

Memoria del proyecto

Take My Movie

Luis Seijas Prieto

Índice

Descripción del problema.....	1
Herramientas utilizadas	2
Aplicación.....	3
Análisis de resultados.....	18
Análisis DAFO	23
Líneas de futuro.....	24
Lecciones aprendidas	24
Bibliografía.....	25

Descripción del problema

Vivimos en una época donde el cine está más presente en nuestras casas que en el propio cine en sí. Estos últimos años las películas se pueden ver con más facilidad que nunca y son muy accesibles. Las plataformas que se nos vienen a la mente cuando pensamos en estos ámbitos suelen ser: Netflix, Amazon Prime Video, HBO y Disney+. Todas éstas tienen una característica en común, y es que estas empresas pertenecen al cine comúnmente llamado "de Hollywood". A la mayoría de la gente le gusta este cine, es obvio, de ahí todos los números que obtienen cada vez que estrenan una película. El problema llega cuando esa monopolización cerca de manera absoluta la diversidad de la proveniencia de las películas y hace muy difícil el poder visionar alguna cinta, por ejemplo, creada en Turquía. El sistema de recomendación de este tipo de plataformas se basa en mostrar al usuario películas americanas, las cuales además no suelen tener una fecha de lanzamiento anterior al 2000.

Take My Movie es un proyecto creado como alternativa a este tipo de sistemas de recomendación. En esta página web, los usuarios podrán compartir de manera instantánea una película que les haya llamado la atención, para que así los demás usuarios puedan

conocer dicha película. De esta manera, las decenas y cientos de cintas que no se muestran en las plataformas anteriormente citadas, se pueden dar a conocer en esta página.

La persona que ingrese en la página web tendrá la oportunidad de crear una nueva película para añadir a la propia base de datos, así como poder visionar el nombre de los actores y directores que componen dicha base de datos.

Además, Take My Movie cuenta con un sistema de recomendación que se basa en las anteriores búsquedas realizadas por el usuario, así como en el género y el año más buscado. De esta manera si existe una película coreana que pertenece al género de *Thriller* y el usuario ha dado a entender al algoritmo de recomendación que es su género favorito, entonces esa misma película se mostrará en pantalla (si se encuentra en la base de datos), independientemente de su proveniencia y de sus distribuidoras y productoras.

Herramientas utilizadas

En el proyecto en cuestión se han empleado las diferentes herramientas/aplicaciones y frameworks:

1. Visual Studio Code: editor de texto.
2. Node.js: es uno de los entornos de ejecución de JavaScript más importantes que existen, está diseñado para crear aplicaciones network escalables.
3. Express: framework especializado para Nodejs que trabaja el back-end de una página, creado para servir aplicaciones web y API's.
4. Postman: programa que se utiliza para el desarrollo de API's. Este software lo he utilizado sobre todo al principio del proyecto, ya que nunca me había adentrado en el mundo de las páginas webs. Muy útil para realizar diferentes peticiones (GET, POST, etc...) al servidor creado.
5. Nodemon: herramienta muy sencilla de instalar que resetea automáticamente la aplicación "node" (en nuestro caso *app.js*) cuando hacemos cualquier cambio en los archivos del directorio donde se encuentra la misma.
6. Bootstrap: conjunto de herramientas que permiten al usuario crear una página web con una buena arquitectura y diseño.

Para crear la arquitectura y el diseño de la página web he utilizado el lenguaje de etiquetado HTML junto a CSS. Cabe destacar que los archivos tienen la extensión .ejs porque he utilizado este mismo lenguaje de plantillas (para generar HTML junto con Javascript). También se ha utilizado JQuery y Javascript para aumentar la accesibilidad e interactividad de la propia página.

Para la base de datos he utilizado el Neo4j, un software cómodo, fácil de utilizar e intuitivo que está orientado a grafos. En especial, he utilizado la propia Sandbox del mismo para

realizar bases de datos “basura” o pequeñas para probar el funcionamiento de las mismas en la página web creada.

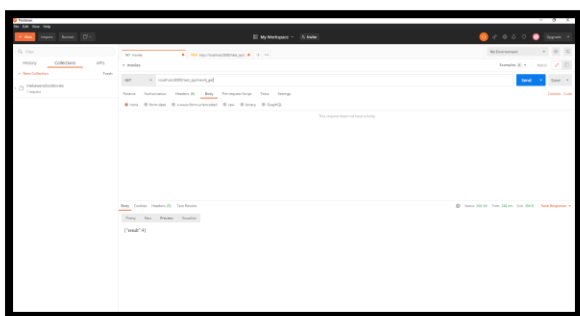
Aplicación

(No se explicará todo el código de la página, ya que contiene miles de líneas de código, pero se explicarán ciertos puntos interesantes o convenientes para explicar el funcionamiento de ciertas interacciones).

Antes de mostrar al completo el funcionamiento de esta subpágina, haremos un repaso de atrás hacia delante de cómo empezó toda la idea de este proyecto.

Al principio de todo, mi conocimiento acerca de Nodejs, JavaScript y Neo4j era nulo. En el momento que empecé a investigar por Internet, encontré un tutorial de cómo crear una página web junto con el Neo4j como base de datos de la misma (*Neo4j + Express.js Tutorial: Easy Server Setup!* - YouTube, n.d.)

Fue el vídeo que me impulsó a empezar con el proyecto motivado ya que me dio esperanzas de que podía funcionar bien. En el vídeo utiliza el software antes mencionado Postman, con el que crea una API y realiza diferentes tipos de peticiones GET y POST para comprobar el funcionamiento de la misma.



En esta imagen se realiza una petición tipo GET que devuelve el número de nodos que contiene la base de datos (no he podido conseguir dicha imagen de la base de datos en Neo4j).

Tras conocer el funcionamiento de ambas herramientas juntas (Nodejs y Neo4j), creé una página web “basura” donde poder realizar inputs para añadir diferentes películas y actores que se convertirían en nodos para la base de datos. En este momento ya era capaz de crear nodos y un tipo de relación entre ellos. La página también mostraba los nodos existentes en la base de datos.

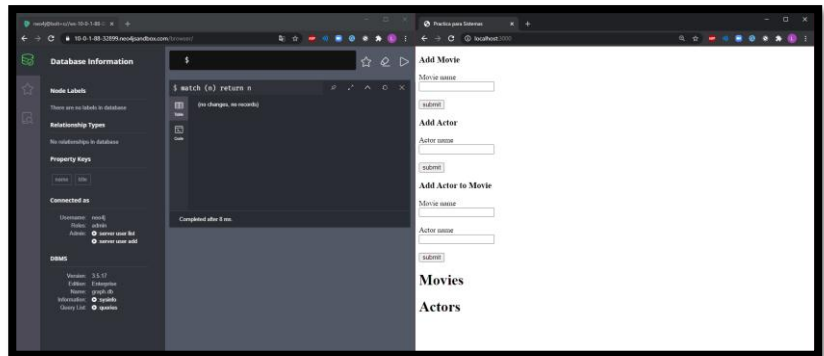
```

<div class="">
  <p>Add Movie</p>
  <form method= "post" action="/movie/add">
    <label>Movie name</label><br>
    <input type= "text" name="name">
    <br><br>
    <input type= "submit" value="submit">
  </form>

  <p>Add Actor</p>
  <form method= "post" action="/actor/add">
    <label>Actor name</label><br>
    <input type= "text" name="name">
    <br><br>
    <input type= "submit" value="submit">
  </form>

  <p>Add Actor to Movie</p>
  <form method= "post" action="/movie/actor/add">
    <label>Movie name</label><br>
    <input type= "text" name="name">
    <br><br>
    <label>Actor name</label><br>
    <input type= "text" name="name">
    <br><br>
    <input type= "submit" value="submit">
  </form>
</div>

