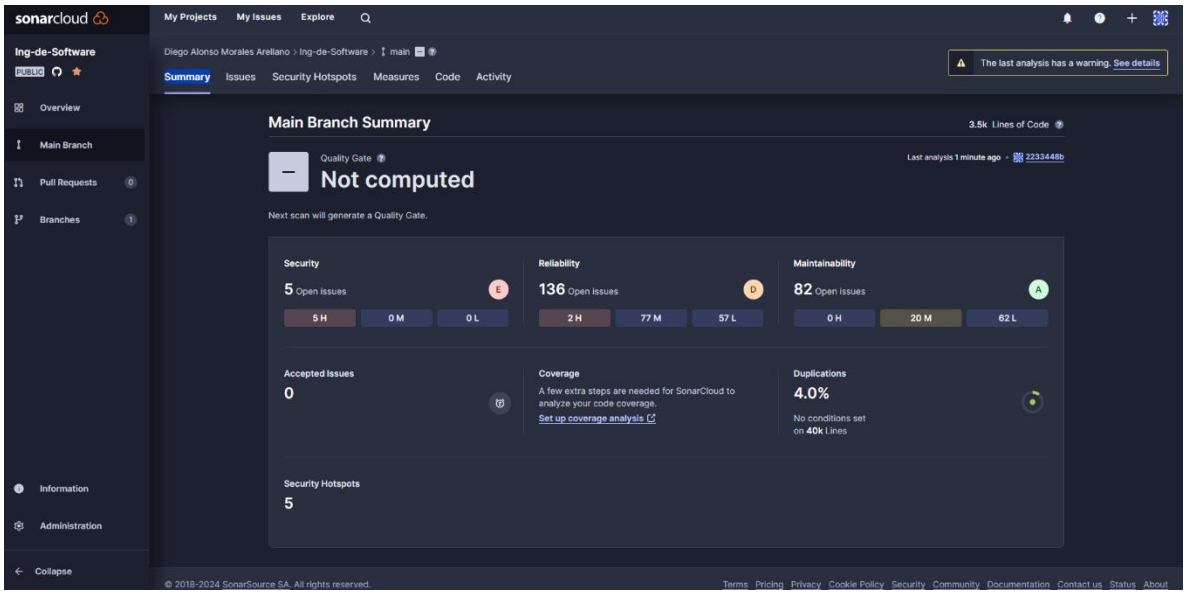
Resultados de la inspección del software

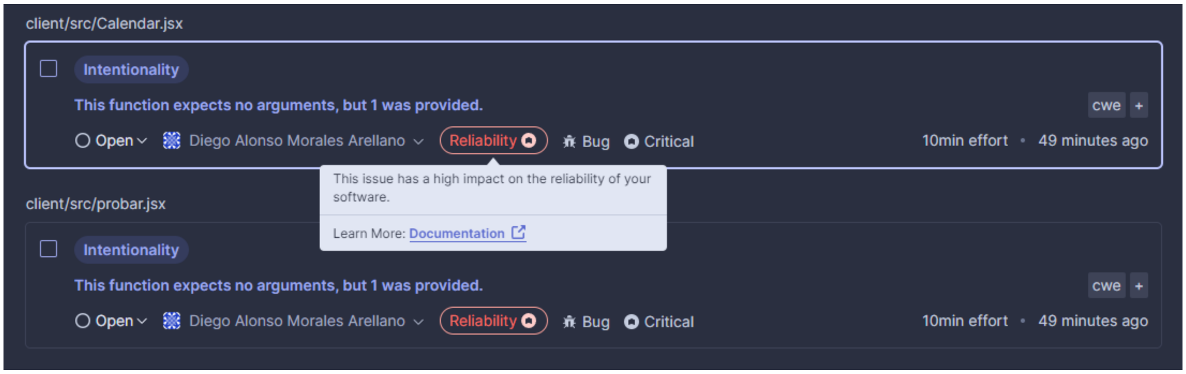
Una vez iniciada la sesión en Sonarcloud e iniciada la evaluación del proyecto, tendremos los siguientes resultados a nivel general y los problemas con su severidad.

