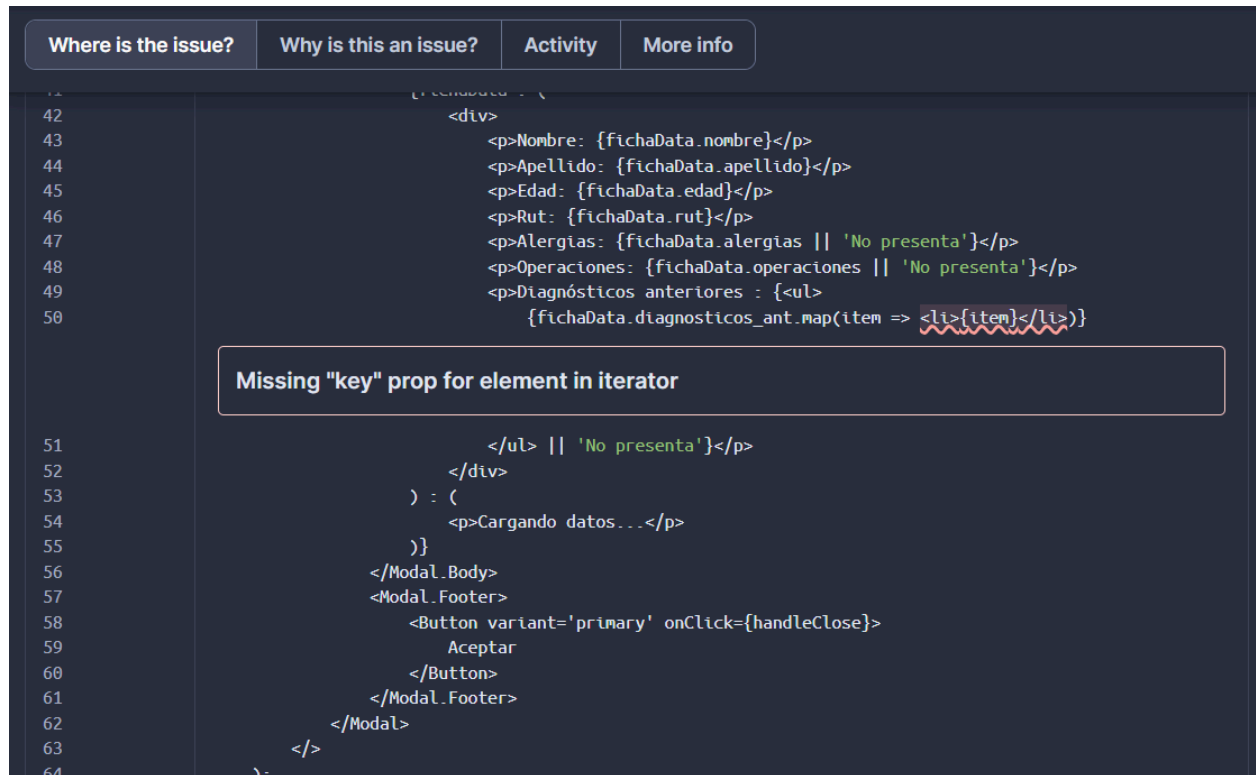


# QUALITY ISSUES

## 1. Missing "key" prop for element in iterator



### Por qué es una issue

Para optimizar la representación de los componentes de lista de React, se requiere un identificador único (UID) para cada elemento de la lista. Este UID permite a React identificar el elemento durante su vida útil. Para proporcionar, use el atributo clave del elemento de la lista. Cuando falta el atributo clave, React utilizará de forma predeterminada el índice del elemento dentro del componente de lista.

Si cambia el orden de los elementos, las claves no coinciden entre las representaciones, lo que recreará el DOM. Puede afectar negativamente al rendimiento y puede causar problemas con el estado del componente.

