

# Proyecto final Ingeniería Web: **KINO**

26/05/2023

---

Adrian Guiral, Iñigo Gastón y Fermín Elcano upna

<b>1. Introducción.....</b>	<b>3</b>
a. Presentación de las personas integrantes del grupo.....	3
b. Nombre, descripción y objetivo del proyecto.....	3
c. Target de usuarios del servicio/app/página web.....	3
<b>2. Prototipado.....</b>	<b>4</b>
a. Prototipos iniciales y explicaciones de interacción.....	4
b. Comparación de prototipos con resultado final.....	5
<b>3. Detalles de análisis y diseño.....</b>	<b>9</b>
a. Modelo (DB).....	9
b. Patrones de diseño.....	9
c. Diagrama UML.....	11
<b>4. Detalles de implementación.....</b>	<b>12</b>
a. Patrones de diseño e implementación.....	12
b. APIs.....	13
c. Librerías y frameworks (en caso de utilizar).....	15
d. Detalles de implementación curiosos/complejos.....	15
<b>5. Detalles de despliegue.....</b>	<b>16</b>
<b>6. Rendimiento.....</b>	<b>17</b>
a. Pruebas realizadas.....	17
b. Resultados obtenidos.....	17
c. Modificaciones propuestas.....	18
<b>7. SEO.....</b>	<b>19</b>
a. Destacar aspectos tenidos en cuenta sobre SEO durante el desarrollo.....	19
b. Pruebas/auditorías realizadas.....	19
c. Resultados obtenidos.....	20
d. Modificaciones propuestas.....	21
<b>8. Accesibilidad.....</b>	<b>22</b>
a. Destacar aspectos tenidos en cuenta sobre accesibilidad durante el desarrollo.....	22
b. Pruebas/auditorías realizadas.....	24
c. Resultados obtenidos.....	25
d. Modificaciones propuestas.....	27
<b>9. Usabilidad.....</b>	<b>29</b>
a. Destacar aspectos tenidos en cuenta sobre usabilidad durante el desarrollo.....	29
b. Pruebas/auditorías realizadas.....	32
c. Resultados obtenidos.....	32
d. Modificaciones propuestas.....	33
<b>10. Gestión de configuración.....</b>	<b>34</b>
a. Uso de repositorios.....	34
b. Integración continua.....	35
c. Trabajo en equipo y reparto de responsabilidades.....	35
<b>11. Conclusiones.....</b>	<b>37</b>
<b>12. Anexos.....</b>	<b>37</b>

# 1. Introducción

## a. Presentación de las personas integrantes del grupo

Iñigo Gaston, Adrian Guiral, Fermin Elcano.

## b. Nombre, descripción y objetivo del proyecto.

Nombre de la aplicación web: KINO.

¿Tienes muchas ganas de ver una película que te han recomendado? Puedes ver el rating general que tiene esa película, ver si está dirigida por un director/a conocido, los actores principales.... Mira las reviews que ha escrito la gente a ver si merece la pena.

El objetivo era crear una aplicación web para que los usuarios puedan registrar las películas que ven y compartir reviews, así como buscar información acerca de películas y series que más les gustan. Es decir, una rápida y sencilla wikipedia interactiva de películas.

## c. Target de usuarios del servicio/app/página web

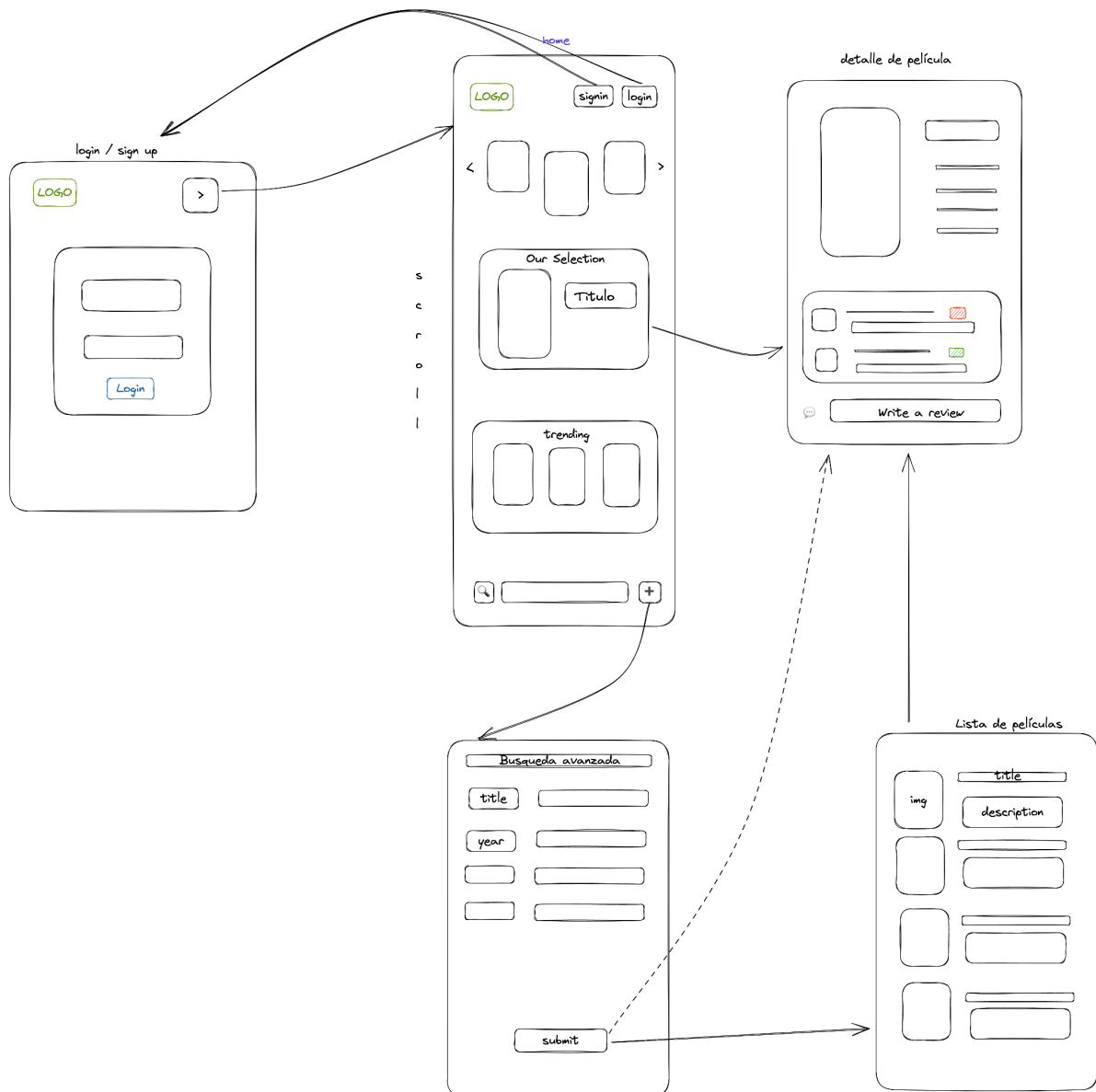
El target de usuarios son entusiastas del cine, gente que busca una aplicación gratis sin anuncios, donde poder dejar reflejada su opinión. Se estima que la demografía que usaría esta aplicación serían adultos jóvenes. Es decir, de entre 16 y 39 años. Con tiempo libre, conocimiento de la tecnología y familiarizados con aplicaciones similares en las que también se pueden dejar opiniones, reviews, etc.

Es posible que, como en otros casos de aplicaciones web similares, algún usuario confunda el uso real de la web, y se piense que a través de la aplicación se pueden ver las películas que busca, como si de un servicio de streaming se tratase. Cuando el uso real consiste en ver información relativa a películas y dejar opiniones, compartiendo experiencias con el resto de usuarios. Esto suele ocurrir a menudo, y para evitarlo, se ha especificado que no es una web de streaming y no hay ningún reproductor de vídeo integrado.

## 2. Prototipado

### a. Prototipos iniciales y explicaciones de interacción.

La visión general de la página web ha sido simplista, pero no necesariamente simple. La estructura que deseamos que tenga es la clásica de: Login, página principal con opción de búsqueda y resultados de búsqueda. El diseño que nos ha ayudado a llevar a cabo esta visión es el siguiente:



Para navegar de una página a otra siempre hay un elemento que ayude a realizar dicho desplazamiento. Desde el inicio de sesión al registro (por si el usuario no existe). Desde la página principal hacia la búsqueda avanzada mediante el botón con la lupa con el signo (+). Desde la página principal o resultado de búsqueda hacia una película en concreto haciendo

click en ella. Y desde cualquier página hacia la principal haciendo click en el icono de central de la web.

Como se puede comprobar en el prototipo inicial, desde el principio se ha buscado un diseño que cumpla con los estándares de accesibilidad y usabilidad. Se trata de un diseño Mobile First, en cuanto al apartado visual, las primeras horas de trabajo de diseño se han destinado, en parte, a elegir una apariencia visualmente satisfactoria. Para ello se han utilizado diversas webs de apoyo ([REF1](#)), en las cuales se puede ver ejemplos de:

- Paletas de colores útiles para diseños de móvil. Eligiendo ciertos colores, recomienda otras posibles opciones para la paleta final. Dando lugar a una combinación de colores que no choca al usuario, y que no distrae ni quita atención del propósito de la aplicación.
- Colores útiles para contraste, ya sea de imágenes, texto...
- Técnicas de CSS aplicadas al uso de colores y proporciones de elementos de una web que los utilizan.

