

TRABAJO DE FIN DE GRADO

Desarrollo de aplicaciones Web

VEREFLIX

Dimitri Rodríguez Antón

ÍNDICE

1. Introducción.

- 1.1. Descripción del proyecto.
- 1.2. Objetivos del proyecto.
- 1.3. Diagrama de Gantt.
- 1.4. Motivación del proyecto.

2. Desarrollo de la web.

- 2.1. Tecnologías utilizadas.

3. Estructura y funcionalidades.

- 3.1. Capa de persistencia.
- 3.2. Capa lógica.
- 3.3. Capa de presentación.

4. Manual de usuario.

5. Conclusión.

- 5.1. Problemas encontrados.
- 5.2. Mejoras.

6. Bibliografía.

1. Introducción

1.1. Descripción del proyecto

Hay muchas plataformas enfocadas a la visualización de contenido online. Netflix, hbo, prime video y Disney+ son algunas de las muchas que hay y todas tienen algo en común, una suscripción para su disfrute.

Vereflix es una plataforma que intenta romper con eso y ofrecer contenido gratuitamente, lo único que te va a hacer falta es registrarte y logearte con una cuenta y podrás disfrutar de miles de títulos gratuitamente.

1.2. Objetivos del proyecto

El proyecto se centra en la idea de crear una plataforma de visualización de contenido donde no se cobre por el servicio, así puede ser utilizado por la gran parte de la gente.

1.3. Diagrama de Gantt

El diagrama de Gantt es una herramienta útil para planificar proyectos. Al proporcionarte una vista general de las tareas programadas, todas las partes implicadas sabrán qué tareas tienen que completarse y en qué fecha.

Este es el diagrama de Gantt realizado para este proyecto:

| Actividad / Semana | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|--------------------------------------|---|---|---|---|---|---|---|---|---|----|----|----|
| Objetivos y captación de información | ✓ | ✓ | | | | | | | | | | |
| BBDD y lenguajes | | | ✓ | ✓ | | | | | | | | |
| BackEnd | | | | | ✓ | ✓ | ✓ | ✓ | | | | |
| login y registro | | | | | | | ✓ | ✓ | | | | |
| FrontEnd | | | | | | | | ✓ | ✓ | ✓ | | |
| Diseño | | | | | | | | | ✓ | ✓ | | |
| Documentación | | | | | | | | | | | ✓ | ✓ |
| Pruebas | | | | | | | | | | | ✓ | ✓ |

Se puede observar cómo está organizada por semanas. Es muy importante tener en cuenta porque se ha escogido este orden de realización de tareas. Cabe señalar que la documentación, aunque solo está expuesta las últimas dos semanas, se ha ido recogiendo datos fundamentales durante la realización de todo el proyecto.

-Es fundamental recaudar información y tener claros los objetivos del proyecto antes de la realización de este. De esta forma se tiene en cuenta mucho más claro la dirección que hay que tomar.

-La estructura principal en la BBDD y el lenguaje que se va a utilizar es muy importante, sin tener esto claro no se puede seguir avanzando.

-Previo al front, es muy importante estructurar el back, ya que hay que tener en cuenta todas las llamadas necesarias con la base de datos (y a la api), como toda la estructura. Para ejemplificarlo mejor, a un cuerpo no le puedes poner el musculo si no tiene esqueleto, pues en este caso el esqueleto es el back.

-Junto al front viene el diseño ya que es necesario complementarlos para comprobar un resultado final de la presentación del proyecto.

-Junto a la realización de la documentación llega lo más importante del proyecto antes de presentarlo: las pruebas. Las pruebas son fundamentales para comprobar el correcto funcionamiento de todas las funciones de la aplicación.

Usar un diagrama de Gantt en el proceso de gestión de proyectos ha proporcionado las siguientes ventajas:

- Claridad.
- Una vista general simplificada.
- Datos sobre el rendimiento.
- Una mejor gestión del tiempo.
- Flexibilidad.

1.4. Motivación del proyecto

La motivación de este proyecto viene a raíz de poder utilizar un lenguaje de programación distinto a los utilizados previamente y poder aprender. Aparte, el tema de visualización de contenido ya sea películas u otros contenidos multimedia es algo que llama la atención porque a día de hoy ¿quién no se sienta después de una jornada de trabajo, o salir de clases y se pone cara a una pantalla a ver contenido?

2. Desarrollo de la web

2.1. Tecnologías utilizadas

-JavaScript

JavaScript es un lenguaje de programación o de secuencias de comandos que te permite implementar funciones complejas en páginas web, cada vez que una página web hace algo más que sentarse allí y mostrar información estática para que la veas, muestra oportunas actualizaciones de contenido, mapas interactivos, animación de Gráficos 2D/3D, desplazamiento de máquinas reproductoras de vídeo, etc., puedes apostar que probablemente JavaScript está involucrado.

-Node.js

Este es un entorno de tiempo de ejecución de JavaScript, de ahí su terminación «.js». Este entorno de tiempo es open source, es decir, de código abierto, multiplataforma y que se ejecuta del lado del servidor.

