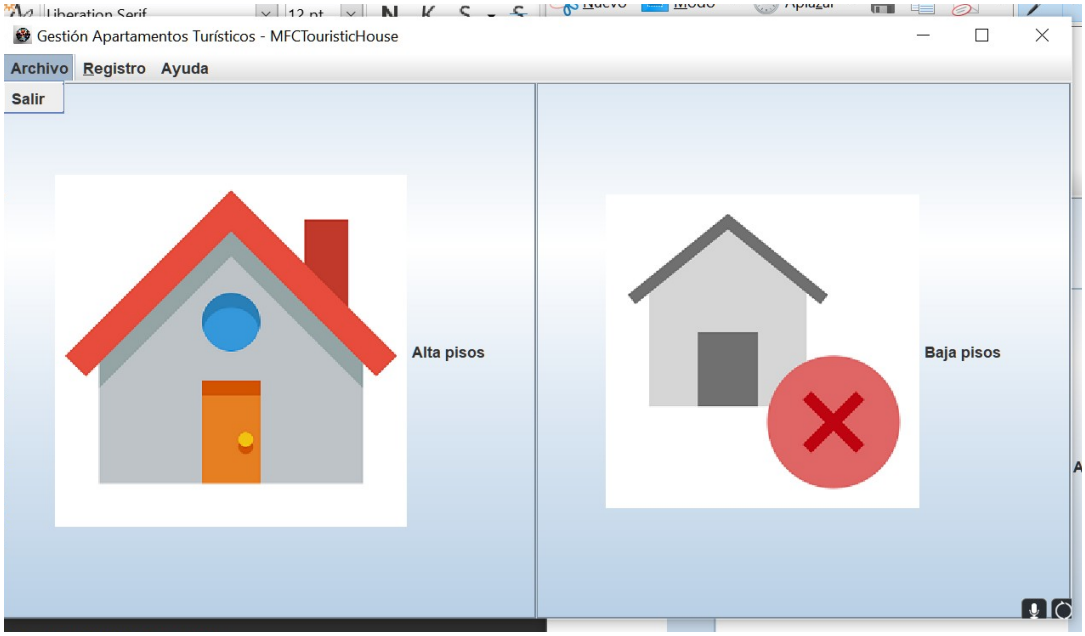
- **Implementación:** Uso nombres y descripciones que se asemejan a conceptos del mundo real (por ejemplo, "Alta Pisos" y "Baja Pisos").
- **Mejora sugerida:** Debería asegurarme de que los términos utilizados sean intuitivos y claros para los usuarios pero ya no tenía más tiempo.

```
JPanel botonAlta = new JPanel();
botonAlta.setLayout(new BorderLayout());
JButton altaPisosButton = new JButton("Alta pisos");
altaPisosButton.setIcon(new ImageIcon(getClass().getResource("/resources/Alta.png")));
altaPisosButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        abrirVentanaAltaPisos();
    }
});


// Creo el boton de baja de pisos
JPanel botonBaja = new JPanel();
JButton bajaPisosButton = new JButton("Baja pisos");
bajaPisosButton.setIcon(new ImageIcon(getClass().getResource("/resources/Baja.png")));
bajaPisosButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // muestro el mensaje de que no está desarrollado
        JOptionPane.showMessageDialog(VentanaPrincipal.this, "Esta opción no se ha desarrollado aún", "Aviso",
                JOptionPane.WARNING_MESSAGE);
    }
});
```



3. Control y Libertad del Usuario:

- **Implementación:** Ofrezco opciones de salida, como el botón "Salir" en el menú "Archivo".
- **Mejora sugerida:** Podría agregar confirmaciones para acciones críticas como salir.




```java
// Añado a la barra el menu Archivo y dentro el item salir
JMenu menuArchivo = new JMenu("Archivo");
JMenuItem salirItem = new JMenuItem("Salir");
salirItem.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        System.exit(0);
    }
});
menuArchivo.add(salirItem);
barraMenu.add(menuArchivo);
```

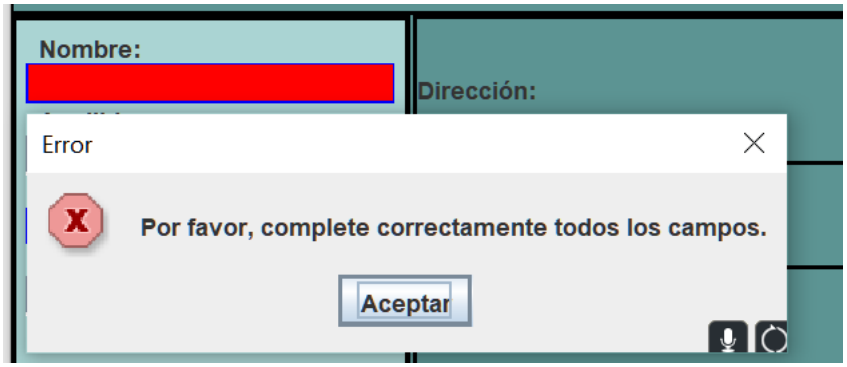
4. Consistencia y Estándares:

- **Implementación:** Sigo un diseño consistente en todo el programa y convenciones estándar de Java para la creación de interfaces gráficas, uso bordes negros para diferenciar los elementos.
- **Mejora sugerida:** Dbería seguir un estilo más visual en vez de bordes negros.

```
panel2 = new Panel2(panel4);
panel2.setBackground(new Color(0xAAD4D3));
panel2.setBorder(BorderFactory.createLineBorder(Color.BLACK, 3));

panel3 = new Panel3();

panel3.setBackground(new Color(0x5C9493));

panel4 = new Panel4();
panel4.setBackground(new Color(0xAAD4D3));
panel4.setBorder(BorderFactory.createLineBorder(Color.BLACK, 3));
```

5. Prevención de Errores:

- **Implementación:** Realizo validaciones en la entrada de datos, como verificar el formato del DNI y el número de teléfono.
- **Mejora sugerida:** debería roporcionar mensajes de error más descriptivos y guía al usuario sobre cómo corregir los errores.



```
 */
public boolean datosIngresados() {
    boolean datosCorrectos = true;

    // Verificar nombre
    if (nombre.getText().isEmpty()) {
        mostrarErrorCampo(nombre);
        datosCorrectos = false;
    }

    // Verificar apellidos
    if (apellidos.getText().isEmpty()) {
        mostrarErrorCampo(apellidos);
        datosCorrectos = false;
    }

    // Verificar DNI
    if (!campoDNI.getText().matches("\\d{8}\\w")) {
        mostrarErrorCampo(campoDNI);
        datosCorrectos = false;
    }

    // Verificar teléfono
    if (!telefono.getText().matches("\\d{9}")) {
        mostrarErrorCampo(telefono);
        datosCorrectos = false;
    }

    return datosCorrectos;
}

// Método para mostrar un mensaje de error en el campo específico
private void mostrarErrorCampo(JComponent campo) {
    campo.setBackground(new Color(0xFF0000));
    campo.setBorder(BorderFactory.createLineBorder(Color.BLUE));
    mostrarMensajeError("Por favor, complete correctamente todos los campos.");
}
```

6. Reconocimiento en lugar de Recuerdo:

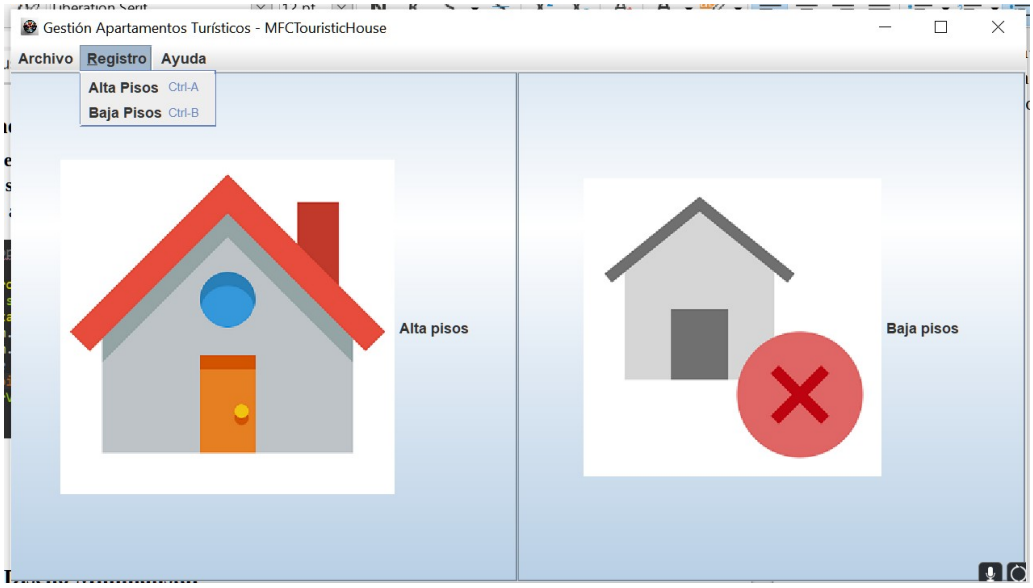
- **Implementación:** Uso menús y botones con etiquetas descriptivas para que los usuarios no tengan que recordar comandos específicos.
- **Mejora sugerida:** Podria agregar información contextual o sugerencias en línea para facilitar aún más la interacción.

```
JButton toolkiBoton = new JButton("Boton en pruebas");
toolkiBoton.setToolTipText("Este boton está en pruebas aún");

toolkiBoton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        JOptionPane.showMessageDialog(VentanaDialogo.this, "Botón en pruebas", "Información",
                JOptionPane.INFORMATION_MESSAGE);
    }
});
```