## b. Comparación de prototipos con resultado final.

- Búsqueda avanzada

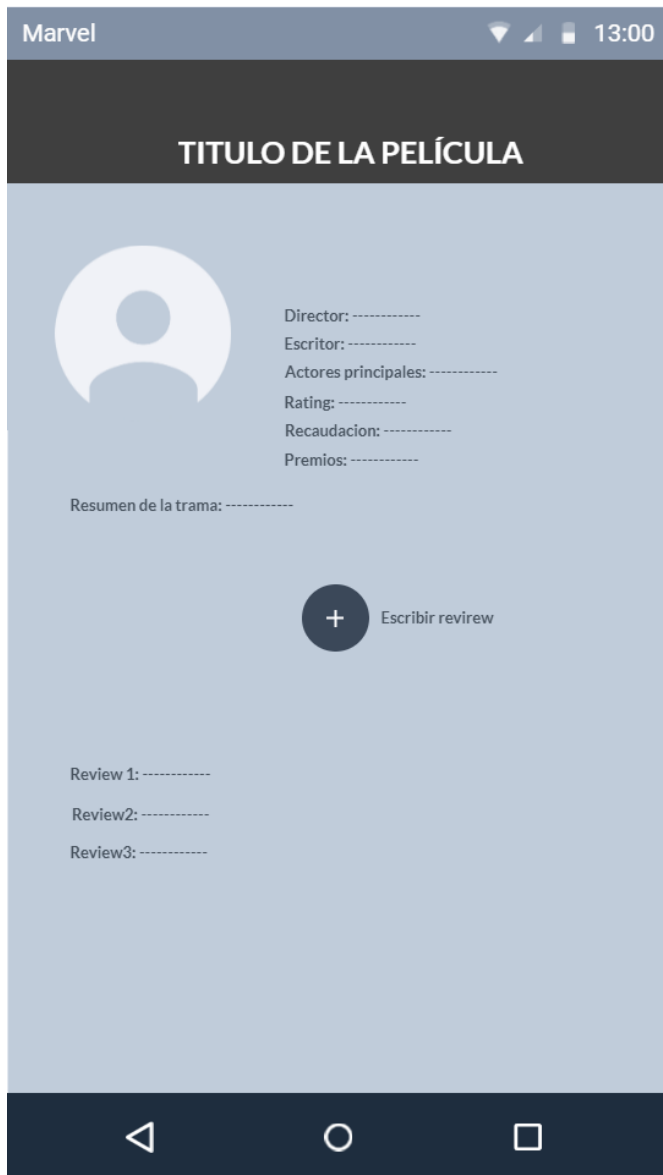
PROTOTIPO

RESULTADO FINAL

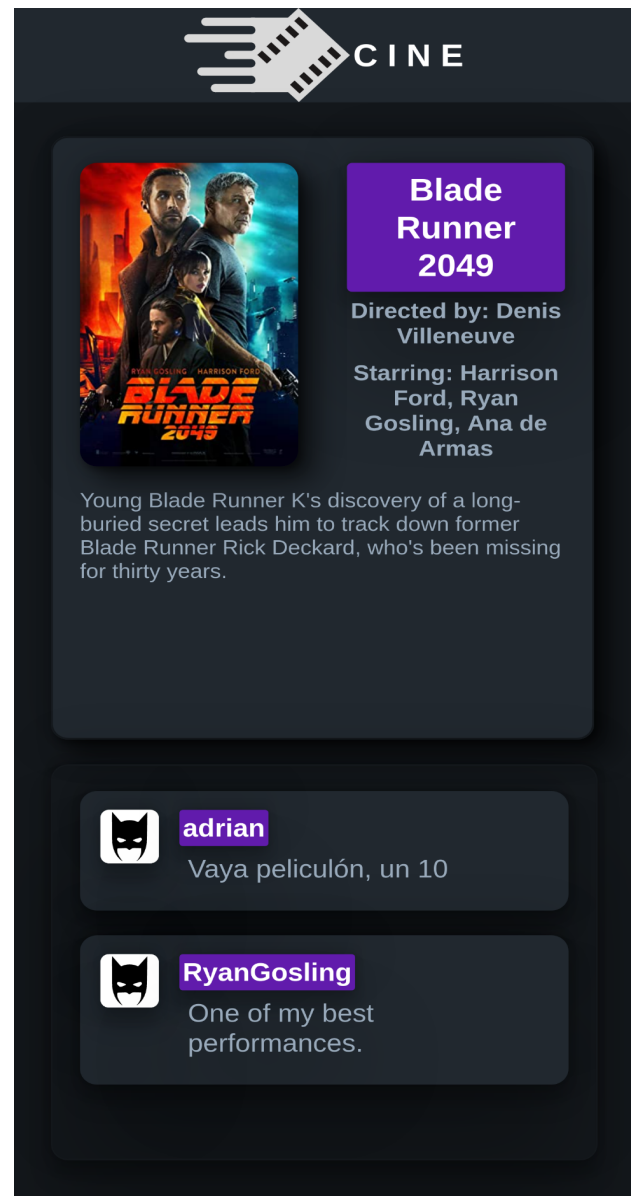
Posee un desplegable y más filtros (con opción de toggle)

- Página de película

## PROTOTIPO

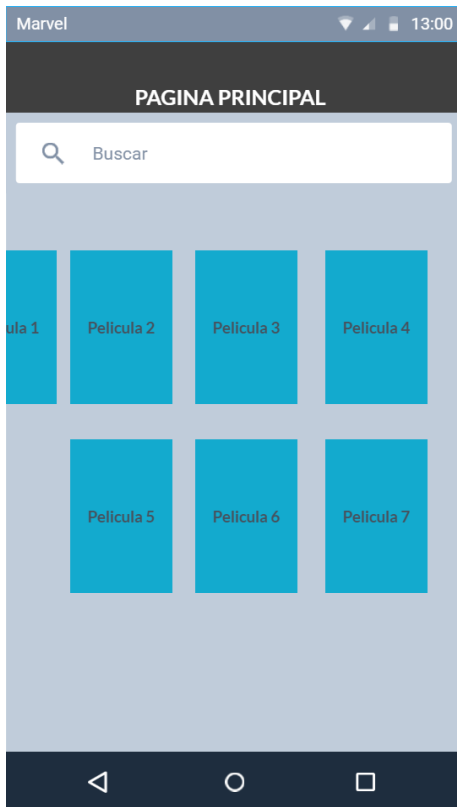


## RESULTADO FINAL

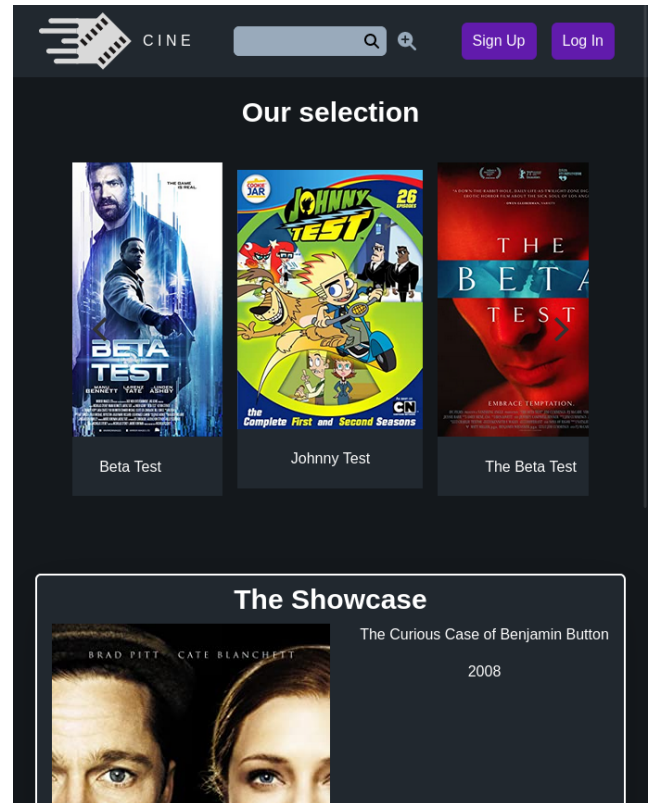


- Página principal

## PROTOTIPO



## RESULTADO FINAL

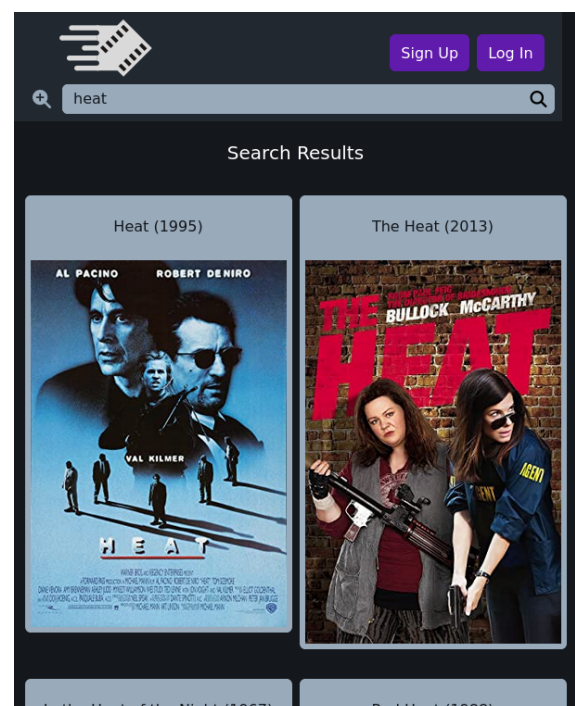


- Resultado de búsqueda

## PROTOTIPO

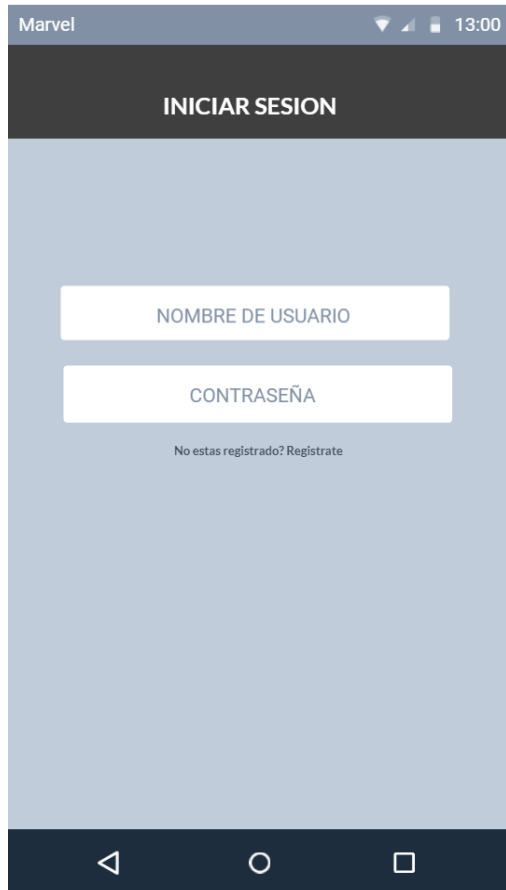


## RESULTADO FINAL



- Registro e inicio de sesión

## PROTOTIPO



Marvel 13:00

**INICIAR SESION**

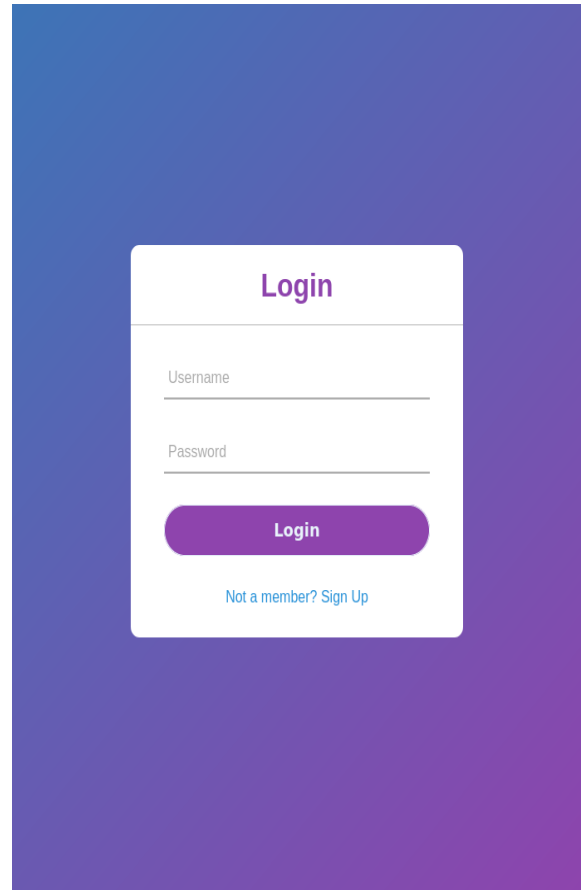
NOMBRE DE USUARIO

CONTRASEÑA

No estas registrado? [Registrate](#)

This is a mobile app prototype for a login screen. It features a dark grey header with the text 'INICIAR SESION'. Below the header, there are two white input fields for 'NOMBRE DE USUARIO' and 'CONTRASEÑA'. At the bottom, there is a link that says 'No estas registrado? Registrate'. The screen is framed by a dark blue bar at the bottom with standard Android navigation icons.

