

# Instituto Politécnico Nacional

## Escuela Superior de Cómputo

Web App Development.

Reporte de práctica 1: Eventos Servlet

*Profesor: M. en C. José Asunción Enríquez Zárate*

*Alumno: Aarón Antonio García González*

*aarongarcia.ipn.escom@gmail.com*

*3CM9*

November 8, 2020

# Contents

<b>1</b>	<b>Introducción</b>	<b>2</b>
<b>2</b>	<b>Conceptos</b>	<b>3</b>
2.1	Java Servlet	3
2.2	Java Server Pages (JSP)	3
2.3	Data Access Object	3
<b>3</b>	<b>Desarrollo</b>	<b>5</b>
3.1	<i>Evento.java</i>	5
3.2	<i>EventoDAO.java</i>	6
3.3	<i>EventoServlet.java</i>	10
<b>4</b>	<b>Resultados</b>	<b>17</b>
4.1	Ejecución del Programa	17
4.1.1	<i>Inicio</i>	17
4.1.2	<i>Lista de eventos</i>	18
4.1.3	<i>Ver elemento</i>	18
4.1.4	<i>Actualizar evento</i>	19
4.1.5	<i>Nuevo evento</i>	19
<b>5</b>	<b>Conclusión</b>	<b>21</b>
<b>6</b>	<b>Referencias Bibliográficas</b>	<b>22</b>

Lista de figuras

1	Página de inicio . . . . .	17
2	Página que lista todos los eventos previamente registrados . . . . .	18
3	Página de vista de un evento en específico . . . . .	18
4	Página de actualizar por ejemplo el evento 39 . . . . .	19
5	Formulario de crear evento . . . . .	20

**Lista de tablas**

# 1 Introducción

El servidor Jakarta-Tomcat, es uno de los proyectos de código abierto liderado por la Apache Software Foundation. El servidor Tomcat es una aplicación web basada en Java creada para ejecutar servlets y páginas JSP, siendo la implementación oficial de referencia de las especificaciones Servlet 2.3 y JavaServer Pages 1.2.

Antes de continuar, es necesario tener un conocimiento básico del concepto de Aplicación Web, que fue introducido en la versión 2.2 de la especificación servlet. De acuerdo con esta especificación, una aplicación web es una colección de servlets, páginas JSP, clases Java, archivos de descripción de la aplicación, documentos estáticos: HTML, XHTML, imágenes, etc. y otros recursos que pueden ser empaquetados y ejecutados en distintos servidores de diferentes proveedores. Es decir, una aplicación web se podría definir como la capa web de cualquier aplicación.

Las aplicaciones web son populares debido a lo práctico del navegador web como cliente ligero, a la independencia del sistema operativo, así como a la facilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software a miles de usuarios potenciales. Existen aplicaciones como los correos web, wikis, blogs, tiendas en línea y la propia Wikipedia que son ejemplos bastante conocidos de aplicaciones web.

Tomcat es un contenedor de servlets, puede utilizarse como un servidor de aplicaciones Web con HTML, servlets y JSPs o como complemento al servidor Apache. Además, Tomcat es compatible con otras tecnologías como Java Expression Language y Java WebSocket, del ecosistema Java.

Tomcat puede funcionar de manera autónoma como motor de aplicaciones web desarrolladas con Java, aunque habitualmente se usa en combinación con otros productos como el servidor web Apache, para dar un mayor soporte a tecnologías y aumentar sus características.

## 2 Conceptos

### 2.1 Java Servlet

Un servlet es una clase en el lenguaje de programación Java, utilizada para ampliar las capacidades de un servidor. Aunque los servlets pueden responder a cualquier tipo de solicitudes, estos son utilizados comúnmente para extender las aplicaciones alojadas por servidores web, de tal manera que pueden ser vistos como applets de Java que se ejecutan en servidores en vez de navegadores web.

El uso más común de los servlets es generar páginas web de forma dinámica a partir de los parámetros de la petición que envíe el navegador web.

¿Cómo funciona un contenedor de Servlets?

- El navegador (cliente) pide una página al servidor HTTP que es un contenedor de Servlets.
- El servlet procesa los argumentos de la petición, es decir, el contenedor de Servlets delega la petición a un Servlet en particular elegido de entre los Servlets que contiene.
- El Servlet, que es una objeto java, se encarga de generar el texto de la página web que se entrega al contenedor.
- El contenedor devuelve la página web al navegador (cliente) que la solicitó, normalmente en HTML.

Por lo tanto nos encontramos en una arquitectura Cliente-Servidor. Lo normal para esto es utilizar Apache Tomcat como contenedor de servlets. Recordemos que apache es un servidor HTTP.

### 2.2 Java Server Pages (JSP)

JavaServer Pages (JSP) es una tecnología que ayuda a los desarrolladores de software a crear páginas web dinámicas basadas en HTML y XML, entre otros tipos de documentos. JSP es similar a PHP, pero usa el lenguaje de programación Java.

Para desplegar y correr JavaServer Pages, se requiere un servidor web compatible con contenedores servlet como Apache Tomcat o Jetty.

El rendimiento de una página JSP es el mismo que tendría el servlet equivalente, ya que el código es compilado como cualquier otra clase Java. A su vez, la máquina virtual compilará dinámicamente a código de máquina las partes de la aplicación que lo requieran. Esto hace que JSP tenga un buen desempeño y sea más eficiente que otras tecnologías web que ejecutan el código de una manera puramente interpretada.

