Tecnología Multimedia 2014 - 2015

Memoria Práctica Final

Pablo Riutort Grande
43185584W
pablo.riutort@gmail.com

27-05-15

Contents

1 Introducción

El objetivo de la práctica es el de utilizar la tecnología XML junto a HTML, Javascript y CSS para simular una web de servicios online. En este documento se presenta la memoria de dicha práctica.
Particularmente en esta práctica se ha escogido el tema de una heladería online, donde un cliente puede seleccionar un helado, elegir distintos sabores y solicitar que un heladero se lo traiga. El cliente también puede puntuar el servicio proporcionando feedback al heladero y a próximos clientes.

2 Desarrollo

En esta práctica pide que mediante diversos XMLs podamos verlos y editarlos en una página web. En nuestro caso los XML son:

1. **heladeros.xml**: Contiene información relacionada con nuestro profesional.
2. **formatos.xml**: Contiene información sobre los nombres y contenido del formato de un helado.
3. **sabores.xml**: Contiene información sobre los diferentes sabores que puedes seleccionar así como su precio e información sobre contenido multimedia relacionado.

Para hacer esta página web se ha seguido el siguiente criterio de forma generalizada: *Una página equivale a un documento HTML y a un documento de Javascript que controla dicho HTML.*
Así pues, si tenemos un documento HTML titulado: index.html, entonces tendremos un documento en titulado index.js que lo acompañe.

2.0.1 Entorno de desarrollo

Todas las vistas se encuentran en la raíz del fichero de la práctica así como los ficheros de PHP.

1. **js/**: Este directorio contiene todos los ficheros de Javascript utilizados por los HTML.
2. **css/**: Contiene todos los documentos de estilos utilizados por los HTML. Estos documentos no siguen el mismo criterio que los archivos de Javascript debido a su naturaleza general y reutilizable en diversos contextos.
3. **chats/**: Esta carpeta contiene archivos XML que guardan las conversaciones que se mantienen entre un profesional (heladero) y un cliente.
4. **img/**: Contiene todos los archivos que sean imágenes que utiliza la práctica. También contiene los archivos SVG.
5. **XML/**: Contiene los archivos XML anteriormente mencionados.
6. **includes/**: Esta carpeta contiene archivos HTML que son reutilizables o plantillas que pueden servir para generar el mismo código repetidas veces, como el caso del footer de la página y del botón de RSS.
7. **documentación/**: En esta carpeta se encuentran los archivos relacionados con la documentación de esta práctica. Este mismo archivo mas otro que explica la primera versión de la creación de los XML.

Siguiendo este sistema podemos encontrar más fácilmente qué funciones actúan sobre cada archivo y vemos una relación más directa entre la vista y lo que realmente ocurre dentro de ella.

2.1 Mains

Aparte de los archivos específicos de cada vista, existen archivos que son usados de forma general por todo el proyecto, estos son:

- **js/main.js**: Este archivo es llamado por todas las vistas, sirve para incluir el footer en todas etiquetas "footer" de todas las vistas. También existe para extender el uso fácilmente sin tener que ir a cada js individualmente.
- **css/main.css**: De igual forma que existe un archivo Javascript usado por todas las vistas, existe un archivo de estilos general para todas las vistas que extiende el css aplicado por Bootstrap, por ejemplo la clase ".centered" se encuentra en este archivo, que centra los textos.

2.2 Tecnologías

Para esta práctica se han utilizado algunas herramientas de desarrollo y desarrollo web que han permitido agilizar la creación de la misma. Estas herramientas son frameworks y otros programas sobre los que se apoya la práctia y el proceso de desarrollo.

2.2.1 jQuery

jQuery es una libreria de Javascript ampliamente utilizado en el desarrollo web. En el vienen definidas muchas funcionalidades de Javascript de una manera mucho más accesible para el entorno web de lo que es Javascript como lenguaje en sí. `https://jquery.com/`.

