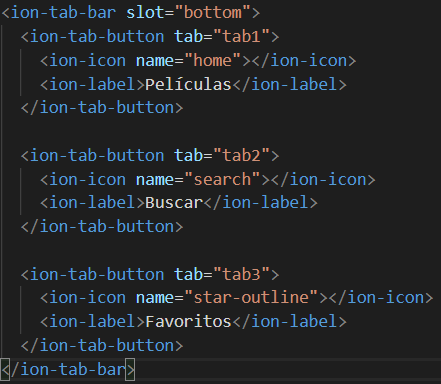
PeliculasApp  
Creando el poyecto  


Ionic serve

Primero modificaremos el nombre de los tabs

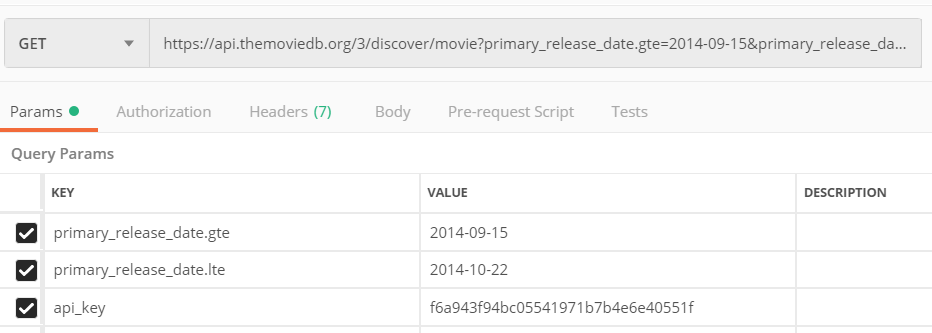


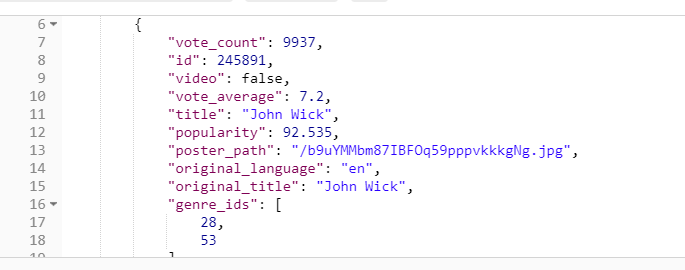
Creando un API para tomar información de la aplicación.

<https://www.themoviedb.org/>

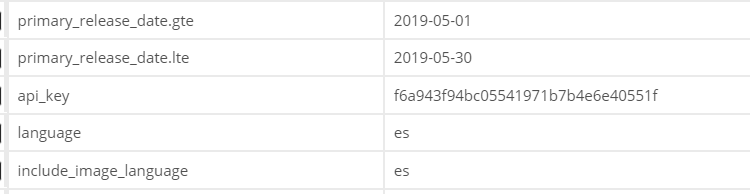
Luego crearemos el servicio para obtener películas de estreno.

En la pagina del api encontramos ejemplo de los endpoints asi que probaremos con postman.



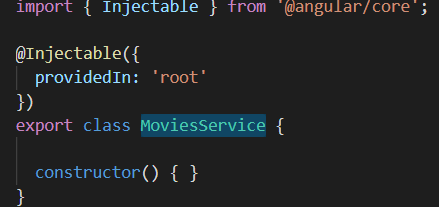


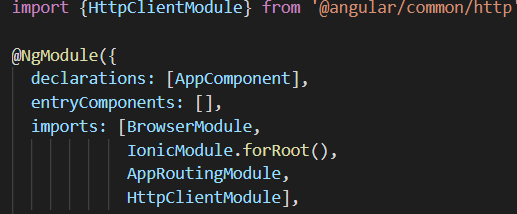
El api nos permite hacer las peticiones en diferentes idiomas, inclusive para las imágenes.



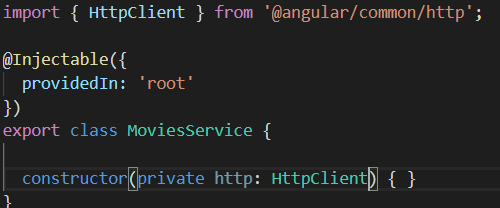
Creando el servicio  


La siguiente estructura nos muestra que no hay que importarlo en otro lugar ya que esta proveido en el root.

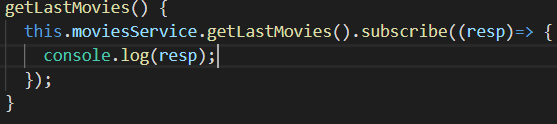


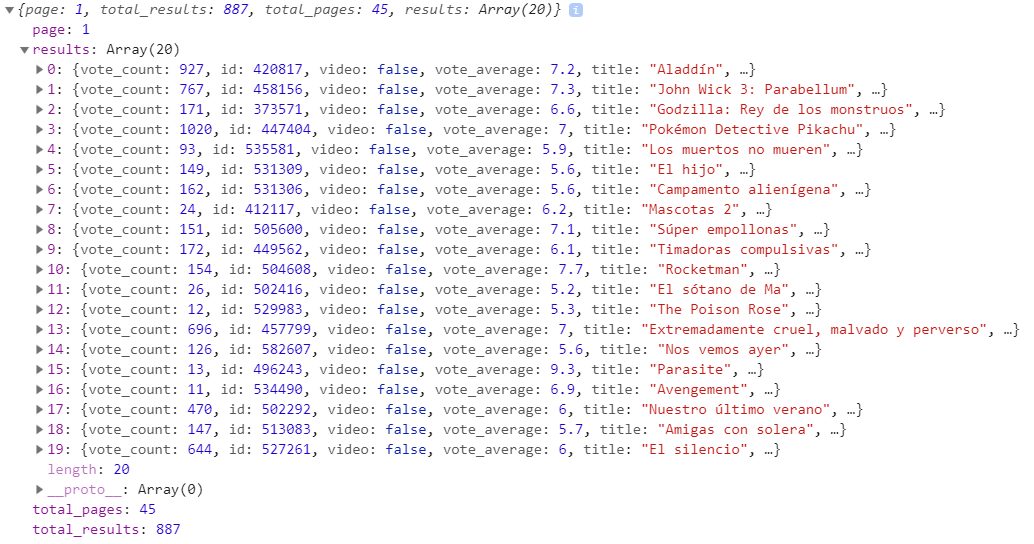
Ahora hay que importar en el app module el **HttpClientModule**  