```



Aquí se observa la arquitectura de la página web creada y la página en sí.

```

app.post('/actor/add',function(req, res){
  var nameActor=req.body.actorName;
  console.log(nameActor);

  sessionNueva
    .run('MERGE (n:Actor{nombre:{nameParam}})', {nameParam:nameActor})
    .then(function (result){
      /* sessionNueva.close(); */
      res.render('addActor',{
      });
    })
    .catch(function(err){
      console.log(err);
    });
});

```

Fig1

Por ejemplo, en el apartado de Add Actor, al hacer el Submit específico de ese apartado, el <form post="/actor/add"... llama a *app.js* y ejecuta una Query en Cypher específica para crear un nodo con el nombre escrito en el input, el cual es guardado en la variable *nameActor* como se puede observar en la Fig1.

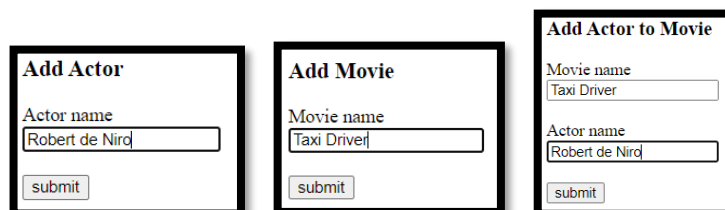
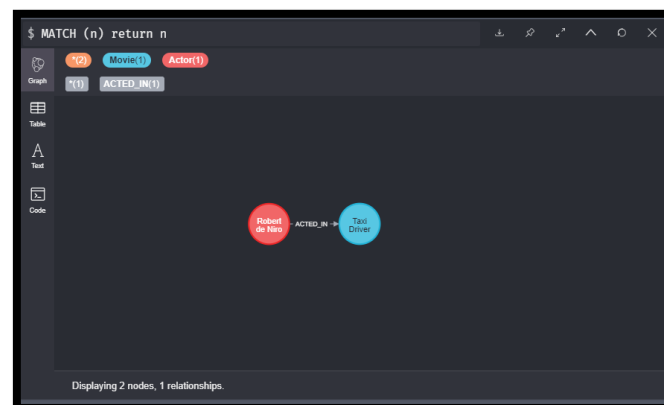
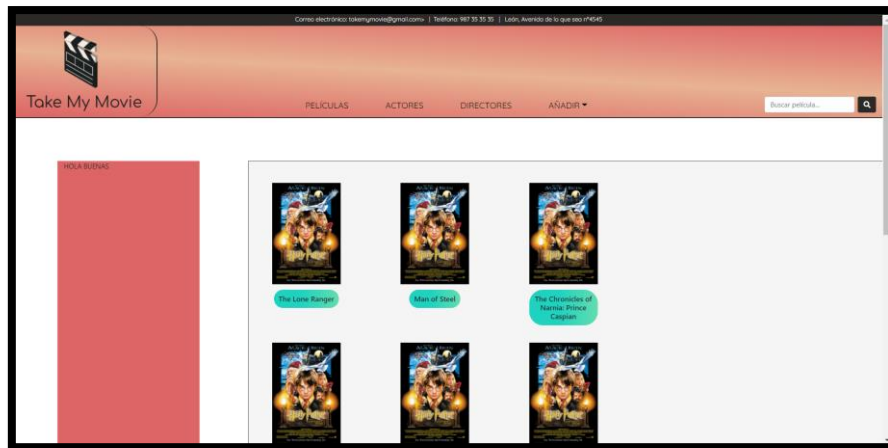


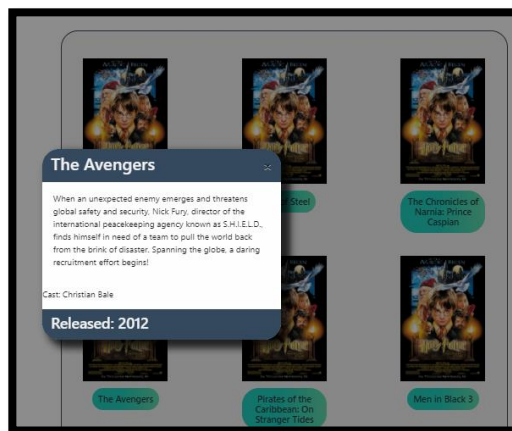
Fig2





Lo siguiente que hice fue crear una página web prototipo. Mi idea era crear una página de películas pero ya no solo el título, sino una imagen de la película encima de él. No sabía cómo me iba a quedar así que solo utilicé una única imagen para todo el proceso y así poder organizar todos los elementos, sin preocuparme de nada más.

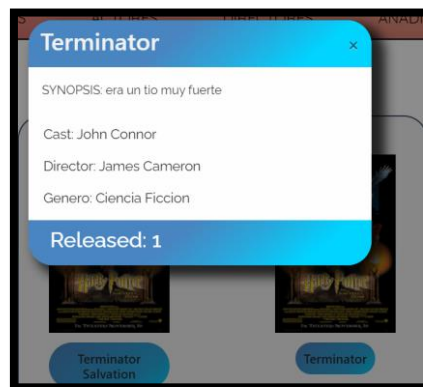
Más tarde se me ocurrió la idea de hacerlo aún más interactivo con el usuario. Investigué en la página oficial de Bootstrap (*Bootstrap · The Most Popular HTML, CSS, and JS Library in the World.*, n.d.) sobre elementos que creía que quedarían elegantes en mi página y encontré las ventanas modales.



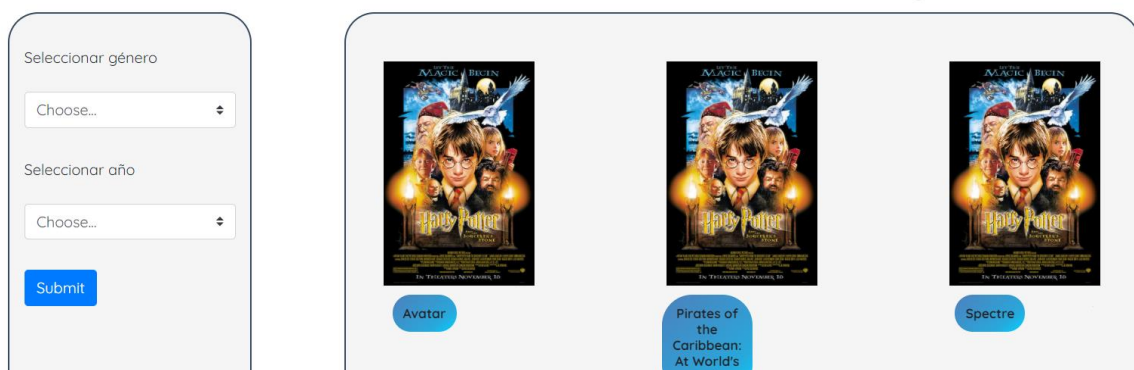
Esto es una ventana modal.