Uno de los puntos fuertes de Node.js es *Event Loop*, también conocido como Bucle de eventos, un modelo que permite gestionar una gran cantidad de eventos de forma **asíncrona**, ya que se ejecutan de forma independiente y sin interferir unos en otros.

Para conseguir esto, Node.js ha modificado la forma en la que se realizan las conexiones al servidor. En vez de generar un hilo para cada cliente, algo que resulta muy ineficaz debido al alto consumo de memoria RAM ante múltiples conexiones, utiliza un modelo en el que genera un evento para cada petición que se gestiona de manera independiente y sin bloqueos.

Esta capacidad de dar respuesta a muchas más peticiones de forma concurrente, hace de Node.js un entorno muy estable y con gran rendimiento, especialmente para proyectos de gran envergadura.

-Express.js

Express.js es un framework de backend Node.js minimalista y rápido, que proporciona características y herramientas robustas para desarrollar aplicaciones de backend escalables. Te ofrece el sistema de enrutamiento y características simplificadas para

ampliar el framework con componentes y partes más potentes en función de los casos de uso de tu aplicación.

-EJS

En el desarrollo web, la extensión de archivo **.ejs** representa el tipo de archivo Embedded JavaScript Template (**.ejs**). Embedded JavaScript (EJS) es una biblioteca de código abierto escrita en JavaScript para el procesamiento de plantillas del lado del cliente, ampliamente utilizada en el desarrollo de aplicaciones web y en la nube, por ejemplo, con plataformas de aplicaciones basadas en JavaScript como Node.js. Las plantillas EJS sirven para generar documentos HTML o XML dinámicos ('vistas') en el lado del cliente utilizando matrices de datos JSON recuperados del servidor.

-MySQL

MySQL es un sistema de gestión de bases de datos que cuenta con una doble licencia. Por una parte es de código abierto, pero por otra, cuenta con una versión comercial gestionada por la compañía Oracle.

MySQL presenta algunas ventajas que lo hacen muy interesante para los desarrolladores. La más evidente es que trabaja con bases de datos relacionales, es decir, utiliza tablas múltiples que se interconectan entre sí para almacenar la información y organizarla correctamente.

Al ser basada en código abierto es fácilmente accesible y la inmensa mayoría de programadores que trabajan en desarrollo web han pasado usar MySQL en alguno de sus proyectos porque al estar ampliamente extendido cuenta además con una ingente comunidad que ofrece soporte a otros usuarios.

Pero estas no son las únicas características como veremos a continuación:

Arquitectura Cliente y Servidor: MySQL basa su funcionamiento en un modelo cliente y servidor. Es decir, clientes y servidores se comunican entre sí de manera diferenciada para un mejor rendimiento. Cada cliente puede hacer consultas a través del sistema de registro para obtener datos, modificarlos, guardar estos cambios o establecer nuevas tablas de registros, por ejemplo.

Compatibilidad con SQL: SQL es un lenguaje generalizado dentro de la industria. Al ser un estándar MySQL ofrece plena compatibilidad por lo que si has trabajado en otro motor de bases de datos no tendrás problemas en migrar a MySQL.

Vistas: Desde la versión 5.0 de MySQL se ofrece compatibilidad para poder configurar vistas personalizadas del mismo modo que podemos hacerlo en otras bases de datos SQL. En bases de datos de gran tamaño las vistas se hacen un recurso imprescindible.

Procedimientos almacenados. MySQL posee la característica de no procesar las tablas directamente sino que a través de procedimientos almacenados es posible incrementar la eficacia de nuestra implementación.

Desencadenantes. MySQL permite además poder automatizar ciertas tareas dentro de nuestra base de datos. En el momento que se produce un evento otro es lanzado para actualizar registros u optimizar su funcionalidad.

Transacciones. Una transacción representa la actuación de diversas operaciones en la base de datos como un dispositivo. El sistema de base de registros avala que todos los procedimientos se establezcan correctamente o ninguna de ellas. En caso por ejemplo de una falla de energía, cuando el monitor falla u ocurre algún otro inconveniente, el sistema opta por preservar la integridad de la base de datos resguardando la información.

-Xampp

XAMPP es un servidor independiente de plataforma de código libre. Te permite instalar de forma sencilla Apache en tu propio ordenador, sin importar tu sistema operativo (Linux, Windows, MAC o Solaris). Y lo mejor de todo es que su uso es gratuito.

13 Proyecto 2º DAW Curso 21 / 22 XAMPP incluye además servidores de bases de datos como MySQL y SQLite con sus respectivos gestores phpMyAdmin y phpSQLiteAdmin. Incorpora también el intérprete de PHP, el intérprete de Perl, servidores de FTP como ProFTPD ó FileZilla FTP Serve, etc. entre muchas cosas más.

