

# **AUT4\_04\_Marvel-Films**

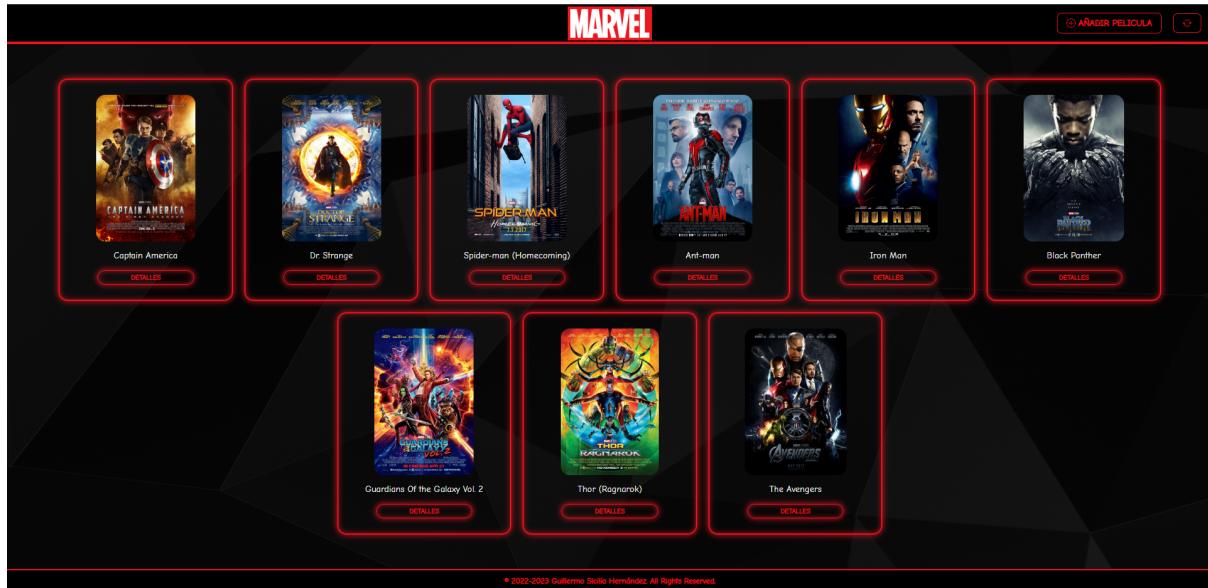
**Manual de Usuario y de Programador**

**Guillermo Sicilia Hernández  
2º CFGS - DAW**

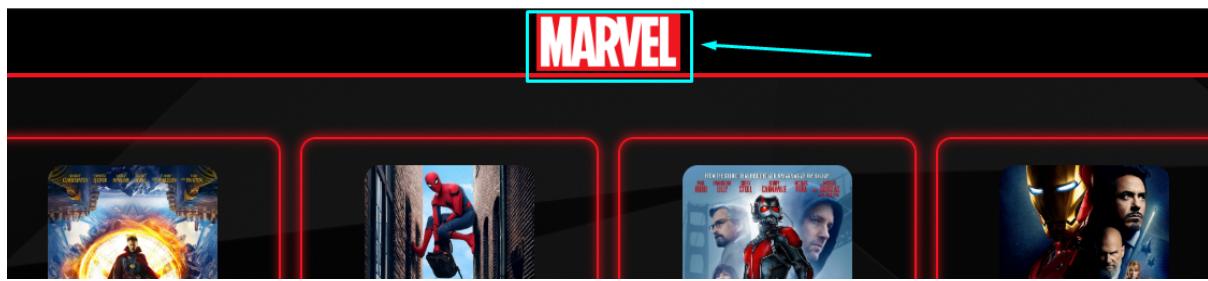
# **ÍNDICE.-**

<b>Vista principal de la página web.-</b>	<b>3</b>
<b>Información sobre marvel.-</b>	<b>3</b>
<b>Listado de peliculas.-</b>	<b>4</b>
<b>Detalles peliculas..-</b>	<b>7</b>
<b>Pipe títulos películas.-</b>	<b>10</b>
<b>Pipe acortamiento de la sinopsis.-</b>	<b>11</b>
<b>Directiva fase marvel.-</b>	<b>13</b>
<b>Crear película.-</b>	<b>14</b>
<b>Modificación y eliminado de una película.-</b>	<b>16</b>

# Vista principal de la página web.-



## Información sobre marvel.-



Para llegar hasta la información de Marvel, deberemos pinchar en el logo señalado en la anterior imagen. El código que se encargará de ello será el siguiente que se encuentra en el typescript del componente "**navbar**":