Esto me llevó mucho tiempo llevarlo a cabo e implementarlo ya que no sabía cómo funcionaba y era un sistema complejo que desconocía de HTML. Al principio quería incluir cierta información acerca de la película, por lo cual tenía que acceder a la base de datos para mostrar por pantalla dicha información.

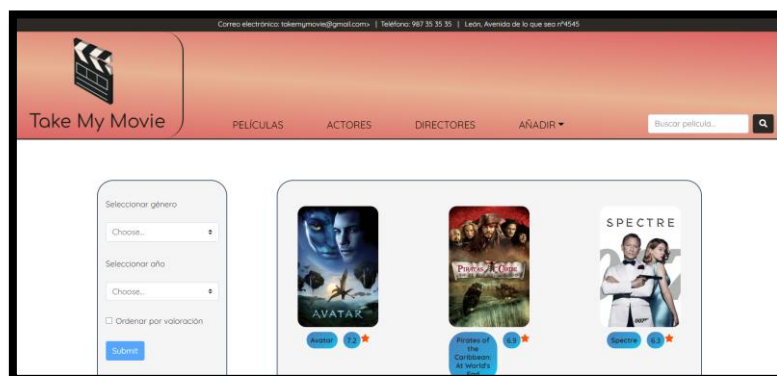
Tras diversos cambios y prototipos en la ventana modal, este sería el modelo perfeccionado.



Tiempo después me empecé a centrar en la búsqueda de películas basada en el filtrado de género y de año. Para ello, en el contenedor de la izquierda coloqué dos tipos de Selectores de texto para así recoger la información introducida por el usuario. De esta manera se podrían buscar películas por su género, por su año de lanzamiento o por ambas (películas de Thriller lanzadas en el año 2005).



Finalmente, en lo que se refiere a esta subpágina, añadí otra cosa a mayores. Dado que la base de datos que descargué de Kaggle (*TMDB 5000 Movie Dataset* | Kaggle, n.d.) comprendía datos sobre el "average" o el voto promedio de cada película, decidí también "jugar" con esos datos.



Realicé cambios en todas las tarjetas de las películas añadiendo el voto promedio al lado del título de la película. Además de la opción de ordenar por valoración.

Con todo el proyecto terminado, la página web consta de 7 diferentes redireccionamientos donde se pueden observar distintos elementos:

1. La página **Inicio**: organizada en el archivo *index.ejs*.

Formada por el header, el menú de navegación (donde se encuentran el logo y los diferentes elementos para poder moverse entre las diferentes subpáginas que componen la página), un contenedor compuesto de imágenes, texto, etc. y el footer.



Fig3.



Fig4.

En esta página se muestra una pequeña introducción acerca de la página web, un carrusel de imágenes mostrando diferentes características de la misma y una imagen en grande ocupando el ancho de la ventana dando por finalizado el contenido de la primera parte.

En la Figura 4 podemos apreciar la arquitectura principal que compone esta página.

2. La página **Películas**: organizada en el archivo *redirect.ejs*.

Formada por el header, el menú de navegación (donde se encuentran el logo y los diferentes elementos para poder moverse entre las diferentes subpáginas que componen la página, también hay un input de texto donde se podrán buscar películas por su título).

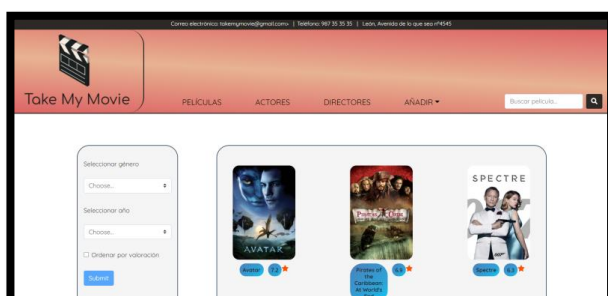


Fig5.



Fig6.

En la Figura 5 observamos el diseño principal de la página y donde se va a desarrollar la mayoría de la interacción con el usuario. Antes de seleccionar género o seleccionar año, la página nos muestra 18 películas, las cuales corresponden a las 18 primeras películas insertadas en la base de datos en Neo4j.

Si no se selecciona ningún género o año, el botón Submit aparecerá en modo *disabled*, por lo cual no se podrá pulsar. El script utilizado es el siguiente:

```
<script>

var e = document.getElementById("inlineFormCustomSelectPref");
var strUser = e.value;
var e2 = document.getElementById("inlineFormCustomSelectPref2");
var strUser2 = e2.value;

if(strUser.length==0 && strUser2.length==0){
    document.getElementById("elemento").disabled=true;
}

function myFunction() {
    document.getElementById("elemento").disabled=false;
}

</script>
```

Se recogen los inputs que ha elegido el usuario en los elementos pertinentes (los relacionados con "Seleccionar género" y "Seleccionar año"). Si el usuario no ha escogido ninguna opción, lo que significaría que la longitud de ambas opciones es igual a 0 (lo recoge la condición *if*) y las características del botón Submit (recogido por su ID de valor "elemento") harán que este en modo "disabled=true" por lo que no podrá

pulsarse. En el momento que cambie cualquier característica del elemento select, se llamará a la función *myFunction()* que hará que el modo disabled pase de "true" a "false". Así es como se describe en el lenguaje de etiquetado seguido del `<select>`:

```
onchange="myFunction()">
```

En la Figura 6 se observa la arquitectura que tiene el menú de navegación. Cada uno de los elementos, que antes de modificarlos tenían la función de link (en HTML, sería el tag `<a>`), los he convertido en un elemento submit, que al hacer click en él nos redirige a otra subpágina. El logo, que forma parte de una etiqueta ``, también se convierte en un Submit, que al pulsar sobre él, nos redirige a la página de **Inicio**. Esta conversión se ha realizado gracias a JQuery. Los métodos form se explicarán más adelante, ya que pertenecen en su mayoría al archivo *app.js*.

Cada una de las tarjetas de las películas tiene este código HTML:

```
<div class="flip-card mb-4" style="width: 10rem">
  <div class="flip-card-inner" style="width: 10rem">
    <div class="flip-card-front">
      <img class="card-image"
        style="width: 10rem"
        src=""
        alt="Avatar"
        style="width: 300px; height: 300px"
      />
      <div class="card-body-propio">
        <h5 class="card-title" id="carta2">Card title</h5>
        <h5 class="card-average">Card Average</h5>
        <span>
          <i class="fas fa-star" style="color: rgb(255, 81, 0);"></i>
        </span>
      </div>
    </div>
    <div class="flip-card-back">
      <a href="#" class="abrir">Leer más</a>
    </div>
  </div>
</div>
```

Cada una de ellas está formado por la imagen de la película, el título de la película y el puntaje o media de la misma. También se ha añadido un `` que es un icono de una estrella, a modo de decoración.

