

Análisis Estático de Código Fuente

El análisis estático de software es un tipo de análisis de software que se realiza sin ejecutar el programa.

Algunos beneficios de este tipo de análisis son:

- Ayuda a la detección de bugs, vulnerabilidades, code smells.
- Mejorar la capacidad de mantenimiento del código.
- Posibilidad de identificar errores antes de realizar el despliegue a producción.

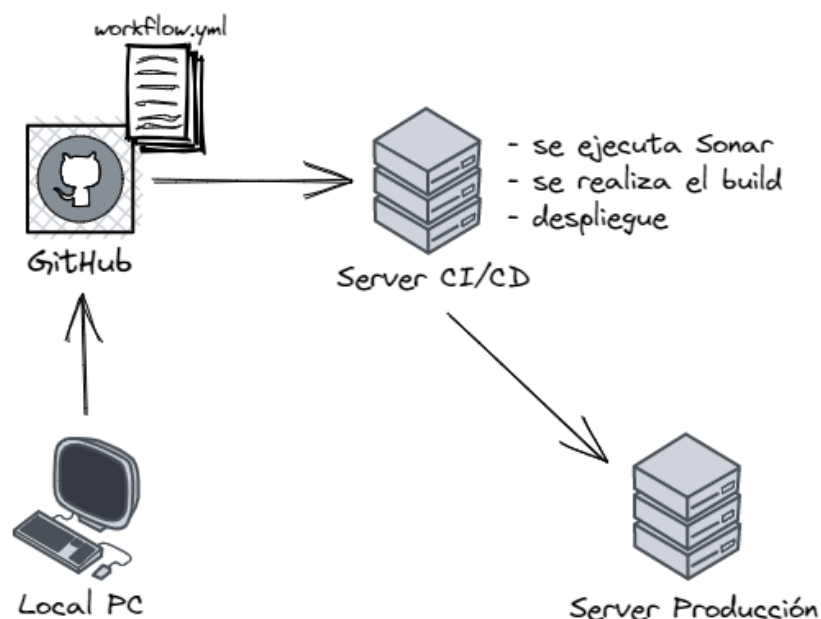
SonarQube

SonarQube es una herramienta para realizar análisis estático de código fuente. La misma nos sirve para obtener métricas que pueden ayudar a mejorar la calidad del código de un programa. Permite a los equipos de desarrollo hacer seguimiento y detectar errores y vulnerabilidades de seguridad para mantener el código limpio.

SonarQube Incluye soporte para más de 20 lenguajes de programación, y detecta principalmente tres tipos de problemas:

- Bugs: error de codificación que rompe el código. debe corregirse de inmediato para poder ejecutar (esto ya debería ser detectado en el IDE, y no llegar hasta aquí).
- Vulnerabilidad: puntos de código abiertos a ataques o hacking.
- CodeSmells: fragmentos de código confuso que lo hacen difícil de mantener.

Un punto clave de esta herramientas es su integración con entornos de CI/CD (Jenkins, AWS, Github Actions, etc). Por ejemplo, si estamos usando un pipeline en Jenkins para construir, inspeccionar y desplegar el código, el paso de inspección continua, sería el que serviría de filtro para, si pasa el Quality Gate (o Umbrales de Calidad), desplegar, o, por el contrario, si no lo pasa, notificar del error y no desplegar hasta que se corrija, se suba de nuevo al repositorio de código, y empiece de nuevo el ciclo de despliegue continuo.



Quality Gate predeterminado de SonarQube

Coverage: Indica el porcentaje de código que está cubierto por pruebas unitarias. SonarQube establece en el Quality Gate predeterminado que la cobertura debe ser superior al 80%

Duplications: indica la cantidad de líneas que se repiten, en comparación con el total de líneas del proyecto. En este caso, se establece que este valor sea inferior al 3%.

Bugs: esta calificación se obtiene en relación a la severidad más alta de los bugs detectados. Para esta métrica, se determina que la calificación debe ser A, lo que significa que no haya ningún bug.

Vulnerabilities: se refiere también a la severidad más alta, pero de las vulnerabilidades encontradas. También debe ser A, lo que indica que no existiría ninguna vulnerabilidad en el código nuevo.

Hotspots Reviewed: son los fragmentos de código sensible a la seguridad que el desarrollador debe revisar. La principal diferencia entre un hotspot y una vulnerabilidad es la necesidad de una revisión antes de decidir si aplicar una corrección.