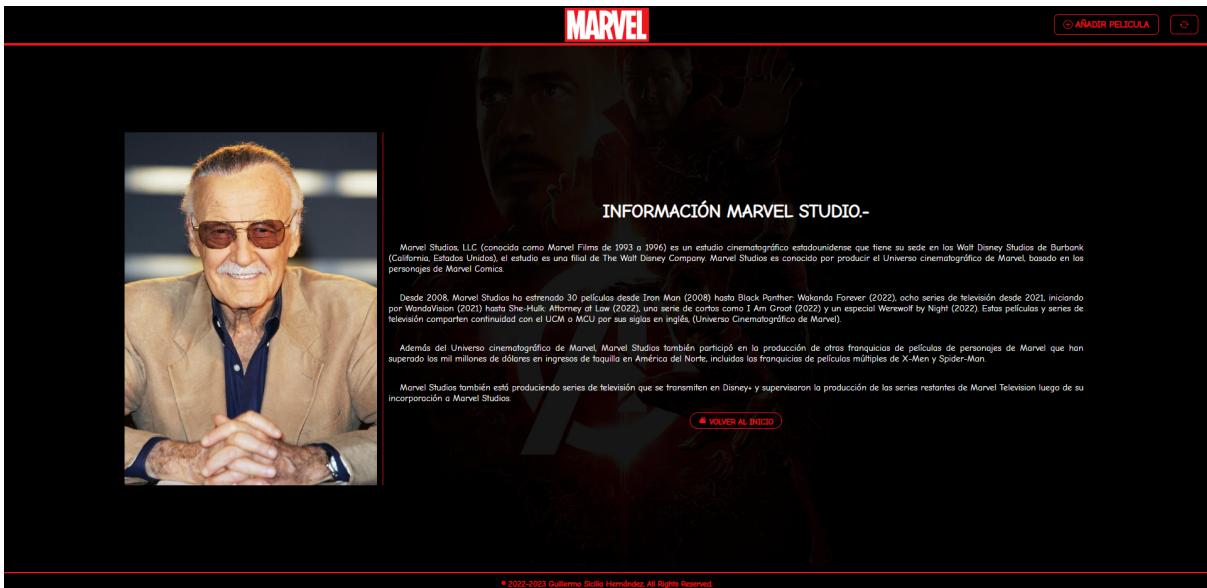
```
 1  /**
 2   * Si la ruta actual no es la ruta marvel-info, navegue hasta la ruta marvel-info. Si la ruta actual
 3   * es la ruta marvel-info, navegue a la ruta de inicio
 4  */
 5  cambiarRuta() {
 6    if (this.router.url != "/marvel-info") {
 7      this.router.navigate(['/marvel-info']);
 8    } else {
 9      this.router.navigate(['']);
10    }
11  }
```

```

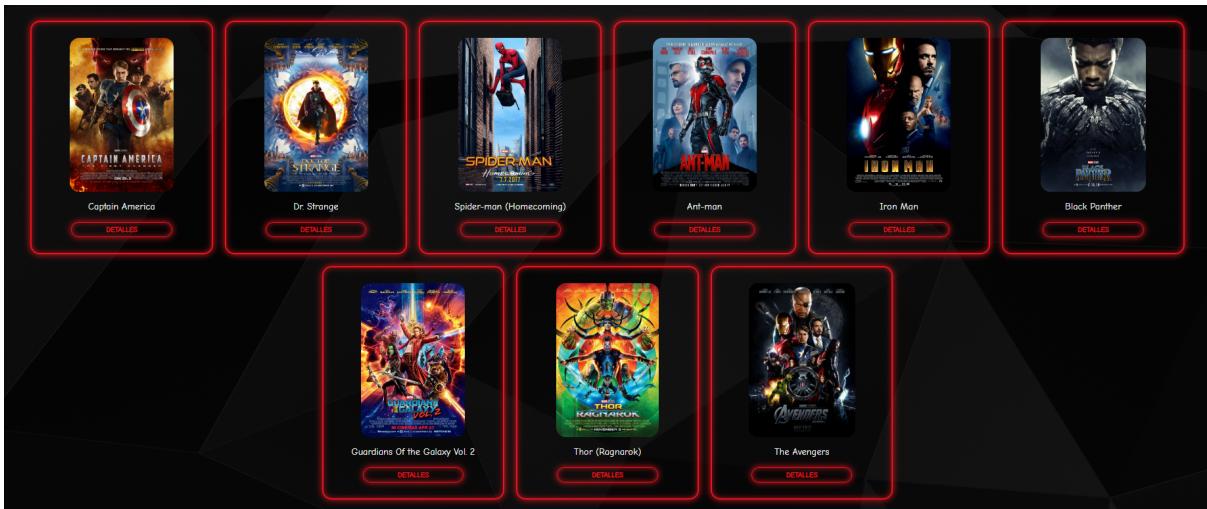
1 <div class="img-container">
2     <img src={{marvelStudioLogo}} alt="Marvel Studio Logo" (click)=cambiarRuta()/>
3 </div>