## RESULTADO FINAL



**Login**

Username

Password

**Login**

[Not a member? Sign Up](#)

This is the final design of the login screen. It has a blue-to-purple gradient background. A white card in the center contains the title 'Login' in purple. Below the title are two input fields labeled 'Username' and 'Password'. A purple 'Login' button is positioned below the fields. At the bottom of the card, there is a link that says 'Not a member? Sign Up'.

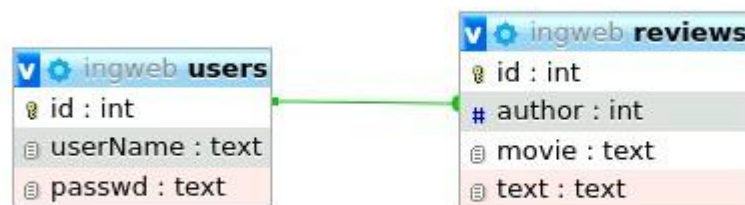


### 3. Detalles de análisis y diseño

#### a. Modelo (DB)

Para la base de datos y en nuestro caso se precisan de únicamente dos tablas: una para guardar los usuarios registrados en la web, los cuales pueden escribir reviews sobre películas y otra para las propias reviews.

La tabla de usuarios está compuesta por tres campos, un id autoincremental (clave primaria), el nombre de usuario y contraseña, la cual recibe un encriptado con md5 para que la base de datos no contenga información sensible de los usuarios. Por otro lado, la tabla de reviews tiene 4 campos; de nuevo un id autoincremental que actúa como clave primaria, el autor de la review, referencia al usuario, la película y el texto de la review.



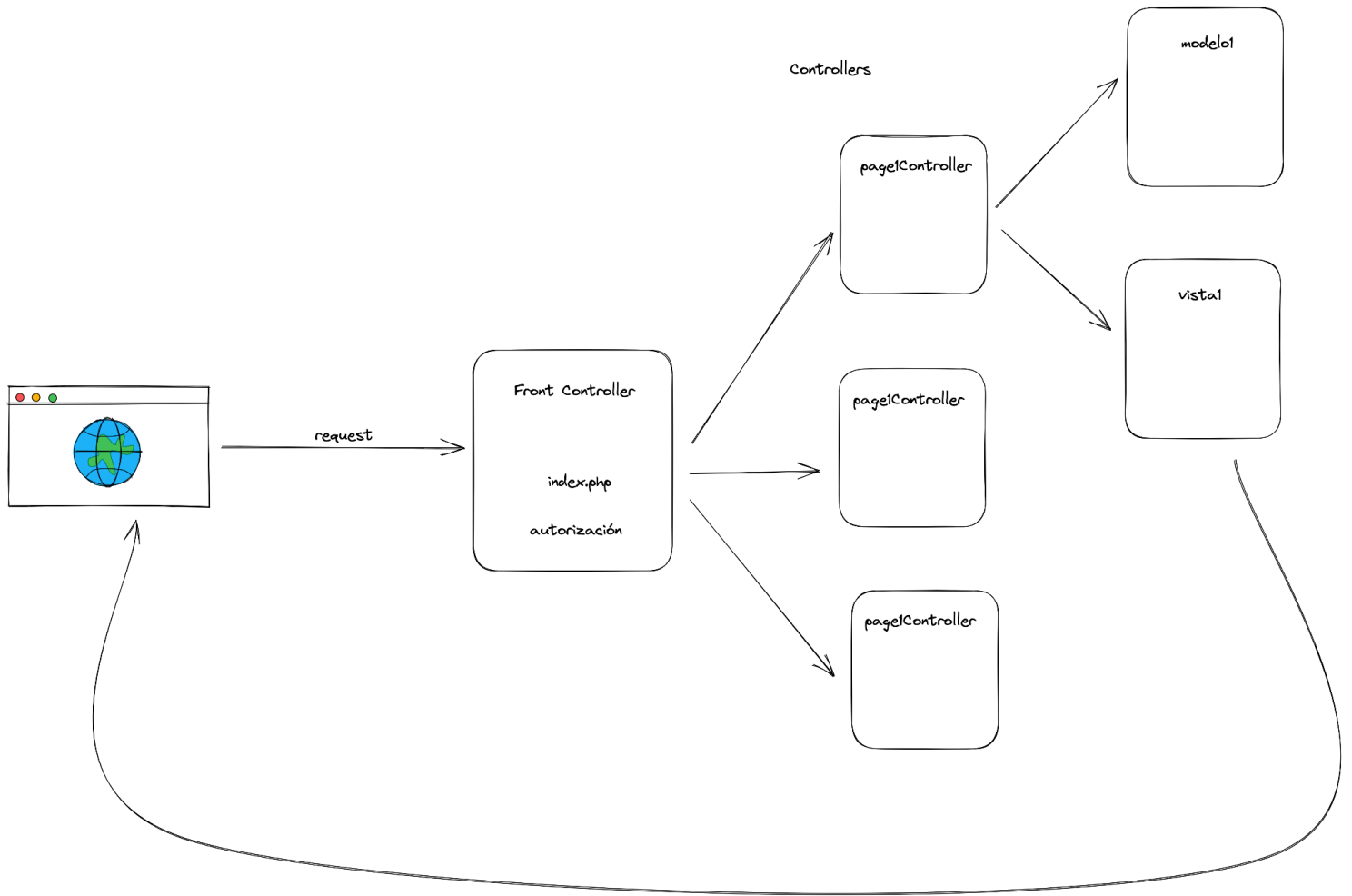
#### b. Patrones de diseño

Hemos utilizado el patrón del front controller, el cliente hace una petición que pasa por el archivo `/src/index.php`. Este archivo se encarga de redirigir al usuario al controlador específico de la página solicitada. Con estos controladores hacemos uso de uno o varios modelos y mostramos la vista al usuario.

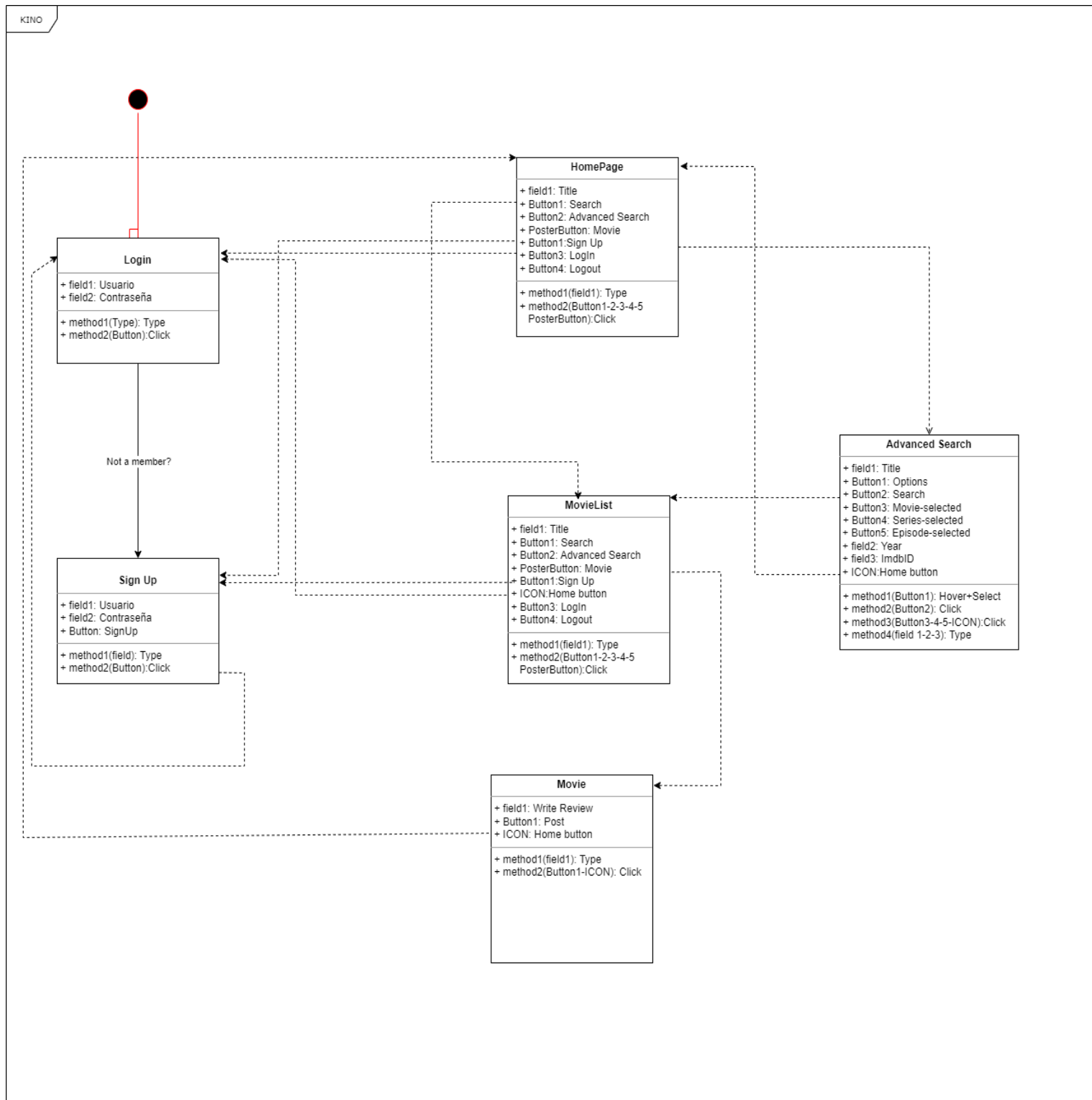
Existen redirecciones entre páginas como la página principal y el listado de películas. Para redireccionar al usuario utilizamos dos estrategias según nos sea más sencillo. Por un lado podemos usar la etiqueta html `form` para redirigir al `index.php` con un parámetro `page` que indique la página deseada. Por otro lado podemos capturar un evento mediante javascript y cambiar la url mediante la propiedad `window.location.href`. En ambos casos la dirección de destino ha de ser `index.php`.

