Dialogo TFG 2.0

1- Justificación del proyecto

Buenos días, mi nombre es Juan y hoy voy a presentaros mi Proyecto, que consiste en una plataforma web de alquiler de películas parecida a un videoclub y que está desarrollada con Laravel, MySQL, Blade, Tailwind CSS, y React.

Este proyecto surgió con la idea de recuperar el espíritu de los antiguos videoclubs, pero adaptado a la era digital. El objetivo es ofrecer una plataforma donde los usuarios puedan registrarse, alquilar películas, visualizarlas por tiempo limitado y gestionar su historial.

Elegí este proyecto porque me permite aplicar una amplia variedad de tecnologías modernas del desarrollo web y consolidar los conocimientos adquiridos durante el ciclo.

2- Estilos

He dividido los estilos del proyecto en varias partes:

- Backend en Laravel, donde se estructura la lógica de negocio y la conexión con la base de datos.
- Frontend con Blade y React, lo que me ha permitido generar una interfaz atractiva y dinámica.
- Base de datos MySQL, con tablas como usuarios, peliculas, alquileres, etc.

También he utilizado Tailwind CSS para el diseño visual y Carbon para la gestión de fechas y tiempo restante de alquileres.

Paleta de colores

Colores primarios:

En la interfaz de una web se emplean colores primarios para resaltar acciones clave y captar la atención del usuario de manera inmediata.

Uno de los elementos más notables de mi web es el botón de búsqueda, que utiliza un rojo brillante, el cual actúa como un punto de enfoque tanto en el modo claro en el que es azul como en el oscuro.

Este color también funciona como acento visual en otras partes de la interfaz.

Además, el texto del carrito de la compra se comporta dinámicamente, alternando entre varios colores primarios llamativos (como verde neón o naranja) para atraer la mirada del usuario y transmitir un estado cambiante. Esta rotación de color no solo agrega dinamismo, sino que también refuerza el enfoque interactivo del sistema.

Colores secundarios y de contraste:

Los colores secundarios se utilizan para definir funciones auxiliares o enfatizar advertencias y decisiones importantes del usuario.

El rojo, por ejemplo, se emplea de forma consistente en los botones de "Quitar Producto", indicando una acción destructiva o irreversible.

También se utilizan tonos blancos en ciertos estados visuales en forma de contraste, como en los botones de vaciar el carrito o realizar el pedido, que contrastan con los colores de fondo que son tonos más oscuros. Estos colores refuerzan la jerarquía visual al permitir al usuario distinguir claramente entre acciones primarias y secundarias.

Colores neutros:

Los colores neutros forman la base estructural de la interfaz, adaptándose a los dos modos disponibles: claro y oscuro. En el modo oscuro, predominan el negro y los grises oscuros, proporcionando un fondo elegante y sobrio que permite que los colores vibrantes destaquen más.

Por el contrario, en el modo claro se utiliza blanco o gris muy claro como fondo, con texto en negro o gris oscuro para asegurar una legibilidad óptima. Estos colores neutros también se aplican en tarjetas de productos, barras laterales y contenedores, facilitando una experiencia visual ordenada y sin distracciones.

Además de implementar mi propio css y tailwind también he utilizado diversas herramientas de las cuales ayudarme para mejorar el estilo de mi página web.

Un ejemplo de esto sería la página web uiverse.io de la cual he tomado inspiración para implementar diversos elementos css y adaptarlos así a mi página o la página flowbiteCss de la cual me he tomado para generar el header y el footer de mi página.

3- Tecnologías Utilizadas

A continuación, voy a hablar de las tecnologías utilizadas en mi web y el por qué se han utilizado algunas herramientas frente a otras alternativas de su misma categoría:

1. Laravel (Framework Backend)
Laravel ha sido elegido por su estructura clara, su sistema de rutas y autenticación integrada lo que acelera el desarrollo.

2. Blade y React (Frontend)
Blade permite la generación rápida de vistas del lado del servidor, mientras que React mejora la interactividad en secciones dinámicas. Esta combinación supera las limitaciones de utilizar solo Blade, ofreciendo una experiencia de usuario más fluida.

3. Tailwind CSS (Estilado)
Tailwind se ha utilizado debido a que frente a otros frameworks Tailwind ofrece mayor control visual y se integra mejor con componentes React.

4. CSS3
Se ha utilizado también CSS para ciertos elementos de la página que son más complicados de realizar con tailwindcss.

5. MySQL (Base de datos)
Se ha optado por MySQL por su fiabilidad, facilidad de uso y compatibilidad con Laravel. Aunque PostgreSQL es más robusto para estructuras complejas, MySQL es suficiente para la lógica relacional de un videoclub, manteniendo buen rendimiento.

7. Composer
Sistema de gestión de paquetes de código de PHP para manejar dependencias y librerías en el proyecto.

8. Javascript
También se ha utilizado un poco de javascript para reaccionar a ciertos eventos durante el funcionamiento de la página web.

Puedes mencionarlo brevemente:
(Carbon (Manejo de fechas en PHP)
Carbon mejora el manejo de fechas frente a las funciones nativas de PHP, simplificando tareas comunes como calcular la duración de un alquiler o fechas de vencimiento.)

La elección de cada herramienta se basó en un análisis comparativo, valorando factores como la comunidad, la documentación, el rendimiento y la facilidad de aprendizaje.