```

Y el resultado final es el siguiente al hacer **click** en el logo:



## Listado de películas.-



Para el listado de películas hay 3 métodos de obtenerlas dentro del servicio de “**api-peliculas.service.ts**”:

- Desde el **mock**:

```
1  /**
2   * Toma una matriz de objetos de Pelicula del Mock y devuelve una matriz de objetos de
3   * Pelicula.
4   * @returns Una matriz de objetos de Pelicula.
5   */
6  getPeliculasMock(): Pelicula[] {
7    let peliculas: Pelicula[] = [];
8    Mock.forEach(pelicula => {
9      peliculas.push(pelicula);
10   });
11
12   return peliculas;
13 }
```

- Desde la **api**:

```
1  /**
2   * Esta función devuelve un Observable de tipo Pelicula[]
3   * @returns Un Observable de Pelicula[]
4   */
5  getPeliculasApi(): Observable<Pelicula[]> {
6    return this.http.get<Pelicula[]>('https://www.qando.es/docs/films.php');
7 }
```

- Desde el **localStorage**:

```
1  /**
2   * Toma una matriz de objetos de Pelicula del storage y devuelve una matriz de objetos de
3   * Pelicula.
4   * @returns Una matriz de objetos de Pelicula o undefined si el localStorage está vacío.
5   */
6  getPeliculasStorage(): Pelicula[] | undefined {
7    let listaPelis = localStorage.getItem("listaPelis");
8    if (listaPelis != null) {
9      return JSON.parse(listaPelis);
10    } else {
11      return undefined;
12    }
13 }
```

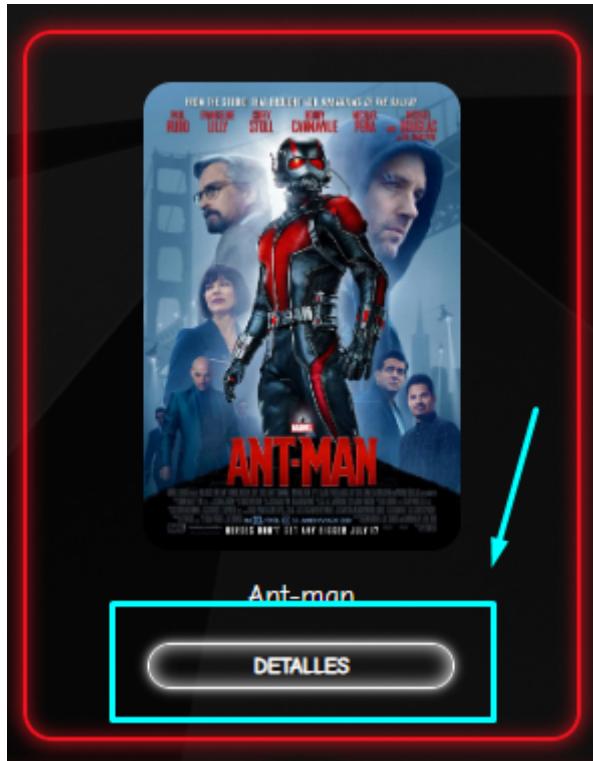
Y dependiendo de si usamos el **Mock** o el **localStorage**, los llamaremos en el componente “**lista-peliculas**” de las siguientes maneras. Si queremos trabajar con el **Mock** sería así:

```
1  /**
2   * Estamos llamando a getPeliculasMock() para inicializar la lista de películas y poder asignarla a la
3   * variable peliculas.
4   */
5  ngOnInit() {
6    this.peliculas = this.peliculasService.getPeliculasMock();
7 }
```

Si fuese con el **localStorage** sería de la siguiente manera dependiendo de si el localStorage está inicializado o no:

```
1  /**
2   * Si hay una lista de películas en el almacenamiento local, la obtenemos y la analizamos en un
3   * objeto JSON. Si no la hay, obtenemos la lista de películas de la API y la guardamos en el
4   * almacenamiento local.
5   */
6  getListapelis() {
7    let listaPelis = this.peliculasService.getPeliculasStorage();
8    if (listaPelis != undefined) {
9      this.peliculas = listaPelis;
10    } else {
11      this.peliculasService.getPeliculasApi().subscribe(peliculas => this.inicializarStorage(peliculas));
12    }
13  }
14
15 /**
16  * Toma una matriz de objetos de Pelicula, la convierte en una cadena y la almacena en localStorage.
17  *
18  * @param {Pelicula[]} peliculas - Película[]
19  */
20  inicializarStorage(peliculas: Pelicula[]) {
21    localStorage.setItem("listaPelis", JSON.stringify(peliculas));
22    this.peliculas = peliculas;
23  }
24
25 /**
26  * Estamos llamando a la función getListapelis() del archivo peliculasService.ts, que es una función
27  * que devuelve un observable. Luego nos suscribimos a ese observable y asignamos los datos que
28  * obtenemos a la variable peliculas.
29  */
30  ngOnInit() {
31    this.getListapelis();
32 }
```

