

Examen Parcial

2024

Laboratorio de Servidores
Web

Documentación de Examen Segundo
Parcial

Alumna: Sofía Rendón Aragón

Docente: Alejandro Diaz Ruiz

Contenido

.....	1
Construcción de la base de datos	2
Tablas.....	2
Triggers.....	3
Estructura del proyecto.....	4
Métodos e implementación.....	4
Métodos doGet	5
Estudiantes	5
Cursos	7
Inscripciones.....	9
Métodos doPost.....	12
Estudiantes	12
Cursos	14
Inscripciones.....	16
Métodos doDelete.....	16
Estudiantes	17
Cursos	18
Inscripciones.....	20

Construcción de la base de datos

Tablas

```
1 CREATE TABLE estudiante (  
2     id_estudiante INT PRIMARY KEY,  
3     nombre VARCHAR(100) NOT NULL,  
4     carrera ENUM('Civil', 'Gastronomia', 'Software', 'Contaduria', 'Ambiental') NOT NULL,  
5     semestre INT NOT NULL  
6 );
```

```

1 CREATE TABLE curso (
2     id_curso INT PRIMARY KEY,
3     nombre VARCHAR(50) NOT NULL,
4     descripcion VARCHAR(150),
5     credits DOUBLE NOT NULL,
6     nivel ENUM('Básico', 'Intermedio', 'Avanzado') NOT NULL
7 );
8
9 CREATE TABLE inscripcion (
10     id_estudiante INT NOT NULL,
11     id_curso INT NOT NULL,
12     fecha_inscripcion DATETIME NOT NULL,
13     FOREIGN KEY (id_estudiante) REFERENCES estudiante(id_estudiante),
14     FOREIGN KEY (id_curso) REFERENCES curso(id_curso)
15 );

```

Triggers

```

1 CREATE TRIGGER `trigger_borrarInscripciones1` BEFORE DELETE ON
  `estudiante`
2 FOR EACH ROW BEGIN
3     DELETE FROM inscripcion WHERE id_estudiante = OLD.id_estudiante;
4 END

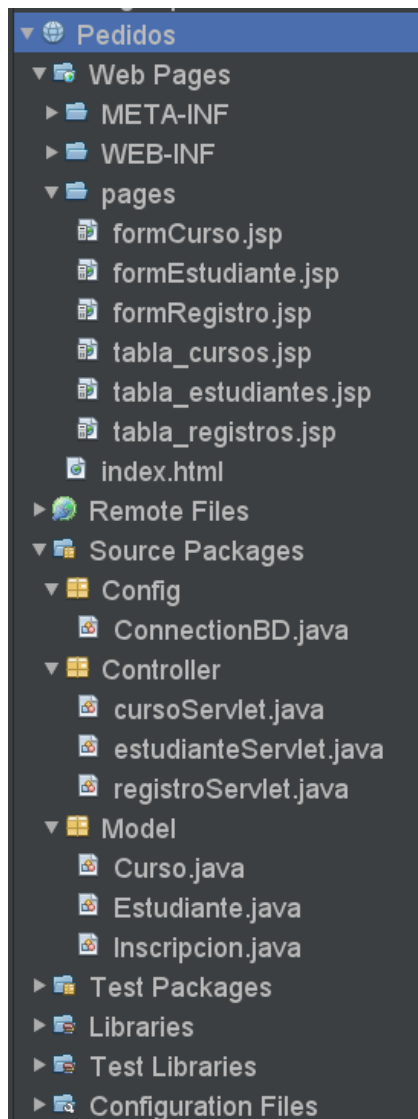
```

```

1 CREATE TRIGGER `trigger_borrarInscripciones2` BEFORE DELETE
  ON `curso`
2 FOR EACH ROW BEGIN
3     DELETE FROM inscripcion WHERE id_curso = OLD.id_curso;
4 END

