

## **Actividad 6 – Aplicativo de Arquitectura de Software**

**Anyi Dayana Idrobo Parra**

**Docente Joaquín Sánchez**

**Corp. Universitaria Iberoamericana**

**Arquitectura de Software**

**Facultad de Ingeniería**

**Ingeniería de Software**

**Palmira, 2024**

## Actividad 6 – Aplicativo de Arquitectura de Software

El desarrollo de Software se ha convertido en una herramienta estratégica para el impulso en diferentes ámbitos, en el mundo corporativo, a través de los avances tecnológicos, las empresas pueden proporcionar mayor satisfacción a sus clientes, mejorando la eficiencia de sus procesos productivos. Sin embargo, al ser un mercado tan competitivo, las organizaciones deben preocuparse por aspectos diferentes a su producción, uno de estos es mejorar la experiencia de sus clientes o usuario, para ello se debe tener data específica y clave para generar estrategias enfocadas en los tipos de clientes, por esta razón el objetivo de este proyecto es presentar un pequeño esquema para captar datos de usuarios, que a mayor escala podría ampliarse con la funcionalidad de un CRM.

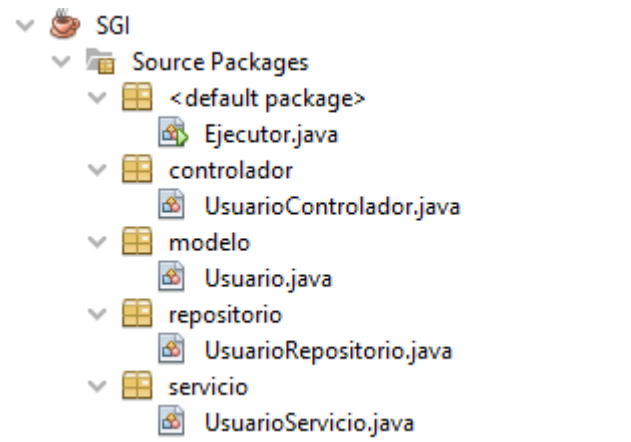
En el entorno de trabajo NetBeans, haciendo uso del lenguaje de programación JAVA, se crea el proyecto SGI (Sistema de Gestión Interna), su desarrollo está basado en la arquitectura Cliente – Servidor, contemplado la creación de paquetes y sus respectivas clases que permiten tener los códigos seccionados y se evidencia la interacción entre el servidor y el cliente.

## Desarrollo del Software

1. Se crea el proyecto SGI (Sistema de Gestión Interno), con los paquetes y sus respectivas clases: controlador - UsuarioControlador, modelo - Usuario, repositorio - UsuarioRepositorio, servicio - UsuarioServicio y el default que contiene a la clase principal Ejecutor.

### Ilustración 1

Proyecto SGI



**Nota:** la imagen contiene los paquetes y clases creados para el proyecto SGI. Fuente: creación propia en la herramienta NetBeans

2. En la clase Usuario se crearon los atributos identificación, nombre y correoElectronico de la clase con el tipo de dato respectivo, estos son los datos que se solicitarán para los nuevos usuarios. adicionalmente, se crea el método Constructor que nos permite inicializar los atributos, se aplican los métodos Getter and Setter, Getter permite que obtener datos privados y Setter permite modificarlos.

## Ilustración 2

### Clase Usuario

```
package modelo;
// crea los atributos
public class Usuario {
    private Long identificacion = 1L;
    private String nombre;
    private String correoElectronico;
    // aplica el método constructor
    public Usuario(Long identificacion, String nombre, String correoElectronico) {
        this.identificacion = identificacion;
        this.nombre = nombre;
        this.correoElectronico = correoElectronico;
    }
    // aplica el método Getter and Setter
    public Long getIdentificacion() {
        return identificacion;
    }
    public void setIdentificacion(Long identificacion) {
        this.identificacion = identificacion;
    }
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public String getCorreoElectronico() {
        return correoElectronico;
    }
    public void setCorreoElectronico(String correoElectronico) {
        this.correoElectronico = correoElectronico;
    }
}
```

**Nota:** la imagen contine el código generado para la clase usuario. Fuente: creación propia en la herramienta NetBeans.

3. En la clase UsuarioRepositorio se importaron las librerías requeridas, las librerías se importan con el propósito de reutilizar el código que ya existe en el programa, algunas son propias de JAVA como es el caso de ArrayList y List, otras con las clases que se han creado previamente como modelo.Usuario. También se crea y se inicializa el atributo idMostrador y se crean los métodos para registrar, guardar, buscar y eliminar usuarios; estos métodos son los que le dicen al programa cuales son las acciones que se realizarán con los atributos y permiten la interacción con el cliente.