## Detalles películas.-



Para visualizar los detalles de una película deberemos pinchar en el botón de detalles que hay en cada una de las “cartas” de las películas. Para obtener los detalles de cada peli, dependiendo de si queremos usar el mock, la api o el localStorage tendremos diferentes métodos en el fichero **“detalles-peliculas.service.ts”**:

- Desde el **mock**:

```
1  /**
2   * Devuelve los detalles de la pelicula de la pelicula con la identificación pasada en el parámetro.
3   * @param {string} id - de la pelicula
4   * @returns el objeto que está en el arreglo Mock, en la posición que se le pasa como parámetro.
5   */
6  getDetallesMock(id : string): Pelicula {
7    return Mock[Number(id) - 1];
8  }
9
```

- Desde la **api**:

```
1  /**
2   * Devuelve un Observable de tipo Pelicula
3   * @param {string} id - de la pelicula
4   * @returns Un Observable de tipo Pelicula.
5   */
6  getDetallesApi(id : string): Observable<Pelicula> {
7    return this.http.get<Pelicula>('https://www.qando.es/docs/films.php?id=' + id)
8 }
```

- Desde el **localStorage**:

```
1  /**
2   * Devuelve los detalles de la película de la película con la identificación pasada en el parámetro.
3   * @param {string} id - de la película
4   * @returns el objeto que está en el localStorage, con el id que se le pasa como parámetro.
5   */
6  getDetallesStorage(id: string): any {
7    let listaPelis = localStorage.getItem("listaPelis");
8
9    if (listaPelis != null) {
10      let jsonListaPelis: Pelicula[] = JSON.parse(listaPelis);
11
12      for (const pelicula of jsonListaPelis) {
13        if (pelicula.id == Number(id)) {
14          return pelicula;
15        }
16      }
17    }
18 }
```

Todos estos serán llamados en el componente “**detalles-peliculas**”. En el caso del mock sería:

```
1  /**
2   * La función obtiene los detalles de la película del servicio a partir del Mock y los asigna a la
3   * variable detallesPelícula.
4   */
5  ngOnInit() {
6    this.detallesPelícula = this.detallesService.getDetallesMock(this.router.url.replace("/", ""));
7 }
```

En el caso de leer desde el localStorage sería con este otro **ngOnInit**, pero dependería de si ya están definidas las variables **releaseDate** o **description** porque esto significaría que ya los detalles de la película han sido cargados desde la api:

```

1  /**
2   * Obtiene la lista de películas del almacenamiento local, obtiene la identificación de la película
3   * de la URL, verifica si la película tiene una fecha de lanzamiento y una descripción, si no la
4   * tiene, llama a la función getDetallesApi de DetallesService, si la tiene, llama a la función
5   * getDetallesStorage del DetallesService.
6   */
7  ngOnInit() {
8    let listaPelis = localStorage.getItem("listaPelis");
9    let idPeli = Number(this.router.url.replace("/", ""));
10
11   if (listaPelis != null) {
12     let jsonListaPelis: Pelicula[] = JSON.parse(listaPelis);
13
14     if (jsonListaPelis[idPeli - 1].releaseDate == null || jsonListaPelis[idPeli - 1].description == null) {
15       this.detallesService.getDetallesApi(idPeli + "").subscribe(pelicula => this.setDetallesPeli(jsonListaPelis, idPeli, pelicula));
16     } else {
17       this.detallesPelicula = this.detallesService.getDetallesStorage(idPeli + "");
18     }
19   }
20
21   this.leerMasVisibility();
22 }

```

Si no están definidas las variables se llama a **setDetallesPeli** introduciendo la película en la llamada cuando se hace el subscribe del observable que devuelve el **getDetallesApi**. Esto significa que, hasta que no se pinche en los detalles de cada película, el localStorage no tendrá los datos completos de la película sino los iniciales que nos da la lectura de la **Api** sin especificar el **id** de cada peli:

```

1 /**
2  * Toma una matriz JSON de películas, una ID y una película, y luego configura la película en la
3  * matriz JSON con la ID dada para la película dada.
4  *
5  * @param {Pelicula[]} jsonListaPelis - Pelicula[] es el conjunto de películas que obtenemos del
6  * localStorage.
7  * @param {number} id - número, película: película
8  * @param {Pelicula} pelicula - Pelicula es el objeto que contiene los nuevos datos que ha ingresado
9  * el usuario.
10 */
11 setDetallesPeli(jsonListaPelis: Pelicula[], id: number, pelicula: Pelicula) {
12   for (const peli of jsonListaPelis) {
13     if (peli.id == id) {
14       jsonListaPelis[id - 1] = pelicula;
15     }
16   }
17
18   this.detallesPelicula = jsonListaPelis[id - 1];
19
20   localStorage.setItem("listaPelis", JSON.stringify(jsonListaPelis));
21 }