Además se ha hecho uso del patrón de diseño “Singleton” para realizar la conexión a la base de datos.

Diagrama de patrón de diseño front controller:



## c. Diagrama UML



## 4. Detalles de implementación

### a. Patrones de diseño e implementación

Para implementar el “Singleton” se ha declarado una clase SingletonDB en php. Esta clase se construye usando el método getInstance() que revisa si ya se ha construido antes y de ser así devuelve la instancia ya creada. Al construirse realiza una conexión con la base de datos que contiene la información para el funcionamiento de la web.

```
1  <?php
2  class SingletonDB {
3      private static $instance = null;
4      private $connection;
5      private function __construct() {
6          $this->connection = mysqli_connect("localhost", "root", "starscourage", "ingweb");
7          //$this->connection = mysqli_connect("localhost", "root", "", "ingweb");
8      }
9
10     public static function getInstance(): self
11     {
12         if(self::$instance == null) {
13             self::$instance = new self();
14         }
15         return self::$instance;
16     }
17
18     public function getConnection() {
19         return $this->connection;
20     }
21 }
22 ?>
```

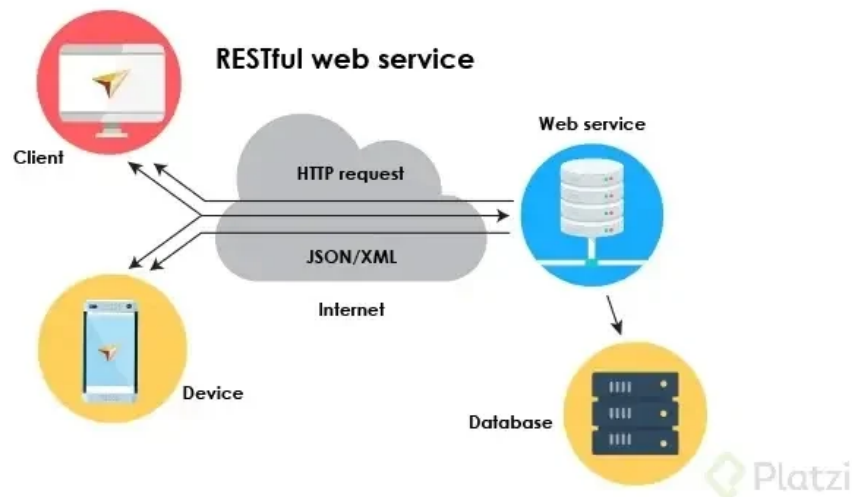
Durante la implementación, se ha mantenido una separación del modelo, la vista y el controlador para todos los archivos. Se ha distribuido en carpetas cada documento de tal manera que cada *pageController* llame a su *view* correspondiente, siguiendo la lógica del *model* respectivo.

En cuanto a ficheros externos, tales como fotos, símbolos, etc, se utiliza una carpeta llamada assets. Para las páginas de estilos se utiliza otra carpeta y para las funciones y llamadas de javascript se usa otra carpeta adicional. Cada vez que se llama a una de estos elementos, se utiliza una ruta relativa desde el sitio de origen de llamada.

```
> assets
> controllers
> css
> html
> js
> models
> views
```

## b. APIs

La web tenía que ser capaz de buscar cualquier película, serie u episodio, para que posteriormente el usuario la pudiese seleccionar y escribir su review. En eso se basa la mayor parte de su funcionalidad. Para ello, hacemos uso de la OMDb API [\[REF2\]](#), la “Open Movie Database”. Se trata de una api RESTful. Es decir, una arquitectura basada en REST, con interacciones muy simples y respuestas que se pueden recibir tanto en JSON como en XML.



Esta API tiene un límite de uso, ya que solo permite hacer 1000 peticiones diarias. Cada petición requiere de una API key. Si se realiza una petición y esta petición falla porque se ha agotado el uso permitido diario de dicha key, salta una excepción y se ejecuta esa misma petición con otra API key de las que hemos solicitado. Tras hacer una breve comprobación del funcionamiento de la API en Postman, ya la podíamos incorporar en el modelo correspondiente de nuestra aplicación web.

Las peticiones en formato URL que se realizan son de la siguiente forma:

- Se puede buscar una película, y filtrar mediante los siguientes parámetros:

Parameter	Required	Valid Options	Default Value	Description
i	Optional*		<empty>	A valid IMDb ID (e.g. tt1285016)
t	Optional*		<empty>	Movie title to search for.
type	No	movie, series, episode	<empty>	Type of result to return.
y	No		<empty>	Year of release.
plot	No	short, full	short	Return short or full plot.
r	No	json, xml	json	The data type to return.
callback	No		<empty>	JSONP callback name.
v	No		1	API version (reserved for future use).

- También se puede buscar una LISTA de películas, en base a ciertos parámetros:

Parameter	Required	Valid options	Default Value	Description
s	<input checked="" type="checkbox"/>		<empty>	Movie title to search for.
type	<input type="checkbox"/>	movie, series, episode	<empty>	Type of result to return.
y	<input type="checkbox"/>		<empty>	Year of release.
r	<input type="checkbox"/>	json, xml	json	The data type to return.
page <span>New!</span>	<input type="checkbox"/>	1-100	1	Page number to return.
callback	<input type="checkbox"/>		<empty>	JSONP callback name.
v	<input type="checkbox"/>		1	API version (reserved for future use).

Resumidamente, las dos peticiones primarias que se realizan en el modelo *MovieModel.php* son:

- La petición de una lista de películas:

El array de películas que devuelve contiene por cada una: un título, un póster y el IMDb ID correspondiente.

- Se realiza en la búsqueda principal de la “HomePage” y al ser una búsqueda por título (no muy preciso), devuelve una lista con todas las películas, episodios y series que coincidan con el título introducido.
- Se realiza también en la búsqueda avanzada. Con parámetros adicionales.

- La petición de información específica de una película.

Los parámetros que se pueden obtener de una única película son: El título, póster y IMDbID (como la anterior petición). Solo que ahora también se incluye el año, un resumen de la película, el director, el escritor, los premios que ha recibido, los actores principales, la duración, la puntuación de la película en IMDb, la recaudación en taquilla y el género.

- Se realiza una vez ya se ha obtenido la lista de películas, al hacer click sobre una de ellas, se obtiene la imdbID [\[REF3\]](#) de la película, así como el título. Entonces se realiza la petición. Con esos dos datos ya se puede encontrar toda la información relativa a esa película exacta.

### c. Librerías y frameworks (en caso de utilizar)

En el caso de la pantalla de inicio (*homeView.php*) hemos utilizado la librería [Tailwind](#). Esta librería permite definir los estilos css que se aplican a cada elemento html desde el propio html. Para ello escribimos en el atributo *class* de la etiqueta a la que queremos aplicar un estilo el nombre de una clase definida en la librería. Los nombres de estas clases suelen ser breves y estar relacionados con la transformación o estilo que aplicamos. Por ejemplo si queremos aplicar a un div un margen por la derecha escribiríamos algo así:

```
<div class="mr-2"></div>
```

Si nos fijamos hemos declarado el div con un atributo *mr-2*. Este se “traduce” como margin right 2. Internamente tailwind tiene definidas reglas para estas pseudo-clases, en el caso anterior la regla sería esta:

Este código css se genera en una fase de compilación. Para ello nos hemos instalado un [programa](#) para transformar nuestro php en código css. Tuvimos que crear un archivo de configuración especial (*tailwind.config.js*) para especificar que tipo de ficheros queremos compilar, en nuestro caso html y php. Todo el código css generado se guarda en el archivo *tailwind.css*. Las páginas que utilizan esta librería son la ya mencionada *homeView.php* y *listView.php*.

Las experiencias que hemos tenido con Tailwind en general ha sido positiva. En nuestra opinión Tailwind es una herramienta muy útil a la hora de considerar el diseño mobile-first.

Otra librería que hemos usado en *homeView.php* es [flickity](#) para el carrusel de películas. Descargamos esta librería, ficheros *flickity.pkgd.min.js* y *flickity.min.css* y en el código php solo declaramos un div con los ajustes que queremos para nuestro carrusel.

### d. Detalles de implementación curiosos/complejos.

La única complejidad encontrada ha sido el desarrollo Mobile First ya que es difícil adaptar una página web móvil que en principio dispone de menos espacio para trabajar mientras que la página web de ordenador dispone de espacio más que de sobra y en la mayoría de los casos había que buscar información adicional que mostrar para la página web de ordenador. Exceptuando este punto, no hemos encontrado ningún otro detalle complejo o curioso que merezca ser comentado.

## 5. Detalles de despliegue.

A la hora de desplegar nuestro código hemos utilizado una estrategia manual. Una vez hacíamos un cambio en el repositorio que quisiéramos desplegar en producción nos conectamos al servidor y bajamos los cambios. También comprobamos que todo funcionase como se esperaba antes de continuar con el desarrollo de otras características.

Intentamos crear una pipeline de gitlab para mejorar nuestra estrategia de despliegue pero no tuvimos éxito. Conseguimos crear dos workers en dos de nuestros servidores, queríamos que los workers hiciesen pull de los cambios establecidos. Ante cualquier cambio el servidor se descargaría la última versión de la rama “dev” (rama de producción). Esto nos supondría un ahorro al no tener que entrar tan a menudo a la máquina virtual para actualizar los cambios. No obstante no funcionó, creemos que fue por no poder autenticarnos en gitlab de manera automática. Intentamos crear ficheros con nuestras contraseñas para iniciar sesión de manera automática. Además de no funcionar esta opción es altamente insegura. La alternativa que nos hubiese gustado probar sería la autenticación mediante claves ssh.