2.2.2 Bootstrap

Bootstrap es un framework de CSS creado por Twitter pensado para diseñar webs de forma rápida y mantenible en múltiples dispositivos y pantallas. Este framework junto a un tema del mismo ha servido para confeccionar todos los estilos de la web. `http://getbootstrap.com/`.

2.2.3 Fonts Awesome

Los iconos de toda la web provienen de la página Fonts Awesome, que mediante tags permiten llamar a una amplia selección de iconos en distintos tamaños. La importación de los iconos es tan sencilla como introducir un tag en el header del html, al igual que hacemos con jQuery.
`http://fortawesome.github.io/`

2.2.4 Git & GitHub

Git es un programa de control de versiones. Git permite crear versiones del código y volver atrás de forma segura por si no se obtiene el resultado esperado o algo ha dejado de funcionar correctamente. Con Git también se pueden hacer ramas que no interfieren en el desarrollo principal del proyecto y si el resultado es desable, se puede juntar todo.
Este programa ha sido de gran utilidad para el desarrollo de la práctica. `https://git-scm.com/`
Podemos encontrar una copia de esta práctica en:

`https://github.com/pabloriutort/Tecnologia-Multimedia`.

2.2.5 Vagrant

Vagrant permite crear entornos de desarrollo de manera fácil y ligera con VirtualBox.
Vagrant puede crear entornos de desarrollo de manera virtual e instalar las dependencias de programas allí sin necesidad de sobrecargar el ordenador que se esté usando programar. También permite la conexión por IP que ha permitido la construcción de una web local con la práctica para ver los resultados. `https://www.vagrantup.com/`

2.2.6 SVG e Identicons

El contenido de algún SVG era obligatorio en esta práctica, para cumplir este objetivo se ha utilizado la siguiente web `http://philbit.com/svgpatterns/#honeycomb` y se han modificado dos patrones, cambiando los colores. Se han utilizado en los paneles superiores de la heladería y del backend como fondos, añadiendo una regla en el fichero main.css.
Para los identicons, que son las "fotos de perfil" de nuestros heladeros hemos utilizado la siguiente página web: `https://github.com/identicons/miquel.png`. Sustituyendo el nombre del final por el deseado genera un identicon único en formato .png.

3 Heladeria

3.1 index.html

Es la vista principal de la página, contiene información sobre el servicio que se va a ofrecer. Cuando se pulsa el botón de continuar se vacía el **localStorage** de HTML5, pues es lo que utilizamos para gestionar los datos del usuario.

3.2 helados.html

Esta página muestra la selección de helados y sus sabores. Permite al cliente seleccionar un formato para su helado así como el número de bolas de cada sabor. Una vez seleccionado todo esto, podemos pasar a contactar con el heladero.
El archivo helados.js carga los formatos desde el XML, su nombre y la ruta de la imágen en el directorio img/:

```
//Carga los sabores
function set_flavours() {

 $.get('XML/sabores.xml', function (d) {

  $(d).find('sabor').each(function () {

    var sabor = $(this);
    var nombre = sabor.find("nombre").text();
    var imageurl = sabor.find('img').text();
    var precio = sabor.find('precio').text();

    var html = '<article>';
    html += '<div class="col-sm-6 col-md-2">';
    html += '<div class="thumbnail">';
    html += '<img src="' + imageurl + '" alt="' + nombre + '">';
    html += '<div class="caption centered">';
    html += '<header>
        <h3 class="nombre-sabor">' + nombre + '</h3>
                <span class="precio"></span>
        </header>';
            html += '<p class="bolas"><span>Quiero </span>
        <input type="number" name="quantity" min="0" max="7"> bolas
        </p>';
            html += '</div>';
            html += '</div>';
            html += '</div>';
            html += '</article>';
```

```
        $('#sabores').append(html);

        });
});
}
```

** Pasar la vista directamente desde Javascript no es una buena práctica, pero diversas complicaciones han hecho que quedara de esta forma.*

