UNIVERSIDAD MARIANA
FACULTAD INGENIERÍA – PROGRAMA INGENIERÍA DE SISTEMAS

ESTRUCTURAS DE DATOS 1

LABORATORIO NIVEL 11

Objetivos

1. Realizar un ejercicio que permita incorporar java web
2. Almacenar información en la memoria temporal
3. Serializar la información de un aplicativo
4. Demostrar la comunicación en la web

Número de participantes

Duración de la actividad

Materiales

- Internet
- Eclipse o Netbeans
- Apache Tomcat
- Maven

Proyecto:
Desarrollo de una aplicación web para un directorio de contactos

Descripción:
Desarrollar una aplicación java web teneindo como base un árbol binario que pueda realizar lo siguiente:

1.Insertar un nuevo contacto
2. Listar el directorio en alguno de los recorridos (inOrden, PreOrden o PosOrden) se debe especificar cual
3.Eliminar un contacto
Los ordenamientos ya los tiene incorporados la plantilla con javascript
Editar contacto(opcional

Link al git: https://github.com/DavidNoguera1/Arbol-Contactos

Autoría

Proyecto Curso:	**Estructuras de datos II**
Ejercicio:	Aplicación web para un directorio de contactos
Autores:	1. David Noguera
	2. Juan Diego Arevalo
	3. Samuel Bolaños
Fecha realización:	

UNIVERSIDAD MARIANA
FACULTAD INGENIERÍA – PROGRAMA INGENIERÍA DE SISTEMAS

ESTRUCTURAS DE DATOS 1

LABORATORIO NIVEL 11

Listado de Requerimientos:

Nombre	**R1- Mostrar la lista de los contactos**
Resumen	Muestra todos los contactos agregados en una lista junto a sus datos.
Entradas	Ninguna
Resultados	Se muestra la lista de los contactos agregados.

Nombre	**R2- Visualizar datos**
Resumen	El programa debe permitirle al usuario visualizar los datos del contacto que seleccione.
Entradas	Ninguna
Resultados	El usuario ha podido visualizar los datos del contacto que seleccionó.

Nombre	**R3- Añadir contacto**
Resumen	El programa debe permitirle al usuario agregar nuevos contactos al directorio
Entradas	IdContacto, nombre, apellido, correo electrónico y celular.
Resultados	Contacto nuevo agregado al directorio correctamente

Nombre	**R4- Eliminar contacto**
Resumen	El programa debe permitirle al usuario eliminar contactos del directorio
Entradas	Nombre del contacto a eliminarse
Resultados	Contacto eliminado

Nombre	**R5- Ordenar InOrder automático**
Resumen	Los contactos se añaden y organizan de manera “InOder” (alfabéticamente).
Entradas	Ninguna
Resultados	Lista ordenada de forma “InOrder”.

Nombre	**R6- Editar contacto**
Resumen	El programa debe permitirle al usuario editar los datos de los contactos del directorio. Tomando como dato identificador y NO editable el nombre del contacto
Entradas	Nuevos datos del contacto a editar o reemplazar (id, apellido, correo electrónico o celular).
Resultados	Contacto editado correctamente.