-Visual Studio Code

Visual Studio Code (VS Code) es un editor de código fuente desarrollado por Microsoft. Es software libre y multiplataforma. VS Code tiene una buena integración con Git, cuenta con soporte para depuración de código, y dispone de un sinnúmero de extensiones, que

básicamente te da la posibilidad de escribir y ejecutar código en cualquier lenguaje de programación.

VS Code tiene una gran variedad de características útiles para agilizar el trabajo, que lo hacen el editor preferido por muchos (me incluyo) para trabajar los proyectos.

Multiplataforma: Es una característica importante en cualquier aplicación y más si trata de desarrollo. Visual Studio Code está disponible para Windows, GNU/Linux y macOS.

IntelliSense: Esta característica está relacionada con la edición de código, autocompletado y resaltado de sintaxis, lo que permite ser más ágil a la hora de escribir código. Como su nombre lo indica, proporciona sugerencias de código y terminaciones inteligentes en base a los tipos de variables, funciones, etc. Con la ayuda de extensiones se puede personalizar y conseguir un IntelliSense más completo para cualquier lenguaje.

Depuración: Visual Studio Code incluye la función de depuración que ayuda a detectar errores en el código. De esta manera, nos evitamos tener que revisar línea por línea a puro ojo humano para encontrar errores. VS Code también es capaz de detectar pequeños errores de forma automática antes de ejecutar el código o la depuración como tal.

Uso del control de versiones: Visual Studio Code tiene compatibilidad con Git, por lo que puedes revisar diferencias o lo que conocemos con git diff, organizar archivos, realizar commits desde el editor, y hacer push y pull desde cualquier servicio de gestión de código fuente (SMC). Los demás SMC están disponible por medio de extensiones.

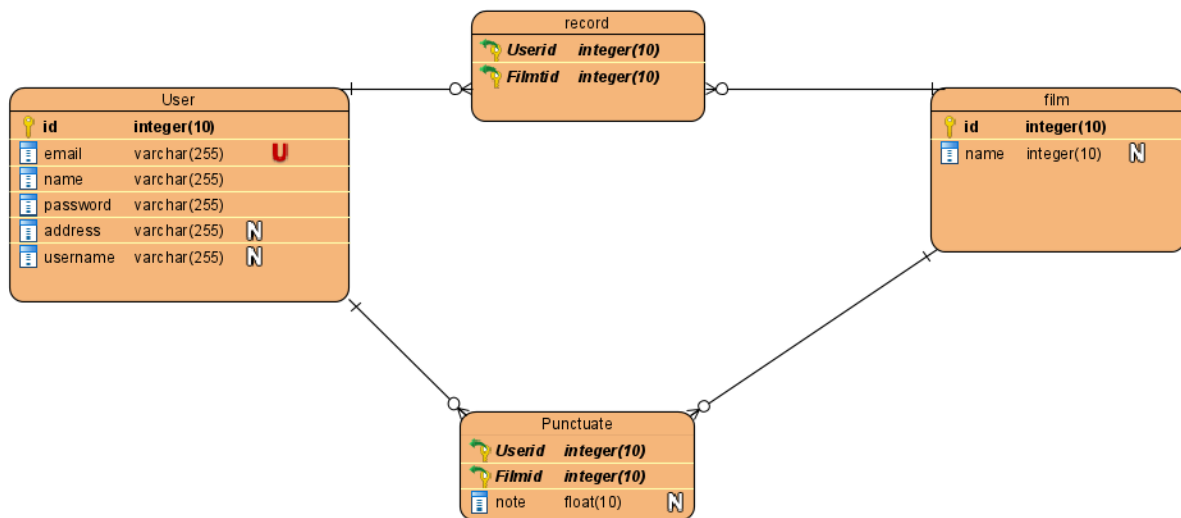
Extensiones: Hasta ahora, he mencionado varias veces el término extensiones porque es uno de los puntos fuertes. Visual Studio Code es un editor potente y en gran parte por las extensiones. Las extensiones nos permiten personalizar y agregar funcionalidad adicional de forma modular y aislada. Por ejemplo, para programar en diferentes lenguajes, agregar nuevos temas al editor, y conectar con otros servicios. Realmente las extensiones nos permiten tener una mejor experiencia, y lo más importante, no afectan en el rendimiento del editor, ya que se ejecutan en procesos independientes.

3. Estructura y funcionalidades

3.1. Capa de persistencia

Una capa de persistencia encapsula el comportamiento necesario para mantener los objetos. O sea: leer, escribir y borrar objetos en la base de datos. La persistencia de la información es la parte más crítica en una aplicación de software. MySQL es la base de datos utilizada. A continuación, se observará el diagrama entidad-relación de esta base de datos

Un diagrama entidad-relación, también conocido como modelo entidad relación o ERD, es un tipo de diagrama de flujo que ilustra cómo las "entidades", como personas, objetos o conceptos, se relacionan entre sí dentro de un sistema.