Ahora en el service importamos el **HttpClient**



Creamos la función para obtener las películas y lo implementamos desde el tab1

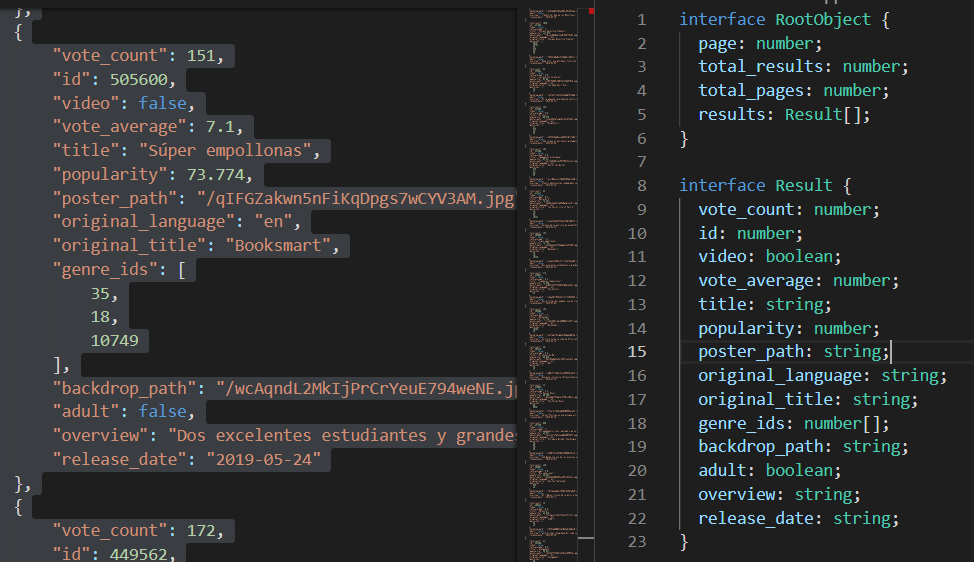




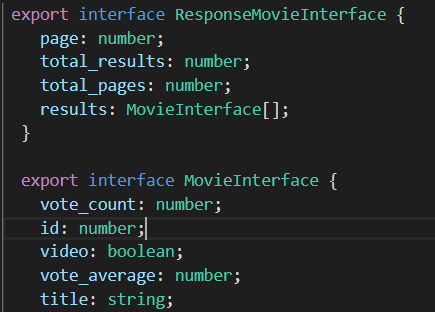
Crearemos ahora la interfaz y respuesta del servicio

Creamos una interfaz en src/app/interfaces/interface.ts

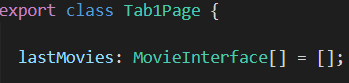
Y ahí pegamos la respuesta de postman, luego con control +shift + p buscamos el convertidor a json from selection.

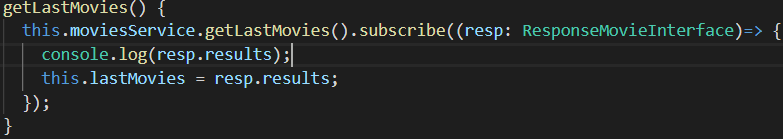


Y renombramos los objetos

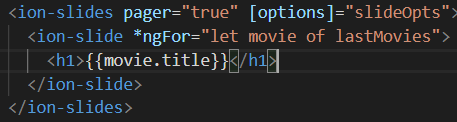


Ahora que esta tipada nuestra respuesta podemos hacerlo desde la función del tab.





Mostrar Películas  
Pipe.module y pipe imagen



Para mostrar la imagen nos referimos a la documentación del api para conseguir el url completo ya que en la respuesta del servicio tenemos un url incompleto



<https://developers.themoviedb.org/3>



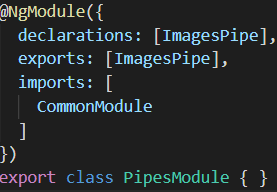
Para importar las imágenes utilizaremos un pipe que una el path de la imagen con el restante del url y asi poder usar el mismo código en todas las pantallas.

Creamos el modulo de pipes

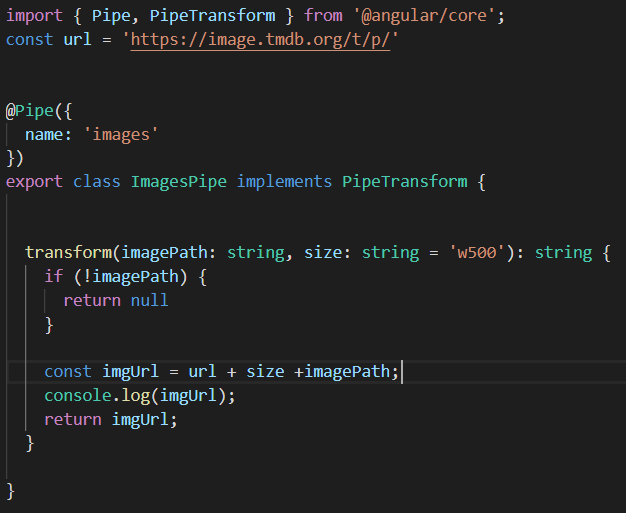


Y dentro de la carpeta que se genera creamos el pipe  


Luego dentro del modulo de pipes nos aseguramos que el images este en los declarations y el exports



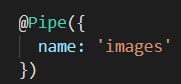
Y nuestro pipe como el tamaño puede variar lo ponemos como parámetro a recibir, pero si no llega por que por lo general es el mismo, lo inicializamos como w500 y lo agregamos al path ya que forma parte de la estructura de la imagen del api.

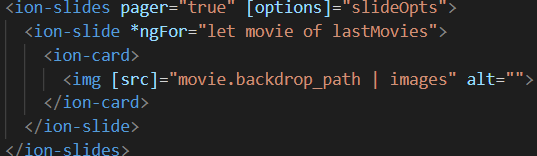


Luego en el modulo de la pagina donde vamos a utilizarlo, importamos el PipesModule.

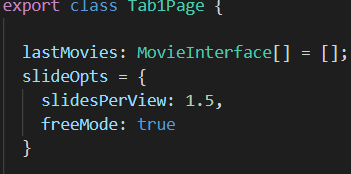
En este caso en el tabs1.module en los imports.

