5.2 FORMULARIO

EJ formulario contendrá dos campos de tipo TEXT donde el visitante introducirá su ***nombre y apellidos.*** A continuación, deberá indicar la opinión que le merece la página visitada eligiendo una entre tres posibles ***(Buena, Regular y Mala).*** Por último, se ofrece al usuario la posibilidad esscribir un ***comentario*** si así lo considera oportuno. En la figura a continuación puede observarse el diseño del formulario creado.

El código correspondiente a la ***página HTML*** que contiene este formulario es el siguiente (fichero ***MiServlet.htm):***

```
<HTML>
<HEAD>
<TITLE>Envíe su opinión</TITLE>
</HEAD>
<BODY>
<H2>por favor, envíenos su opinión acerca de este sitio web</H2>
<FORM ACTION="http://labsys.frc.utn.edu.ar:8080/servlet/servletopinion. METHDD="POST">
Nombre:        <INPUT TYPE="TEXT" NAME="nombre" SIZE=15><BR>
Apellidos:  <INPUT TYPE="TEXT" NAME="apellidos" SIZE=30><P>
Opinión que le ha merecido este sitio web<BR>
 <INPUT TYPE="RADIO" CHECKED NAME="opinion" VALUE="Buena">Buena<BR>
 <INPUT TYPE="RADIO"          NAME="opinion" VALUE "Regular">Regular<BR>
 <INPUT TYPE="RADIO"          NANE="opinion" VALUE="Mala">Mala<p>
Comentarios <BR>
<TEXTAREA NAME="comentarios" ROWS=6 COLS=40>
</TEXTAREA><P>
<INPUT TYPE="SUBMIT' NAME=botonEnviar' VALUE="Enviar">
<INPUT TYPE="RESET" NANE="botonLimpiar" VALUE="Limpiar">
</FORM>
</BODY>
</HTML>
```

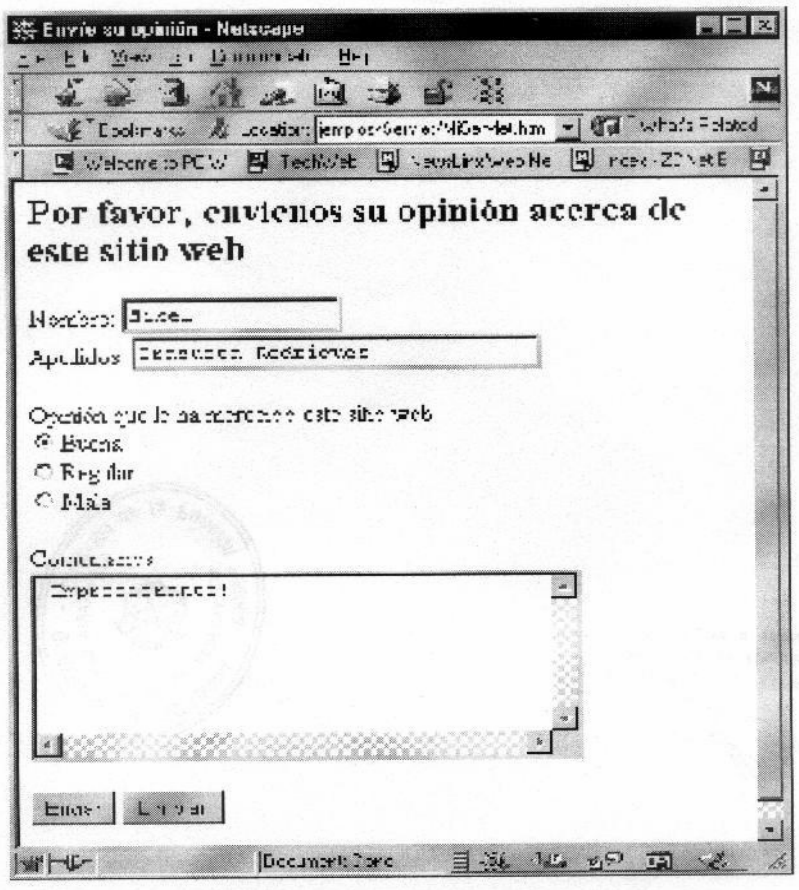


Figura 4. Diseño del formulario de adquisición de datos.

