

Lab 7 – Bases de Datos

Objetivos

- Al finalizar el laboratorio debes ser capaz de crear una aplicación que se conecte a una base de datos.

Preparación

- Este laboratorio asume que está instalada la versión de NetBeans completa que incluye un contenedor Web Glassfish o Apache Tomcat atendiendo (*listening*) por el puerto 8080, y la base de datos JavaDB (Derby) por el puerto 1527, y que se tiene creado un nuevo proyecto llamado **lab7**.

Ejercicio

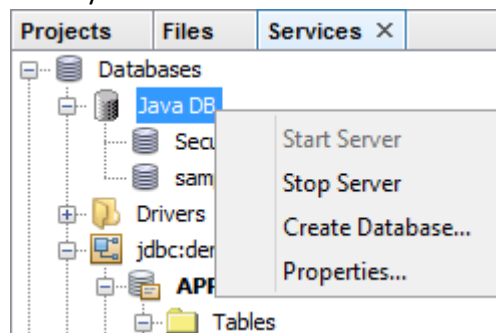
En este ejercicio realizarás lo siguiente:

- Crear una base de datos en JavaDB.
- Crear el modelo y una clase para la transferencia de datos.
- Crear tres páginas de vista.
- Crear un Servlet controlador.

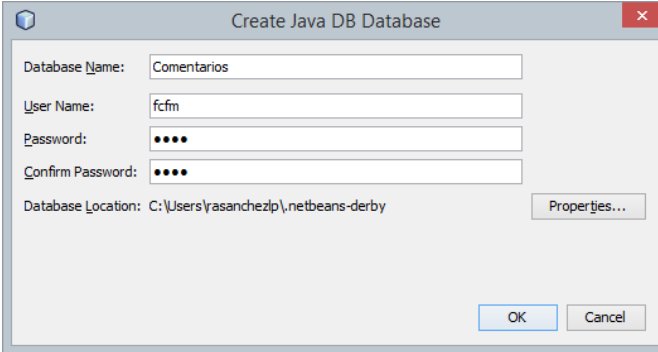
Actividad 1. Crear una base de datos en JavaDB.

Lo primero que harás en este laboratorio será crear una base de datos en JavaDB, que es el motor de base de datos (RDBMS) que incluye NetBeans. Con Java es posible conectarse a prácticamente cualquier base de datos para la que exista un driver de JDBC.

1. En la sección **Services** del navegador (lado izquierdo) expandir **Databases**. Debe mostrarse **JavaDB** que es el RDBMS incluido con NetBeans.
 - a. **NOTA [Esto NO es para el lab, solo es información adicional]:** En caso de querer conectarse a una base de datos distinta, es necesario descargar el *driver* correspondiente para la base de datos, dar clic derecho sobre **Databases** y después sobre **New Connection...**, y ahí se selecciona el archivo .jar o .zip con el *driver* correspondiente.
2. Dar clic derecho sobre **JavaDB** y en **Create Database...**



3. En el cuadro de diálogo de **Create Java DB Database**, crear una base de datos con los siguientes datos:
 - a. Database Name: **Comentarios**
 - b. User Name: **fcfm**
 - c. Password: **1sti01**

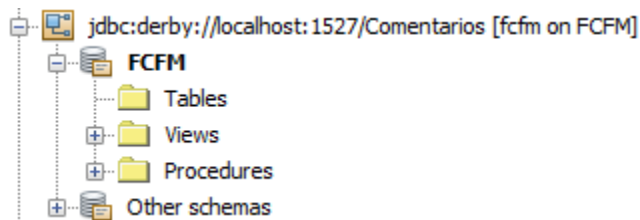


Dialog box titled "Create Java DB Database". It contains the following fields:

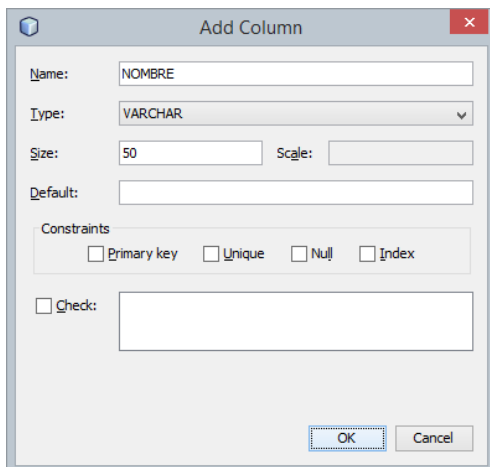
- Database Name: Comentarios
- User Name: fcfm
- Password: (masked with dots)
- Confirm Password: (masked with dots)
- Database Location: C:\Users\yasanchezlp\netbeans-derby

Buttons: Properties..., OK, Cancel.