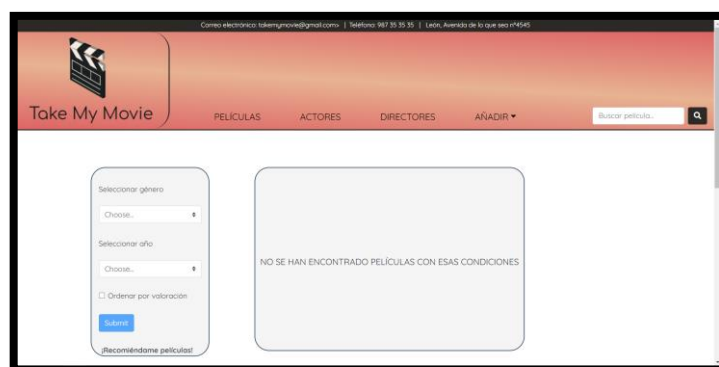
Para hacer la página (a mi parecer) más elegante e interactiva, he añadido unas ventanas modales a cada tarjeta de película, donde se recogen todos los datos de cada una de las propiedades de la película: título de la misma, sinopsis, cast (reparto/actorPrincipal), director, género y fecha de lanzamiento. Más adelante se hablará más a fondo de como mostrar estos datos.



Todas las tarjetas tendrán este diseño realizado en HTML:

```
<div class="modal miModal">
  <div class="flex out" id="flex">
    <div class="contenido-modal">
      <div class="modal-header-propio flex">
        <h2 id="tituloModal"></h2>
        <span class="close" id="close">&times;</span>
      </div>
      <div>
        <p class="card-text">insertarTexto</p>
        <p class="card-reparto">insertarReparto</p>
        <p class="card-director">insertarDirector</p>
        <p class="card-genero">insertarGenero</p>
      </div>
      <div class="footerPropio">
        <h3 class="printReleased"></h3>
      </div>
    </div>
  </div>
</div>
```

La página también cambiará de forma en el momento que se seleccionen géneros o años en los que no haya alguna película con esas características. En ese momento la página pasará a tener esta forma.



Esto se consigue gracias a la utilización de otro script. En él se recoge la longitud del array de películas encontradas en función de los inputs que ha recibido el programa por parte del usuario. Si la longitud de ese array es igual a 0, entonces en el mismo contenedor donde se encuentran la películas se incluirá un código HTML. Este código (el cual se guarda en una variable llamada *html* dará valor al elemento `<div>` que tiene un `id="escribirPelículasNulas"`.

También tenemos la opción de **ordenar por valoración** las diferentes películas, más adelante se mostrará el código Cypher utilizado en *app.js* para tal fin.

Por último, la página cuenta con un buscador arriba a la derecha donde se podrá escribir el nombre de la película que el usuario quiera encontrar. Está programado a prueba de errores, no hace falta que el usuario inserte el título de la película exactamente igual (con sus mayúsculas y palabras). Por ejemplo: si yo quiero buscar la película "Harry Potter y el prisionero de Azkaban" bastará con escribir "potter", "pot", "harry" o "azkab".

3. La página **Actores**: organizada en el archivo *actors.ejs*.

Mismo header y menú de navegación que en la anterior subpágina. Arquitectura basada en un contenedor donde se muestran todos los actores que componen la base de datos en Neo4j. Se actualiza a medida que añadimos más actores.



4. La página **Directores**: organizada en el archivo *directors.ejs*.

Mismo estilo que la subpágina **Actores**, en vez de mostrar el nombre de los actores, nos mostrará el nombre de los directores que componen la base de datos.



5. La página **Añadir-Nueva película**: organizada en el archivo *addMovie.ejs*.

Crear esta subpágina me pareció útil e interesante, tanto para mí como para el usuario. Me basé en la Figura 3, donde se unían un actor y una película y se creaba una relación de :ACTED_IN (lenguaje Cypher).

Este sería el primer prototipo de arquitectura:

Más tarde, tras tener la idea antes mencionada de la valoración de las películas, tendría que añadir un campo donde insertar el valor. También se necesitaba uno para agregar la imagen a la tarjeta de la película. El modelo final es este:

Cada input de este contenedor es obligatorio ponerlo, tal como indica el "required" del código HTML siguiente. Si algún input tiene valor nulo, entonces el *submit* no se llevará a cabo.

```
<div class="col-md-4 mb-3">
  <label for="validationCustom01" style="font-family:Quicksand;">Título de la película</label>
  <input name="titleMovie" type="text" class="form-control" style="font-family:Quicksand;" id="validationCustom01" placeholder="Terminator" required>
  <div class="invalid-feedback">
    Por favor escriba el título de la película.
  </div>
</div>
```

Al igual que para el input de la subpágina **Películas**, cada input tiene un valor "name" el cual recogerá *app.js* y lo guardará en una variable para así realizar la query necesaria en Cypher para crear la película. En el caso de la imagen superior, el valor name será igual a "titleMovie".

6. La página **Añadir-Nuevo actor**: organizada en el archivo *addActor.ejs*.

De arquitectura simple, consta de un pequeño contenedor donde se puede añadir un actor a la base de datos. Obviamente después podrá ser mostrado en la página **Actores**. No crea ningún tipo de relación con ninguna película, simplemente crea un nodo de tipo Actor.

7. La página **Añadir-Nuevo director**: organizada en el archivo *addDirector.ejs*.

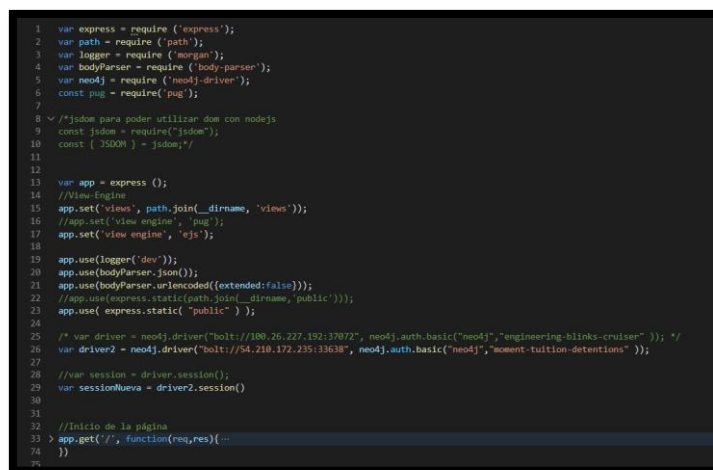
Misma dinámica que la subpágina mencionada anteriormente, los resultados del *submit* se verán en la subpágina **Directores**.

De ahora en adelante explicaré las funciones que tiene el archivo JavaScript *app.js*.

Por decirlo de alguna manera, este archivo es el cerebro de todo el programa. Recoge cada input introducido por el usuario, redirige a las subpáginas, ejecuta los comandos Cypher para Neo4j, etc.