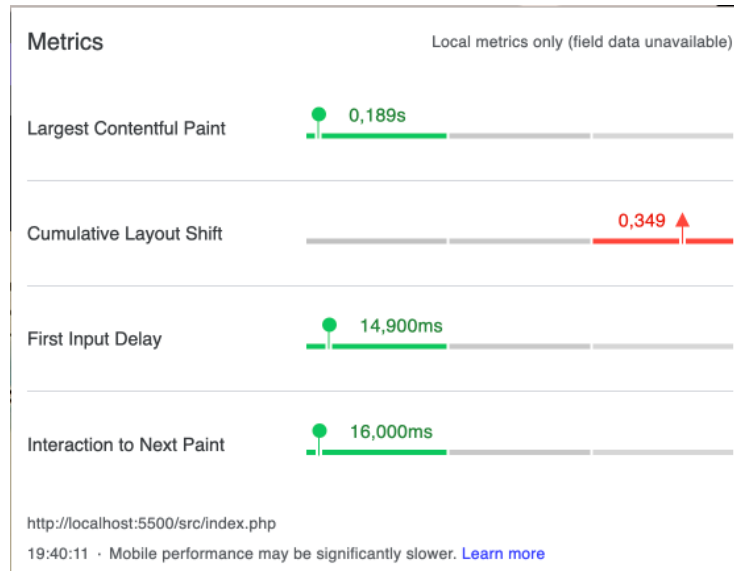
Consideramos que los recursos que ofrece este servidor podrían ser insuficientes llegados a un determinado número de usuarios. Por esto consideramos explorar otras opciones como las arquitecturas serverless u otros servicios en la nube para alojar nuestra página en el futuro.

Otras tareas pendientes serían la recuperación ante posibles pérdidas o caídas del servidor. Podríamos aprovechar la disponibilidad de nuestros tres servidores teniendo dos como repuestos. Esto exigiría planear una estrategia o protocolo ante situaciones en las que el servidor principal no pueda responder. También deberíamos considerar que los tres servidores pueden ser muy dependientes y en caso de un fallo técnico en las instalaciones los tres quedarían inutilizables.



## 6. Rendimiento.

### a. Pruebas realizadas.



Hemos utilizado la extensión de Google Chrome Web Vitals para monitorizar el rendimiento de la página. Lo que mostramos en la figura de arriba son las métricas que obtenemos para la página principal (homeView). Es importante destacar que las pruebas se han hecho contra un servidor local. Para obtener unos resultados más próximos a los reales habría que realizar las mismas pruebas contra el servidor de producción.

### b. Resultados obtenidos.

En cuanto a los resultados de las pruebas anteriores vemos que tres de las cuatro métricas obtienen una excelente nota. La primera de ellas, Largest Contentful Paint, se refiere al tiempo de carga hasta que el usuario visualiza nuestra página. Es lógico que obtengamos un tiempo relativamente corto ya que la página no es excesivamente grande aunque es cierto que muestra unas cuantas imágenes. Otro motivo puede ser la distancia hasta el servidor, como ya hemos comentado estamos trabajando y probando en local.

Quizás la métrica que más merece la pena comentar es el Cumulative Layout Shift. Está es la métrica en la que peor nota obtenemos y con mucha diferencia. Esta prueba mide el nivel de “movimiento” que experimenta la página mientras carga. Se considera una mala experiencia de usuario que los elementos de las páginas se muevan mucho de posición. Aun así creemos tener una explicación para este mal resultado. Puede que el *carrusel* de pósters de películas esté tergiversando la métrica haciéndola parecer peor de lo que es. Este contenedor de imágenes hacen que se muevan por tanto tiene sentido que se detecte como un desplazamiento indeseado. Sin embargo no tenemos razones para pensar que esto pueda suponer una peor experiencia de usuario. Para comprobar esto último hemos realizado una prueba más.

Usando Google Chrome hemos limitado los recursos disponibles para ver una carga más lenta y monitorizada. Las opciones que describimos a continuación se encuentran en la pestaña Performance del citado navegador. Aquí limitamos la conexión a internet (slow 3g) y los hilos de cpu disponibles los fijamos a 1. También seleccionamos la opción Screenshots para ver como va cambiando la página mientras los elementos se van cargando. Haciendo esto observamos que la página no sufre cambios o desplazamientos verticales a consecuencia del carousel. La primera vez que se renderiza en pantalla vemos que los elementos ya están en su posición aunque no hayan cargado las imágenes.

Así es como se ve la página en ese primer momento, podemos ver como la disposición de los elementos no va a cambiar. Además podemos ver como aun no se han cargado las imágenes de las películas ni el logo de la aplicación. Con esto creemos estar bastante seguros de estar interpretando la métrica de manera correcta.

### c. Modificaciones propuestas.

Una de las modificaciones que hemos hecho es eliminar los archivos más pesados. Para esto utilizamos las herramientas para desarrolladores de Google Chrome. Esta vez en la pestaña Network, desactivando la caché y probando varias velocidades de conexión. Si utilizamos la máxima velocidad el archivo que más tiempo tarda en cargar es index.php. No obstante si utilizamos un fast 3G o cualquier otra velocidad inferior a esta observamos resultados diferentes. En este caso los archivos más pesados son los que más tiempo tardan. Hablamos de la fuente Raleway-Light.ttf. Por tanto una posible optimización es eliminar este archivo, reducirlo o buscar alternativas. En nuestro caso decidimos eliminarlo y quedarnos con una fuente que no requiera ser enviada por el servidor.

## 7. SEO.

### a. Destacar aspectos tenidos en cuenta sobre SEO durante el desarrollo.

Nada de iframes, embeds, objects usados, se han diseñado todas las páginas de cara a la práctica mobile first, ya que diversas métricas utilizadas por google priorizan estos sitios.

Se han añadido etiquetas <meta name=description a cada página.

### b. Pruebas/auditorías realizadas.

No se ha podido utilizar ninguna comprobación o prueba usando las páginas más comunes para comprobar el SEO de la aplicación web, ya que la web está desplegada en la red interna de la universidad, usando una máquina virtual.

Para obtener algún tipo de información relativa al SEO de nuestro servicio, hemos recurrido a extensiones del navegador firefox. De tal manera que se realizan las comprobaciones mientras la web esta corriendo. Para ello hemos utilizado la extensión SEOquake [\[REF5\]](#);

La extensión analiza:

- La URL.
- Título.
- Metadescripciones.
- Palabras clave meta.
- Encabezados.
- Texto alternativo en imágenes.
- Relación Texto/HTML.
- Compatibilidad con dispositivos móviles.
- Meta viewport.
- Cumplimiento del sitio.

## c. Resultados obtenidos.

URL	✓	79 caracteres — óptimo. eim-alu-69044.lab.unavarra.es/grupo-ocelote/src/index.php?page=home&action=home	Consejos▼
Canónico	🔊	No se ha configurado ninguna etiqueta canónica para este página.	Consejos▼
Título	🔊	7 caracteres — medio. El número óptimo es entre 10 y 70 caracteres. K I N O	Consejos▼
Metadescripción	⚠	0 caracteres — trata de ampliar la descripción a 160 caracteres.	Consejos▼
Palabras clave meta	🔍	0 caracteres, 0 palabras.	Consejos▼
Idioma	✓	¡Genial! Has especificado el idioma de tus sitios web. en	Consejos▼
Tipo de doc	✓	¡Genial! Has especificado el tipo de documento. HTML5	Consejos▼
Codificación	✓	Si se declara una codificación de caracteres/idioma se fortalecerá considerablemente la SEO. También evita complicaciones cuando se visualiza la página. UTF-8	Consejos▼
Google™ Analytics	⚠	Google™ Analytics no está supervisando tu sitio web. Te sugerimos que saques el máximo partido de esta increíble herramienta.	Consejos▼
Favicon	✓	¡Qué bien que tienes una imagen favicon! http://eim-alu-69044.lab.unavarra.es/grupo-ocelote/src/assets/cinema-center.png	Consejos▼
Imágenes	✓	Todas las imágenes tienen el atributo ALT.	Consejos▼
Relación Texto/HTML	⚠	10.87 % — ¡Ooohh! La relación de texto - código HTML de tus sitios web es inferior a 15 %. Te sugerimos añadir mucho más texto a tu sitio web.	Consejos▼
Marcos	✓	No detectado	Consejos▼
Flash	✓	No detectado	Consejos▼

#### d. Modificaciones propuestas.

Hemos cambiado los títulos de las páginas para que sean únicos y, a excepción de la página principal tengan más de 7 caracteres. Incluidas nuevas metaetiquetas y labels que favorecen la inclusión de la página web en posiciones superiores de búsqueda.

También se ha incluido un párrafo extenso con palabras clave como “film”, “filmmaker”, “movie”, “cinema”, etc. Para captar de manera más eficientes *key words* de búsqueda.

## 8. Accesibilidad.

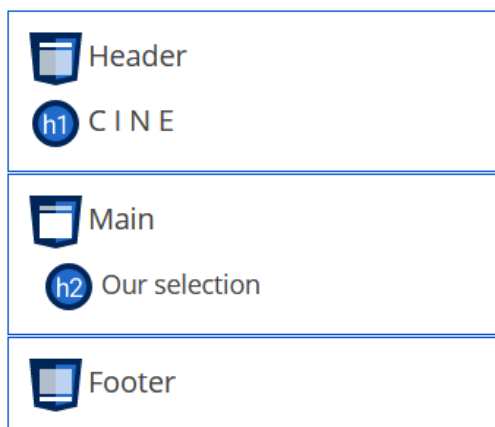
### a. Destacar aspectos tenidos en cuenta sobre accesibilidad durante el desarrollo.

Se han tenido en cuenta varios aspectos durante la realización de la aplicación web. Se ha intentado seguir las WCAG.

No se han incluido límites de tiempo, ni para formularios, ni para búsquedas ni para inicios de sesión ni registros. Se han escogido colores pensados para el contraste, como se ha mencionado en el apartado de diseño.

También nos hemos basado en más principios de accesibilidad. Estos han sido:

- Estructura del documento: Se basa en el uso de estructuras de orden para dividir la página. Tales como header, main, nav, footer para mostrar una vista clara del contenido. Y encabezados y sus etiquetas.



- Orden de navegación: Aunque está directamente relacionado con la propia estructura del documento, no siempre se navega de la misma forma al usar la web. Por ejemplo al usar tabulaciones y narradores. El uso de etiquetas labels y botones determina el orden por el cual el tabulador se va moviendo.

