**TRABAJO FIN DE GRADO**

***Cartelera Online***

**Curso: 2020-2021**

**Alumno**: Diego García Losada

**Fecha: 11**/06/2021

**ÍNDICE DE CONTENIDOS**

[1. INTRODUCCIÓN 3](#_Toc74081774)

[1.1. TECNOLOGÍAS 3](#_Toc74081775)

[1.2. APLICACIONES SIMILARES 3](#_Toc74081776)

[2. DESCRIPCIÓN 3](#_Toc74081777)

[3. INSTALACIÓN Y PREPARACIÓN 4](#_Toc74081778)

[4. GUÍA DE ESTILOS Y PROTOTIPADO 5](#_Toc74081779)

[4.1. COLORES Y TIPOGRAFÍA 5](#_Toc74081780)

[4.1.1. COLORES 5](#_Toc74081781)

[4.1.2. TIPOGRAFÍA 5](#_Toc74081782)

[4.1.3. ICONOS 6](#_Toc74081783)

[4.2. WIREFRAMES 6](#_Toc74081784)

[4.2.1. WIREFRAMES ESCRITORIO 6](#_Toc74081785)

[4.2.2. WIREFRAMES MÓVIL 6](#_Toc74081786)

[4.3. MOCK UPS 6](#_Toc74081787)

[4.3.1. MOCK UPS ESCRITORIO 6](#_Toc74081788)

[4.3.2. MOCK UPS MÓVIL 7](#_Toc74081789)

[5. DISEÑO 8](#_Toc74081790)

[5.1. DESCRIPCIÓN 8](#_Toc74081791)

[5.2. DIAGRAMAS DE DISEÑO 8](#_Toc74081792)

[5.2.1. ENTIDAD RELACIÓN 8](#_Toc74081793)

[6. DESARROLLO 9](#_Toc74081794)

[7. PRUEBAS 11](#_Toc74081795)

[8. DESPLIEGUE 12](#_Toc74081796)

[8.1. DOCKER-COMPOSE-YML 12](#_Toc74081797)

[8.2. DOCKERFILE (backend) 13](#_Toc74081798)

[8.3. DOCKERFILE (frontend) 13](#_Toc74081799)

[9. MANUAL 14](#_Toc74081800)

[10. CONCLUSIONES 15](#_Toc74081801)

[11. ÍNDICE DE IMÁGENES 15](#_Toc74081802)

[12. *Bibliografía* 15](#_Toc74081803)

[13. ANEXOS 16](#_Toc74081804)

# INTRODUCCIÓN

## TECNOLOGÍAS

La tecnología que he usado para front-end ha sido React, he elegido React porque es la que estuvimos viendo en clase más a fondo.

En [esta](https://www.codeinwp.com/blog/angular-vs-vue-vs-react/)(1) página me he basado mucho en la elección, dice que Vue y React son los más populares, pero a la hora del mercado de trabajo Vue apenas se ve. Entonces he elegido React porque es uno de los framework front-end más populares, y además también tiene un alto mercado de trabajo

Para backend no sabía cuál elegir, en las prácticas empezamos con Flask en Python, así que decidí seguir en la misma línea que en las prácticas, para poder tener una base sobre la que empezar mi proyecto. Usando FlaskRestFul junto a SQLAlchemy, que es una librería de Python. Al ser un ORM facilita el manejo de la base de datos.

Para la base de datos he usado la misma que en las prácticas, PostgreSQL.

## APLICACIONES SIMILARES

Algunas aplicaciones similares pueden ser: Netflix, Rottentomatoes, Filmaffinity y Plex.

# DESCRIPCIÓN

En esta aplicación lo que busco es que las personas aficionadas al cine tengan un lugar donde puedan comentar y buscar películas.

El usuario será capaz de crear una lista personalizada de películas en las que podrá añadir cualquier película que se encuentre en la página.

