PROYECTO DE LÓGICA Y MATEMÁTICA PARA SIMULACIÓN

GENERACIÓN DE UNA PÁGINA WEB A PARTIR DE UN CÓDIGO CON HTML TERMERIZADO EN PROLOG

Por

Ing. Ixhel Mejías. C.I. 18.924.408
Correo electrónico: ixalejandra@gmail.com

Prof. Jacinto Dávila

Marzo 2015

Índice

Agradecimientos

A la señora Annie Ogborn y al profesor Jacinto Dávila, por sus sugerencias, receptividad y excelente atención.

1 Introducción

Existe poca disponibilidad de información acerca del desarrollo de aplicaciones web en Prolog, sobre todo en español. Por tal motivo, se creó un manual que pretender servir de guía para aquellas personas que desean aprender cómo pueden generar una página web a partir de un código con HTML termerizado en Prolog. Está dirigido a personas con conocimientos básicos de SWI-Prolog, HTML, CSS y JavaScript.

Este informe describe el procedimiento para desarrollar una página web a partir de un código con HTML termerizado en Prolog, como un aporte al proyecto ULAnix Mathematica: "Una plataforma educativa abierta y libre para aprender matemáticas" (Dávila, 2014).

SWI-Prolog se puede descargar gratuitamente a través del siguiente enlace http://www.swi-prolog.org/Download.html.

1.1 Objetivo

Generar una página web a partir de un código con HTML termerizado en Prolog.

1.2 Requisitos de la página web

- La página web a desarrollar corresponde a una parte del proyecto ULAnix Mathematica, referida al tema de *Lógica*. Debe tener una sección donde se puedan hacer consultas de preguntas y respuestas y otra sección donde se desplieguen todas las preguntas del tema.
- Se debe incorporar JavaScript para agregar dinamismo a la página.
- Se deben utilizar hojas de estilo CSS, invocando las .css desde un archivo HTML.

2 Estructura básica de una página web sencilla generada a partir de un código con HTML termerizado en Prolog

Para lograr el objetivo planteado, se estableció como primer paso realizar una revisión bibliográfica de los principales conceptos involucrados. A través de esta revisión, se pudo obtener información referente a la creación de una página web básica y simple a partir de un código con HTML termerizado en Prolog, la cual se presenta de manera resumida a continuación. En la figura 1 se muestra el código para tales fines, implementado en SWI-Prolog. Además, en la figura 2 se puede observar la página web generada.

hello_html_term.pl

```
:- use_module(library(http/thread_httpd)).
:- use_module(library(http/http_dispatch)).
:- use_module(library(http/html_write)).

:- http_handler('/', say_hi, []).

server(Port) :-
        http_server(http_dispatch, [port(Port)]).

say_hi(Request) :-
        reply_html_page(
           [title('Mi Página Web')],
            [\page_content(Request)]).

page_content(_Request) -->
        html(
            [
            h1('Una Página Web Simple Generada a partir de un Código con HTML Termerizado en Prolog'),
            p(['El ', b(primer), ' párrafo']),
            p(i('Ejemplo en itálicas sin listas')),
            p([style='font-size: 36pt', title='tooltip text'], 'Texto adicional'),
            '<b>esto no saldrá en negrita</b>',
            'Esto tiene ~w argumentos que terminan en ~w' - [2, pruebas],
            \['<i> En itálica</i>', '<b> ahora si en negrita </b>'],
            \['<i>~w</i>' - ['De nuevo en itálica']],
            \some_included_stuff,
            \more_included_stuff(';Ingrese texto a incluir!')
    ]).

some_included_stuff --> html([p('Algunas cosas incluidas...')]).

more_included_stuff(X) --> html([p(['Más cosas incluidas: ', b(X)])]).

?- server(5000).
```

Figura 1. Código para generar una página web a partir de un código con HTML termerizado en Prolog.

Fuente: Elaboración propia, basándose en el código presentado por Ogborn (2013).