### Ilustración 3

#### Clase UsuarioRepositorio

```
package repositorio;

// se importan las librerías
import java.util.ArrayList;
import java.util.List;
import modelo.Usuario;

public class UsuarioRepositorio {
    private List<Usuario> usuarios = new ArrayList<>();
    private Long idMostrador = 1L;

    //se crean los métodos para registrar, guardar, buscar y eliminar usuarios
    public List<Usuario> registrarUsuarios() {
        return usuarios;
    }

    public void guardarUsuarios(Usuario usuario) {
        usuario.setIdentificacion(idMostrador++);
        System.out.println("Usuario guardado con identificación: " + usuario.getIdentificacion());
        usuarios.add(usuario);
    }

    public Usuario buscarUsuarios(Long identificacion) {
        return usuarios.stream()
            .filter(usuario -> usuario.getIdentificacion().equals(identificacion))
            .findFirst()
            .orElse(null);
    }

    public void eliminarUsuario(Long identificacion) {
        usuarios.removeIf(usuario -> usuario.getIdentificacion().equals(identificacion));
    }
}
```

**Nota:** la imagen contiene el código generado para la clase UsuarioRepositorio. Fuente: creación propia en la herramienta NetBeans.

4. En la clase UsuarioServicio se importaron las librerías necesarias, se crea la instancia de la clase UsuarioRepositorio, es decir, se crea un objeto a partir de la clase que ya estaba creada y se replican sus métodos: registrarUsuarios, guardarUsuarios, buscarUsuarios y eliminarUsuario, estos métodos son las acciones que realizará el programa.

#### Ilustración 4

##### Clase UsuarioServicio

```
package servicio;

/** se importan las librería */
import java.util.List;
import modelo.Usuario;
import repositorio.UsuarioRepositorio;

public class UsuarioServicio {

    /** se crea una instancia de la clase UsuarioRepositorio*/
    private UsuarioRepositorio usuarioRepositorio = new UsuarioRepositorio();

    /** se traen los métodos de la clase UsuarioRepositorio*/
    public List<Usuario> registrarUsuarios() {
        return usuarioRepositorio.registrarUsuarios();
    }

    public void guardarUsuarios(Usuario usuario) {
        usuarioRepositorio.guardarUsuarios(usuario);
    }

    public Usuario buscarUsuarios(Long identificacion) {
        return usuarioRepositorio.buscarUsuarios(identificacion);
    }

    public void eliminarUsuario(Long identificacion) {
        usuarioRepositorio.eliminarUsuario(identificacion);
    }
}
```

**Nota:** la imagen contiene el código generado para la clase UsuarioServicios. Fuente: creación propia en la herramienta NetBeans.

5. En la clase UsuarioControlador se importan las librerías requeridas, se genera una instancia de la clase UsuarioServicio, se crea el método mostrarOpciones y mientras sea verdadero se solicita al cliente que digite una opción (tal como se evidencia en el menú), de acuerdo a la opción que se seleccione: 1, 2, 3, 4 o 5, se crea un switch para cada número. Es decir, en la opción 1 se registran los datos del nuevo usuario, llamando el método guardarUsuarios desde la clase usuarioServicios. En la opción 2, permite enlistar los usuarios que se van creando, generando una identificación según el orden de entrada, ejemplo: Primer usuario registrado es el 1. En la opción 3, se digita la identificación del usuario y este confirma si existe. En la opción 4, permite digitar la identificación del usuario a eliminar y confirma si fue eliminado. La opción 5 está habilitada para salir del sistema. Si el cliente digita una opción diferente se genera un mensaje indicando que la opción no es válida.