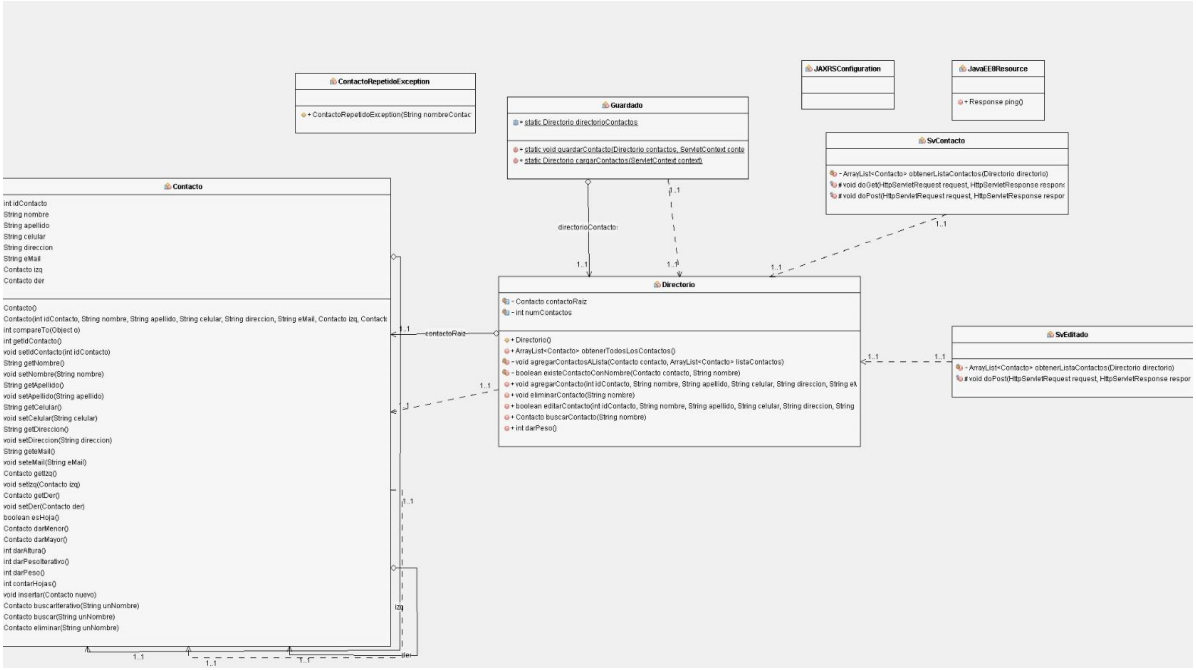
Nombre	**R7- Buscar contacto**
Resumen	El programa debe permitirle al usuario buscar contactos en el directorio.
Entradas	Nombre del contacto a buscarse
Resultados	Contacto encontrado o contactos con similitudes más cercanas