## Cómo arreglarlo

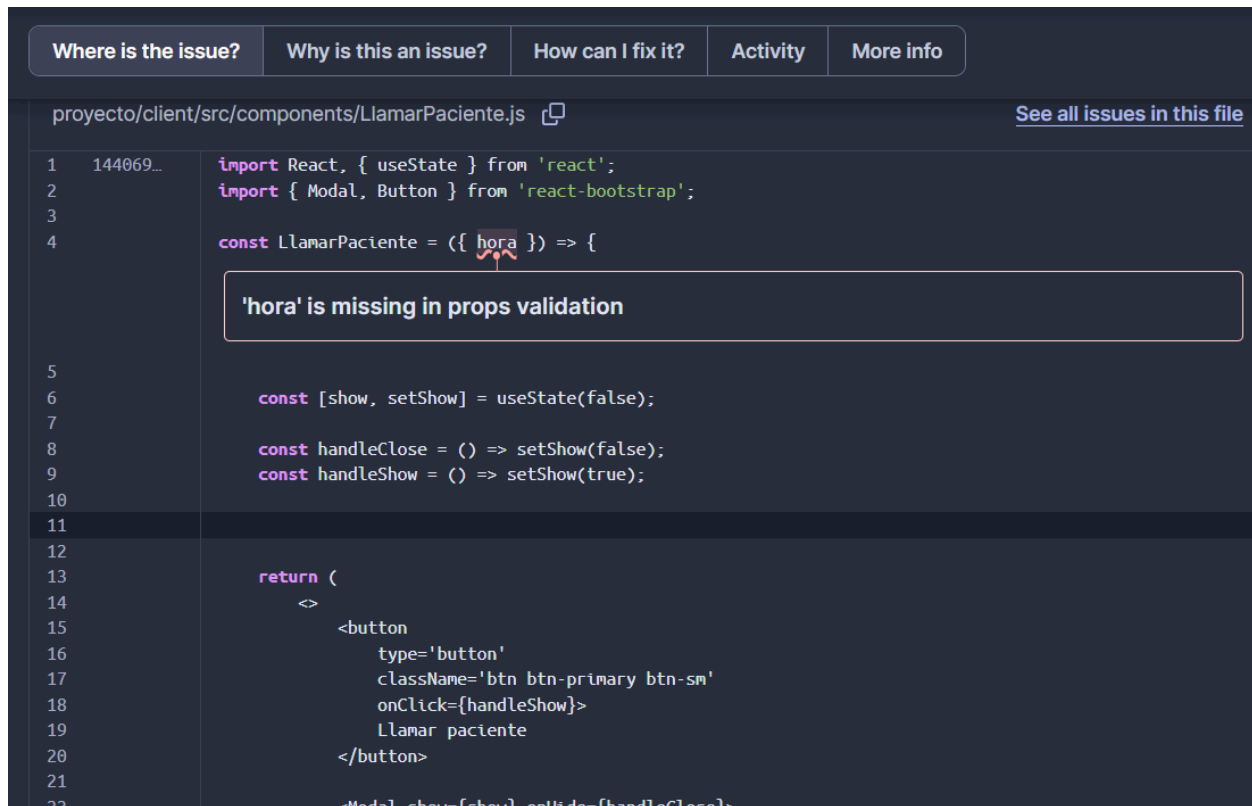
Para solucionarlo, utilice una cadena o un número que identifique de forma única el elemento de la lista. La clave debe ser única entre sus hermanos, no globalmente. Si los datos provienen de una base de datos, los identificadores de base de datos ya son únicos y son la mejor opción. De lo contrario, utilice un contador o un generador de UUID.

Evite utilizar índices de matriz ya que, incluso si son únicos, el orden de los elementos puede cambiar.

```
function Blog(props) {
  return (
    <ul>
      {props.posts.map((post) =>
        <li> <!-- Noncompliant: When 'posts' are reordered, React will need to recreate the list DOM -->
          {post.title}
        </li>
      )}
    </ul>
  );
}
```

```
function Blog(props) {
  return (
    <ul>
      {props.posts.map((post) =>
        <li key={post.id}> <!-- Compliant: id will always be the same even if 'posts' order changes -->
          {post.title}
        </li>
      )}
    </ul>
  );
}
```

## 2. 'hora' is missing in props validation



The screenshot shows a code editor with a file named `proyecto/client/src/components/LlamarPaciente.js`. The code is as follows:

```
1 144069... import React, { useState } from 'react';
2 import { Modal, Button } from 'react-bootstrap';
3
4 const LlamarPaciente = ({ hora }) => {
5
6     const [show, setShow] = useState(false);
7
8     const handleClose = () => setShow(false);
9     const handleShow = () => setShow(true);
10
11
12
13     return (
14         <>
15             <button
16                 type='button'
17                 className='btn btn-primary btn-sm'
18                 onClick={handleShow}>
19                 Llamar paciente
20             </button>
21
22             <Modal show={show} onHide={handleClose}>
```

An error message is displayed in a box above line 11: **'hora' is missing in props validation**. A red squiggly line points from the error message to the `hora` prop in the function signature on line 4.

### Por qué es una issue

En JavaScript, las propiedades se pasan normalmente como objetos simples, lo que puede generar errores y confusión al trabajar con componentes que tienen requisitos específicos de propiedades. Sin embargo, carece de seguridad de tipos y claridad al pasar propiedades a componentes en una base de código.

Al definir tipos para las propiedades de componentes, los desarrolladores pueden aplicar seguridad de tipos y proporcionar documentación clara para las propiedades esperadas de un componente. Esto ayuda a detectar posibles errores en tiempo de compilación. También mejora la capacidad de mantenimiento del código al facilitar la comprensión de cómo se deben usar los componentes y qué propiedades aceptan.

## Cómo arreglarlo

Este es un ejemplo de cómo podría ser la solución proporcionado por SonarCloud:

### Noncompliant code example

```
import PropTypes from 'prop-types';

function Hello({ firstname, lastname }) {
  return <div>Hello {firstname} {lastname}</div>; // Noncompliant: 'lastname' type is missing
}
Hello.propTypes = {
  firstname: PropTypes.string.isRequired
};

// Using legacy APIs

class Hello extends React.Component {
  render() {
    return <div>Hello {this.props.firstname} {this.props.lastname}</div>; // Noncompliant: 'lastname' type is missing
  }
}
Hello.propTypes = {
  firstname: PropTypes.string.isRequired,
};
```

### Compliant solution

```
import PropTypes from 'prop-types';

function Hello({ firstname, lastname }) {
  return <div>Hello {firstname} {lastname}</div>;
}
Hello.propTypes = {
  firstname: PropTypes.string.isRequired,
  lastname: PropTypes.string.isRequired,
};

// Using legacy APIs

class Hello extends React.Component {
  render() {
    return <div>Hello {this.props.firstname} {this.props.lastname}</div>;
  }
}
Hello.propTypes = {
  firstname: PropTypes.string.isRequired,
  lastname: PropTypes.string.isRequired,
};
```