### Ilustración 5

#### Clase UsuarioControlador

```
package controlador;

import java.util.List;
import java.util.Scanner;
import javax.swing.JOptionPane;
import modelo.Usuario;
import servicio.UsuarioServicio;

public class UsuarioControlador {

    private UsuarioServicio usuarioServicio = new UsuarioServicio();
    private Scanner scanner = new Scanner (System.in);

    public void mostrarOpciones() {

        while(true) {
            System.out.println("Digite el número de la opción requerida");
            System.out.println("1. Registrar usuario");
            System.out.println("2. Guardar usuario");
            System.out.println("3. Buscar usuario con el número de identificación");
            System.out.println("4. Eliminar usuario");
            System.out.println("5. Salir del sistema");

            int opcion = scanner.nextInt();
            scanner.nextLine(); //Permite limpiar el bufer
        }
    }
}
```

```

switch(opcion){
    case 1:
        System.out.println("Registrar la identificación: ");
        Long identificacion = scanner.nextLong();
        System.out.println("Registrar el nombre: ");
        String nombre = scanner.nextLine();
        System.out.println("Registrar el correo electrónico: ");
        String correoElectronico = scanner.nextLine();
        usuarioServicio.guardarUsuarios(new Usuario (identificacion, nombre, correoElectronico));
        System.out.println("El usuario registrado es: " + identificacion + " " + nombre + " " + correoElectronico);
        break;

    case 2:
        List<Usuario> usuarios = usuarioServicio.registrarUsuarios();
        usuarios.forEach(usuario -> System.out.println(usuario.getIdentificacion() + " _ " + usuario.getNombre()));
        break;

    case 3:
        System.out.println("Digite la identificación del usuario: ");
        Long idMostrador = scanner.nextLong();
        Usuario usuario = usuarioServicio.buscarUsuarios(idMostrador);

        if(usuario != null){
            System.out.println(usuario.getIdentificacion() + " - " + usuario.getNombre());
        } else{
            System.out.println("El usuario no se encuentra en la base de datos");
        }
        break;

    case 4:
        System.out.println("Digite la identificación del usuario a eliminar: ");
        Long idEliminar = scanner.nextLong();
        usuarioServicio.eliminarUsuario(idEliminar);

        break;

    case 5:
        System.exit(0);

        break;

    default:
        System.out.println("Esta opción no es válida");
}

```

**Nota:** la imagen contine el código generado para la clase UsuarioControlador. Fuente: creación propia en la herramienta NetBeans.



6. La clase Ejecutor se marca como clase principal a través del método `public static void main`, se instancia a la clase `UsuarioControlador` y se llama al método `mostrarOpciones`. Al ser la clase principal será la utilizada para ejecutar el proyecto.

#### Ilustración 4

##### Clase Ejecutor

```
import controlador.UsuarioControlador;

public class Ejecutor {

    public static void main(String[] args) {

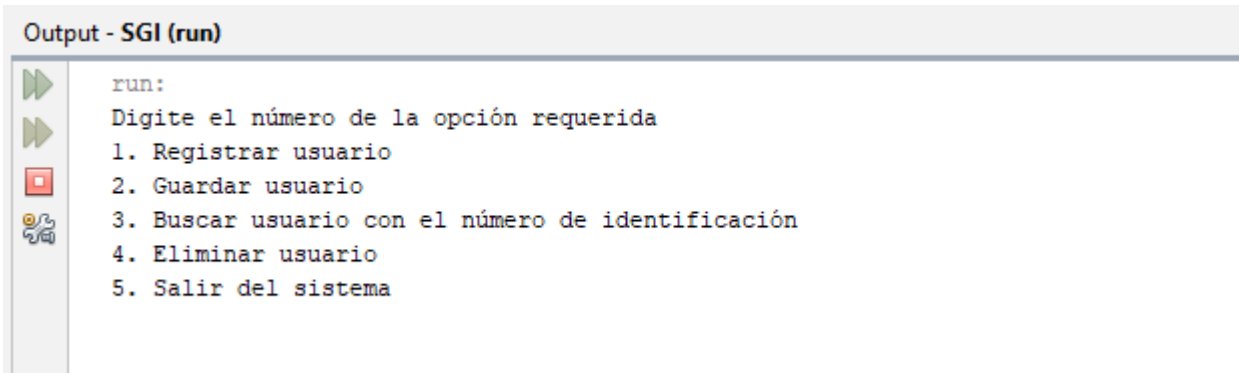
        UsuarioControlador controlador = new UsuarioControlador();
        controlador.mostrarOpciones();
    }
}
```

**Nota:** la imagen contiene el código generado para la clase Ejecutor. Fuente: creación propia en la herramienta NetBeans.

7. A continuación, se presenta una muestra de la ejecución del proyecto, en donde inicia la interacción con el Cliente.

### Ilustración 7

#### Ejecución del Proyecto



```
Output - SGI (run)

run:
Digite el número de la opción requerida
1. Registrar usuario
2. Guardar usuario
3. Buscar usuario con el número de identificación
4. Eliminar usuario
5. Salir del sistema
```

**Nota:** la imagen contiene el resultado de la ejecución del proyecto y la ventana que permitirá la interacción con el Cliente. Fuente: creación propia en la herramienta NetBeans.

### Referencias bibliográficas

- Arciniegas Herrera, J. L., Collazos Ordóñez, C. A., Fernández de Valdenebro, M. V., Hormiga Juspian, M. A., Tulande Arroyo, A. (2010). Patrones arquitectónicos sobre usabilidad en el dominio de las aplicaciones web. *Ingeniería e Investigación*, 30 (1), 52-55.
- Sánchez, J. (23 septiembre 2024). *Arquitectura Cliente Servidor*. [Tutoría] Curso Arquitectura de Software Corporación Universitaria Iberoamericana.