

Check list de código

Equipo: Vm's

Nombre del verificador: Carlos Genaro Cutz Anguas

Fecha (D/M/Y) :30/09/2024

Código revisado: Login Auth.

Criterios	Si	No
1. Organización del Código		
1.1 ¿Se organiza el código de manera lógica, utilizando adecuadamente los componentes y hooks en React?	X	
1.2 ¿Se siguen buenas prácticas de nomenclatura clara y coherente para nombres de variables y funciones en el código del componente?	X	
2. Uso de Hooks y Funcionalidades de React/Next.js		
2.1 ¿Se usan correctamente los hooks (useSession, useTypewriter) para manejar el estado y efectos en el componente?	X	
2.2 ¿El componente está optimizado para evitar renders innecesarios al utilizar hooks como useSession?	X	
3. Manejo de la Interfaz de Usuario y Enlaces		

3.1 ¿Se gestionan correctamente los enlaces de Next.js (Link) para evitar redirecciones o comportamientos inesperados?	X	
--	---	--

3.2 ¿La interfaz es accesible y reactiva, respondiendo adecuadamente a la interacción del usuario (ej. botones, enlaces, etc.)?	X	
3.3 ¿Se manejan correctamente los eventos de usuario, como clics o envíos de formularios, para asegurar que las acciones se ejecuten de forma controlada y predecible?	X	
4. Manejo de Respuestas y Errores		
4.1 ¿El código maneja adecuadamente la autenticación del usuario (session?.user) para mostrar contenido personalizado o mensajes genéricos?	X	
4.2 ¿Hay una gestión clara de posibles errores en la sesión o en la carga de los datos del usuario?		X
5. Seguridad		
5.1 ¿Se validan las entradas del usuario correctamente (por ejemplo, si el usuario está autenticado)?	X	
5.2 ¿Se protege adecuadamente el acceso a la información personal?	X	
6. Legibilidad y Mantenibilidad		

6.1 ¿El código tiene comentarios claros que explican su funcionalidad, sin ser excesivamente verbosos	X	
6.2 ¿Se sigue el principio de separación de responsabilidades, evitando funciones demasiado largas o con múltiples tareas?	X	
6.3 ¿El código es fácil de leer y mantener por otros desarrolladores?	X	
7. Buenas Prácticas en Next.js		
7.1 ¿Se está utilizando el sistema de APIs de Next.js para manejar correctamente las rutas dinámicas (<code>params</code> , cuerpo de la solicitud)?	X	
7.2 ¿Se evita el uso innecesario de <code>console.log</code> en producción y se utiliza un sistema de logging más apropiado?		X
7. Buenas Prácticas en Next.js		
8.1 ¿El código está preparado para manejar un mayor volumen de usuarios o datos sin problemas de escalabilidad?	X	
8.2 ¿Se siguen prácticas que faciliten la futura migración o integración de nuevas tecnologías en el proyecto?	X	
7. Buenas Prácticas en Next.js		
9.1 ¿Se valida que los datos requeridos (<code>username</code> , <code>email</code> , <code>weight</code>) están presentes y cumplen con el formato adecuado antes de insertarlos en la base de datos?		X
9.2 ¿Se gestionan correctamente las posibles excepciones por datos faltantes o mal formados antes de realizar consultas en la base de datos?		X

Nombre del verificador: _____

Fecha (D/M/Y) : _____

Criterios	Si	No
Caso de uso		
Los botones tienen íconos con atributos aria-hidden para mejorar la accesibilidad.		
¿La interfaz responde adecuadamente a la interacción del usuario (clics, navegación, entradas de datos)?		
Se proporciona texto significativo que puede ser leído por tecnologías de asistencia.		
¿Se está validando correctamente el tipo de datos de las props recibidas?		
Calidad del código		
Se evita la duplicación de código innecesario.		
El código es fácil de leer y seguir, con nombres descriptivos y claros.		
¿Se aplican los principios de React correctamente, usando componentes funcionales, hooks, y separando la lógica de la presentación?		
Eficiencia y rendimiento		
¿Se manejan adecuadamente los cambios de estado para evitar renders innecesarios?		
¿Se optimiza la carga de recursos como imágenes y efectos visuales (p. ej., evitar la carga innecesaria de imágenes)?		

Seguridad		
¿Los enlaces y redirecciones (Link de Next.js) están bien gestionados para prevenir comportamientos inesperados?		
¿Las entradas del usuario están validadas correctamente para evitar ataques de inyección de datos?		
Documentación		
¿Cada componente tiene comentarios que describen su propósito y el uso de props?		
¿Se documentan adecuadamente los componentes, interfaces y funciones para facilitar su comprensión por otros desarrolladores?		
¿Se documenta correctamente el uso de APIs externas o servicios?		
Mantenibilidad		
¿Los componentes son lo suficientemente pequeños y enfocados en una tarea específica, evitando componentes demasiado grandes o complejos?		
¿Los componentes son reutilizables y desacoplados de otros módulos específicos?		
Portabilidad		
¿Se utiliza una combinación de tecnologías que facilita la portabilidad (React, Next.js, TypeScript)?		

¿Se limita el uso de dependencias externas no esenciales para reducir problemas de		
--	--	--

compatibilidad en diferentes entornos?		
Cumplimiento de normativas		
¿El código incluye buenas prácticas para accesibilidad (p. ej., etiquetas “alt” descriptivas, soporte para navegadores con lector de pantalla)?		
¿El código sigue las normativas de seguridad, como el uso de HTTPS y la gestión segura de tokens y sesiones?		