# INSTALACIÓN Y PREPARACIÓN

Entrar al github de mi proyecto: <https://github.com/digl7/proyecto-final> y descargar el código fuente del proyecto. Una vez hecho nos aseguramos de que estamos en la carpeta donde se encuentran las carpetas ‘front’, ‘back’ y el archivo docker-compose.yml. A continuación, nos dirigimos a la carpeta ‘front’ y comprobamos que esté la carpeta de ‘node-modules’, sino está, debemos abrir la cmd en esa carpeta (front) y hacer el siguiente comando:

* npm install

Una vez terminado el comando nos dirigimos a la carpeta padre del proyecto, abrimos la cmd y escribimos el siguiente comando:

* docker compose up

Una vez terminado nos dirigimos a localhost:3000 y ya podremos hacer uso de la página.

# GUÍA DE ESTILOS Y PROTOTIPADO

## COLORES Y TIPOGRAFÍA

### COLORES

|  |  |
| --- | --- |
| Amarillo: #F9BC50 | Negro: #181B1D |

El Amarillo lo uso para resaltar algunas opciones (como Iniciar sesión) o al hacer hover sobre los elementos en la barra de navegación.

### TIPOGRAFÍA

Como tipografía principal he usado la tipografía ‘Roboto’ de la familia sans serif (bold y regular), tanto como para párrafos en el cuerpo de la página, como para los títulos al hacer hover en los posters de las películas.

Ejemplo de ‘Roboto’: Título y descripción.

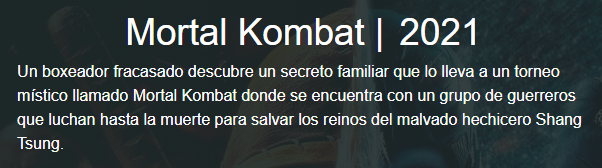


Ilustración 1: tipografía Roboto

Para la barra de navegación he usado la tipografía ‘Antonio’ también de la familia sans serif (bold). Ejemplo de ‘Antonio’:



Ilustración 2: tipografía Antonio

Para el logo se ha usado la tipografía ‘Nunito’

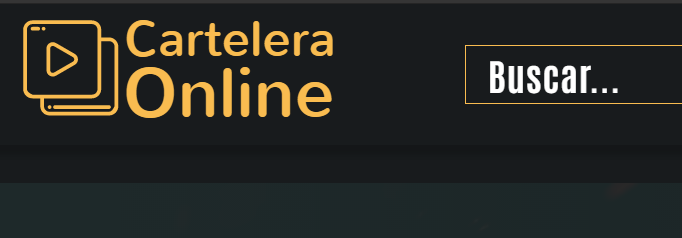


Ilustración 3: Logo tipografía Nunito

### ICONOS

He usado el icono “[bars](https://fontawesome.com/icons/bars?style=solid)” solid, para abrir el menú desplegable móvil.

Ilustración 4: Icono bars

Y el icono “[times](https://fontawesome.com/icons/times?style=solid)” solid, para cerrar el menú desplegable móvil.

Ilustración 5: Icono times

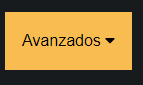
El icono “[caret-down](https://fontawesome.com/icons/caret-down?style=solid)” para abrir el menú oculto de filtros.

Ilustración 6: Icono caret-down

## WIREFRAMES

### WIREFRAMES ESCRITORIO

Esta en anexos.

### WIREFRAMES MÓVIL

Esta en anexos.

## MOCK UPS

### MOCK UPS ESCRITORIO

Esta en anexos.

### MOCK UPS MÓVIL

Esta en anexos.