3.2. Capa lógica

Los componentes de software de las aplicaciones se pueden visualizar como residentes en un número de capas lógicas. Estas capas representan la independencia física y lógica de los componentes de software en función de la naturaleza de los servicios que proporcionan. A continuación, se van a explicar las capas que componen

```

//Register
app.post('/register', async (req, res) => {
  const name = req.body.name;
  const email = req.body.email;
  const address = req.body.address;
  const username = req.body.username;
  const password = req.body.password;
  const passwordrepeat = req.body.passwordrepeat;
  let passwordHash = await bcryptjs.hash(password, 8);

  connection.query('SELECT * FROM users WHERE username = ?', [username], async (error, results) => {
    if (results.length == 0) {
      if (password == passwordrepeat) {
        connection.query('INSERT INTO users SET ?', { name: name, email: email, address: address, username: username, password: passwordHash }, async (error, results) => {
          if (error) {
            res.render('register', {
              alert: true,
              alertTitle: "Registro",
              AlertMessage: "Registro fallido",
              alertIcon: 'error',
              showConfirmButton: false,
              timer: 2500,
              ruta: 'register'
            })
          } else {
            res.render('register', {
              alert: true,
              alertTitle: "Registro",
              AlertMessage: "¡Registro realizado!",
              alertIcon: 'success',
              showConfirmButton: false,
              timer: 2500,
              ruta: 'login'
            })
          }
        })
      } else {
        res.render('register', {
          alert: true,
          alertTitle: "Registro",

```

```

132      }
133    } else {
134      res.render('register', {
135        alert: true,
136        alertTitle: "Registro",
137        AlertMessage: "Las contraseñas no coinciden",
138        alertIcon: 'error',
139        showConfirmButton: false,
140        timer: 2500,
141        ruta: 'register'
142      })
143    }
144  } else {
145    res.render('register', {
146      alert: true,
147      alertTitle: "Registro",
148      AlertMessage: "Usuario ya registrado",
149      alertIcon: 'error',
150      showConfirmButton: false,
151      timer: 2500,
152      ruta: 'register'
153    })
154  }
155 })
156
157 })
158

```

El registro consta de un formulario que rellenas con tus datos, donde al terminar haces un submit. Desde el front, antes de realizar el submit se comprueba que los datos del formulario están en un formato correcto. Una vez los datos están en un formato correcto se comprueba si

el usuario existe ya en la BBDD. En caso de no existir, se guardan los datos del formulario, entre ellos la contraseña encriptada.

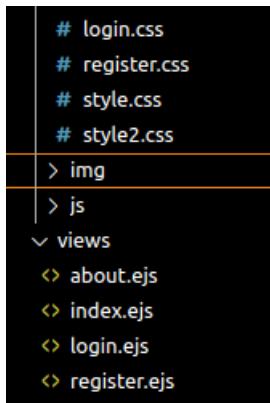
```
//Login
app.post('/auth', async (req, res) => {
  const user = req.body.username;
  const pass = req.body.password;
  let passwordHash = await bcryptjs.hash(pass, 8);

  if (user && pass) {
    connection.query('SELECT * FROM users WHERE username = ?', [user], async (error, results) => {
      if (results.length == 0 || !(await bcryptjs.compare(pass, results[0].password))) {
        res.render('login', {
          alert: true,
          alertTitle: "Inicio sesión",
          AlertMessage: "¡Inicio de sesión incorrecto!",
          alertIcon: 'warning',
          showConfirmButton: true,
          timer: 2000,
          ruta: 'login'
        });
      } else {
        req.session.loggedin = true;
        req.session.name = results[0].name;
        req.session.userId=results[0].id;
        res.render('login', {
          alert: true,
          alertTitle: "Login",
          AlertMessage: "¡Incorrect login!",
          alertIcon: 'warning',
          showConfirmButton: true,
          timer: false,
          ruta: ''
        });
      }
    });
  }
});
})
```

El Login es un formulario sencillo donde pones tu usuario y contraseña y aprietas al botón de submit. El front comprueba que están rellenos los dos inputs y, una vez comprobado esto, se comprueba que el usuario existe en la BBDD.

3.3. Capa de presentación

La capa de presentación es la que ve el usuario (también se la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo proceso.

