Práctica 2: Análisis Estático

El objetivo de esta práctica es profundizar en los conceptos explicados en las clases de teoría. De una manera más precisa, utilizaremos los conceptos explicados en el tema de análisis estático, (Tema 4). La práctica está formada por dos partes, en la primera, utilizaremos el juez Domjudge para probar expresiones regulares, mientras que en la segunda montaremos un entorno de integración continua en GitHub.
Ante cualquier duda durante la resolución de la práctica, escribir un email a isaac.lozano@urjc.es **con copia a** raul.martin@urjc.es. En caso de no poderse resolver la duda vía mail, se puede concertar una tutoría, siempre y cuando se concierte en un período de **hasta 48 horas antes** de la fecha de entrega de la práctica.

Normas y evaluación

- La realización de la práctica se podrá realizar de forma individual o por parejas. **Aunque se realice por parejas, es necesario que ambos miembros del equipo suban el código al juez.**
- En la memoria deberá describirse los pasos que han realizado y explicar el procedimiento utilizado en cada uno de los ejercicios. Incluye capturas en todas aquellas partes que ayuden a la comprensión o aporten información interesante.
- Es suficiente con que un miembro del equipo entregue la memoria en el aula virtual. La memoria será entregada (en formato `pdf`). Es imprescindible que **todos** los integrantes del grupo estén correctamente identificados en la entrega.
- La extensión máxima de la memoria será de 10 páginas, incluyendo portada.
- El repositorio de GitHub necesario dónde se lleve a cabo la práctica debe ser público y su dirección incluida en la memoria.
- La fecha límite para entregar la práctica será el **16 de abril a las 23:55**.
- Esta práctica se corresponde con un 15% de la nota final.
- Aviso: todos los ejercicios han sido probados con ChatGPT, y hay casos fácilmente detectables para los que falla.

1 Práctica 1: Análisis estático

1.1 Expresiones regulares [4 puntos]

1.1.1 Objetivo

El objetivo de este apartado es que el alumno se familiarice con las expresiones regulares y sea capaz de crear expresiones regulares para buscar y validar determinados tipos de datos, en los diferentes ejercicios planteados.

1.1.2 Forma de trabajo

Para la realización de la práctica, cada persona dispondrá de un usuario en el juez https://mds-gcib.numa.host/. Existirá un concurso en el que podrán evaluar su expresión regular. En la memoria final de la práctica se tendrá que explicar la expresión regular utilizada.
El texto a utilizar deberá leerse de la entrada estándar (stdin / System.in) y los resultados deberán escribirse a la salida estándar (stdout / System.out). El lenguaje de programación a utilizar queda a elección del alumno entre los siguientes: Python 3, Java, C, C++.

1.1.3 Ejercicio 1 [0.6 puntos]

Dato un texto de una sola línea, determinar todos los años (números formados estrictamente por 4 dígitos) que aparecen. Un año es una cadena numérica de longitud 4, con cualquier valor en el rango $[0000, 9999]$. Se tendrán que imprimir por pantalla en el lenguaje deseado usando una expresión regular todos los años que vienen en el texto en orden de aparición, en una línea cada uno.

1.1.4 Ejercicio 2 [0.6 puntos]

Dato un texto de una línea, determinar todas las matrículas que aparecen. Una matrícula es una cadena que tiene las siguientes características:

- 4 dígitos seguidos de un separador (guion, espacio o nada) y 3 letras en mayúsculas al final tal que $[0000 - AAA, ..., 9999 - ZZZ]$.
- Las matrículas pueden llevar una E mayúscula delante para indicar que se trata de un vehículo especial. Ejemplos: E1337ZZZ, E-0000 PCB.
- Los dígitos pueden estar separadas de las letras utilizando un guion, un espacio, o ningún separador.

Se tendrán que imprimir por pantalla usando el lenguaje elegido usando una expresión regular todas las matrículas que vienen en el texto en orden de aparición.

