

Tienda de música

MongoDB + ExpressJS + JQuery + NodeJS

Introducción

Para esta práctica se ha creado una parte de lo que sería una tienda online de discos de música. Las páginas creadas fueron la **home**, página de **categoría**, página de **disco**, página para que un usuario pueda registrarse o **suscribirse** y una página de **panel de administrador** en la que sólo podrán entrar usuarios con ciertos privilegios.

Los módulos que se han usado han sido los siguientes:

- **MongoDB**: Donde se ha establecido la base de datos no relacional.
- **ExpressJS**: Para montar el servidor donde estará alojada la web.
- **JQuery**: Para realizar eventos según las diferentes acciones que haga el usuario/cliente a través de su navegador (FrontEnd).
- **NodeJS**: Para toda la parte que el servidor puede crear sin necesidad de la interacción continua con el usuario/cliente y que éste visualiza (BackEnd).

Lógica de la aplicación

Anteriormente se ha explicado brevemente para que se ha usado cada módulo, pero para poder usar **ExpressJS**, framework de **NodeJS**, con ellos, son necesarios otros módulos adicionales para diferentes funcionalidades de la tienda de música, que más adelante se explicarán y que son:

- **Jade**.
- **Stylus**.
- **Express session**.
- **Cookie parser**.

Servidor web

El servidor en el que estará alojada la tienda de música será en un servidor local (localhost) y la aplicación que se encarga de proporcionar el servicio para acceder a ésta será **ExpressJS**.

Para tener **ExpressJS** funcionando en nuestro servidor local éste tiene que estar en escucha continua, en caso contrario aunque intentamos acceder a la web, no podremos conectarnos. Los archivos que se encargan de ello son **app.js** y **www** (éste se ubica en la carpeta bin). El archivo **www** se encarga de establecer el puerto donde el servidor estará escuchando y luego crea la conexión. Posteriormente es el archivo **app.js** el que carga todas las librerías y establecerá donde se encontrarán los archivos necesarios para mostrar

la web, como son las rutas (routes, archivos Javascript con el código necesario para la carga de cada página de la web), carpeta de estilos (Stylus), etc.

Base de datos

Para poder conectarnos a la base de datos de **MongoDB** con **ExpressJS** debemos tener el módulo instalado en éste y crear la instancia de la base de datos a la que se va a acceder, así como el puerto en el que se va a encontrar en escucha continua (éste es el puerto que va a establecer ExpressJS no el puerto de escucha de la aplicación de MongoDB en el servidor) para poder acceder a él de forma sencilla.

Donde tengamos el archivo Javascript que controla nuestra aplicación (app.js) es donde establecemos la conexión a la base de datos:

```
db.connect('mongodb://localhost:27017/tienda', function(err) {  
  if (err) {  
    console.log('Unable to connect to Mongo.');    process.exit(1);  
  } else {  
    app.listen(3024, function() {  
      console.log('Listening on port 3024...');  
    });  
  }  
});
```

La base de datos está en un servidor local por lo que en la url que nos conectamos escribimos **localhost**; luego el puerto donde la parte del servidor está en escucha con la aplicación de mongo, que en este caso es **27017** (por defecto); y finalmente el nombre de la base de datos, como podemos ver aquí es **tienda**.

Finalmente para acceder a la base de datos desde **ExpressJS** simplemente realizamos la consulta que queramos, si queremos obtener los usuarios que hay en la base de datos:

```
//Obtener todos los usuarios  
db.get().collection('users').find();  
  
//Obtener todos los usuarios que son de Granada  
db.get().collection('users').find({city: 'Granada'});
```

Backend (Lado del servidor)

Para escribir el código que se mostrará al cliente normalmente se ha usado **HTML** (ya sea o HTML puro o creándolo desde PHP), pero en este caso se ha usado **Jade**. Este lenguaje hace que resulte más sencillo programar ya que su código es más simple y para crear elementos no es necesario establecer ni etiquetas ni etiquetas dentro de etiquetas,

simplemente en el propio código si queremos saber hasta donde termina un elemento se usa la indentación:

Código en HTML

```
<nav class='navbar navbar-default navbar-fixed-top'>
  <div class='container'>
    <div class='navbar-header pull-left'>
      <a class="btn btn-default btn-xs navbar-btn" href="/">Orders</a>
      <a class="btn btn-default btn-xs navbar-btn" href="/manage_orders">Manage</a>
      <a class="btn btn-default btn-xs navbar-btn" href="/users">Users</a>
    </div>
    <div class='navbar-header pull-right'>
      <a class="btn btn-default btn-xs navbar-btn" data-method="delete"
href="/users/sign_out" rel="nofollow">Logout</a>
    </div>
  </div>
</nav>
```

Código en Jade

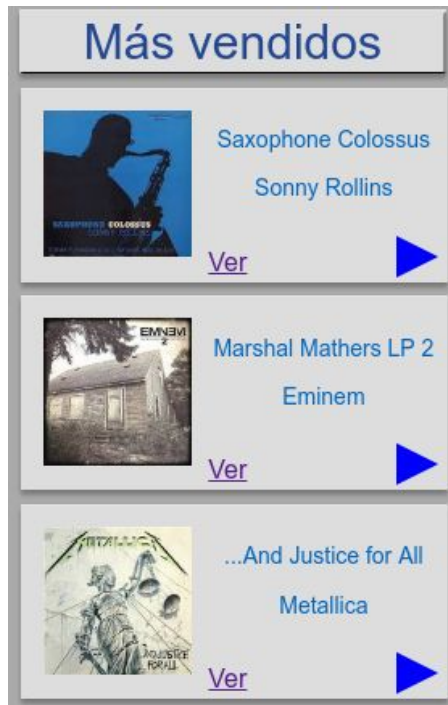
```
nav.navbar.navbar-default.navbar-fixed-top
  .container
    .navbar-header.pull-left
      a.btn.btn-default.btn-xs.navbar-btn(href='/') Orders
      a.btn.btn-default.btn-xs.navbar-btn(href='/manage_orders') Manage
      a.btn.btn-default.btn-xs.navbar-btn(href='/users') Users
    .navbar-header.pull-right
      a.btn.btn-default.btn-xs.navbar-btn(data-method='delete', href='/users/sign_out',
rel='nofollow') Logout
```

Este lenguaje trae más ventajas a parte de la sencillez, podemos usar condiciones, bucles y muchas más funcionalidades.

Cuando desde **ExpressJS** conectamos con el código **Jade** podemos pasarle a éste un JSON de datos que podemos usar posteriormente en él para abstraernos de la información que queremos enviar. Por ejemplo imaginemos que tenemos un JSON con datos específicos de un disco, entonces cuando creemos con **Jade** la página del disco, podremos abstraernos y en lugar de tener un archivo **Jade** para cada disco, tenemos uno solo:

```
.box.discs(style='margin-left: 3%')
  img.square(src= disc.image_url)
  .adjust-to-center-left
    .center-data-disc= disc.title
    .center-data-disc= disc.author
  .play-triangles
  a.center-bottom-alignment(target='_blank', href= disc.url) Ver
```

La palabra **disc** es el JSON de datos que contiene la información respecto al disco que se quiere mostrar. ¿Cómo se realiza correctamente esta abstracción? Para ello desde **ExpressJS**, en el momento que vamos a renderizar llamando al archivo **Jade**, le pasamos un JSON donde en alguna parte de éste estará la información relevante de cada disco por lo que si queremos por ejemplo obtener los 3 discos más vendidos como se ve en la imagen:



En el **Jade** debemos tener algo como esto:

```
for disc in best_seller_discs
  .box.discs(style='margin-left: 3%')
    img.square(src= disc.image_url)
    .adjust-to-center-left
    .center-data-disc= disc.title
    .center-data-disc= disc.author
    .play-triangles
    a.center-bottom-alignment(target='_blank', href= disc.url) Ver
```

Para que el **Jade** entienda correctamente el array de datos **best_seller_discs**, éste debe existir éste en el JSON que se le ha enviado. Por lo que en **ExpressJS** tendremos algo como esto:

```
discs.find().sort( { sold: -1 } ).limit(3).toArray(function(err, best_seller_docs) {
  res.render(archivo-jade, { title: 'DiscoShop', best_seller_discs: best_seller_docs,
page: 'home' });
});
```

En la base de datos de mongoDB obtenemos los tres primeros discos ordenados de mayor a menor (sold: -1) y se convierten a un array **best_seller_docs** que se guardará en el valor JSON **best_seller_discs** del JSON enviado al archivo **Jade**.