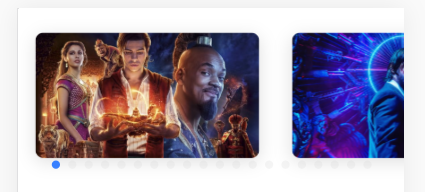
Recordando el nombre del pipe lo implementamos en el html



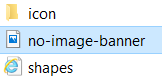


En nuestro html tenemos las options y con esto podemos modificar la forma que se ve el slide creando el objeto y añadiéndole las propiedades.



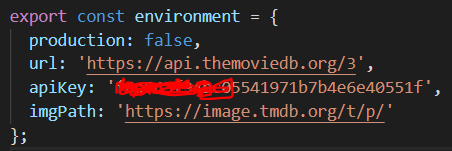


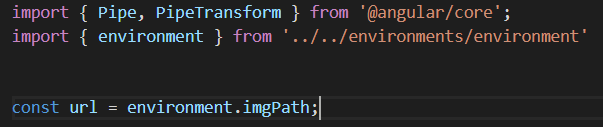
Puede una película no tener imagen o fallar en la respuesta.  
Por defecto en los assets podemos seleccionar una.



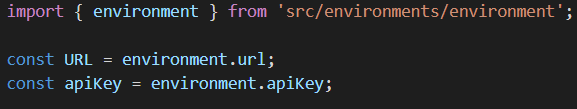
Y la devolvemos desde el pipe  

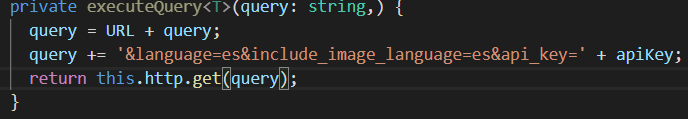

Variables globales de la aplicación

Desde el enviroment creamos las variables al igual que en el de prod  


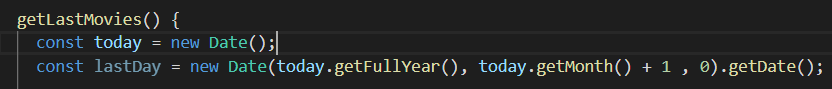
Y reemplazamos las variables, desde nuestro pipe  


Y creamos una función que centralice los querys que se harán de petición para manejar las constantes en una sola función.





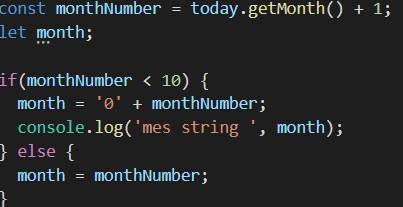
Necesitamos optimizar las fechas para los query, para esto utilizaremos el date

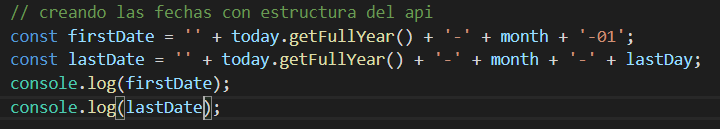
Para last day los parámetros que se pasan son el año actual, el mes siguiente y el dia cero (Al pedir el dia cero del siguiente mes nos manda el dia anterior ósea el ultimo dia del mes actual…Weird).  


Como la fecha que debemos construir es la siguiente vamos a manipular lo que obtenemos y construirlo.



Construimos el mes para que si esmenor que 10 tenga la estructura de ‘01’









Componente de slideshow

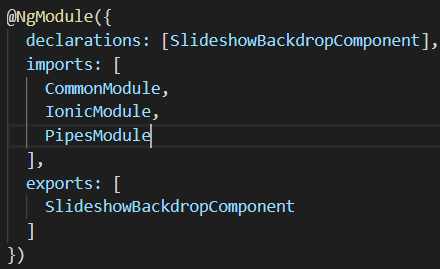
Vamos a componentizar los slides para asi poder reutilizarlo, llamando al servicio igual desde el tab 1 y pasandole como parámetro el arreglo recibido para que este componente se encarge de hacer un slideshow con esto.

Creamos un modulo de componentes para que este pueda ser reutilizado.

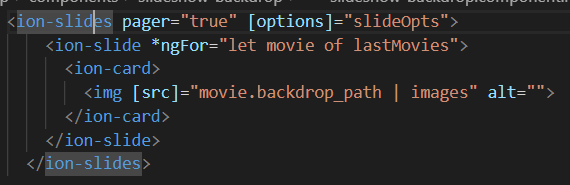




El componente creado es necesario agregarlo a los declarations y los exports del modulo.  
Tambien es necesario importar los modulos del pile y el ionicModule ya que utilizaremos la estructura de ionic para generar el slide en el html.



En el html del slideshow pegamos el código anterior



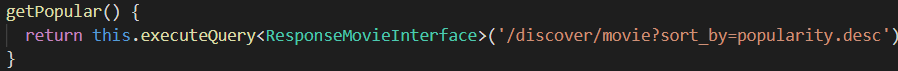
Y desde el tab 1 lo llamamos pasandole el arreglo que ya recibimos.



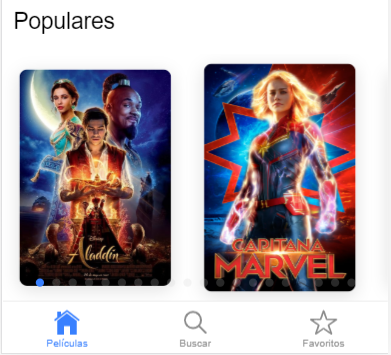
Ahora vamos a repetir el proceso pero con los posters.

Mostrar películas populares.

Llamando al mismo servicio con otro key para traer las populares.