La principal ventaja de JSP frente a otros lenguajes es que el lenguaje Java es un lenguaje de propósito general que excede el mundo web y que es apto para crear clases que manejen lógica de negocio y acceso a datos de una manera prolija. Esto permite separar en niveles las aplicaciones web, dejando la parte encargada de generar el documento HTML en el archivo JSP.

Otra ventaja es que JSP hereda la portabilidad de Java, y es posible ejecutar las aplicaciones en múltiples plataformas sin cambios. Es común incluso que los desarrolladores trabajen en una plataforma y que la aplicación termine siendo ejecutada en otra.

Los servlets y Java Server Pages (JSPs) son dos métodos de creación de páginas web dinámicas en servidor usando el lenguaje Java. En ese sentido son similares a otros métodos o lenguajes tales como el PHP, ASP o los CGIs, programas que generan páginas web en el servidor. Sin embargo, se diferencian de ellos en otras cosas.

Las JSPs son en realidad una forma alternativa de crear servlets ya que el código JSP se traduce a código de servlet Java la primera vez que se le invoca y en adelante es el código del nuevo servlet el que se ejecuta produciendo como salida el código HTML que compone la página web de respuesta.

### 2.3 Data Access Object

Un objeto de acceso a datos ( DAO ) es un patrón que proporciona una interfaz abstracta a algún tipo de base de datos u otro mecanismo de persistencia. Al asignar las llamadas de la aplicación a la capa de persistencia, el DAO proporciona algunas operaciones de datos específicas sin exponer los detalles de la base de datos. Este

aislamiento respalda el principio de responsabilidad única . Separa qué acceso a datos necesita la aplicación, en términos de objetos y tipos de datos específicos del dominio (la interfaz pública de la DAO), de cómo estas necesidades pueden satisfacerse con un DBMS específico , esquema de base de datos, etc.

El acceso a datos se realiza mediante DAO (Data Access Object), con ello se obtiene una abstracción sobre el modelo de datos. Solo se accede a los datos a través de métodos definidos en el DAO. Los DAO incluyen otro concepto que es el DTO (Data Transfer Object). Los DTO son un tipo de objetos que sirven únicamente para transportar datos. EL DTO contiene las propiedades del objeto. Datos que pueden tener su origen en una o más entidades de información. Estos datos son incorporados a una instancia de un JavaBean. Y esta instancia puede ser pasada a través de la aplicación, localmente o, lo que es más importante, puede ser "serializada" y enviada a través de la red, de forma que los clientes puedan acceder a información del modelo desde un solo objeto y mediante una sola llamada.

Un DTO normalmente no provee lógica de negocios o validaciones de ningún tipo. Solo provee acceso a las propiedades del bean. Algunos autores remarcan que el bean debe ser inmutable, dado que sus cambios no deben reflejarse en el sistema. Pero obviamente esto choca con la especificación de los JavaBean, que requiere que todos los atributos privados tengan sus métodos set y get. Es necesario definir un DTO por cada bean que exista en la aplicación.

Con este modelo se obtiene, la separación entre el Controlador y el Modelo. Ante cualquier cambio que se diera en la forma de acceder a los datos, el modelo no se va a modificar ya que recibe el mismo DTO.

## 3 Desarrollo

Esta práctica se realizó durante el periodo del primer departamental de la unidad de aprendizaje optativa "Web Application Development", se inició con la creación del modelo Evento, sus respectivos atributos y métodos, hasta tener un CRUD funcional utilizando Servlet.

### 3.1 *Evento.java*

Modelo en el que se basa la práctica, el modelo es llamado Evento, donde el contexto que le dimos a lo largo del periodo fue en la ESCOM, donde un evento es un acontecimiento que tiene lugar en el Instituto Politécnico Nacional, tal como una feria educativa, de empleo, semanas culturales o de registro a algún trámite o servicio, días especiales celebrados en las instalaciones físicas y virtuales, etc. Cada evento tiene un identificador, un nombre, un lugar de sede, una fecha de inicio y una fecha de término.

Para realizar los métodos clásicos de una clase, tales como constructor, accesores y mutadores, utilice una librería llamada "lombok", la cual con una serie de anotaciones antes de declarar la clase, hace posible no saturar la clase al generar dichos métodos directamente, esto solo funciona cuando dichos métodos son genéricos, es decir no requerimos personalizarlos, así como cuando no son heredados de otra clase superior.

---

```
1 package com.escom.wad.baseproject.model;
2 import java.io.Serializable;
3 import java.sql.Date;
4 import lombok.*;
5
6 @Getter
7 @Setter
8 @AllArgsConstructor
9 @NoArgsConstructor
10 @Builder
11 @ToString
12
13 /**
14  * @author aarongarcia
15  */
16 public class Evento implements Serializable
17 {
18     private Integer idEvento;
19     private String nombreEvento;
20     private String sede;
21     private Date fechaInicio;
22     private Date fechaFin;
23 }
```

---

*Algoritmo:* Clase Evento



### 3.2 *EventoDAO.java*

Dentro de este método, es únicamente para el acceso a datos del modelo de base de datos Evento, dentro de esta clase lo que se hace es definir genéricamente los queries en Strings a los cuales son enviados como parámetros al método `prepareStatement` de la interfaz `connection` del paquete `java.sql`, es en esta clase donde definimos los métodos: consultar todos, consultar por identificador, actualizar, eliminar o crear nuevos eventos.

Para poner en funcionamiento esta clase, antes ya debe de estar creada la base de datos Evento, adjunto en la entrega de esta actividad.