UNIVERSIDAD MARIANA
FACULTAD INGENIERÍA – PROGRAMA INGENIERÍA DE SISTEMAS

ESTRUCTURAS DE DATOS 1

LABORATORIO NIVEL 11

Modelo conceptual generado por NetBeans

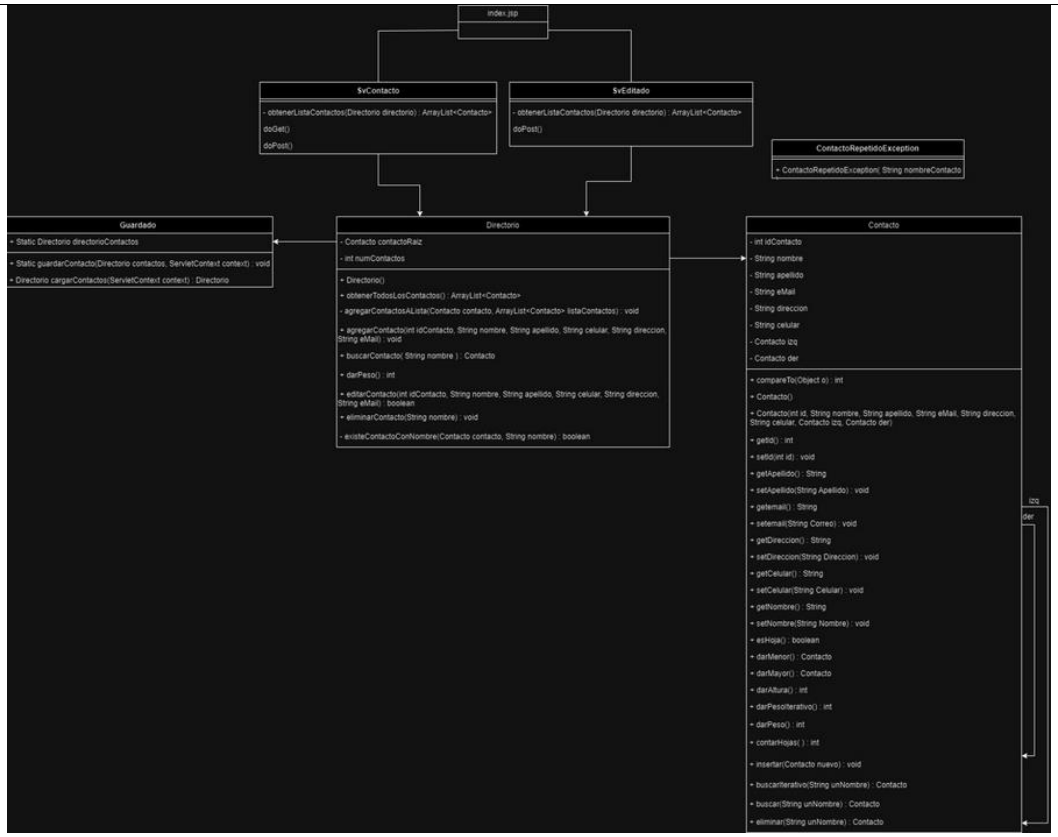


Modelo conceptual dibujado por el equipo en draw.io

UNIVERSIDAD MARIANA
FACULTAD INGENIERÍA – PROGRAMA INGENIERÍA DE SISTEMAS

ESTRUCTURAS DE DATOS 1

LABORATORIO NIVEL 11



Desarrollo del procedimiento

Cada miembro cumplió adecuadamente con sus responsabilidades trabajando como equipo y señalando posibles mejoras a realizarse o implementado soluciones y colaborando cuando alguna dificultad se presentase.

1) Creación de clase Contacto (Atributos del contacto, Getters y Setters). (Noguera y Arevalo).

```java
/**
 *
 * @author David Noguera
 */
import java.io.Serializable;
import java.util.ArrayList;

/**
 *
 * @author juand
 */
public class Contacto implements Comparable, Serializable{
```

UNIVERSIDAD MARIANA
FACULTAD INGENIERÍA – PROGRAMA INGENIERÍA DE SISTEMAS

ESTRUCTURAS DE DATOS 1

LABORATORIO NIVEL 11

```java
public Contacto(int idContacto, String nombre, String apellido, String celular, String direccion, String eMail, Contacto izq, Contacto der) {
    this.idContacto = idContacto;
    this.nombre = nombre;
    this.apellido = apellido;
    this.celular = celular;
    this.direccion = direccion;
    this.eMail = eMail;
    this.izq = null;
    this.der = null;
}
```

2) Creación clase Directorio (Métodos de agregado, ordenamiento automático, eliminación y edición). (Noguera y Arevalo).

```java
import java.io.Serializable;
import java.util.ArrayList;

public class Directorio implements Serializable {

    private Contacto contactoRaiz;

    private int numContactos;

    public Directorio() {
        contactoRaiz = null;
        numContactos = 0;
    }

    public ArrayList<Contacto> obtenerTodosLosContactos() {
        ArrayList<Contacto> listaContactos = new ArrayList<>();
        agregarContactosAListaa(contacto: contactoRaiz, listaContactos);
        return listaContactos;
    }

    private void agregarContactosAListaa(Contacto contacto, ArrayList<Contacto> listaContactos) {
        if (contacto != null) {
            agregarContactosAListaa(contacto: contacto.getIzq(), listaContactos);
            listaContactos.add(e: contacto);
            agregarContactosAListaa(contacto: contacto.getDer(), listaContactos);
        }
    }
```

UNIVERSIDAD MARIANA
FACULTAD INGENIERÍA – PROGRAMA INGENIERÍA DE SISTEMAS

ESTRUCTURAS DE DATOS 1

LABORATORIO NIVEL 11

```
public void agregarContacto(int idContacto, String nombre, String apellido, String celular, String direccion, String eMail)
        throws ContactoRepetidoException {
    Contacto nuevoContacto = new Contacto(idContacto, nombre, apellido, celular, direccion, eMail, izq:null, der:null);

    // Verificar si ya existe un contacto con el mismo nombre
    if (existeContactoConNombre(contacto: contactoRaiz, nombre)) {
        throw new ContactoRepetidoException(nombreContacto: nombre);
    }

    // Continuar con la lógica para agregar el nuevo contacto
    if (contactoRaiz == null) {
        contactoRaiz = nuevoContacto;
    } else {
        contactoRaiz.insertar(nuevo: nuevoContacto);
    }

    numContactos++;
}

public void eliminarContacto(String nombre) {
    contactoRaiz = contactoRaiz.eliminar(unNombre: nombre);
    numContactos--;

}

public boolean editarContacto(int idContacto, String nombre, String apellido, String celular, String direccion, String eMail) {
    Contacto contacto = buscarContacto(nombre);

    if (contacto != null) {
        // Actualizar los atributos del contacto
        contacto.setIdContacto(idContacto);
        contacto.setApellido(apellido);
        contacto.setCelular(celular);
        contacto.setDireccion(direccion);
        contacto.seteMail(eMail);

        return true; // La edición fue exitosa
    }

    return false; // Si no se encontró el contacto por nombre, la edición no se realiza
}
```

3) Creacion clase Guardado (Metodos de serializacion y deserealizacion) (Bolaños y Noguera).