# DISEÑO

## DESCRIPCIÓN

## DIAGRAMAS DE DISEÑO

### ENTIDAD RELACIÓN

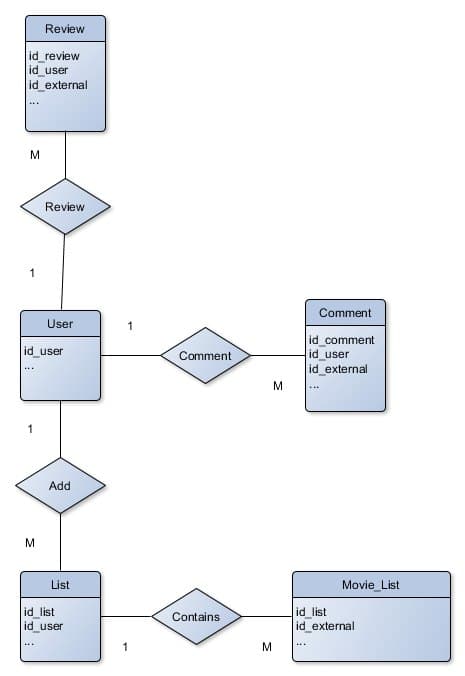


Ilustración 7: Entidad Relación

# DESARROLLO

Empecé haciendo el componente de la barra de navegación (“navbar.jsx”) junto a su css, porque al tener que ir en todas las páginas me resultó más fácil crear este primero y luego ir añadiéndolo a las demás.

**También tengo un componente de búsqueda películas (en el Navbar), pero lo uso en el cuerpo del “home” a través de los props, en este, tuve un pequeño problema al estar en otra página que no sea home y buscar una película. Para arreglarlo hice redirect de la página a Home a través de onClick en la barra de búsqueda, y autofocus al entrar a Home. Uso el useEffect para cada vez que escriba alguna letra me busque la película.**

Continué con la página de Inicio “Home.jsx”, no tuve muchos problemas al renderizar las películas de la API externa, el problema me surgió al tener que hacer la paginación. Me devolvía las películas en una lista de 20 y no tenía muy claro la forma en la anidar las listas, al final usé la [sintaxis extendida](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Operators/Spread_syntax) **que mantiene la primera lista (las 20 de la primera llamada) de películas y le añado la siguiente página con un contador cada vez que le damos al botón de “Cargar más”.**

**En Home tengo un onClick en los carteles de cada película que me lleva a MoviePage.jsx, que contiene más información, como géneros, descripción, actores que trabajaron en esa película y los comentarios de los usuarios. En esta página tuve un error en las fotos por defecto (cuando no encontraba foto de actores o de portada de la película) no podía acceder a las fotos de la carpeta “public”, así que lo hice fue crear una carpeta con las fotos que usaba en “public” y las importo en “MoviePage.jsx”.**

**En este momento empecé a hacer el back-end, empezando por los comentarios de cada película, tengo tres endpoints: AllComments, CommentCreation y CommentFromMovie.**

* **AllComments (/comments) me muestra todos los comentarios de la base de datos.**
* **CommentCreation (/comment/1) lo utilizo cuando el usuario decide comentar en cualquier película.**
* **CommentFromMovie (/comment/view/567189) recoge todos los comentarios de una película.**

**Seguí con la parte de backend de Movie, mis end-points: AddMovie, AllMovies, MoviesFromList.**

* **AddMovie (/movie/1) te permite añadir una película a una lista (la que el usuario seleccione en la página de MoviePage).**
* **AllMovies (/movies): obtienes todas las películas de la base de datos.**
* **MoviesFromList (/movie/list/1) la uso para obtener todas las películas de una lista y luego renderizarlas en la página de Listas.**

**Después hice la parte de las listas, estos son mis end-points: AddList, AllLists, ListDelete, ListFromUser, ListbyID, MovieDeleteFromList**

* **AddList (/list/1) cuando el usuario crea una lista.**
* **AllLists (/lists) para mostrar todas las listas de la base de datos.**
* **ListDelete (/list/delete/1) para que el usuario pueda borrar una lista, esa lista se borra por el id.**
* **ListFromUser (/lists/user/1) para mostrar todas las listas de un usuario en concreto.**
* **ListbyID (/list/1) devuelve una lista cuando le pasas el id**
* **MovieDeleteFromList (/list/1/delete/527774) borra la película que le pases (IdExterno)**

**Con las listas, en el back-end tuve varios problemas para poder obtener las películas (con sus datos, título, año, etc.) que el usuario había guardado en su lista. Al final lo que terminé haciendo fue una llamada a la API externa en mi backend para guardar todos los datos de la película en las listas. Y por último llamo a mi API para obtener las películas con sus datos.**

**Recorro con “map” todos los actores que he cogido de la API externa, y también recorro todos los comentarios de la película y listas del usuario de mi API.**

Después de estas tres páginas, seguí con la página de Login, Registro y Activar Email.