---

```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package com.escom.wad.baseproject.dao;
7  import com.escom.wad.baseproject.model.Evento;
8  import java.sql.Connection;
9  import java.sql.DriverManager;
10 import java.sql.PreparedStatement;
11 import java.sql.ResultSet;
12 import java.sql.SQLException;
13 import java.util.ArrayList;
14 import java.util.List;
15 import java.util.logging.Level;
16 import java.util.logging.Logger;
17
18 /**
19  *
20  * @author aarongarcia
21  */
22 public class EventoDAO {
23     private static final String SQL_INSERT =
24         "insert into Evento (nombreEvento, sede, fechalnicio, fechaFin) "
25         + "values (?, ?, ?, ?)";
26
27     private static final String SQL_UPDATE =
28         "update Evento set "
29         + "nombreEvento = ?, sede = ?, fechalnicio = ?, fechaFin = ? "
30         + "where idEvento = ? ";
31
32     private static final String SQL_DELETE =
33         "delete from Evento where idEvento = ?";
34
35     private static final String SQL_SELECT =
36         "select * from Evento where idEvento = ?";
37
38     private static final String SQL_SELECT_ALL =
39         "select * from Evento";
40
41     private Connection connection;
42
43     private void obtenerConexion()
44     {
45         String user = "root";
46         String password = "rootroot";
47         String url = "jdbc:mysql://localhost:3306/WAD?useUnicode=true&useJDBCCompliantTimezon";
48         String driverMySQL = "com.mysql.cj.jdbc.Driver";
49
50         try {
51             Class.forName(driverMySQL);
52             connection = DriverManager.getConnection(url, user, password);
53         } catch (ClassNotFoundException | SQLException ex) {
54             Logger.getLogger(EventoDAO.class.getName()).log(Level.SEVERE, null, ex);
55         }
56     }
```

```

57
58 public void create(Evento evento) throws SQLException
59 {
60     obtenerConexion();
61     PreparedStatement ps = null;
62
63     try {
64         ps = connection.prepareStatement(SQL_INSERT);
65         ps.setString(1, evento.getNombreEvento());
66         ps.setString(2, evento.getSede());
67         ps.setDate(3, evento.getFechaInicio());
68         ps.setDate(4, evento.getFechaFin());
69         ps.executeUpdate();
70     } finally {
71         if (ps != null) {
72             ps.close();
73         }
74         if (connection != null){
75             connection.close();
76         }
77     }
78 }
79
80
81 public void update(Evento e) throws SQLException {
82     obtenerConexion();
83     PreparedStatement ps = null;
84     try {
85         ps = connection.prepareStatement(SQL_UPDATE);
86         ps.setString(1, e.getNombreEvento());
87         ps.setString(2, e.getSede());
88         ps.setDate(3, e.getFechaInicio());
89         ps.setDate(4, e.getFechaFin());
90         ps.setInt(5, e.getIdEvento());
91         ps.executeUpdate();
92     } finally {
93         if (ps != null) {
94             ps.close();
95         }
96         if (connection != null) {
97             connection.close();
98         }
99     }
100 }
101
102
103 public void delete(Evento evento) throws SQLException
104 {
105     obtenerConexion();
106     PreparedStatement ps = null;
107
108     try
109     {
110         ps = connection.prepareStatement(SQL_DELETE);
111         ps.setInt(1, evento.getIdEvento());
112         ps.executeUpdate();
113     }
114     finally
115     {
116         if (ps != null) {
117             ps.close();
118         }
119         if (connection != null){
120             connection.close();
121         }
122     }

```

```

123     }
124
125
126     public List readAll() throws SQLException
127     {
128         obtenerConexion();
129         PreparedStatement ps = null;
130         ResultSet rs = null;
131
132         try {
133             ps = connection.prepareStatement(SQL_SELECT_ALL);
134             rs = ps.executeQuery(); // por ser de seleccion
135             List resultados = obtenerResultados(rs);
136
137             if(resultados.size() > 0)
138             {
139                 return resultados;
140             }
141             else{
142                 return null;
143             }
144         } finally{
145             if (rs != null) {
146                 rs.close();
147             }
148             if (ps != null){
149                 ps.close();
150             }
151             if(connection != null){
152                 connection.close();
153             }
154         }
155     }
156
157
158     public Evento read(Evento evento) throws SQLException
159     {
160         obtenerConexion();
161         PreparedStatement ps = null;
162         ResultSet rs = null;
163
164         try {
165             ps = connection.prepareStatement(SQL_SELECT);
166             ps.setInt(1, evento.getIdEvento());
167             rs = ps.executeQuery(); // por ser de seleccion
168             List resultados = obtenerResultados(rs);
169
170             if(resultados.size() > 0)
171             {
172                 return (Evento)resultados.get(0);
173             }
174             else{
175                 return null;
176             }
177         } finally{
178             if(rs != null) rs.close();
179             if(ps != null) ps.close();
180         }
181     }
182
183
184     private List obtenerResultados(ResultSet rs) throws SQLException
185     {
186         List resultados = new ArrayList();
187
188         while(rs.next())

```

```
189     {
190         Evento evento = new Evento();
191         evento.setIdEvento(rs.getInt("idEvento"));
192         evento.setNombreEvento(rs.getString("nombreEvento"));
193         evento.setSede(rs.getString("sede"));
194         evento.setFechaInicio(rs.getDate("fechaInicio"));
195         evento.setFechaFin(rs.getDate("fechaFin"));
196
197         resultados.add(evento);
198     }
199
200     return resultados;
201 }
202 }
```

---

*Algoritmo:* Clase de acceso a datos (DAO).