Resultados generales:



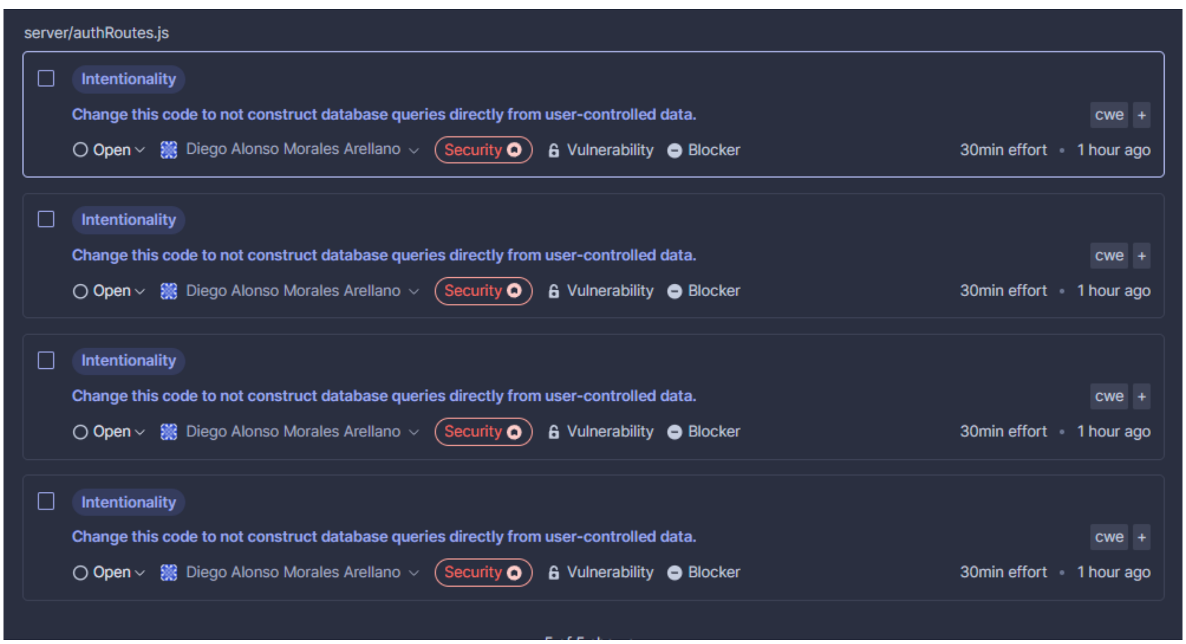
Cantidad de los problemas según su nivel de severidad:

Primer problema seleccionado según su calidad e impacto en el software:



Los problemas seleccionados en la imagen anterior contaran solo como un problema, ya que Calendar.jsx y probar.jsx son una entidad en conjunto donde se hacían pruebas respectivas para Calendar.jsx en un apartado distinto en código, por lo tanto, solucionarlos solo contaría como un problema seleccionado. Ahora bien, ambos seleccionados afectan a la fiabilidad del código, con un impacto alto a nivel del software, por lo tanto, su severidad es grave y hay que tomarlo en consideración.

Segundo problema seleccionado según su calidad e impacto en el software:



Los 4 problemas que se pueden apreciar en la imagen anterior aluden a los insercion de los datos para el registro de los usuarios, ya que estos datos para verificar si existe un usario anteror con dichos, en otras palabras no hay forma de asegurar que la clave primaria este brien escrita por el usuario que quiere registrarse, por lo tanto lo trabajaremos como un solo problema para abordar, ya que estos problemas poseen un imacto elevando en el software a nivel de seguridad (severidad alta

