# Documentación Fase 3: Integración de CRUDL con Interfaces Gráficas

### Lavadero App - Sistema de Gestión

## Índice

1. [Introducción](#introducción)
2. [Arquitectura del Sistema](#arquitectura-del-sistema)
3. [Componentes Gráficos](#componentes-gráficos)
4. [Integración CRUDL-GUI](#integración-crudl-gui)
5. [Flujo de Datos](#flujo-de-datos)
6. [Patrones de Diseño](#patrones-de-diseño)
7. [Guía de Implementación](#guía-de-implementación)

## 1. Introducción

La tercera fase del proyecto Lavadero App se centra en la integración de las operaciones CRUDL (Create, Read, Update, Delete, List) con una interfaz gráfica de usuario moderna y funcional. Esta fase representa la evolución del sistema desde una interfaz de línea de comandos a una aplicación de escritorio completa.

### 1.1 Objetivos

* Implementar una interfaz gráfica intuitiva y amigable
* Integrar las operaciones CRUDL existentes con componentes visuales
* Mejorar la experiencia de usuario
* Mantener la consistencia de datos y validaciones

## 2. Arquitectura del Sistema

### 2.1 Capas de la Aplicación

1. **Capa de Presentación (GUI)**
   * Componentes Swing
   * Componentes personalizados
   * Gestión de eventos
2. **Capa de Lógica de Negocio**
   * Controladores
   * Validadores
   * Gestores de datos
3. **Capa de Persistencia**
   * SQLite
   * Gestión de archivos JSON

### 2.2 Diagrama de Arquitectura

[GUI (Swing)] <-> [Controladores] <-> [CRUDL] <-> [Base de Datos]

## 3. Componentes Gráficos

### 3.1 Componentes Base (Swing)

* **JFrame**: Ventana principal
* **JPanel**: Contenedores modulares
* **JMenuBar**: Navegación principal
* **JScrollPane**: Listas desplazables
* **JButton**: Acciones
* **JLabel**: Etiquetas informativas
* **JDialog**: Formularios modales
* **JComboBox**: Selectores

### 3.2 Componentes Personalizados

1. **Input**
   * Validación integrada
   * Estilos consistentes
   * Mensajes de error
2. **LabelValue**
   * Presentación de datos
   * Formato estandarizado
3. **IconDetail**
   * Iconos informativos
   * Contadores dinámicos
4. **InputIcon**
   * Campos de búsqueda
   * Retroalimentación visual
5. **Boton y BotonModerno**
   * Estilos personalizados
   * Estados interactivos
6. **ComboItem**
   * Selección mejorada
   * Datos estructurados
7. **MenuItem**
   * Navegación mejorada
   * Diseño consistente

## 4. Integración CRUDL-GUI

### 4.1 Create (Crear)

// Ejemplo de integración para crear un nuevo cliente  
public void crearCliente() {  
 JDialog dialog = new JDialog();  
 Input nombre = new Input("Nombre");  
 Input identificacion = new Input("Identificación");  
 BotonModerno guardar = new BotonModerno("Guardar");  
   
 guardar.addActionListener(e -> {  
 Cliente cliente = new Cliente(nombre.getText());  
 clienteCRUDL.agregar(cliente);  
 actualizarLista();  
 });  
}

### 4.2 Read (Leer)

// Ejemplo de visualización de detalles  
public void mostrarDetalles(String id) {  
 Cliente cliente = clienteCRUDL.buscar(id);  
 JDialog detalles = new JDialog();  
 detalles.add(new LabelValue("Nombre", cliente.getNombre()));  
 detalles.add(new LabelValue("ID", cliente.getId()));  
}

### 4.3 Update (Actualizar)

// Ejemplo de actualización con validación  
public void actualizarCliente(Cliente cliente) {  
 JDialog dialog = new JDialog();  
 Input nombre = new Input("Nombre", cliente.getNombre());  
   
 dialog.add(new BotonModerno("Guardar", e -> {  
 if (nombre.isValid()) {  
 cliente.setNombre(nombre.getText());  
 clienteCRUDL.editar(cliente);  
 }  
 }));  
}

### 4.4 Delete (Eliminar)

// Ejemplo de confirmación de eliminación  
public void eliminarCliente(String id) {  
 JDialog confirmar = new JDialog();  
 confirmar.add(new JLabel("¿Eliminar cliente?"));  
 confirmar.add(new BotonModerno("Confirmar", e -> {  
 clienteCRUDL.eliminar(id);  
 actualizarLista();  
 }));  
}