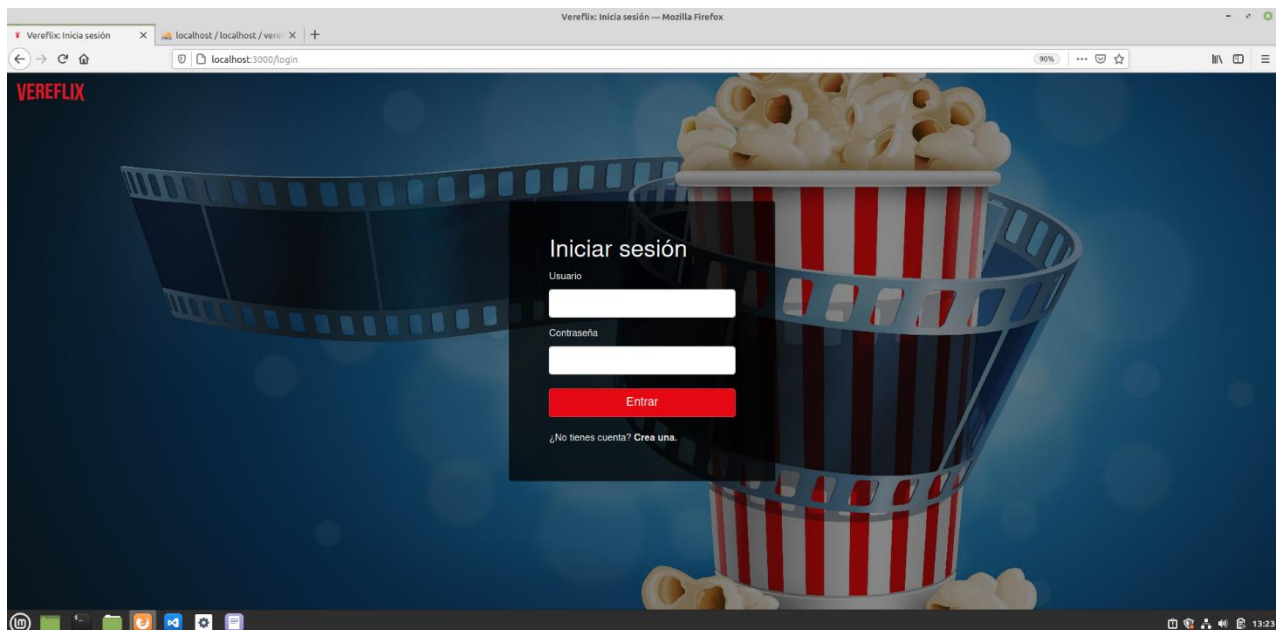


En el proyecto se utiliza el modelo **ejs**, explicado anteriormente. En el se utiliza lenguaje html.

4. Manual de usuario

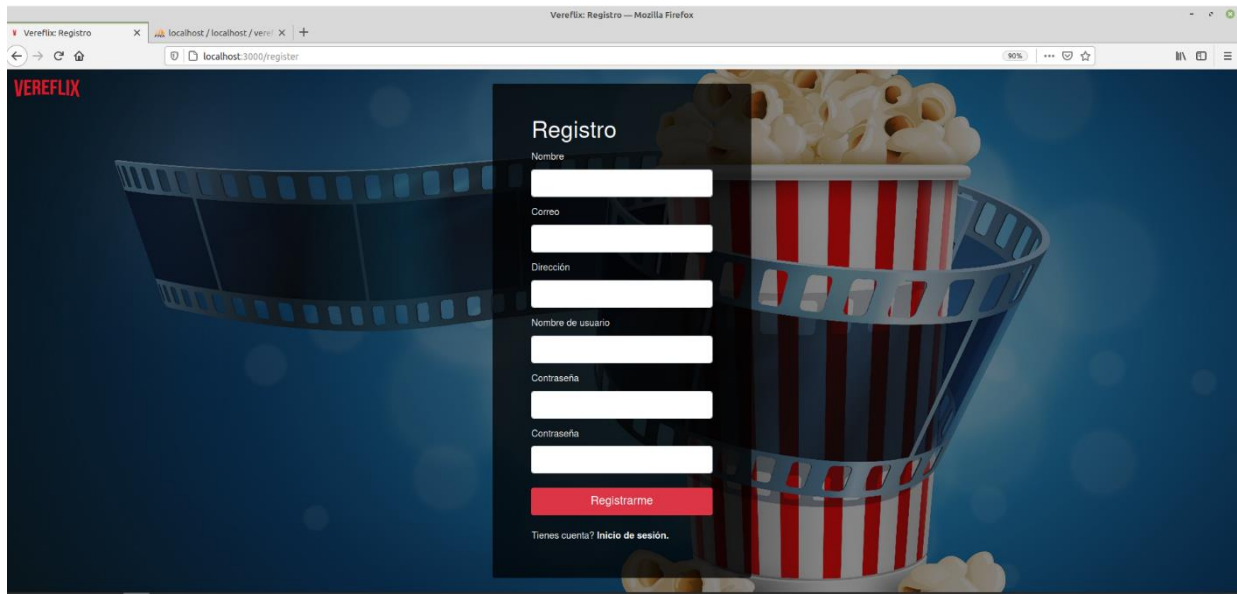
El proyecto desarrollado consta de una plataforma de visualización de películas donde el usuario inicia sesión.

Para empezar a interactuar con la aplicación, primero, el usuario debe estar añadido a la base de datos. Para acceder a la cuenta deberá de rellenar un formulario de inicio de sesión donde va a tener que introducir su usuario y contraseña.



En caso de fallar con el usuario o la contraseña la aplicación te lo dirá.

También se puede dar el caso de que no se haya dado de alta al sistema. Entonces deberá pulsar donde pone “Crea una”. Aparecerá un formulario de registro, donde, si rellenas bien los datos, el usuario se dará de alta.