```

Y el resultado sería el siguiente:



**Spider-man (Homecoming)**

**FECHA LANZAMIENTO:** 07/07/2017

**FASE:** 3

**SINOPSIS:** Spider-Man: Homecoming (titulada Spider-Man: De regreso a casa en Hispanoamérica) es una película de superhéroes estadounidense de 2017 basada en el personaje de Marvel Comics Spider-Man, coproducida por Columbia Pictures y Marvel Studios, y distribuida por Sony Pictures Releasing. Es el segundo re... Leer más.

## Pipe títulos películas.-



**Spider-man (Homecoming)**

**FECHA LANZAMIENTO:** 07/07/2017

**FASE:** 3

**SINOPSIS:** Spider-Man: Homecoming (titulada Spider-Man: De regreso a casa en Hispanoamérica) es una película de superhéroes estadounidense de 2017 basada en el personaje de Marvel Comics Spider-Man, coproducida por Columbia Pictures y Marvel Studios, y distribuida por Sony Pictures Releasing. Es el segundo re... Leer más.

Para los títulos que contengan ":" se ha creado un pipe que modifica dicho título y pone lo que vaya detrás del ":" entre paréntesis:

```
1  /**
2   * Si el nombre contiene dos puntos, divide el nombre en dos partes, la primera parte es el nombre,
3   * la segunda parte es el tipo y luego devuelve el nombre y el tipo entre paréntesis.
4   * @param {string} name - El nombre del elemento que se va a transformar.
5   * @returns El nombre del artículo.
6  */
7  transform(name: string): string {
8    let modName: string = name;
9    if (modName.includes(":")) {
10      name = modName.split(":")[0] + " (" + modName.split(":")[1].replaceAll(" ", "") + ")";
11    }
12    return name;
13 }
```

## Pipe acortamiento de la sinopsis.-

**SINOPSIS:** Spider-Man: Homecoming (titulada Spider-Man: De regreso a casa en Hispanoamérica) es una película de superhéroes estadounidense de 2017 basada en el personaje de Marvel Comics Spider-Man, coproducida por Columbia Pictures y Marvel Studios, y distribuida por Sony Pictures Releasing. Es el segundo re... [Leer más.](#)

Para mostrar que la sinopsis de cada película no salga completa se ha usado otra pipe que la recorta a 300 caracteres y añade tres puntos al final de ella:

```
1  /**
2   * La función toma una cadena como argumento y, si la cadena tiene más de 300 caracteres, devuelve
3   * los primeros 299 caracteres de la cadena, seguidos de puntos suspensivos.
4   * @param {string} description - string - La descripción del evento.
5   * @returns Una cuerda
6  */
7  transform(description: string): string {
8    if (description.length > 300) {
9      description = description.substring(0, 299) + "...";
10    }
11    return description;
12 }
```

Además, si el texto es mayor a 300 caracteres, se muestra un link de “Leer Más” para mostrar el texto al completo. Este código se encuentra en el componente de “detalles-peliculas”:

```
● ● ●
1  /**
2   * Toma el elemento con el id "sinopsis" y cambia su HTML interno a la descripción de la película
3   * para mostrar la sinopsis completa de esta.
4   */
5  expandirTexto() {
6      let sinopsis = (<HTMLElement>)document.getElementById("sinopsis");
7      sinopsis.innerHTML = "<strong style='color: #f0131f; text-transform: uppercase;'>Sinopsis: </strong>" + this.detallesPelicula.description;
8  }
```

```
● ● ●
1  <div class="right-container centered">
2      <h1>{{detallesPelicula.name | textTransform}}</h1>
3      <p><strong>Fecha lanzamiento: </strong>{{detallesPelicula.releaseDate}}</p>
4      <p [releaseDate]="detallesPelicula.releaseDate" appPhaseCalculator></p>
5      <p id="sinopsis">
6          <strong>Sinopsis: </strong>{{detallesPelicula.description | shortenText}}
7          <a id="leermas" class="linkTexto" (click)=expandirTexto()>Leer más.</a>
8      </p>
9  </div>
```