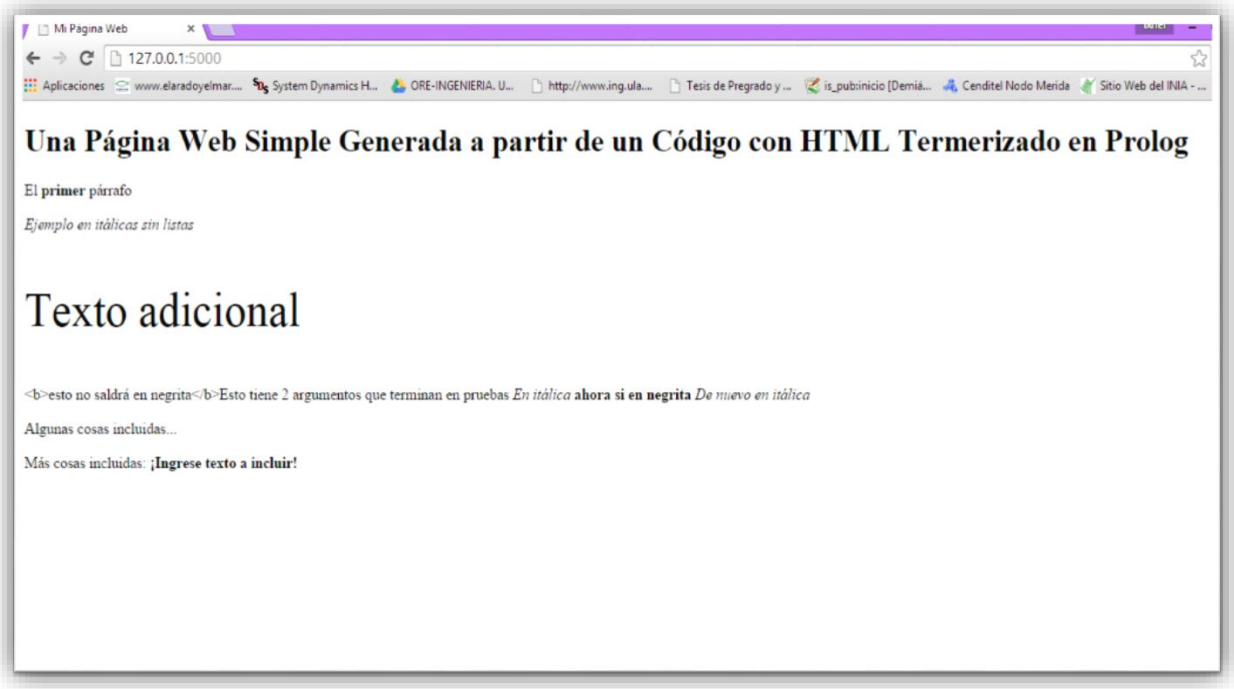

Figura 2. Página web generada a partir de un código con HTML termerizado en Prolog.

Fuente: Elaboración propia.

2.1 Servidor Web

Existen diversas maneras de ejecutar una aplicación web, una de ellas es simplemente corriendo un servidor web propio. De esta manera, una vez compilado el programa, la página generada se puede observar a través de la dirección http://127.0.0.1:5000/. Específicamente, se utilizó el número de puerto 5000; sin embargo, este número puede variar desde 1001 hasta 65000, dependiendo del sistema operativo usado (SWI-Prolog, 2015).

La línea de código mostrada a continuación hace una solicitud (*query*) para arrancar el servidor en el puerto 5000.

```
?- server (5000).
```

Las siguientes líneas incluyen módulos necesarios para levantar y correr el servidor básico.

```
:- use_module(library(http/thread_httpd)).
:- use_module(library(http/http_dispatch)).
```

Estas líneas representan el enlace principal al servidor**:**

```
server(Port) :-
    http_server(http_dispatch, [port(Port)]).
```

2.2 Definición de manejadores (*handlers*)

La siguiente línea de código es un *handler* que maneja la ruta del root:

```
:- http_handler('/', say_hi, []).
```

El primer argumento (/) es un átomo que se refiere a la ruta de la URI (*Uniform Resource Identifier*). El segundo argumento representa la meta para realizar un *query* y será llamado mediante **say_hi(Request)**. El último argumento es un conjunto de opciones. También se pueden utilizar rutas abstractas; por ejemplo, se reemplaza la línea anterior por la que se muestra a continuación:

```
:- http_handler(root(hello_html_term), say_hi, []).
```

Luego, se puede visualizar la página web a través de la siguiente dirección: **http://127.0.0.1:5000/hello_html_term.**

2.3 Generando HTML

Las líneas de código para definir **say_hi(Request)** son:

```
say_hi(Request) :-
        reply_html_page(
          [title('Mi Página Web')],
           [\page_content(Request)]).
```

El uso de **reply_html_page** permite la visualización de la página web, ya que **reply_html_page** maneja las etiquetas *headers*, *head* y *body*, así como el tipo de documento y otros elementos HTML.

Es importante señalar que se debe agregar la siguiente línea de código, con la finalidad de tener acceso al módulo necesario para traducir los términos de Prolog a HTML:

```
:- use_module(library(http/html_write)).
```

2.4 HTML Termerizado

HTML termerizado se refiere a un *Domain Specific Language* (DSL) que define el código HTML que reconoce. El HTML termerizado se incorpora al código como un argumento de "html", que representa una biblioteca *Definite Clause Grammar* (DCG). En el código presentado como ejemplo de estudio, el HTML termerizado se destaca con el color rojo.

```
page_content(_Request) -->
        html(
          [
          h1('Una Página Web Simple Generada a partir de un Código con HTML Termerizado en Prolog'),
          p(['El ', b(primer), ' párrafo']),
          p(i('Ejemplo en itálicas sin listas')),
          p([style='font-size: 36pt', title='tooltip text'], 'Texto adicional'),
          '<b>esto no saldrá en negrita</b>',
          'Esto tiene ~w argumentos que terminan en ~w' - [2, pruebas],
          \['<i> En itálica</i>', '<b> ahora si en negrita </b>'],
          \['<i>~w</i>' - ['De nuevo en itálica']],
          \some_included_stuff,
          \more_included_stuff('¡Ingrese texto a incluir!')
    ]).
```

Por otra parte, las líneas de código restantes corresponden a dos ejemplos de cómo hacer "inclusiones" en el código. Las inclusiones tienen que ver con el mecanismo que utiliza SWI-Prolog para encapsulamiento de código HTML.

```
some_included_stuff --> html([p('Algunas cosas incluidas...')]).
more_included_stuff(X) --> html([p(['Más cosas incluidas: ', b(X)])]).
```

2.5 Notas adicionales acerca del HTML termerizado

Se puede generar HTML directamente (no termerizado) a través de bloques de código que contienen etiquetas HTML dentro de **\[' ']**. Por ejemplo, para escribir en itálicas, se puede utilizar la siguiente línea de código HTML no termerizado:

```
\['<i> En itálica</i>'],
```

¿Es mejor utilizar siempre HTML termerizado?