### 3.3 *EventoServlet.java*

Esta clase hereda de `HttpServlet`, por lo que lo convierte en un servlet, de acuerdo al modelo vista controlador, esta clase es nuestro controlador, es que la parte de mas alto nivel, esta clase hará uso del DAO y del modelo. De acuerdo a la URI capturada en el navegador, sera la acción a procesar que caerá en alguno de los métodos de esta clase, dichos métodos coinciden con los definidos en el DAO, es decir el CRUD, donde cada método mandara a llamar a la respectiva aeración DAO, y los resultados los pintara en el navegador, donde las líneas html son embebidas dentro de cada método del servlet.

```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package com.escom.wad.baseproject.controller;
7
8  import com.escom.wad.baseproject.dao.EventoDAO;
9  import com.escom.wad.baseproject.model.Evento;
10 import java.io.IOException;
11 import java.io.PrintWriter;
12 import static java.lang.System.out;
13 import java.sql.Date;
14 import java.sql.SQLException;
15 import java.util.List;
16 import java.util.logging.Level;
17 import java.util.logging.Logger;
18 import javax.servlet.RequestDispatcher;
19 import javax.servlet.ServletException;
20 import javax.servlet.annotation.WebServlet;
21 import javax.servlet.http.HttpServlet;
22 import javax.servlet.http.HttpServletRequest;
23 import javax.servlet.http.HttpServletResponse;
24
25 /**
26  *
27  * @author aarongarcia
28  */
29 @WebServlet(name = "EventoServlet", urlPatterns = {"/EventoServlet"})
30 public class EventoServlet extends HttpServlet {
31
32     /**
33      * Processes requests for both HTTP GET and POST
34      * methods.
35      *
36      * @param request servlet request
37      * @param response servlet response
38      * @throws ServletException if a servlet-specific error occurs
39      * @throws IOException if an I/O error occurs
40      */
41     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
42         throws ServletException, IOException {
43
44         String accion = request.getParameter("accion");
45         if(accion.equals("listaDeEventos")){
46             listaDeEventos(request, response);
47         }else{
48             if(accion.equals("nuevo")){
49                 nuevoEvento(request, response);
50             }else{
51                 if(accion.equals("eliminar")){
52                     eliminarEvento(request, response);
53                 }else{
54                     if(accion.equals("actualizar")){
55                         actualizarEvento(request, response);
```

```

56         } else {
57             if (accion.equals("guardar")) {
58                 almacenarEvento(request, response);
59             } else {
60                 if (accion.equals("ver")) {
61                     verEvento(request, response);
62                 }
63             }
64         }
65     }
66 }
67 }
68 }
69
70 // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the
71 /**
72  * Handles the HTTP <code>GET</code> method.
73  *
74  * @param request servlet request
75  * @param response servlet response
76  * @throws ServletException if a servlet-specific error occurs
77  * @throws IOException if an I/O error occurs
78  */
79 @Override
80 protected void doGet(HttpServletRequest request, HttpServletResponse response)
81     throws ServletException, IOException {
82     processRequest(request, response);
83 }
84
85 /**
86  * Handles the HTTP <code>POST</code> method.
87  *
88  * @param request servlet request
89  * @param response servlet response
90  * @throws ServletException if a servlet-specific error occurs
91  * @throws IOException if an I/O error occurs
92  */
93 @Override
94 protected void doPost(HttpServletRequest request, HttpServletResponse response)
95     throws ServletException, IOException {
96     processRequest(request, response);
97 }
98
99 /**
100  * Returns a short description of the servlet.
101  *
102  * @return a String containing servlet description
103  */
104 @Override
105 public String getServletInfo() {
106     return "Short description";
107 } // </editor-fold>
108
109 private void listaDeEventos(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
110     response.setContentType("text/html; charset=UTF-8");
111     try (PrintWriter out = response.getWriter()) {
112         /* TODO output your page here. You may use following sample code. */
113         out.println("<!DOCTYPE html>");
114         out.println("<html>");
115         out.println("<head>");
116         out.println("<title>Lista de eventos</title>");
117         out.println("<link href='https://unpkg.com/boxicons@2.0.7/css/boxicons.min.css' rel='stylesheet'>");
118         out.println("<link rel='stylesheet' href='https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css' rel='stylesheet'>");
119         out.println("</head>");
120         out.println("<body>");
121

```

```

122 out.println("<div class=\"jumbotron jumbotron-fluid\">");
123 out.println("<div class=\"container\">");
124 out.println("<h1>ESCOM Eventos</h1>");
125 out.println("</div>");
126 out.println("</div>");
127
128 out.println("<div class='container'>");
129
130 out.println("<div style='margin-bottom:12px'>");
131 out.println("<h3>Lista de eventos</h3>");
132 out.println("<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed c");
133 out.println("<a href='EventoServlet?accion=nuevo' class='btn btn-success'> <i");
134 out.println("</div>");
135
136
137 out.println("<table class='table table-hover table-bordered '>");
138
139 out.println("<tr>");
140
141 out.println("<thead class='thead-dark'>");
142
143 out.println("<th class='text-center'>");
144 out.println("Clave de Evento");
145 out.println("</th>");
146
147 out.println("<th class='text-center'>");
148 out.println("Nombre del Evento");
149 out.println("</th>");
150
151 out.println("<th class='text-center'>");
152 out.println("Sede");
153 out.println("</th>");
154
155 out.println("<th class='text-center'>");
156 out.println("Fecha de Inicio");
157 out.println("</th>");
158
159 out.println("<th class='text-center'>");
160 out.println("Fecha de T rmino");
161 out.println("</th>");
162
163 out.println("<th class='text-center'>");
164 out.println("Acciones");
165 out.println("</th>");
166
167 out.println("</tr>");
168
169 out.println("</thead>");
170
171 int idEvento;
172 String nombreEvento;
173 String sede;
174 Date fechaInicio;
175 Date fechaTermino;
176
177 EventoDAO dao = new EventoDAO();
178 try {
179 List lista = dao.readAll();
180 for (int i = 0; i < lista.size(); i++) {
181 Evento e = (Evento) lista.get(i);
182 idEvento = e.getIdEvento();
183 nombreEvento = e.getNombreEvento();
184 sede = e.getSede();
185 fechaInicio = e.getFechaInicio();
186 fechaTermino = e.getFechaFin();
187

```

```

188         out.println("<tr>");
189         out.println("<td class='text-center>" + idEvento + "</td>");
190         out.println("<td>" + nombreEvento + "</td>");
191         out.println("<td>" + sede + "</td>");
192         out.println("<td class='text-center>" + fechaInicio + "</td>");
193         out.println("<td class='text-center>" + fechaTermino + "</td>");
194
195         out.println("<td class='text-center>");
196         // Ocultar mediante un formulario para no ver id, si no quiero mostrar nada
197         out.println("<a href='EventoServlet?accion=eliminar&id=" + idEvento + "' class='");
198
199         out.println("<a href='EventoServlet?accion=actualizar&id=" + idEvento + "' clas");
200         out.println("<a href='EventoServlet?accion=ver&id=" + idEvento + "' class='btn");
201
202         out.println("</td>");
203
204         out.println("</tr>");
205     }
206     } catch (SQLException e) {
207
208     }
209
210     out.println("</table>");
211
212     out.println("</div>");
213     out.println("</body>");
214     out.println("</html>");
215
216 }
217 }
218 }
219
220 private void nuevoEvento(HttpServletRequest request, HttpServletResponse response) {
221     try {
222         //response.sendRedirect("eventosForm.html");
223         RequestDispatcher vista = request.getRequestDispatcher("createForm.html");
224         vista.forward(request, response);
225     } catch (IOException ex) {
226         Logger.getLogger(EventoServlet.class.getName()).log(Level.SEVERE, null, ex);
227     } catch (ServletException ex) {
228         Logger.getLogger(EventoServlet.class.getName()).log(Level.SEVERE, null, ex);
229     }
230 }
231
232 private void eliminarEvento(HttpServletRequest request, HttpServletResponse response) throws
233     // Capturar los posibles valores
234     EventoDAO dao = new EventoDAO();
235     Evento e = new Evento();
236
237     try {
238         e.setIdEvento(Integer.parseInt(request.getParameter("id")));
239         dao.delete(e);
240
241         response.sendRedirect("EventoServlet?accion=listaDeEventos");
242
243         //listaDeEventos(request, response);
244     } catch (SQLException ex) {
245         Logger.getLogger(EventoServlet.class.getName()).log(Level.SEVERE, null, ex);
246     }
247 }
248
249 private void actualizarEvento(HttpServletRequest request, HttpServletResponse response) throws
250
251     EventoDAO dao = new EventoDAO();
252     Evento e = new Evento();
253

```



```

254     try {
255         e.setIdEvento(Integer.parseInt(request.getParameter("id")));
256         e = dao.read(e);
257
258
259         try (PrintWriter out = response.getWriter()) {
260             /* TODO output your page here. You may use following sample code. */
261             out.println("<!DOCTYPE html>");
262             out.println("<html>");
263             out.println("<head>");
264             out.println("<title>Actualizar evento</title>");
265             out.println("<link href='https://unpkg.com/boxicons@2.0.7/css/boxicons.min.css' rel='stylesheet'>");
266             out.println("<link rel='stylesheet' href='https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css' rel='stylesheet'>");
267             out.println("</head>");
268             out.println("<body>");
269
270             out.println("<div class='jumbotron jumbotron-fluid'>");
271                 out.println("<div class='container'>");
272                     out.println("<h1>ESCOM Eventos</h1>");
273                     out.println("</div>");
274                 out.println("</div>");
275
276             out.println("<div class='container'>");
277
278                 out.println("<div class='row justify-content-center'>");
279                     out.println("<div class='col-md-8'>");
280                         out.println("<div class='card'>");
281                             out.println("<div class='card-header'>Actualizar evento</div>");
282                             out.println("<div class='card-body'>");
283
284                                 out.println("<form method='POST' action='EventoServlet?accion=guardar'>");
285
286                                     out.println("<div class='form-group row'>");
287                                         out.println("<label for='id' class='col-md-4 col-form-label'>ID</label>");
288                                         out.println("<div class='col-md-6'>");
289                                             out.println("<input type='text' value='<id>'>");
290                                         out.println("</div>");
291                                     out.println("</div>");
292
293                                     out.println("<div class='form-group row'>");
294                                         out.println("<label for='nombreEvento' class='col-md-4 col-form-label'>Nombre</label>");
295                                         out.println("<div class='col-md-6'>");
296                                             out.println("<input type='text' value='<nombreEvento>'>");
297                                         out.println("</div>");
298                                     out.println("</div>");
299                                     out.println("<div class='form-group row'>");
300                                         out.println("<label for='sede' class='col-md-4 col-form-label'>Sede</label>");
301                                         out.println("<div class='col-md-6'>");
302                                             out.println("<input type='text' value='<sede>'>");
303                                         out.println("</div>");
304                                     out.println("</div>");
305                                     out.println("<div class='form-group row'>");
306                                         out.println("<label for='fechaInicio' class='col-md-4 col-form-label'>Fecha Inicio</label>");
307                                         out.println("<div class='col-md-6'>");
308                                             out.println("<input type='date' value='<fechaInicio>'>");
309                                         out.println("</div>");
310                                     out.println("</div>");
311                                     out.println("<div class='form-group row'>");
312                                         out.println("<label for='fechaFin' class='col-md-4 col-form-label'>Fecha Fin</label>");
313                                         out.println("<div class='col-md-6'>");
314                                             out.println("<input type='date' value='<fechaFin>'>");
315                                         out.println("</div>");
316                                     out.println("</div>");
317                                     out.println("<div class='form-group row'>");
318                                         out.println("<button type='submit' class='btn btn-primary'>Actualizar</button>");
319                                     out.println("</div>");
320                                 out.println("</div>");
321                             out.println("</div>");
322                         out.println("</div>");
323                     out.println("</div>");
324                 out.println("</div>");
325             out.println("</body>");
326         }
327     }
328 }

