

Proyecto Fin de Ciclo (PFC)

Desarrollo de Aplicaciones Multiplataforma

Martín García Ramos

Versión: Final marzo 2024

**Índice**

[1.- Descripción del proyecto y ámbito de implantación: 3](#_Toc162128989)

[1.A. - ¿Qué has desarrollado y por qué? 3](#_Toc162128990)

[1.B. - ¿Quién lo usará? 3](#_Toc162128991)

[1.C. - ¿Qué tecnologías has usado? 3](#_Toc162128992)

[1.D. - ¿Se prevén cambios en el futuro? 4](#_Toc162128993)

[2. – Temporalización del proyecto y fases de desarrollo 5](#_Toc162128994)

[2.A - PLANIFICACIÓN: 5](#_Toc162128995)

[2.A.1. PERT: 5](#_Toc162128996)

[2.A.2. GANTT: 8](#_Toc162128997)

[2.B. - RETOS ENCONTRADOS: 9](#_Toc162128998)

[3. – Recursos de Hardware y software 10](#_Toc162128999)

[3.A. - Requisitos de Hardware: 10](#_Toc162129000)

[3.A.1. Requisitos del navegador: 10](#_Toc162129001)

[3.A.2. Requisitos para ejecutar el servidor de ReactJS: 10](#_Toc162129002)

[3.B. - Requisitos software: 11](#_Toc162129003)

[3.A.1. Requisitos del navegador: 11](#_Toc162129004)

[3.A.2. Requisitos para ejecutar el servidor de ReactJS: 11](#_Toc162129005)

[4. – Arquitectura software y de sistemas 13](#_Toc162129006)

[4.A. - Diagrama de Secuencia: 13](#_Toc162129007)

[4.B. - Diagrama de Navegación: 14](#_Toc162129008)

[4.C. - Diagrama de Jerarquía de componentes en cada página: 14](#_Toc162129009)

[5. – Descripción de datos 17](#_Toc162129010)

[6. – Documentación adjunta 18](#_Toc162129011)

[6.A. GITHUB: 18](#_Toc162129012)

[6.B. DIAGRAMAS: 18](#_Toc162129013)

[6.C. TRAILER: 18](#_Toc162129014)

# 1.- Descripción del proyecto y ámbito de implantación:

## 1.A. - ¿Qué has desarrollado y por qué?

Se ha desarrollado una aplicación web que tiene como objetivo ser un lugar seguro y privado en el que poder registrar anónimamente los contenidos audiovisuales que te gustan, así como ver detalles interesantes de los mismos.

La idea surgió de la necesidad de tener siempre a mano un blog de notas para apuntar donde te habías quedado en esa serie o hace 3 meses. Si eres un ferviente seriéfilo o cinéfilo, más pronto que tarde empezarás a tener una duda real sobre qué series y qué películas te gustaron en el pasado. Evidentemente en un corto periodo de tiempo puedes recordarlas pero no en un plazo de 1, 2, 5, 10 o más años donde unos títulos se te entremezclarán con otros y donde muchos de esos contenidos simplemente los olvidarás, para luego un día decir: ¿Cómo se llamaba aquella película/serie que me gustó tanto?  
  
Con ese propósito nace AVMaster, una plataforma web, con la que el usuario puede interactuar de manera intuitiva e ir gestionando de manera automática todos los contenidos que le gustan para luego poderlos filtrar como se desee.

## 1.B. - ¿Quién lo usará?

En principio, cualquier persona puede, aunque en realidad esta plataforma tiene un interés especial en el colectivo de los seriéfilos y cinéfilos dedicados, aquellas personas que en sus ratos libres no conciben un mejor plan que ponerse a ver solos o en compañía contenidos audiovisuales.

En ese sentido todo el branding de la web está diseñado con la finalidad de atraer al público de todas las diversas edades, así como ayudándoles a localizar fácilmente cada una de las opciones con las que pueden interactuar.

## 1.C. - ¿Qué tecnologías has usado?

Las tecnologías usadas fueron varias:

* **HTML:** Es un lenguaje de marcado utilizado para crear y estructurar páginas web.
* **CSS:** Es un lenguaje utilizado para definir el estilo y la presentación de documentos HTML.
* **JavaScript:** Es un lenguaje de programación utilizado principalmente para agregar interactividad y dinamismo a páginas web.
* **React:** Es una biblioteca de JavaScript utilizada para construir interfaces de usuario interactivas y reutilizables para aplicaciones web.
* **Bootstrap****:** Es un marco de trabajo de código abierto utilizado para diseñar y desarrollar interfaces web responsivas y estilizadas.

1.D. - ¿Se prevén cambios en el futuro?

Uno de los cambios que serían más urgentes sería el de ajustar correctamente el centrado de los elementos de manera responsiva, a la hora de mostrar el listado de ítems. Es lo único(Junto con el encuadre de la barra lateral en la página de detalles)que he intentado hacer y no he conseguido realizar para que quedara correctamente establecido.

Por otro lado, uno de los objetivos iniciales que se preveían para la aplicación era que en la página de detalles de la serie o la película también saliera su tráiler, así que esta sería una mejora sencilla a implementar.