Según lo expresado por Ogborn (2015, comunicación personal), no siempre es mejor utilizar HTML termerizado. El problema con el estilo "plantilla" de la generación de HTML es que las páginas web modernas a menudo pueden terminar en plantillas absurdamente complicadas y, efectivamente, se está programando en el lenguaje de plantillas. Así que, HTML termerizado es mejor cuando se tiene una gran cantidad de contenido generado dinámicamente, pero no tiene sentido convertir todo el código "boilerplate" (repetitivo o estandarizado) en HTML termerizado.

3 Incorporación de hojas de estilo CSS y JavaScript a una página web generada a partir de un código HTML termerizado en Prolog

A la página web presentada en la Sección 2, se le incorporó JavaScript para agregarle dinamismo y hojas de estilo CSS, invocando las .css desde el archivo HTML. Las hojas de estilo se añadieron basándose en el código expuesto en uno de los tutoriales de la página oficial de SWI-Prolog (2015).

En la figura 3 se puede observar la página web generada a través del código implementado que se muestra en la figura 4.

Figura 3. Página web generada a partir de un código con HTML termerizado en Prolog, utilizando JavaScript y hojas de estilo CSS.

Fuente: Elaboración propia.

```
:- use_module(library(http/thread_httpd)).
:- use_module(library(http/http_dispatch)).
:- use_module(library(http/html_write)).
:- use_module(library(http/html_head)).

:- http_handler(css('ptable1.css'), http_reply_file('ptable1.css', []), []).

http:location(css, root(css), []).


:- http_handler('/', say_hi, []).


server(Port) :-
        http_server(http_dispatch, [port(Port)]).

say_hi(Request) :-
        reply_html_page(
           [title('Mi Página Web')],
            [\page_content(Request)]).


page_content(_Request) -->
        html(
           [\html_requires(css('ptable1.css')),
            h1('Una Página Web Simple Generada a partir de un Código con HTML Termerizado en Prolog'),
            p(['El ', b(primer), ' párrafo']),
            p(i('Ejemplo en itálicas sin listas')),
            p(
            [style='font-size: 36pt', title='tooltip text'], 'Texto adicional'),
            \['<body bgcolor="ABEBEC"></body>'],
            '<b>esto no saldrá en negrita</b>',
            'Esto tiene ~w argumentos que terminan en ~w' - [2, pruebas],
            \['<i> En itálica</i>', '<b> ahora si en negrita </b>'],
            \['<i>~w</i>' - ['De nuevo en itálica']],
            \some_included_stuff,
            \more_included_stuff('¡La tabla representa un ejemplo de cómo incluir CSS!'),
            \['<br></br>'],
            table(class(properties),[ tr([ th('Nombre'), th('Apellido'), th('Cédula')]),tr([ td('Ixhel'), td('Mejías'), td('18.924.40
8')])]),
            \['<br></br>'],
            \['<form><input type="button" name="boton" value="¡Presione aquí!" onClick="holaMundo()"></form>'],
            \['<script language="javascript" type="text/javascript"> function holaMundo(){alert("¡Hola Mundo!")}</script>'],
            \['<br></br>'],
            \['<p align="right"><script>document.write(Date());</script></p>'],
            \['<script language="javascript" type="text/javascript"> alert("¡Bienvenidos a mi página web con JavaScript incluido!")</
script>']
     ]).

some_included_stuff --> html([p('Algunas cosas incluidas...')]).

more_included_stuff(X) --> html([p(['Más cosas incluidas: ', b(X)])]).

?- server(5000).
```

Figura 4. Código para generar una página web a partir de un código con HTML termerizado en Prolog, utilizando JavaScript y hojas de estilo CSS.

Fuente: Elaboración propia, basándose en el código presentado por Ogborn (2013) y SWI-Prolog (2015).

En comparación con la página web presentada en la Sección 2, esta muestra ciertos elementos adicionales:

- Un fondo de color azul.
- Una ventana con un mensaje de bienvenida, generada mediante el uso de JavaScript.
- Una tabla que representa un ejemplo de cómo incluir hojas de estilo CSS.
- Un botón que, al ser presionado, muestra una ventana con otro mensaje para el usuario.
- El día, la fecha y la hora estándar de Venezuela, utilizando JavaScript.

El color de fondo se agregó en el código HTML termerizado a través de la siguiente línea:

\['<body bgcolor="ABEBEC"></body>'],

A continuación se explicarán algunos detalles que deben añadirse al código HTML para poder visualizar los demás elementos descritos.

3.1 Incorporación de CSS

Para que la tabla tenga ese estilo, se creó un archivo .css llamado *ptable1.css*, donde se presenta la información correspondiente a las propiedades de las tablas en general, por ejemplo: colores, márgenes, propiedades de los bordes, etc. En la figura 5 se puede observar el código implementado para tales fines.

```
table.properties
{ margin-left: 5mm;
  border-width: 1px;
  border-style: solid;
  border-collapse: collapse;
}

table.properties td
{
  border-width: 1px;
  border-style: solid;
  padding-left: 5px;
  padding-right: 5px;
  vertical-align: top;
}

table.properties th
{ background-color: green;
  color: white;
}
```

Figura 5. Código implementado en el archivo *ptable1.css*, para definir las propiedades de las tablas.