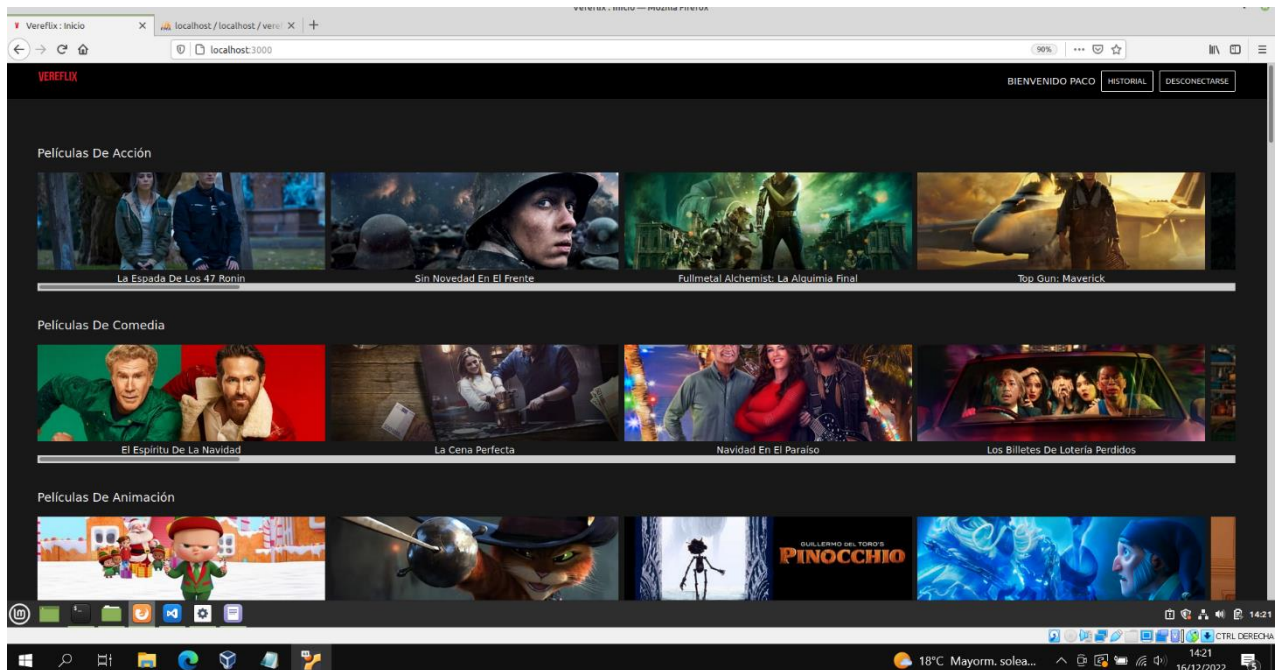


The screenshot shows a web browser window with the title "Vereflix: Registro - Mozilla Firefox". The address bar shows "localhost:3000/register". The page features a dark blue background with a film strip and a popcorn bucket. A white registration form is centered on the page. The form includes the following fields and elements:

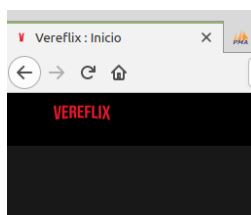
- Registro** (Section Header)
- Nombre** (Text input field)
- Correo** (Text input field)
- Dirección** (Text input field)
- Nombre de usuario** (Text input field)
- Contraseña** (Text input field)
- Contraseña** (Text input field)
- Registrarme** (Red button)
- Tienes cuenta? Inicio de sesión.** (Text link)

En caso de fallar al repetir el password, o poner un usuario ya existente, la aplicación te lo dirá. También en caso de querer volver al inicio de sesión disponemos de un botón debajo del todo donde pone “Inicio de sesión”.

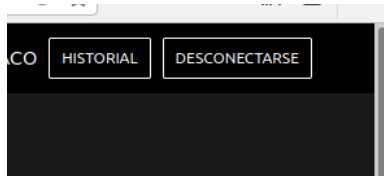
Hemos accedido correctamente mediante el login, donde al entrar se podrá ver la página de inicio.



En esta página disponemos de 3 botones.

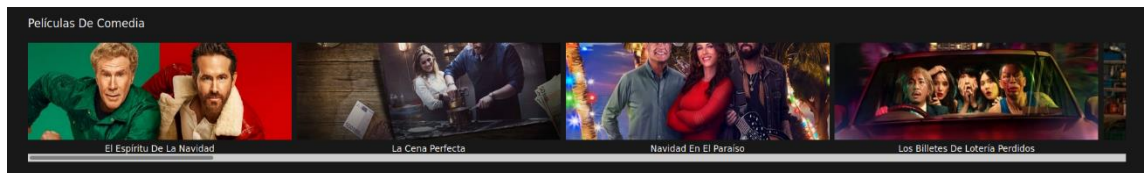


El primer botón se encuentra arriba a la izquierda (el logo) y nos permite volver a la página principal en cualquier momento.