[Refresh Navigation Order](#)

[Show WAVE Icons](#)

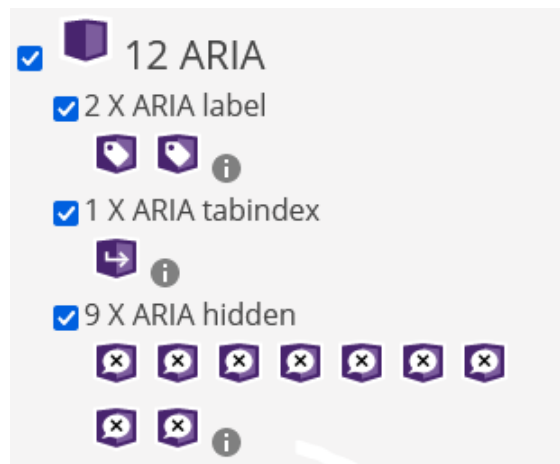
En ésta foto, se ve como el primer elemento de navegación que se recorre es el botón de búsqueda y luego el campo de texto de escribir la review

*Order, role, and accessible name (what is read by a screen reader) for all navigable page elements are listed. Elements that do not have a function should not be listed.*

1 Button: Search

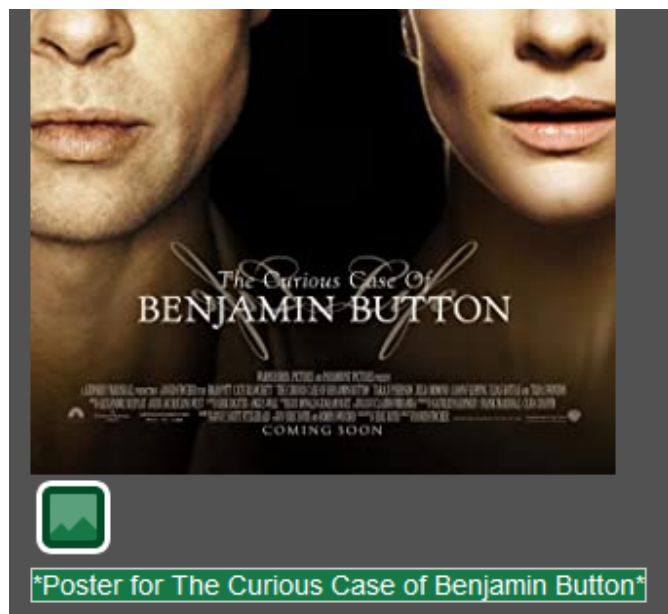
2 Textarea: Write review...

Para cierto contenido interactivo, como un *carrousel* de películas, se hace uso de ARIA para gestionar ciertos labels. La librería de javascript que se ha utilizado para acciones similares ([flickity](#)) también hace uso de ARIA.



- Uso de texto alternativo:

La mayor parte de nuestra aplicación web se basa en buscar películas. No hay nada más identificativo de una película que su poster. Cada póster de una película posee un texto alternativo que se modifica automáticamente y muestra el título original.



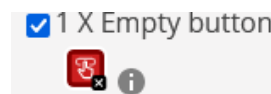
## b. Pruebas/auditorías realizadas.

Para las pruebas de accesibilidad, hemos utilizado la Web Accessibility Evaluation Tool (WAVE) [\[REF4\]](#). Concretamente en su formato de extensión de navegador, para poder ir navegando por nuestro sitio web viendo cada detalle que afecta negativamente y positivamente a la accesibilidad.


Una vez se han analizado los resultados de cada uno de los contenidos destacables de la aplicación web, se han corregido hasta que los escaneos de la extensión no mostraran los errores previos.

El procedimiento ha sido el siguiente:

- Ver el error



- Ver si el fallo está justificado y cómo solucionarlo

 **Errors**  
Empty button

**What It Means**  
A button is empty or has no value text.

**Why It Matters**  
When navigating to a button, descriptive text must be presented to screen reader users to indicate the function of the button.

**How to Fix It**  
Place text content within the <button> element or give the <input> element a value attribute.

**The Algorithm... in English**  
A <button> element is present that contains no text content (or alternative text), or an <input type="submit">, <input type="button">, or <input type="reset"> has an empty or missing value attribute.

**Standards and Guidelines**

- [1.1.1 Non-text Content \(Level A\)](#)
- [2.4.4 Link Purpose \(In Context\) \(Level A\)](#)

[Icon index](#)

- Ver el código erróneo gracias a la built-in function de la extensión

```
<button type="submit" class="mx-2">  
    
</button>
```

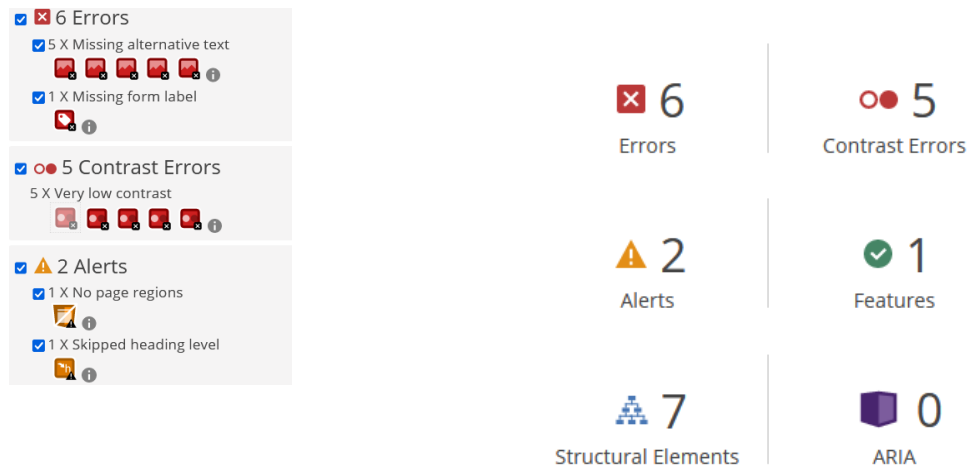
- Corregir el código y volver a repetir el escaneo de la página hasta que el error desaparezca



## c. Resultados obtenidos.

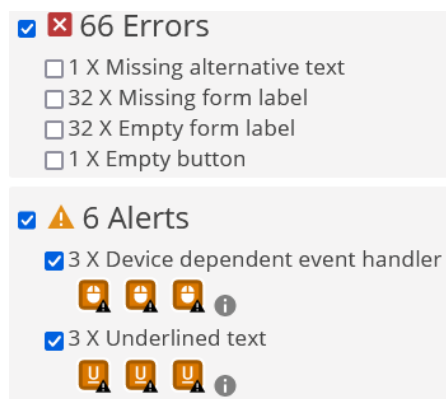
- Película individual

El problema en esta página es el orden de ciertos encabezados, ya que los nombres de los usuarios visibles en cada review presentaba un encabezado “h1”, mientras que los datos presentes en la parte superior de la vista presentaban un “h3”. Lo cual suponía un mal ejemplo de jerarquía y uso de etiquetas. Otros problemas menores eran un par de textos alternativos sin escribir y un contraste muy bajo entre un texto y el fondo.

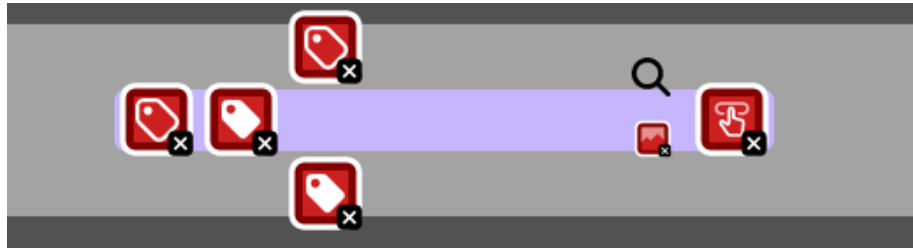


- Lista de películas

Los encabezados y la jerarquía se encuentran en perfecto estado, además, el uso del tabulador navega entre los elementos visibles. El resultado indica multitud de errores porque se trata de un error que se ejecuta cada vez que muestra uno de los resultados de búsqueda, de los cuales hay bastantes. El error es la ausencia de un label en cada película, que haga referencia a los parámetros que se recogen mediante un *form* para pasar a la pestaña siguiente.

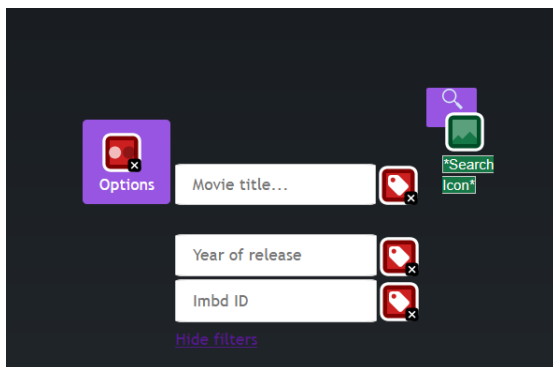


Las otras alertas, que no representan un error de por sí, se refieren al acceso de login y signUp, que al parecer requieren de ratón para hacer click. El texto alternativo que falta es el del icono de la lupa del campo de búsqueda.



- Búsqueda avanzada

Mostraba errores de contraste en los colores de los botones y faltaban labels para elementos del formulario.



**3 Errors**

- 3 X Missing form label
  -

**4 Contrast Errors**

4 X Very low contrast

-

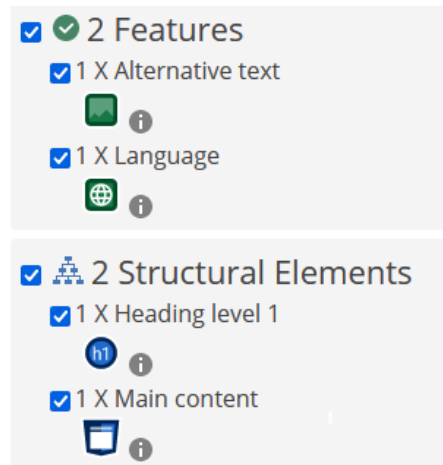
**1 Alerts**

- 1 X Missing first level heading
  -

## d. Modificaciones propuestas.

- Búsqueda avanzada

Se han dado colores con mayor contraste y se ha corregido un error de encabezado que requería un “h1”. También se ha incluido el idioma y se ha estructurado mejor el contenido.

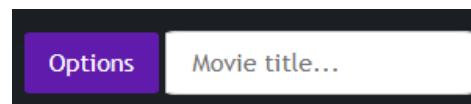


En cuanto a los labels, el error que detecta es que falta un nombre para el elemento del formulario en cuestión. Esto sucede porque el escáner de accesibilidad no detecta el placeholder del campo de texto a rellenar.

Por lo tanto en vez de recibir un:

<Movie Title> <Campo de texto>

Estaba recibiendo un :



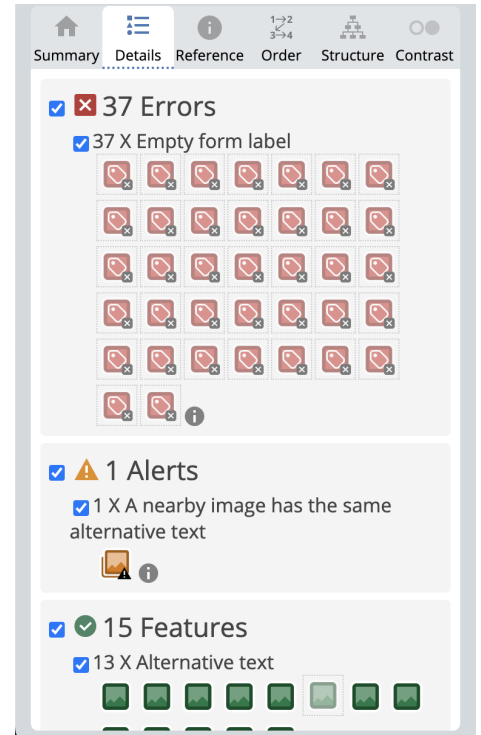
De todas maneras, se ha incluido un texto alternativo a todos los botones para evitar cualquier tipo de confusión.