Recomendaciones por Sonarcloud hacia el primer problema

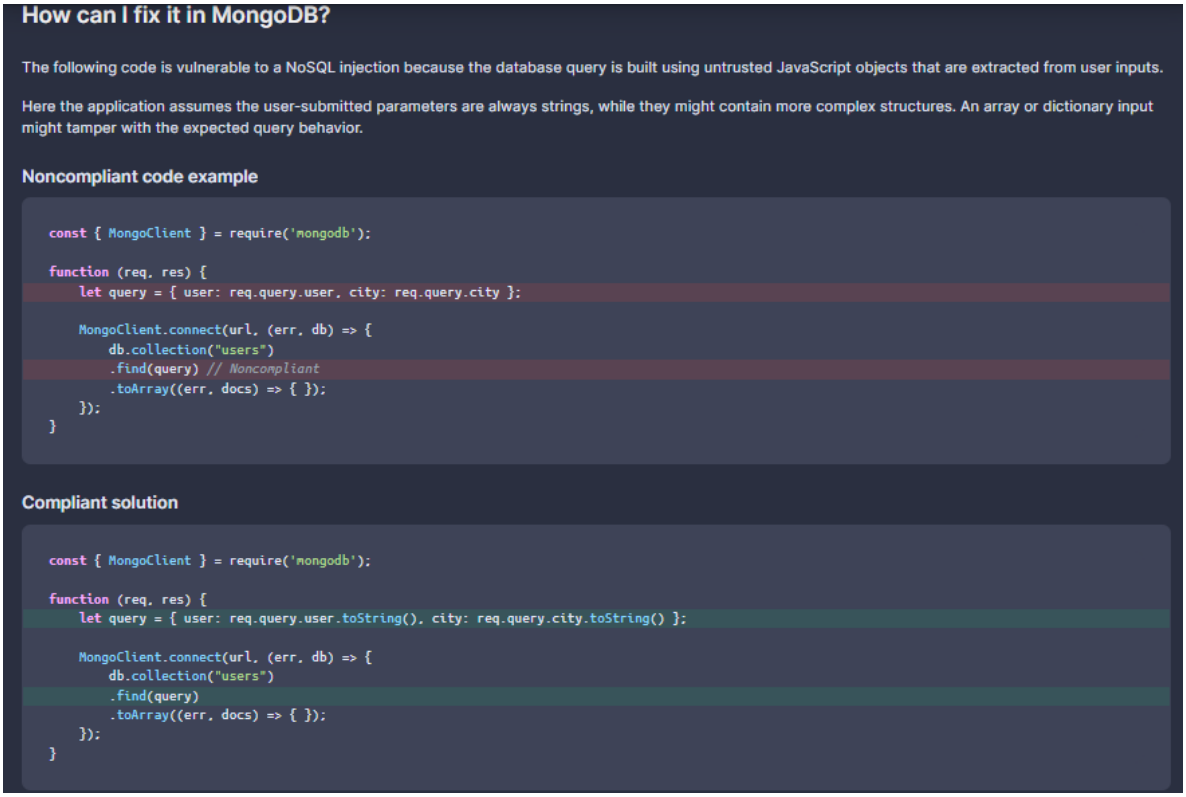
Dentro de las soluciones propuestas por Sonarcloud simplemente seria descartar los argumentos que no se necesitan dentro de la función para Calendar.jsx, abordando este problema más específico al momento de editar los eventos de las horas médicas.

```
const handleEditar = async () => {
  console.log("VOY A ELIMINAR", editEventId);
  try {
    const isEditEvent = selectedEdit || selectedEdit.id;

    if (!isEditEvent) {
      console.error("No se ha seleccionado un evento válido para editar");
      return;
    }

    // Obtener el RUT del personal administrativo en este punto
    const rutPA = rut;

    //Eliminar el evento existente
    await fetch(`http://localhost:5000/api/deleteEvent/${editEventId}`, {
      method: 'DELETE',
    });
```

Como podremos ver en el código de handleEditar no recibe parámetros, y nuestro error era pasarle como parámetro el ID del evento que queríamos editar, cuando esta función ya lo llama a nivel general, por lo tanto, no es necesario pasarlo como parámetro, con tan solo eliminando ese llamado extra para solucionar el problema significara en una mejora para la fiabilidad de l software.

Recomendaciones por Sonarcloud hacia el segundo problema

La solucion propuesta por Sonarcloud era asegurar que el tipo de dato sea de tipo “String” ya que en nuestra estructura de los datos que se tienen que rellenar en el formulario del registro son de aquel tipo, sin embargo, esto no aseguraba del todo de que la persona no pueda equivocarse, a tal punto de rellenar los datos de los campos necesarios con cualquier informacion. Sonarcloud menciona de que como es un campo que lo gestiona el usuario, puede rellenar con contenido malicioso para la base de datos, impactando la seguridad de este ultimo.



Como podremos ver, dentro de la solucion propuesta por Sonarcloud, agrega la funcion toString() para sercionar de que el parametro que se quiere comprobar sea de aquel tipo tal como la estructura que definimos para los datos de los usuarios, por lo tanto, al agregar dicha funcion a cada parametro que se quiera comprobar en la existencia de algun usuario previo, nos aseguraremos que cumpla con dicha estructura, sin tener que limitar al usuario en el llenado del formulario para no cambiar tanto el codigo y mejorando la eficiencia.