```

```

320         out.println("</div>");
321         out.println("</div>");
322         out.println("</div>");
323
324
325
326
327         out.println("<a href='EventoServlet?accion=listaDeEventos' class='btn btn-outline-success'>");
328
329         out.println("</div>");
330         out.println("</body>");
331         out.println("</html>");
332     }
333
334
335     } catch (SQLException ex) {
336         Logger.getLogger(EventoServlet.class.getName()).log(Level.SEVERE, null, ex);
337     }
338 }
339
340 private void almacenarEvento(HttpServletRequest request, HttpServletResponse response) throws IOException {
341     Evento e = new Evento();
342     EventoDAO dao = new EventoDAO();
343
344     System.out.println(request.getParameter("id"));
345     // nuevo evento
346     if (request.getParameter("id") == null || request.getParameter("id").isEmpty()) {
347         e.setNombreEvento(request.getParameter("nombreEvento"));
348         e.setSede(request.getParameter("sede"));
349         e.setFechaInicio(Date.valueOf(request.getParameter("fechaInicio")));
350         e.setFechaFin(Date.valueOf(request.getParameter("fechaFin")));
351
352         try {
353             dao.create(e);
354             response.sendRedirect("EventoServlet?accion=listaDeEventos");
355
356         } catch (SQLException ex) {
357             Logger.getLogger(EventoServlet.class.getName()).log(Level.SEVERE, null, ex);
358         }
359     } else {
360         e.setIdEvento(Integer.parseInt(request.getParameter("id")));
361         e.setNombreEvento(request.getParameter("nombreEvento"));
362         e.setSede(request.getParameter("sede"));
363         e.setFechaInicio(Date.valueOf(request.getParameter("fechaInicio")));
364         e.setFechaFin(Date.valueOf(request.getParameter("fechaFin")));
365
366         try {
367             dao.update(e);
368             response.sendRedirect("EventoServlet?accion=listaDeEventos");
369         } catch (SQLException ex) {
370             Logger.getLogger(EventoServlet.class.getName()).log(Level.SEVERE, null, ex);
371         }
372     }
373 }
374
375
376 private void verEvento(HttpServletRequest request, HttpServletResponse response) throws IOException {
377     EventoDAO dao = new EventoDAO();
378     Evento e = new Evento();
379
380     try {
381         e.setIdEvento(Integer.parseInt(request.getParameter("id")));
382         e = dao.read(e);
383
384
385         try (PrintWriter out = response.getWriter()) {

```

```

386      /* TODO output your page here. You may use following sample code. */
387      out.println("<!DOCTYPE html>");
388      out.println("<html>");
389      out.println("<head>");
390      out.println("<title>Datos de evento</title>");
391      out.println("<link href='https://unpkg.com/boxicons@2.0.7/css/boxicons.min.css' rel='stylesheet'>");
392      out.println("<link rel='stylesheet' href='https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css'>");
393      out.println("</head>");
394      out.println("<body>");
395
396      out.println("<div class='jumbotron jumbotron-fluid'>");
397          out.println("<div class='container'>");
398              out.println("<h1>ESCOM Eventos</h1>");
399              out.println("</div>");
400          out.println("</div>");
401
402      out.println("<div class='container'>");
403
404
405      out.println("<div class='card mb-3' style='max-width: 1200px;'>");
406          out.println("<div class='row no-gutters'>");
407              out.println("<div class='col-md-4'>");
408                  out.println("<img src='./IMAGENES/escom.jpg' class='card-img' alt='...'>");
409              out.println("</div>");
410              out.println("<div class='col-md-8'>");
411                  out.println("<div class='card-body'>");
412                      out.println("<h5 class='card-title'>ID de evento: " + e.getIdEvento() + "</h5>");
413                      out.println("<p class='card-text'>Nombre: " + e.getNombreEvento() + "</p>");
414                      out.println("<p class='card-text'>Sede: " + e.getSede() + "</p>");
415                      out.println("<p class='card-text'>Fecha de inicio: " + e.getFechaInicio() + "</p>");
416                      out.println("<p class='card-text'>Fecha de termino: " + e.getFechaFin() + "</p>");
417                      out.println("<p class='card-text'><small class='text-muted'>Información adicional</small>");
418                  out.println("</div>");
419              out.println("</div>");
420          out.println("</div>");
421      out.println("</div>");
422
423      out.println("<a href='EventoServlet?accion=listaDeEventos' class='btn btn-outline-success'>Ver eventos</a>");
424
425      out.println("</div>");
426      out.println("</body>");
427      out.println("</html>");
428      // tarea terminar fucnionalidad para actualizar un evento
429  }
430
431
432  } catch (SQLException ex) {
433      Logger.getLogger(EventoServlet.class.getName()).log(Level.SEVERE, null, ex);
434  }
435  }
436
437  }

