# SALECOLD

## INFORME DE CALIDAD EN EL CÓDIGO FUENTE



## ¿Qué es SonarQube?

SonarQube es una plataforma de código abierto para la inspección de la calidad del código a través de diferentes herramientas de análisis de código fuente. Esta plataforma proporciona métricas que ayudan a mejorar la calidad del código.

---

## Principales métricas de SonarQube

- **Complejidad:** Es una métrica de calidad software basada en el cálculo del número de caminos independientes que tiene nuestro código.
- **Duplicados:** Nos indica el número de bloques de líneas duplicados.
-

**Bugs:** Son errores o defectos en el software.

- **Vulnerabilidad:** Son problemas de seguridad en el software y deben ser solucionados de inmediato.
- **Security Hotspot:** Es un fragmento de código sensible a la seguridad, pero es posible que no afecte la seguridad general del software.
- **Mantenibilidad:** Se refiere al recuento total de problemas de Code Smell o malas practicas implementadas en el proyecto.
- **Tamaño:** Permiten hacerse una idea del volumen del proyecto en términos generales.
- **Pruebas:** Son una forma de comprobar el correcto funcionamiento de una unidad de código y de su integración.

## Instalación y configuración de SonarQube en Linux

## Descargar  e instalar openjdk 11 o jdk 11

> Java Archive Downloads - Java SE 11

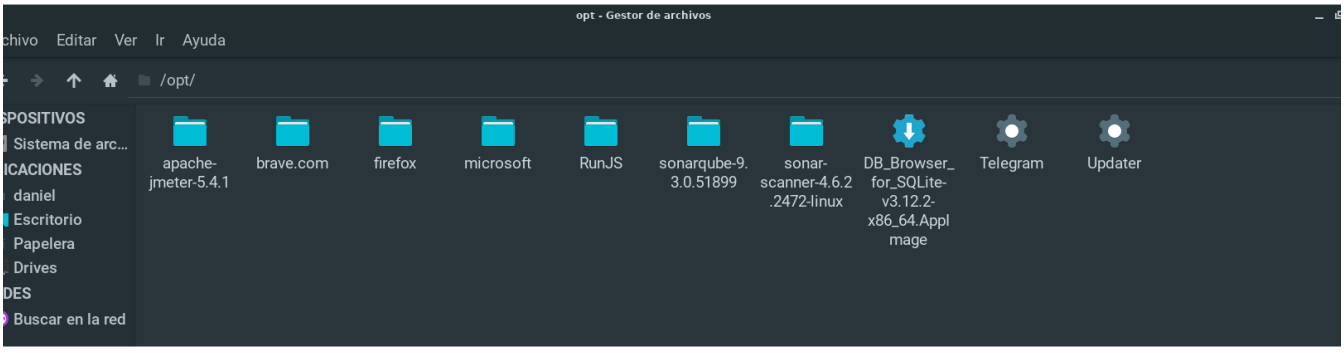
> JDK 11

---

## Descargar SonarQube Community edition

> Code Quality and Code Security | SonarQube
> sonarqube

#### 1. Extraer el zip en la raiz del disco duro



### 2. Modificar la configuración del archivo wrapper.conf el cual esta en la carpeta conf de sonarqube

> En la tercer linea del archivo se especifica el path donde esta instalado el openjdk o el jdk.

```
# Path to JVM executable. By default it must be available in PATH.
# Can be an absolute path, for example:
wrapper.java.command=/usr/lib/jvm/java-11-openjdk-amd64/bin/java
#wrapper.java.command=java
```

### 3. Acceder a la carpeta bin, luego a linux-x86-64 y abrir una terminal

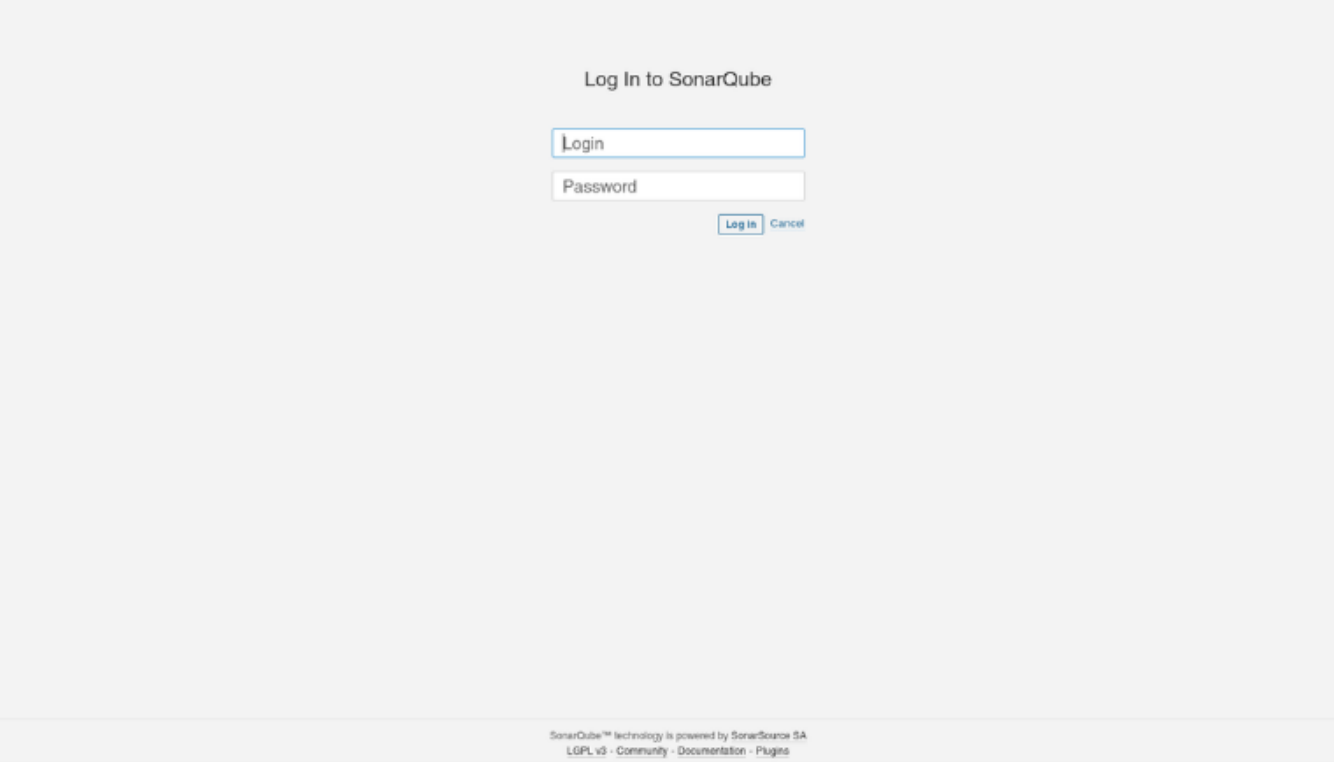
Ejecutar el siguiente comando para iniciar el servidor de SonarQube.

```
sh sonar.sh start
```

### 4. Acceder en el navegador a la interfaz de sonarqube

Se debe colocar en el navegador localhost:9000 o 127.0.0.1:9000.

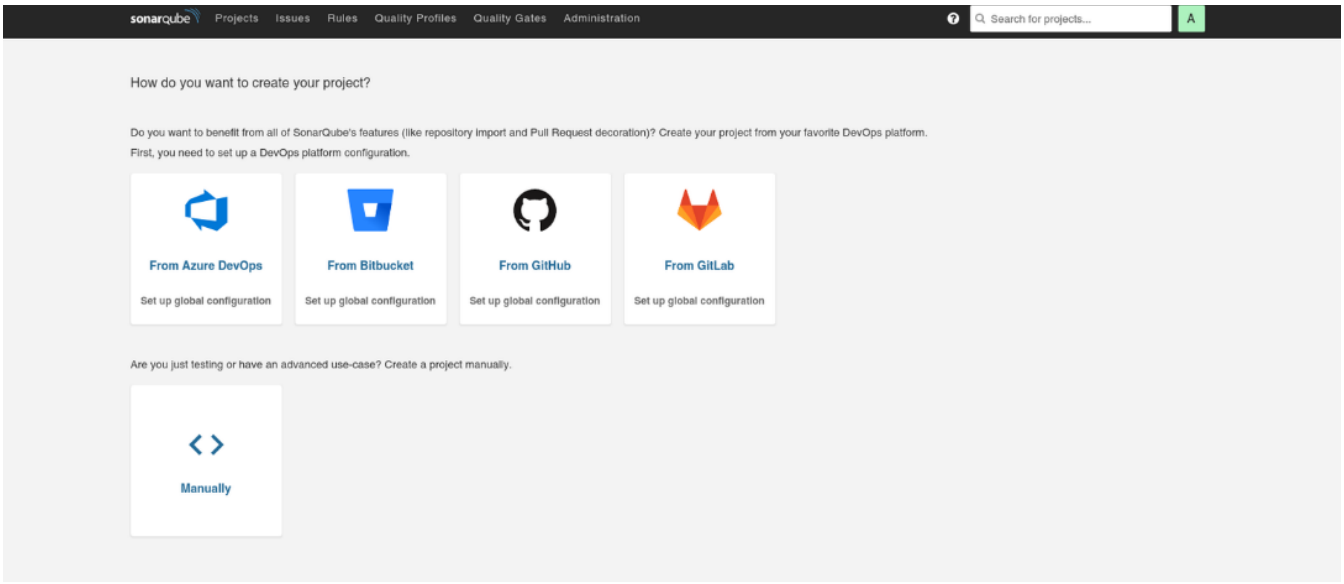
> La primera vez que se accede el usuario es admin y la contraseña admin, después nos pedirá actualizar la contraseña.



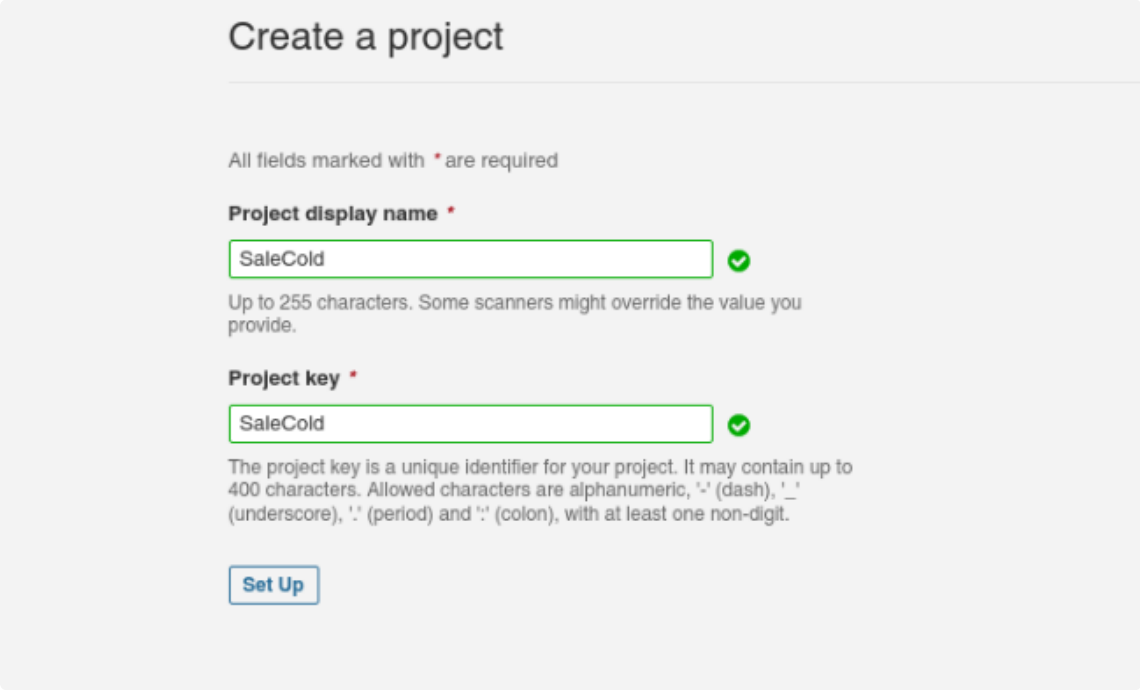
# Configuración de SonarQube en el proyecto

### 1. Seleccionar la forma en como se creara el nuevo proyecto en SonarQube

En este caso se seleccionara la opcion “Manually”.

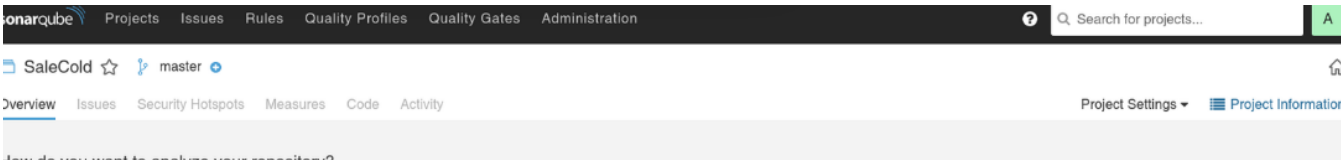


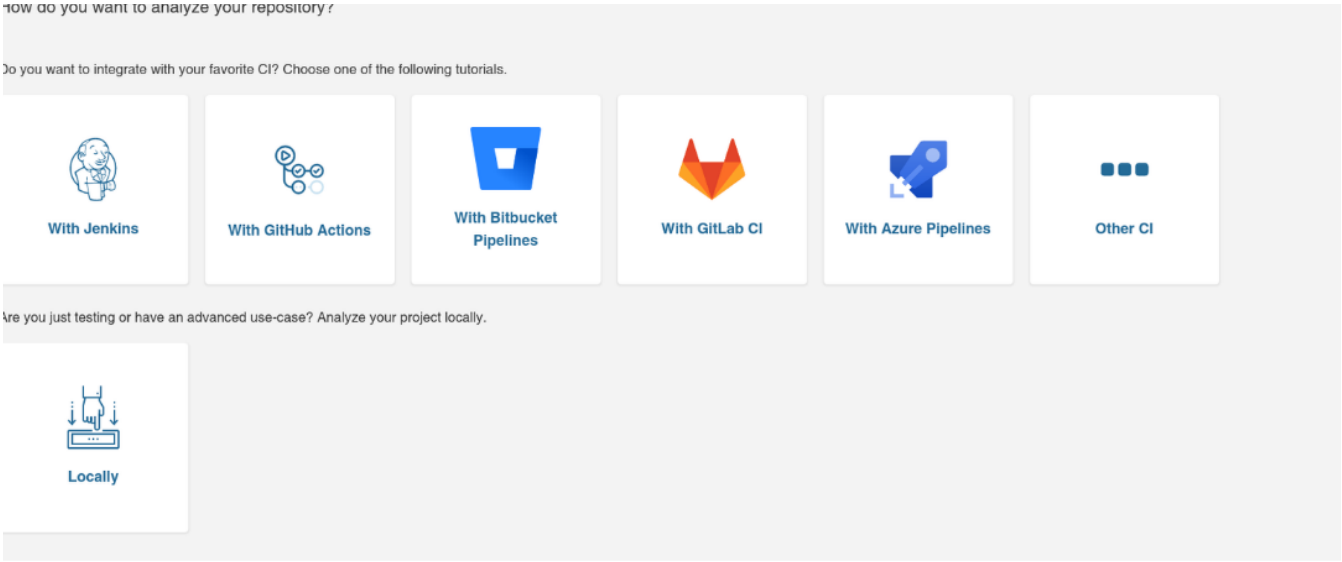

### 2. Especificar el nombre del nuevo proyecto y el identificador único



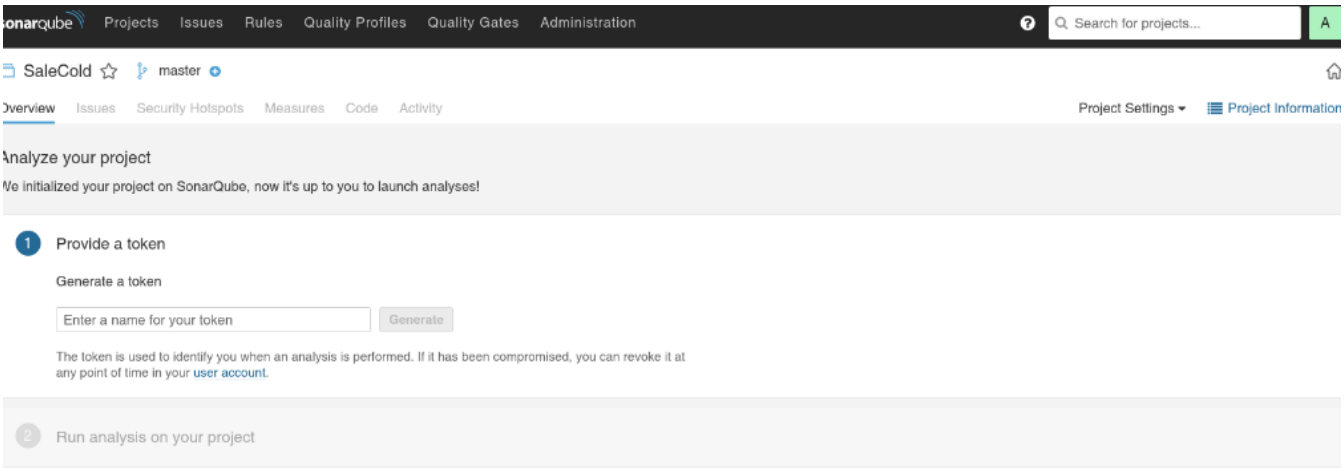

### 3. Seleccionar la forma en como se analizara el código del proyecto

En este caso seleccionaremos la opcion “Locally”.

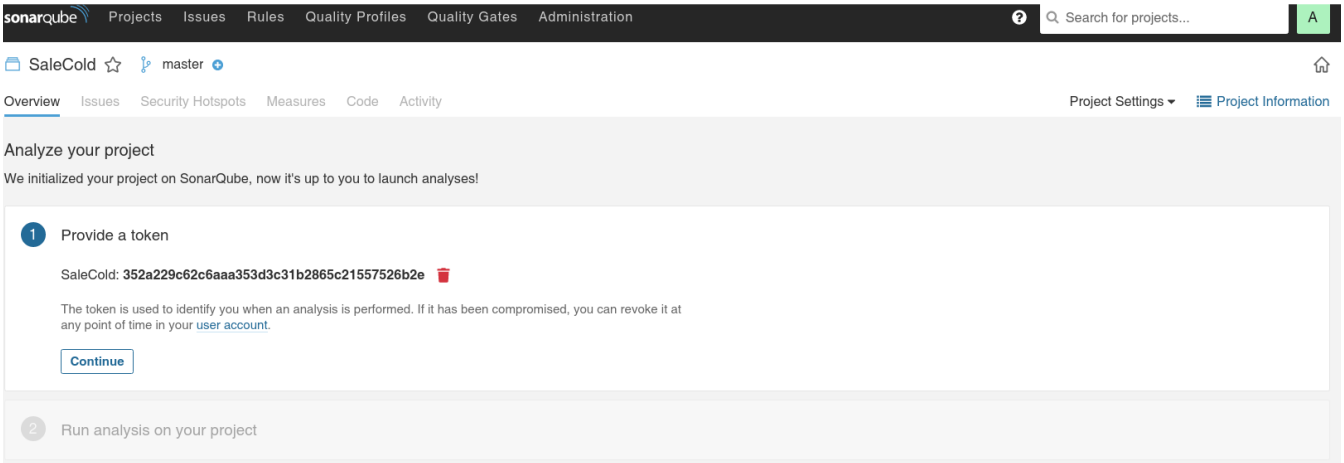




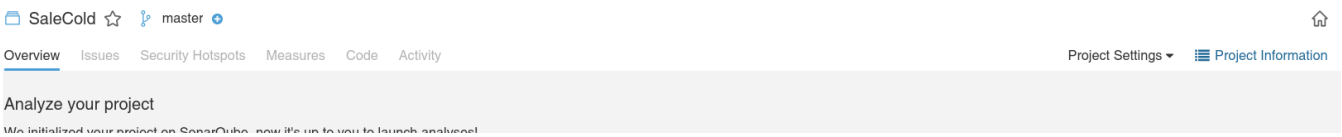

## 4. Se crea o se utiliza un token ya generado

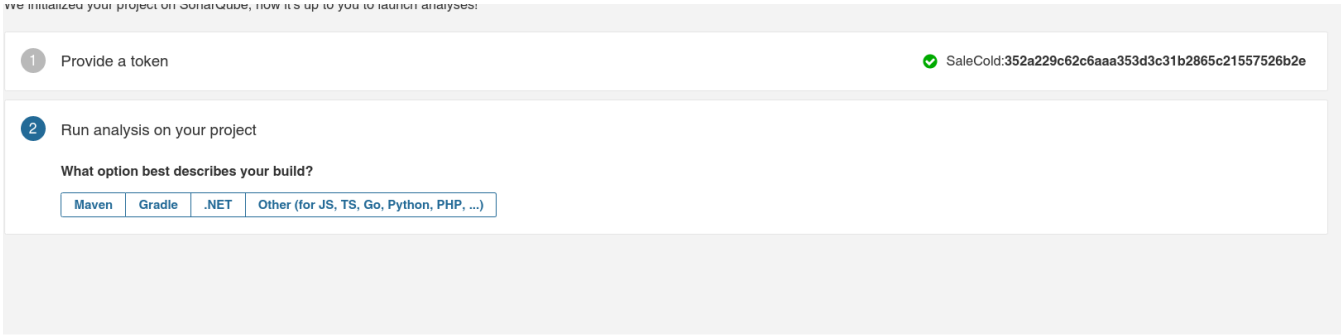



Damos clic en generate:

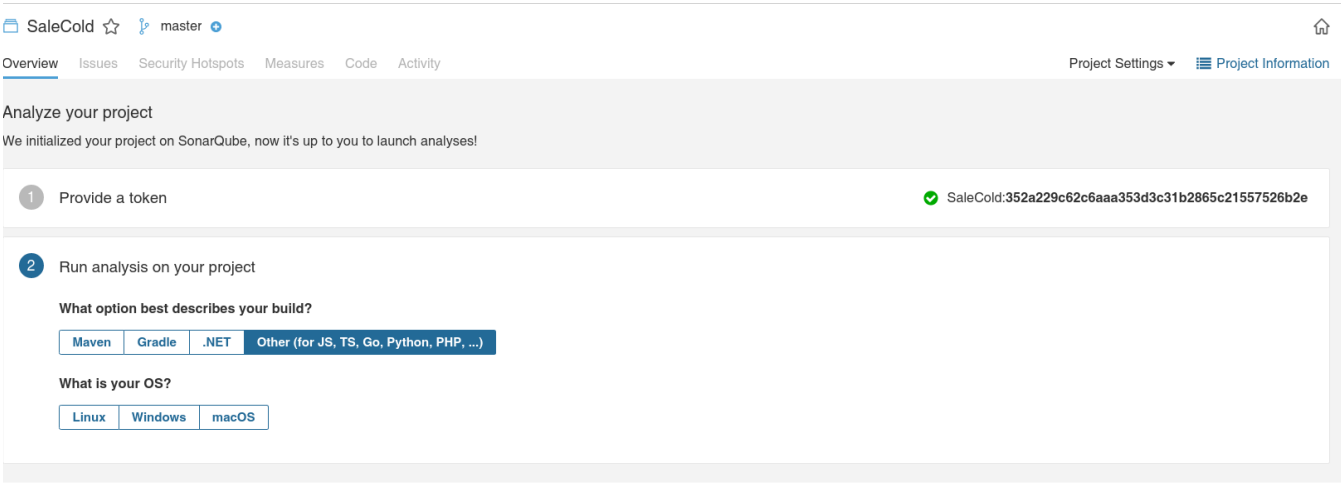



## 5. Seleccionar la tecnología o el lenguaje de programación que se uso en el proyecto





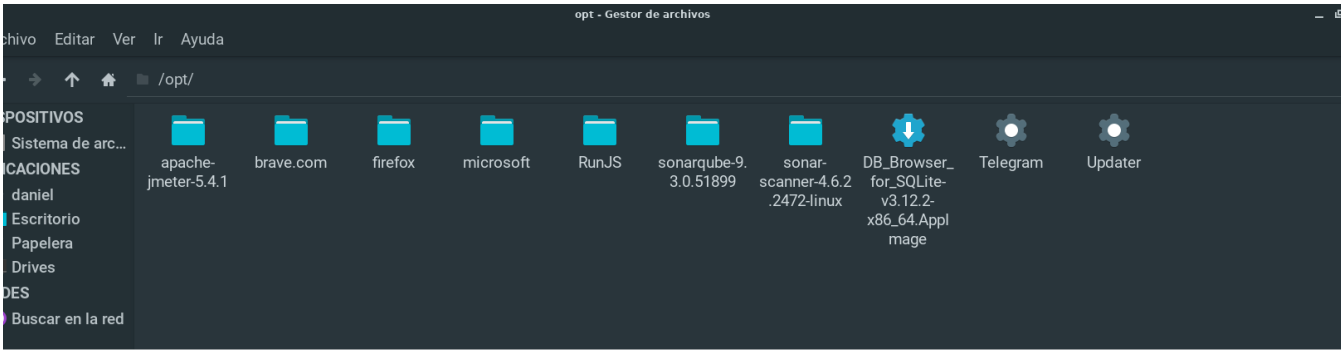

## 6. Seleccionar el sistema operativo que estamos usando




## 7. Descargar SonarScanner

SonarScanner | SonarQube Docs

## 8. Extraer el zip en la raiz del disco duro

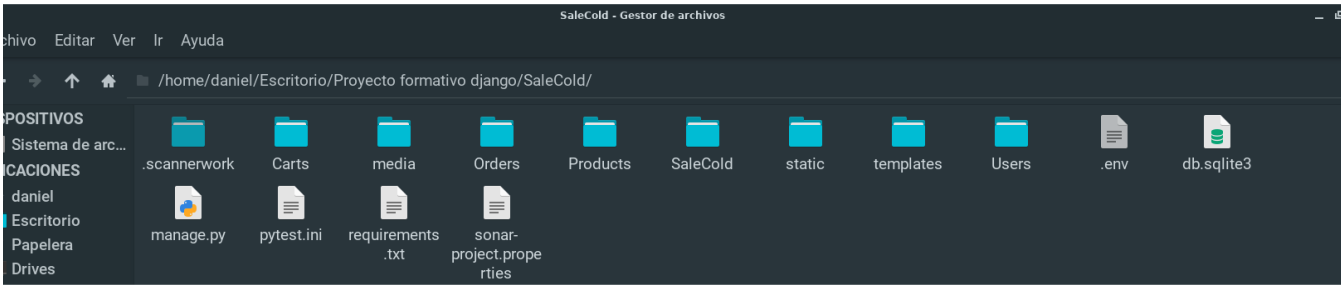



## 9. En el archivo .bashrc se agrega el path o la ruta donde esta almacenada la carpeta de SonarScanner

```
1 PATH="/opt/sonar-scanner-4.6.2.2472-linux/bin:$PATH"
```

### 10. Crear un archivo de configuración de sonarqube en la raiz del proyecto

> ⓘ El archivo se debe llamar sonar-project.properties



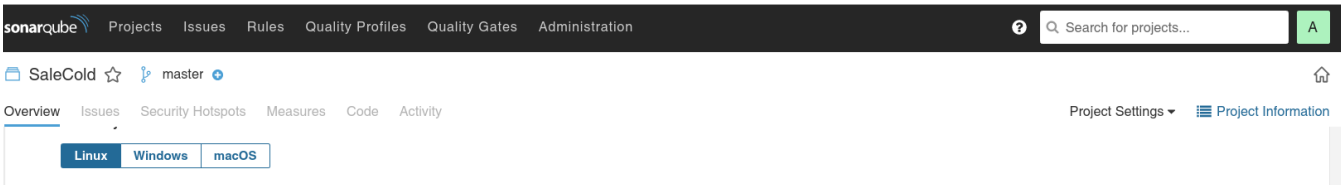

### 11. Agregar la siguiente configuración al archivo creado en el paso anterior

- **sonar.projectKey =** Acá debe ir el nombre del proyecto que especificamos en sonarqube.
- **sonar.projectName =** Acá debe ir el nombre de la carpeta raiz del proyecto que vamos a analizar.

```
1  # must be unique in a given SonarQube instance
2  sonar.projectKey=SaleCold
3  # --- optional properties ---
4
5  # defaults to project key
6  sonar.projectName=SaleCold
7  # defaults to 'not provided'
8  sonar.projectVersion=1.0
9
10 # Path is relative to the sonar-project.properties file. Defaults to .
11 sonar.sources=.
12
13 # Encoding of the source code. Default is default system encoding
14 sonar.sourceEncoding=UTF-8
15
16 sonar.python.version=3.8
```

### 12. Ejecutar el análisis al proyecto

Después de escoger la tecnología del proyecto y el sistema operativo, ademas de configurar e instalar el sonarScanner, sonarqube nos genera un código el cual se debe ejecutar en una nueva terminal y la terminal debe estar en la carpeta raiz del proyecto.



**Download and unzip the Scanner for Linux**

Visit the official documentation of the Scanner to download the latest version, and add the `bin` directory to the `PATH` environment variable

**Execute the Scanner**

Running a SonarQube analysis is straighforward. You just need to execute the following commands in your project's folder.

```
sonar-scanner \
  -Dsonar.projectKey=SaleCold \
  -Dsonar.sources=. \
  -Dsonar.host.url=http://localhost:9000 \
  -Dsonar.login=352a229c62c6aaa353d3c31b2865c21557526b2e
```

Copy

Please visit the official documentation of the Scanner for more details.

**Is my analysis done?** If your analysis is successful, this page will automatically refresh in a few moments.

You can set up Pull Request Decoration under the project settings. To set up analysis with your favorite CI tool, see the tutorials.

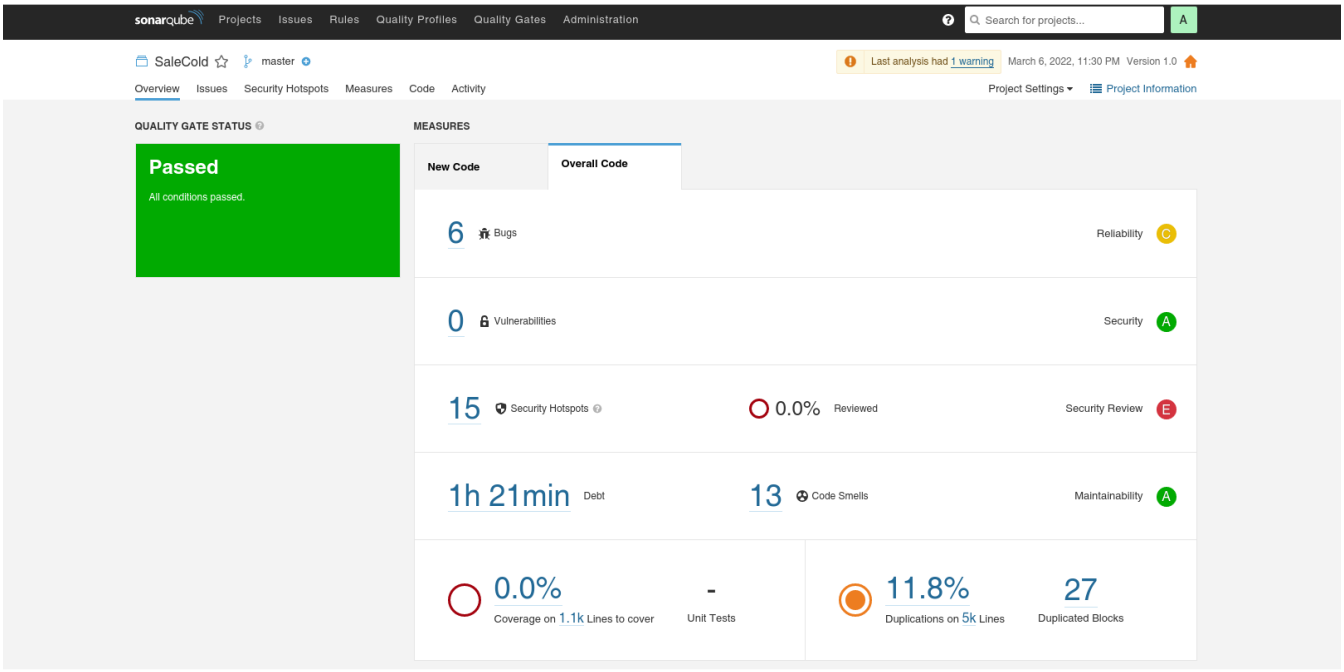
Check these useful links while you wait: Branch Analysis, Pull Request Analysis.

```
daniel ~ > Escritorio > Proyecto formativo django > SaleCold sonar-scanner \
    -Dsonar.projectKey=SaleCold \
    -Dsonar.sources=. \
    -Dsonar.host.url=http://localhost:9000 \
    -Dsonar.login=352a229c62c6aaa353d3c31b2865c21557526b2e
```

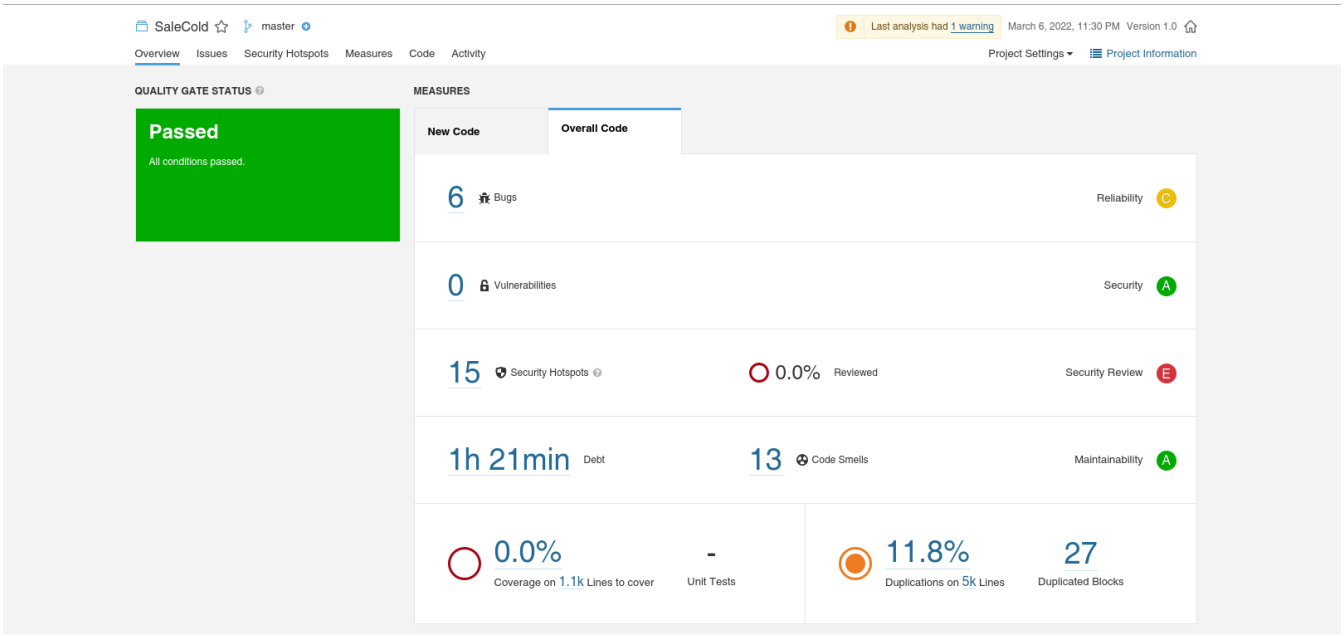
Si el análisis se realizo de manera correcta nos debe generar el siguiente mensaje en la terminal:

```
FO: ANALYSIS SUCCESSFUL, you can browse http://localhost:9000/dashboard?id=SaleCold
FO: Note that you will be able to access the updated dashboard once the server has processed the submitted
ysis report
FO: More about the report processing at http://localhost:9000/api/ce/task?id=AX83nEPkwYQgiyCj-nB_
FO: Analysis total time: 52.514 s
FO: ------------------------------------------------------------------------
FO: EXECUTION SUCCESS
FO: ------------------------------------------------------------------------
FO: Total time: 59.664s
FO: Final Memory: 23M/80M
FO: ------------------------------------------------------------------------
aniel ~ > Escritorio > Proyecto formativo django > SaleCold
```

Pegamos el link que nos genero sonarqube en el navegador para ver el resultado del análisis:



# Resultado del análisis hecho por SonarQube al código fuente




Resultado del análisis hecho por Sonarqube al código fuente

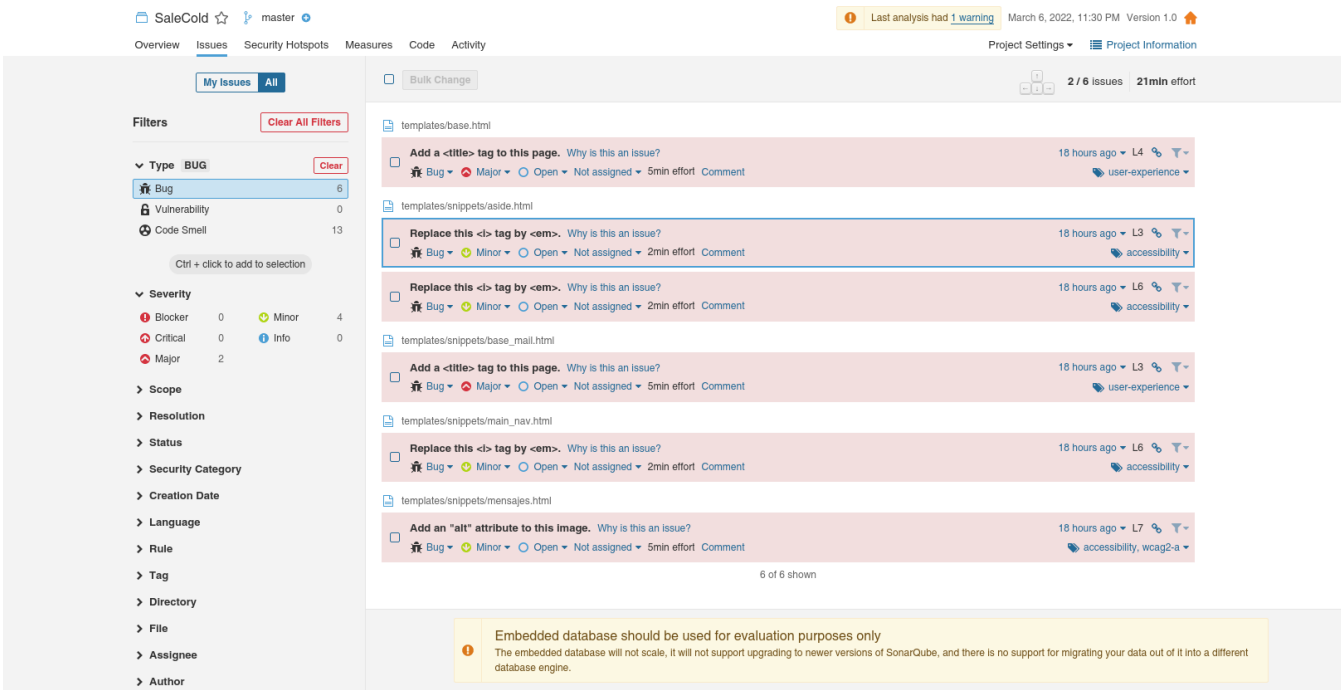
Resumen del análisis:

- Se encontraron 6 bugs.
- El software tiene 0 vulnerabilidades de seguridad.
- Se encontraron 13 Code smells  o malas practicas.
- Se encontraron 15 fragmentos de codigo que podrían afectar la seguridad del software.
- Se demoraria aproximadamente 1h 21 minutos para refactorizar el codigo.
- El 11.8% del codigo esta duplicado.

# Ajustes en el código fuente

- [ ] Bugs
- [ ] Security Hotspots
- [ ] Code smell

# Bugs




---

# Solución bugs 2, 3 y 5.

Para solucionar este bug se le agrega el atributo aria-hidden para que la tecnología de asistencia del navegador omita este icono, según la MDN este atributo ayuda a mejorar la accesibilidad de la pagina.

> aria-hidden - Accessibility | MDN


---

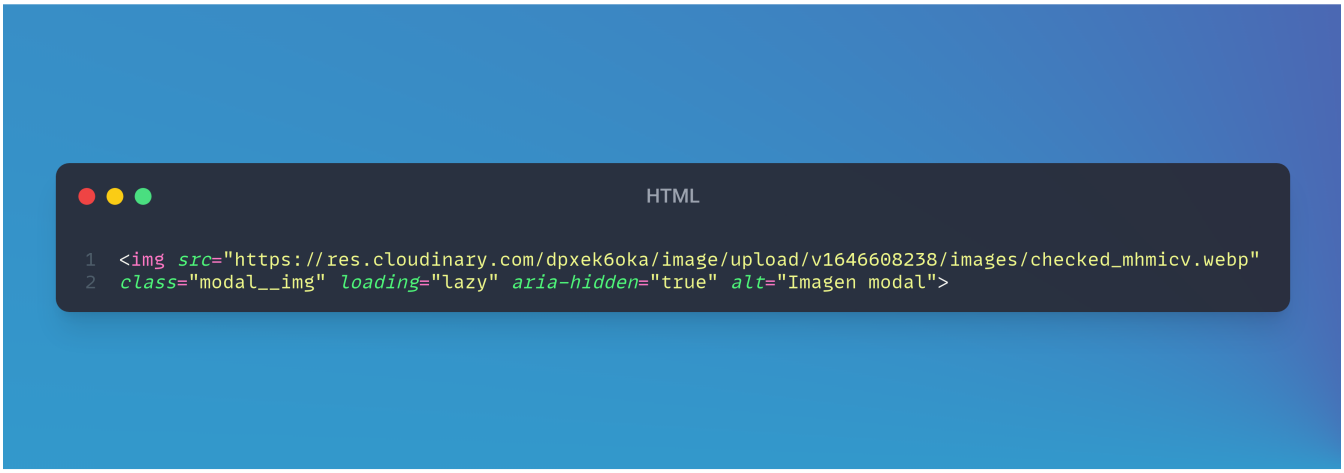
## Solución bugs 1 y 4

Estos bugs son falsos positivos porque  se usan para la herencia entre templates, es decir son plantillas por ende no necesitan la etiqueta title.

---

## Solución bug 6

Para la solución de este bug se agrega el atributo alt para ofrecer un texto alternativo en caso de que el navegador no cargue la imagen.

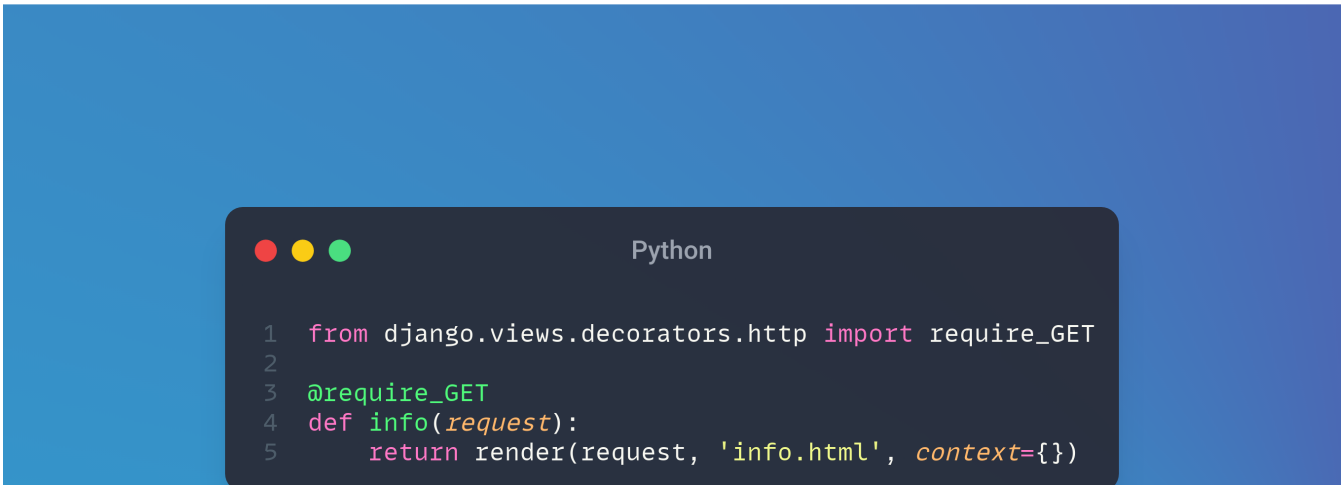



## Security Hotspots

## Solución security hotspots 1 al 14

Para solucionar estos security hotspots se le agrego un decorador a cada una de las vistas para restringir el tipo de petición que pueden recibir, de esta manera se solucionan las posibles vulnerabilidades de seguridad.

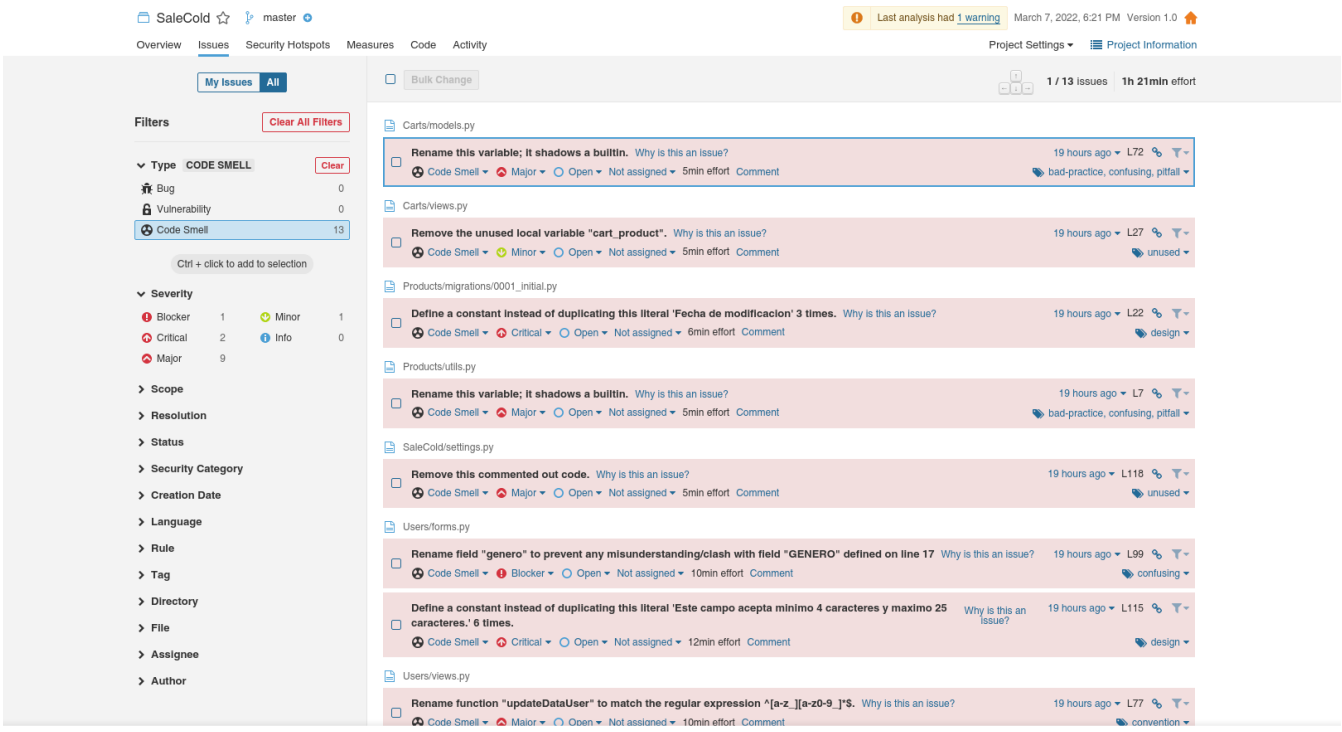
**Ejemplo #1 de una vista con el decorador agregado:**



### Ejemplo #2 de una vista con el decorador agregado:

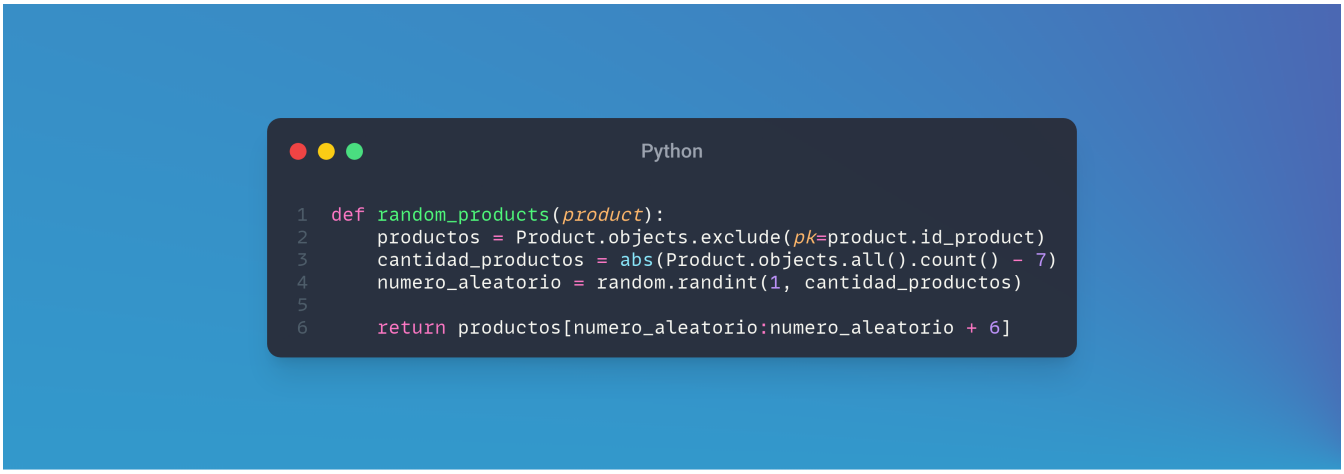
```python
from django.views.decorators.http import require_http_methods

@require_http_methods(['GET', 'POST'])
def cart_view(request):
    cart = get_or_create_cart(request)

    return render(request, 'cart/carrito de compras.html', context = {
        'carrito': cart,
    })
```

## Code smell



## Solucion code smell 1 y 3

Para solucionar estos code smell se re nombraron dos variables porque estaban usando  palabras reservada del lenguaje.




---

## Solucion code smell 4

Para solucionar este code smell se elimino una linea de codigo que no se estaba usando.

---

## Solucion code smell 7

Para solucionar este code smell se creo una variable constante al inicio del archivo y luego se utilizo en los lugares donde se repetía la cadena.


Ejemplo de una funcion para validar un campo del formulario usando la variable constante:

```python
def clean_nombre(self):
    nombre = self.cleaned_data.get('nombre')

    if len(nombre) == 0:
        raise forms.ValidationError(CAMPO_OBLIGATORIO)
    elif len(nombre) < 4 or len(nombre) > 25:
        raise forms.ValidationError(ERROR_CANTIDAD_CARACTERES)

    return nombre
```

## Solucion code smells 10 al 13

La solución de estos tres code smells fue eliminar el codigo repetido.

## Conclusión

Después de realizar la refactorizacion al codigo para solucionar los problemas de calidad, se ejecuta de nuevo el análisis dando un resultado satisfactorio, ya que el codigo fuente del aplicativo cuenta con calidad.