Se ha seguido una serie de pasos como los anteriores para las operaciones relacionadas con la base de datos y que son enviadas al **Jade** para mostrar diferentes partes de cada una de las páginas, como son obtener las categorías que existen, los discos con mayor puntuación, más vendidos, últimos añadidos, etc. Y también para escribir en la base de datos con formularios que son recibidos desde el **Jade**, para registrar nuevos usuarios y para dar de alta nuevos discos.

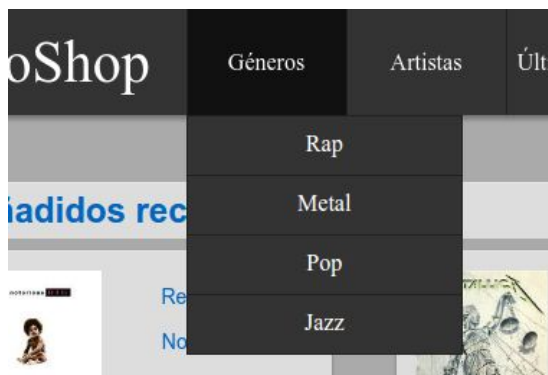
FrontEnd (Lado del cliente)

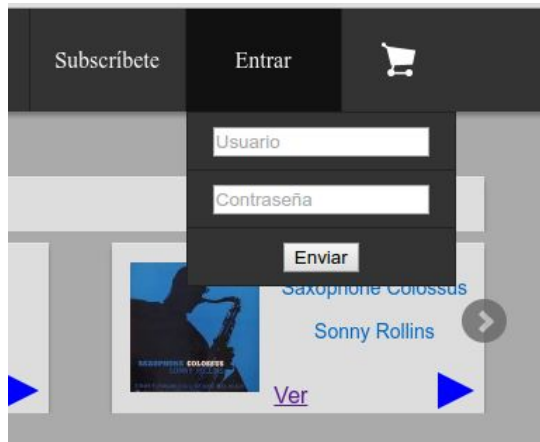
Para la interacción del cliente con su navegador y éste con el servidor de la tienda de música se ha usado la librería **JQuery**. Algunas de las implementaciones que se han realizado han sido:

- **Slider** en la home de la tienda en la que se muestran una serie de discos y el usuario puede interactuar con éste moviéndose a la derecha o a la izquierda:



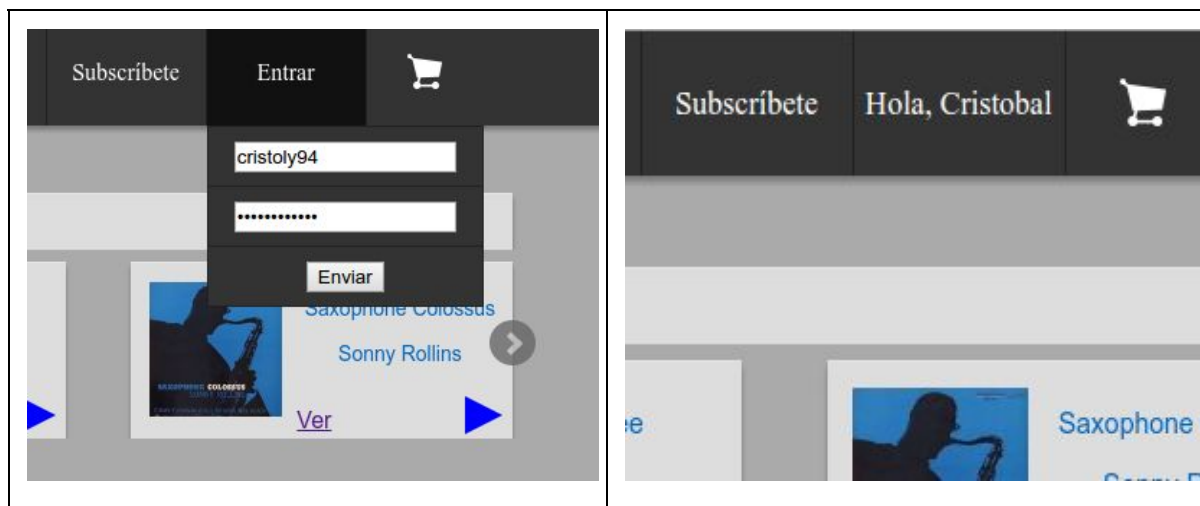
- **Desplegable** en el menú de navegación superior para diferentes acciones, ver las diferentes categorías o secciones de discos existentes o acceder y entrar en su cuenta como usuario:





Otra funcionalidad ha sido mantener la **sesión activa de usuario**. Esto se ha realizado desde **ExpressJS**. Para ello se ha usado el formulario de la imagen anterior para hacer login, que conlleva a una consulta en la base de datos de mongoDB para comprobar si el usuario con la contraseña proporcionada se encuentra en ésta y de ser así mantener su sesión activa. Los módulos necesarios para mantener las sesiones son **cookie-parser** y **express-session**.

Una vez que un usuario se ha registrado la web le muestra algunos de sus datos, por ejemplo ya no le pide que entre a su cuenta, y le muestra un mensaje de bienvenida:




Si el usuario no tiene cuenta puede hacer click en **Suscribirse** donde podrá meter sus datos y se escribirán en la base de datos para poder iniciar sesión posteriormente:

Formulario de Inscripción

Nombre	Apellidos
Nombre de usuario	contraseña
Dirección	
Ciudad	Código postal
Provincia	email
DNI	VISA
Observaciones	

Envío Diario ☐ Semanal ☐ Mensual ☐

Para añadir un extra a la funcionalidad a lo anterior también se ha diseñado que una vez que el usuario esté con su sesión activa, si éste tiene privilegios de admin entonces podrá acceder a la página de **panel de administración** para dar de alta nuevos discos, en caso contrario, si no ha iniciado sesión se le mostrará un mensaje de que inicie sesión con una cuenta de administrador, y si tiene sesión activa pero no tiene privilegios el mensaje le comunicará que no tiene los permisos suficientes para realizar esa acción. La página de admin es la siguiente:

DiscoShop Géneros Artistas Últimos discos Suscríbete Hola, Cristobal 

Panel de administración

Alta de nuevos discos

<input type="text" value="Título"/>	<input type="text" value="Autor"/>	<input type="text" value="Imagen URL"/>	<input type="text" value="Genero"/>	<input type="text" value="Precio"/>	<input type="text" value="Canción URL"/>	<input type="button" value="Enviar"/>
-------------------------------------	------------------------------------	---	-------------------------------------	-------------------------------------	--	---------------------------------------

Diseño

En cuanto al diseño no se ha usado archivos **CSS** sino archivos **Stylus**. Éstos últimos van acompañados con **Jade**, es decir, se suelen usar juntos. Lo positivo es que reduce en gran medida el código a parte de hacerlo más sencillo de escribir, entre otras cosas.

La forma de escribir el código en **Stylus** se asemeja al de **Jade**, este es un ejemplo de la diferencia entre **CSS** y **Stylus**:

Código en CSS	Código en Stylus
<pre>.dropdown li{ margin-bottom: 10px; text-align: center; border-bottom: 1px solid #222; height: 30px; } .dropdown li:first-child{ margin-top: 10px; } .dropdown li:last-child{ margin-bottom: 0; }</pre>	<pre>.dropdown li margin-bottom 10px text-align center border-bottom 1px solid #222 height 30px &:first-child margin-top 10px &:last-child margin-bottom 0</pre>

Cambios y/o complicaciones

- En un principio se iba a usar **AngularJS** para el lado del cliente en lugar de **JQuery** pero no fue posible debido a los diversos errores encontrados con la carga de las librerías.
- Se iba a usar **MySQL** para gestionar la base de datos pero finalmente se usó **MongoDB** para aprender y conocer el funcionamiento de las bases de datos no relacionales.

Referencias:

<http://stackoverflow.com/>

<http://expressjs.com/es/starter/installing.html>

<http://codehero.co/nodejs-y-express-instalacion-e-iniciacion/>

<http://www.copterlabs.com/json-what-it-is-how-it-works-how-to-use-it/>

<http://bxslider.com/examples/carousel-demystified>

<https://cdnjs.com/libraries>

<https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/>

<https://docs.mongodb.com/getting-started/node/client/>

<http://vstark.net/2014/10/18/jade-vs-html/>

<http://stylus-lang.com/>