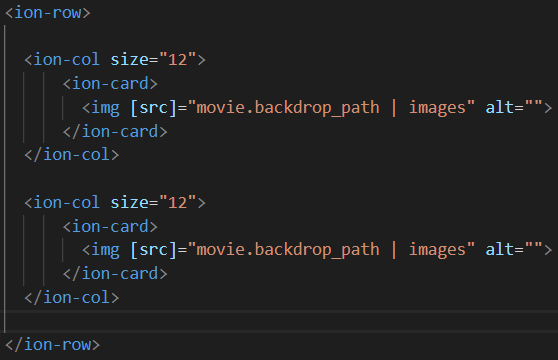


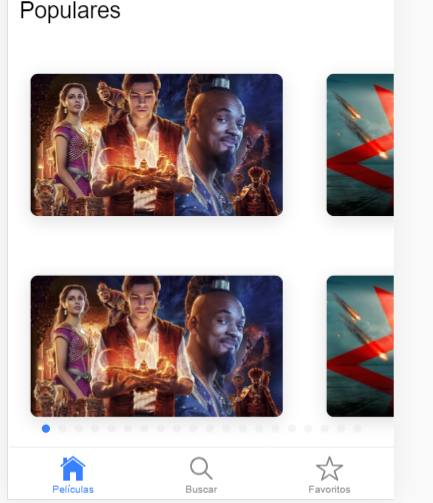


Mostrando el slide en pares  
Creamos otro componente que tendrá el código paramostrarlas en pares.



Lo agregamos al modulo de componentes. En el html creamos la estructura con un row.

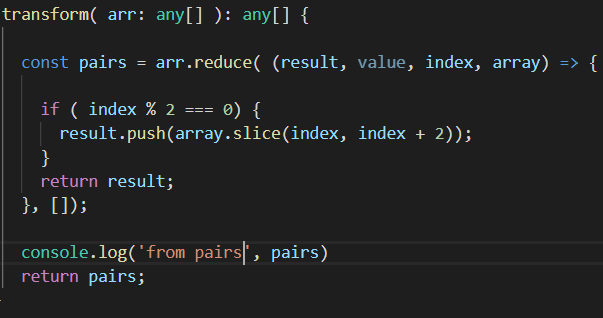


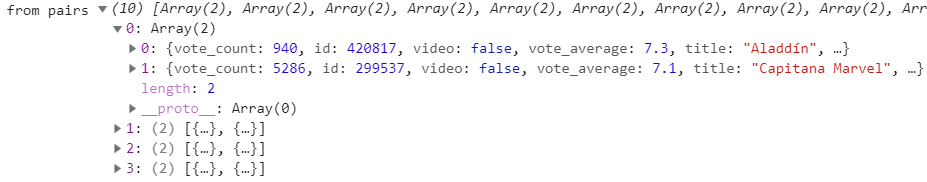


Ahora para alternar las películas vamos a crear un pipe  

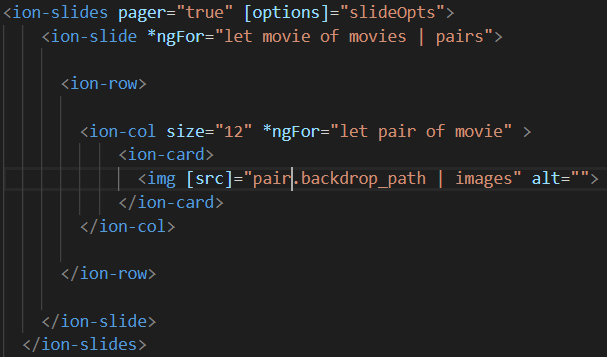

Agregarlo a los exports del pipe.module

La opción reduce en un arreglo permite hacer división de este.

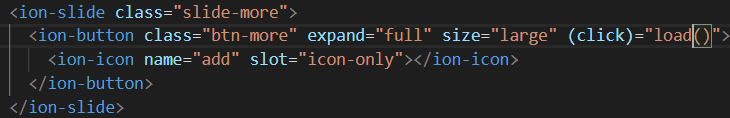




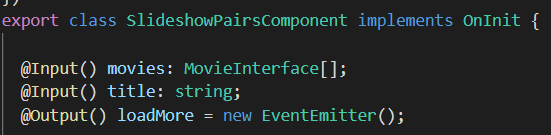
Y en el html implementamos el pipe y dentro en el col iteramos el arreglo de 2 que retorno y mostramos las películas.

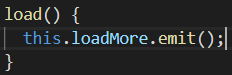


Cargar más películas horizontalmente



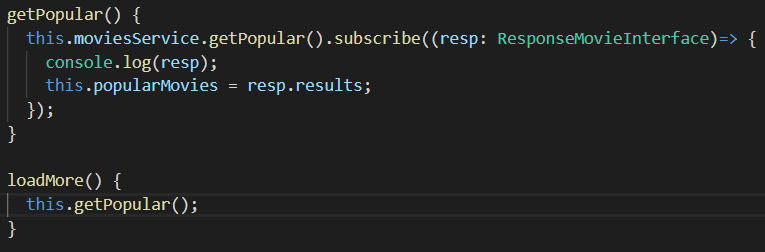
Utilizaremos un output para cargar más, ya que el tab1 es el que llama al servicio.





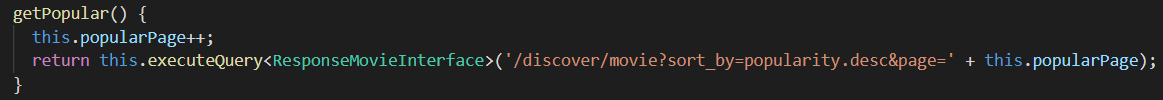
En el tab1.html

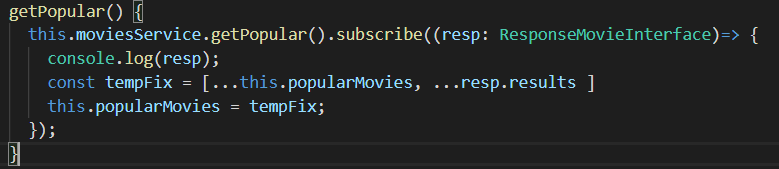




Ahora debemos cambiar la lógica del getPopular desde el servicio y desde el componente page 1.

En el servicio

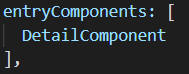


Y en el componente local tab1 creamos un arreglo temporal que va a concatenar el arreglo actual con el arreglo de la respuesta y finalmente asignarlo al arreglo del componente para mostrarlo.  


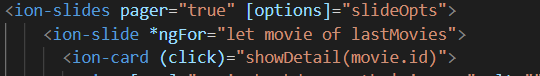
Modal para detalles de las películas

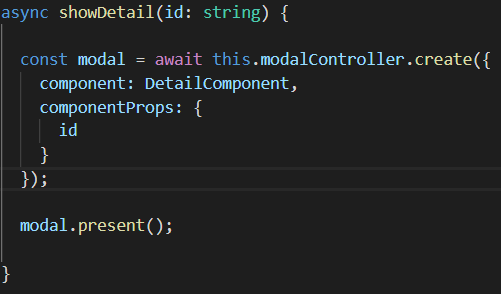


Agregarlo al modulo de componentes en los declarations, export y en **entryComponent**  ya que este será llamado como un modal y es requerida esta importación.