Cada apartado es un componente “LoginForm.jsx”, “RegisterForm.jsx” y por último “Recover.jsx”. Los datos del usuario al hacer Login los guardo en LocalStorage y quería tener dos roles, pero no me ha dado tiempo así que en el fetch (post) de Login le paso rol de user. También uso props para cambiar de componente (a registro y a activar el email).

Después de tener completas las páginas en React con sus componentes le añadí el backend de usuario, tengo 5 endpoint: Logout, Register, Login, AllUsers y EmailConfirmation

* Logout (/user/logout), permite al usuario cerrar sesión.
* Register (/user/register) permite al usuario registrarse en la aplicación, y se le enviará un correo con el código de confirmación (EmailConfirmation)
* Login (/user/Login) permite que el usuario inicie sesión (el usuario no puede iniciar sesión si no ha activado el correo a través del código)
* Login (/users) para mostrar todos los usuarios
* EmailConfirmation (/user/activate/InJ1Ym9AZ21haWwuY29tIg.YMIbbw.E1iOqJbO\_UoQSME85oCFWGXHzq4) Es el enlace con el cual el usuario activa su propio correo.

Luego hice la página de filtros que no tuve ningún problema.

Las herramientas de control de versiones utilizadas principalmente he usado Github junto a “SourceTree” que es un programa que simplifica Git con interfaz simple, solo he usado la rama main porque al estar haciendo el proyecto en solitario lo tengo bastante controlado.

# PRUEBAS

Las pruebas que he realizado han sido automáticamente realizadas por postman, a través de swagger. (carpeta: html-postmanDocumentation)

# DESPLIEGUE

Para el despliegue he usado la aplicación Docker, lo primero que he hecho ha sido crear un Docker-compose.yml en la carpeta padre (proyectofinal), y después 2 Dockerfiles en sus respectivas carpetas, back y front

## DOCKER-COMPOSE-YML

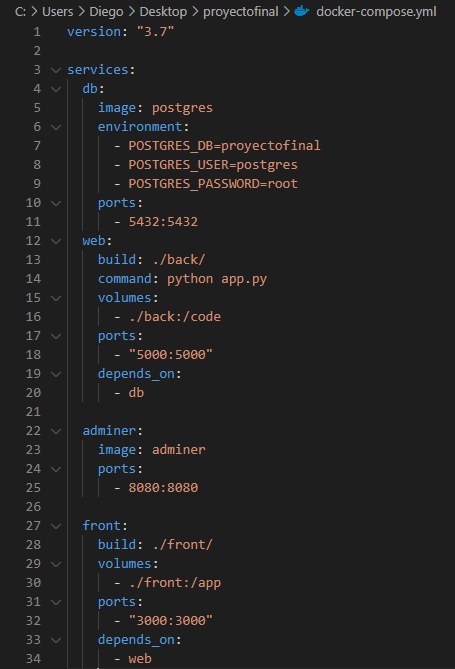


Ilustración 8: Docker-compose.yml

db: Es la base de datos (Postgres), con la información para conectarme a la base de datos de ‘proyectofinal’ con el usuario Postgres y contraseña ‘root’ junto con el puerto 5432:5432.

web: Es el backend de mi aplicación (Flask) que se ejecuta en la carpeta /back/ y dentro ejecuta el comando ‘python app.py’ para levantar el backend, conectado al puerto 5000:5000

adminer: Es una herramienta de gestión de base de datos.

front: Es el frontend de mi aplicación (React), que se ejecuta en la carpeta /front/ conectado al puerto 3000:3000