Es necesario asignar un identificador único (es decir, un valor de la propiedad NAME) a cada uno de los ***campos*** del formulario, ya que la información que reciba el ***servlet*** estará organizada en forma de ***pares de valores,*** donde uno de los elementos de dicho par será un ***String*** que contendra e ***nombre del campo.*** Así, por ejemplo, si se introdujera como nombre del visitante *"Mikel",* e ***servlet*** recibiría del browser el par ***nombre=Mikel,*** que permitirá acceder de una forma sencilla al nombre introducido mediante el método ***getParameter(),*** tal y como se explicará posteriormente al analizar el ***servlet*** del ejemplo introductorio. Por este motivo es importante no utilizar nombres duplicados en los elementos de los formularios.

Por otra parte puede observarse que en el *tag* **<FORM>** se han utilizado dos propiedades ***ACTION*** y ***METHOD.*** El método ***(METHOD)*** utilizado para la transmisión de datos es el método ***HTTP POST.*** También se podría haber utilizado el método ***HTTP GET,*** pero este método tiene algunas limitaciones en cuanto al volumen de datos transrnisible, por lo que es recomendable utilizar el método ***POST.*** Mediante la propiedad ***ACTION*** deberá especificarse el ***URL*** del *servlet* que debe procesar los datos. Este ***URL*** contiene, en el ejemplo presentado, las siguientes características:

El ***servlet*** se encuentra situado en un servidor cuyo nombre es ***miServidor*** Este nombre dependerá del ordenador que proporcione los servicios de red. La forma de saber cuál es el nombre de dicho ordenador y su dirección ***IP*** es acudiendo al ***Panel de Control (Control Panel)*** de ***Windows.*** Clickeando en ***Red (Network),*** se accede a las propiedades de la red (obviamente, si se quiere utilizar ***servlets,*** será necesario(tener instalados los drivers de red, en concreto los drivers de ***TCP/IP*** que vienen con ***Windows*** *95/98/NT).* Dentro de la lengúeta ***Protocolos (Protocols)*** se escoge ***TCP/IP*** y se clickea en ***Propiedades (Properties).*** Aparecerá un nuevo cuadro de diálogo en el que habrá que seleccionar la lengúeta *DNS (Domain Name System).* Allí se encuentra definido el nombre del ordenador ***(Host*** *Name)* así como su dominio ***(Domain*** *Name).* En cualquier caso, para poder hacer pruebas, se puede utilizar el como nombre de servidor el *host local* o *localhost.* cuyo número ***IP*** es ¿??***.?.?.?.*** Por ejemplo, se podría haber escrito (aunque estos servicios no serían accesibles desde el exterior de la Regional Córdoba):

<FORM AcTION='http: //localhost:8080/servlet/Servletopinion" METHOD="POST">

o de otra forma,

<FORM ACTION="http: //???.?.?.?:8080/servlet/Servletopinion" METHOD="POST">

El ***servidor HTTP*** está 'escuchando" por el puerto el ***puerto*** 8080. Todas las llamadas utilizando dicho puerto serán procesadas por el módulo del servidor encargado de la gestión de los *servlets.* En principio es factible la utilización de cualquier puerto libre del sistema siempre que se indique al ***servidor HTTP*** cuál va a ser el puerto utilizado para dichas llamadas. Por diversos motivos, esto último debe ser configurado por el administrador del sistema.

El *servlet* se encuentra situado en un subdirectorio (virtual) del servidor llamado *servlet.* Estc nombre es en principio opcional, aunque la mayoría de los servidores lo utilizan por defecto.