# Directiva fase marvel.-

## Spider-man (Homecoming)

FECHA LANZAMIENTO: 07/07/2017

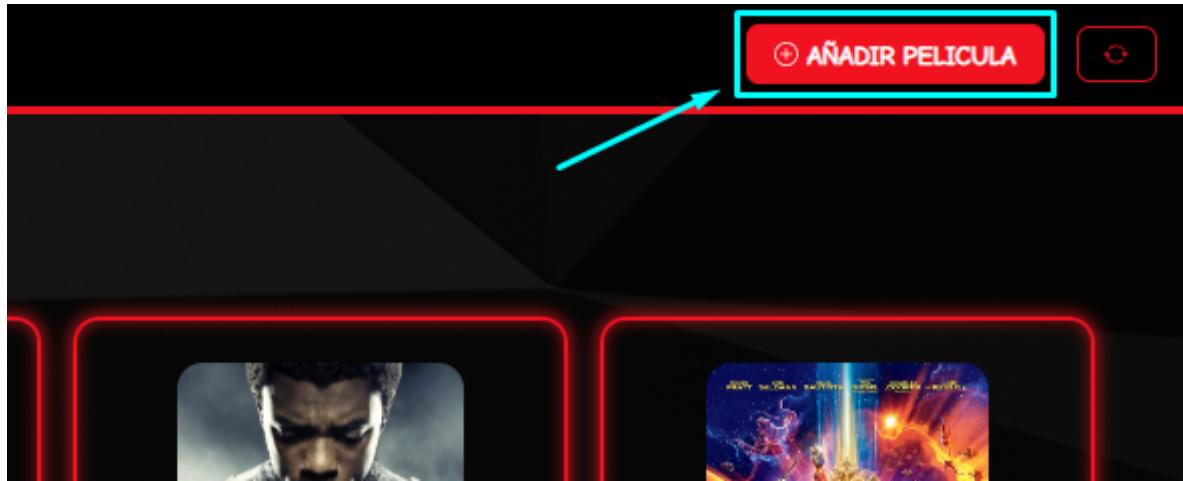
FASE: 3

**SINOPSIS:** Spider-Man: Homecoming (titulada Spider-Man: De regreso a casa en Hispanoamérica) es una película de superhéroes estadounidense de 2017 basada en el personaje de Marvel Comics Spider-Man, coproducida por Columbia Pictures y Marvel Studios, y distribuida por Sony Pictures Releasing. Es el segundo re... Leer más.

Para mostrar la fase de cada una de las películas en los detalles, usaremos la directiva “**phase-calculator**”, que evaluará dependiendo de la fecha la fase MCU de la película:

```
1  /**
2   * Usamos el gancho del ciclo de vida `ngOnInit()` para obtener la fecha de lanzamiento de la
3   * película y luego usamos esa fecha para determinar a qué fase de la MCU pertenece la película.
4   */
5  ngOnInit() {
6    let phase: string;
7    let anio = Number(this.releaseDate.split("/")[2]);
8    let mes = Number(this.releaseDate.split("/")[1]);
9
10   if (anio >= 2008 && anio <= 2012) {
11     phase = "1";
12   } else if (anio >= 2013 && anio <= 2015) {
13     phase = "2";
14   } else if (anio >= 2016 && anio <= 2019) {
15     phase = "3";
16   } else if (anio >= 2021 && anio <= 2022) {
17     phase = "4";
18   } else if (anio >= 2023 && anio <= 2024) {
19     if (mes < 11) {
20       phase = "5";
21     } else {
22       phase = "6";
23     }
24   } else if (anio >= 2025 && anio <= 2026) {
25     phase = "6";
26   } else {
27     phase = "N/A";
28   }
29
30   this.elementTag.nativeElement.innerHTML = "<strong style='color: #f0131f'>FASE: </strong> " + phase;
31 }
```

## Crear película.-



Para añadir una película, hemos usado los siguientes métodos dependiendo de si queremos crearla en el mock o en el localStorage:

- Desde el **mock**:

```
1  /**
2   * Toma los valores del formulario y los inserta en el mock
3   */
4  registrarPeliculaMock() {
5    let name = (<HTMLInputElement>document.getElementById("name")).value;
6    let poster = (<HTMLInputElement>document.getElementById("poster")).value;
7    let releaseDate = this.formatDate((<HTMLInputElement>document.getElementById("releaseDate")).value);
8    let description = (<HTMLInputElement>document.getElementById("description")).value;
9    let id = Mock[Mock.length + 1].id + 1;
10
11   Mock.push(new Pelicula(id, name, poster, releaseDate, description));
12   this.router.navigate(['']);
13 }
```

- Desde el **localStorage**:

```

1  /**
2   * Toma los valores del formulario y crea un nuevo objeto Pelicula, luego lo inserta en la matriz
3   * listaPelis y lo guarda en localStorage.
4  */
5  registrarPeliculaStorage() {
6    let name = (<HTMLInputElement>document.getElementById("name")).value;
7    let poster = (<HTMLInputElement>document.getElementById("poster")).value;
8    let releaseDate = this.formatDate((<HTMLInputElement>document.getElementById("releaseDate")).value);
9    let description = (<HTMLInputElement>document.getElementById("description")).value;
10
11   let listaPelis = localStorage.getItem("listaPelis");
12
13   if (listaPelis != null) {
14     let jsonListaPelis = JSON.parse(listaPelis);
15     let id = jsonListaPelis[jsonListaPelis.length - 1].id + 1;
16     jsonListaPelis.push(new Pelicula(id, name, poster, releaseDate, description));
17     localStorage.setItem("listaPelis", JSON.stringify(jsonListaPelis));
18     this.router.navigate(['']);
19   }
20 }

