Entrega 2 - MVC películas

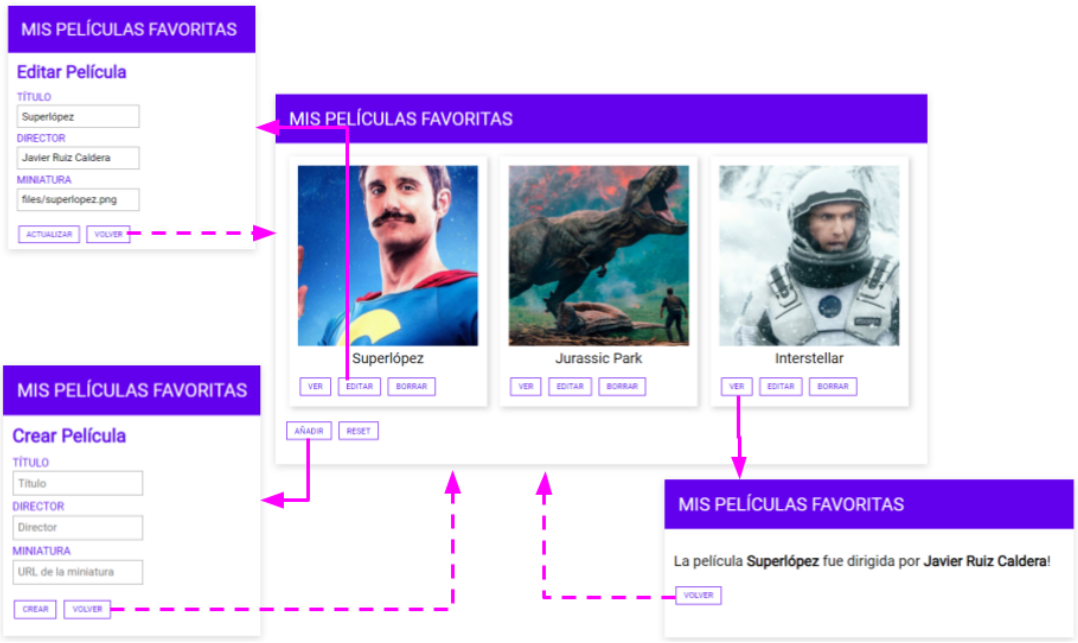
Versión: 27 de Enero de 2020

Objetivos

- Practicar HTML, CSS y JS y afianzar el concepto de MVC (Modelo-Vista-Controlador).
- Entender el uso y la problemática de localStorage como almacén de datos.

Descripción de la práctica

Esta entrega se basa en el desarrollo de una aplicación web de cliente usando las príncipales tecnologías web (HTML, CSS y JS) y el patrón MVC. La aplicación consiste en una base de datos de películas, en la que se almacena el título, director y URL de la carátula de cada película. En esta base de datos se puede consultar la información de las películas, añadir películas nuevas, editar las existentes, borrarlas y reiniciar la base de datos. Estas acciones se llevan a cabo al hacer clic en los distintos botones que proporciona la aplicación web.



En el código proporcionado sólo está implementada la funcionalidad de listar las películas existentes y editar película. El alumno debe implementar las funcionalidades restantes (crear, mostrar, eliminar y reiniciar).

Descargar el código del proyecto

El proyecto debe clonarse en el ordenador desde el que se está trabajando:

```
$ git clone https://github.com/CORE-2020/Entrega2_MVC_Cliente
```

A continuación se debe acceder al directorio de trabajo y abrir el fichero index.html con el editor de la elección del alumno.

```
$ cd Entrega2_MVC_Cliente
```

El fichero index.html contiene el código de la aplicación. Incluye tanto el HTML de la página, como el CSS y el código JavaScript que implementa la lógica de la aplicación siguiendo el patrón MVC. En las siguientes secciones se explica cada parte de esta lógica.

Modelo de datos

La información de las películas se modela mediante un array en el que cada película consiste en un objeto con las siguientes claves:

- **`titulo`** : Título de la película.
- **`director`** : Director de la película.
- **`miniatura`** : URL de la carátula de la película que se muestra como miniatura. Puede proporcionarse una URL pública o la URL de las imágenes alojadas en la carpeta `files` .