Fuente: Elaboración propia, basándose en el código presentado por SWI-Prolog (2015).

Además, en el archivo principal se agregaron las siguientes líneas de código:

- Se incluyó el módulo necesario para la declaración abstracta de los recursos CSS y JavaScript disponibles y sus dependencias.

  :- use_module(library(http/html_head)).

- Se definió un *handler* o manejador y la localización de la hoja de estilo *ptable1.css*. Los manejadores para las hojas de estilo utilizan http_reply_file/3 como biblioteca-predicado para acceder a un archivo estático.

  :- http_handler(css('ptable1.css'), http_reply_file('ptable1.css', []), []).
  http:location(css, root(css), []).

- Donde se incluye el código HTML termerizado, se incorporó una línea de código que le indica a la infraestructura HTML que la página requiere el recurso *css('ptable1.css')*.

  \html_requires(css('ptable1.css')),

- Finalmente, al definir la tabla se incluyó una clase-atributo HTML para el archivo CSS.

  table(class(properties),[tr([th('Nombre'), th('Apellido'), th('Cédula')]),tr([td('Ixhel'), td('Mejías'), td('18.924.408')])]),

3.2 Incorporación de JavaScript

- Se agregó un botón que, al ser presionado, ejecuta una función *holaMundo()* a través de JavaScript.

  \['<form><input type="button" name="boton" value="¡Presione aquí!" onClick="holaMundo()"></form>'],
  \['<script language="javascript" type="text/javascript"> function holaMundo(){alert("¡Hola Mundo!")}</script>'],

- Mediante el uso de JavaScript, se muestra el día, la fecha y la hora.

  \[' <script>document.write(Date());</script> '],

- Finalmente se añadió una ventana con un mensaje de bienvenida.

  \['<script language="javascript" type="text/javascript"> alert("¡Bienvenidos a mi página web con JavaScript incluido!")</script>']

3.3 Notas adicionales acerca de la incorporación de JavaScript

En el ejemplo presentado anteriormente, se incorpora JavaScript dentro del mismo código; sin embargo, para tales fines también se pueden utilizar archivos .js separados del archivo principal. Estos se incluyeron de manera similar a las hojas de estilo CSS. Se definen los recursos a utilizar y la localización de los archivos. Además, se incorpora una línea de código que le indica a la infraestructura HTML que la página requiere el recurso. En vez de *css*, se utiliza *js_script*. En la Sección 4 se presentará un ejemplo de incorporación de JavaScript para su mejor compresión.

4 Diseño de una página web para el proyecto ULAnix Mathematica

Se hizo una revisión del código elaborado por un grupo de estudiantes de matemáticas de pregrado, en el marco del desarrollo de la página web del proyecto ULAnix Mathematica. Este código se encuentra disponible en http://nux.ula.ve/mathematica/mat_discretas.zip. Se seleccionó una de las páginas, específicamente la página /mat_discretas/temas/logica.html, para ser generada desde Prolog. El objetivo fue incluir los enlaces a JavaScript y a las hojas de estilo CSS, así como los campos de texto y botones que se pueden observar en la página previamente diseñada. En esta sección se presenta la página del tema de Lógica, creada a través de un código con HTML termerizado en Prolog.

La página web desarrollada presenta un diseño similar al propuesto e implementado por Santiago y Sánchez, estudiantes de matemáticas de pregrado. Se le hicieron algunas modificaciones al diseño de la misma con la finalidad de mejorar su apariencia. Dicha página contiene una sección de búsqueda donde se presentan las preguntas y respuestas del tema de Lógica y otra sección donde se muestran todas las preguntas del mismo tema. También contiene un menú desplegable con enlaces a la página principal, así como a todos los demás temas de matemáticas, igualmente a las páginas de ULAnix y SERBIULA. Además, contiene un botón que al hacer *click*, se envían los datos del formulario para ser procesados y los resultados se contemplan en una nueva página abierta.

En la figura 6 se puede observar el diseño propuesto por los estudiantes de pregrado para la página web.

En la figura 7 se muestra la página web generada a partir de un código con HTML termerizado en Prolog, utilizando el código implementado por Santiago y Sánchez (2014). Se incorporaron los archivos *bootstrap.css*, *main.css, bootstrap.js* y *jquery.min.js*.