El nombre del ***servlet*** empleado es ***ServletOpinion,*** y es éste el que recibirá la información enviada por el cliente al servidor (el formulario en este caso), y quien se encargará de diseñar la respuesta, que pasará al servidor para que este a su vez la envíe de vuelta al cliente. A final se ejecutará una clase llamada ***ServletOpinion.class.***

5.3 CÓDIGO DEL SERVLET

Como un ***servlet*** es una clase de ***Java,*** deberá por tanto encontrarse almacenado en un fichero con el nombre ***ServetOpinion.java***. Para hacer lo más simple posible este ejemplo introductorio, este ***servlet*** se limitará a responder al usuario con una página ***HTML*** con la información introducida en el formulario. El código fuente de la clase ***ServletOpinion*** es el siguiente:

```
import java.io.*;
import javax.servlet.* ;
import javax.servlet.http.* ;
```

```
public class ServletOpinion extends Http5ervlet {

    //      Declaración de variables miembro correspondientes
    //      a los campos del formulario
     private String nombre=null;
     private String apellidos=null;
     private String opinion=null;
     private String comentarios=null;


        //          init() se ejecuta una única vez (al ser inicializado el servlet)
        //          Se suelen inicializar variables y realizar operaciones costosas en
     //      tiempo de ejecución (abrir ficheros, bases de datos, etc)
     public void init(ServletConfig config) throws ServletException {
     //      Llamada al método mito de la supercíase (GenericServlet)
     //      Así se asegura una correcta inicialización del servlet
     super. init (config);
     System.out.println( "Iniciando ServíetOpinion. .");
    }       // fin del método init()


        //          destroy() es llamado por el servidor web al "apagarse" (al hacer
     //  shutdown) . Sirve para proporcionar una correcta desconexión de una
     //  base de datos, cerrar ficheros abiertos, etc.
     public void destroy() {
     System.out.println("No hay nada que hacer...');
    }       // fin del método destroy()


        // doPost() llamado mediante un HTTP POST. Se invoca automáticamente
        // al ejecutar un formulario HTML
     public void doPost (HttpServletRequest req, HttpServletResponse resp)
            throvs ServletException, IOException {

            //  Adquisición de los valores del formulario a través del objeto req
            nombre =req.getparameter("nombre");
            apellidos=req.getParameter("apellidos");
            opinión =req.getparameter("opinion");
            comentarios=req.getparameter ("comentarios");

            //  Devolver al usuario una página HTML con los valores adquiridos
            devolverPaginaHTML (resp);
     }      // fin del método doPost()

    public void devolverPaginaHTML(HttpservletResponse resp){

            // En primer lugar se establece el tipo de contenido MIME de la respuesta
            resp. setContentType ( “textlhtml");

            //      Se obtiene un PrintWriter donde escribir (sólo para mandar texto)
            PrintWriter out = null;
            try {
                    out=resp.getWriter ();
           catch (IOException io) System.out.println("Se ha producido una excepcion");}

            //      Se genera el contenido de la página HTML
            out.println("<html>");
            out . printin < "<head>");
            out.println("<title>valores recogidos en el out.println("</head>");
            out . println ( "<body>");
            out.println("<b><font size=+2>Valores recogidos del “);
```