Code Smells: se mide el porcentaje de deuda técnica que existe en el código. La calificación debe ser A, que vendría a ser que el porcentaje de deuda técnica es menor o igual al 5%.

Ejemplos de utilización de SonarQube

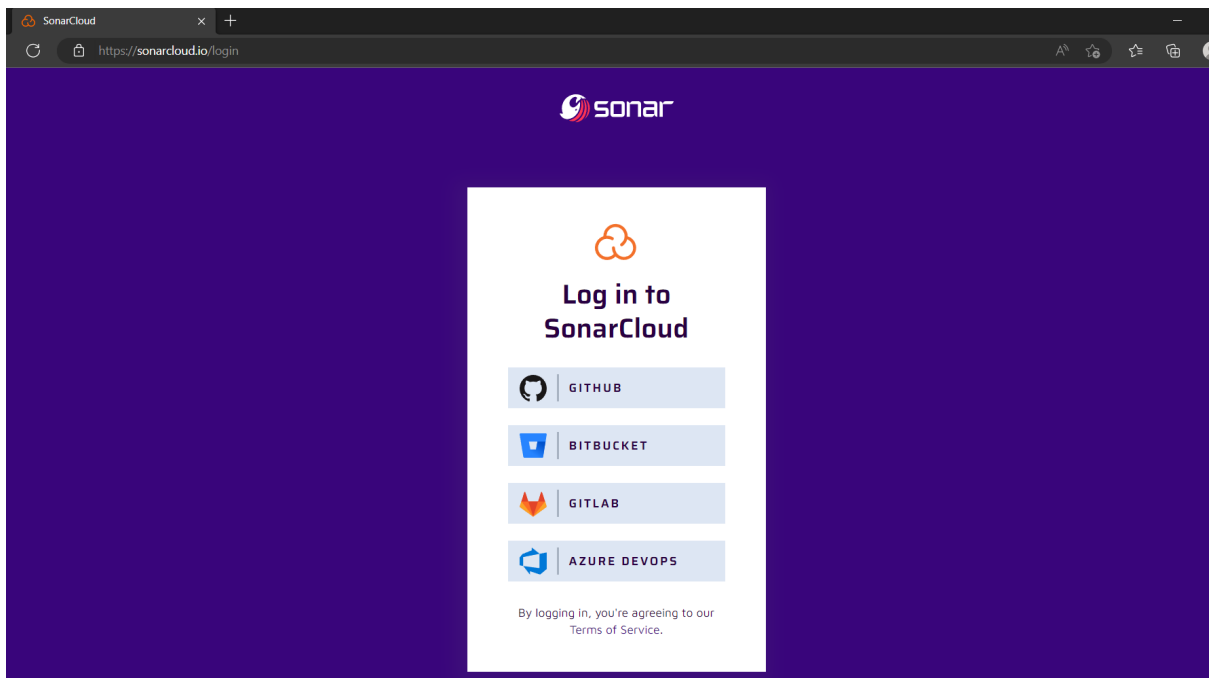
Para explicar y ejemplificar el funcionamiento de SonarQube analizaremos un desarrollo propio con archivos de ejemplo (test-project) con el objeto de ver, paso a paso, desde su instalación hasta un análisis general de sus funcionalidades principales.

Dos formas para utilizar SonarQube son:

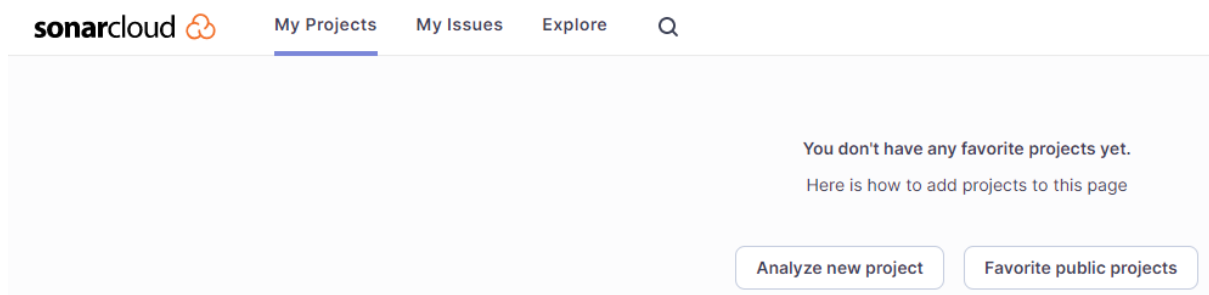
1. Desde SonarCloud (en este caso el código analizado es compartido en su plataforma a menos que se pague la versión premium).
2. Instalar SonarQube en un servidor propio.