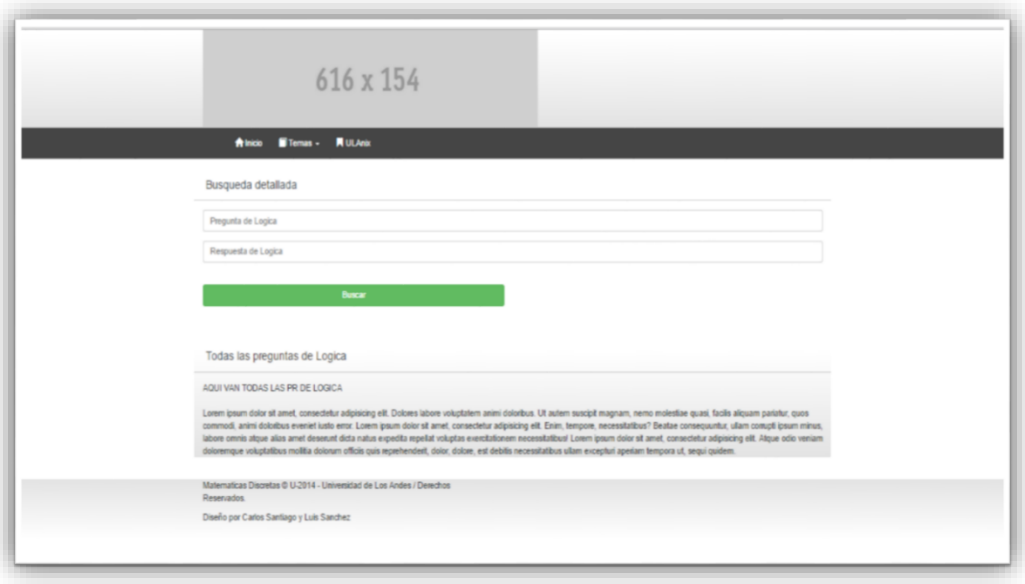

Figura 6. Diseño propuesto para la página web del tema Lógica.

Fuente: Elaboración propia, basándose en la página elaborada por Santiago y Sánchez (2014).

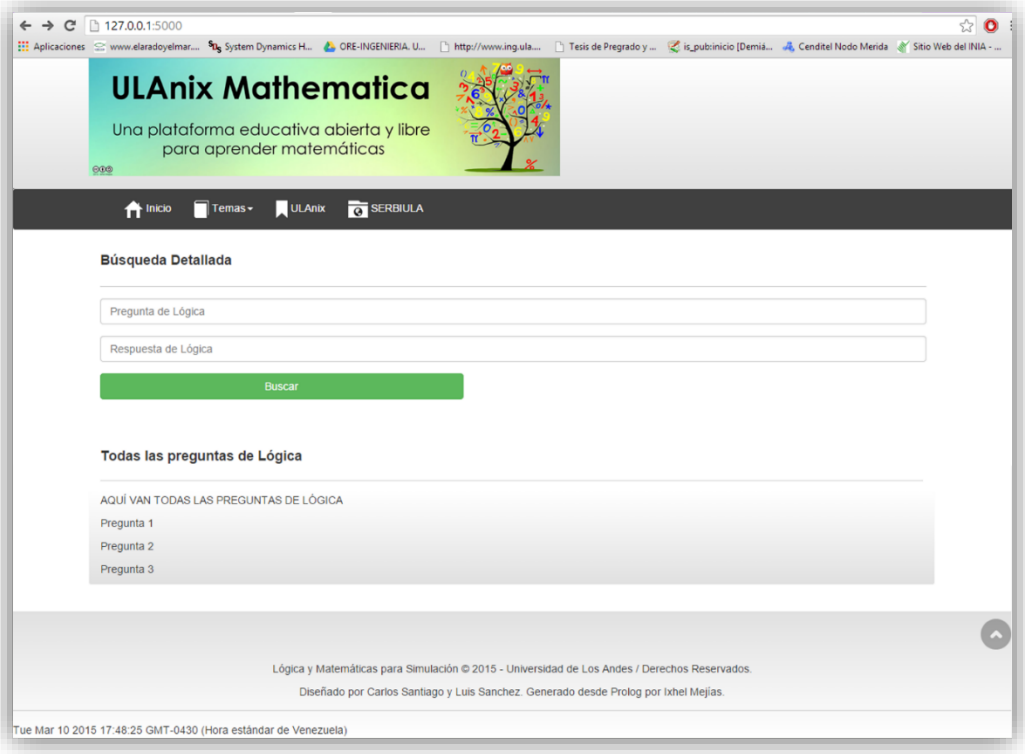

Figura 7. Página web del tema Lógica generada a partir de un código con HTML termerizado en Prolog.

Fuente: Elaboración propia, basándose en la página elaborada por Santiago y Sánchez (2014).

En el archivo *logica.pl* se puede observar el código fuente utilizado para generar la página mostrada en la figura 7. A continuación se explicará de manera general dicho código.

Se arranca el servidor en el puerto 5000. Además, se debe cargar el archivo *mathematica.pl*, para realizar la consulta de las preguntas y respuestas. También se debe incluir al código un módulo de la biblioteca *HTTP client* para la consulta ya que permite el uso de *http_read_data*:

```
:- use_module(library(http/http_client)).
```

Se incluyen todos los manejadores y los módulos definidos y explicados en el ejemplo presentado en la Sección 2. Además, se incluyen las siguientes líneas de código, con la finalidad de incorporar las hojas de estilo CSS y JavaScript.

```
:- http_handler(css('bootstrap.css'), http_reply_file('bootstrap.css', []), []).
:- http_handler(css('main.css'), http_reply_file('main.css', []), []).

:-html_resource(jquery,[virtual(true),
requires('https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js')]).
:-html_resource(bootstrap_js, [virtual(true), ordered(true), requires([jquery,
'https://maxcdn.bootstrapcdn.com/bootstrap/3.3.2/js/bootstrap.min.js'])]).
:- html_resource(js_script('bootstrap.js'), [requires(js_script('jquery.min.js'))]).

http:location(css, root(css), []).
http:location(js_script, root(js_script), []).
```

En estas líneas de código se muestran dos formas de incluir los archivos CSS y JavaScript, la primera es accediendo a los archivos almacenados en la carpeta Prolog (ver incorporación de CSS) y la segunda es mediante los enlaces a las páginas que los contienen (ver incorporación de JavaScript).

Es importante señalar que bootstrap depende de jquery, entonces jquery debe ser cargado antes. Para forzar esto, se crearon recursos virtuales y se forzó el orden mediante **ordered(true)**.