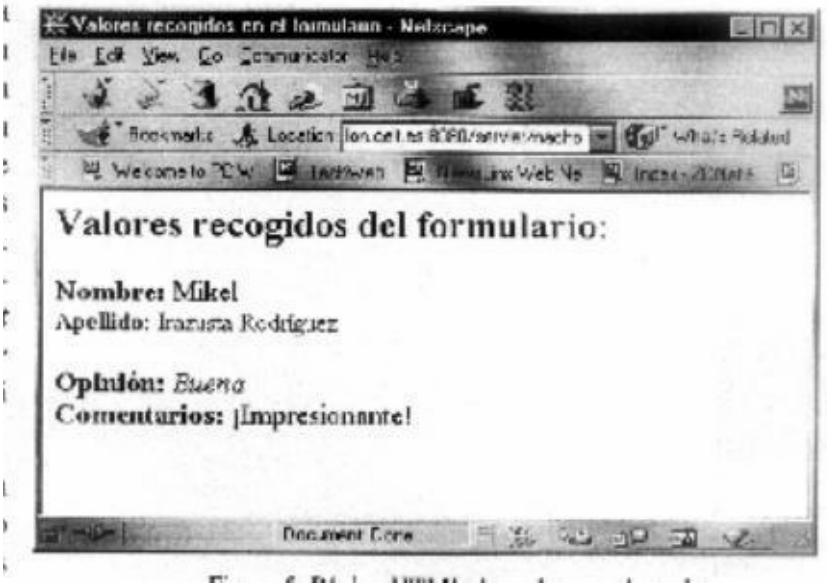
```
                out.println< "formulario: </font></b>">;
                        out.println(        "<p><font        size=+l><b>Nombre:
                </b>"+nombre+"</font>") out.println("<br><fontsize=+l><b>Apellido: <Ib>"
                        +apellidos+"</font><b><font size=+l></font></b>')
                out.println("<p><font size=+l> <b>Opini&oacute;n: </b><i>"  opinion +
                        "</i></font>")
                out.println(    "<br><font size=+1><b>Comentarios: <Ib>" + comentarios
                        +"</font>")
                out.println<"</body>");
                out.println("</html>");
                //  Se fuerza la descarga del buffer y se cierra el PrIntWriter,
                //    liberando recursos de esta forma
                out.flush();
                out close();
        }       // fin de devolverPaginaHTML()


        //      Función que permite al servidor web obtener una pequeña descripción del
        //      servlet, qué cometido tiene, nombre del autor, comentarios
        //      adicionales, etc.
        public String getServletInfo(> {
                return "Este servlet lee los datos de un formulario" + y los muestra en pantalla";
        }       // fin del método getServletInfo()
}           //      ServletOpinion
```

El resultado obtenido en el browser tras la ejecución del ***servlet*** puede apreciarse en la Figura mostrada a continuación:

En aras de una mayor simplicidad en esta primera aproximación a los ***servlets,*** se ha evitado tratar de conseguir un código más sólido, que debería realizar las comprobaciones pertinentes (verificar que los ***String*** no son ***null*** después de leer el parámetro, excepciones que se pudieran dar etc.) e informar al usuario acerca de posibles errores en caso de que fuera necesario. El alumno puede remitirse al ejemplo del cliente del práctico en red, tecnología CGI, donde si se tuvo en cuenta la consistencia.

En cualquier caso, puede observarse que el aspecto del código del ***servlet*** es muy similar al de cualquier otra clase de ***Java.*** Sin embargo, cabe destacar algunos aspectos particulares:



La clase ***ServietOpinion*** hereda de la clase ***HttpServlet,*** que a su vez hereda de ***GenericServlet.*** La forma más sencilla (y por tanto la que debería ser siempre empleada) de

crear un ***servlet,*** es heredar de la clase ***HttpServlet.*** De esta forma se está identificando la clase como un ***servlet*** que se conectará con un ***servidor HTTP***

El método **init()**es el primero en ser ejecutado. Sólo es ejecutado ***la primera*** *vez* que el ***servlet*** es llamado. Se llama al método in*itO* de la super-clase *GenericServlet* a fin de que la inicialización sea completa y correcta. La interface ***ServletConfig*** proporciona la información que necesita el ***servlet*** para inicializarse (parámetros de inicialización, etc.).

El método ***destroyO*** no tiene ninguna función en este ***servlet,*** ya que no se ha utilizado ningún recurso adicional que necesite ser cerrado, pero tiene mucha importancia silo que se busca es proporcionar una descarga correcta del ***servlet*** de la memoria, de forma que no queden recursos indebidamente alocados, o haya conflictos entre recursos en uso. Tareas propias de este método son por ejemplo el cierre de las conexiones con otros ordenadores o con bases de datos.