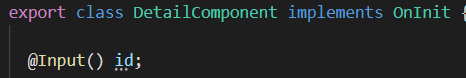


Desde el componente de los slides al hacer click en un card, se acciona la función pasandole el id de la película.





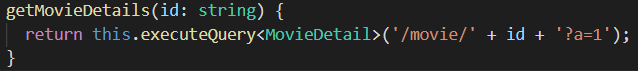
Y desde el detail component recibimos el id



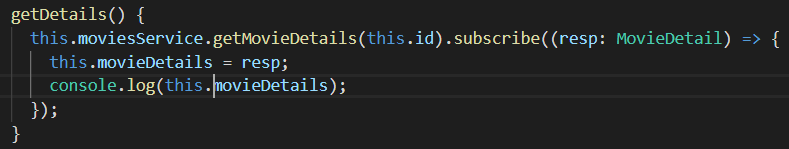
Consultando otro endpoint para ver detalles de la película



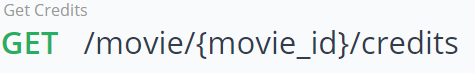
Utilizamos un ejemplo primero con postman para poder mapear el objeto de respuesta y crear la interfaz en su archivo correspondiente. En este caso como el único parámetro que se manda debería ser el apikey, no funcionará ya que en este momento lo estamos concatenando con &, entonces pasamos un parámetro falso paraque pueda funcionar.



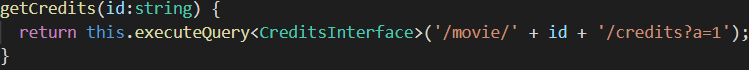
Y en el get details recuperamos los datos del servicio



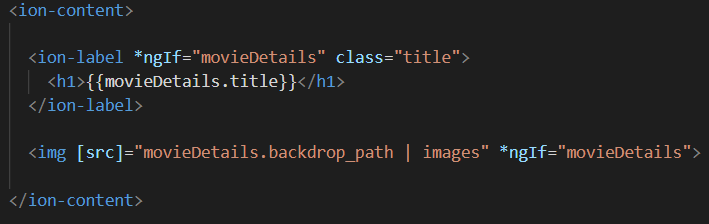
Ahora también debemos recuperar los créditos para mostrar los actores de la película:

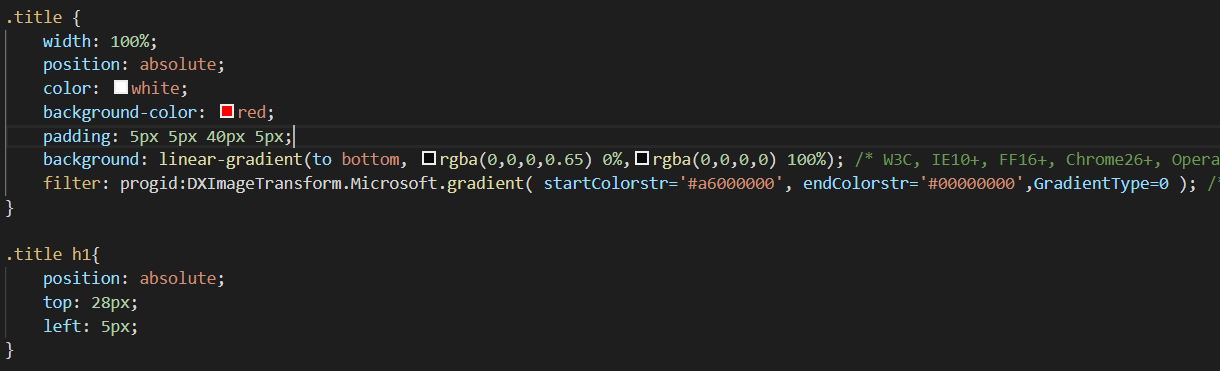


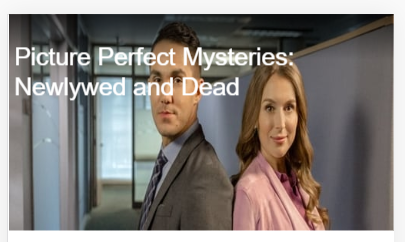
Básicamente igual



Html del detalle

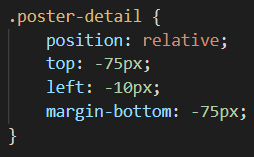






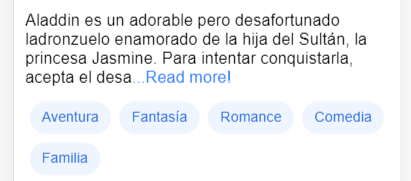
Luego agregamos el poster y las filas de votos y rating.