- Home view

Hemos arreglado los errores Missing Form Label. El problema estaba en no utilizar la misma id en el input y el for del label. Para las listas la id utiliza un índice para no crear identificadores duplicados.

En cuanto a los errores de Empty form label la descripción del problema nos dice: *"Labels are not required for image, submit, reset, button, or hidden form controls"*. Lo que más nos interesa es el último elemento hidden form controls, elementos del formulario que están ocultos y solo se utilizan como variables. Este sería nuestro caso ya que usamos esos campos del formulario para especificar la página, acción o diversos campos como el título de la película. Para ocultar estos elementos se utiliza la clase hidden de Tailwind.

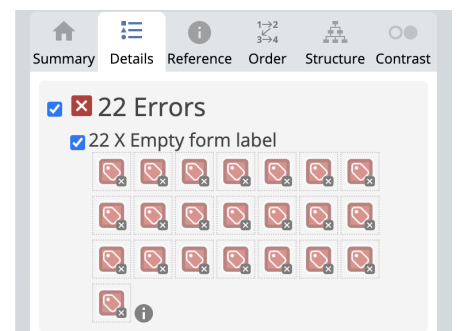
Otro de los problemas solucionados es el atributo alt de las imágenes y hemos hecho la página más navegable al utilizar más botones en vez de divs.



- Lista de Películas

Para la vista de películas se ha corregido los mismos errores que en la pantalla home. Los errores que permanecen son iguales que en en la vista principal, labels de formularios ocultos.

También hemos utilizado la misma estrategia de emplear un índice para los formularios de las películas.



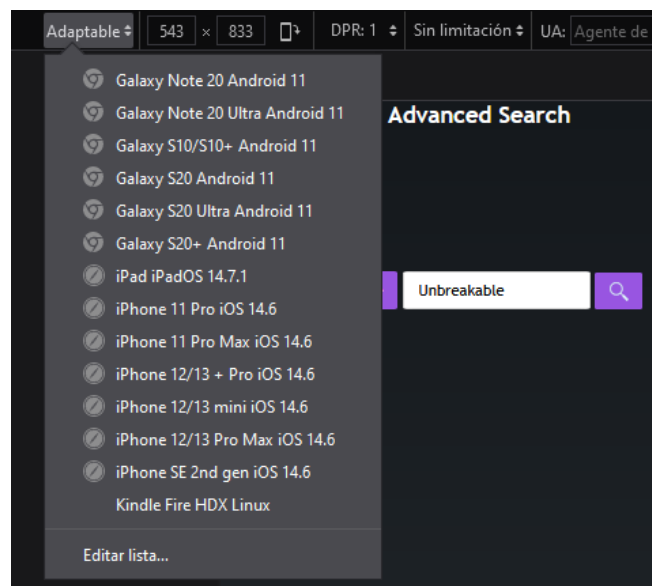
## 9. Usabilidad.

### a. Destacar aspectos tenidos en cuenta sobre usabilidad durante el desarrollo.

Desde el prototipado del proyecto y primeros diseños, hemos mencionado anteriormente que hemos trabajado anteponiendo el uso del móvil.

Para el uso de diferentes dispositivos con tamaños diferentes, se han usado *mediaQueries* en css para adaptar la pantalla a las diferentes resoluciones.

Cada vez que se hacía un cambio mayoritario al diseño, se comprobaba que el formato era resistente y no cambiaba en diferentes dispositivos.



En cuanto a otros apartados de usabilidad:

- Facilidad de aprendizaje

La web no posee ninguna funcionalidad o herramienta que requiera que el usuario tenga que aprender previamente algo para poder usarla.

- Flexibilidad

El usuario y el sistema comparten información constantemente. Ya sea tras escribir una review, rellenar los campos del formulario de búsqueda avanzada, etc. Gracias en parte a la implementación MobileFirst, la adaptabilidad es amplia y en ningún momento la interfaz va a suponer un problema para el usuario. Incluso si abre la aplicación web desde un Smart Fridge.

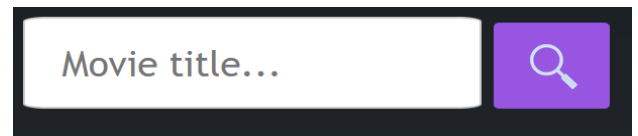
- Consistencia

Los mecanismos que se utilizan son los mismos. Ej: Escribir y pulsar botón.

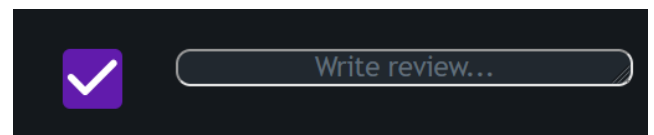
Búsqueda común



Búsqueda avanzada



Escribir review



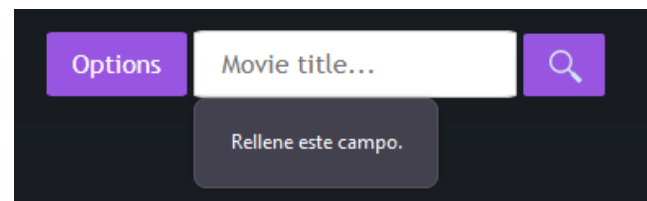
Para volver a pestañas anteriores, se puede volver con las flechas del navegador, gestos del dispositivo móvil, flechas del propio dispositivo.

- Recuperabilidad

El usuario recibe feedback en base a sus acciones, si están mal, no se le penaliza injustamente. Y puede realizar su acción de nuevo.

Por ejemplo:

Si el usuario no introduce un título al buscar, no se le redirige a una pantalla de error, ni un mensaje de error que le impida volver atrás. En cambio, se le informa de manera sutil que debe hacerlo de nuevo .



Lo mismo con el registro y el inicio de sesión.

- Tiempo de respuesta

Las peticiones que se realizan son simples, no hay tiempos de carga y cada vez que se inicia sesión no hay que hacer nada más para tener control total sobre la web y sus funcionalidades.

No hay clicks excesivos. Vamos a contar los clicks desde que se inicia sesión hasta que se busca una película por título y se añade una review:

Iniciar sesión:

- Con atajos de teclado: 2 clicks
- Sin atajos de teclado: 4 clicks

Buscar película:

- Con atajos de teclado: 1 click
- Sin atajos de teclado: 3 clicks

Añadir review:

- 2 clicks

Es decir, para hacer el proceso más largo de la aplicación, no hacen falta ni una decena de clicks. Y en móvil, obviamente es igual de sencillo o más.

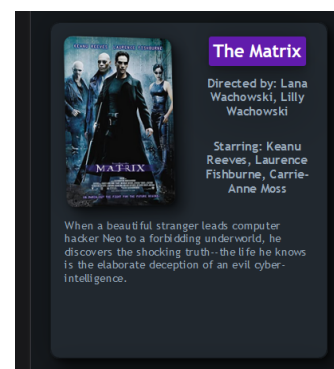
- Adecuación de las tareas

Una acción del usuario desencadena una serie de tareas que definitivamente el usuario no tiene intención de realizar, sin embargo, el resultado obtenido es el esperado. Cuando el usuario hace click en una película, el usuario espera acceder a esa misma película, ya que le ha llamado la atención. Entonces se accede a esa película. Si el usuario ve que esa película tiene malas reviews, e instintivamente desea volver atrás y hace click en la flecha del navegador para retroceder, espera volver a la lista de películas que ha obtenido antes. No el menú principal.

- Satisfacción

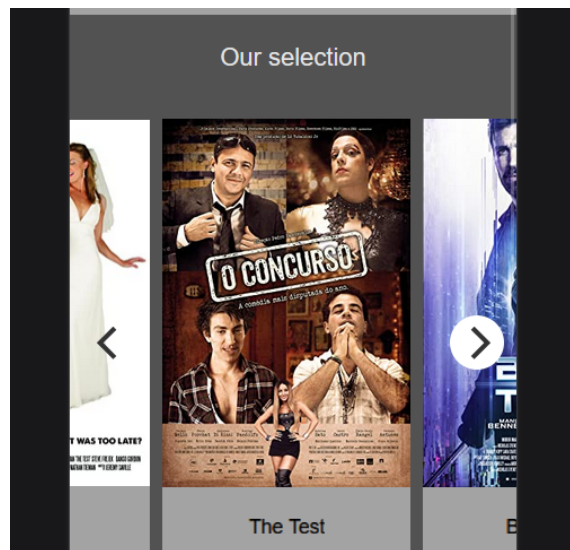
Hay ciertos elementos que el usuario no va a esperar, pero tal vez los descubra instintivamente y esto le suponga cierta satisfacción.

Por ejemplo, al pulsar un póster, dentro de la página de una película, este reacciona al input del usuario. Esto le aporta una sensación de control sobre el contenido .



Lo mismo con el uso del *carrousel*. Un usuario de móvil va a ver como se muestran dos flechas a cada lado del borde de la pantalla, lo que indica que puede desplazarse por esa selección de películas. Si instintivamente decide hacer un “swipe” con el dedo, arrastrándolo por el medio de la pantalla, también va a servir para desplazarse. Tal vez no lo esperaba, pero el hecho

de que esa funcionalidad esté incorporada, por simple que sea, aporta satisfacción y una sensación de control sobre la aplicación.



## b. Pruebas/auditorías realizadas.

Para las pruebas de usabilidad, se ha instruido a varias personas de diferentes edades y diferentes nociones tecnológicas.

A estas personas se les ha indicado lo siguiente: “Deja una review en una película que te guste”

Para ello, el usuario se tiene que dar cuenta que no puede dejar una review sin haber iniciado sesión, y no puede iniciar sesión sin registrarse primero. Entonces, el recorrido que supuestamente debería realizar es:

Registrarse, Iniciar sesión, buscar una película y dejar una review.

Se ha utilizado a las siguientes personas a modo de usuarios para las pruebas:

- Alumno de ingeniería informática de la UPNA (1):
- Madre de un alumno de ingeniería informática.

## c. Resultados obtenidos.

- Alumno de ingeniería informática de la UPNA (1):

Feedback devuelto: La web es intuitiva y carga rápido más allá de las limitaciones de la máquina virtual. No se actualizan los botones de *login* y *sign up* una vez el usuario se ha



logueado, el cuadrado para escribir la review se hace pequeño si el usuario no mantiene el ratón encima, botón para ir al home no actúa con hover al hacer focus en él.

