Asystent AI w repozytorium PlusWorkflow sam sprawdza zmieniony kod regułami naszego serwera Sonara, zanim zgłosi, że praca jest gotowa. Instrukcję, jak to robić, ma w skillu `.claude/skills/sonarqube-analysis/SKILL.md` (skill to plik z instrukcją, którą asystent wczytuje przy pracy w danym obszarze). Dzięki temu uwagi Sonara poprawiamy przed MR-em, a nie w osobnym zgłoszeniu po scaleniu.

> Analiza przez asystenta wymaga serwera MCP SonarQube. Konfigurację dla omp opisuje strona SonarQube - konfiguracja serwera MCP dla omp. Skill czytają też Claude Code, Cursor i Codex, ale dla nich połączenia z Sonarem nie opisaliśmy — bez niego asystent poprosi o uwagi z IDE (sekcja „Gdy asystent nie ma połączenia z Sonarem").

Jak to działa

1. Po zakończeniu zmian w plikach `.java`, `.ts`, `.js`, `.vue`, `.jsp`, `.html`, `.less` lub `.xml` asystent sam uruchamia analizę. Nie trzeba go o to prosić.
2. Analizuje tylko pliki zmienione w bieżącym zadaniu, względem gałęzi master. Twoich niezacommitowanych zmian, których sam nie robił, nie analizuje.
3. Używa profilu reguł projektu PlusWorkflow z serwera (dla Javy „Suncode java"), czyli tych samych reguł co serwer i wtyczka w IDE w trybie Connected Mode.
4. Dzieli uwagi na dwie grupy: na zmienionych liniach (te liczą się na serwerze do nowego kodu i bramki jakości) oraz w starym kodzie.
5. Każdej uwadze na zmienionej linii nadaje werdykt i od razu go stosuje (tabela niżej).
6. Po poprawkach uruchamia analizę ponownie.
7. Na końcu odpowiedzi pokazuje tabelę: plik i linia, reguła, werdykt, co zrobił. Pod nią osobno uwagi w starym kodzie.

Werdykty

Werdykt	Kiedy	Co robi asystent
Poprawić	Reguła wskazuje rzeczywisty problem albo poprawka jest prostsza niż uzasadnianie wyjątku	Poprawia kod
Fałszywy alarm	Kontekst kodu wyklucza problem, co asystent sprawdził w kodzie	Nie zmienia kodu. Podaje uzasadnienie do oznaczenia zgłoszenia na serwerze po scaleniu
Artefakt analizy	Reguła `java:S1598` (ścieżka pliku a pakiet) — narzędzie nie dostaje ścieżki pliku	Pomija, nie raportuje

Czego asystent nie robi bez Twojej zgody:

- Nie poprawia uwag w starym kodzie — nie liczą się do nowego kodu, a poprawki poszerzają zakres MR-a. Jeśli chcesz je poprawić, powiedz mu o tym.
- Nie zmienia statusów zgłoszeń na serwerze (False Positive, Accepted). Zmianę widzi cały zespół i jest podpisana Twoim tokenem, więc robi to tylko na wyraźną prośbę, np. „oznacz te zgłoszenia jako False Positive z uzasadnieniem".
- Nie ukrywa fałszywych alarmów w kodzie adnotacją `@SuppressWarnings` ani

komentarzem `// NOSONAR`.

Ograniczenie analizy przez asystenta

Asystent przekazuje do Sonara sam plik, bez bibliotek projektu. Reguły, które muszą znać typy z bibliotek, mogą się wtedy nie uruchomić. Przykład: reguła `java:S6814` nie zgłasza parametru REST `@RequestParam( required = false ) int`, choć wtyczka w IDE i serwer go zgłoszą. Brak uwag od asystenta nie gwarantuje więc braku uwag na serwerze — przed MR-em warto rzucić okiem na uwagi wtyczki w IDE (strona SonarQube - Connected Mode w IntelliJ).

Gdy asystent nie ma połączenia z Sonarem

Jeśli serwer MCP nie działa albo nie jest skonfigurowany, asystent poprosi o uwagi z wtyczki SonarQube for IDE. Wtyczka pokazuje uwagi bez nazwy pliku, np.:

```
(46, 43) Define a constant instead of duplicating this literal
"Origin" 3 times. [java:S1192]
```

Wklejając uwagi, dopisz nad każdą grupą nazwę pliku, którego dotyczą.

Przykładowe polecenia

- „Sprawdź stan bramki jakości projektu PlusWorkflow” — warunki bramki i ich wartości.
- „Wylistuj zgłoszenia Sonara o wysokiej wadze w nowym kodzie i zweryfikuj je w kodzie” — lista z oceną, które są prawdziwe.
- „Przeanalizuj ten plik Sonarem” — analiza na żądanie, poza automatycznym flow.