4. En la línea de conexión, dar clic derecho en **Connect...** y expandir. Debe aparecer el nuevo esquema llamado **FCFM**.



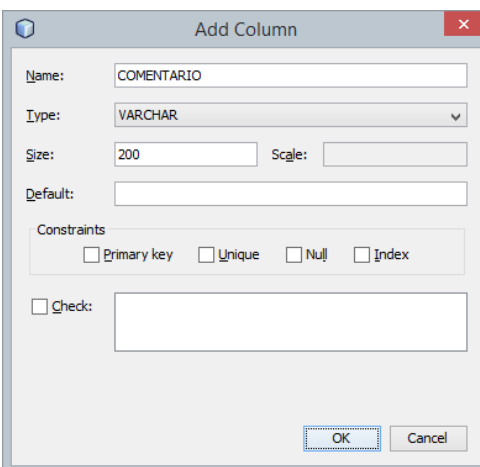
5. Dar clic derecho en **Tables** y en **Create Table**. La nueva tabla se llamará **COMENTARIOS** y contendrá dos campos (columnas):
 - a. **NOMBRE**, de tipo **VARCHAR**, tamaño **50**, no permite Null.
 - b. **COMENTARIO**, de tipo **VARCHAR**, tamaño **200**, no permite Null.



Dialog box titled "Add Column". It contains the following fields:

- Name: NOMBRE
- Type: VARCHAR
- Size: 50
- Scale: (empty)
- Default: (empty)
- Constraints:
 - ☐ Primary key
 - ☐ Unique
 - ☐ Null
 - ☐ Index
- ☐ Check: (empty)

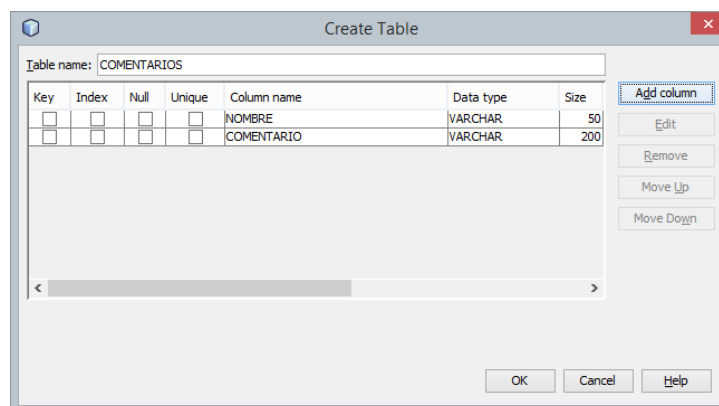
Buttons: OK, Cancel.



Dialog box titled "Add Column". It contains the following fields:

- Name: COMENTARIO
- Type: VARCHAR
- Size: 200
- Scale: (empty)
- Default: (empty)
- Constraints:
 - ☐ Primary key
 - ☐ Unique
 - ☐ Null
 - ☐ Index
- ☐ Check: (empty)

Buttons: OK, Cancel.



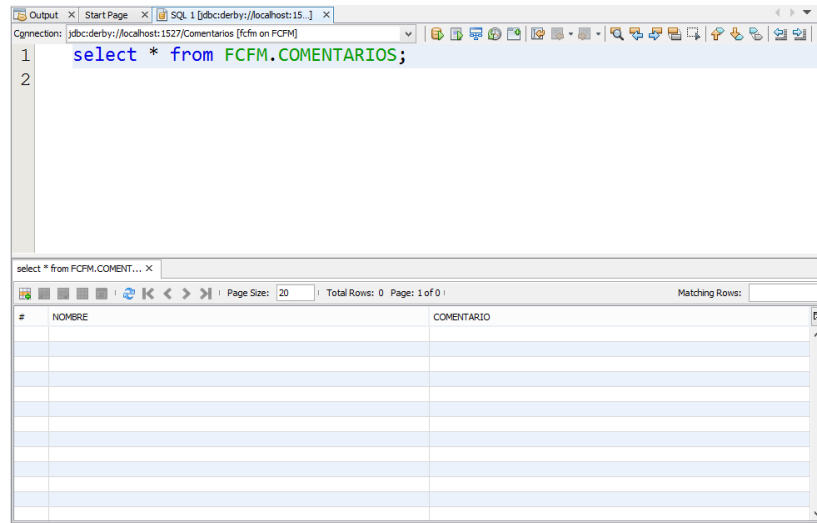
Dialog box titled "Create Table". It contains the following fields:

- Table name: COMENTARIOS

Key	Index	Null	Unique	Column name	Data type	Size
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NOMBRE	VARCHAR	50
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	COMENTARIO	VARCHAR	200

Buttons: Add column, Edit, Remove, Move Up, Move Down, OK, Cancel, Help.

6. Para ver los datos de la tabla (que al momento de crearla está vacía), puedes dar clic derecho sobre la tabla y en **View Data...** para que se muestren los registros.



En esta base de datos, específicamente en la tabla COMENTARIOS, se insertarán los datos de los comentarios que haga el usuario. Eso se hará en las secciones siguientes.

Actividad 2. Crear el modelo y una clase para la transferencia de datos.

Para interactuar con la base de datos (consultar, insertar, borrar, etc.) por lo general se hace uso del modelo. En esta ocasión crearás una clase que sirva como un objeto de acceso a datos (DAO, *Data Access Object*), y otra clase que sirva como un objeto de transferencia (*Transfer Object*). Comenzaremos con el objeto de transferencia y posteriormente con el DAO.

Transfer Object

1. En el proyecto **lab7**, crear un nuevo paquete llamado **modelo**.
2. Dentro del paquete **modelo**, crear una nueva Java class llamada **ComentariosPOJO**. Un POJO (*Plain Old Java Object*) es un objeto que sirve para representar un registro de una tabla de base de datos.
3. Agrega dos variables de instancia privadas a la clase **ComentariosPOJO**, uno por cada campo de la tabla COMENTARIOS de la base de datos:
 - a. Una variable de tipo **String** llamada **nombre** que corresponde al campo NOMBRE de tipo VARCHAR en la base de datos.
 - b. Una variable de tipo **String** llamada **comentario** que corresponde al campo NOMBRE de tipo VARCHAR en la base de datos.
4. Agrega los métodos *getters* y *setters* para cada campo. Una forma fácil de hacerlo es dando clic derecho en el código, la opción **Refactor** y seleccionando **Encapsulate Fields**. Se seleccionan todos los getters y setters necesarios (en este caso los 4) y “**Refactor**”.

Data Access Object

5. Dentro del paquete **modelo**, crear una nueva Java class llamada **ComentariosDAO**.
6. Modifica la línea que está antes de la declaración de clase donde dice **@author** y escribe tu nombre completo a continuación, seguido de tu número de matrícula, en caso de que aún no lo hayas hecho.
7. En la clase **ComentariosDAO** agrega una variable de instancia privada llamada **conexion** que sea de tipo **Connection**. El **import** que se debe agregar es **java.sql.Connection**.
8. Agrega un método privado **abrirConexion()** que no regrese nada y que:
 - a. Declare una cadena con la URI de conexión a la base de datos.
 - b. Declare cadenas con el nombre de usuario y contraseña. Esta, evidentemente, no es una práctica correcta de programación segura, pero se hará de esta manera para simplificar el laboratorio. En la vida real, las contraseñas se guardan cifradas en archivos de propiedades, o bien directamente en el servidor de aplicaciones.
 - c. Obtenga una conexión del **DriverManager** con el método **getConnection()**.
 - d. Declare que puede lanzar la excepción **SQLException**.

```
private void abrirConexion() throws SQLException {
    String dbURI = "jdbc:derby://localhost:1527/Comentarios";

    // *** ¡NO deben almacenarse datos de conexión en el código!
    String username = "fcfm";
    String password = "lsti01"; // *** ¡Esto NUNCA debe hacerse! :/
    conexion = DriverManager.getConnection(dbURI, username, password);
}
```

9. Agrega un método privado **cerrarConexion()** que no regrese nada y que invoque al método **close()** de la conexión.

```
conexion.close();
```

Ahora crearás los métodos del DAO que sirven para interactuar con la base de datos.