1.1.5 Ejercicio 3 [0.6 puntos]

Dado un formato de fechas yyyy-mm-dd, se pide convertir a dd.mm.yyyy.
Para cada match encontrado en los documentos propuestos se tendrá que imprimir en el siguiente formato (los rangos de fecha puede ser erróneos, pueden existir un mes 20).

- Para el caso “El profesor Isaac Lozano puso una fecha de entrega el 2023-04-16 a las 23:55”. Se imprimirá “El profesor Isaac Lozano puso una fecha de entrega el 16.04.2023 a las 23:55”.

1.1.6 Ejercicio 4 [0.6 puntos]

Dado un texto, determinar cuando se ha encontrado un email de alumno de nuestra universidad “@alumnos.urjc.es” o profesor “@urjc.es”. Los emails de alumnos están formados del siguiente patrón (puedes asumir que el input siempre estará en minúscula):

- Inicial del usuario, seguido de punto.
- Apellido del usuario, siempre mayor o igual a 2 caracteres.
- Seguidos de un punto y la fecha de matriculación.
- todos finalizan con “@alumnos.urjc.es”.

Los correos de los profesores constan de:

- Nombre del profesor seguido de un punto.
- Apellido del profesor.
- Finalizando con “@urjc.es”.

Para cada match encontrado se tendrá que imprimir en el siguiente formato.

- Para el caso de prueba i.lozanoo.2015@alumnos.urjc.es reportaremos “alumno lozanoo matriculado en 2015”
- Para un profesor reportaremos para el ejemplo isaac.lozano@urjc.es “profesor isaac apellido lozano”

1.1.7 Ejercicio 5 [0.6 puntos]

Dado un texto devolver las direcciones postales.
Una dirección estará compuesta de una calle representada por “C/” o ”Calle” seguido de un espacio con el nombre de la calle (una sola palabra) donde la primera letra debe estar en mayúscula, opcionalmente una coma, un número arbitrario de espacios, el número en cualquiera de los siguientes formatos (`Nº7, nº7, N 7, n 7, 7, Nº 7, nº 7, n7, N7`). No es válido N º7, n º7, ni º7, la N podría estar en mayúsculas o minúsculas. Seguido, una coma, un número arbitrario de espacios y un número de 5 dígitos correspondiente a un código postal. Los nombres de las calles deben poder validar calles de Madrid formadas por una sola palabra, no es necesario que reconozca “Calle Almendro Azul”, porque el nombre de la calle tiene dos palabras en vez de una.
Ejemplos de casos:

- “C/ Dulcinea Nº 10, 28936”
- “Calle Dulcinea 10, 28106”
- “Calle Dulcinea N10, 28091”

Para cada calle encontrada se reportará: “CP-Calle-Numero”, por ejemplo: “28926-Dulcinea-10”.

> Pista: Si te da error, busca las calles más bonitas de Madrid y revisa como se escriben.

1.1.8 Ejercicio 6 [1.0 puntos]

Dado un fichero de logs como el siguiente ejemplo:

```
2023-02-07 01:14:28.313 INFO 1174086 --- [main] drfp.Main : Starting Main v0.1-
    ↪ SNAPSHOT using Java 17.0.1 on raul2-ubuntu with PID 1174086 started by
    ↪ rmartin
2023-02-07 01:14:28.317 INFO 1174086 --- [main] drfp.Main : No active profile set,
    ↪ falling back to default profiles: default
2023-02-07 01:14:29.428 INFO 1174086 --- [main] a.p.q.PostPCheck : Bean '
    ↪ eventAsyncConfigurer' of type [es.urjc.etsii.grafo.solver.services.events.
    ↪ EventAsyncConfigurer] is not eligible for getting processed by all
    ↪ BeanPostProcessors (for example: not eligible for auto-proxying)
2023-02-07 01:14:29.806 INFO 1174086 --- [main] TomcatWebServer : Tomcat
    ↪ initialized with port(s): 8080 (http)
2023-02-07 01:14:29.818 INFO 1174086 --- [main] o.a.c.c.StdSvc : Starting service [
    ↪ Tomcat]
```

Transformar cada línea a CSV extrayendo la siguiente información:

- Nivel de log
- Hilo dónde se ha producido el log (este hilo se corresponde con letras en mayúscula, minúscula y de números)
- Clase responsable de emitir el log
- Mensaje de log

Ejemplo para la línea:

```
2023-02-07 01:14:28.313 INFO 1174086 --- [main] drfp.Main : Starting Main v0.1-
    ↪ SNAPSHOT using Java 17.0.1 on raul2-ubuntu with PID 1174086 started by
    ↪ rmartin
```

Se debe imprimir:

```
"INFO","main","Main","Starting Main v0.1-SNAPSHOT using Java 17.0.1 on raul2-ubuntu
    ↪  with PID 1174086 started by rmartin"
```

> Nota: Las comillas que se muestran en los ejemplos, corresponden a las comillas simples, si copiáis y pegáis desde el documento PDF pueden poner unas comillas diferentes.

1.2 Integración Continua [6 puntos]

1.2.1 Objetivo

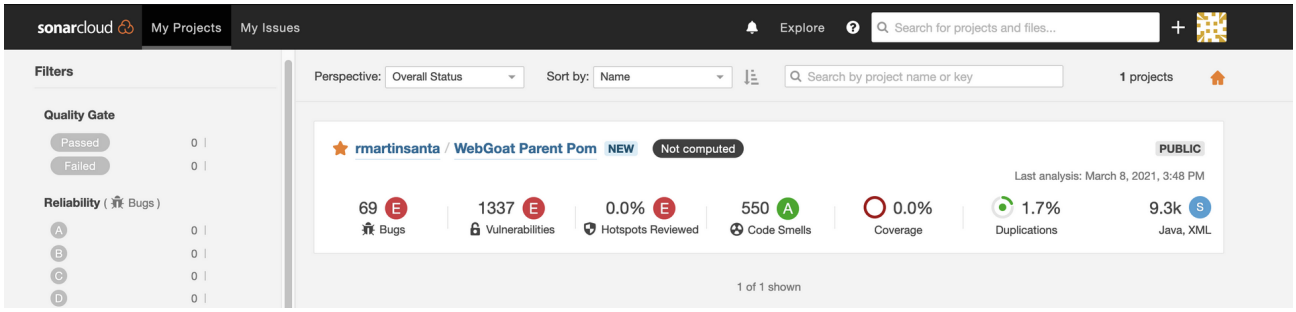
El objetivo de esta práctica es que el alumno se familiarice con la herramienta de análisis de código estático de código abierto SonarQube, la cual utilizaremos para analizar el código de un proyecto intencionalmente vulnerable.

1.2.2 Parte 1: Puesta en marcha del proceso de análisis automático. [3 puntos]

Para la realización de la práctica será necesario disponer de una cuenta de GitHub, recordad que podéis disfrutar de bastantes beneficios si os dais de alta como estudiantes: https://education.github.com/students
El proceso resumido es el siguiente:

1. Hacer fork al siguiente repositorio: https://github.com/rmartinsanta/WebGoat
2. Acceder a https://sonarcloud.io/ e iniciar sesión con GitHub.
3. Autorizar a SonarCloud a acceder al menos al repositorio WebGoat.
4. Seguir el proceso de configuración del repositorio WebGoat utilizando GitHub Actions:
5. Modificar las propiedades en el pom.xml, añadiendo las que da SonarCloud y las siguientes propiedades también:

```
<sonar.moduleKey>${project.groupId}:${project.artifactId}</sonar.moduleKey>
<skipTests>true</skipTests>
```



6. Eliminar todos los ficheros en .github/workflows. Crear el fichero de configuración de GitHub Actions para configurar la integración contínua (justo debajo de la configuración te viene el fichero el cual debes añadir en el path que te pone). En el archivo de configuración del workflow, se debe modificar java-version de la 11 a la 17.

Revisa la lista de commits y verifica que se está ejecutando el proceso de integración continua (punto naranja). Una vez haya finalizado el análisis, deberíais ver algo similar a la siguiente captura en la portada de SonarCloud.

