Alumno: Miguel Cortés Cambeses

Pec 1

URL git: https://github.com/mcortescambeses/lupin_pec1

URL web: https://serene-frangollo-0274f8.netlify.app/

1. Creación del boilerplate basado en Parcel:

Descripción: En este paso, he configurado un boilerplate utilizando Parcel como herramienta de bundling y transpilación para gestionar las dependencias y optimizar el proyecto. Para ello he seguido los pasos que se nos explicaba en el Módulo 2 de esta asignatura.

Entorno de Desarrollo:

Descripción: El entorno de desarrollo se configura utilizando Parcel como bundler. Parcel simplifica el proceso de construcción y desarrollo al manejar automáticamente la configuración. No se necesitan configuraciones complejas para comenzar.

Entorno de Producción:

Descripción: Parcel proporciona una fácil configuración para el entorno de producción mediante el comando npm run build. Este comando ejecuta Parcel en modo de construcción, optimizando y minimizando los recursos para su implementación en producción. Además se ha configurado de forma que elimine de cada vez los archivos creaos en la carpeta dist, para que no se haga muy grande.

Configuración del Soporte para Navegadores Antiguos:

Browserslist: "> 0.5%, last 2 versions, not dead"

Descripción: Se utiliza la configuración de Browserslist en el archivo package.json para indicar a Parcel los navegadores que se pretende dar soporte. En este caso, se especifica que se debe dar soporte a navegadores con una participación de mercado superior al 0.5%, las dos versiones más recientes y aquellos que no se consideran obsoletos ("not dead").

Razón: Parcel es una opción eficiente y fácil de configurar que automatiza muchas tareas comunes en el desarrollo web, como la transpilación de código, la gestión de dependencias y la creación de bundles para producción. Esto ayuda a simplificar el proceso de desarrollo.

Decisiones técnicas: Se eligió Parcel por su configuración sencilla y su capacidad para manejar automáticamente diversas tareas, como la instalación de dependencias, la transpilación de código y la creación de bundles. Otras opciones consideradas fueron Webpack.

Resultados: Se obtuvo un entorno de desarrollo y producción configurado automáticamente con el soporte necesario para transpilación y optimización. Parcel generó la estructura de carpetas y archivos necesarios para el proyecto.

2. Gestión de dependencias: pre- o postprocesadores y dependencias adicionales:

Descripción: En este paso, se gestionaron las dependencias del proyecto, incluyendo preprocesadores y otras dependencias adicionales.

Razón: La gestión adecuada de dependencias es crucial para garantizar que el proyecto funcione correctamente y para mantener un entorno de desarrollo eficiente.

Decisiones técnicas: Se incluyeron dependencias como Babel para transpilación de código, Autoprefixer. Se decidió utilizar Browserslist para declarar el soporte de navegadores y permitir que las herramientas utilicen esta información.

Resultados: Se logró una gestión de dependencias eficiente, lo que facilitó el desarrollo y aseguró la compatibilidad del código con diferentes navegadores.

3. Creación del repositorio Git:

Descripción: Se creó un repositorio Git para el control de versiones del proyecto.

Razón: El control de versiones es esencial para realizar un seguimiento de los cambios, colaborar con otros desarrolladores y mantener la integridad del código.

Decisiones técnicas: Se eligió utilizar Git como sistema de control de versiones y se creó un repositorio en una plataforma como GitHub para facilitar la colaboración.

4. Adecuación a la temática y estructura de la práctica:

Descripción: Se adaptó el proyecto a la temática y estructura específicas de la práctica, asegurando que el diseño y la estructura reflejen los requisitos del proyecto. En este paso lo principal era seleccionar el tema concreto, que en mi caso fue la serie Lupin.

Decisiones técnicas: Se tomaron decisiones relacionadas con la disposición de los elementos en la interfaz, la elección de colores y estilos que se alinean con la temática de Lupin, y la organización de la información según la estructura propuesta. Primero decidí que la portada fuera con una imagen de fondo y las 3 opciones del menú de navegación en el centro, para llamar la atención del usuario desde un primer momento.

Luego en la pestaña de categorías, decidí hacer una lista con 3 items destacables en la serie. Cada uno de ellos, está compuesto de más elementos en detalle, a los que se puede acceder ya que son links. Por ahora solo los 2 primeros actores tienen funcionalidad.

Una vez pinches en uno de los actores, se muestra la pantalla de detalles del mismo, que tiene una estructura básica de párrafos de rasgos generales, un apartado de listado de películas, una fotografía y un pequeño video de YouTube. Al final de la misma tendremos un enlace para movernos al siguiente actor, y así interactuar entre las páginas de detalle del mismo nivel.

En cuanto a la página de presentación y índice, no hay mucho que comentar de su diseño, me ciño a seguir las instrucciones dadas en el enunciado de la práctica.

Resultados: Se obtuvo una web con un diseño coherente y una estructura que cumplía con los requisitos de la práctica, proporcionando una experiencia de usuario consistente con información sobre la serie Lupin.

5. Diseño responsive, complejidad y estética:

Descripción: Se implementó un diseño responsive para garantizar que el proyecto sea accesible y funcione bien en una variedad de dispositivos y tamaños de pantalla.

Razón: La accesibilidad y la adaptabilidad son fundamentales para llegar a una audiencia más amplia y garantizar una experiencia de usuario positiva en diferentes dispositivos.

Decisiones técnicas: Se utilizó CSS media queries para adaptar el diseño a diferentes tamaños de pantalla y se realizaron pruebas en varios dispositivos para garantizar la usabilidad. Se

priorizó la simplicidad y la estética para una experiencia de usuario agradable. Además también se hace alternativas a las imágenes y vídeos por si hay algún problema a la hora de cargalos.

Resultados: Se logró un diseño responsive que se ve y funciona bien en una variedad de dispositivos, manteniendo una estética atractiva y evitando la complejidad innecesaria.

6. Semántica y accesibilidad:

Descripción: Se aseguró de que el código HTML siga prácticas semánticas y se implementaran características de accesibilidad para garantizar que el proyecto sea comprensible para máquinas y accesible para personas con discapacidades.

Razón: La semántica mejora la comprensión del código y facilita la navegación para las herramientas de asistencia, mientras que la accesibilidad es esencial para garantizar que todas las personas puedan interactuar con el proyecto.

Decisiones técnicas: Se utilizaron elementos semánticos HTML5 apropiados (por ejemplo, <header>, <section>, <article>) y se implementaron atributos

7. Publicación en internet.

Usamos la herramienta netify para hacer el deploy de nuestra web.

URL: https://serene-frangollo-0274f8.netlify.app/