Además, en esa misma página de detalles también iba a tener las fotografías de los actores, directores, etc, que estuvieran disponibles en la API de themoviedb.com, pero se terminó descartando al complicarse innecesariamente, también se podría incluir en una segunda iteración.  
  
Por otro lado, originalmente en el proyecto también se planteó que en vez de solo mostrarse el listado de favoritos, el usuario pudiera listar por muchos más atributos e incluso pudiera hacer sus propios subconjuntos, creando por ejemplo: “Series que dejé de ver”, de esa manera la aplicación mejoraría notablemente su usabilidad e interés por parte de su público potencial; lamentablemente, no se terminó descartando por su complejidad, pero se podría atajar en una posible segunda iteración del proyecto.

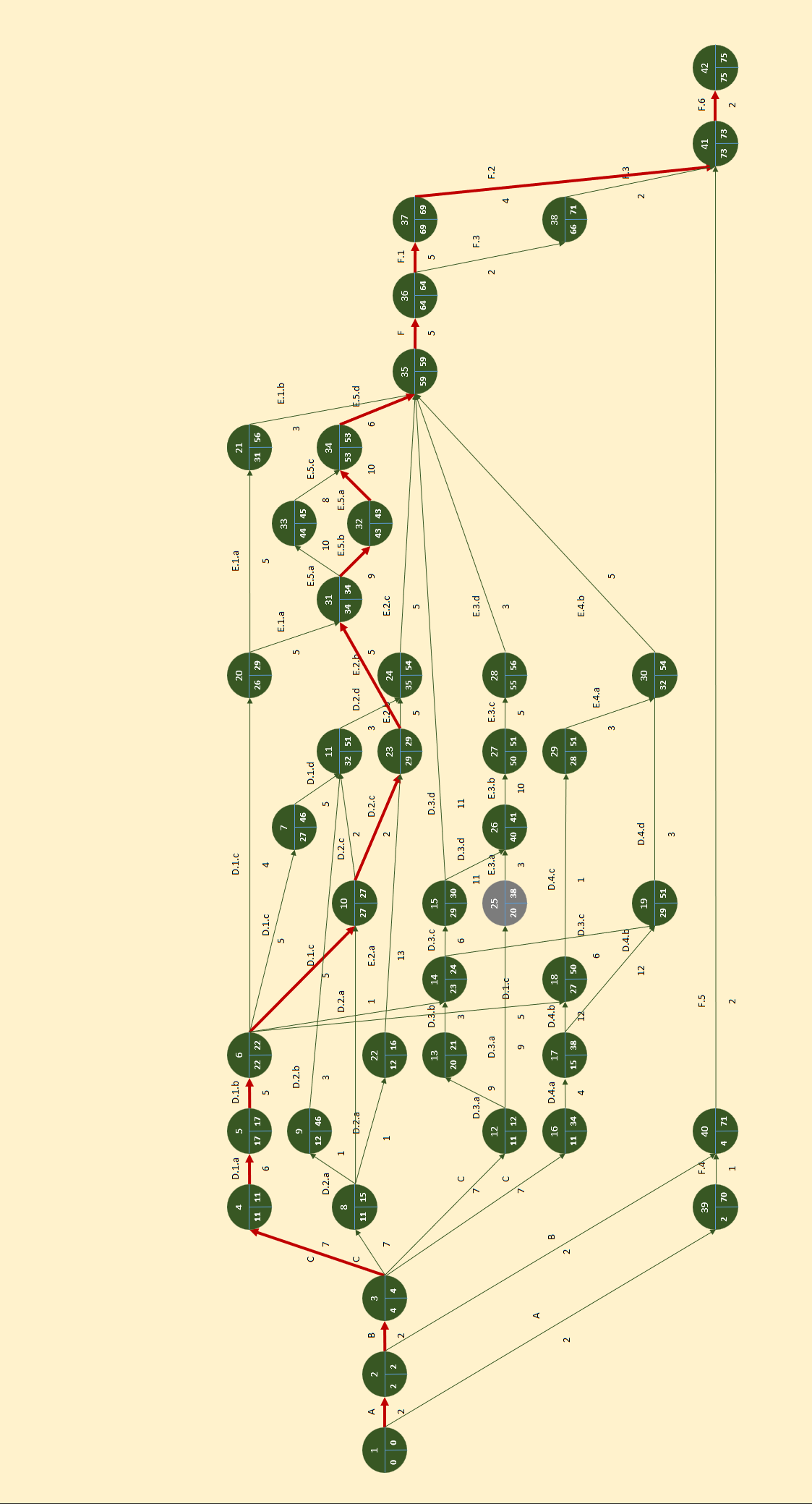
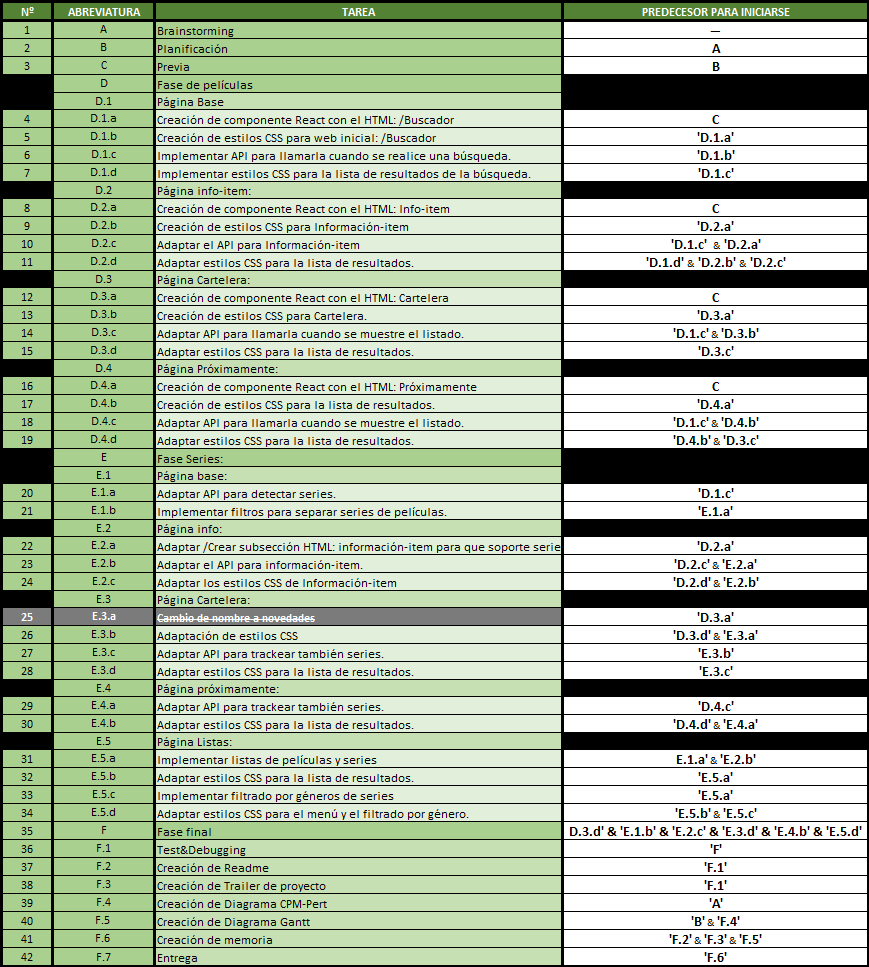
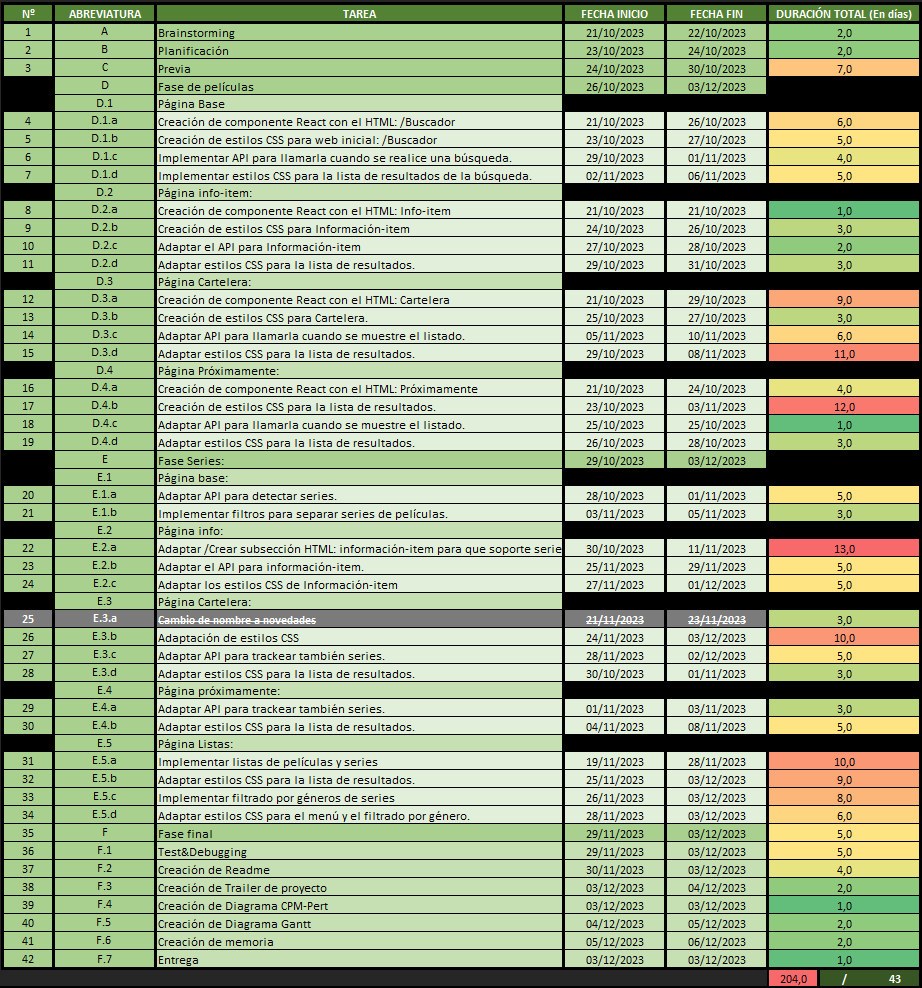
# 2. – Temporalización del proyecto y fases de desarrollo