```

Estructura del proyecto



Métodos e implementación

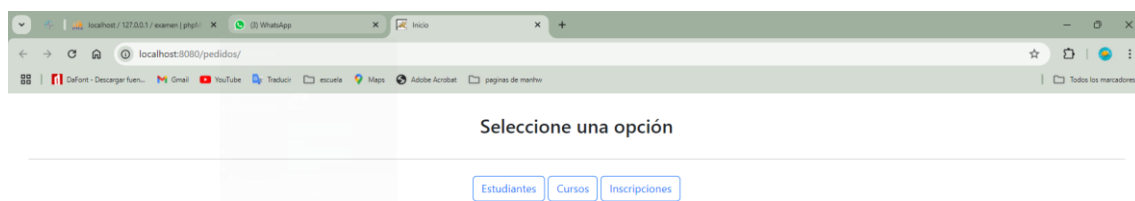


Imagen 1. Pantalla de inicio

Métodos doGet

Estudiantes

```
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    System.out.println("Se ejecuta el doGet");
    ConnectionBD conexion = new ConnectionBD();
    List<Estudiante> listaEstudiantes = new ArrayList<>();
    String sql = "SELECT id_estudiante, nombre, carrera, semestre FROM estudiante";

    try {
        conn = conexion.getConnectionBD();
        PreparedStatement ps = conn.prepareStatement(sql);
        ResultSet rs = ps.executeQuery();

        while (rs.next()) {
            Estudiante estudiante = new Estudiante();
            estudiante.setId(rs.getInt("id_estudiante"));
            estudiante.setNombre(rs.getString("nombre"));
            estudiante.setCarrera(rs.getString("carrera"));
            estudiante.setSemestre(rs.getInt("semestre"));
            listaEstudiantes.add(estudiante);
        }

        // Pasa la lista de usuarios al JSP
        request.setAttribute("estudiantes", listaEstudiantes);
        request.getRequestDispatcher("/pages/tabla_estudiantes.jsp").forward(request, response);

    } catch (Exception e) {
        e.printStackTrace();
        response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR, "Error al obtener los estudiantes" + e);
    } finally {
        // Close resources
        // Close resources
        try {
            if (rs != null) {
                rs.close();
            }
            if (ps != null) {
                ps.close();
            }
            if (conn != null && !conn.isClosed()) {
                conn.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Imagen 2. Método doGet

```

<table class="table table-bordered table-striped table-hover" >
  <thead>
    <tr>
      <th scope="col">ID</th>
      <th scope="col">Nombre</th>
      <th scope="col">Semestre</th>
      <th scope="col">Carrera</th>
      <th scope="col">Acciones</th>
    </tr>
  </thead>
  <tbody >
    <%
      ArrayList<Estudiante> listaEstudiantes = (ArrayList<Estudiante>)
      request.getAttribute("estudiantes");

      if (listaEstudiantes != null && !listaEstudiantes.isEmpty()) {
        for (Estudiante est : listaEstudiantes) {
          <%>
          <tr>
            <th scope="row"><%= est.getId() %></th>
            <td><%= est.getNombre() %></td>
            <td><%= est.getSemestre() %></td>
            <td><%= est.getCarrera() %></td>
            <td >
              <button type="button" class="btn btn-outline-danger"
              onclick="eliminarEstudiante(<%= est.getId() %>)">
                <i class="bi bi-trash3-fill"></i>
              </button>
            </td>
          </tr>
          <%
            }
          } else {
            <%>
            <tr>
              <td colspan="5">No hay estudiantes registrados.</td>
            </tr>
            <%>
          }
        <%>
      </tbody>
    </table>

```

Imagen 3. Implementación en tabla_estudiantes.jsp

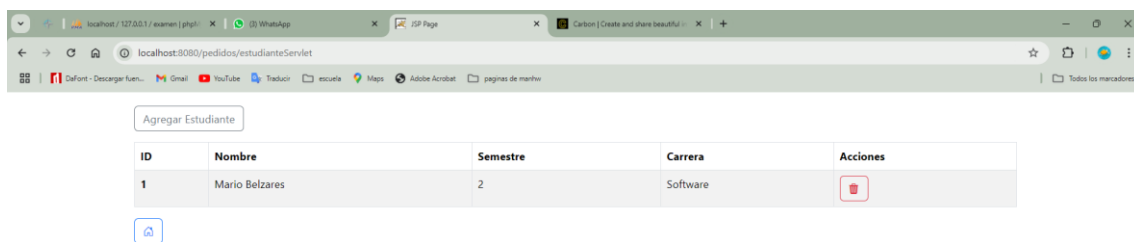


Imagen 4. Funcionalidad

Cursos

```
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    System.out.println("Se ejecuta el doGet");
    ConnectionBD conexion = new ConnectionBD();
    List<Curso> listaCursos = new ArrayList<>();
    String sql = "SELECT id_curso, nombre, descripcion, creditos, nivel FROM curso";

    try {
        conn = conexion.getConnectionBD();
        PreparedStatement ps = conn.prepareStatement(sql);
        ResultSet rs = ps.executeQuery();