El array que contiene las películas se almacena en el localStorage del navegador bajo la clave `mis_peliculas` . Al iniciarse la aplicación, se comprueba si ya hay un array de películas almacenado en localStorage,

Vistas

Las vistas generan el código HTML que se inserta en el bloque `<div id="main"></div>` , como respuesta a los eventos que la aplicación recibe. Son funciones JavaScript que generan dinámicamente el código HTML de cada pantalla de la aplicación en función de los parámetros recibidos. Las vistas con las que cuenta la aplicación son las siguientes:

- **`indexView(peliculas)`** : Es la vista principal de la aplicación. Recibe como parámtero el array con todas las películas y genera el código HTML necesario para mostrar todas las películas con su título y miniatura, así como los botones que lanzan las diferentes acciones que se pueden aplicar sobre las mismas.
- **`editView(i, pelicula)`** : Es la vista que permite editar la información sobre una película existente. Recibe como parámetro la posición que ocupa la película en el array y el objeto que

contiene la información de la película. Renderiza un formulario que permite editar dicha información

Existen dos vistas adicionales que están incompletas y que el alumno debe implementar:

- **`showView(pelicula)`** : Debe mostrar el título y director de una película. Recibe como parámetro el objeto que contiene la información de la película.
- **`newView()`** : Debe mostrar un formulario para añadir una película nueva. Renderiza un input de texto para cada campo del objeto de película (título, director y miniatura), permitiendo al usuario introducir la información de la nueva película.

Controladores

Los controladores implementan la lógica de la aplicación que permite actuar en función de las acciones del usuario. Se encargan de acceder al modelo para extraer la información que necesitan y actualizarlo en función de las operaciones realizadas sobre los datos. También se encargan de llamar a las funciones que implementan las vistas y, mediante el método innerHTML de JavaScript, insertar el HTML que devuelven dichas funciones en el elemento con id `main` de la página. Los controladores con los que cuenta la aplicación son los siguientes:

- **`indexContr()`** : Es el controlador principal de la aplicación. Se encarga de acceder al array de peliculas y pasárselo a la vista `indexView` para que pueda generar el HTML correspondiente a la lista de películas.
- **`editContr(i)`** : Se encarga de llamar a `editView` pasándole la posición en el array de la película a editar (la película i-ésima, obtenida del modelo), así como su contenido, y mostrar un formulario de edición para modificar la información existente sobre la película.
- **`updateContr(i)`** : Se encarga de actualizar la información de la película i-ésima en el modelo una vez que el usuario ha terminado de rellenar el formulario de edición. Despues de modificarla vuelve a la vista principal.

Adicionalmente, la aplicación cuenta con otros cinco controladores incompletos que el alumno debe implementar:

- **`showContr(i)`** : Debe encargarse de la lógica que permite ver la información detallada (título y director) de una película. Recibe como parámetro la posición en el array (i) de la película cuya información se desea consultar. Debe obtener la película correspondiente a dicha posición y pasársela a `showView` para que pueda generar el HTML adecuado.
- **`newContr()`** : Debe encargarse de llamar a la vista necesaria para mostrar el formulario de creación de una película nueva.
- **`createContr()`** : Debe encargarse de añadir una película nueva al modelo según la información introducida por el usuario en el formulario de creación. Después de añadirla debe volver a la vista principal.

- **`deleteContr(i)`** : Debe encargarse de eliminar la película que ocupa la posición i-ésima en el array de películas y refrescar la lista de películas mostradas llamando al controlador principal. Antes de eliminar la película debe pedir confirmación al usuario mediante el método `confirm()` de JavaScript.
- **`resetContr()`** : Debe encargarse de dejar el array de películas en su estado original y refrescar la vista principal para cargar dichas películas.

Router

Se encarga de asociar los eventos de clic del usuario con los controladores adecuados. Para este propósito, cada vez que el usuario hace clic, se comprueba si el atributo de clase del elemento clicado coincide con alguna de las clases recogidas por el router. En caso afirmativo, se llama al controlador asociado a esa acción. En caso negativo, se ignora la acción. Algunas acciones requieren parámetros para poder ejecutarse. Por ejemplo, la acción de editar película precisa saber en qué posición del array de peliculas se encuentra la película a editar. Para ello, en cada botón de "editar", se añade un atributo `data-my-id` que contiene dicho índice. Para facilitar el acceso a dicho atributo se proporciona la función `myId` . La aplicación contempla las acciones resultado de hacer clic en elementos HTML con las siguientes clases:

- **`index`** : La clase `index` la tienen los botones de "Volver" de las vistas de `show` , `edit` y `new` que permiten regresar a la página principal. Al recibir este evento, el router llama al controlador principal (`indexContr`).
- **`edit`** : La clase `edit` la tienen los botones de "Editar" que se muestran bajo el título de cada película en la vista principal. Estos botones permiten editar la información de cada película llamando a `editContr` y pasándole la posición en el array de películas de la película que se desea mostrar a través del atributo `data-my-id` del botón.
- **`update`** : La clase `update` la tiene el botón de "Actualizar" película, que muestra el formulario de edición de cada película. Llama al controlador `updanteContr` , pasándole como parámetro el atributo `data-my-id` del botón.