SonarCloud online.

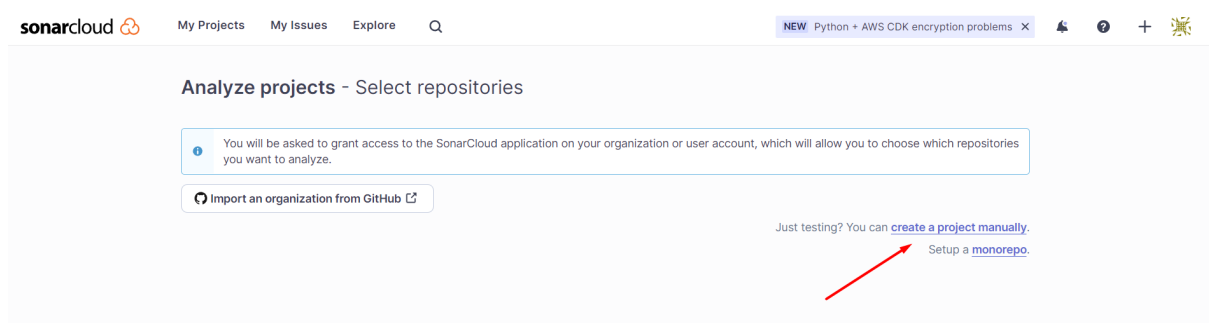
Dirigirse al sitio de SonarCloud (<https://sonarcloud.io>) y loguearse.



Dirigirse a la seccionar "My Projects" seleccionar "Analyze new project":



Seleccionar "Crear proyecto manualmente":



Cargamos los datos del formulario:

Analyze projects - Set up manually

Manual setup is not recommended, and leads to missing features like appropriate setup of your project or analysis feedback in the Pull Request.

Organization *
Jesus Andres Jesusandres31 [Create another organization](#)

Project key *
mti-pds [?](#)
Up to 400 characters. All letters, digits, dash, underscore, period or colon.

Display name *
mti-pds [?](#)
Up to 255 characters

Project visibility

☒ Public
Anyone will be able to browse your source code and see the result of your analysis.

☐ Private
Only members of the organization will be able to browse your source code and see the result of your analysis.

[Set Up](#)

Analyze private projects with our Paid Plan from €10

- ✓ Unlimited private projects
- ✓ Strict control over who can view your private data
- ✓ No commitments, cancel anytime
- ✓ 14 days free trial.

[Learn more](#)

[Upgrade](#)

Como puede verse en Project visibility está inhabilitada la opción Private, con lo cual el código subido para analizar estará disponible para cualquiera que quiera ver los resultados.

Luego de pulsar el botón Set Up el formulario es enviado y la próxima pantalla que se muestra es la de configuración del proyecto dado de alta:

mti-pds
PUBLIC

Configure

Choose your Analysis Method

With GitHub Actions [>](#)
Recommended

With Travis CI [>](#)

With other CI tools [>](#)
SonarCloud integrates with your workflow no matter which CI tool you're using.

Manually [>](#)
Use this for testing. Other modes are recommended to help you set up your CI environment.

Como se mencionó antes SonarQube es muy utilizado en entornos de integración continua. Pero en este caso de prueba vamos a analizar el código manualmente.

Seleccionamos la opción más acorde a nuestro lenguaje de programación y seleccionamos el sistema operativo donde se ejecutará SonarScanner.

sonarcloud

mti-pds
PUBLIC ★

Configure

Main Branch

Pull Requests 0

Branches 0

Information

Administration

← Collapse

My Projects My Issues Explore Q

Jesus Andres > mti-pds > Configure

Analyze from your local sources

Run analysis on your project

What option best describes your build?

Maven Gradle .NET C, C++ or ObjC Other (for JS, TS, Go, Python, PHP, ...)

Which OS do you run your build on?

Linux Windows macOS

Download and unzip the SonarScanner for Windows

And add the `bin` directory to the `%PATH%` environment variable

[Download](#)

Configure the **SONAR_TOKEN** environment variable

- 1 Name of the environment variable: `SONAR_TOKEN`
- 2 Value of the environment variable: `5828e6a41e7e4e0f17d8d972fd290bb9adce4ddd`

Execute the SonarScanner from your computer

Run the following command in your project's folder.

Descargamos SonarScanner y lo agregamos como variable de entorno.

My Projects My Issues Explore Q NEW Python

Jesus Andres > test-tmi > Configure

Maven Gradle .NET C, C++ or ObjC Other (for JS, TS, Go, Python, PHP, ...)

Which OS do you run your build on?

Linux Windows macOS

Download and unzip the SonarScanner for Windows

And add the `bin` directory to the `%PATH%` environment variable

[Download](#)

Con los datos del SONAR_TOKEN y del comando autogenerado creamos un archivo llamado sonar-project.properties en la raíz de nuestro proyecto y copiamos esos datos.

1

Name of the environment variable: SONAR_TOKEN

2

Value of the environment variable: 53755eeca3216d2b80e696c95ff26b6988cc5a0d

Execute the SonarScanner from your computer

Run the following command in your project's folder.

```
sonar-scanner.bat \
-D"sonar.organization=oscarfalcone" \
-D"sonar.projectKey=mti-test" \
-D"sonar.sources=." \
-D"sonar.host.url=https://sonarcloud.io"
```

Copy

Please visit the [SonarScanner documentation](#) for more details.



And you are done!

You should see the page refresh itself in a few moments with your analysis results if everything runs successfully.

```
sonar-project.properties X
sonar-project.properties
1 sonar.organization=jesusandres31
2 sonar.projectKey=sonar-tp-mti-pds
3 sonar.sources=src/ # nombre de la carpeta donde está el código fuente a analizar
4 sonar.host.url=https://sonarcloud.io
5 sonar.login=b4b7a815b0a42b33c974b60486e9d96344c0f423 # SONAR_TOKEN
6
```

Ejecutamos sonar-scanner en la raíz de nuestro proyecto y esperamos a que termine de analizar.

```
TERMINAL
PS C:\Users\jesus\projects\MTI\test-project>
C:\Users\jesus\projects\MTI\test-project>
PS C:\Users\jesus\projects\MTI\test-project> sonar-scanner
INFO: Scanner configuration file: C:\Users\jesus\Downloads\sonar-scanner-4.7.0.2747-windows\bin\..conf\sonar-scanner.properties
INFO: Project root configuration file: C:\Users\jesus\projects\MTI\test-project\sonar-project.properties
INFO: SonarScanner 4.7.0.2747
INFO: Java 11.0.14.1 Eclipse Adoptium (64-bit)
INFO: Windows 11 10.0 amd64
INFO: User cache: C:\Users\jesus\sonar\cache
INFO: Scanner configuration file: C:\Users\jesus\Downloads\sonar-scanner-4.7.0.2747-windows\bin\..conf\sonar-scanner.properties
```

```
▼ TERMINAL
INFO: CPD Executor CPD calculation finished (done) | time=14ms
INFO: Analysis report generated in 436ms, dir size=218 KB
INFO: Analysis report compressed in 504ms, zip size=62 KB
○ INFO: Analysis report uploaded in 2981ms
INFO: ANALYSIS SUCCESSFUL, you can find the results at: https://sonarcloud.io/dashboard?id=sonar-tp-mti-pds
INFO: Note that you will be able to access the updated dashboard once the server has processed the s
submitted analysis report
INFO: More about the report processing at https://sonarcloud.io/api/ce/task?id=AYR4SaDHxcnPmYz0Gd11
INFO: Analysis total time: 1:27.178 s
INFO: -----
INFO: EXECUTION SUCCESS
INFO: -----
INFO: Total time: 1:36.504s
INFO: Final Memory: 44M/154M
INFO: -----
PS C:\Users\jesus\projects\MTI\test-project>
```

Podemos ver nuestro proyecto ya analizado en la interfaz de SonarCloud.

The screenshot shows the SonarCloud web interface for user 'Jesus Andres'. The 'Projects' tab is active, displaying a list of projects. The project 'sonar-tp-mti-pds' is highlighted with a red box. The project is marked as 'NEW' and 'PUBLIC'. The last analysis was performed on 11/14/2022 at 7:35 PM, resulting in 171 Lines of Code (TypeScript). The analysis results are as follows:

Metric	Value	Grade
Bugs	0	A
Vulnerabilities	0	A
Hotspots Reviewed	0.0%	E
Code Smells	8	A
Coverage	0.0%	
Duplications	0.0%	

El análisis indica que se encontraron cero Bugs (calificación A) y cero Vulnerabilidades (calificación A).