7. Flexibilidad y Eficiencia de Uso:

- **Implementación:** Uso atajos de teclado y accesos directos para acciones frecuentes.
- **Mejora sugerida:** Podría agregar más atajos de teclado y personalización de la interfaz para usuarios avanzados.

```
// Añado la opcion de registro, con lo skey listener para la alta y baja de
// pisos
JMenu registroMenu = new JMenu("Registro");
registroMenu.setMnemonic('R');
JMenuItem altaPisosItem = new JMenuItem("Alta Pisos");
altaPisosItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_A, InputEvent.CTRL_DOWN_MASK));
altaPisosItem.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        abrirVentanaAltaPisos();
    }
});
```



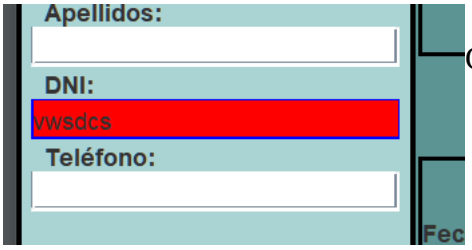
8. Estética y Diseño Minimalista:

- **Implementación:** Uso colores y diseño para hacer la interfaz visualmente agradable.

9. Ayuda a los Usuarios a Reconocer, Diagnosticar y Recuperarse de Errores:

- **Implementación:** Muestro mensajes de error cuando hay problemas y doy información sobre cómo solucionarlos.

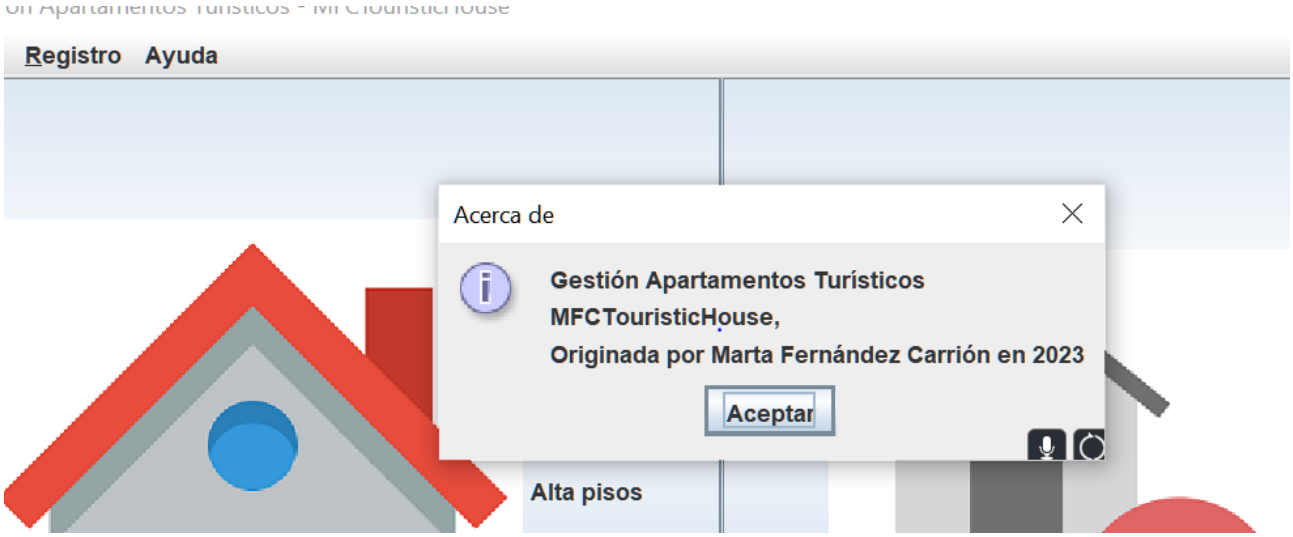
```
// Método para mostrar un mensaje de error
private void mostrarMensajeError(String mensaje) {
    JOptionPane.showMessageDialog(this, mensaje, "Error", JOptionPane.ERROR_MESSAGE);
}

@Override
```



Cuando no ingresa bien los datos se cambia a color rojo

10. Ayuda y Documentación:

- **Implementación:** Uso elementos de ayuda, como el menú "Ayuda" con la opción "Acerca de...".




```java
// Añado el menu ayuda dond evoy a mostrar la info de la empresa
JMenu ayudaMenu = new JMenu("Ayuda");
JMenuItem acercaDeItem = new JMenuItem("Acerca de...");
acercaDeItem.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // Muestro la informacion que se pide en ayuda, usando showMessageDialog
        JOptionPane.showMessageDialog(VentanaPrincipal.this,
                "Gestión Apartamentos Turísticos\nMFCTouristicHouse, \nOriginada por Marta Fernández Carrión en 2023",
                "Acerca de", JOptionPane.INFORMATION_MESSAGE);
    }
});
ayudaMenu.add(acercaDeItem);
barraMenu.add(ayudaMenu);
```

Gama de Colores Utilizada:

- Uso colores como azul y verde para la interfaz, con un enfoque en tonos suaves y amigables, además de que dan una imagen de tranquilidad y naturaleza que es lo quiero para mi alquiler turistico, que al igual que el logo, se encuentra en la naturaleza.