Universidad Autónoma de Baja California Facultad de Ciencias Químicas e Ingeniería

Plan de Ingeniero en Software y Tecnologías Emergentes



Materia

Verificación y Validación del Software (361)

Meta 2.1.3

Diseña una prueba estática para verificación

Docente

Claudia Gabriel Tona Castro

Participante(es)

Luis Eduardo Galindo Amaya (1274895)

Meta 2.1.3

Diseña una prueba estática para verificación

Introducción

Las pruebas estáticas son un componente esencial del proceso de aseguramiento de la calidad en el desarrollo de software. Las pruebas estáticas se realizan sin necesidad de ejecutar el código. Por lo general se centran en la revisión del código fuente, el diseño y la documentación relacionada con el software.

Desarrollo

para esta practica decidí usar sonarlint (SonarQube), para esto instalé el plugin de visual estudio code en la sección de plugins:

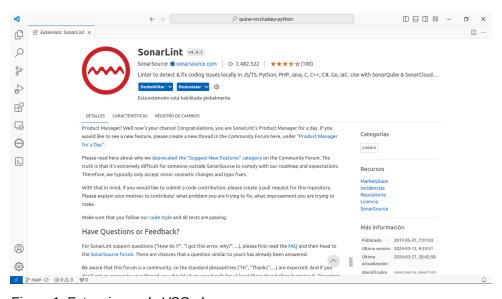


Figura 1: Extenciones de VSCode

Después elegí uno de mis proyectos anteriores, decidí usar un programa para reducir expresiones lógicas al mínimo utilizando el algoritmo de Algoritmo Quine–McCluskey¹

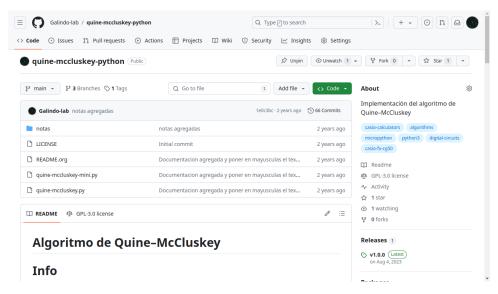


Figura 2: Repositorio del proyecto

A continuación abrí el proyecto en VSCode e inmediatamente Sonarlint empezó a mostrar posibles problemas con el código:

```
Ð
                       def captura char(message="", end="?\n") -> str:
Q 215
                              while(True):
print("DATO INVALIDO")
223
224
                         return foo[0]
def en_rango(numero_variables:int, activaciones:int, redundancias:list):
    lim = (2 << (numero_variables - 1))-1</pre>
8
                               for i in range(len(activaciones)):
    if activaciones[i] > lim:
        print(activaciones[i], "FUERA DE RANGO")
        activaciones[i] = None
0
235
236
                        for i in range(len(redundancias)):
   if redundancias[i] > lim:
             PROBLEMAS (10) SALIDA CONSOLA DE DEPURACIÓN PUERTOS

    quine-mcduskey.py (10)
    if kt his "_getitem_" operation; Previous type checks suggest that "activaciones" does not have this method. sonarlint(python:55864) [Lin. 231, col. 12]
    A Fix his "_getitem_" operation; Previous type checks suggest that "activaciones" does not have this method. sonarlint(python:55864) [Lin. 232, col. 13]
    A Fix his "_setitem_" operation; Previous type checks suggest that "activaciones" does not have this method. sonarlint(python:55864) [Lin. 233, col. 13]

                  A Refactor this function to reduce its Cognitive Complexity from 18 to the 13 allowed. F7 locations] sonariint(python:53776) [Lin. 260, col. 5]

Use the opposite operator ("not in") instead, sonariint(python:51940) [Lin. 295, col. 20]

A Remove this commented out cost consulint(python:5129 [Lin. 13, col. 5]

Use the opposite operator ("not in") instead, sonariint(python:51940) [Lin. 334, col. 16]

▲ Use the opposite operator ("not in") instead. sonarlint(python:S1940) [Lin. 352, col. 29]

                   A Remove this commented out code. sonarlint(python:S125) [Lín. 362, col. 5]
                    △ Define a constant instead of duplicating this literal "---+----" 3 times. [+2 locations] sonarlint(python:S1192) [Lin. 391, col. 11]

$\mathcal{p}$ main ⊕ ⊗ 0 \( \Delta \) 10 \( \M \) 0 SonarLint focus: overall code
                                                                                                                                                                                                                              Lín. 231, col. 34 Espacios: 4 UTF-8 LF Python Q
```

Figura 3: Cosas detectadas por SonarLint

^{1 &}lt;a href="https://github.com/Galindo-lab/quine-mccluskey-python">https://github.com/Galindo-lab/quine-mccluskey-python

Si hacemos clic en las notas que hace sonar lint podemos obtener una descripción detallada sobre el error que esta marcando:

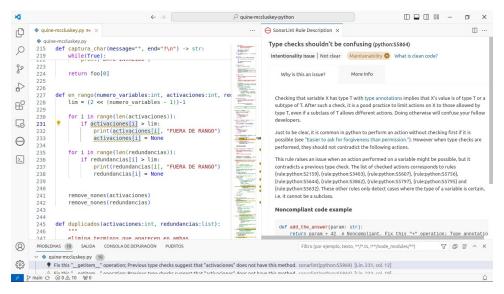


Figura 4: Explicacion detallada del problema identificado

Sonarlint ofrece algunas opciones para refactorizar el código, pero son por lo general problemas sencillos como cambiar operadores o invertir ifs:

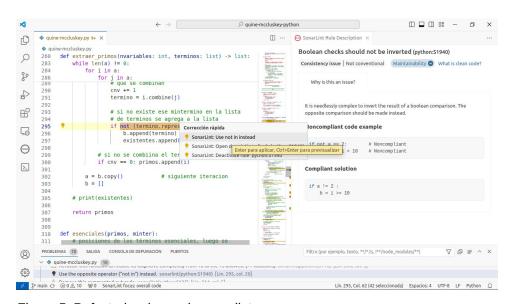


Figura 5: Refactorizaciones de sonarlint

Algunos de los errores que aparecen en sonarlint no son muy graves, la mayoría son simplificaciones de expresiones lógicas o renombrado de variables.



Figura 6: Lista de errores en todo el código

Conslusion

A lo largo de esta meta aprendí como hacer pruebas estáticas de manera sencilla y en que situaciones pueden ser útiles, las herramientas modernas para análisis estático son tan avanzadas que pueden integrarse fácilmente en cualquier desarrollo, permitiendo que todos los programadores puedan utilizar buenas técnicas de programación durante el desarrollo.