1. En **ComentariosDAO** agrega un método público **insertar()** que reciba como argumento un objeto de tipo **ComentariosPOJO**. El propósito de este método es insertar los datos recibidos del formulario y que pasan a través del controlador. Los **import** que se requieren son del package **java.sql**.

Lo que debe hacer este método es lo siguiente:

- a. Abrir la conexión a la base de datos (usando el método correspondiente del DAO). El código debe manejar las posibles excepciones con **try-catch**.
- b. Crear una cadena con la expresión SQL para insertar. El formato es el siguiente:

```
insert into COMENTARIOS values ('Ana', 'nunca es tarde')
```

(Lo resaltado en **rojo** es lo que se capturó en el formulario y que se agregará más adelante en el POJO que se recibe como argumento en el método **insertar()**. Esos valores se concatenarán (por el momento) dentro de la cadena SQL)

- c. Crear un Statement: **Statement stmt = conexion.createStatement();**
 - d. Ejecutar la consulta SQL: **stmt.executeUpdate(sql);**
 - e. Cerrar la conexión con el método correspondiente del DAO.
2. Agrega un método público **buscar()** que devuelva un objeto de tipo **List** (java.util) y que reciba como argumento un objeto POJO. Este objeto POJO tomará los datos que son proporcionados dentro de un formulario de búsqueda que harás más adelante. Los **import** son del package **java.sql**.

- a. Declara un objeto de tipo **ResultSet**. Este objeto almacenará los registros que coincidan con la búsqueda, en caso de haber.
- b. Declara una referencia de tipo **List** con un objeto de tipo **ArrayList()**:
List beans = new ArrayList();
- c. Abre una conexión a la base de datos. El código a partir de esta línea debe manejar las posibles excepciones con **try-catch**. La lista del paso anterior debe quedar fuera del **try-catch** para que pueda ser devuelta al final.
- d. Crear una cadena con la expresión SQL para buscar. El formato es el siguiente:

```
select * from COMENTARIOS where NOMBRE = 'Ana'
and COMENTARIO like '%tarde%'
```

(Lo resaltado en **rojo** es lo que se capturó en el formulario y que se agregará más adelante en el controlador. Ese valor se concatenará (por el momento) dentro de la cadena SQL)

- e. Crear un Statement: **Statement stmt = conexion.createStatement();**
- f. Ejecutar la consulta SQL: **mensajes = stmt.executeQuery(sql);**
(Asumiendo que mensajes es el **ResultSet**)
- g. Hacer un ciclo **while**, que se ejecute mientras en el **ResultSet** haya un siguiente registro:
while(mensajes.next()) {
 - i. Del **ResultSet** obtén la cadena que corresponde al campo "NOMBRE" y asigna a una variable de tipo **String**.

- ii. Del ResultSet obtén la cadena que corresponde al campo “COMENTARIO” y asígnala a una variable de tipo String.
- iii. Crea un nuevo objeto de tipo **ComentariosPOJO**.
- iv. Coloca el nombre que leíste del registro del ResultSet en el campo correspondiente del POJO. Haz lo mismo con el comentario.
- v. Agrega el POJO al ArrayList con el método **add()**. El método **add** recibe como argumento el objeto que agregarás a la lista.
- h. Cerrar la conexión con el método correspondiente del DAO.
- i. Regresar (**return**) el objeto **ArrayList** (que es un **List**) que contiene todos los POJOs con los registros obtenidos de la base de datos.

Actividad 3. Crear las páginas HTML/JSP para la vista.

Las páginas que harás a continuación servirán para enviar los comentarios y consultarlos.

Página para enviar comentarios.

1. Abre la página **index.html** y modifícala para que contenga un formulario para enviar los comentarios:
 - a. Un formulario HTML que envíe los datos por POST y que contacte a un servlet controlador que se llamará **ComentariosControlador**. Este servlet lo harás más adelante. Dentro del formulario debe incluirse:
 - i. Un campo de texto para enviar el nombre.
 - ii. Un área de texto para enviar los comentarios.
 - iii. Un botón de submit.
 - iv. Un campo oculto (**hidden**) de nombre “**accion**” con el valor “**comentar**”. Este campo oculto lo usará el controlador para determinar desde qué página lo están invocando.
2. Agrega todos los formatos que consideres necesarios.

Página para consultar los comentarios.