La estructura de este archivo se basa en 3 partes:

- 1- En esta parte, se recogen los objetos y variables necesarias para hacer funcionar el Nodejs y poner en funcionamiento la página web. Como podemos observar en la línea 26, se crea una variable será la que conecte con nuestra base de datos de Neo4j, esto fue posible gracias a la documentación en Github sobre el driver Neo4j-JavaScript (*Neo4j/Neo4j-Javascript-Driver: Neo4j Bolt Driver for JavaScript*, n.d.). También cuenta con el `app.get('/'...]` en la línea 33, que va a ser lo primero que nos muestre la página web tras ingresar en nuestro buscador **"localhost:3000"**.



```
1 var express = require('express');
2 var path = require('path');
3 var logger = require('morgan');
4 var bodyParser = require('body-parser');
5 var neo4j = require('neo4j-driver');
6 const pug = require('pug');
7
8 //Jsdm para poder utilizar dom con nodejs
9 const jsdom = require('jsdom');
10 const { JSDOM } = jsdom;
11
12
13 var app = express();
14 //View Engine
15 app.set('views', path.join(__dirname, 'views'));
16 //app.set('view engine', 'pug');
17 app.set('view engine', 'ejs');
18
19 app.use(logger('dev'));
20 app.use(bodyParser.json());
21 app.use(bodyParser.urlencoded({extended:false}));
22 //app.use(express.static(path.join(__dirname, 'public')));
23 app.use(express.static('public'));
24
25 /* var driver = neo4j.driver("bolt://100.26.227.192:37072", neo4j.auth.basic("neo4j","engineering-blinks-cruiser")); */
26 var driver2 = neo4j.driver("bolt://54.210.172.235:33638", neo4j.auth.basic("neo4j","moment-tuition-detentions"));
27
28 //var session = driver.session();
29 var sessionNueva = driver2.session()
30
31
32 //Inicio de la página
33 app.get('/', function(req,res){
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
```

Cuando se pulse en el menú de navegación "Películas", el submit ejecutará (como se muestra en la Figura 6) los siguientes comandos y funciones.

```
//Muestra las películas iniciales a modo introducción
app.post('/movies/list', function(req,res){

    sessionNueva
    .run("MATCH(n:Movie) RETURN n  order by n.id LIMIT 18 ")
    .then(function (result){
        var movieArray=[];
        result.records.forEach(function(record){
            movieArray.push({
                id: record._fields[0].identity.low,
                title: record._fields[0].properties.title,
                released: record._fields[0].properties.released,
                overview: record._fields[0].properties.overview,
                avgVote: record._fields[0].properties.avgVote,
                image: record._fields[0].properties.image,
            });
        });
    });
});
```

(Se hará lo mismo con actores, directores y géneros)

Esta función, lo que realiza es:

- 1- Crea una sesión nueva de la mano de Neo4j.
- 2- Ejecuta una query en lenguaje Cypher (en este caso, retorna las primeras 18 -- películas de la base de datos).
- 3- Guarda dichas películas en un array.
- 4- Se repite el paso 1 con actores, directores y géneros de dichas películas.

Al final de la función, se necesitará una manera de enlazar nuestros arrays creados, donde se almacenan los datos que hemos requerido, con los archivos .ejs para mostrarlos por pantalla. Este proceso lo ocupa la función **render()**, Figura 8. Después de este proceso se nos redirigirá al archivo .ejs descrito entre paréntesis y se le pasarán las siguientes variables descritas.

```
2401 //llevar al index de alerta cuando no hay suficiente data
2402 > app.post('/index',function(req, res){...
2414 })
2415
2416 //Redirige a la página donde se muestran los actores
2417 > app.post('/actors/list', function(req,res){ ...
2439 })
2440
2441 //Redirige a la página donde se muestran los directores
2442 > app.post('/directors/list', function(req,res){...
2463 })
2464
2465 //Redirige a la página donde se añade la película
2466 > app.post('/adding-movie',function(req, res){...
2477 })
2478
2479 //Redirige a la página donde se añade el actor
2480 > app.post('/adding-actor',function(req, res){...
2490 })
2491
2492 //Redirige a la página donde se añade el director
2493 > app.post('/adding-director',function(req, res){...
2503 })
2504
2505 //Añade película
2506 > app.post('/movie/add',function(req, res){...
2534 })
2535
2536 //Añade actor
2537 > app.post('/actor/add',function(req, res){...
2552 });
2553
2554 //Añade director
2555 > app.post('/director/add',function(req, res){...
2570 });
2571
```

Fig 7.

```
res.render('redirect', {
    movies: movieArray,
    actors: actorArray,
    directors: directorArray,
    genero: generoArray,
    contadorVecesSubmitGenero:contadorVecesSubmitGenero,
    contadorVecesSubmitPelícula:contadorVecesSubmitPelícula,
});
```

Fig 8.

Como ejemplo destacable, menciono esta función:

```
418 //Muestra películas dependiendo de las elecciones de la izquierda
419 app.post('/show-movies',function(req, res){
420     contadorVecesSubmitPelícula++;
421
422     var yearIntroducido = req.body.year;
423     var generoIntroducido = req.body.genero;
424     var check = req.body.check;
425
426     if(yearIntroducido==undefined){...
427     }
428
429     //LOS TRES PRIMEROS IF'S SON PARA CUANDO NO SE PULSA LA OPCIÓN DE ORDENAR
430
431     //IF PARA CUANDO INSERTEMOS UN GENERO Y UN AÑO ----- TICK NO
432     if(generoIntroducido.length>0 && yearIntroducido.length>0 && check==undefined){...
433     }
434
435     //IF PARA CUANDO INSERTEMOS SOLO EL AÑO ----- TICK NO
436     if(yearIntroducido.length>0 && generoIntroducido.length==0 && check==undefined){...
437     }
438
439     //IF PARA CUANDO INSERTEMOS SOLO EL GENERO ----- TICK NO
440     if(generoIntroducido.length>0 && yearIntroducido.length==0 && check==undefined){...
441     }
442
443     //LOS TRES SEGUNDOS IF'S SON PARA CUANDO SÍ SE PULSA LA OPCIÓN DE ORDENAR
444
445     //IF PARA CUANDO INSERTEMOS UN GENERO Y UN AÑO ----- TICK SI
446     if(generoIntroducido.length>0 && yearIntroducido.length>0 && check=="on"){...
447     }
448
449     //IF PARA CUANDO INSERTEMOS SOLO EL AÑO ----- TICK SI
450     if(yearIntroducido.length>0 && generoIntroducido.length==0 && check=="on"){...
451     }
452
453     //IF PARA CUANDO INSERTEMOS SOLO EL GENERO ----- TICK SI
454     if(generoIntroducido.length>0 && yearIntroducido.length==0 && check=="on"){...
455     }
456
457     });
```

Fig9.

En ella se recogen todas las posibilidades que, dentro de la página [Películas](#), pueden ocurrir en lo que se refiere a la selección de géneros, años y opción de ordenación por valoración. Básicamente se recogen los **posibles** datos introducidos por el usuario, que serán year, genero y check. Estos mismos serán guardados en variables. Algunos ejemplos los veremos en la sección de Análisis de Resultados.

Algoritmo de Recomendación

El algoritmo de recomendación utilizado para esta práctica se basa en la técnica de **filtrado colaborativo**, más específicamente usa la técnica de filtrado colaborativo de contenido. En este tipo de filtrado, las recomendaciones se hacen según los contenidos que pueden gustar o interesar al usuario.