```

---

*Algoritmo:* Clase Evento

## 4 Resultados

A lo largo de las sesiones virtuales que tuvimos, desarrollamos aproximadamente el 80% de la practica en lo que a código fuente se refiere, así que solo quedo pendiente en el servlet el método de actualización, el cual lo realice satisfactoriamente haciendo uso del método almacenarEvento, pero añadiendo el caso cuando el identificador de evento no es nulo, ya que se quiere actualizar sobre el id.

### 4.1 Ejecución del Programa

#### 4.1.1 Inicio

Al ejecutar el programa, en automático nos abre una ventana en el navegador predeterminado, lo siguiente:

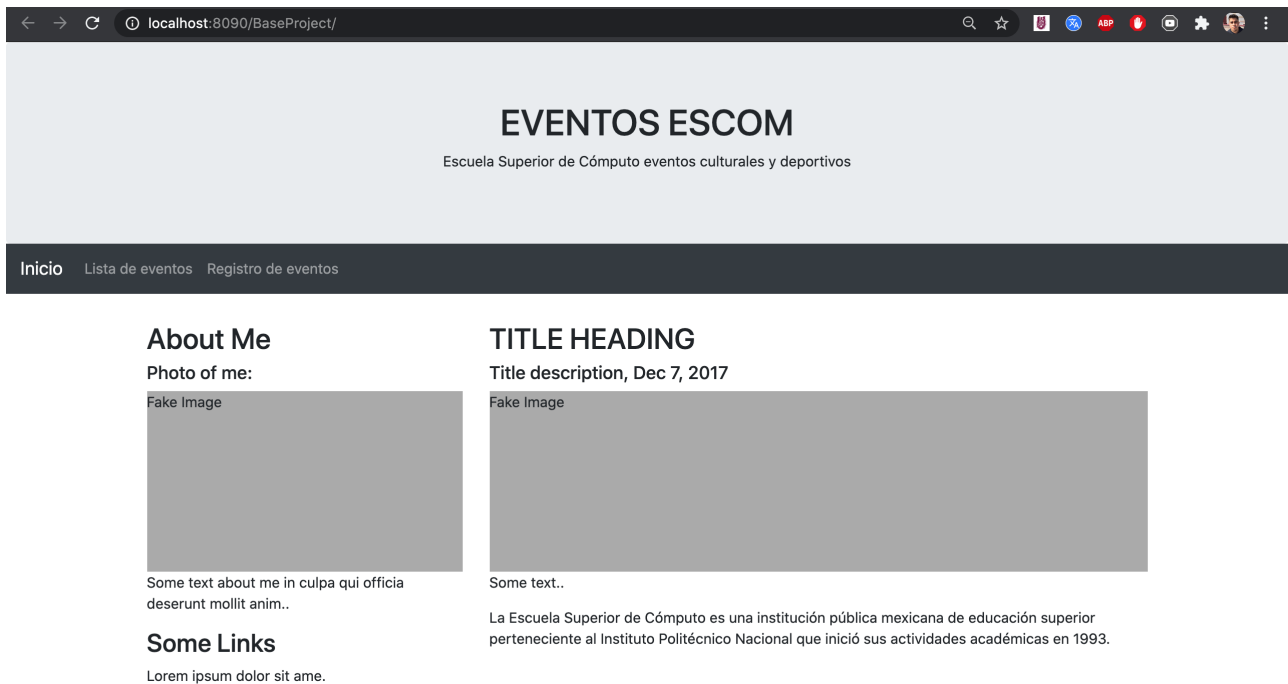
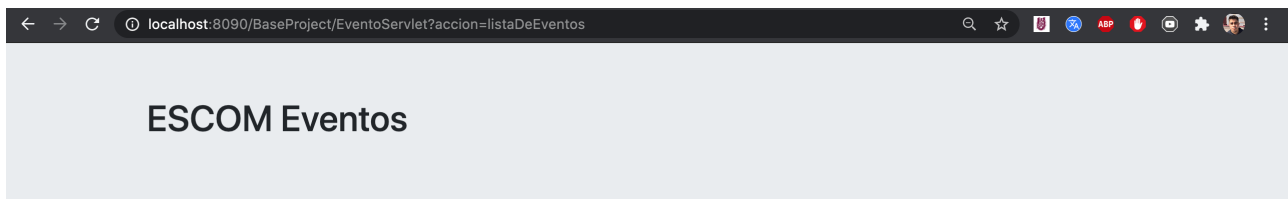


Figura 1: Página de inicio

#### 4.1.2 *Lista de eventos*

Se muestran todos los eventos registrados, la ultima columna del lado derecho, encontramos los iconos donde podemos eliminar, actualizar o ver un evento, en la parte superior encontramos un botón "nuevo evento", donde como su nombre lo dice, es para crear nuevos eventos.