## DOCKERFILE (backend)

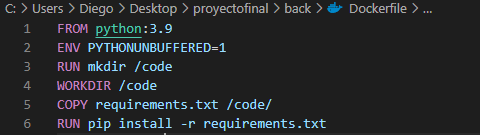


Ilustración 9: dockerfile backend

## DOCKERFILE (frontend)

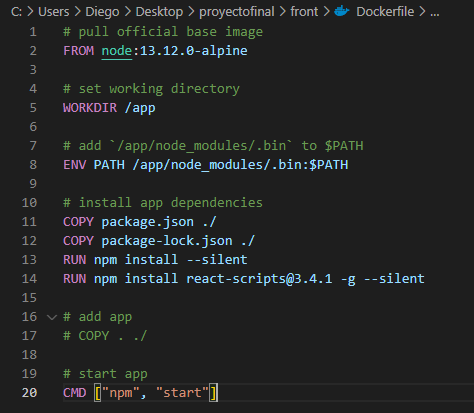


Ilustración 10: dockerfile frontend

# MANUAL

En mi página no importa si el usuario está registrado para ver, filtrar películas, pero si quiere crear una lista o comentar en alguna película, necesita estar registrado y haber iniciado sesión (con el email confirmado).

Entonces lo primero que tienes que hacer es registrarte con un correo válido, te llegará un código de confirmación y lo pegas en la ventana de confirmar email. Una vez confirmado podrás iniciar sesión y crear listas y comentar en las películas.

Para crear una lista tienes que dirigirte a la ventana de ‘Mi lista’ una vez dentro te aparecerá la opción de Crear lista, le damos clic y podremos escribir el nombre que queramos. Una vez creada la lista aparecerá que no tenemos ninguna película añadida, ahora podemos ir a la página principal o a la página de filtrar.

Cuando hayas visto una película que quieras añadir a tu lista, le das clic al cartel de esa película y entonces entrarás en una nueva página donde podrás información más detallada de la película en concreto, al lado del título hay un botón ‘Añadir a mi lista’ le damos clic y aparecerán todas nuestras listas, y por último le damos a la lista en la que queramos añadir esa película.

Una vez terminado lo anterior podremos volver a la página de las listas y ahí podrás ver las películas que tengas en tus listas. En esta página también podemos borrar la lista, borrar películas de cada lista y renombrar la lista.

Para poder comentar le damos clic a una película y nos dirigimos a la parte de abajo de la página y encontraremos un text-area en el que podremos escribir lo que queramos (máximo 240 caracteres, indicado dentro del recuadro). Una vez terminado de escribir le damos a enviar y lo podremos ver a la derecha (en modo escritorio, o abajo en versión móvil).

# CONCLUSIONES

Al final he conseguido realizar casi todo lo que me propuse en mi idea inicial, me han faltado por terminar algunas cosas, como el segundo rol (admin) que tenía en mente que pudiese borrar cualquier comentario y una última página de Reviews pero por falta de tiempo no he podido.

# ÍNDICE DE IMÁGENES

[Ilustración 1: tipografía Roboto 5](#_Toc74081731)

[Ilustración 2: tipografía Antonio 5](#_Toc74081732)

[Ilustración 3: Logo tipografía Nunito 6](#_Toc74081733)

[Ilustración 4: Icono bars 6](file:///C:\Users\Diego\Desktop\TrabajoFinalGrado.docx#_Toc74081734)

[Ilustración 5: Icono times 6](file:///C:\Users\Diego\Desktop\TrabajoFinalGrado.docx#_Toc74081735)

[Ilustración 6: Icono caret-down 6](file:///C:\Users\Diego\Desktop\TrabajoFinalGrado.docx#_Toc74081736)

[Ilustración 7: Entidad Relación 8](#_Toc74081737)

[Ilustración 8: Docker-compose.yml 12](#_Toc74081738)

[Ilustración 9: dockerfile backend 13](#_Toc74081739)

[Ilustración 10: dockerfile frontend 13](#_Toc74081740)

# *Bibliografía*

1. **https://www.codeinwp.com/blog/angular-vs-vue-vs-react/.** *angular vs vue vs react.*

# ANEXOS