Creo que es el algoritmo idóneo para este proyecto ya que la página web cuenta con los selectores de género, año y las búsquedas por título; esto ayuda a que dichos valores se puedan guardar y utilizar en los momentos que me convenga para crear el algoritmo.

Para hacer un sistema de recomendación más optimizado, he impuesto la regla de que el usuario deberá **al menos**:

- Utilizar el "Seleccionar género" al menos 5 veces.
- O Utilizar el "Seleccionar año" al menos 5 veces.
- O buscar películas en la barra buscadora y la que más se repita tras 5 búsquedas (o más, recordemos), recoger ese título y que el sistema recomiende en base a esa película.

Estas 3 condiciones se pueden dar por separado o juntas; en la sección de análisis de resultados lo veremos más a fondo.

```
//SACAR LA PELICULA MAS BUSCADA
if(contadorVecesSubmitPelícula>=5){ ...
}

//SACAR EL GENERO MAS BUSCADO
if(contadorVecesSubmitGenero>=5){ ...
}

//SACAR EL AÑO MAS BUSCADO
if(contadorVecesSubmitYear>=5){ ...
}
```

Fig10.

```
//SACAR LA PELICULA MAS BUSCADA
if(contadorVecesSubmitPelícula>=5){
  var counts = {};
  var compare = 0;

  (function(moviesSearched){
    for(var i = 0, len = moviesSearched.length; i < len; i++){
      var word = moviesSearched[i].title;

      if(counts[word] === undefined){
        counts[word] = 1;
      }else{
        counts[word] = counts[word] + 1;
      }
      if(counts[word] > compare){
        compare = counts[word];
        mostFrequentFilm = moviesSearched[i].title;
      }
    }
    console.log("La película más buscada es " +mostFrequentFilm)
    return mostFrequentFilm;
  })(moviesSearched);
}
```

Fig11.

Para cada *if* que compone la Figura 9, existe un conjunto de 3 *if*'s que comprueba que se hayan cumplido las 3 condiciones impuestas anteriormente. Así nuestro algoritmo de recomendación tendrá una previa información acerca de los intereses del usuario y se podrá elaborar un filtrado más avanzado.

El primer *if* de la Figura 10 hace referencia al *if* de la Figura 11. Es un método para encontrar la película que más se repite dentro del array de películas buscadas por el usuario (habiendo sido buscadas por cualquiera de los 3 métodos, género, año o título). Este mismo método es el que se utiliza en los 2 siguientes *if* de la Figura10, pero cogiendo como referencia el género y el año.

Para acceder al sistema de recomendación basta con pulsar en el submit **¡Recomiéndame una película!**:

Selecciónar género

Choose...

Selecciónar año

Choose...

☐ Ordenar por valoración

Submit

¡Recomiéndame películas!

Nuestro sistema de recomendación mostrará el mismo tipo de tarjetas de películas pero con la ventana modal deshabilitada.

Nuestro sistema de recomendación entonces, tiene hasta 3 algoritmos:

- 1- Se han recogido una **película** y un **género** donde el usuario ha puesto especial interés pero no hay datos acerca del **año**.
- 2- Se han recogido una **película** y un **año** donde el usuario ha puesto ha puesto especial interés pero no hay datos acerca del **género**.
- 3- Se han recogido datos acerca de los **tres** posibles factores.

Estas tres condiciones son recogidas en `app.post('/recommend-movie[...]`

```
2298 > if(mostFrequentYear==undefined && mostFrequentFilm!=undefined && mostFrequentGenre!=undefined){ ...
2323 > }else if(mostFrequentYear!=undefined && mostFrequentFilm!=undefined && mostFrequentGenre==undefined){ ...
2347 > }else if(mostFrequentYear!=undefined && mostFrequentFilm!=undefined && mostFrequentGenre!=undefined){ ...
2372 > }else if(mostFrequentYear==undefined && mostFrequentFilm!=undefined && mostFrequentGenre==undefined){ ...
2383 > }else{ ...
2394 > }
```

Vamos a observar el algoritmo que sigue, por ejemplo, el primer *if*, donde el año de interés no se conoce pero se conoce la **película de interés** y el **género de interés**.

```
if(mostFrequentYear==undefined && mostFrequentFilm!=undefined && mostFrequentGenre!=undefined){
  sessionNueva2
  .run('match (n:Movie)-[:ACTED_IN]-(a:Actor)-[:ACTED_IN]->(n2:Movie) where n.title={titleMovie} return n2 as name order by n2.avgVote
union match (n:Movie)-[:tieneGenero]->(g:Genero)-[:tieneGenero]-(n3:Movie) where g.genero={generoMovie} return n3 as name order by
n3.avgVote', {titleMovie:mostFrequentFilm, generoMovie:mostFrequentGenre})

  .then(function (result){
    var movieArray2=[];
    result.records.forEach(function(record){
      movieArray2.push({
        id: record.fields[0].identity.low,
        title: record.fields[0].properties.title,
        released: record.fields[0].properties.released,
        overview: record.fields[0].properties.overview,
        avgVote: record.fields[0].properties.avgVote,
        image: record.fields[0].properties.image,
      });
    });

    res.render('recommend', {
      movies2: movieArray2,
      film:film,
    });
  })
  .catch(function(err){
    console.log(err);
  });
});
```

La query lanzada en lenguaje Cypher será:

```
match (n:Movie)-[:ACTED_IN]-(a:Actor)-[:ACTED_IN]->(n2:Movie)
where n.title={titleMovie}
return n2 as name
order by n2.avgVote
union
match (n:Movie)-[:tieneGenero]->(g:Genero)-[:tieneGenero]-(n3:Movie)
where g.genero={generoMovie}
return n3 as name
order by n3.avgVote
```

En esta query (que será puesta a prueba en el análisis de resultados), vemos como conocidos la película más repetida del array y el género, el Neo4j retornará dos tipos de películas:

- Todas las películas donde el actor de la **película de interés** ha actuado.
- Todas las películas que tienen el mismo género que el **género de interés**.

Ya que he tratado de crear la página a prueba de errores, si el usuario pulsa el botón para activar el sistema de recomendación y no se cumple ninguna de las 3 condiciones anteriores, entonces la página retornará una ventana de alerta explicando lo sucedido y redirigirá al inicio. Esto lo veremos en el análisis de resultados.