4- Estructura del proyecto

La estructura de mi proyecto sigue el modelo predeterminado de los proyectos de Laravel. Algunos de sus archivos más importantes son algunos como:

- **Archivos raíz**
  - Los archivos raíz de un proyecto gestionan la configuración general, las dependencias y herramientas necesarias para que el entorno funcione correctamente.
  - Esto ya no da tiempo:
    - Incluyen archivos como .env para datos sensibles de configuración, composer.json y package.json para definir dependencias de backend (PHP) y frontend (JavaScript), junto con sus respectivos archivos lock que aseguran versiones exactas.
    - También contienen configuraciones específicas para herramientas como TailwindCSS, PostCSS y Vite, que se encargan del procesamiento de estilos y la construcción del frontend. En conjunto, estos archivos permiten instalar, mantener y ejecutar el proyecto de manera coherente y segura.

- **App/Http/Controllers:** Contiene los controladores que gestionan la lógica del backend.
- **App/Http/Models:** Contiene los modelos que representan y gestionan las tablas de la base de datos. Cada modelo está vinculado directamente a una tabla y se encarga de interactuar con ella.
- **Config**: Contiene la configuración de cada aspecto del framework Laravel.

- **Public:** La carpeta public en un proyecto Laravel es el punto de entrada de la aplicación; contiene el archivo index.php, que carga todo el framework. Además, aloja recursos públicos como imágenes, archivos CSS y JS.

- **Resources:** La carpeta resources contiene los archivos que necesitan ser procesados antes de usarse en producción, como vistas Blade, archivos Sass/CSS, JavaScript y recursos de frontend. Es el lugar donde se desarrolla la interfaz y se organizan los componentes visuales.

- **Routes:** La carpeta routes define las rutas de la aplicación. Contiene archivos como web.php y api.php, donde se organizan las URL y las acciones que debe ejecutar el sistema.

- **Storage:** La carpeta storage almacena archivos generados por la aplicación, como logs, archivos temporales, cachés, y archivos subidos por los usuarios. También guarda datos que deben persistir pero no ser públicos

- **Vendor:** La carpeta vendor contiene todas las dependencias externas del proyecto instaladas mediante Composer. Aquí están las librerías necesarias para que la aplicación funcione. No se modifica manualmente.

5- Implementación Técnica (Explicar un poco el código)

Ahora hablare de los puntos más importantes de mi código:

Al principio decidí realizar este proyecto en php básico ya que era el lenguaje de programación con el que me sentía más cómodo, pero al final me decidí por utilizar el framework de laravel ya que, aunque no lo habíamos visto mucho me intereso el probar su funcionalidad para el proyecto y al final me gusto tanto que decidí implementar completamente mi proyecto con laravel.

Además de laravel, también he utilizado otros lenguajes y frameworks como react, javascript o tailwindcss.

Como se puede apreciar he utilizado diferentes archivos para configurar el funcionamiento de tailwindcss en mi proyecto y por otra parte también he utilizado un archivo css llamado styles.css para poder utilizar css en algunas ocasiones.

También se puede apreciar en algunas partes de mi código el uso de javascript como se puede ver en algunos Blades como registro.blade.php en el que al final de la pagina se añade un script que utiliza js para solo mostrar mensajes de error o de confirmación durante un periodo limitado de tiempo.

Lo que me ha sorprendido mas sin lugar a dudas a sido el funcionamiento de react el cual, aunque al principio se me hizo difícil de entender conseguí utilizarlo para mostrar componentes funcionales en mi página y que así quedase más bonita.

Además, también he utilizado algunas librerías de laravel como pueden ser auth para identificar a los usuarios o Carbon que permite reducir el tiempo automáticamente desde la base de datos o incluso Mail el cual permite mandar datos de un lugar a otro para que lleguen al email de los usuarios.

Por último, me gustaría añadir que también he subido mi proyecto a un servidor gratuito llamado infinityfree el cual como no me permitía subir archivos muy

grandes manualmente tuve que ayudarme de herramientas como Filezilla el cual permite conectarse a una ip externa para traspasar archivos de un sitio a otro.

6- Explicar el funcionamiento de la pagina

Ahora mostrare mi web y realizare una serie de pruebas para comprobar que mi página web funciona correctamente:

- Explicar cada una de las páginas y su función
- Mostrar que la web es responsive
- La barra de búsqueda
- Diferencia entre acceder como un admin y un usuario normal
- Mostrar que al registrarte y al hacer un pedido se manda un email a maitrap y que para que se le mande a un usuario de verdad solo tendrías que cambiar la configuración del archivo .env y del mail.php dentro de config.
- Mostrar el funcionamiento de paypal en estado sandbox y que para que funcione solo tendrías que cambiar del estado sandbox a uno live.

Como mayor ejemplo podrías ir creando un usuario normal hacer un pedido con el, enseñar que puede hacer pedidos y ver pelis y luego al final eliminarlo y enseñar que su pedido se elimina también.

Y ya después muestras las cosas que los administradores pueden hacer y los usuarios no.

Conclusiones Personales

Este proyecto me ha ayudado a afianzar conceptos como MVC, relaciones en bases de datos, y a trabajar con herramientas nuevas como Tailwind, laravel y React. Me siento satisfecho con el resultado, ya que combina diseño, funcionalidad y lógica de negocio real. Además, aprendí a solucionar problemas reales, como la gestión del tiempo y de roles, y a organizar un proyecto complejo de forma modular.