1. En Web Pages agrega una página JSP llamada **buscar.jsp**. Esta página será usada tanto para enviar la palabra que se buscará en los comentarios como para mostrar los resultados de la búsqueda. Debe contener:
 - a. Un formulario HTML que envíe los datos por POST y que contacte a un servlet controlador que se llamará **ComentariosControlador**. Este servlet lo harás más adelante. Dentro del formulario debe incluirse:
 - i. Un campo de texto para enviar el nombre.
 - ii. Un área de texto para enviar los comentarios.
 - iii. Un botón de submit.
 - iv. Un campo oculto (**hidden**) de nombre “**accion**” con el valor “**buscar**”. Este campo oculto lo usará el controlador para determinar desde qué página lo están invocando.

2. Después de agregar el formulario, agrega una tabla HTML que tenga siguiente estructura:

```
<table border="1">
  <tr>
    <th>Nombre</th>
    <th>Comentario</th>
  </tr>
  <tr>
    <td>Ana</td>
    <td>nunca es tarde</td>
  </tr>
</table>
```

La tabla solo se mostrará en caso de que haya comentarios para mostrar. El bloque resaltado, que corresponde a un renglón de comentarios, será variable y se llenará dentro de un ciclo, con un renglón para cada comentario obtenido.

3. A continuación agregarás antes de iniciar la tabla el código Java necesario para leer los comentarios que provengan de una búsqueda, en caso de que la acción haya sido la de buscar. El código deberá colocarse intercalado con el HTML, cuidando de abrir y cerrar adecuadamente las llaves de JSP: `<% %>`, y las llaves de los bloques de código Java: `{ }`.
- Si el objeto **session** no es null:
 - Obtener el atributo que llamaremos “**comentarios**” y asignarlo a un objeto de tipo **List**. Considera que el objeto se guarda en la sesión como **Object**, por lo que será necesario hacer un *casting* a **List**.
 - Si el objeto con la lista de comentarios (**List**) no es null:
 - Hacer un ciclo **for** que recorra la lista.
 - Asignar el objeto obtenido a una variable de tipo **ComentariosPOJO** (que es el tipo de objeto en el que se envían los datos desde el modelo).

El código del JSP se muestra a continuación. Es fundamental que entiendas cómo está construida la página:

```
<% if (session != null) {
    List comentarios = (List) session.getAttribute("comentarios");
    if (comentarios != null) {
%>

        <table border="1">
            <tr>
                <th>Nombre</th>
                <th>Comentario</th>
            </tr>

            <%
                for (Object o : comentarios) {
                    ComentarioBean comentario = (ComentarioBean) o;
                %>

                <tr>
                    <td><%=comentario.getNombre()%></td>
                    <td><%=comentario.getComentario()%></td>
                </tr>

            <% } // for %>
        </table>

    <% } // if comentarios not null
  } // if session not null %>
```

- b. El resultado debe ser algo similar a:


Buscar

Nombre:	<input type="text"/>
Palabra:	<input type="text"/>
<input type="button" value="Buscar"/>	
Nombre	Comentario
Ana	no merezco 0
Ana	ya me quiero ir
Ana	esto es un fraude
Ana	si merezco lo que me pusieron

4. Agrega una página JSP llamada **error.jsp** que sirva para mostrar errores.

Actividad 4. Crear el Controlador.

El controlador será muy simple. Su función será primero determinar la acción, es decir, desde qué página está siendo invocado. Esto lo harás leyendo el parámetro “**accion**” que viene en un campo oculto en las páginas Web. Dependiendo de la acción será el camino a seguir: si viene de la página “comentar”, entonces deberá invocar al modelo para insertar los comentarios en la base de datos. Por el contrario, si viene de la página “buscar”, deberá invocar al modelo para que busque los registros solicitados y los coloque en la sesión para ser desplegados posteriormente en la vista que acabas de hacer.

