Paso 2: Módulo del Dashboard

a) Explicación Conceptual: "¿Por qué hacemos esto?"

El Dashboard es la pantalla principal que el usuario ve después de un login exitoso. Su propósito es:

1. **Proveer una vista general:** Mostrar información resumida relevante (ej: número de perritos disponibles, últimas adopciones).
2. **Navegación principal:** Servir como punto de acceso a las diferentes funcionalidades de la aplicación (CRUDs, gestión de adopciones) según el rol del usuario.
3. **Personalización según rol:** El contenido y las opciones disponibles en el Dashboard pueden variar dependiendo si el usuario es `admin` o `usuario`.

Arquitectura:

- **Vista:** `DashboardFrame.java` (un JFrame que contendrá otros paneles o componentes).
- **Controlador/Lógica:** La lógica de qué mostrar y qué botones habilitar estará principalmente en `DashboardFrame.java`, basándose en el `usuarioLogueado`. Podríamos tener clases de servicio específicas si el Dashboard necesita cargar datos complejos, pero para empezar, la lógica puede residir en el frame.

Buenas Prácticas:

- **Diseño intuitivo:** El Dashboard debe ser fácil de entender y usar.
- **Acceso rápido:** Las funciones más comunes deben ser fácilmente accesibles.
- **Seguridad por rol:** Asegurarse de que los usuarios solo vean y accedan a las funciones permitidas para su rol.

b) Snippet de Código:

1. DashboardFrame.java (Vista - JFrame):

En el paquete com.patitasfelices.vista.

Crea un JFrame Form llamado DashboardFrame.java.

- **Diseño en NetBeans GUI Builder:**
  - **JMenuBar (Barra de Menú):**
    - JMenu "Archivo"
      - JMenuItem "Cerrar Sesión" (nómbralo menuItemCerrarSesion)
      - JMenuItem "Salir" (nómbralo menuItemSalir)
    - JMenu "Gestión" (este menú podría ser visible solo para admin)
      - JMenuItem "Gestionar Usuarios" (nómbralo menuItemGestionUsuarios)
      - JMenuItem "Gestionar Perritos" (nómbralo menuItemGestionPerritos)
      - JMenuItem "Gestionar Adoptantes" (nómbralo menuItemGestionAdoptantes)
      - JMenuItem "Gestionar Adopciones" (nómbralo menuItemGestionAdopciones)
  - **JPanel para contenido principal:** Un panel grande donde podríamos mostrar resúmenes o cambiar vistas. Por ahora, puede tener algunos JLabel para mostrar información del usuario y estadísticas simples.
    - JLabel lblBienvenida: "Bienvenido, [Nombre de Usuario] ([Rol])"

        - JLabel lblPerritosDisponibles: "Perritos Disponibles: [Número]"
        - JLabel lblAdopcionesRecientes: "Últimas Adopciones: [Número]" (opcional, más avanzado)
    - **Botones de acceso rápido (opcional, pueden estar en el panel principal):**
        - JButton btnVerPerritos: "Ver Perritos Disponibles"
        - JButton btnMisAdopciones (si es rol usuario y tiene historial)
        - JButton btnAdminUsuarios (si es rol admin): "Administrar Usuarios"
- **Código para DashboardFrame.java:**

```
package com.patitasfelices.vista;

import com.patitasfelices.modelo.Usuario;
// Importar las otras ventanas de CRUDs cuando las creemos
// import com.patitasfelices.vista.crudusuarios.UsuariosFrame;
// import com.patitasfelices.vista.crudperritos.PerritosFrame;
// ...etc.

import javax.swing.JOptionPane;

public class DashboardFrame extends javax.swing.JFrame {

    private Usuario usuarioLogueado;

    /**
     * Creates new form DashboardFrame
     * @param usuario El usuario que ha iniciado sesión.
     */
```

```
    public DashboardFrame(Usuario usuario) {
        initComponents();
        this.usuarioLogueado = usuario;
        configurarDashboardSegunRol();
        cargarDatosResumen(); // Método para cargar estadísticas
        this.setLocationRelativeTo(null); // Centrar ventana
        this.setTitle("Patitas Felices - Dashboard (" + usuario.getNombreUsuario() +
")");
    }

    private void configurarDashboardSegunRol() {
        lblBienvenida.setText("Bienvenido, " + usuarioLogueado.getNombreUsuario()
+
                              " (Rol: " + usuarioLogueado.getRol() + ")");

        if ("admin".equals(usuarioLogueado.getRol())) {
            // Admin ve todo
            menuGestion.setVisible(true);
            // Si tienes botones específicos de admin, habilítalos aquí
            // btnAdminUsuarios.setVisible(true);
        } else if ("usuario".equals(usuarioLogueado.getRol())) {
            // Usuario normal tiene vistas limitadas
            menuGestion.setVisible(false); // Ocultar menú de gestión completo
            // O podrías deshabilitar items específicos del menú:
            // menuItemGestionUsuarios.setEnabled(false);
            // menuItemGestionAdoptantes.setEnabled(false);
            // etc.

            // btnAdminUsuarios.setVisible(false);
```

```
            // btnVerPerritos.setVisible(true); // Asumiendo que todos pueden ver
perritos
        }
    }

    private void cargarDatosResumen() {
        // Aquí iría la lógica para obtener datos de la BD
        // Por ejemplo, contar perritos disponibles
        // int countDisponibles = perritoDAO.contarPerritosPorEstado("disponible");
        // lblPerritosDisponibles.setText("Perritos Disponibles: " + countDisponibles);
        // Por ahora, valores placeholder:
        lblPerritosDisponibles.setText("Perritos Disponibles: Cargando...");
        lblAdopcionesRecientes.setText("Adopciones Recientes: Cargando...");
        // TODO: Implementar la carga real de estos datos desde los DAOs
correspondientes.
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents
    private void initComponents() {

        pnlPrincipal = new javax.swing.JPanel();
        lblBienvenida = new javax.swing.JLabel();
        lblPerritosDisponibles = new javax.swing.JLabel();
```

```
        lblAdopcionesRecientes = new javax.swing.JLabel();
        btnVerPerritosDisponibles = new javax.swing.JButton();
        jMenuBar1 = new javax.swing.JMenuBar();
        menuArchivo = new javax.swing.JMenu();
        menuItemCerrarSesion = new javax.swing.JMenuItem();
        menuItemSalir = new javax.swing.JMenuItem();
        menuGestion = new javax.swing.JMenu();
        menuItemGestionUsuarios = new javax.swing.JMenuItem();
        menuItemGestionPerritos = new javax.swing.JMenuItem();
        menuItemGestionAdoptantes = new javax.swing.JMenuItem();
        menuItemGestionAdopciones = new javax.swing.JMenuItem();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);


pnlPrincipal.setBorder(javax.swing.BorderFactory.createTitledBorder("Resumen"));

        lblBienvenida.setFont(new java.awt.Font("Segoe UI", 1, 14)); // NOI18N
        lblBienvenida.setText("Bienvenido, [Usuario] ([Rol])");

        lblPerritosDisponibles.setText("Perritos Disponibles: N/A");

        lblAdopcionesRecientes.setText("Adopciones Recientes: N/A");

        btnVerPerritosDisponibles.setText("Ver Perritos Disponibles");
        btnVerPerritosDisponibles.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnVerPerritosDisponiblesActionPerformed(evt);
            }
```

```
        });

        javax.swing.GroupLayout pnlPrincipalLayout = new
javax.swing.GroupLayout(pnlPrincipal);
        pnlPrincipal.setLayout(pnlPrincipalLayout);
        pnlPrincipalLayout.setHorizontalGroup(

pnlPrincipalLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
            .addGroup(pnlPrincipalLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(pnlPrincipalLayout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
                    .addComponent(lblBienvenida,
javax.swing.GroupLayout.DEFAULT_SIZE, 360, Short.MAX_VALUE)
                    .addComponent(lblPerritosDisponibles)
                    .addComponent(lblAdopcionesRecientes)
                    .addComponent(btnVerPerritosDisponibles))
                .addContainerGap())
        );
        pnlPrincipalLayout.setVerticalGroup(

pnlPrincipalLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
            .addGroup(pnlPrincipalLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(lblBienvenida)
                .addGap(18, 18, 18)
                .addComponent(lblPerritosDisponibles)
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(lblAdopcionesRecientes)
                .addGap(18, 18, 18)
                .addComponent(btnVerPerritosDisponibles)
                .addContainerGap(123, Short.MAX_VALUE))
        );

        menuArchivo.setText("Archivo");

        menuItemCerrarSesion.setText("Cerrar Sesión");
        menuItemCerrarSesion.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                menuItemCerrarSesionActionPerformed(evt);
            }
        });
        menuArchivo.add(menuItemCerrarSesion);

        menuItemSalir.setText("Salir");
        menuItemSalir.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                menuItemSalirActionPerformed(evt);
            }
        });
        menuArchivo.add(menuItemSalir);

        jMenuBar1.add(menuArchivo);

        menuGestion.setText("Gestión");
```

```
        menuItemGestionUsuarios.setText("Gestionar Usuarios");
        menuItemGestionUsuarios.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                menuItemGestionUsuariosActionPerformed(evt);
            }
        });
        menuGestion.add(menuItemGestionUsuarios);

        menuItemGestionPerritos.setText("Gestionar Perritos");
        menuItemGestionPerritos.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                menuItemGestionPerritosActionPerformed(evt);
            }
        });
        menuGestion.add(menuItemGestionPerritos);

        menuItemGestionAdoptantes.setText("Gestionar Adoptantes");
        menuItemGestionAdoptantes.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                menuItemGestionAdoptantesActionPerformed(evt);
            }
        });
        menuGestion.add(menuItemGestionAdoptantes);

        menuItemGestionAdopciones.setText("Gestionar Adopciones");
```

```
        menuItemGestionAdopciones.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                menuItemGestionAdopcionesActionPerformed(evt);
            }
        });
        menuGestion.add(menuItemGestionAdopciones);

        jMenuBar1.add(menuGestion);

        setJMenuBar(jMenuBar1);

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(pnlPrincipal, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(pnlPrincipal, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addContainerGap())
```

```
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void menuItemSalirActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_menuItemSalirActionPerformed
        // Confirmación antes de salir
        int confirm = JOptionPane.showConfirmDialog(this,
                "¿Estás seguro de que quieres salir de la aplicación?",
                "Confirmar Salida",
                JOptionPane.YES_NO_OPTION,
                JOptionPane.QUESTION_MESSAGE);
        if (confirm == JOptionPane.YES_OPTION) {
            System.exit(0);
        }
    }//GEN-LAST:event_menuItemSalirActionPerformed

    private void menuItemCerrarSesionActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_menuItemCerrarSesionActionPerformed
        int confirm = JOptionPane.showConfirmDialog(this,
                "¿Estás seguro de que quieres cerrar sesión?",
                "Confirmar Cierre de Sesión",
                JOptionPane.YES_NO_OPTION,
                JOptionPane.QUESTION_MESSAGE);
        if (confirm == JOptionPane.YES_OPTION) {
            LoginFrame.usuarioLogueado = null; // Limpiar usuario de la "sesión"
            this.dispose(); // Cerrar Dashboard
            new LoginFrame().setVisible(true); // Mostrar Login
        }
```

```
    }//GEN-LAST:event_menuItemCerrarSesionActionPerformed

    private void
menuItemGestionUsuariosActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_menuItemGestionUsuariosActionPerformed
        // Aquí abrirías la ventana de gestión de usuarios
        // UsuariosFrame usuariosFrame = new
UsuariosFrame(this.usuarioLogueado); // Pasar usuario por si se necesita para
permisos internos
        // usuariosFrame.setVisible(true);
        JOptionPane.showMessageDialog(this, "Funcionalidad 'Gestionar Usuarios'
aún no implementada.", "En Desarrollo",
JOptionPane.INFORMATION_MESSAGE);
        // TODO: Abrir el JFrame de CRUD Usuarios
    }//GEN-LAST:event_menuItemGestionUsuariosActionPerformed

    private void
menuItemGestionPerritosActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_menuItemGestionPerritosActionPerformed
        JOptionPane.showMessageDialog(this, "Funcionalidad 'Gestionar Perritos'
aún no implementada.", "En Desarrollo",
JOptionPane.INFORMATION_MESSAGE);
        // TODO: Abrir el JFrame de CRUD Perritos
    }//GEN-LAST:event_menuItemGestionPerritosActionPerformed

    private void
menuItemGestionAdoptantesActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_menuItemGestionAdoptantesActionPerformed
```

```
        JOptionPane.showMessageDialog(this, "Funcionalidad 'Gestionar
Adoptantes' aún no implementada.", "En Desarrollo",
JOptionPane.INFORMATION_MESSAGE);
        // TODO: Abrir el JFrame de CRUD Adoptantes
    }//GEN-LAST:event_menuItemGestionAdoptantesActionPerformed

    private void
menuItemGestionAdopcionesActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_menuItemGestionAdopcionesActionPerformed
        JOptionPane.showMessageDialog(this, "Funcionalidad 'Gestionar
Adopciones' aún no implementada.", "En Desarrollo",
JOptionPane.INFORMATION_MESSAGE);
        // TODO: Abrir el JFrame de Gestión de Adopciones
    }//GEN-LAST:event_menuItemGestionAdopcionesActionPerformed

    private void
btnVerPerritosDisponiblesActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_btnVerPerritosDisponiblesActionPerformed
        // Este botón podría ser accesible para todos los roles.
        // Abriría una vista (posiblemente parte del CRUD Perritos pero en modo solo
lectura para 'usuario')
        // PerritosDisponiblesFrame perritosDispFrame = new
PerritosDisponiblesFrame();
        // perritosDispFrame.setVisible(true);
        JOptionPane.showMessageDialog(this, "Funcionalidad 'Ver Perritos
Disponibles' aún no implementada.", "En Desarrollo",
JOptionPane.INFORMATION_MESSAGE);
        // TODO: Abrir una vista de Perritos Disponibles (podría ser el mismo CRUD
Perritos con funcionalidad limitada)
    }//GEN-LAST:event_btnVerPerritosDisponiblesActionPerformed
```

```
public static void main(String args[]) {

        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                //new DashboardFrame().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton btnVerPerritosDisponibles;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JLabel lblAdopcionesRecientes;
    private javax.swing.JLabel lblBienvenida;
    private javax.swing.JLabel lblPerritosDisponibles;
    private javax.swing.JMenu menuArchivo;
    private javax.swing.JMenu menuGestion;
    private javax.swing.JMenuItem menuItemCerrarSesion;
    private javax.swing.JMenuItem menuItemGestionAdoptantes;
    private javax.swing.JMenuItem menuItemGestionAdopciones;
    private javax.swing.JMenuItem menuItemGestionPerritos;
    private javax.swing.JMenuItem menuItemGestionUsuarios;
    private javax.swing.JMenuItem menuItemSalir;
    private javax.swing.JPanel pnlPrincipal;
    // End of variables declaration//GEN-END:variables
}
```

2. Modificación en LoginFrame.java para abrir el Dashboard:

Abre LoginFrame.java y modifica el método btnIngresarActionPerformed para que, tras un login exitoso, cierre el LoginFrame y abra el DashboardFrame.

```
// En LoginFrame.java, dentro de btnIngresarActionPerformed, después de la autenticación exitosa:
// ...

if (usuario != null) {

        usuarioLogueado = usuario; // Guardar usuario en la "sesión"

        JOptionPane.showMessageDialog(this,

            "¡Bienvenido " + usuario.getNombreUsuario() + "! Rol: " + usuario.getRol(),

            "Login Exitoso", JOptionPane.INFORMATION_MESSAGE);

        lblMensaje.setText(""); // Limpiar mensaje de error

        System.out.println("Login exitoso para: " + usuario.getNombreUsuario() + " con rol: " + usuario.getRol());

        // Abrimos el Dashboard y cerramos el Login

        DashboardFrame dashboard = new DashboardFrame(usuarioLogueado); // Pasa el usuario al dashboard

        dashboard.setVisible(true);

        this.dispose(); // Cierra la ventana de login

    } else {
```

```
            // JOptionPane.showMessageDialog(this, "Nombre de usuario o
contraseña incorrectos.", "Error de Autenticación",
JOptionPane.ERROR_MESSAGE);

            lblMensaje.setText("Usuario o contraseña incorrectos.");

            txtContrasena.setText(""); // Limpiar campo de contraseña

            txtContrasena.requestFocusInWindow(); // Devolver foco a contraseña

        }
```

c) Checklist:

- **DashboardFrame.java (JFrame Form) creado en com.patitasfelices.vista.**
- **JMenuBar con menús y JMenuItem añadidos a DashboardFrame.**
- **JPanel con JLabel para bienvenida y resúmenes básicos añadido.**
- **Constructor de DashboardFrame acepta un objeto Usuario.**
- **Método configurarDashboardSegunRol() implementado para mostrar/ocultar/habilitar componentes según el rol del usuarioLogueado.**
- **Método cargarDatosResumen() (placeholder por ahora) definido.**
- **LoginFrame.java modificado para abrir DashboardFrame y cerrarse a sí mismo tras un login exitoso, pasando el usuarioLogueado.**

- **Prueba de Flujo:**
  - Ejecuta `LoginFrame.java`.
  - Ingresa como `admin`. Verifica que el `DashboardFrame` se abre.
  - Verifica que el `lblBienvenida` muestra el nombre y rol correctos.
  - Verifica que el menú "Gestión" y sus ítems están visibles/habilitados.
  - Cierra sesión. Debería volver a `LoginFrame`.
  - Ingresa como `usuario`. Verifica que el `DashboardFrame` se abre.
  - Verifica que el `lblBienvenida` muestra el nombre y rol correctos.
  - Verifica que el menú "Gestión" está oculto o sus ítems relevantes deshabilitados.
  - Prueba "Salir" (con confirmación). Debería cerrar la aplicación.
- Los `JMenuItem` para gestión (Usuarios, Perritos, etc.) tienen placeholders (ej: `JOptionPane`) para indicar que aún no están implementados.

Mejoras UX y Tests Sugeridos:

Mejoras UX:

1. **Confirmación al cerrar sesión y salir:** Ya implementado con `JOptionPane.showConfirmDialog`. Esto evita cierres accidentales.
2. **Iconos en menús y botones:** Añadir pequeños iconos puede hacer la interfaz más atractiva y fácil de escanear visualmente.
   - **NetBeans Tip:** En las propiedades de `JMenuItem` o `JButton`, busca la propiedad `icon`. Puedes añadir imágenes a tu proyecto (ej: en un paquete `com.patitasfelices.recursos.iconos`) y luego seleccionarlas.
3. **Estado "Cargando..." para datos del Dashboard:** Buena práctica mientras se obtienen los datos reales para el resumen. (Ya añadido como placeholder).

4. **Panel Central Dinámico (Avanzado):** En lugar de abrir nuevos JFrames para cada CRUD, el Dashboard podría tener un panel central que cambie su contenido (usando `CardLayout` o reemplazando JPanels) para mostrar las diferentes interfaces de gestión. Esto crea una sensación de aplicación de una sola ventana (Single Page Application - SPA, pero en escritorio). Lo haremos más simple con JFrames separados por ahora.

Tests Básicos (Conceptuales):

1. **Test de Visibilidad de Componentes (Admin):**
   - **Acción:** Login como `admin`.
   - **Resultado Esperado:** `DashboardFrame` muestra el menú "Gestión" y todos sus ítems están habilitados. Los botones/secciones específicas de admin son visibles.
2. **Test de Visibilidad de Componentes (Usuario):**
   - **Acción:** Login como `usuario`.
   - **Resultado Esperado:** `DashboardFrame` NO muestra el menú "Gestión" (o tiene ítems deshabilitados como "Gestionar Usuarios"). Los botones/secciones específicas de admin están ocultos. El botón "Ver Perritos Disponibles" es visible.
3. **Test de Cierre de Sesión:**
   - **Acción:** Desde el Dashboard, seleccionar "Archivo" > "Cerrar Sesión" y confirmar.
   - **Resultado Esperado:** `DashboardFrame` se cierra, `LoginFrame.usuarioLogueado` es `null`, y `LoginFrame` se muestra nuevamente.
4. **Test de Salida de Aplicación:**
   - **Acción:** Desde el Dashboard, seleccionar "Archivo" > "Salir" y confirmar.

- **Resultado Esperado:** La aplicación se cierra completamente.

¡Hemos creado el esqueleto del Dashboard! Ya podemos autenticar usuarios y dirigirlos a una pantalla principal que se adapta a su rol. Lo siguiente es empezar a construir los módulos CRUD. El primero y fundamental suele ser el **CRUD de Usuarios**, ya que los administradores necesitarán gestionar quién puede acceder y con qué permisos (aunque nuestros roles son simples por ahora).