3.2.1 Filtro

Esta página contiene también un filtro de búsqueda de sabores que utiliza un xslt para encontrar el sabor introducido en el filtro. Dicho filtro es activado cuando el usuario escribe el sabor a filtrar y selecciona el botón "Filtrar":

```
//Esto es la funcionalidad del filtro con xsl
    $("#filter_confirm").click(function(){
        var filter_value = $("#filter_text").val()
        if (filter_value) {
            $("#sabores").load("filter.php?sabor=" + filter_value);
        }else{
            set_flavours();
        }
    });
```

Este código llama al fichero filter.php que a su vez llama al fichero filter.xsl y al XML de los sabores para filtrarlos.

Filter.php:

```
<?php
/**
 * Created by PhpStorm.
 * User: pablo
 * Date: 26/05/15
 * Time: 11:45
 */

$sabor = $_GET['sabor'];

$xsl_filename = "filter.xsl";    /* nom arxiu xsl */
$xml_filename = "XML/sabores.xml";        /* nom arxiu xml */

$doc = new DOMDocument();
$xsl = new XSLTProcessor();

$doc->load($xsl_filename);

$xsl->importStyleSheet($doc);

/* pasam par?metre al xsl */
$xsl->setParameter('', 'param',$sabor);

$doc->load($xml_filename);
```

```
echo $xsl->transformToXML($doc);
?>

Filter.xsl:
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:template match="/">
  <xsl:for-each select="sabores/sabor">
   <xsl:if test="nombre=$param">
    <article>
     <div class="col-sm-6 col-md-2">
      <div class="thumbnail">
       <img src="" alt="">
        <xsl:attribute name="src">
        <xsl:value-of select="img"/>
        </xsl:attribute>
       </img>
       <div class="caption">
        <header>
         <h3 class="nombre-sabor"><xsl:value-of select="nombre"/></h3>
        </header>
        <p><span>Quiero </span>
         <input type="number" name="quantity" min="0" max="7"/> bolas
        </p>
       </div>
      </div>
     </div>
    </article>
   </xsl:if>
  </xsl:for-each>
 </xsl:template>
</xsl:stylesheet>
```

Como se puede observar en el fichero XSL, lo que se hace es pasar una vista idéntica a la de un sabor de helado, pero con sentencias xsl que permiten filtrar por la palabra introducida por el usuario (en este caso un sabor). Este xsl en concreto recorre todos los nodos sabores/sabor que del XML sabores.xml y si encuentra alguna coincidencia con la palabra escrita, entonces muestra la vista del sabor con el nombre seleccionado.
De esta forma creamos el filtro de sabores de la heladería.

3.3 heladeros.html

Después de seleccionar un helado y un formato, pasamos a seleccionar a nuestro heladero.
En esta vista veremos una tabla con los nombres de los heladeros así como información de la heladería donde trabajan y su biografía. Esta tabla se carga de manera muy similar a los sabores de los helados en la vista helados.html, se carga el XML de heladeros.xml y seleccionamos la información que nos interesa, en este caso el ¡nombre¿, la ¡heladeria¿ y la ¡bio¿

```
function set_stores(){

 $.get('XML/heladeros.xml', function(d){
  $(d).find('heladero').each(function(){

   var $heladero = $(this);
   var nombre = $heladero.find("nombre").text();
```

```
    var imageurl = $heladero.find('img').text();
    var descripcion = $heladero.find('bio').text();
    var tienda = $heladero.find('heladeria').text();

    var html = '<tr>';
    html +=    '<td class="seller"><a href="#">'+nombre+'</a></td>';
    html +=    '<td class="description">'+descripcion+'</td>';
    html +=    '<td class="store">'+tienda+'</td>';
    html += '</tr>'

    $('#heladeros').append(html);

   });
  });
}
```

Como se puede observar se sigue el mismo proceso de leer el XML y escoger la información de que nos interesa.