```

Ambos métodos usan una función llamada **formatDate**, porque el input de tipo date usa el formato americano **AAAA-MM-DD** cuando nosotros estamos trabajando con el formato **DD/MM/AAAA**:

```

1  /**
2   * Toma una fecha en el formato AAAA-MM-DD y la devuelve en el formato DD/MM/AAAA.
3   * @param {string} date - La fecha a formatear.
4   * @returns La fecha en el formato dd/mm/aaaa.
5  */
6  private formatDate(date: string): string {
7    let dia = date.split("-")[2];
8    let mes = date.split("-")[1];
9    let anio = date.split("-")[0];
10
11   return dia + "/" + mes + "/" + anio;
12 }

```

Y el resultado sería el siguiente:



The movie details page for "SpeterMan" displays the following information:

- Title: SpeterMan
- Poster: speterman.png
- Release Date: 23/02/2023
- Phase: 5
- Synopsis: Speterman la película

At the bottom, there are buttons for "ELIMINAR", "MODIFICAR", and "VOLVER AL INICIO". A copyright notice at the very bottom reads: "© 2022-2023 Guillermo Sicilia Hernández. All Rights Reserved."

## Modificación y eliminado de una película.-

En los detalles de cada película nos encontraremos los botones de modificación y eliminado de esa película:

The movie details page for "SpeterMan" shows the same information as before. A cyan arrow points to the "MODIFICAR" button, which is highlighted with a cyan border.

Para modificar una película usaremos los métodos en el componente "**editar-pelicula**". Lo primero será introducir los datos de dicha película en el formulario:

- Desde el mock:

```
● ● ●
1 /**
2  * Usamos el enrutador angular para obtener la URL de la página actual, la dividimos en una matriz de
3  * cadenas y luego usamos el segundo elemento de esa matriz para obtener la ID de la película que
4  * queremos mostrar a partir del Mock.
5 */
6 ngOnInit() {
7   this.detallesPelicula = this.peliculasService.getDetallesMock(this.router.url.split("/")[2]);
8 }
```

- Desde el localStorage:

```
● ● ●
1 /**
2  * Usamos el enrutador angular para obtener la URL de la página actual, la dividimos en una matriz de
3  * cadenas y luego usamos el segundo elemento de esa matriz para obtener la ID de la película que
4  * queremos mostrar a partir del localStorage.
5 */
6 ngOnInit() {
7   this.detallesPelicula = this.peliculasService.getDetallesStorage(this.router.url.split("/")[2]);
8 }
```

En este caso, para introducir la fecha dentro del formulario, usaremos el pipe “**format-date**” para darle formato:

```
● ● ●
1 /**
2  * Toma una cadena en el formato "dd/mm/yyyy" y devuelve una cadena en el formato "yyyy-mm-dd"
3  *
4  * @param {string} date - cadena: la fecha que se va a transformar.
5  * @returns La fecha en el formato AAAA-MM-DD
6 */
7 transform(date: string): string {
8   let dia = date.split("/")[0];
9   let mes = date.split("/")[1];
10  let anio = date.split("/")[2];
11
12  return [anio, mes, dia].join("-");
13 }
```

```
● ● ●
1 <form (submit)=modificarPeliculaApi() class="form">
2   <label for="name">Nombre:</label>
3   <input type="text" id="name" value="{{detallesPelicula.name}}> required>
4   <label for="poster">Poster:</label>
5   <input type="text" id="poster" value="{{detallesPelicula.poster}}> required>
6   <label for="releaseDate">Fecha lanzamiento:</label>
7   <input type="date" id="releaseDate" value="{{detallesPelicula.releaseDate | formatDate}}> required>
```