1.2.3 Preguntas

1. ¿Cuántas vulnerabilidades, bugs y code smells ha detectado SonarCloud en el proyecto entero?

2. Haz click sobre el proyecto para abrir la vista de detalle. Revisa las vulnerabilidades detectadas y la lista de Security Hotspots. ¿Qué diferencia crees que existe entre las vulnerabilidades y los Security Hotspots?

3. Haz cualquier commit para forzar un reanálisis (por ejemplo editando el README.md). Espera a que finalice y vuelve a Sonar. ¿Cuál es el estado del proyecto (Passed/Failed)?. ¿A qué crees que se debe? ¿Crees que el número de vulnerabilidades afecta a dicho veredicto?

1.2.4 Parte 2: Mitigación [3 puntos]

Para la realización de esta parte, nos ponemos en la piel de un becario desarrollador, al cual le han asignado como primera tarea resolver una vulnerabilidad en el proyecto.

1.2.5 Preguntas

1. Elige una vulnerabilidad de tipo *Blocker* o *Critical*, explica cuál es la vulnerabilidad detectada, por qué ha sido detectada (y si realmente es una vulnerabilidad y no un falso positivo).

2. Propón una solución e impleméntela en una nueva rama del repositorio. Explica los cambios que arreglan dicha vulnerabilidad.

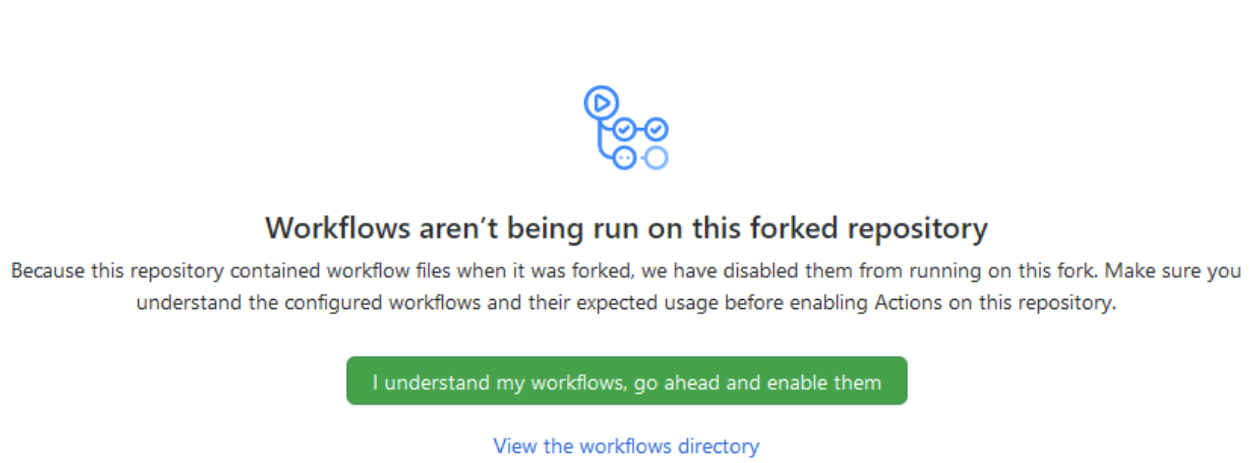
3. Crea una *Pull Request* de la nueva rama a la rama principal. ¿Qué opina Sonar de los cambios realizados?

4. Junta (*merge*) la nueva rama con la rama principal. Una vez finalizado el análisis, ¿qué cambios se han producido en el proyecto? ¿Cuántas vulnerabilidades detecta ahora?

1.3 Dudas generales resueltas en las sesiones presenciales

1.3.1 No me funcionan las acciones.

Una vez haces el fork de un repositorio, puede ser que Github no te otorgue permisos para la ejecución de las mismas. En la pestaña de acciones de Github puedes tener un mensaje como el siguiente que tendrás que aceptar.



1.3.2 No me ejecuta el análisis Sonar

Una vez seguidos los pasos de la memoria, debes comprobar los siguientes paneles.