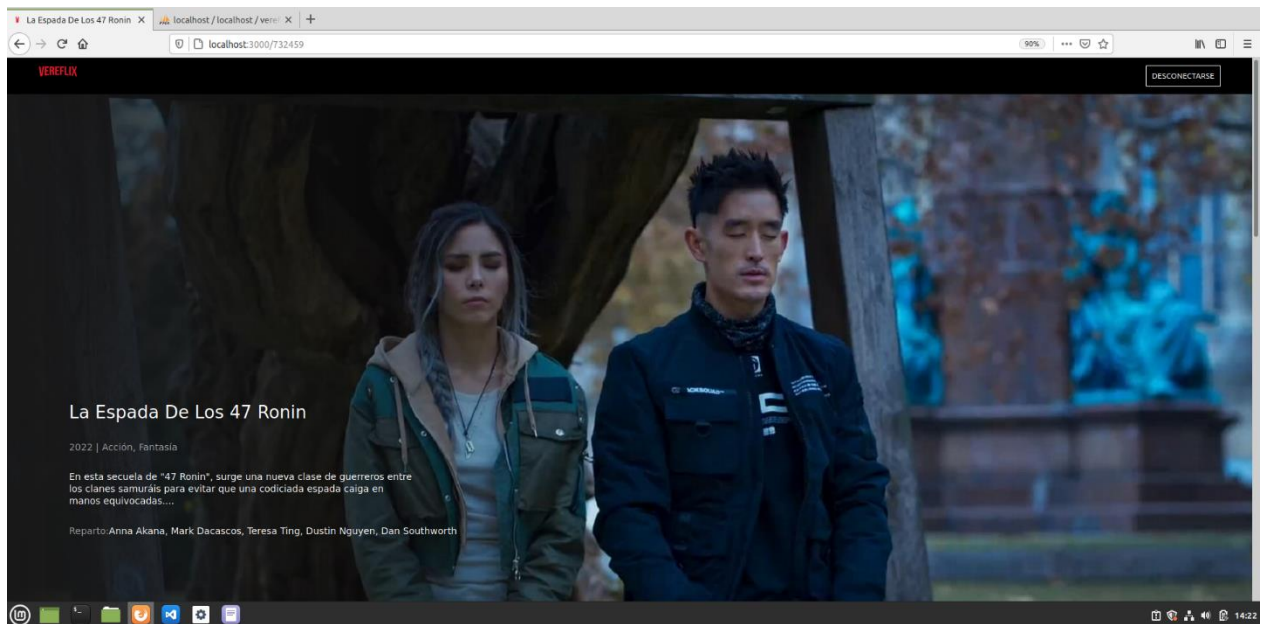


Y el segundo y tercer botón se encuentran arriba a la derecha junto al nombre del usuario. El botón historial y desconectarse, que como su nombre indica, sirven para mostrar el historial de usuario o desconectarse de la cuenta respectivamente.

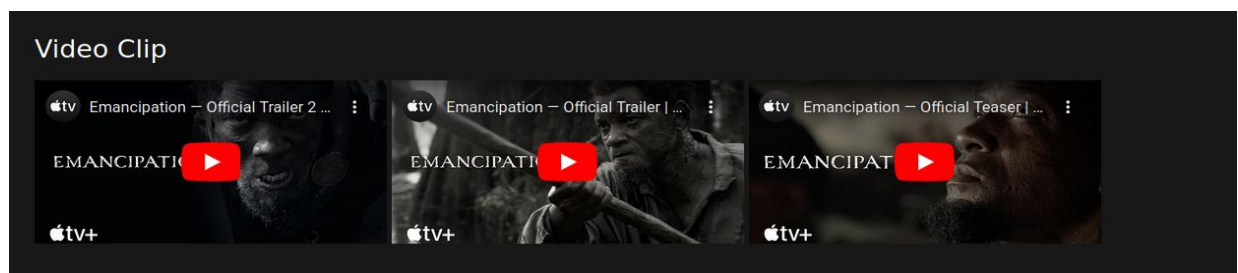
También si se pulsa en cualquier película que haya en el inicio (en cualquier imagen que vea), le redireccionará a una página donde podrá ver una serie de información sobre dicha película. Cada barra de categoría de película cuenta también con 2 botones a los laterales o si lo desea cuenta con una barra de carrusel para poder ir desplazándose y selecciona el título que más guste.



Como se puede observar, disponemos del botón de inicio (el logo) y el de desconectarse, además de una serie de información de dicha película.



En la misma página si bajamos, nos encontraremos con una serie de videos de dicha película, ya sea tráilers o un pequeño fragmento de esta.