        // Itera sobre los resultados y crea objetos UsuarioModel
        while (rs.next()) {
            Curso curso = new Curso();
            curso.setId(rs.getInt("id_curso"));
            curso.setNombre(rs.getString("nombre"));
            curso.setDescripcion(rs.getString("descripcion"));
            curso.setCreditos(rs.getDouble("creditos"));
            curso.setNivel(rs.getString("nivel"));
            listaCursos.add(curso);
        }

        // Pasa la lista de usuarios al JSP
        request.setAttribute("cursos", listaCursos);
        request.getRequestDispatcher("/pages/tabla_cursos.jsp").forward(request, response);
    } catch (Exception e) {
        e.printStackTrace();
        response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
            "Error al obtener los cursos" + e);
    } finally {
        // Close resources
        // Close resources
        try {
            if (rs != null) {
                rs.close();
            }
            if (ps != null) {
                ps.close();
            }
            if (conn != null && !conn.isClosed()) {
                conn.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Imagen 5. Método doGet

```
<table class="table table-bordered table-striped table-hover" >
  <thead>
    <tr>
      <th scope="col">ID</th>
      <th scope="col">Nombre</th>
      <th scope="col">Descripción</th>
      <th scope="col">Créditos</th>
      <th scope="col">Nivel</th>
      <th scope="col">Acciones</th>
    </tr>
  </thead>
  <tbody >
    <%
      ArrayList<Curso> listaCursos = (ArrayList<Curso>) request.getAttribute("cursos");

      if (listaCursos != null && !listaCursos.isEmpty()) {
        for (Curso est : listaCursos) {
          %>
          <tr>
            <th scope="row"><%= est.getId()%></th>
            <td><%= est.getNombre()%></td>
            <td><%= est.getDescripcion()%></td>
            <td><%= est.getCreditos()%></td>
            <td><%= est.getNivel()%></td>
            <td >
              <button type="button" class="btn btn-outline-danger"
                onclick="eliminarCurso(<%= est.getId() %>)">
                <i class="bi bi-trash3-fill"></i>
              </button>
            </td>
          </tr>
          <%
            }
          } else {
            %>
            <tr>
              <td colspan="6">No hay cursos registrados.</td>
            </tr>
            <%
              }
            %>
          </tbody>
        </table>
```

Imagen 6. Implementación en tabla_cursos.jsp



Imagen 7. Funcionalidad

Inscripciones

```
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    System.out.println("Se ejecuta el doGet");
    String action = request.getParameter("action");

    if (action == null) {
        cargarRegistros(request, response);
    } else if (action.equals("loadIds")) {
        loadStudents(request, response);
        loadCursos(request, response);
        request.getRequestDispatcher("/pages/formRegistro.jsp").forward(request, response);
    }
}
```

Imagen 8. Método doGet

```
<table class="table table-bordered table-striped table-hover" >
  <thead>
    <tr>
      <th scope="col">Id Estudiante</th>
      <th scope="col">Id Curso</th>
      <th scope="col">Fecha de Inscripción</th>
      <th scope="col">Acciones</th>
    </tr>
  </thead>
  <tbody>
    <%
      ArrayList<Inscripcion> listaCursos = (ArrayList<Inscripcion>) request.getAttribute("cursos");

      if (listaCursos != null && !listaCursos.isEmpty()) {
        for (Inscripcion est : listaCursos) {
          %>
          <tr>
            <th scope="row"><%= est.getId_estudiante()%></th>
            <th><%= est.getId_curso()%></th>
            <td><%= est.getFecha()%></td>
            <td>
              <button type="button" class="btn btn-outline-danger"
                onclick="eliminarIns(<%= est.getId_estudiante()%>, <%= est.getId_curso()%>)">
                <i class="bi bi-trash3-fill"></i>
              </button>
            </td>
          </tr>
          <%
            }
          } else {
            %>
            <tr>
              <td colspan="4">No hay inscripciones registradas.</td>
            </tr>
            <%
              }
            %>
          </tbody>
        </table>
```