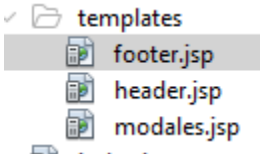
```
public class Guardado implements Serializable {

    public static Directorio directorioContactos;

    public static void guardarContacto(Directorio contactos, ServletContext context) throws IOException {
        String relativePath = "/data/contactos.ser";
        String absPath = context.getRealPath(string: relativePath);
        File archivo = new File(pathname: absPath);

        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(file: archivo))) {
            oos.writeObject(obj: contactos);
            System.out.println(x: "Datos de contactos guardados exitosamente en: contactos.ser");
        } catch (IOException e) {
            e.printStackTrace();
            System.out.println("Error al guardar los datos de contacto: " + e.getMessage());
        }
    }

    public static Directorio cargarContactos(ServletContext context) throws IOException, ClassNotFoundException {
        Directorio directorioContactos = null;
        String relativePath = "/data/contactos.ser";
        String absPath = context.getRealPath(string: relativePath);
        File archivo = new File(pathname: absPath);

        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(file: archivo))) {
            directorioContactos = (Directorio) ois.readObject();
            System.out.println(x: "Datos de contactos cargados exitosamente desde: contactos.ser");
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
            System.out.println("Error al cargar los datos de contactos: " + e.getMessage());
        }

        if (directorioContactos == null) {
            directorioContactos = new Directorio();
        }

        return directorioContactos;
    }
}
```

UNIVERSIDAD MARIANA
FACULTAD INGENIERÍA – PROGRAMA INGENIERÍA DE SISTEMAS

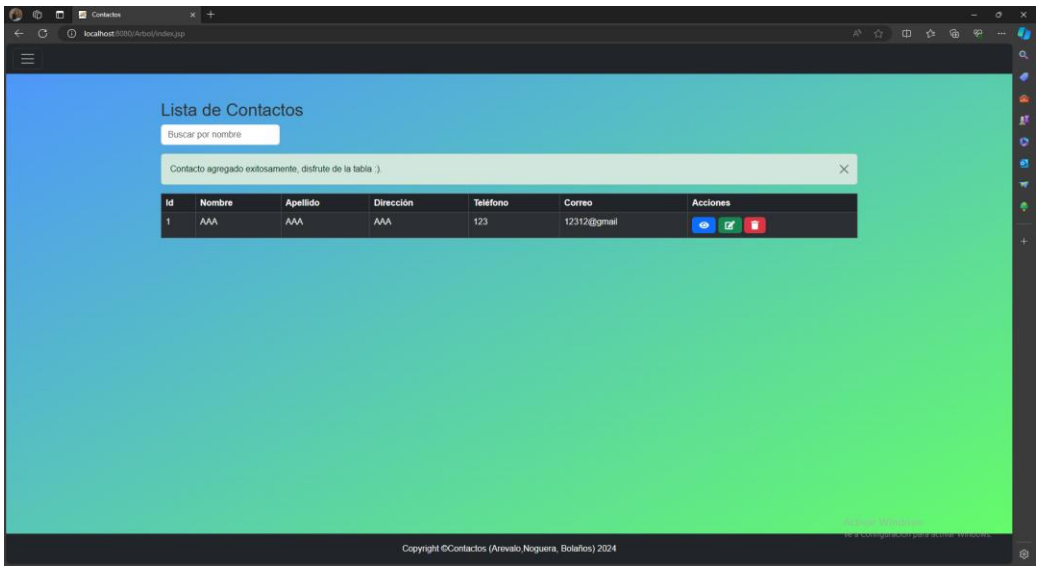
ESTRUCTURAS DE DATOS 1

LABORATORIO NIVEL 11

4) Creación de la index y templates (Arévalo y Noguera)




```
<%@page import="java.util.ArrayList"%>
<%@page import="com.mycompany.arbol.Contacto"%>
<%@include file= "templates/header.jsp" %>

<script src="https://code.jquery.com/jquery-3.6.4.min.js"></script>

<div class="collapse" id="navbarToggleExternalContent" data-bs-theme="dark">
    <div class="bg-dark p-4">
        <h5 class="text-body-emphasis h4">Directorio de Contactos </h5>
        <button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-ta
            Agregar nuevo contacto
        </button>
    </div>
</div>
<nav class="navbar navbar-dark bg-dark">
    <div class="container-fluid">
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-t
            <span class="navbar-toggler-icon"></span>
        </button>
    </div>
</nav>

<div class="container mt-5">
    <h2>Lista de Contactos</h2>

    <div class="input-group mb-3">
        <div class="form-outline" data-mdb-input-init>
            <input id="search-focus" type="search" id="form1" class="form-control"
                   placeholder="Buscar por nombre" />
        </div>

    </div>
```