- Madre de un alumno de ingeniería informática.

Se ha desenvuelto con facilidad. Se ha bloqueado únicamente en el registro, ya que al escribir el nombre de usuario y contraseña en los campos correspondientes, Firefox avisa de que es una web no segura. Y el aviso lanzado bloquea parte de los elementos, como el botón de login. Ha perdido tiempo y ha tenido que hacer click fuera del campo para deshacerse del aviso de firefox.

Una vez completado el registro, ha buscado sin problemas la película deseada. Ha hecho click en el póster para acceder y aunque no conocía la traducción de "Write review", ha sabido, por cómo está diseñado, que dentro del campo de texto tenía que escribir el comentario.

En conclusión: Para un usuario casual, la web es cómoda, intuitiva y fácil de usar.

#### d. Modificaciones propuestas.

- Actualizar el icono central que redirige a la página principal para que se identifique correctamente como algo que realiza una acción.
- Sustituir los botones de iniciar sesión y registrarse por un botón de "cerrar sesión" una vez ya lo has hecho.
- Hacer más interactivo el *text-area* de la review para cuando no está siendo focuseado mientras se escribe el comentario no se haga tan pequeño.

## 10. Gestión de configuración.

### a. Uso de repositorios

#### Branching

Se ha utilizado la técnica de branching de “Branch Based”. La cual se basa en la creación de una rama nueva para cada tarea a realizar. Una vez esa tarea se ha realizado, se “mergea” dicha rama con la rama principal del proyecto. Siendo la principal rama DEV.

Esta técnica es especialmente útil en un proyecto como el nuestro dada la cantidad de archivos que se pueden modificar para una sola implementación menor. El uso de otra estrategia como “One Flow” o un “Git Flow” entorpecería el trabajo individual continuo de los trabajadores. Ya que habría que estar continuamente pendiente de que no se estén realizando cambios cada vez que se desea realizar una acción. Dicha estrategia, elegida principalmente para evitar conflictos y el aislamiento de cada nueva implementación (si surge un fallo repentino se puede saber el origen) ha resultado ser acertada. El número de conflictos ha sido mínimo y la dimensión del proyecto ha aumentado de manera iterativa cada vez que una rama acababa.

#### Normas y buenas prácticas

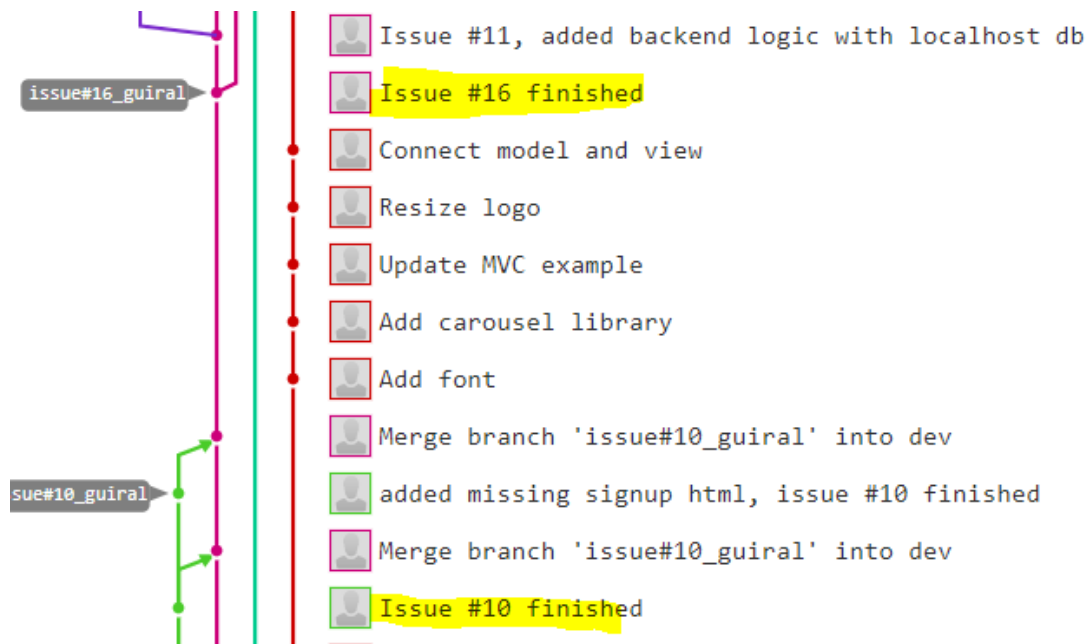
Como se indica en el README del proyecto, cada rama, asignada a una tarea específica, debe ser llamada de una manera en particular:

```
issue#{n°issue}__{devName}
```

En el momento que ya se empieza a trabajar en la misma, se prefiere un uso constante de commits, para dejar constancia de cada nuevo paso en la nueva implementación en la que se está trabajando. Dichos commits no tienen un formato preestablecido, ya que en principio solo va a trabajar en esa rama una sola persona. En cambio, para el último commit antes del “merge” con la rama principal DEV, el formato debe ser:

```
"Issue #{n°issue} finished"
```

De esa manera, cada nueva rama que se ha implementado tiene una fecha de comienzo y una fecha final.



## b. Integración continua

Hemos creado una pipeline en gitlab que se “descarga” los cambios de la rama `dev`. En esta rama es donde vamos haciendo merge de nuestras ramas con los nuevos cambios. Para ello tenemos que configurar un job con los siguientes paso:

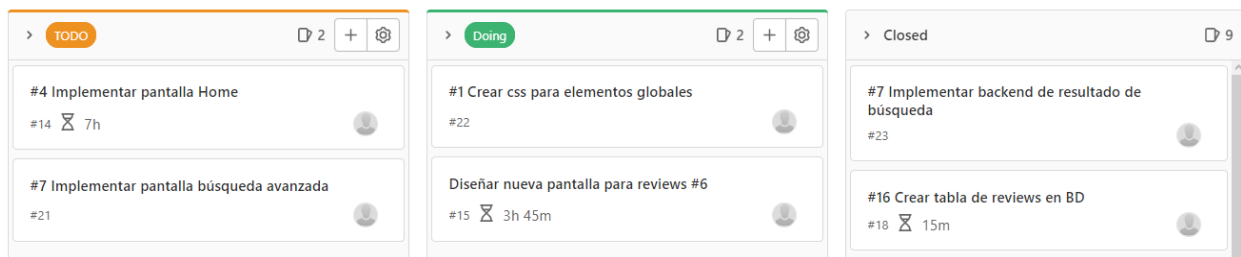
- `git checkout dev`
- `git pull`

La principal ventaja de esta pipeline es no tener que entrar en el servidor de guacamole para bajar los cambios realizados. Cualquier modificación en el repositorio activa estos comandos, si se ha modificado la rama `dev` dichos cambios estarán disponibles en nuestro servidor. La pipeline está configurada en el servidor principal, id 69044. A pesar de estos intentos no hemos conseguido que funcione correctamente. Como hemos comentado en el apartado de despliegue tuvimos problemas en cuanto a la autenticación contra gitlab. Una de las posibles soluciones es utilizar la autenticación mediante ssh.

### c. Trabajo en equipo y reparto de responsabilidades.

Para las primeras puestas en común y reuniones de *brainstorming* necesarias para el arranque del proyecto, se trabajó conjuntamente, sin dividir tareas. Todo ello para realizar los primeros diseños, wireframes etc, mientras el proyecto no estaba enteramente definido. Una vez finalizada esa primera fase, el reparto de tareas era necesario para abarcar el proyecto al completo, así que el principal entorno de interacción entre los miembros del equipo pasó a ser el repositorio de GitLab UPNA.

Se ha seguido un riguroso procedimiento de creación de tareas en base a las historias de usuario. Un solo miembro del equipo se asigna la tarea y se añade un “Time-Tracker” en el caso de que la tarea sea de mantenimiento o suficientemente corta como para crear una rama nueva [\[ver Branching\]](#). Entonces dicho miembro se puede dedicar a esa tarea en concreto, sacando dicha issue de “TODO”, y colocándola en “DOING”. De esa manera, se informa al resto del equipo de la tarea/tareas que se están realizando en ese momento para evitar confusiones de quién está trabajando en qué. Una vez acabada, se cambia la etiqueta de la issue y se cierra, colocándose automáticamente en la pestaña “Closed”.



También se han seguido métodos de XP tales como el “Pair Programming” entre dos miembros del equipo. Sólomente en casos en los cuales las issues a realizar por ambos programadores eran similares. De esa manera, el esfuerzo es compensado y aprovechado en ambas issues. También se ha recurrido al trabajo simultáneo entre varios miembros en casos específicos de errores, dudas y conflictos. Donde la colaboración de todos los miembros era necesaria para evitar confusiones adicionales. Para el trabajo simultáneo, cabe destacar la importancia de la aplicación “Discord” a la que se ha recurrido numerosas veces para: compartir pantalla y enseñar los errores, mandar links, referencias...

## 11. Conclusiones

El resultado final ha sido el conjunto de numerosas prácticas y procedimientos que nunca antes habíamos realizado tan metódicamente para desarrollar un proyecto. Eso ha causado que se vea reflejado en la última iteración del proyecto, ya que cada apartado al que se ha dedicado un tiempo específico, como la usabilidad o accesibilidad, ha provocado que nos acostumbremos a ciertos comportamientos que se han mantenido desde las primeras implementaciones, hasta el final.

El uso del repositorio ha sido satisfactorio, y ha hecho que el despliegue sea sencillo a la hora de la entrega.

La aplicación web ha cumplido con los objetivos iniciales y le vemos un uso real y útil en el panorama actual. Aunque sea de un interés puramente lúdico y de entretenimiento.

## 12. Anexos

REF1: <https://coolors.co>

<https://css-tricks.com/css-only-carousel/>

<https://www.shutterstock.com/blog/neon-color-palettes>

REF2:

Página de la API : <https://www.omdbapi.com>

Ejemplo de petición URL: <https://www.omdbapi.com/?t=unbreakable&apikey=ce16ecd1>

REF3:

Cada película en la página de IMDb : <https://www.imdb.com>. Tiene una ID (IMBDId) que se usa para cada petición de información de una película específica.

REF4:

<https://wave.webaim.org/>

Extensión para firefox: <https://addons.mozilla.org/es/firefox/addon/wave-accessibility-tool/>

REF5:

Extension para firefox (SEOquake):

<https://addons.mozilla.org/es/firefox/addon/seoquake-seo-extension/>

Páginas web que teníamos pensado usar para obtener resultados del SEO de nuestra aplicación web:

<https://validator.ampproject.org/>

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

<https://nibbler.silktide.com/>