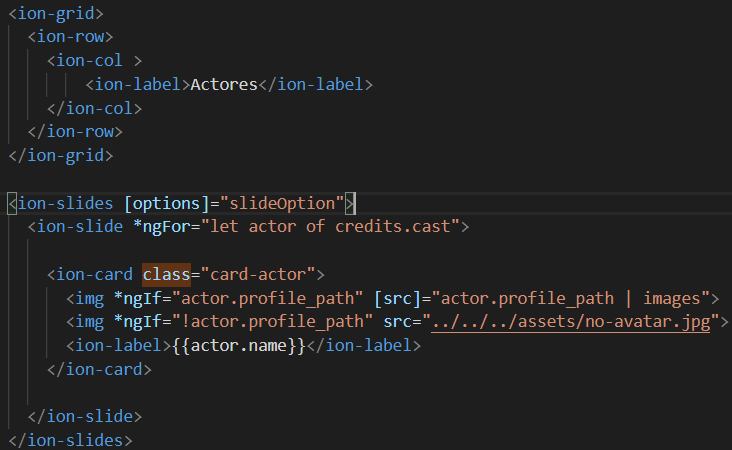


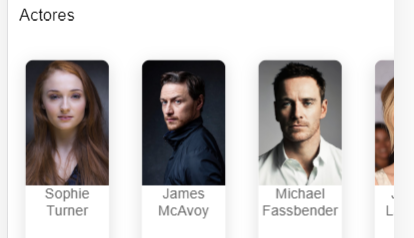
Para el resumen de la película



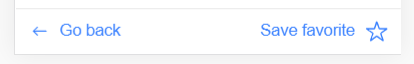


Ahora añadimos los slides de los actores

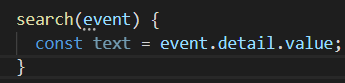
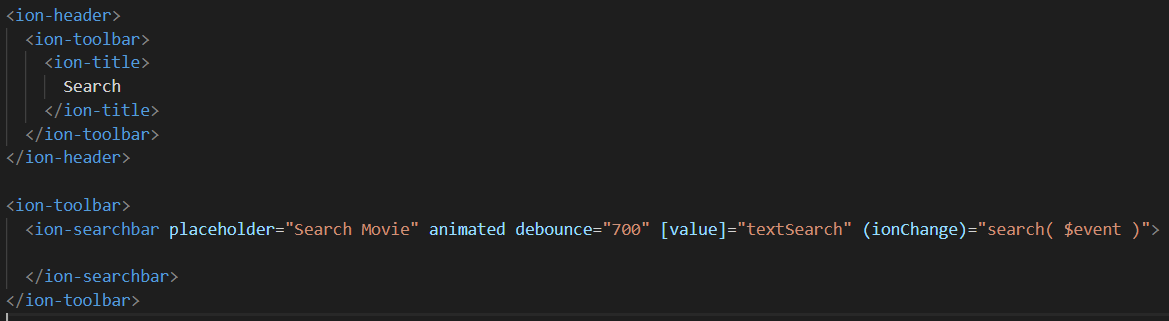


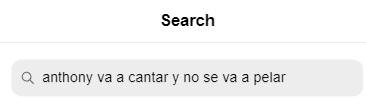


Y el footer   

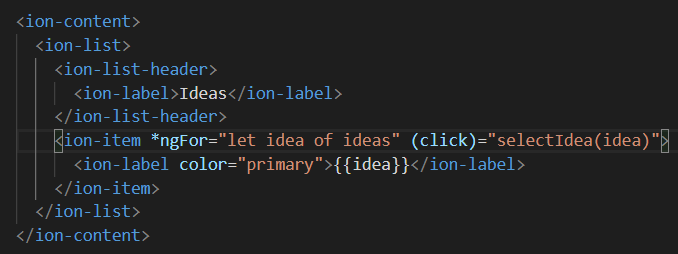
Pagina de búsqueda



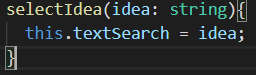


Hacemos una lista de ideas rápidas para que el usuario seleccione, en este paso de ejemplo un arreglo predefinido.

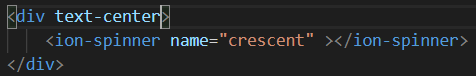


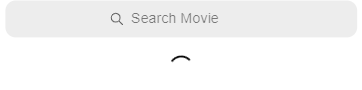


Y cuando el usuario haga click en una de ellas se llenara el campo de búsqueda directamente.



Y agregamos un spinner justo después del searchbar

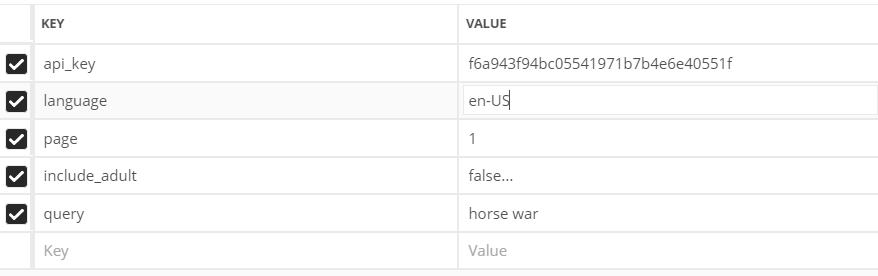


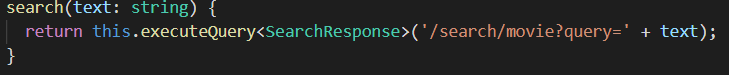


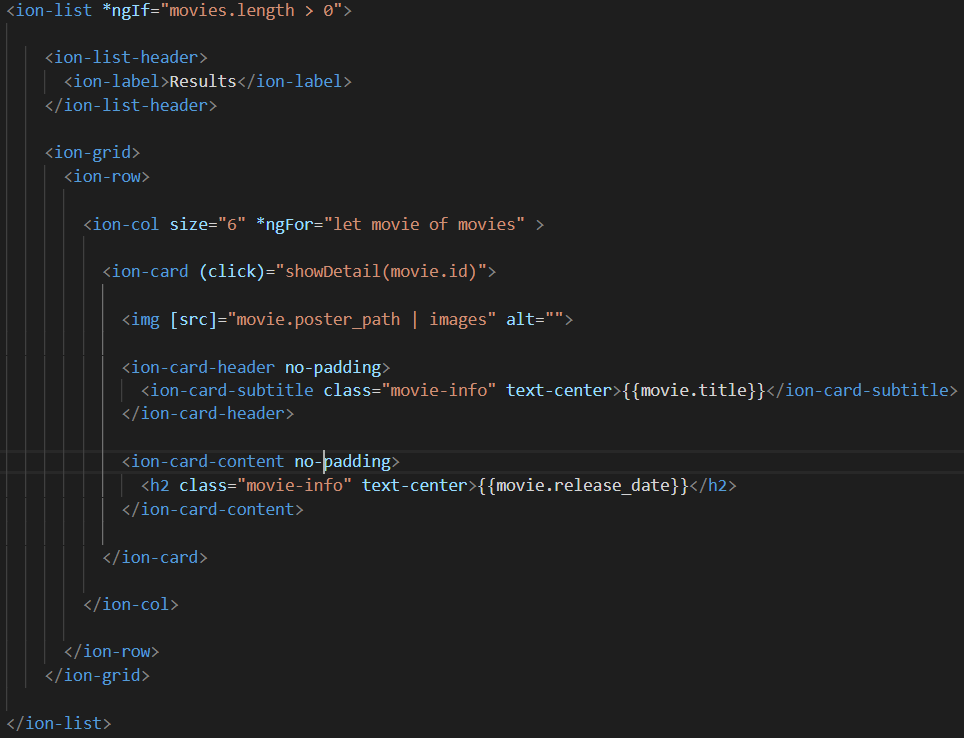
Agregando el servicio de búsqueda de películas.