Para los Hotspots Reviewed la calificación es E y está 0% cubierto, pero eso es debido a que no lo estamos usando (no hemos revisado ni un Hotspot). De cualquier forma, esto no es un impedimento para que el código analizado pase la prueba a menos que se configure lo contrario.

Se encontraron 8 Code Smells a ser factorizados.

El código está cubierto por tests en un 0% (no hemos creado tests para este código) y tiene 0% de duplicación de código.

SonarQube en un servidor

En este caso usaremos nuestra pc personal como servidor, pero lo ideal es instalar la herramienta en un servidor remoto real.

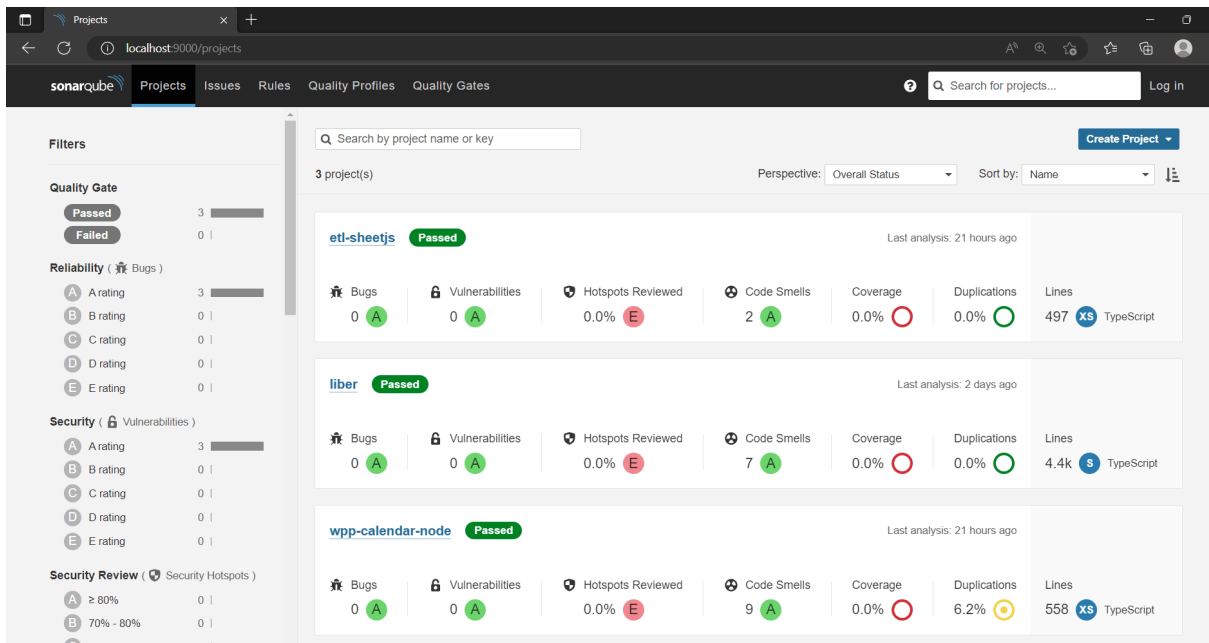
Preparamos un archivo docker-compose.yml que contiene la imagen oficial de SonarQube y una base de datos PostgreSQL. (<https://github.com/jesusandres31/sonar-qube-docker>)

```
docker-compose.yml X
docker-compose.yml
21  version: "3"
22
23  services:
24    sonarqube:
25      image: sonarqube:community
26      depends_on:
27        - db
28      environment:
29        SONAR_JDBC_URL: jdbc:postgresql://db:5432/sonar
30        SONAR_JDBC_USERNAME: sonar
31        SONAR_JDBC_PASSWORD: sonar
32      volumes:
33        - sonarqube_data:/opt/sonarqube/data
34        - sonarqube_extensions:/opt/sonarqube/extensions
35        - sonarqube_logs:/opt/sonarqube/logs
36      ports:
37        - "9000:9000"
38
39    db:
40      image: postgres:12
41      environment:
42        POSTGRES_USER: sonar
43        POSTGRES_PASSWORD: sonar
44      volumes:
45        - postgresql:/var/lib/postgresql
46        - postgresql_data:/var/lib/postgresql/data
47
48  volumes:
49    sonarqube_data:
50    sonarqube_extensions:
51    sonarqube_logs:
52    postgresql:
53    postgresql_data:
54
```