El alumno debe implementar los cuatro eventos que faltan. Para ello debe añadir los botones que lanzan dichos eventos y asociarlos al controlador correspodiente en el router. Para poder recoger los eventos en el router, los botones deben incluir las siguientes clases:

- **`show`** : Se debe agregar un botón con el contenido "Ver" y la clase `show` bajo el título de cada película en la vista principal (delante de "Editar"). Al hacer clic sobre estos botones, se debe llamar al controlador que muestra la información detallada de cada película. Deben incluir el atributo necesario para poder recoger la posición de la película que se desea visualizar y así poder pasársela al controlador correspondiente.
- **`new`** : Se debe agregar un botón con el contenido "Añadir" y la clase `new` dentro del div con clase 'actions' de la vista principal. Al hacer clic sobre él, debe llamarse al controlador que muestra el formulario de creación de una nueva película, llamando al controlador correspondiente.

- **`create`** : Se debe agregar un botón con el contenido "Crear" y la clase `create` dentro del div con clase 'actions' de la vista de añadir película nueva. Este botón debe llamar al controlador que añade una película nueva al modelo a partir de la información introducida por el usuario en el formulario.
- **`delete`** : Se debe agregar un botón con el contenido "Borrar" y la clase `delete` bajo el título de cada película en la vista principal (detrás de "Editar"). Al hacer clic sobre estos botones, se debe llamar al controlador encargado de borrar una película del modelo. Deben incluir el atributo necesario para poder recoger la posición de la película que se desea visualizar y así poder pasársela al controlador correspondiente.
- **`reset`** : Se debe agregar un botón con el contenido "Reset" y la clase `reset` dentro del div con clase 'actions' de la vista principal. Este botón debe llamar al controlador encargado de restaurar el modelo a su estado original (las tres películas iniciales).

Tareas

Se pide modificar el código proporcionado para completar las cuatro funcionalidades que están incompletas:

- **Show**: Mostrar información sobre la película.
- **New**: Mostrar el formulario de añadir una nueva película.
- **Create**: Añade una nueva película al modelo.
- **Delete**: Elimina una película del modelo. Debe pedir la confirmación del usuario.
- **Reset**: Restaura el modelo al estado inicial, guardando las tres películas iniciales en localStorage.

Para implementar cada funcionalidad se recomienda seguir los siguientes pasos:

1. Añadir el botón correspondiente en la vista indicada incluyendo la clase que se indica en el apartado de "Router" y atributos (data-my-id) que sean necesarios.
2. Recoger el evento correspondiente a la clase del botón en el router de eventos y llamar al controlador indicado pasándole los parámetros necesarios.
3. Modificar el controlador correspondiente para realizar las operaciones de lectura/escritura del modelo correspondientes y actualizar la vista si es necesario.
4. Modificar la vista que corresponda a la funcionalidad a implementar.

Prueba de la práctica

Para ayudar al desarrollo, se provee una herramienta de autocorrección que prueba las distintas funcionalidades que se piden en el enunciado. Para utilizar esta herramienta debes tener node.js (y npm) (https://nodejs.org/es/) y Git instalados.

Para instalar y hacer uso de la herramienta de autocorrección en el ordenador local, ejecuta los siguientes comandos en el directorio del proyecto:

```
$ npm install -g autocorector     ## Instala el programa de test
$ autocorector                    ## Pasa los tests al fichero a entregar
.........................         ## en el directorio de trabajo
... (resultado de los tests)
```

También se puede instalar como paquete local, en el caso de que no se dispongas de permisos en el ordenador desde el que estás trabajando:

```
$ npm install autocorector     ## Instala el programa de test
$ npx autocorector             ## Pasa los tests al fichero a entregar
.........................      ## en el directorio de trabajo
... (resultado de los tests)
```

Se puede pasar la herramienta de autocorrección tantas veces como se desee sin ninguna repercusión en la calificación.

Instrucciones para la Entrega y Evaluación.

Una vez satisfecho con su calificación, el alumno puede subir su entrega a Moodle con el siguiente comando:

```
$ autocorector --upload
```

o, si se ha instalado como paquete local:

```
$ npx autocorector --upload
```

La herramienta de autocorrección preguntará por el correo del alumno y el token de Moodle. En el enlace https://www.npmjs.com/package/autocorector se proveen instrucciones para encontrar dicho token.

RÚBRICA: Se puntuará el ejercicio a corregir sumando el % indicado a la nota total si la parte indicada es correcta:

- **20%:** La funcionalidad de "Show" funciona correctamente.
- **20%:** La funcionalidad de "New" funciona correctamente.
- **20%:** La funcionalidad de "Create" funciona correctamente.
- **20%:** La funcionalidad de "Delete" funciona correctamente.
- **20%:** La funcionalidad de "Reset" funciona correctamente.

Si pasa todos los tests se dará la máxima puntuación.

¡Cuidado! Una vez enviadas, tanto la entrega, como la evaluación, no se pueden cambiar.