Imagen 9. Implementación en tabla_registros.jsp

```

<form method="post" action="${pageContext.request.contextPath}/registroServlet">

    <label for="id_est">Id Estudiante</label>
    <select id="id_est" name="id_est" required>
        <%
            ArrayList<Estudiante> idsEstudiantes = (ArrayList<Estudiante>)
            request.getAttribute("estudiantes");

            if (idsEstudiantes != null && !idsEstudiantes.isEmpty()) {
                for (Estudiante est : idsEstudiantes) {
                    %>
                    <option value="<%= est.getId() %>"><%= est.getNombre() %></option>
                    <%
                }
            } else {

                %>
                <option > No hay estudiantes</option>
                <%
            }
        %>
    </select>

    <label for="id_cur">Id Curso:</label>
    <select id="id_cur" name="id_cur" required>
        <%
            ArrayList<Curso> idsCursos = (ArrayList<Curso>) request.getAttribute("cursos");

            if (idsCursos != null && !idsCursos.isEmpty()) {
                for (Curso cr : idsCursos) {
                    %>
                    <option value="<%= cr.getId() %>"><%= cr.getNombre() %></option>
                    <%
                }
            } else {

                %>
                <option > No hay cursos</option>
                <%
            }
        %>
    </select>

    <input type="submit" value="Confirmar">
</form>

```

Imagen 10. Implementación en formRegistro.jsp

La implementación de la imagen 9 hace que se muestren las inscripciones de la base de datos (imagen 11) y la implementación de la imagen 10 hace que los cursos y estudiantes de la base de datos aparezcan como opciones del select en el form para agregar inscripciones (imagen 12).

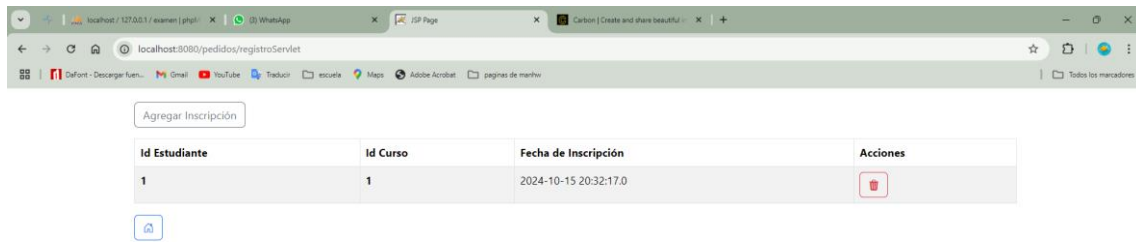


Imagen 11. Implementacion 1

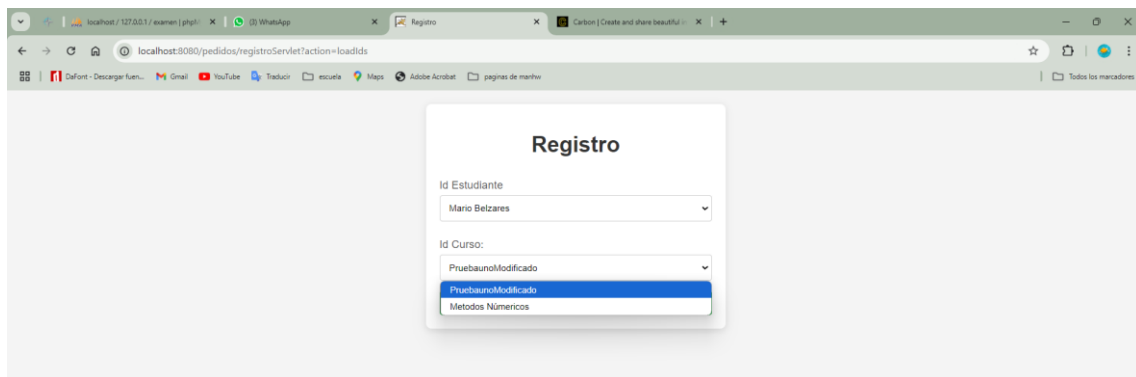


Imagen 12. Implementación 2

Métodos doPost

Estudiantes

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

    request.setCharacterEncoding("UTF-8");
    response.setCharacterEncoding("UTF-8");

    ConnectionBD conexion = new ConnectionBD();

    // Obtener los parámetros del formulario
    String idStr = request.getParameter("id");
    String nombre = request.getParameter("nombre");
    String semestreStr = request.getParameter("semestre");
    String carrera = request.getParameter("carrera");

    int id = Integer.parseInt(idStr);
    int semestre = Integer.parseInt(semestreStr);

    try {
        // Crear la consulta SQL para insertar el usuario
        String sql = "INSERT INTO estudiante (id_estudiante, nombre, semestre, carrera)"
            + "VALUES (?, ?, ?, ?)";
        conn = conexion.getConnectionBD();
        ps = conn.prepareStatement(sql);
        ps.setInt(1, id);
        ps.setString(2, nombre);
        ps.setInt(3, semestre);
        ps.setString(4, carrera);