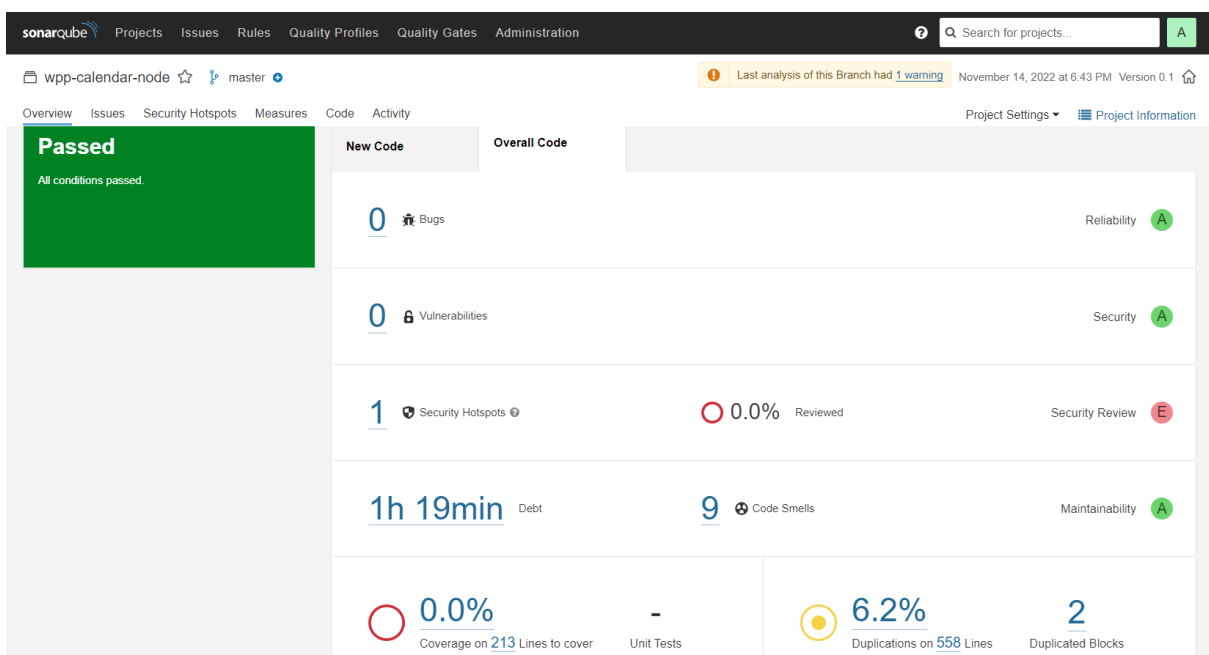
Levantamos los contenedores.

```
EXPLORER
▼ TERMINAL
PS C:\Users\jesus\projects\SonarQube>
● docker-compose up -d
Starting sonarqube_db_1 ... done
Starting sonarqube_sonarqube_1 ... done
● PS C:\Users\jesus\projects\SonarQube> docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED        STATUS        PORTS                NAMES
4bea408fec03   sonarqube:community "/opt/sonarqube/bin/..." 5 minutes ago  Up 2 seconds  0.0.0.0:9000->9000/tcp sonarqube_sonarqube_1
20624954ee65   postgres:12         "docker-entrypoint.s..." 5 minutes ago  Up 3 seconds  5432/tcp             sonarqube_db_1
```

Accedemos al servidor de SonarQube.



Podemos ver listados nuestros proyectos ya analizados o crear nuevos y analizarlos con SonnarScanner tal cual se mostró en la secuencia de pasos anterior.



Si ingresamos a un proyecto vemos como Sonar lista los issues de nuestro código, y hasta estima cuánto tiempo vamos a tardar en solucionar estos issues mediante refactorización.

Si ingresamos por ejemplo a la sección de code smells podemos ver aquí listados los mismos.