1. Crea un servlet llamado **ComentariosControlador**.
2. En el método **processRequest()** lee el parámetro llamado “**accion**”.
3. Si la acción es “**comentar**”:
 - a. Obtén los parámetros del nombre y comentario.
 - b. Construye un objeto del modelo.
 - c. Construye un POJO y agrégale los datos recibidos desde el formulario.
 - d. Invoca al método **insertar()** del modelo, pasándole el POJO como argumento.
 - e. Re-dirige a la página “**buscar.jsp**”
4. Si la acción es “**buscar**”:
 - a. Obtén los parámetros del nombre y comentario.
 - b. Construye un objeto del modelo.
 - c. Construye un POJO y agrégale los datos recibidos desde el formulario.
 - d. Invoca al método **buscar()** del modelo, pasándole el POJO como argumento y asignando el valor de retorno a una variable de tipo **List**.
 - e. Obtén la sesión del request.
 - f. Coloca la variable de tipo **List** (que contiene los registros leídos de la base de datos) en como un atributo de sesión llamado “**comentarios**”.
 - g. Re-dirige a la página “**buscar.jsp**”
5. Si la acción no es ninguna de las dos anteriores, redirige a la página “**error.jsp**”.
6. Ejecuta y prueba la aplicación:
 - a. Inserta comentarios simulando que son personas diferentes (nombres diferentes). Procura que contengan algunas palabras iguales entre comentarios para probar la búsqueda.
 - b. Haz búsquedas escribiendo el nombre de alguno de los “usuarios” y una palabra que posiblemente esté en los comentarios.
 - c. Verifica que los registros que se muestran correspondan a la búsqueda.
 - d. Verifica en NetBeans (en JavaDB) que estén guardados los registros. Actualiza la consulta con el botón  (ventana del **paso 6** de la **Actividad 1**).

Preguntas de Reflexión

1. ¿Cuál piensas que es el propósito de haber hecho una clase DAO en el modelo en lugar de acceder a la base de datos directamente desde el controlador?
2. ¿Para qué sirve un objeto POJO o JavaBean?
3. En caso de que los comentarios fueran muchos (digamos, cientos o miles) sería impráctico mostrarlos todos en una misma página. Generalmente los sitios de búsqueda (como Google) usan una técnica llamada “paginación”, para ir mostrando solo cierta cantidad de registros cada vez. Describe cómo harías esa paginación en esta aplicación (cuál es la lógica que seguirías en el programa).
4. Cuando se muestra la tabla con los resultados de la búsqueda, desaparecen los valores de los campos de búsqueda. ¿Qué harías para que se sigan mostrando?
5. Haz una búsqueda pero ahora, en lugar de escribir un nombre, escribe lo siguiente en el campo de búsqueda de nombre (la comilla inicial es importante, y también los dos guiones al final):

' or 1=1 --

¿Cuál fue el resultado de la búsqueda?

6. A lo que hiciste en la pregunta anterior se le conoce como *SQL Injection (SQLi)*, y es una de las vulnerabilidades más explotadas en las aplicaciones Web. De acuerdo a la cadena de búsqueda y a los resultados obtenidos, explica qué fue lo que ocurrió.
7. ¿Cómo piensas que puede evitarse un SQL injection como el de la pregunta 4? (A estas alturas del curso no se vale responder “no sé” a una pregunta así).
8. Elabora un diagrama donde muestres todos los elementos que construiste en esta práctica y cómo están relacionados entre ellos.

Especificaciones de la Entrega del Reporte de Laboratorio

1. El reporte debe incluir una **portada** con tus datos al principio y una página de **bibliografía** al final. Si no consultaste otras fuentes, al menos debe estar indicada la sesión de clase donde se explicó el tema, de acuerdo a las indicaciones de cómo citar dadas en la tarea 1.
2. En el contenido debes escribir **tu opinión** acerca de esta práctica, indicando qué **dificultades** encontraste para realizarla (si es que hubo alguna), qué cosas nuevas **aprendiste** y, por último, agrega tus respuestas a las **preguntas de reflexión** (puede ser en formato “preguntas y respuestas” o escribiendo un párrafo explicando cada punto).
3. Agrega las **5 capturas de pantalla** del navegador: una con la página inicial donde se captura el comentario, otra con la página de búsqueda sin datos, otra con la página de búsqueda con datos, otra con los resultados del SQLi y otra con la página de error. En todas las capturas debe leerse el URL completo.
4. Este reporte junto con el código fuente contará como tarea.
5. Comprime el directorio **lab7** en un archivo **.zip** (obviamente se incluirá todo su contenido).
6. El envío de la tarea debe incluir **dos** archivos:
 - a. El reporte de laboratorio en formato **PDF**.
 - b. El archivo **.zip** con el código fuente del proyecto.