## 2.A - PLANIFICACIÓN:

Planificación y descripción de las actividades en los diagramas PERT y GANT. Debido a que se permiten iteraciones dinámicas, estos diagramas se han localizado en el Excel adjunto a este pdf.  
No obstante dejo aquí algunas capturas para dejar constancia de ello:

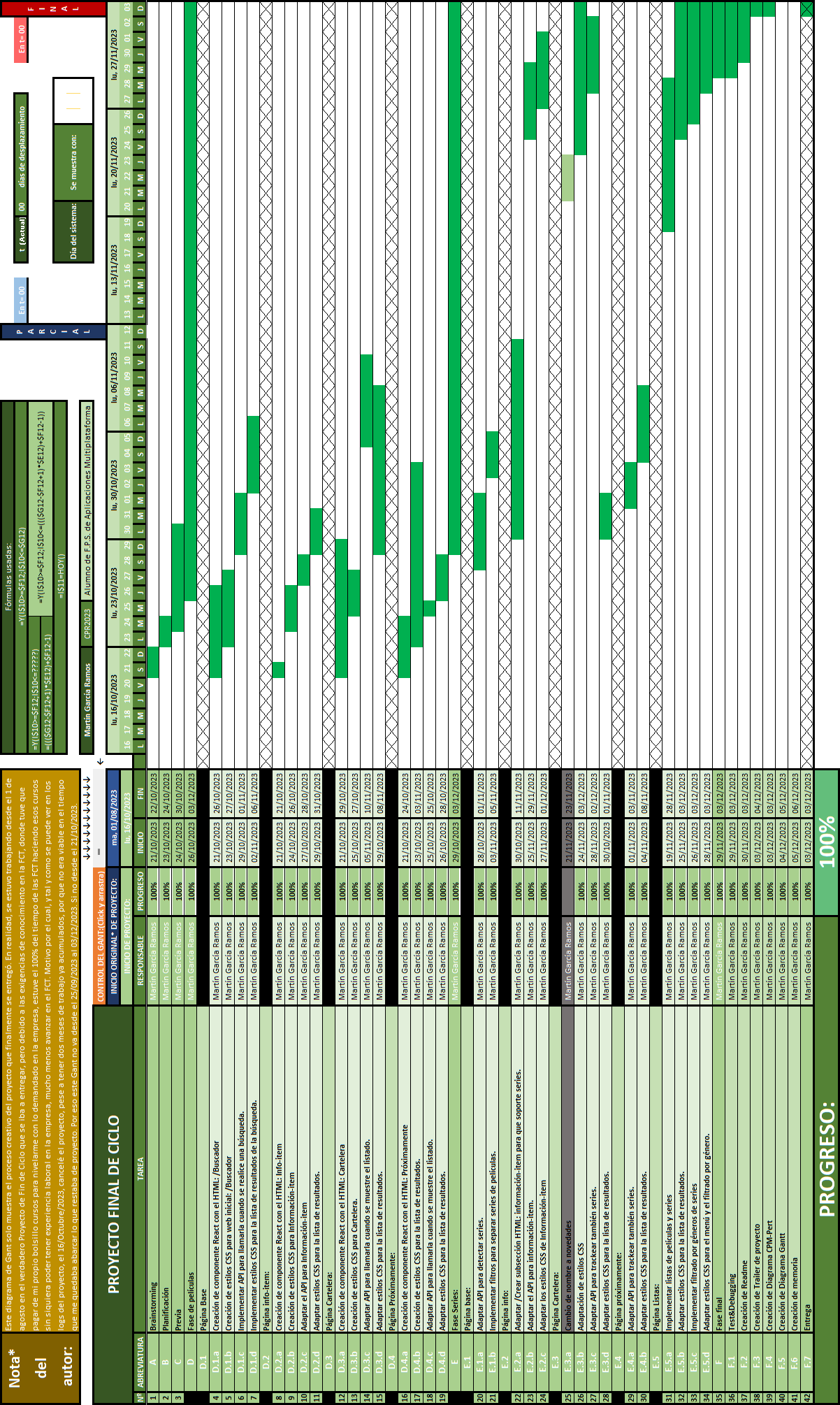
### 2.A.1. PERT:

Para ver en mejor detalle mirar el Excel adjunto.



### 2.A.2. GANTT:

Para ver en mejor detalle el Gantt mirar el Excel adjunto.



## 2.B. - RETOS ENCONTRADOS:

Esencialmente, el tener que desarrollar el proyecto en una tecnología desconocida en el momento del inicio del proyecto, provocó que a lo largo del desarrollo se tuvieran que rehacer diversas secciones varias veces y que el proyecto fuera más laberíntico de lo que ocurriría si se tuviera una planificación correcta.  
  
Además por otro lado, al usar una tecnología tan nueva como ReactJS, he tenido que enfrentarme a diversos problemas con el software externo, uno de ellos por ejemplo fue detectar porqué se producía el error que se indica a continuación cuando se accedía a la aplicación tras haber seleccionado otra pestaña en el navegador y volver a la pestaña donde está la aplicación.