En esta vista se da la peculiaridad de que funciona también como detalle del heladero. Cuando hacemos click en el nombre del heladero que nos interesa, aparece más abajo su información, esto lo hacemos de la siguiente manera:

```
$('body').on('click','td.seller',function(){
 var name = $(this).text();
 var desc = $(this).next().text();
 var store = $(this).next().next().text();

 $("#seller-name").text(name);
 $("#seller-store").text(store);
 $("#seller-section").removeClass('hidden');
 $("#chat").removeClass('hidden');
 $("#confirm").removeClass('hidden');

 $("#seller-jpg").attr('src','img/heladeros/'+name.split(' ')[0].toLowerCase()+'.pn

 sessionStorage.setItem('seller-name',name);
        ...
```

En la misma función utilizamos la información obtenida para declarar que el heladero seleccionado por el usuario es el seleccionado de la tabla tal y como podemos observar con el el sessionStorage.setItem.
Después de seleccionar al heladero, en la parte inferior de la página veremos su información más detallada: Foto de perfil, últimas valoraciones, ubicación y un chat.

3.3.1 Maps

La web también nos da una ubicación aproximada del heladero. Esto se ha conseguido mediante el uso de la API de google maps: `http://www.w3schools.com/googleapi/`

El fichero maps.js contiene todas las funciones necesarias para mostrar un mapa de google maps. La ubicación del heladero no es real, se coge la ubicación del usuario de la página y se le aplica un desfase aleatorio a su latitud y longitud creando así la ilusión de movimiento del heladero.

```
...
var pos = new google.maps.LatLng(
 position.coords.latitude + getRandomArbitrary(0.001,0.02),
 position.coords.longitude + getRandomArbitrary(0.001,0.02));
...

function getRandomArbitrary(min, max) {
    return Math.random() * (max - min) + min;
}
```

3.3.2 Chat

El chat es una conversación entre el heladero y el cliente. Para hacerlo se mantiene un XML por cada heladero con su nombre. En dicho XML se guarda la información del chat introducida y se muestra el contenido del documento en la ventana del chat. Lógicamente si alguien deseara escribir en el chat, el archivo se modificaría con la nueva información, poniendo al final del documento los caracteres introducidos.

```
//tambien en heladeros.js, en la funcion anterior

  getXML(addressed_to);

  $("#chat-form").submit(function(e){
   e.preventDefault();
   addComent(addressed_to,$("#send-message").val());
  });
 });
});

var xmlDoc;

function getXML(who) {
 var req;
 if (window.XMLHttpRequest) {
  req = new XMLHttpRequest();
 } else {
  req = new ActiveXObject("Microsoft.XMLHTTP");
 }
 req.open("GET","chats/"+who+".xml", true);
 req.onreadystatechange = function () {
  if (req.readyState == 4 && req.status == 200) {
   xmlDoc = req.responseXML;
   console.log(xmlDoc);
   parseXML();
  }
 };
 req.send();
}

function parseXML() {
 document.getElementById('chat-body').innerHTML = '';
 var content = xmlDoc.getElementsByTagName("p");
 for (var i = 0; i < content.length; i++) {
```

```
   var coment =xmlDoc.getElementsByTagName("p")[i].childNodes[0].nodeValue;
   document.getElementById('chat-body').innerHTML += '<p>' + coment +'</p>';
  }
}

function saveXML(who) {
 var req;
 if (window.XMLHttpRequest) {
  req = new XMLHttpRequest();
 } else {
  req = new ActiveXObject("Microsoft.XMLHTTP");
 }
 req.open("POST", "save_chats.php?filename=chats/"+who+".xml", true);
 req.send(xmlDoc);
}

function addComent(who,content) {
 var coment = xmlDoc.createElement("p");
 xmlDoc.getElementsByTagName("llistacomentaris")[0].appendChild(coment);
 coment.htm
 coment.innerHTML =  'cliente: '+content ;

 parseXML();
 saveXML(who);
}
```

Las funciones utilizadas son las mismas que las ofrecidas en clase, con la única diferencia de que por parámetro pasamos la persona a la que seestá enviando, para saber qué XML hay que leer y así utilizar la conversación del heladero.
Esta edición del documento XML se puede realizar mediante el formulario que aparece debajo del mapa del heladero, simulando una ventana de chat.