Análisis de resultados

1. Mostrar películas tras haber seleccionado el año (2015) y que se enseñen por orden por valoración descendente.
(En el propio programa hay escritos unos *console.log()* específicos para ayudar al usuario a observar por consola cuál es el género, año o película de interés al momento).

```
Server Started on Port 3000
Generos:
[]
Años:
[ '2015' ]
Películas contadas:
Se han contado 6 películas

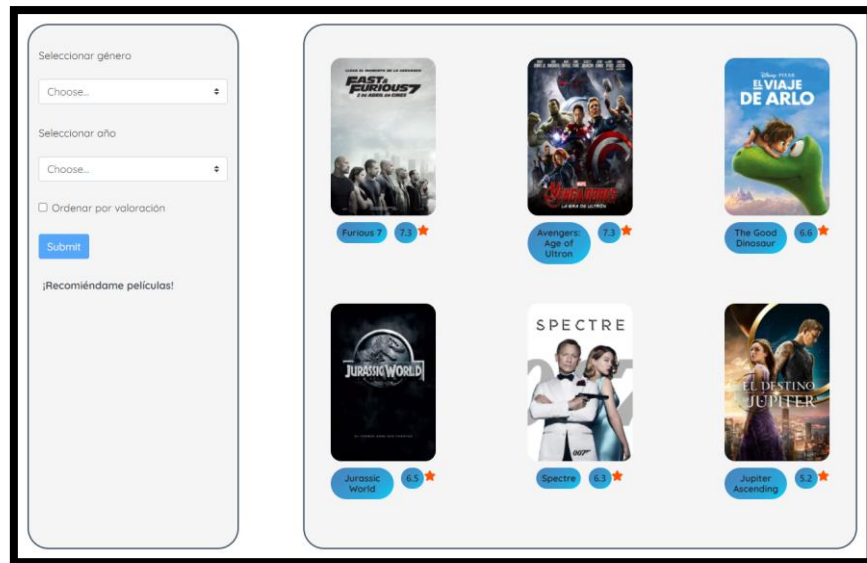
Generos contados:
Se han contado 0 generos

Años contados:
Se han contado 1 años

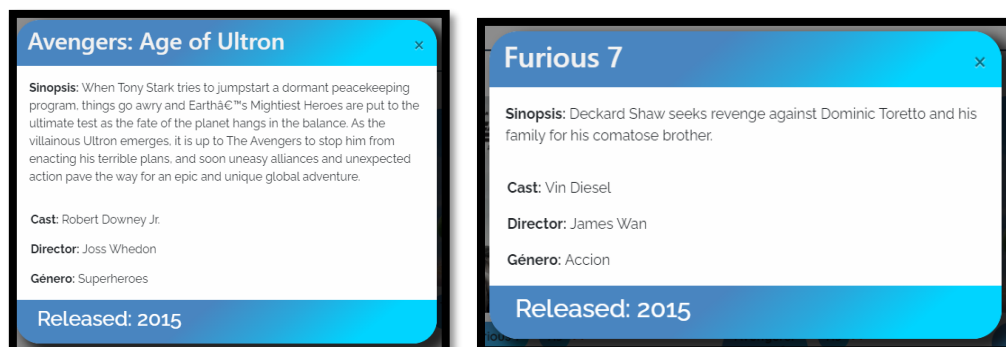
[
  { title: 'Furious 7' },
  { title: 'Avengers: Age of Ultron' },
  { title: 'The Good Dinosaur' },
  { title: 'Jurassic World' },
  { title: 'Spectre' },
  { title: 'Jupiter Ascending' }
]
CONTADOR DE ACTUALIZACIONES: 1
POST /show-movies 200 1245.147 ms - 68891
GET /stylesRedirect.css 304 6.047 ms - -
GET /logo.png 304 0.928 ms - -
```

Como podemos observar en esta imagen, en consola también se muestra el array de los géneros buscados, años buscados y películas mostradas por pantalla.

Como es obvio, el programa nos mostrará las películas que han sido lanzadas en ese año escogido y también se ordenarán en función del voto promedio.



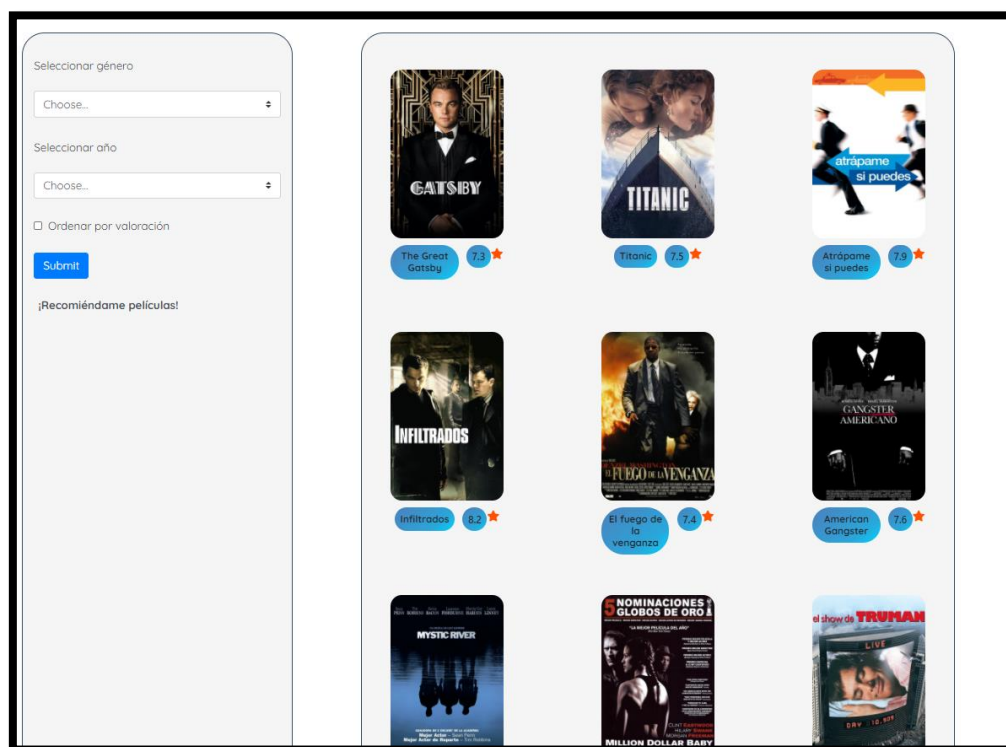
Este será el resultado que se nos mostrará por pantalla, todas ordenadas según el voto promedio de manera descendente y con una misma fecha de lanzamiento. Vemos como las dos primeras cumplen con la condición:



2. Resultado del sistema de recomendación tras conocer película de interés (Shutter Island) y género de interés (Drama).

```
CONTADOR DE ACTUALIZACIONES: 9
La película más buscada es Shutter Island
El genero mas buscado es Drama
POST /show-movies 200 878.456 ms - 82157
GET /stylesRedirect.css 304 0.725 ms - -
GET /logo.png 304 0.835 ms - -
VECES GENERO: 5
VECES AÑO: 0
VECES SUBMIT: 9
Shutter Island —
undefined —
Drama —
```

Como podemos observar, el año de interés tiene valor "undefined" ya que no hemos interactuado con el selector de año 5 o más veces.



Tras pulsar en el botón de mostrar las recomendaciones, se nos devuelve una serie de películas. Las primeras siempre serán las películas donde el actor de, en este caso Shutter Island (Leonardo DiCaprio), ha actuado. Entre estas películas podemos observar El Gran Gatsby, Titanic, Atrápame si puedes y Infiltrados. Las siguientes serán las películas que pertenecen al género de interés recogido, el cual es Drama, donde podemos ver Mystic River, El fuego de la venganza, American Gangster, etc.

