Documento de Estilos de Código

Este documento establece ciertas normas y convenciones que deben de seguirse para el proyecto “Tango App”, el cual es desarrollado con el lenguaje de programación JavaScript junto con CSS y HTML, con React como biblioteca principal.

Índice

---

1. Estructura del Proyecto

- Mantener una estructura clara y coherente para archivos y carpetas.
- Utilizar nombres descriptivos para cada componente y módulo.
- Evitar la redundancia de código, reutilizar si es posible.

2. Nomenclatura

- Utilizar camelCase para nombrar variables y funciones.
- Utilizar PascalCase para nombrar clases y componentes.
- Utiliza nombres descriptivos y significativos para variables, funciones y componentes.
- Evitar el uso de abreviaciones en nombres de variables o funciones.

3. Formato y espaciado

- Mantener una indentación consistente.
- JavaScript:
    - Usar punto y coma (;) al final de cada declaración.
    - Usar comillas simples (') para cadenas de texto.
- CSS:
    - Mantener una línea en blanco entre reglas de estilo.
- HTML:
    - Asegurarse de cerrar correctamente las etiquetas.

4. Comentarios

- Utilizar comentarios para explicar el propósito y funcionamiento del código cuando sea necesario aclararlo

5. Buenas prácticas con JavaScript:

- Usar const y let siempre que se pueda.
    - const para valores constantes.
    - let para variables que pueden cambiar.
    - Evitar el uso de var.
- Uso de arrow functions si es posible para mantener el código más claro y conciso.

6. Buenas prácticas con HTML:

- Declarar el <!DOCTYPE html> para una mejor compatibilidad entre navegadores.
- Cerrar cada etiqueta en el orden correcto.
- Evitar el uso excesivo de etiquetas , usando etiquetas como <header>, <main>.

7. Buenas prácticas con CSS:

- Uso de **Flexbox** para organizar los elementos de forma eficiente, para manejar layouts en lugar de utilizar posicionamiento absoluto o flotantes.
- Diseño responsive mediante media queries.

8. Buenas practicas con React

- Los componentes deben ser pequeños, reutilizables y cumplir con una sola responsabilidad. Evitar hacer componentes que manejen múltiples tareas.
- Utilizar componentes funcionales siempre que sea posible en lugar de Clases.
- Usar los React hooks como useState, useEffect, useContext, entre otros.

9. Conclusión

Este documento fue diseñado para optimizar la legibilidad, mantenibilidad y colaboración en el código, asegurando que todos los miembros del equipo trabajen con una base “uniforme”, se recomienda a los involucrados revisar estas normas de manera regular, y proponer

mejoras continuas, con el objetivo de un desarrollo más eficiente con un código más limpio y fácil de gestionar, de adaptar.