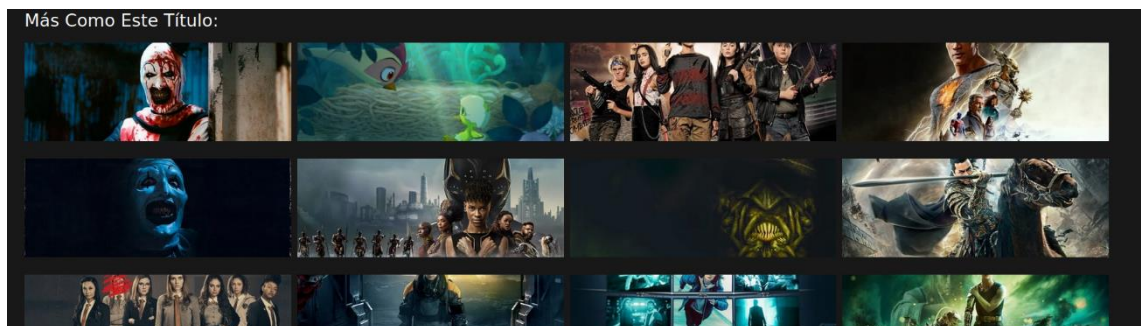
Justo debajo de esta sección disponemos de un apartado donde podemos poner una puntuación a esta película. También podemos ver una valoración media de toda la gente que la ha votado.

Valoración:

Enviar valoración

Valoración media:

Y para concluir con la página, disponemos de una sección de títulos similares al que hemos escogido.



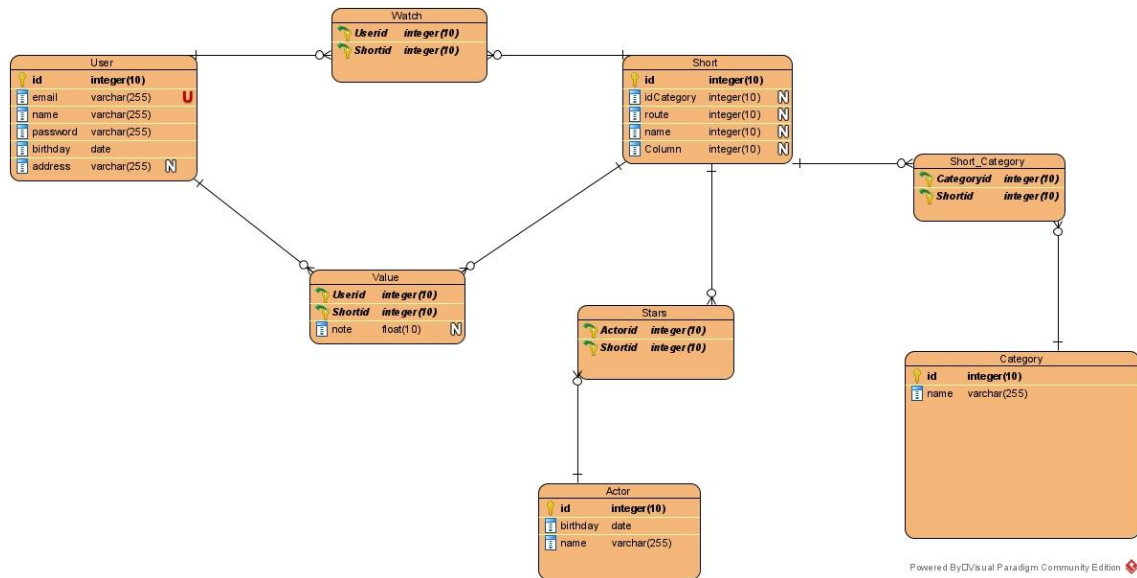
5. Conclusión

5.1. Problemas encontrados

Durante la realización del proyecto se han encontrado una serie de problemas que han dificultado su propio desarrollo. Estos son:

-Al principio se pensó en hacer una web de cortos, pero el problema surgió que no habían apis o alguna fuente para extraerlos. Casualmente buscando apis de cortos, se halló una api de películas la cual se ha aprovechado para este proyecto.

Aquí hay un ER de dicha propuesta:



- Tras muchos intentos ha sido imposible poder hacer una barra de búsqueda.

-A la hora de la realización del proyecto se partía de una base muy floja del lenguaje de programación utilizado y de las distintas herramientas.

5.2. Mejoras

Lo mejor de este proyecto es la gran cantidad de opciones extra e implementaciones que se pueden aplicar. Durante el desarrollo se barajaron la implementación de muchas mejoras para una futura versión. Entre ellas están:

- La barra de navegación para encontrar instantáneamente el título deseado.
- Una sección de editar perfil para el usuario.
- Poder escoger la categoría que quieras y ver solo títulos de la misma.

6. Bibliografía

<https://ull-esit-dsi-1617.github.io/estudiar-cookies-y-sessions-en-expressjs-alejandro-raul-35l2-p4/sessionsexpress.html>

<https://www.digitalocean.com/community/tutorials/how-to-use-ejs-to-template-your-node-application-es>

https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function

<https://expressjs.com/es/guide/routing.html>

<https://www.geeksforgeeks.org/express-js-app-get-request-function/>

<https://asfo.medium.com/desarrollando-una-sencilla-api-rest-con-nodejs-y-express-cab0813f7e4b>

<https://www.tutorialesprogramacionya.com/javascriptya/nodejsya/>

https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction