Estandares de calidad de SoFware

Luis Cano

Gráfico

Descripción generada automáticamente

Contenido

[Implementación de un linter 2](#_Toc184147250)

[Requerimiento funcional 2](#_Toc184147251)

[Requerimiento no funcional 3](#_Toc184147252)

[Análisis de aspecto ISO 27001 presente en mi proyecto 4](#_Toc184147253)

[Análisis basado en IEEE 830 4](#_Toc184147254)

# Implementación de un linter

Para la implementación de un linter en nuestro código debemos primeramente instalarlo en el entorno de desarrollador por medio de la línea de comandos y fijándose estar en la carpeta raíz del proyecto:

Texto

Descripción generada automáticamente

Un linter es una herramienta que analiza el código fuente para identificar errores, inconsistencias o malas prácticas, ayudando a mantener un código limpio y uniforme. El comando “npm install standard -D” añade a “standard” como una dependencia de desarrollo, además se debe agregar su configuración en el “package.json”. Para comprobar su correcta instalación se puede ejecutar el comando “npx standard” y señalará errores o malas prácticas en el código:

Interfaz de usuario gráfica

Descripción generada automáticamente

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

# Requerimiento funcional

* El sistema debe permitir al usuario buscar cartas de la API mediante un campo de texto, mostrando solo aquellas cuyo nombre coincida parcial o completamente con el texto ingresado.

Texto

Descripción generada automáticamente con confianza media

Texto

Descripción generada automáticamente

La variable “results” almacena todas las cartas si el campo de búsqueda está vacío, en el caso contrario, filtra las cartas que en el nombre contenga el texto ingresado, ignorando mayúsculas y minúsculas.

El input que captura el texto de búsqueda está conectado al estado “search” mediante la función “handleSearch”, gracias a esto se logra que cualquier texto ingresado en el campo de búsqueda influya en los resultados mostrados.

# Requerimiento no funcional

* El sistema debe renderizar los resultados de la búsqueda de manera eficiente, asegurando tiempos de respuesta rápidos y una experiencia fluida al interactuar con el input de búsqueda.

Texto

Descripción generada automáticamente

El uso de “useState” para manejar el estado de las cartas y el texto de búsqueda garantiza actualizaciones rápidas y controladas. Al actualizar “search”, el componente no vuelve a renderizar toda la lista desde la API, sino que filtra localmente en la variable. Esto mejora el rendimiento, ya que evita solicitudes innecesarias.

# Análisis de aspecto ISO 27001 presente en mi proyecto

* ISO 27001: A.12.6.1 - Gestión de vulnerabilidades técnicas: este control busca asegurarse de que se identifican, evalúan y mitigan las vulnerabilidades técnicas presentes en un sistema.

Uso de HTTPS para la comunicación con la API externa: esto asegura que los datos transferidos entre mi aplicación y la API están cifrados, evitando que puedan ser interceptados o manipulados por terceros malintencionados.



# Análisis basado en IEEE 830

* Ausencia de requisitos documentados y medibles:

El estándar IEEE 830-1998 enfatiza la necesidad de documentar los requisitos de software de manera clara, completa y verificable. Actualmente, el proyecto no incluye una especificación formal de requisitos, lo que puede dificultar la evaluación de si cumple con las expectativas o necesidades de los usuarios (en este caso es de manera hipotética).

* Implicaciones:
  + Sin requisitos documentados, no hay una referencia objetiva para validar que el sistema funciona correctamente.
  + Es difícil garantizar que se cubren todas las necesidades del usuario o identificar qué aspectos deben priorizarse en futuras iteraciones.