        // Ejecutar la consulta
        int filasInsertadas = ps.executeUpdate();
        if (filasInsertadas > 0) {
            // Si se insertó correctamente, mostrar los registros
            response.sendRedirect(request.getContextPath() + "/estudianteServlet");
        } else {
            // Si falló, redirigir a una página de error
            request.setAttribute("mensaje", "Error al registrar estudiante.");
            request.getRequestDispatcher("/pages/formEstudiante.jsp").forward(request, response);
        }
    } catch (SQLException e) {
        e.printStackTrace();
        request.setAttribute("mensaje", "Ocurrió un error: " + e.getMessage());
        request.getRequestDispatcher("/pages/formEstudiante.jsp").forward(request, response);
    } finally {
        // Cerrar los recursos
        try {
            if (rs != null) {
                rs.close();
            }
            if (ps != null) {
                ps.close();
            }
            if (conn != null) {
                conn.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Imagen 13. Método doPost'

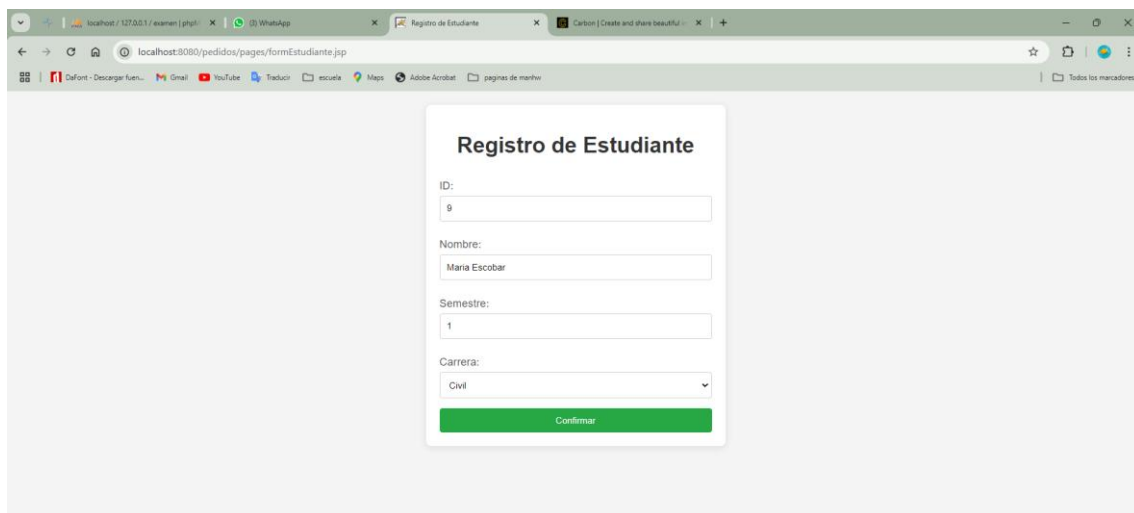
```
<form method="post" action="${pageContext.request.contextPath}/estudianteServlet">
  <label for="id">ID:</label>
  <input type="number" name="id" id="id" required>

  <label for="nombre">Nombre:</label>
  <input type="text" name="nombre" id="nombre" required>

  <label for="semestre">Semestre:</label>
  <input type="number" name="semestre" id="semestre" required>

  <label for="carrera">Carrera:</label>
  <select id="carrera" name="carrera" required>
    <option value="Civil">Civil</option>
    <option value="Gastronomia">Gastronomia</option>
    <option value="Software">Software</option>
    <option value="Contaduria">Contaduria</option>
    <option value="Contaduria">Ambiental</option>
  </select>
  <input type="submit" value="Confirmar">
</form>
```

Imagen 14. Implementación en formEstudiante.jsp



The screenshot shows a web browser window with the URL `localhost:8080/pedidos/pages/formEstudiante.jsp`. The page displays a form titled "Registro de Estudiante". The form has four input fields: "ID:" with the value "9", "Nombre:" with the value "Maria Escobar", "Semestre:" with the value "1", and "Carrera:" which is a dropdown menu currently showing "Civil". Below these fields is a green button labeled "Confirmar". The browser's address bar and tabs are visible at the top.