### 4.5 List (Listar)

// Ejemplo de lista con búsqueda  
public void listarClientes() {  
 JPanel panel = new JPanel();  
 InputIcon busqueda = new InputIcon("Buscar");  
 JScrollPane lista = new JScrollPane();  
   
 busqueda.onTextChange(texto -> {  
 ArrayList<Cliente> filtrados = clienteCRUDL.buscarPorNombre(texto);  
 actualizarLista(filtrados);  
 });  
}

## 5. Flujo de Datos

### 5.1 Proceso Típico

1. Usuario interactúa con la GUI
2. Validación en componentes personalizados
3. Llamada a métodos CRUDL
4. Actualización de la base de datos
5. Retroalimentación visual al usuario

### 5.2 Manejo de Errores

* Validación en tiempo real
* Mensajes de error contextuales
* Prevención de datos inválidos

## 6. Patrones de Diseño

### 6.1 MVC (Model-View-Controller)

* **Modelo**: Clases CRUDL y entidades
* **Vista**: Componentes Swing y personalizados
* **Controlador**: Manejadores de eventos

### 6.2 Observer

* Actualización automática de vistas
* Sincronización de datos

### 6.3 Factory

* Creación de componentes personalizados
* Estandarización de interfaces

## 7. Guía de Implementación

### 7.1 Creación de Nuevos Módulos

1. Crear clase que extienda JPanel
2. Implementar componentes necesarios
3. Integrar con CRUDL existente
4. Agregar validaciones
5. Implementar retroalimentación visual

### 7.2 Mejores Prácticas

* Usar componentes personalizados
* Implementar validaciones en tiempo real
* Mantener consistencia visual
* Documentar cambios
* Realizar pruebas de usabilidad

### 7.3 Ejemplo de Implementación Completa

public class VentanaCliente extends JPanel {  
 private ClienteCRUDL crudl;  
 private JScrollPane listado;  
 private InputIcon buscador;  
 private BotonModerno btnAgregar;  
   
 public VentanaCliente() {  
 // Inicialización  
 crudl = new ClienteCRUDL();  
   
 // Configuración de componentes  
 buscador = new InputIcon("Buscar cliente");  
 btnAgregar = new BotonModerno("Nuevo Cliente");  
 listado = new JScrollPane();  
   
 // Eventos  
 btnAgregar.addActionListener(e -> mostrarFormularioNuevo());  
 buscador.onTextChange(this::filtrarClientes);  
   
 // Layout y estilos  
 setLayout(new BorderLayout());  
 add(buscador, BorderLayout.NORTH);  
 add(listado, BorderLayout.CENTER);  
 add(btnAgregar, BorderLayout.SOUTH);  
 }  
   
 private void mostrarFormularioNuevo() {  
 JDialog dialog = new JDialog();  
 FormularioCliente form = new FormularioCliente();  
   
 form.onGuardar(cliente -> {  
 crudl.agregar(cliente);  
 actualizarLista();  
 dialog.dispose();  
 });  
   
 dialog.add(form);  
 dialog.setVisible(true);  
 }  
   
 private void filtrarClientes(String texto) {  
 ArrayList<Cliente> filtrados = crudl.buscarPorNombre(texto);  
 actualizarLista(filtrados);  
 }  
   
 private void actualizarLista() {  
 // Actualizar JScrollPane con nuevos datos  
 }  
}

## Conclusiones

La integración de las operaciones CRUDL con la interfaz gráfica ha resultado en una aplicación más robusta y amigable. Los componentes personalizados aseguran una experiencia de usuario consistente, mientras que la arquitectura modular facilita el mantenimiento y la extensión del sistema.

### Beneficios Logrados

1. Mejor experiencia de usuario
2. Mayor eficiencia en operaciones
3. Reducción de errores
4. Facilidad de mantenimiento
5. Escalabilidad mejorada

### Próximos Pasos

* Implementar más validaciones
* Mejorar el rendimiento
* Agregar más funcionalidades
* Optimizar la interfaz de usuario
* Realizar pruebas exhaustivas

## Referencias

* Documentación de Java Swing
* Patrones de Diseño GUI
* Mejores prácticas CRUDL
* Guías de usabilidad

*Documento generado el 15 de junio de 2025*  
*Versión 1.0*