Como el formulario ***HTML*** utiliza el método ***HTTP POST*** para la transmisión de sus datos habrá que redefinir el método ***doPost(),*** que se encarga de procesar la respuesta y que tiene como argumentos el objeto que contiene la petición y el que contiene la respuesta (pertenecientes a las clases ***HttpServletRequest y HttpServletResponse,*** respectivamente). Este método será llamado tras la inicialización del ***servlet*** (en caso de que no haya sido previamente inicializado), y contendrá el núcleo del código del ***servlet*** (llamadas a otros ***servlets,*** métodos. etc.).

El método ***getServletInfo()*** proporciona datos acerca del ***servlet*** (autor, fecha de creación. funcionamiento, etc.) al servidor web. No es en ningún caso obligatoria su utilización aunque puede ser interesante cuando se tienen muchos ***servlets*** funcionando en un mismo servidor y puede resultar compleja la identificación de los mismos.

Por último, el método ***devolverPaginaHTML()*** es el encargado de mandar los valores recogidos del cliente. En primer lugar es necesario tener un *stream* hacia el cliente ***(PrintWriter*** cuando haya que mandar texto, ***ServletOutputStream*** para datos binarios). Posteriormente debe indicarse el tipo de contenido **MIME** de aquello que va dirigido al cliente ***(text/html*** en el caso presentado). Estos dos pasos son necesarios para poder enviar correctamente los datos al cliente. Finalmente, mediante el método ***printlnO*** se va generando la página **HTML** propiamente dicho (en forma de ***String).***

El esquema mencionado en este ejemplo se repite en la mayoría de los ***servlets*** y es el fundamento de este tecnología.