3. Resultado tras añadir una nueva película a la base de datos

Ejecutaremos el siguiente *submit* para añadir una película:

Título de la película	Nombre del actor	Nombre del director
<input type="text" value="El caso Fischer"/>	<input type="text" value="Tobey Maguire"/>	<input type="text" value="Edward Zwick"/>
Sinopsis	Año de estreno	
<input (el="" ajedrez="" bobby="" boris="" campeonato="" campeón="" de="" del="" duelo,="" el="" enfrentamiento="" entre="" fischer,="" historia="" la="" legendario="" mundo="" narra="" norteamericano,="" pawn="" peón)="" por="" preparación="" que"="" ruso="" sacrifice\"="" sacrificio="" spassky.="" type="text" value="\" y=""/>	<input type="text" value="2015"/>	
Género de la película	Voto promedio en themoviedb.org	Dirección de la imagen en themoviedb.org
<input type="text" value="Drama"/>	<input type="text" value="6.7"/>	<input type="text" value="https://image.tmdb.org/t/p/w600_and_h?"/>
<input type="button" value="Submit Movie"/>		

Para comprobar el funcionamiento de esta herramienta, buscaremos la película en la barra buscadora a ver si se ha añadido correctamente

fischer

Seleccionar género

Choose...


Seleccionar año

Choose...


☐ Ordenar por valoración

Submit

¡Recomiéndame películas!



EL CASO FISCHER



El caso Fischer

6.7

El caso Fischer

Sinopsis: "Pawn Sacrifice" (El sacrificio del peón) narra la historia de la preparación y del legendario enfrentamiento por el campeonato del mundo entre Bobby Fischer, campeón de ajedrez norteamericano, y el campeón ruso Boris Spassky. El duelo, que tuvo lugar en 1972, en plena Guerra Fría, fue mucho más que un conjunto de partidas para conquistar un campeonato; prueba de ello es que captó la atención de todo el mundo.

Cast: Tobey Maguire

Director: Edward Zwick

Género: Drama

Released: 2015

Como podemos observar, la película se ha añadido perfectamente a la base de datos y los datos mostrados son los correctos.

Cabe destacar que aquí tuve unos pequeños problemas, ya que al añadir una película con un actor/director que ya estaba contenido en la base de datos, en este caso Tobey Maguire ya existe, por ejemplo, con la película Spider-Man 2, la base de datos se me descolocaba entera ya que se añadía otro nodo más a la BBDD.

Esto lo arreglé construyendo una query más precisa para que no me ocurriera dicho problema:

```
//Añade película
app.post('/movie/add',function(req, res){
  var title=req.body.titleMovie;
  var nameActor=req.body.nameActor;
  var nameDirector=req.body.nameDirector;
  var synopsisMovie=req.body.synopsisMovie;
  var released=req.body.releasedMovie;
  var genero=req.body.generoMovie;
  var avgVoteStr=req.body.averageMovie;
  var image=req.body.url_image;

  console.log(genero)
  avgVote=parseFloat(avgVoteStr);

  sessionNueva
    .run('MERGE (n:Movie {title:{titleParam}, overview:{overviewParam}, released:{releasedParam}, avgVote:$averageParam, image:{imageParam}})
    MERGE (a:Actor {nombre:{nameAParam}}) MERGE (a)-[:ACTED_IN]-(n) MERGE (d:Director {nombre:{nameDParam}}) MERGE (d)-[:haDirigido]-(n)
    MERGE (g:Genero{genero:{generoParam}}) MERGE (n)-[:tieneGenero]-(g) return n,g,a,d', {titleParam:title, overviewParam: synopsisMovie,
    releasedParam:released}, nameAParam:nameActor, nameDParam:nameDirector, generoParam:genero, averageParam:avgVote,
    imageParam:image))
    .then(function (result){
      //res.redirect('/');
      //sessionNueva.close();
      res.render('addMovie',{
      });
    })
    .catch(function(err){
      console.log(err);
    });
});
})
```

De esta manera si el actor que se introduce para la nueva película, ya existe en la base de datos, ese nodo de tipo actor no se creará.



Si nos dirigimos a la página Actores, podemos observar que aunque haya 5 películas donde Tobey Maguire actúa, solo aparece una vez.

Análisis DAFO



Una de las ventajas principales era que, en este cuatrimestre, solo tenía este proyecto grande por hacer. No tenía ninguna asignatura más la cual me mandara un trabajo de esta medida.

Otra de las ventajas que me ayudaron para crear la página, era el previo conocimiento sobre HTML y CSS. No era un conocimiento absoluto ni muy grande pero me ayudó.

Y una de las limitaciones que encontré en el proyecto fue el tema de la escalabilidad. La página web no está creada para mostrar una gran cantidad de películas debido al diseño; debe de actualizarse para tal fin.

Líneas de futuro

Hacer la página web viable para una base de datos más grande.

Solucionar los problemas con el UTF8 (caracteres especiales, tildes, etc.)

Convertir la página en una página responsive para móviles y tabletas.

Utilizar diferentes propiedades de las películas que no he podido utilizar, tales como el presupuesto y añadir otras propiedades como co-actores. De esta manera,

Mejorar el algoritmo para que las recomendaciones muestren primero si el usuario quiere que se le ofrezcan películas extranjeras y no americanas.

Poder crear usuarios con su contraseña. De esta manera se podrán seguir entre ellos y crear nuevos algoritmos de recomendación, los cuales estén basados en los gustos de los usuarios a los que sigas.

Lecciones aprendidas

Una de las cosas más importantes que he aprendido gracias a este proyecto ha sido que la ambición a veces puede jugar una mala pasada. Para esta página web he querido hacer ciertas cosas que, por lo que sea, me han llevado mucho tiempo y a veces he andado justo de tiempo. Al principio no tenía ningún tipo de organización sobre mi trabajo (incluso a veces no he podido asistir a clase), pero poco a poco he ido aprendiendo. Al final me ha salido todo como yo deseaba.

Otra lección aprendida es que siempre va a haber alguien en Internet que te ayude a solucionar un problema, y esto es muy importante de saber. Muchas veces habrá libros, foros o páginas web que resuelvan tu duda de manera instantánea.

Después de cursar la asignatura de Bases de Datos, este ámbito dentro de la informática no me llamaba nada la atención. Gracias a esta asignatura he descubierto el mundo de Neo4j, que es totalmente distinto a lo anterior mencionado. Se trabaja muy bien con esta herramienta y enlazarla con cualquier página web se ha convertido en una tarea muy fácil para mí.

Ha sido el proyecto más importante que he hecho en toda la carrera de Ingeniería Informática y al que más he dedicado horas con diferencia. Me siento realizado por completar este proyecto a tiempo.

Bibliografía

Bootstrap · The most popular HTML, CSS, and JS library in the world. (n.d.). Retrieved December 16, 2020, from <https://getbootstrap.com/>

neo4j/neo4j-javascript-driver: Neo4j Bolt driver for JavaScript. (n.d.). Retrieved December 16, 2020, from <https://github.com/neo4j/neo4j-javascript-driver>

Neo4j + Express.js Tutorial: Easy Server Setup! - YouTube. (n.d.). Retrieved December 16, 2020, from <https://www.youtube.com/watch?v=aD9exhioWO4>

TMDB 5000 Movie Dataset | Kaggle. (n.d.). Retrieved December 16, 2020, from <https://www.kaggle.com/tmdb/tmdb-movie-metadata>