The screenshot shows the SonarQube 'Measures' page for the project 'wpp-calendar-node'. The left sidebar indicates a 'Technical Debt' of 1h 19min. The main content area lists the following components and their estimated effort to resolve:

Component	Effort
src/helpers/notifyStatus.ts	37min
src/app.ts	20min
src/helpers/sendAllMessages.ts	15min
src/config/index.ts	5min
src/services/whatsapp.service.ts	2min

A message at the bottom indicates: 'There are 34 hidden components with a score of 0.' with a 'Show Them' button.

Como ejemplo de un code smell, aquí Sonar nos sugiere remover esta asignación inútil de variable.

The screenshot shows a 'Code Smell' issue in the file 'src/helpers/sendAllMessages.ts'. The issue is titled 'Remove this useless assignment to variable "payload"'. The code snippet shows a function 'if (contact) { ... }' where 'payload' is assigned but not used. The issue is classified as a 'Code Smell' with a 'Major' severity and a '15min effort' to resolve.

```
31 if (contact) {
32   const number = contact.number;
33   if (number) {
34     const name = formatName(playerFullName);
35     const time = formatTime(startDate);
36     const message = createMessage(name, time);
37     const payload = await whatsappSvc.sendWhatsAppMessage(
38       message,
39       number,
40     );
41   }
42 }
```

En este otro caso nos marca con una línea gris el código duplicado que podría ser refactorizado por ejemplo extrayendo ese código en una función; y que se utilice esa función en todos los lugares necesarios.

Project Overview

- > Reliability ?
- > Security ?
- > Security Review ?
- > Maintainability ?
- > Coverage
- ▼ Duplications **DUPLICATED LINES (...)**
- Overview
- Overall
- Density 51.5%
- Duplicated Lines 50
- Duplicated Blocks 2
- Duplicated Files 1
- > Size
- > Complexity ?
- > Issues

wpp-calendar-node / src / helpers / sendAllMesagges.ts

```
6  jesu...
7
8      /**
9      * send all messages
10     */
11     export const sendAllMesagges = async (
12     contacts: IContact[],
13     events: calendar_v3.Schema$Event[],
14     ) => {
15         // responses
16         const responses: IContact[] = [];
17         // not match yet
18         let notMatchingContacts: string[] = [];
19
20         // iterate through each event
21         for (const event of events) {
22             const playerFullName = event.summary;
23             const startDate = event.start?.dateTime;
24
25             if (playerFullName && startDate) {
26                 // find contact if "event title" and "contact name" matchs
27                 const contact = contacts.find(
28                     contact => trimString(contact.name) === trimString(playerFullName),
29                 );
30
31                 if (contact) {
```

Duplicated block. Click for details.

Lint

Cabe también mencionar la existencia de los Lints. Esta es una herramienta (plugin para el IDE) para el análisis estático de código en tiempo de codificación. Detecta errores de programación (vulnerabilidades y code smells) que escapan al habitual análisis sintáctico que hace el compilador.

Utilizando un lint podemos detectar en tiempo de desarrollo:

- Usos de variables antes de ser inicializadas o creadas.
- Variables/objetos instanciadas pero nunca utilizadas.
- Condiciones que no varían (siempre verdaderas o siempre falsas).
- Uso de diversas "malas prácticas".

La utilización de un lint en nuestro entorno de desarrollo nos ayuda a detectar este tipo de vulnerabilidades y code smells incluso antes de que estos lleguen a SonarQube.

Algunos ejemplos de estas herramientas son SonarLint (soporte para varios lenguajes) o ESLint (typescript o javascript).

```
.eslintrc.js > ...
1  module.exports = {
2    parser: "@typescript-eslint/parser",
3    plugins: ["@typescript-eslint/eslint-plugin", "prettier"],
4    extends: ["plugin:@typescript-eslint/recommended", "prettier"],
5    rules: {
6      "prettier/prettier": "error",
7      "eqeqeq": "error",
8      "no-empty-pattern": "off",
9      "array-callback-return": "error",
10     "@typescript-eslint/ban-types": "off",
11     "@typescript-eslint/ban-ts-comment": "off",
12     "@typescript-eslint/no-explicit-any": "off",
13     "@typescript-eslint/no-unused-vars": "error",
14     "@typescript-eslint/no-empty-function": "off",
15     "@typescript-eslint/no-use-before-define": "off", // Not Recommended on TypeScript https://palantir
16     "@typescript-eslint/no-useless-constructor": "error",
17     "@typescript-eslint/consistent-type-assertions": "error",
18     "@typescript-eslint/explicit-module-boundary-types": "off",
19   },
20 };

```