***Uncaught runtime errors:***

***×***

***ERROR***

***Cannot redefine property: googletag***

***TypeError: Cannot redefine property: googletag***

***at Function.defineProperty (<anonymous>)***

***at <anonymous>:1:196***

***at <anonymous>:1:524***

Tras muchas pruebas y concluir, que el error no se debía a mi propio proyecto, hallé la solución en inhabilitar la extensión de navegador: Adblocker Stands. Por lo visto, es un error conocido desde Marzo de 2024.

# 3. – Recursos de Hardware y software

Al tratarse de una web, los únicos requisitos de hardware necesarios son aquellos que se demandan para ejecutar debidamente un navegador como Chrome o Firefox, estos son:

## 3.A. - Requisitos de Hardware:

### 3.A.1. Requisitos del navegador:

#### 3.A.1.a. Mínimos:

* **Procesador:** Procesador Intel Pentium 4 o AMD Athlon 64.
* **Memoria RAM:** 1 GB.
* **Espacio en disco duro:** 100 MB de espacio libre.
* **Resolución de pantalla:** 1280 x 1024.

#### 3.A.1.b. Recomendados:

* **Procesador:** Procesador Intel Core i3 o AMD Ryzen 3, o superior.
* **Memoria RAM:** 4 GB o más para un rendimiento óptimo.
* **Espacio en disco duro:** 500 MB de espacio libre o más para la instalación y almacenamiento en caché.
* **Resolución de pantalla:** 1920 x 1080 o superior para una experiencia de navegación de alta definición.

### 3.A.2. Requisitos para ejecutar el servidor de ReactJS:

#### 3.A.2.a. Requisitos Mínimos:

* **Procesador:** Procesador compatible con arquitectura x86 o x64.
* **Memoria RAM:** 2 GB o más.
* **Espacio en disco duro:** Al menos 100 MB de espacio libre para la instalación de Node.js y ReactJS, aunque el espacio requerido dependerá del tamaño de tus proyectos y las dependencias que utilices.

#### 3.A.2.b. Requisitos Recomendados:

* **Procesador:** Procesador de al menos 2 núcleos, preferiblemente con capacidad para multihilos (Hyper-Threading o similar) para mejorar el rendimiento en tareas de compilación y ejecución.
* **Memoria RAM:** 4 GB o más para manejar proyectos de tamaño mediano a grande de manera eficiente.
* **Disco duro:** Un disco SSD (Solid State Drive) puede mejorar significativamente los tiempos de compilación y carga del servidor debido a su velocidad de acceso más rápida en comparación con un disco duro HDD (Hard Disk Drive).

## 3.B. - Requisitos software:

Cualquier navegador debería de ser capaz de ejecutar la web indistintamente, aunque esta fue testeada con Chrome y Firefox, así que cualquier navegador que use el motor web: Blink o el motor Gecko, así como tener el intérprete de JavaScript V8 o superior.

### 3.A.1. Requisitos del navegador:

#### 3.A.1.a. Mínimos:

* **Sistema operativo:** Windows 7 o posterior, macOS X 10.10 o posterior, Ubuntu 14.04 o posterior, Debian 8 o posterior, openSUSE 13.3 o posterior, Fedora Linux 24 o posterior.

#### 3.A.1.b. Recomendados:

* **Sistema operativo:** La última versión disponible es recomendada para garantizar la seguridad y la compatibilidad con las últimas tecnologías web.

### 3.A.2. Requisitos para ejecutar el servidor de ReactJS:

#### 3.A.2.a. Requisitos Mínimos:

* **Node.js:** ReactJS se ejecuta sobre Node.js, por lo que necesitarás instalar Node.js en tu sistema. Se recomienda tener al menos la versión LTS (Long Term Support) para garantizar la estabilidad y compatibilidad. Puedes obtener Node.js desde su sitio web oficial: nodejs.org.