6 EL SERVLET API2.O

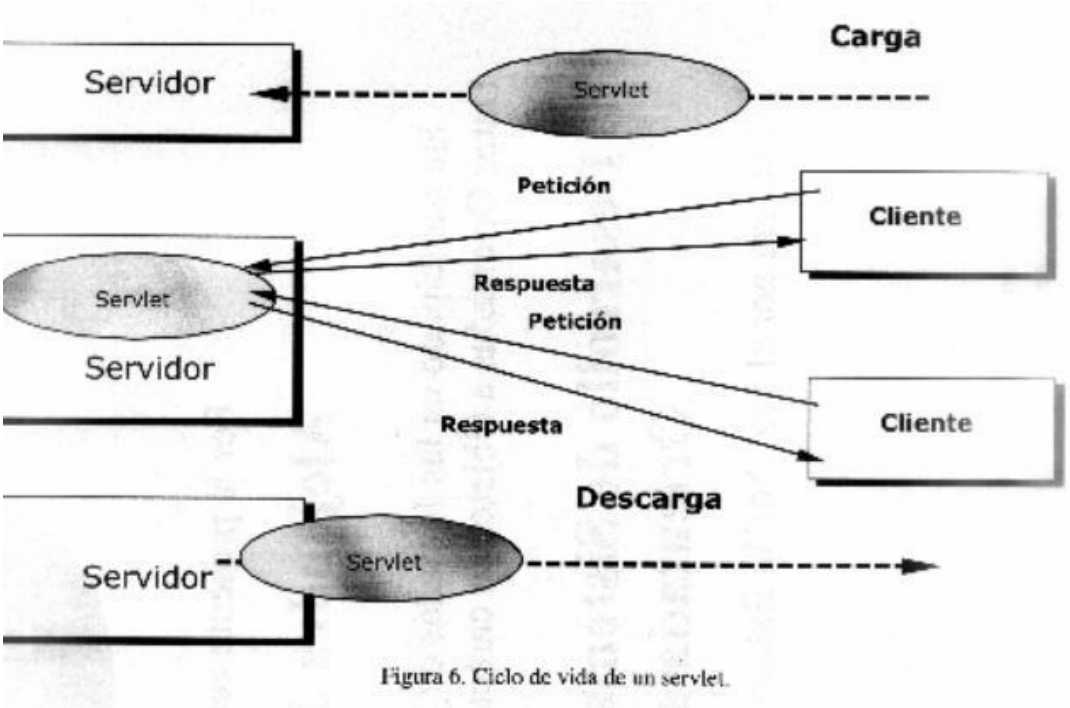


Figura 6. Ciclo de vida de un servlet.

El Java Servlet API 2.0 es una extensión al API de Java 1.1.x, y también de Java 2. Contiene los packagesj avax.servlet y javax.servlet.http. El API proporciona soporte en cuatro áreas:

1. Control del ciclo de vida de un ***servlet:*** clase ***GenericServlet***

2. Acceso al contexto del ***servlet (servlet context)***

3. Clases de utilidades

4. Clases de soporte específicas para ***HTTP:*** clase ***HttpServlet***

6.1 EL CICLO DE VIDA DE UN SERVLET: CLASE GENERICSERVLET

Como se ha dicho en el punto 4.1, la clase ***GenericServlet*** es una clase abstracta porque declara el método ***service()*** como ***abstract.*** Aunque los ***servlets*** desarrollados en conexión con páginas web suelen derivar de la clase ***HttpServlet,*** puede ser útil estudiar el ciclo de vida de un ***servlet*** en relación con los métodos de la clase ***GenericServlet.***

Los ***servlets*** se ejecutan en el ***servidor HTTP*** como parte integrante del propio proceso dell servidor. Por este motivo, el ***servidor HTTP*** es el responsable de la inicialización, llamada y destrucción de cada objeto de un ***servlet,*** tal y como puede observarse en la Figura 6.

Un ***servidor web*** se comunica con un ***servlet*** mediante los métodos de]a initerface ***javax.servlet.Servlet.***
Esta interfase está constituida básicamente por tres métodos principales

- **init(), destroy() y service()** y dos auxiliares:
- ***getServletconfigO, getServletInfo()***

6.1.1 El método INIT() en la clase GenericServlet

Cuando un ***servlet*** es cargado por primera vez, el método ***initO*** es llamado por el ***servidor HTT,*** por única vez. En este método se ejecutan operaciones de inicialización potencialmente costosas. en Esto es una ventaja importante frente a los ***programas CGI,*** que son cargados en memoria cada vez que hay una petición por parte del cliente. Por ejemplo si en un día hay 500 consultas a una base de datos, mediante un ***CGI*** habría que conectarse con la base de datos 500 veces, frente a una única conexión si utilizamos tecnología ***servlet.***
Si el servidor permite pre-cargar los ***servlets,*** el método *init()* será llamado al iniciarse el servidor. Si el servidor no tiene esa posibilidad, será llamado la primera vez que haya una petición por parte de un cliente.

El método ***init()*** tiene un único argumento, que es una referencia a un objeto de la interfase ***ServletConfig,*** que proporciona los argumentos de inicialización del ***servlet.*** Este objeto dispone del método ***getServletContext()*** que devuelve una referencia de la interface ***ServletContext,*** con información acerca del entorno en el que se está ejecutando el ***servlet.***

Siempre que se redefina el método ***init()*** de la clase base ***GenericServlet*** (o de ***HttpServlet),*** será preciso llamar al método ***init()*** de la super-clase, a fin garantizar que la inicialización se efectúe correctamente. Por ejemplo:

```
public void init(ServletConfig config) throws ServletException {

super.init(config); // Llamada al método init() de la supercíase
        System.out.println("Iniciando ...");
    // Definición de variables
    //      Apertura de conexiones, ficheros, etc.
}
```

El método ***init()*** termina su ejecución antes de que sea llamado otro método del ***servlet.***