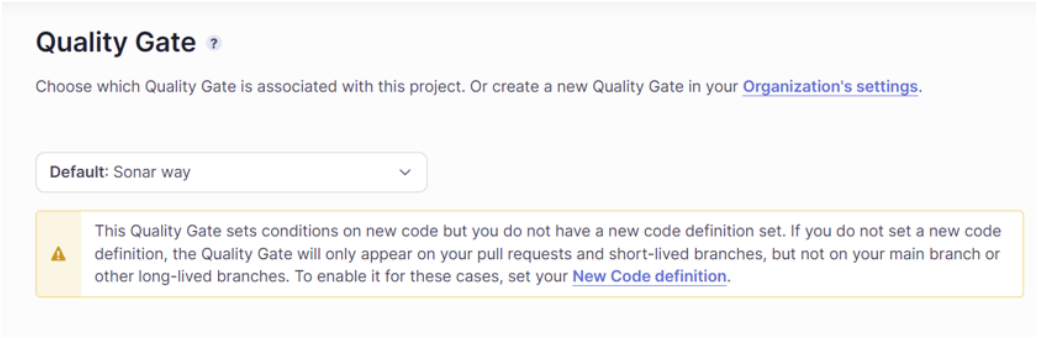


Fig. 1: En `https://sonarcloud.io/project/quality_gate?id=PROYECTO`

En este primero se debe seleccionar Sonar way como por defecto.

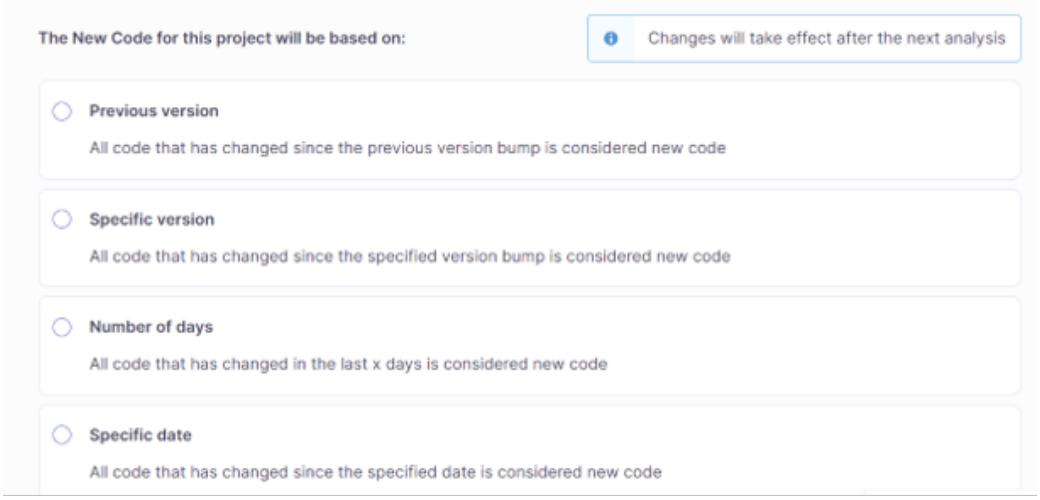


Fig. 2: En `https://sonarcloud.io/project/new_code?id=PROYECTO`

En este segundo se debe seleccionar Previous version como por defecto.

1.3.3 Acabo de hacerlo pero no sale nada

Necesitas hacer otro cambio y sonar analizará a partir de este cambio.

1.3.4 ¿Cómo puede ver mi compañero el análisis?

Con que un usuario tenga el fork es suficiente, ese proyecto se debe compartir mediante invitación con colaboración con nuestro compañero. Se configurará ese proyecto en Sonar, la URL del proyecto es pública y ambos pueden observar los datos de los análisis realizando cambios.

1.3.5 Para los usuarios que hicieron fork en la primera sesión de prácticas.

Recordad que se hizo un cambio en la configuración del pom.xml, es necesario eliminar una serie de datos para el correcto funcionamiento del proyecto. Revisad el cambio y realizarlo.