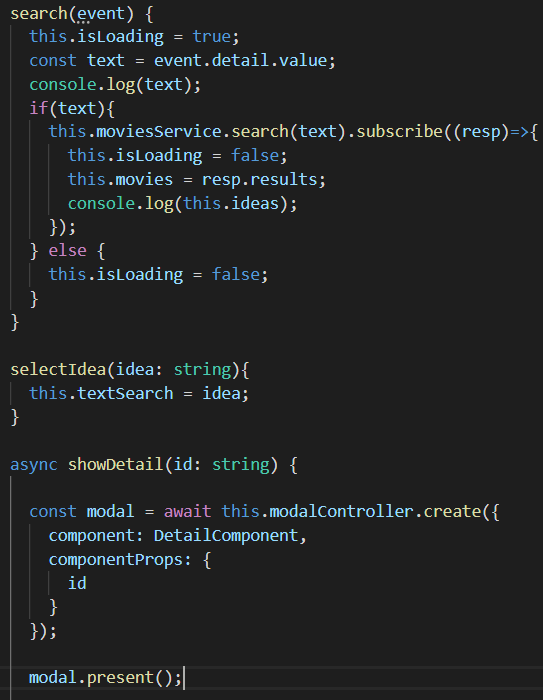
<https://developers.themoviedb.org/3/search/search-movies>

https://api.themoviedb.org/3/search/movie?api\_key=<<api\_key>>&language=en-US&page=1&include\_adult=false









Guardar películas en el storage

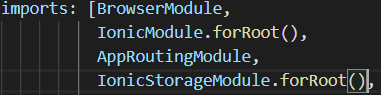
Lo ideal es tener un servicio dedicado al storage

ionic cordova plugin add cordova-sqlite-storage

npm install --save @ionic/storage

y luego importar el storage module en el app.module

import { IonicStorageModule } from '@ionic/storage';





Dentro del servicio importamos en el constructor el storage

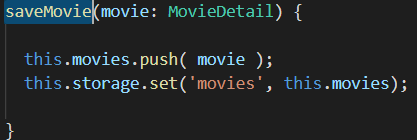




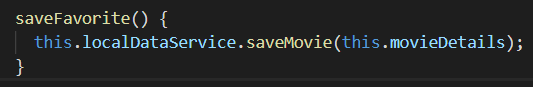
Creamos un arreglo local de películas



Y en el servicio recibimos como parámetro una película y guardamos en el storage con una llave.

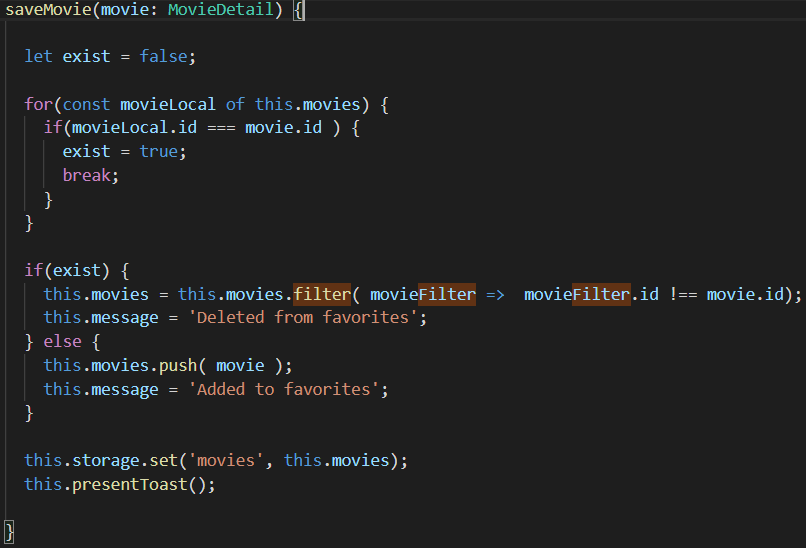


Y desde el modal llamamos al servicio desde el botón de favoritos.



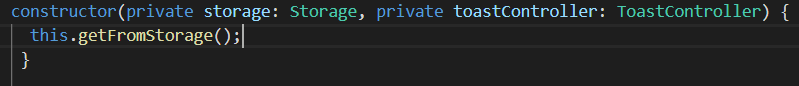
Prevenir el duplicado de películas

Desde el servicio creamos una variable que controle la existencia de la película.  
Luego recorremos el arreglo local preguntando por el id de la película que llego como parámetro.  
De encontrarlo la variable de control es cierta y rompe la búsqueda del for y pasa hacer el filtro local del arreglo de películas para eliminarla.  
  
Luego sale del ciclo y guarda el arreglo en el storage  
De no encontrarla la agrega al storage y al arreglo local.

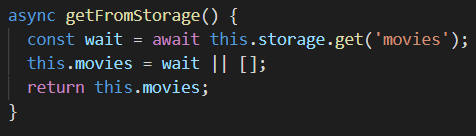


Ahora al cargar la pantalla debemos saber si esta película existe actualmente en el arreglo de favoritos del storage. Pero puede que al momento de cargarla puede que el arreglo local del servicio aun no tenga la información.   
Por eso antes de verificar si la película existe, hay que asegurarse que todas las películas del storage han sido cargadas.

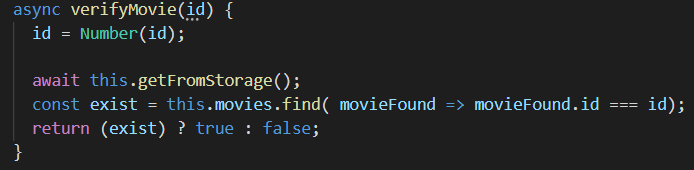
Al momento de inizializar el servicio debemos llamar al método que carga favoritos



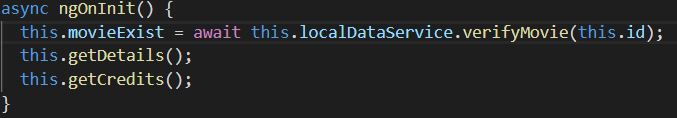
Retornamos la respuesta del storage y en caso de que no exista para que no retorne null retornamos un arreglo vacio.



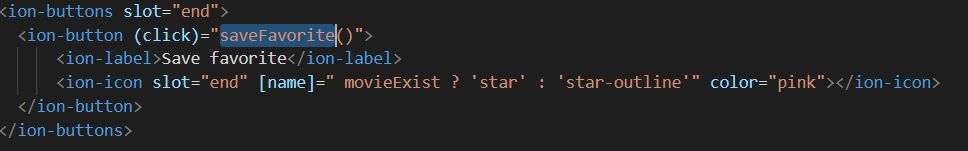
Ahora creamos un método que verifique si existe la película.  
Recibimos un id por parámetro, llamamos al storage para tener el arreglo local de películas actualizado. Y asignamos a una variable que busque en el arreglo si existe o no esa película.  
Si existe retorna el objeto de la película, si no existe retorna undefined.



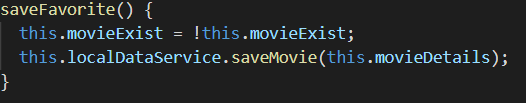
Ahora desde el detail component, en el ngOnInit verificamos si la película existe o no.



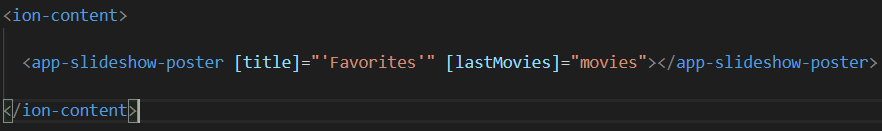
Y desde el htlm asignamos el icono dependiendo de la variable local de si existe o no la película agregada.

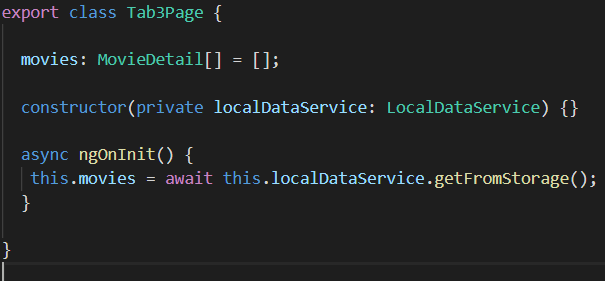


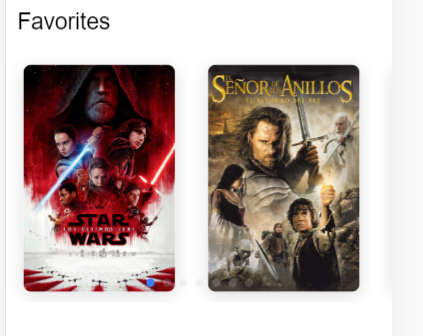
Y al momento de guardar favorito la actualizamos al momento.



Pantalla de favoritos

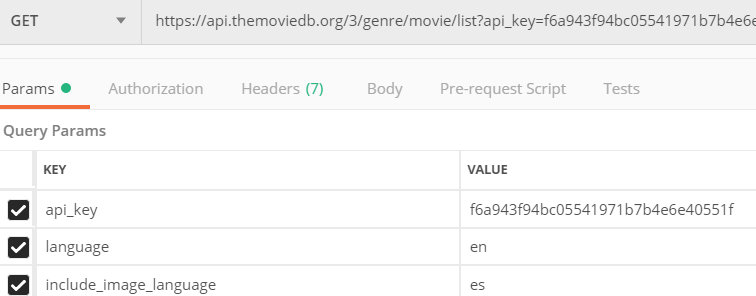




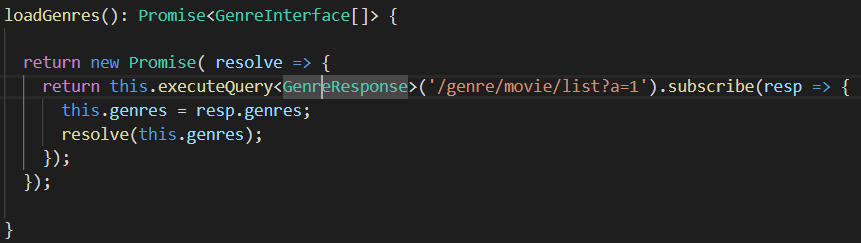


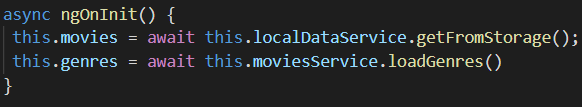
Ahora desde el servicio traeremos los géneros

https://api.themoviedb.org/3/genre/movie/list?api\_key=f6a943f94bc05541971b7b4e6e40551f&language=en&include\_image\_language=es



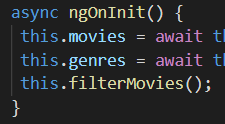
Creamos una promesa que retorna la subscripcion al execurequery y se resuelve cuando tenga los géneros y los regrese.

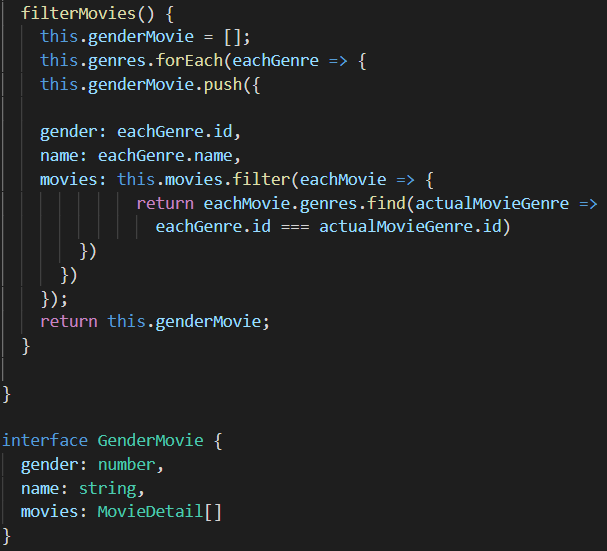




Refrescar después de abrir desde favoritos….

Creamos un método que sea llamado luego de que esten cargadas las películas y los géneros. Recorrerá el arreglo de géneros uno por uno y dentro hará un push a un arreglo de genero/películas en donde hará un recorrido por el arreglo de películas y filtrará las películas que contengan ese genero y lo agrega al arreglo en la propiedad movies que es otro arreglo perteneciente a el arreglo principal.





Y desde el html mandamos cada arreglo de películas a nuestro componente. Mostrando únicamente los géneros que contengan películas.