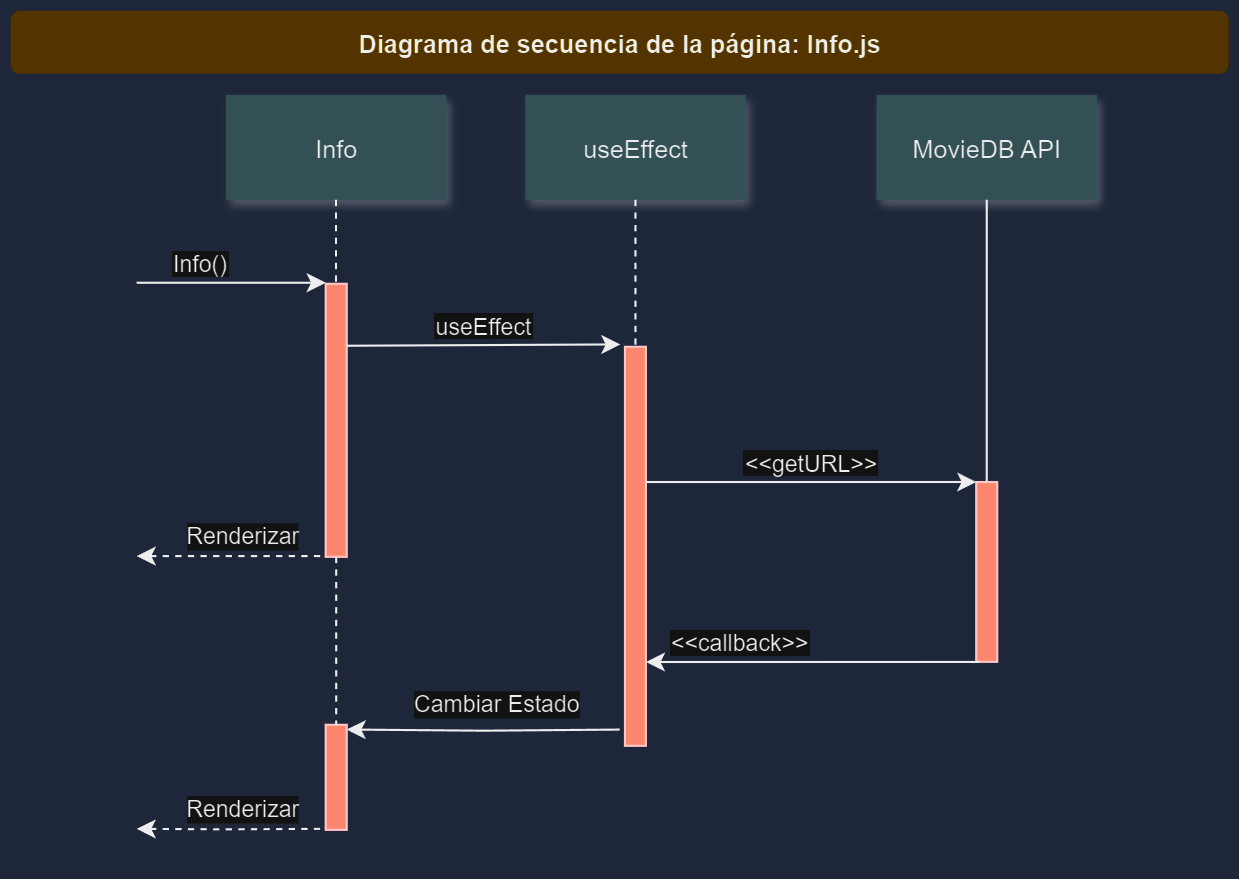
#### 3.A.2.b. Requisitos Recomendados:

* **Gestor de paquetes npm:** npm es el gestor de paquetes de Node.js y se utiliza para instalar y administrar las dependencias de los proyectos de ReactJS. Se recomienda tener la última versión de npm instalada junto con Node.js.
* **Editor de código:** Se recomienda utilizar un editor de código con características que faciliten el desarrollo en ReactJS, como Visual Studio Code, Atom o Sublime Text.
* **Sistema de control de versiones:** Utilizar un sistema de control de versiones como Git para gestionar el código fuente de tus proyectos.

# 4. – Arquitectura software y de sistemas

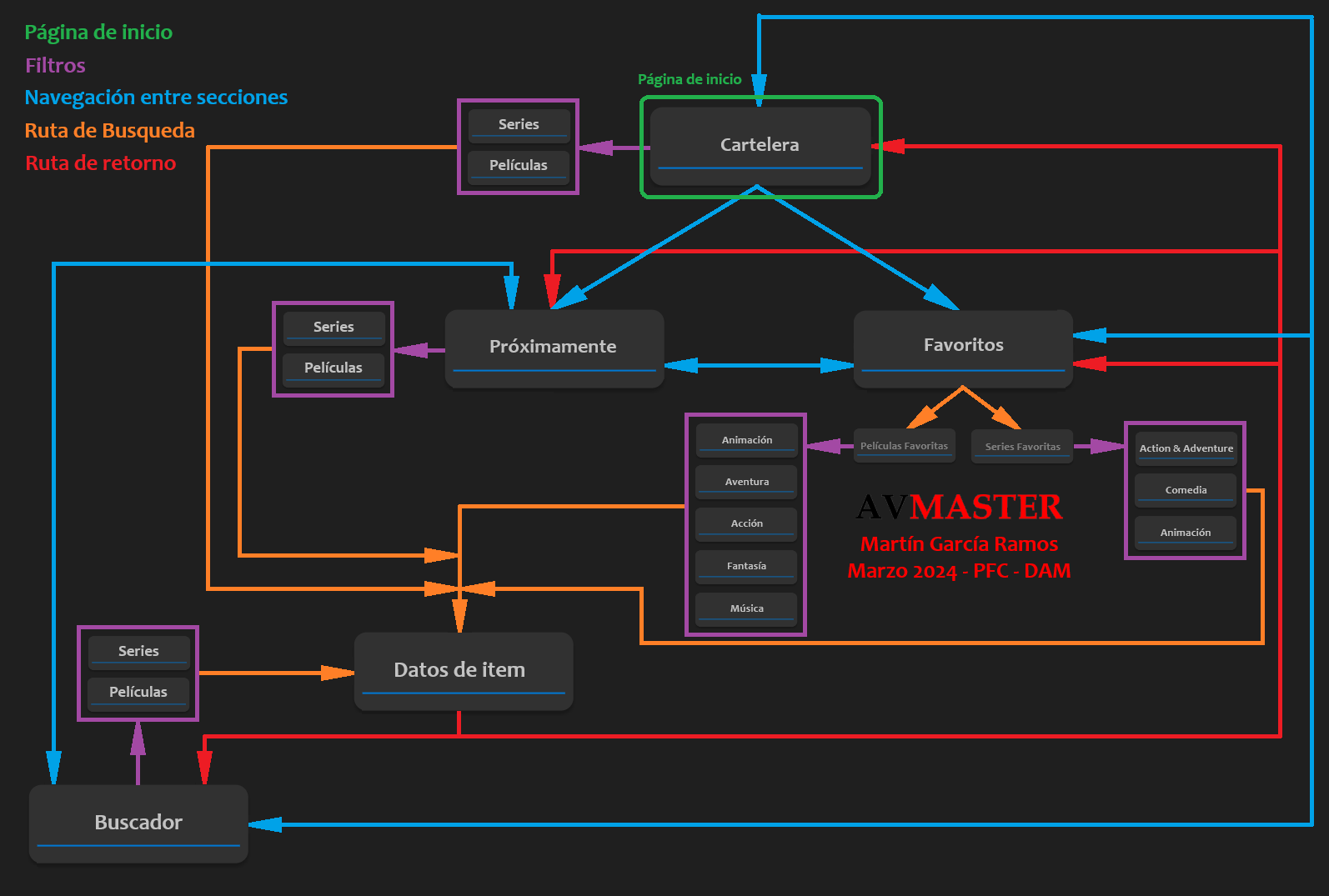
Mi proyecto al ser una web en ReactJS, no tiene “clases” o una orientación a objetos persé, tiene algo parecido a lo que se les denomina componentes y helpers; que en función a su propósito se le puede subdividir en helpers, pages, etc.…