**Lista de eventos**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Maecenas ultricies mi eget mauris pharetra et ultrices. Iaculis nunc sed augue lacus viverra vitae congue. Amet consectetur adipiscing elit dui tristique sollicitudin nibh. Velit sed ullamcorper morbi tincidunt ornare massa. Odio tempor orci dapibus ultrices in iaculis nunc. Eu lobortis elementum nibh tellus molestie nunc non.

[Nuevo Evento](#)
















Clave de Evento	Nombre del Evento	Sede	Fecha de Inicio	Fecha de Término	Acciones
39	Feria de empleos	Pasillo cultural	2020-11-09	2020-11-13	  
40	Registro de protocolos	CATT	2020-11-03	2020-11-09	  
41	Calaberitas	Pasillo cultural	2020-10-31	2020-11-02	  
42	Semana de ciencias sociales	Sala 14	2020-11-16	2020-11-20	  
43	Kermes	Cancha de basquetbol	2020-11-23	2020-11-24	  

Figura 2: Página que lista todos los eventos previamente registrados

#### 4.1.3 *Ver elemento*

Dentro de esta página, podemos ver los datos de un determinado evento, cada pagina incluye un botón con la leyenda "regresar", que regresa a la página de lista de eventos.

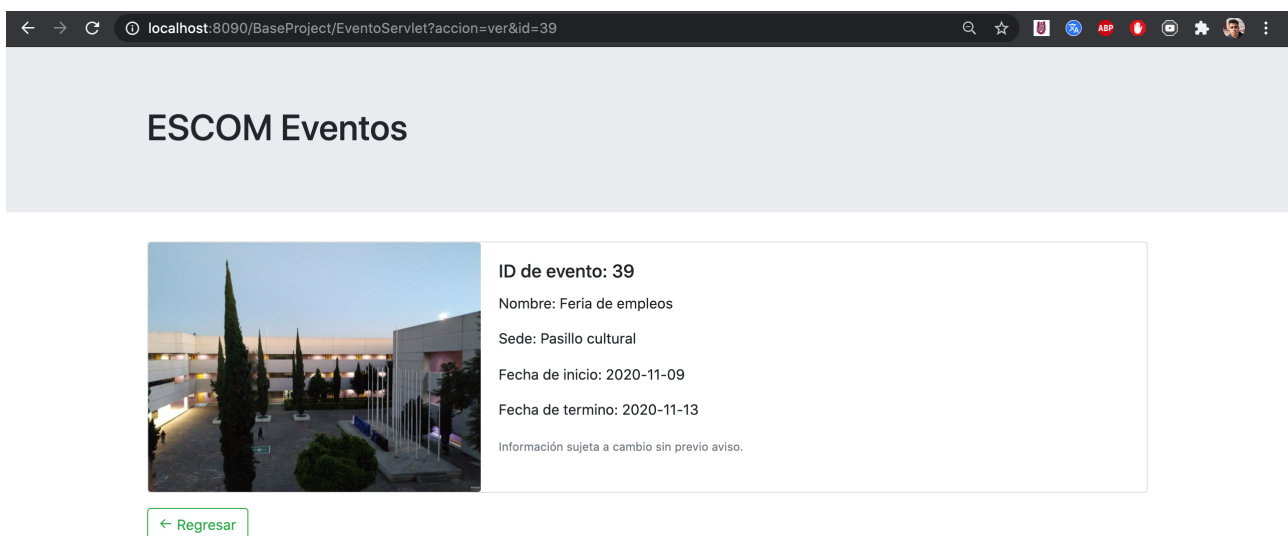


Figura 3: Página de vista de un evento en específico

#### 4.1.4 Actualizar evento

Esta página nos permite actualizar un evento, donde todos los atributos de un evento son modificables, exceptuando el identificador.

localhost:8090/BaseProject/EventoServlet?accion=actualizar&id=39

## ESCOM Eventos

### Actualizar evento

Identificador	39
Nombre:	Feria de empleos
Sede:	Pasillo cultural
Fecha Inicio:	09/11/2020
Fecha Término:	13/11/2020

Actualizar

← Regresar

Figura 4: Página de actualizar por ejemplo el evento 39

#### 4.1.5 Nuevo evento

Esta página nos permite registrar un nuevo evento, como validación se incluyó que el nombre y la sede no pueden ser nulos, etc.

localhost:8090/BaseProject/EventoServlet?accion=nuevo

# ESCOM Eventos

Registrar Evento

Nombre:

Sede:

Fecha Inicio:

Fecha Término:

[Registrar](#)

[← Regresar](#)

Figura 5: Formulario de crear evento

## 5 Conclusión

Jamás había trabajado con servlets, no se que tan actual sea y si aun se sigan utilizando, en lo personal no me gusta mezclar el código html como tal, pero es una buena aproximación para iniciar esta unidad de aprendizaje.

La dificultad más grande que tuve al realizar esta practica fue a la hora de hacer la pagina html de la operación actualizar, específicamente en el punto de obtener el id, ya que dentro del formulario, jamas puse value igual con el id, sino que lo puse en palceholder, fue una tontería pero no lo detectaba, y no me actualizaba, solo me agregaba nuevos eventos con todo igual menos con el identificador, ya que internamente en el servlet, se detecta que el identificador esta vacío, por lo que se trata de un nuevo evento.



## 6 Referencias Bibliográficas

### Referencias

[applelives, 2017] applelives. *How to Install Apache Tomcat 9 on Mac OS X*

[Neosoft , 2018] Neosoft. *¿Qué es una aplicación Web?*

[Apache Software Foundation, 2020] Apache Software Foundation. *Apache Tomcat 9 documentation*