3.4 form.html

Después de seleccionar al heladero y nuestro pedido, se pasa al formulario de pago.
En el formulario tenemos los campos más frecuentes: nombre y apellidos, teléfono, mail, etc.
Mediante HTML5 hacemos una validación en la misma vista, así nos ahorramos tener que tratar los datos introducidos nosotros mismos. Esto se hace de la siguiente manera:

```
<input type="name" class="form-control" id="name"
placeholder="Introduzca su nombre" title="Se necesita un nombre"
required>
```

El atributo "required" hace que si el usuario no introduce ningún valor en este campo, se pare la ejecución de la acción de submit y muestre el contenido del atributo "title".
La validación del formulario en HTML puede ser más compleja:

```
<input type="email" class="form-control"
id="email" placeholder="Introduzca un email" required>
```

El atributo de "type" espera un tipo email, por tanto, si el input no contiene una @ o algo del tipo "nombre"@"dominio"."com/es/etc.." mostrará un title predeterminado para estos casos.

Aparte del formulario y la validación también mostramos un pequeño resumen del pedido. Este resumen lo hemos construido mediante HTML y su localStorage. Como ya hemos visto antes en el localStorage tenemos guardado el nombre del heladero, pero antes también habremos guardado los

helados que hemos pedido de la siguiente forma:

```
//En helados.js
//Si hemos terminado, entonces recogemos los datos introducidos.
 $('#confirm').click(function(){
 var item = 0;
 //Recorremos todos los sabores
 $("#sabores").find('article').each(function(){
  var ice_value = $(this).find('input').val();
  //Si ha sido marcado, entonces lo guardamos en el Storage
  if (ice_value) {
   var flavour = $(this).find('.nombre-sabor').text();
   var command = [flavour,ice_value];
   localStorage.setItem('sabor_cantidad'+item,command);
   item += 1;
  }
 });
 localStorage.setItem('total_helados',item);
});
```

Hacemos un recorrido de todos los inputs, si contienen algo guardamos el sabor y el número de bolas en localStorage y así, podemos mostrarlo en el resumen.

Una vez hecho el recorrido de localStorage, podemos sacar todos los datos que necesitemos para el resumen. El usuario simulará un pago introduciendo sus datos en el formulario anteriomente mencionado y pasamos a la evaluación.

3.5 gracias.html

En esta vista mostraremos un mensaje de agradecimiento al cliente y le pediremos, si lo desea, valorar el servicio.

3.5.1 Valoracion

La valoración será recogida en el documento XML de heladeros.xml haciendo alusión al heladero en cuestión y añadiendo un nodo al final llamado "rating" guardaremos las valoraciones de los usuarios para mostrarlos a futuros usuarios y al propio heladero.

```
$(document).ready(function(){
 //Si pulsamos un bot n, entonces sera una valoracion.
 $("button").click(function(){
 //Hacemos una peticion ajax a save_valuation.php
  $.ajax({
   url: 'save_valuation.php',
   type: 'POST',
   dataType: "json",
   data: {
   //Y le pasamos los parametros siguientes
    message: $('textarea').val(), //mensaje
    rating: $(this).text(), //valoraci n
    to: sessionStorage.getItem('seller-name') //a quien va dirigido
   }
  });
```

```
  alert("Le agradecemos su valoraci n");
 });
});
```

A continuación, el siguiente script se encarga de guardar la valoración en el XML.

```
//save_valuation.php
<?php
/**
 * Created by PhpStorm.
 * User: pablo
 * Date: 19/05/15
 * Time: 18:36
 */

$message = $_POST['message'];
$rating = $_POST['rating'];

if ($message) $rating = $_POST['rating'].': '.$message;

$to = $_POST['to'];

$doc = new DOMDocument();
$doc->load( 'XML/heladeros.xml' );

$employees = $doc->getElementsByTagName( "heladero" );
foreach( $employees as $employee )
{
    $names = $employee->getElementsByTagName( "nombre" );
    $name = $names->item(0)->nodeValue;

    if ($name == $to){
        $employee->appendChild($doc->createElement('rating',$rating));
    }

}

echo $doc->save('XML/heladeros.xml');
```

3.6 RSS

Para el RSS se han seguido las instrucciones establecidas en la web de la asignatura, como resultado tenemos el fichero heladeria.rss, que muestra el siguiente contenido.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rss version="2.0">
 <channel>
 <title>Heladeria Online</title>
 <link>pabloriutort.github.com/Tecnologia-Multimedia</link>
 <description>En la helader a online un heladero vendr  a traerte lo que necesite

 <item>
 <title>Nuevos heladeros!</title>
 <link>pabloriutort.github.com/Tecnologia-Multimedia/heladeros.html</link>
 <description>Mira qu  nuevos heladeros vienen</description>
```

```
 </item>

 </channel>
</rss>
```

4 Backend

El Backend es donde el heladero puede conectarse y ver quien le ha hablado por chat y su última valoración.

4.1 login.html

El login consiste en un formulario que comprueba que el nombre introducido pertenece a un heladero y simula una contraseña. Para saber si el nombre coincide volvemos a consultar el nombre del heladero en nuestro archivo de heladeros.xml.
Para hacer login hay que introducir **el nombre** de uno de nuestros heladeros, a elegir entre:

1. Miquel Noguera
2. John Doe
3. Mary Moe

4.2 backend.html

Por último, nos encontramos el backend. Este lugar es para que el heladero se comunique con el cliente y vea su última review, por tanto, solo tiene una vista del chat, que ya hemos explicado como funcionaba en la sección de heladeros.html, y una vista de la última valoración que le haya hecho un cliente.
Para mostrar la última valoración utilizamos la siguiente función del archivo backend.js:

```
function set_valuations() {

 $.get('XML/heladeros.xml', function (d) {

 var valoracion = "";

 $(d).find('heladero').each(function () {

  var heladero = $(this);
  var nombre = heladero.find('nombre').text();

  if (nombre == localStorage.getItem('login-name')){
   heladero.find('rating').each(function(){
    valoracion = $(this).text();
   });
  }

 });

 $(".alert").each(function () {
  if ($(this).attr('id') == valoracion.split(':')[0]) {
   $(this).text(valoracion);
   $(this).removeClass('hidden');
  }
```

```
  });

});
}
```

Consultamos el fichero heladeros.xml y si coincide el nombre introducido y guardado en localStorage con el del fichero, entonces miramos su valoraciones.
Las valoraciones pueden ser cuatro:

1. Exelente: La mejor valoración, corresponde a un recuadro verde.
2. Bien: La segunda mejor valoración, corresponde a un recuadro verde.
3. Mal: La segunda peor valorción, corresponde con un recuadro naranja.
4. Horrible: La peor valoración, corresponde con un recuadro rojo.

Los recuadros se consiguen mediante las clases de bootstrap aplicadas a los paneles: .alert, .info, .warning, .error.

Si la review coincide con el id de alguno de estos recuadros, entonces es mostrada.
Se ha optado por este método de valoración porque es bastante más sencillo y limpio que calcular medias introducidas por otros usuarios.

5 Conclusiones

El proyecto ha sido relativamente fácil de realizar apoyándose en los distintos frameworks y herramientas y siguiendo la metodología de trabajo anteriormente especificada.
La principal dificultad que he econtrado ha sido el tratramiento de los XML, pero siguiendo las indicaciones mencionadas en la web de la asignatura, estas dificultades han ido decrementando.