Y luego habrá que introducir dichos datos en cada método de almacenamiento ya sea **mock** o **localStorage**:

```
1 /**
2  * Toma los valores de las entradas y crea un nuevo objeto Pelicula con ellos, luego reemplaza el
3  * antiguo objeto Pelicula en la matriz Mock con el nuevo.
4 */
5 modificarPeliculaMock() {
6     let name = (<HTMLInputElement>document.getElementById("name")).value;
7     let poster = (<HTMLInputElement>document.getElementById("poster")).value;
8     let releaseDate = this.formatDate((<HTMLInputElement>document.getElementById("releaseDate")).value);
9     let description = (<HTMLInputElement>document.getElementById("description")).value;
10    Mock[Number(this.detallesPelicula.id) - 1] = new Pelicula(this.detallesPelicula.id, name, poster, releaseDate, description);
11    this.router.navigate(['/ + this.detallesPelicula.id]);
12 }
```

```
1 /**
2  * Toma los valores de las entradas y crea un nuevo objeto Pelicula con ellos, luego reemplaza el
3  * antiguo objeto Pelicula en el localStorage con el nuevo.
4 */
5 modificarPeliculaApi() {
6     let name = (<HTMLInputElement>document.getElementById("name")).value;
7     let poster = (<HTMLInputElement>document.getElementById("poster")).value;
8     let releaseDate = this.formatDate((<HTMLInputElement>document.getElementById("releaseDate")).value);
9     let description = (<HTMLInputElement>document.getElementById("description")).value;
10
11    let listaPelis = localStorage.getItem("listaPelis");
12    if (listaPelis != null) {
13        let jsonListaPelis: Pelicula[] = JSON.parse(listaPelis);
14        for (let i = 0; i < jsonListaPelis.length; i++) {
15            if (jsonListaPelis[i].id == this.detallesPelicula.id) {
16                jsonListaPelis[i] = new Pelicula(i + 1, name, poster, releaseDate, description);
17            }
18        }
19        localStorage.setItem("listaPelis", JSON.stringify(jsonListaPelis));
20        this.router.navigate(['']);
21    }
22
23    this.router.navigate(['/ + this.detallesPelicula.id]);
24 }
```

Y para guardar la **releaseDate**, usaremos de nuevo el método **formatDate** para ajustar el formato al que estamos usando para guardar los datos:

```
1 /**
2  * Toma una fecha en el formato AAAA-MM-DD y la devuelve en el formato DD/MM/AAAA.
3  * @param {string} date - La fecha a formatear.
4  * @returns La fecha en el formato dd/mm/aaaa.
5 */
6 private formatDate(date: string): string {
7     let dia = date.split("-")[2];
8     let mes = date.split("-")[1];
9     let anio = date.split("-")[0];
10
11    return dia + "/" + mes + "/" + anio;
12 }
```

Y todo esto daría el siguiente formulario cumplimentado con los datos de dicha película:

The screenshot shows a modal dialog titled "Modificar Spider-man: Homecoming.-". The form fields are as follows:

- Nombre:** Spider-man: Homecoming
- Poster:** <https://i.pinimg.com/originals/3d/92/7f/3d927fa36356cb300fd16b5c1ac2e>
- Fecha lanzamiento:** 07/07/2017
- Sinopsis:** Spider-Man: Homecoming (título original Spider-Man: De regreso a casa en Hispanoamérica) es una película de superhéroes estadounidense de 2017 basada en el personaje de Marvel Comics Spider-Man, coproducida por Columbia Pictures y Marvel Studios, y distribuida por Sony Pictures Releasing. Es el segundo reinicio cinematográfico de Spider-Man y la decimosexta película del Universo cinematográfico de Marvel (MCU). La película es dirigida por Jon Watts, a partir de un guion de los equipos de Jonathan Goldstein y John Francis Daley, Watts y Christopher Ford, y Chris McKenna y Erik Sommers, y es protagonizada por Tom Holland como el personaje principal, junto a Michael Keaton, Jon Favreau, Laura Harrier, Zendaya, Donald Glover,...

At the bottom of the form are two buttons: "MODIFICAR" (Modify) and "VOLVER AL INICIO" (Return to Home).

Luego, para eliminar una película usaremos las dos funciones que se encuentran en el componente **“detalles-pelicula”** dependiendo de si queremos eliminarlas en el **mock** o en el **localStorage**:

- Desde el **mock**:

The code editor displays a component named "detailsPelicula". At the top, there are three circular icons: red, yellow, and green. Below them is the following code:

```
1  /**
2   * Elimina la pelicula de la lista de películas del Mock.
3   */
4  eliminarPeliculaMock() {
5      Mock.splice(this.detallesPelicula.id - 1);
6      this.router.navigate(['']);
7  }
```

- Desde el **localStorage**:

```
1  /**
2   * Obtiene la lista de películas del almacenamiento local, las recorre y, si la identificación de la
3   * película coincide con la identificación de la película que estamos viendo, la elimina de la lista.
4   */
5  eliminarPeliculaApi() {
6    let listaPelis = localStorage.getItem("listaPelis");
7    if (listaPelis != null) {
8      let jsonListaPelis: Pelicula[] = JSON.parse(listaPelis);
9      for (let i = 0; i < jsonListaPelis.length; i++) {
10        if (jsonListaPelis[i].id == this.detallesPelicula.id) {
11          jsonListaPelis.splice(i, 1);
12        }
13      }
14      localStorage.setItem("listaPelis", JSON.stringify(jsonListaPelis));
15      this.router.navigate(['']);
16    }
17 }
```