Imagen 15. Funcionalidad

```
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    request.setCharacterEncoding("UTF-8");
    response.setCharacterEncoding("UTF-8");

    ConnectionBD conexion = new ConnectionBD();

    // Obtener los parámetros del formulario
    String idStr = request.getParameter("id");
    String nombre = request.getParameter("nombre");
    String descripcion = request.getParameter("descripcion");
    String creditosStr = request.getParameter("creditos");
    String nivel = request.getParameter("nivel");

    int id = Integer.parseInt(idStr);
    Double creditos = Double.parseDouble(creditosStr);

    try {
        // Crear la consulta SQL para insertar el usuario
        String sql = "INSERT INTO curso (id_curso, nombre, descripcion, creditos, nivel) VALUES (?, ?, ?, ?, ?)";
        conn = conexion.getConnectionBD();
        ps = conn.prepareStatement(sql);
        ps.setInt(1, id);
        ps.setString(2, nombre);
        ps.setString(3, descripcion);
        ps.setDouble(4, creditos);
        ps.setString(5, nivel);

        // Ejecutar la consulta
        int filasInsertadas = ps.executeUpdate();
        if (filasInsertadas > 0) {
            // Si se insertó correctamente, mostrar los registros
            response.sendRedirect(request.getContextPath() + "/cursoServlet");
        } else {
            // Si falló, redirigir a una página de error
            request.setAttribute("mensaje", "Error al registrar curso.");
            request.getRequestDispatcher("/pages/formCurso.jsp").forward(request, response);
        }
    } catch (SQLException e) {
        e.printStackTrace();
        request.setAttribute("mensaje", "Ocurrió un error: " + e.getMessage());
        request.getRequestDispatcher("/pages/formEstudiante.jsp").forward(request, response);
    } finally {
        // Cerrar los recursos
        try {
            if (rs != null) {
                rs.close();
            }
            if (ps != null) {
                ps.close();
            }
            if (conn != null) {
                conn.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Imagen 16. Método doPost

```
<form method="post" action="${pageContext.request.contextPath}/cursoServlet">
  <label for="id">ID:</label>
  <input type="number" name="id" id="id" required>

  <label for="nombre">Nombre:</label>
  <input type="text" name="nombre" id="nombre" required>

  <label for="descripcion">Descripción</label>
  <input type="text" name="descripcion" id="descripcion" required>

  <label for="creditos">Créditos</label>
  <input type="number" step="0.1" name="creditos" id="creditos" required>

  <label for="nivel">Nivel:</label>
  <select id="nivel" name="nivel" required>
    <option value="Básico">Básico</option>
    <option value="Intermedio">Intermedio</option>
    <option value="Avanzado">Avanzado</option>
  </select>
  <input type="submit" value="Confirmar">
</form>
```

Imagen 17. Implementación en formCurso.jsp

Registro de Curso

ID:
3

Nombre:
Humanidades

Descripción
Diosito

Créditos
12.5

Nivel:
Básico

Confirmar

Imagen 18. Funcionalidad

Inscripciones

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

    request.setCharacterEncoding("UTF-8");
    response.setCharacterEncoding("UTF-8");

    ConnectionBD conexion = new ConnectionBD();

    // Obtener los parámetros del formulario
    String idEstStr = request.getParameter("id_est");
    String idCurStr = request.getParameter("id_cur");

    int id_est = Integer.parseInt(idEstStr);
    int id_cur = Integer.parseInt(idCurStr);

    try {
        // Crear la consulta SQL para insertar el usuario
        String sql = "INSERT INTO inscripcion (id_estudiante, id_curso) VALUES (?, ?)";
        conn = conexion.getConnectionBD();
        ps = conn.prepareStatement(sql);
        ps.setInt(1, id_est);
        ps.setInt(2, id_cur);

        // Ejecutar la consulta
        int filasInsertadas = ps.executeUpdate();
        if (filasInsertadas > 0) {
            // Si se insertó correctamente, mostrar los registros
            response.sendRedirect(request.getContextPath() + "/registroServlet");
        } else {
            // Si falló, redirigir a una página de error
            request.setAttribute("mensaje", "Error al registrar.");
            request.getRequestDispatcher("/pages/formRegistro.jsp").forward(request, response);
        }
    } catch (SQLException e) {
        e.printStackTrace();
        request.setAttribute("mensaje", "Ocurrió un error: " + e.getMessage());
        request.getRequestDispatcher("/pages/formRegistro.jsp").forward(request, response);
    } finally {
        // Cerrar los recursos
        try {
            if (rs != null) {
                rs.close();
            }
            if (ps != null) {
                ps.close();
            }
            if (conn != null) {
                conn.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Imagen 19. Método doPost

La implementación se puede observar en la [imagen 10](#) y la funcionalidad en la imagen 12.

Métodos doDelete

La implementación se encuentra en el botón de eliminar de cada registro en sus respectivas tablas.

Estudiantes

```
@Override
protected void doDelete(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    ConnectionBD conexion = new ConnectionBD();
    System.out.println("se ejecuta doDelete");

    String id = request.getParameter("id");
    // Validate input
    if (id == null || id.trim().isEmpty()) {
        response.setStatus(HttpServletResponse.SC_BAD_REQUEST); // Invalid request
        System.out.println("invalid request");
        return;
    }

    String sql = "DELETE FROM estudiante WHERE id_estudiante like ?";

    try {
        conn = conexion.getConnectionBD();
        ps = conn.prepareStatement(sql);
        ps.setString(1, id);

        int rowsAffected = ps.executeUpdate();
        if (rowsAffected > 0) {
            response.setStatus(HttpServletResponse.SC_OK); // Eliminar exitoso
            System.out.println("se eliminó");
        } else {
            response.setStatus(HttpServletResponse.SC_NOT_FOUND); // No se encontró el usuario
            System.out.println("se ejecuta doDelete");
        }
    } catch (Exception e) {
        e.printStackTrace();
        response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR); // Error del servidor
    } finally {
        try {
            if (ps != null) {
                ps.close();
            }
            if (conn != null && !conn.isClosed()) {
                conn.close();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Imagen 20. Método doDelete

```

<script>
  function elimiarEstudiante(id) {
    console.log('eliminarEstudiante?id=' + id);
    if (confirm("¿Estás seguro de que quieres eliminar este usuario?")) {
      fetch(`${pageContext.request.contextPath}/estudianteServlet?id=` + id, {
        method: 'delete'
      }).then(response => {
        if (response.ok) {
          alert('Estudiante eliminado exitosamente');
          location.reload();
        } else {
          alert('Error al eliminar estudiante');
        }
      }).catch(error => console.error('Error:', error));
    }
  }
</script>

```

Imagen 21. Implementación en tabla_estudiantes.jsp

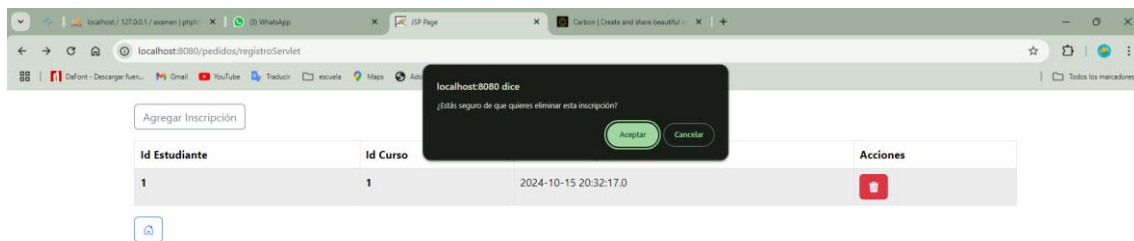


Imagen 22. Funcionalidad

Cursos

```

<script>
  function eliminarCurso(id) {
    console.log('eliminarCurso?id=' + id);
    if (confirm("¿Estás seguro de que quieres eliminar este curso?")) {
      fetch(`${pageContext.request.contextPath}/cursoServlet?id=` + id, {
        method: 'delete'
      }).then(response => {
        if (response.ok) {
          alert('Curso eliminado exitosamente');
          location.reload();
        } else {
          alert('Error al eliminar curso');
        }
      }).catch(error => console.error('Error:', error));
    }
  }
</script>

```

Imagen 23. Implementación en tabla_cursos.jsp

```

@Override
protected void doDelete(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    ConnectionBD conexion = new ConnectionBD();
    System.out.println("se ejecuta doDelete");

    String id = request.getParameter("id");
    // Validate input
    if (id == null || id.trim().isEmpty()) {
        response.setStatus(HttpServletResponse.SC_BAD_REQUEST); // Invalid request
        System.out.println("invalid request");
        return;
    }

    String sql = "DELETE FROM curso WHERE id_curso like ?";

    try {
        conn = conexion.getConnectionBD();
        ps = conn.prepareStatement(sql);
        ps.setString(1, id);

        int rowsAffected = ps.executeUpdate();
        if (rowsAffected > 0) {
            response.setStatus(HttpServletResponse.SC_OK); // Eliminar exitoso
            System.out.println("se eliminó");
        } else {
            response.setStatus(HttpServletResponse.SC_NOT_FOUND); // No se encontró el usuario
            System.out.println("se ejecuta doDelete");
        }
    } catch (Exception e) {
        e.printStackTrace();
        response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR); // Error del servidor
    } finally {
        try {
            if (ps != null) {
                ps.close();
            }
            if (conn != null && !conn.isClosed()) {
                conn.close();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

Imagen 24. Método doDelete

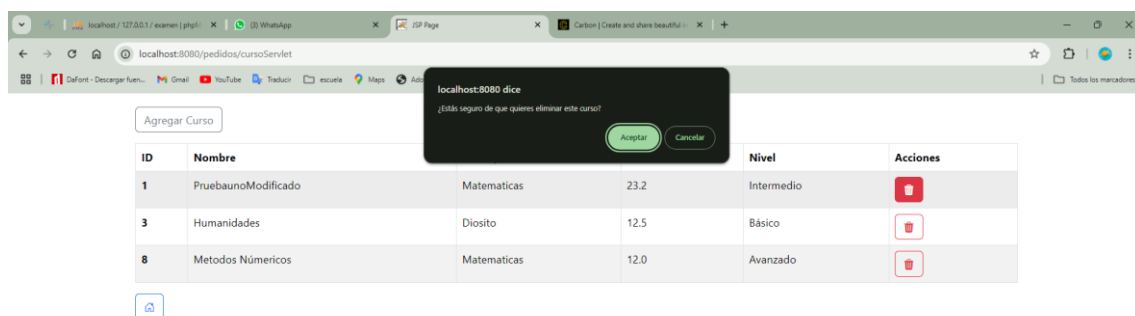


Imagen 25. Funcionalidad

Inscripciones

```
@Override
protected void doDelete(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    ConnectionBD conexion = new ConnectionBD();
    System.out.println("se ejecuta doDelete");

    String idEstudiante = request.getParameter("idEstudiante");
    String idCurso = request.getParameter("idCurso");

    String sql = "DELETE FROM inscripcion WHERE id_estudiante = ? AND id_curso = ?";

    try {
        conn = conexion.getConnectionBD();
        ps = conn.prepareStatement(sql);
        ps.setInt(1, Integer.parseInt(idEstudiante));
        ps.setInt(2, Integer.parseInt(idCurso));

        int rowsAffected = ps.executeUpdate();
        if (rowsAffected > 0) {
            response.setStatus(HttpServletResponse.SC_OK); // Eliminar exitoso
            System.out.println("se eliminó");
        } else {
            response.setStatus(HttpServletResponse.SC_NOT_FOUND); // No se encontró el usuario
            System.out.println("se ejecuta doDelete");
        }
    } catch (Exception e) {
        e.printStackTrace();
        response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR); // Error del servidor
    } finally {
        try {
            if (ps != null) {
                ps.close();
            }
            if (conn != null && !conn.isClosed()) {
                conn.close();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Imagen 26,Método doDelete

```
<script>
function eliminarIns(idEstudiante, idCurso) {
    console.log('eliminarIns?idEstudiante=' + idEstudiante + '&idCurso=' + idCurso);
    if (confirm("¿Estás seguro de que quieres eliminar esta inscripción?")) {
        fetch(`${pageContext.request.contextPath}/registroServlet?idEstudiante=`
            + idEstudiante + '&idCurso=' + idCurso, {
            method: 'delete'
        }).then(response => {
            if (response.ok) {
                alert('Inscripción eliminada exitosamente');
                location.reload();
            } else {
                alert('Error al eliminar inscripción');
            }
        }).catch(error => console.error('Error:', error));
    }
}
</script>
```

Imagen 27. Implementación en tabla_registros.jsp

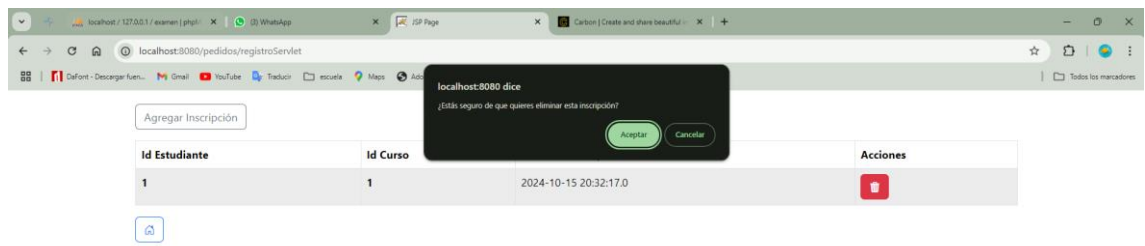


Imagen 28. Funcionalidad