5) Creación del Servlet SvContactos para la añadidura de contactos y eliminación de estos (Bolaños y Noguera)

UNIVERSIDAD MARIANA
FACULTAD INGENIERÍA – PROGRAMA INGENIERÍA DE SISTEMAS

ESTRUCTURAS DE DATOS 1

LABORATORIO NIVEL 11

```java
@WebServlet(name = "SvContacto", urlPatterns = {"/SvContacto"})
public class SvContacto extends HttpServlet implements Serializable {

    private ArrayList<Contacto> obtenerListaContactos(Directorio directorio) {
        return directorio.obtenerTodosLosContactos();
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        // Obtener el contexto del servlet
        ServletContext context = getServletContext();

        // Obtener la lista de contactos del contexto
        Directorio listaContactos = (Directorio) context.getAttribute(string: "directorio");

        if (listaContactos == null) {
            response.sendRedirect(string: "index.jsp");
            return;
        }

        String action = request.getParameter(string: "action");

        if ("deleteContact".equals(anObject:action)) {
            String nombre = request.getParameter(string: "nombre");

            // Output some debug information
            System.out.println("Contact to delete: " + nombre);
```

```java
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    // Obtener el contexto del servlet
    ServletContext context = getServletContext();

    Directorio directorio = (Directorio) context.getAttribute(string: "directorio");

    if (directorio == null) {
        directorio = new Directorio();
        context.setAttribute(string: "directorio", o: directorio);
    }

    // Data del formulario
    String nombre = request.getParameter(string: "nombre");
    String apellido = request.getParameter(string: "apellido");
    String celular = request.getParameter(string: "telefono");
    String direccion = request.getParameter(string: "direccion");
    String email = request.getParameter(string: "correo");
    String id = request.getParameter(string: "id");

    try {
        // Verificar si el contacto ya existe antes de agregarlo
        directorio.agregarContacto(idContacto: Integer.parseInt(s: id), nombre, apellido, celular, direccion, eMail: email);

        // Call guardarContacto to save the updated contact list
        Guardado.guardarContacto(contactos: directorio, context);

        // Load the updated contact list
        ArrayList<Contacto> listaContactos = obtenerListaContactos(directorio);

        // Set the updated contact list as a session attribute
        HttpSession session = request.getSession();
        session.setAttribute(string: "listaContactos", o: listaContactos);
        session.setAttribute(string: "alertType", o: "success"); // Agrega este atributo para el tipo de alerta

        // Redirect to the "index.jsp" page
        response.sendRedirect(request.getContextPath() + "/index.jsp");

    } catch (ContactoRepetidoException e) {
        // Handle the exception if the contact already exists
```

6) Creacion del Servlet SvEditado para la edicion (Noguera)

UNIVERSIDAD MARIANA
FACULTAD INGENIERÍA – PROGRAMA INGENIERÍA DE SISTEMAS

ESTRUCTURAS DE DATOS 1

LABORATORIO NIVEL 11

```java
@WebServlet(name = "SvEditado", urlPatterns = {"/SvEditado"})
public class SvEditado extends HttpServlet {

    private ArrayList<Contacto> obtenerListaContactos(Directorio directorio) {
        return directorio.obtenerTodosLosContactos();
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        // Obtener el contexto del servlet
        ServletContext context = getServletContext();

        // Obtener el directorio del contexto
        Directorio directorio = (Directorio) context.getAttribute(string: "directorio");

        if (directorio == null) {
            directorio = new Directorio();
            context.setAttribute(string: "directorio", o: directorio);
        }

        // Data del formulario
        String nombre = request.getParameter(string: "nombre");
        String apellido = request.getParameter(string: "apellido");
        String celular = request.getParameter(string: "telefono");
        String direccion = request.getParameter(string: "direccion");
        String email = request.getParameter(string: "correo");
        String id = request.getParameter(string: "id");
```

7) Desarrollo de diagramas de clases (Bolaños)
8) Implementación de alerts error o éxito (Noguera)
9) Testeo e implementación de comentarios y mejoras (Arévalo)