## 4.A. - Diagrama de Secuencia:



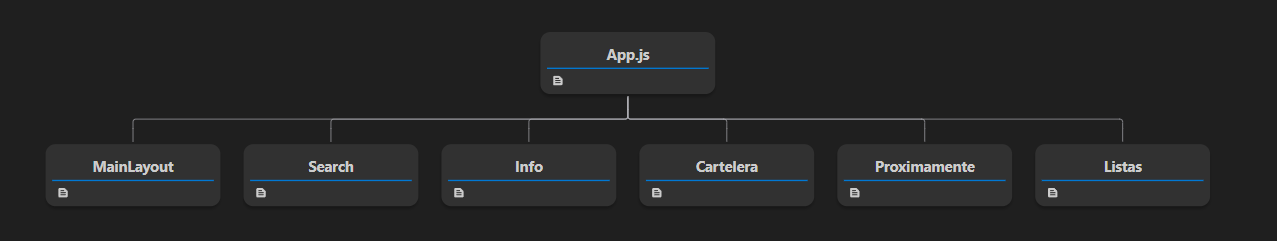
En este diagrama se puede ver cómo se envía Info(), a su vez Info llama al useEffect y este hará una petición asíncrona para obtener los datos de la película. Y mientras tanto, el Info devolverá el renderizado de la páginas de información.  
Cuando el MovieDB devuelva los datos, el useEffect hará una petición para cambiar el estado de la variable película y por ende se vuelve a realizar una petición para volver a renderizar la página de Info con los nuevos datos actualizados.

## 4.B. - Diagrama de Navegación:



## 4.C. - Diagrama de Jerarquía de componentes en cada página:

El proyecto consta de una aplicación que se reparte sus funcionalidades en los siguientes pages y layout, siendo llamados estos desde App.js mediante el Route:

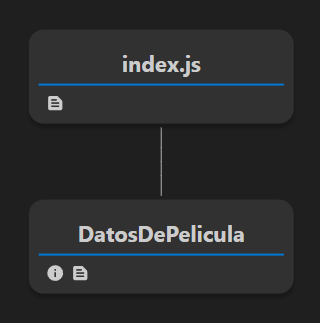


El **MainLayout** determina las características de la barra superior de la aplicación.

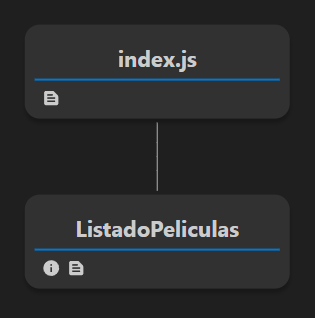
El **Search** por su parte, determina las propiedades del recuadro de búsqueda superior en su index que a su vez conecta con el componente ListadoPeliculas, que es el que se encarga de listar cada uno de los ítems, bien sean películas o series que se estén devolviendo en este caso como resultado de la búsqueda.



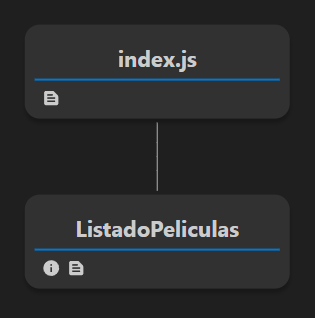
El **Info** por su parte, se encarga de obtener información sobre una película o serie de televisión desde una API, y luego renderiza esta información utilizando el componente DatosDePelicula.