Luego, se generó el HTML a través de las siguientes líneas de código:

```
say_hi(Request) :-
        reply_html_page(
          [title('Lógica')],
           [\page_content(Request)]).
```

```
page_content(_Request) -->
        html(
          [\html_requires(css('bootstrap.css')),
           \html_requires(css('main.css')),
           \html_requires(jquery),
           \html_requires(bootstrap_js),
           \head,
           \enlaces,
           \busqueda,
           \foot
   ]).
```

Al código presentado se le añadieron varias líneas que le indican a la infraestructura HTML que la página requiere los recursos CSS y JavaScript. Por otra parte, se hicieron cuatro inclusiones que representan el encapsulamiento del código HTML para la generación de las distintas partes que conforman la página web.

Mediante **\head** se incluye el siguiente código:

```
head --> html (header ( [div (class = 'img-responsive container row col-xs-12 col-sm-12 col-md-12 col-lg-12', img ( [src = 'http://s14.postimg.org/dossnbh4h/ULAnix.jpg', class = 'img-responsive', style = 'float: left; margin: 0px 0px 0px 100px;'])])]).
```

De esta manera, se incluye una cabecera a la página, con ciertas propiedades que están especificadas en las hojas de estilo. Además se incluye un contenedor con la imagen mostrada en la figura 8, a través de un enlace. La imagen fue montada en un servidor de almacenamiento de imágenes mediante la página web http://postimage.org/.

Figura 8. Diseño de la imagen en la cabecera de la página web.

Fuente: Elaboración propia.

Al agregar **col-xs-12 col-sm-12 col-md-12 col-lg-12**, se define el tamaño del elemento que en este caso es un contenedor con una imagen. El valor representa la cantidad de columnas que ha de ocupar el elemento, mientras que xs, sm, md y lg se refieren al tipo de dispositivo donde se va a visualizar la página, es decir, extra pequeños (teléfonos), pequeños (tablas), medianos y grandes (computadoras), respectivamente (Bootstrap, 2015).

Para obtener mayor información acerca de las propiedades y estilos de los distintos elementos utilizados en esta página, se recomienda revisar el tutorial de Bootstrap disponible en http://www.tutorialspoint.com/bootstrap/.

Mediante **\enlaces** se incluye el código que genera la barra de navegación con el menú desplegable que contiene los enlaces a los otros temas de matemáticas y a otras páginas web. Esta parte se implementó con código no termerizado dentro de un formulario. Se utiliza un formulario para crear "enlaces" y poder acceder a los otros temas. No se puede simplemente utilizar hipervínculos ya que todas las páginas serán generadas desde Prolog y se puede acceder a ellas solamente mediante sus *handlers*. Las imágenes son agregadas de manera análoga a la incorporación de la imagen mostrada en la figura 8.

A través de **\busqueda** se incluye el código que crea los campos de texto, el botón "Buscar" y el contenedor donde se podrán agregar dinámicamente todas las preguntas de Lógica. Esta parte es totalmente termerizada.

Finalmente, con **\foot** se incluye el código para mostrar el contenido del pie de la página. Contiene un botón para ir a la parte superior de la página, los datos de los diseñadores y desarrolladores de la página e información acerca del día, fecha y hora estándar de Venezuela, generada mediante JavaScript.

Es importante señalar que se pueden hacer consultas debido al uso del código contenido en *servicio.pl* del proyecto ULAnix Mathematica, el cual consiste en un programa para levantar un servicio web.

5 Enlaces entre las páginas web del proyecto ULAnix Mathematica

La página *logica.pl* es completamente funcional en el sentido que permite hacer consultas; sin embargo, no brinda acceso a la página principal ni a las páginas de los otros temas, ya que estas no han sido implementadas. Con la finalidad de poner en funcionamiento algunos enlaces, se creó una página denominada *index.pl*, cuyo código permite la generación de la página principal (una versión muy simple), la página del tema de Lógica (expuesta en la Sección 4) y la página donde se despliegan los resultados de la consulta. Se logró enlazar estas tres páginas. A continuación se muestra su funcionamiento.

En la figura 9 se puede observar la página principal. Al hacer *click* en el botón "Lógica", se redirecciona a la página de Lógica, mostrada en la figura 11.

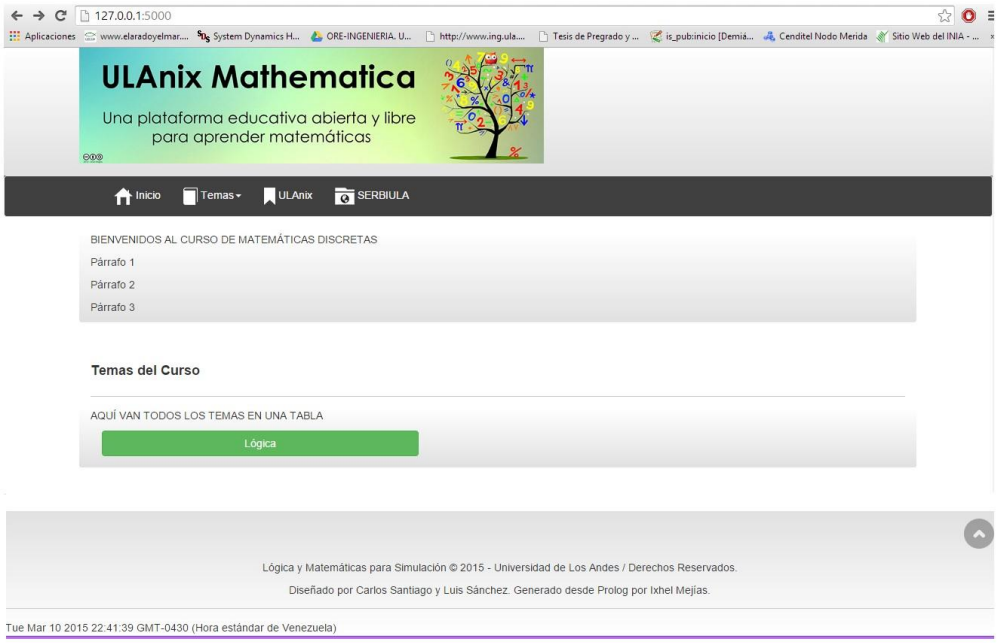

Figura 9. Página principal generada a partir de un código con HTML termerizado en Prolog.

Fuente: Elaboración propia.

En la figura 10 se puede observar el menú desplegable en la página principal. Al hacer *click* en el primer sub-ítem de "Temas", se redirecciona también a la página de Lógica mostrada en la figura 11.

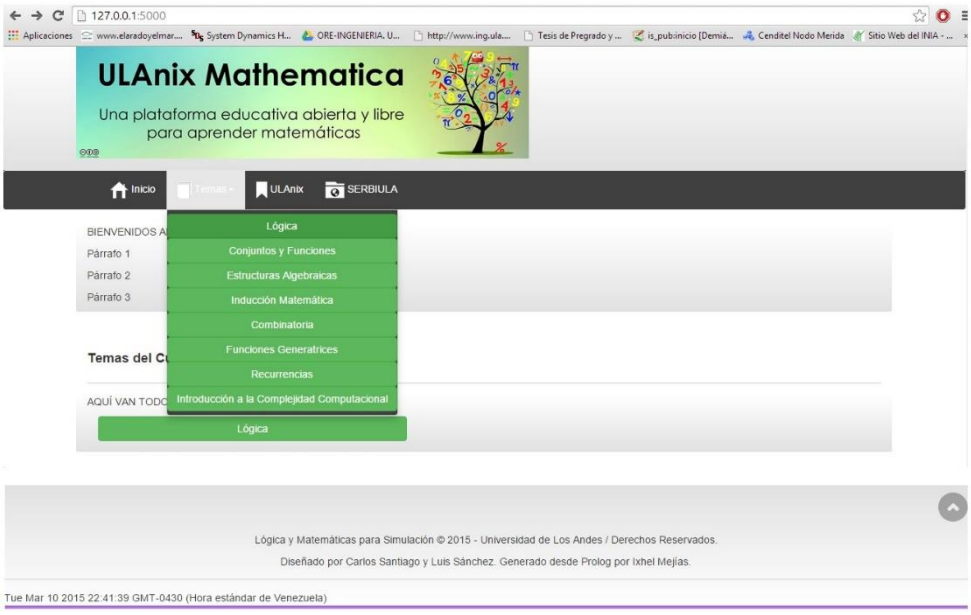

Figura 10. Menú desplegable en la página principal.
Fuente: Elaboración propia.

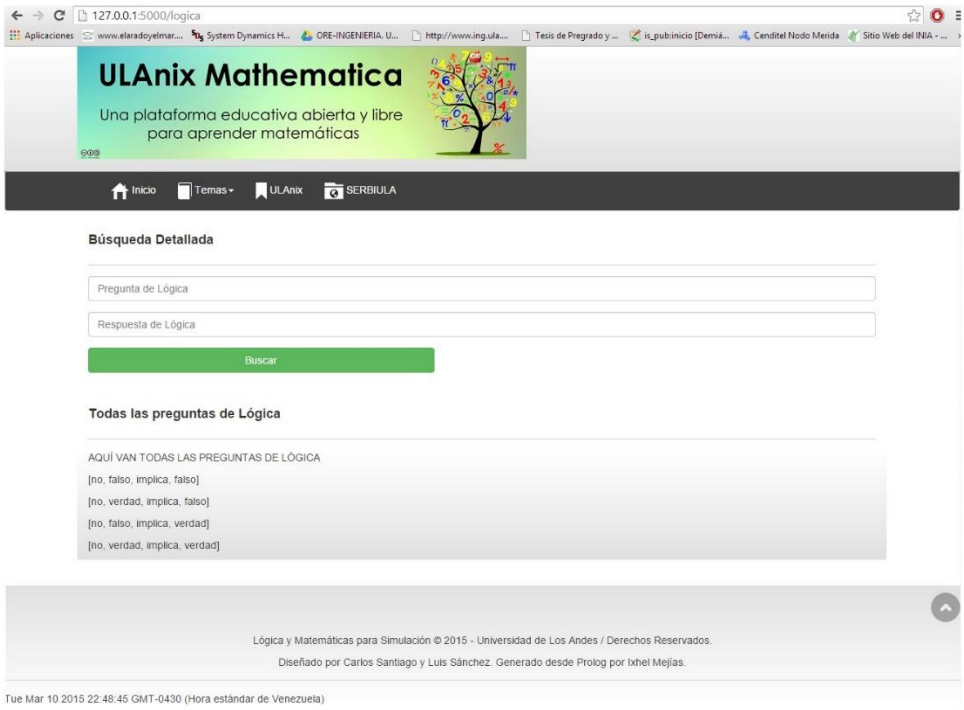

Figura 11. Página de Lógica generada a partir de un código con HTML termerizado en Prolog.
Fuente: Elaboración propia.

Desde la página de Lógica, se puede regresar a la principal a través del botón de "Inicio" del menú. Además, al ingresar una pregunta y darle *click* al botón "Buscar", se redirecciona a la página mostrada en la figura 12.

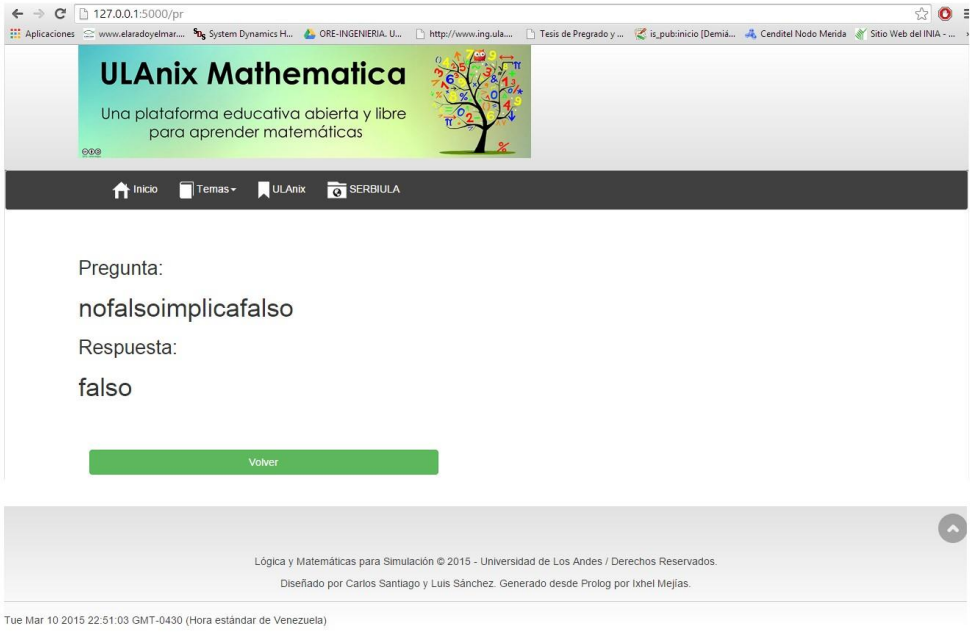

Figura 12. Página de Pregunta y Respuesta generada a partir de un código con HTML termerizado en Prolog.
Fuente: Elaboración propia.

6 Conclusiones

El producto obtenido del proyecto es la página web del tema de Lógica, generada a partir de un código HTML termerizado en Prolog. Dicha página web permite la consulta de preguntas y respuestas para el aprendizaje de Lógica. Se encuentra enlazada con la página principal. Puede servir de plantilla para crear las otras páginas de los distintos temas del proyecto ULAnix Mathematica.

Se realizó una extensa revisión bibliográfica en búsqueda de información para la creación de la página, a través de la cual se logró generar una página simple que luego se fue actualizando para incorporar nuevas aspectos de estilo y funcionalidades. Finalmente, se recopiló gran parte de la información encontrada y se elaboró el manual con el fin de servir de guía para futuros desarrollos.

6.1 Recomendaciones

La página web puede ser mejorada y/o ampliada en diversos aspectos, entre los cuales se tienen los siguientes:

- Considerar agregar los iconos de Bootstrap (Glyphicons) para sustituir las imágenes propias que se incluyen en la página. Evaluar si en realidad esto es conveniente ya que se ha dado el caso en que los Glyphicons no se muestran en el navegador Firefox.
- Agregar dinámicamente las preguntas de lógica.
- Cambiar botones de tipo btn-success a btn-link si se desea permitir abrir enlaces hacia los otros temas desde nuevas pestañas.
- Validar los campos de texto con JavaScript, se sugiere revisar el ejemplo hecho por Ogborn, disponible en: https://github.com/Anniepoo/weblog/blob/master/prolog/html_form/html_form.pl.
- Servir las imágenes desde Prolog (indicando en el código la ruta de la misma) para no utilizar hipervínculos hacia las imágenes en otro servidor.
- Agregar un enlace a la página web principal hacia otra página que contenga enlaces a material bibliográfico de cada uno de los temas de matemáticas.

Referencias Bibliográficas

Bootstrap (2015). Bootstrap - CSS. Disponible en: http://getbootstrap.com/css/. Consultada el 28 de febrero de 2015.

Dávila, J. (2014). ULAnix Mathematica. Una plataforma educativa abierta y libre para aprender matemáticas. Disponible en: http://jacinto-davila.blogspot.com/2014/09/ulanix-mathematica.html. Consultada el 03 de febrero de 2015.

Ogborn, A. (2013). Creating Web Applications in SWI-Prolog. Disponible en: http://www.pathwayslms.com/swipltuts/html/. Consultada el 28 de enero de 2015.

Santiago, C. y Sánchez, L. (2014). Página web del proyecto ULAnix Mathematica. Disponible en: http://nux.ula.ve/mathematica/mat_discretas.zip. Consultada el 20 de febrero de 2015.

SWI-Prolog (2015). How to create a web service easily? Disponible en: http://www.swi- prolog.org/howto/http/. Consultada el 31 de enero de 2015.