El **Cartelera** muestra una lista de películas o series dependiendo del tipo seleccionado por el usuario, haciendo solicitudes a una API y mostrando los resultados utilizando componentes personalizados. El usuario puede cambiar entre películas y series seleccionando el tipo deseado; conectando a su vez con el componente DatosDePelicula para mostrar los datos del item seleccionado según proceda.



En **Próximamente**, gracias a su index, se muestra una lista de películas o series próximamente disponibles, haciendo solicitudes a una API y mostrando los resultados utilizando el componente de ListadoPelículas que también se reutiliza en Search y Cartelera. El usuario puede cambiar entre películas y series seleccionando el tipo deseado.



En **Listas** se encarga de la presentación general de las listas de favoritos y las opciones de filtrado, mientras que el ItemFavorito se encarga de representar cada elemento favorito individual y proporcionar opciones para interactuar con ellos, como eliminarlos de la lista.



En todas las páginas, siempre se puede volver a la página de inicio iterando con los respectivos botones del **MainLayout**.

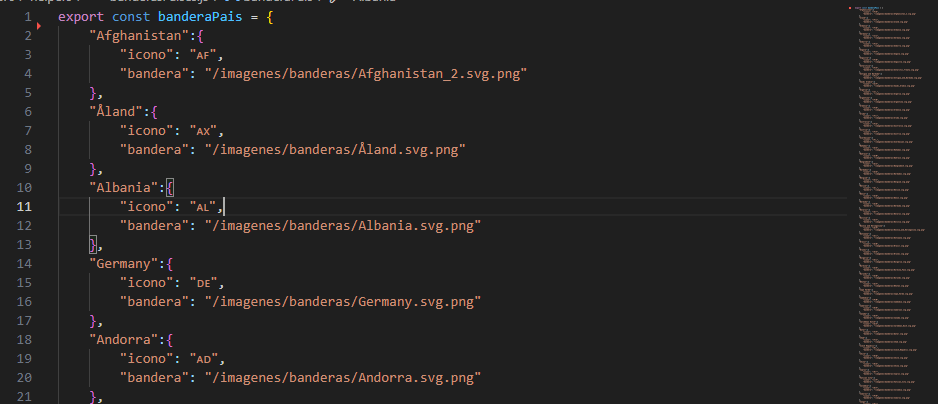
# 5. – Descripción de datos

No tengo mucho tiempo para extenderme con las muchas partes del código que me parecen interesantes, pero una que fue especialmente tediosa de realizar fue la del helper banderasPaises.js

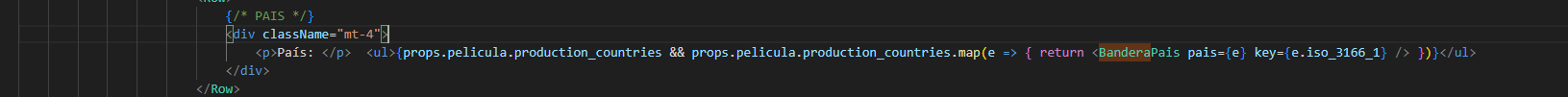
Quería que cuando el usuario accediera a la información de una película o una serie apareciera el país de procedencia, pero el api que usaba devolvía los idiomas todos en inglés, así que tomé la decisión de usar emojis para así evitarme el tener que buscar todas las banderas una a una. En el archivo aún hoy día se pueden ver comentados la primera versión en español del diccionario que hice, así como su posterior traducción al inglés del listado. Esto lo hice así porque asegurándome tener todos los países del mundo ordenados alfabéticamente en español luego si algún país tenía una mala traducción o una sentencia incorrecta lo podría localizar fácilmente.

Lamentablemente, y aunque la idea parecía ser una buena idea en un principio lo cierto es que en la práctica, los emojis solo mostraban dos letras, y en muchos casos como el de Kosovo (🇽🇰) el emoji muestra letras que tal vez corresponden al nombre del país en su propia lengua más que en español o en inglés.  
  
Por esa causa, tuve que tomarme las molestias de buscar todas las banderas de todos los países del mundo, tanto estados soberanos como no, con la finalidad de que apareciera la bandera en lugar del emoji con las dos letras.

La idea era que al poner el cursor encima de la bandera apareciera un simple texto emergente aclaratorio mostrando el nombre completo del país, pero como no tenía más tiempo al final lo dejé así:



Siendo esto llamado desde el componente: DatosDePelicula.js (Línea 223-225)



En resumen, una parte, sobre el papel sencilla que me supuso un trabajo inesperadamente arduo y repetitivo.

# 6. – Documentación adjunta

## 6.A. GITHUB:

<https://github.com/AGuyLearningCode/pfc_2023>

## 6.B. DIAGRAMAS:

Están adjuntos en el zip que se entregó en classroom, así como también están subidos en la carpeta Docs del proyecto en el mismo repositorio de GitHub.

6.C. TRAILER:

En el siguiente enlace está disponible en alta calidad.

<https://drive.google.com/file/d/1vjVgibddjwg5xZZsF7gHrEj4gn_Wn5ks/view?usp=drive_link>

No obstante, también está subido al repositorio de GitHub una versión comprimida.