Bramka jakości (quality gate) to zestaw warunków, które musi spełnić nowy kod projektu PlusWorkflow, żeby analiza na serwerze dostała status „Passed”. Projekt używa bramki „Suncode way”. Bramka ocenia tylko nowy kod, więc dług techniczny w starszym kodzie jej nie blokuje — każda zmiana ma jednak spełniać progi.

Warunki

Warunek	Metryka w Sonarze	Próg	Wartoś 14.11.2
Pokrycie testami nowego kodu	`new_coverage`	co najmniej 40%	32,9%
Zduplikowane linie w nowym kodzie	`new_duplicated_lines_density`	najwyżej 3%	1,7%
Przejrzane hotspoty bezpieczeństwa	`new_security_hotspots_reviewed`	100%	brak hotspot
Ocena niezawodności	`new_software_quality_reliability_rating`	A	D
Ocena bezpieczeństwa	`new_software_quality_security_rating`	A	C
Ocena utrzymywalności	`new_software_quality_maintainability_rating`	A	A

Wszystkie warunki muszą być spełnione jednocześnie. Aktualne wartości są na stronie projektu w Sonarze; tabela pokazuje stan z dnia przygotowania tej strony.

Oceny A–E

Oceny niezawodności i bezpieczeństwa zależą od wagi najpoważniejszego otwartego zgłoszenia w nowym kodzie:

Ocena	Najpoważniejsze otwarte zgłoszenie
A	brak zgłoszeń albo tylko o wadze Info
B	Low
C	Medium
D	High
E	Blocker

Próg „A” oznacza, że jedno otwarte zgłoszenie o wadze Low w nowym kodzie wystarcza, żeby bramka nie przeszła. Zgłoszenia oznaczone jako False Positive albo Accepted nie liczą się do oceny (strona SonarQube - fałszywe alarmy i oznaczanie zgłoszeń). Ocena utrzymywalności liczy się inaczej: jako stosunek szacowanego czasu naprawy długu technicznego do szacowanego czasu napisania nowego kodu (A to najwyżej 5%).

Czym jest nowy kod

Nowy kod to linie dodane lub zmienione od analizy bazowej z 16.10.2026. Serwer ma tak ustawiony okres nowego kodu dla gałęzi main (tryb „konkretna analiza”, ustawienie zmienione 16.10.2026). Wszystko, co zmieniło się w kodzie od tej daty, liczy się do bramki — także linie tylko przeformatowane albo przeniesione, np. przy migracji JSP ze Struts na Spring.

Okres nowego kodu ustawia administrator Sonara. Gdy zostanie przesunięty (np. przy wydaniu wersji), zgłoszenia ze starszych zmian przestają liczyć się do bramki.

Jak nie psuć bramki

- Zgłoszenia: poprawiaj uwagi wtyczki w IDE i asystenta AI na zmienionych liniach przed MR-em (SonarQube - Connected Mode w IntelliJ, SonarQube - praca z asystentem AI). Każde zgłoszenie wpływające na niezawodność lub bezpieczeństwo obniża ocenę poniżej A.
- Pokrycie: pisz testy jednostkowe do nowej logiki. Próg 40% dotyczy całego nowego kodu w okresie, więc kod bez testów obniża wynik całego zespołu.
- Duplikacja: nie kopiuj większych fragmentów kodu; wydziel wspólną część.
- Fałszywe alarmy: oznaczaj je na serwerze z uzasadnieniem po scaleniu, nie ukrywaj w kodzie.

Analiza raportu w interfejsie Sonara

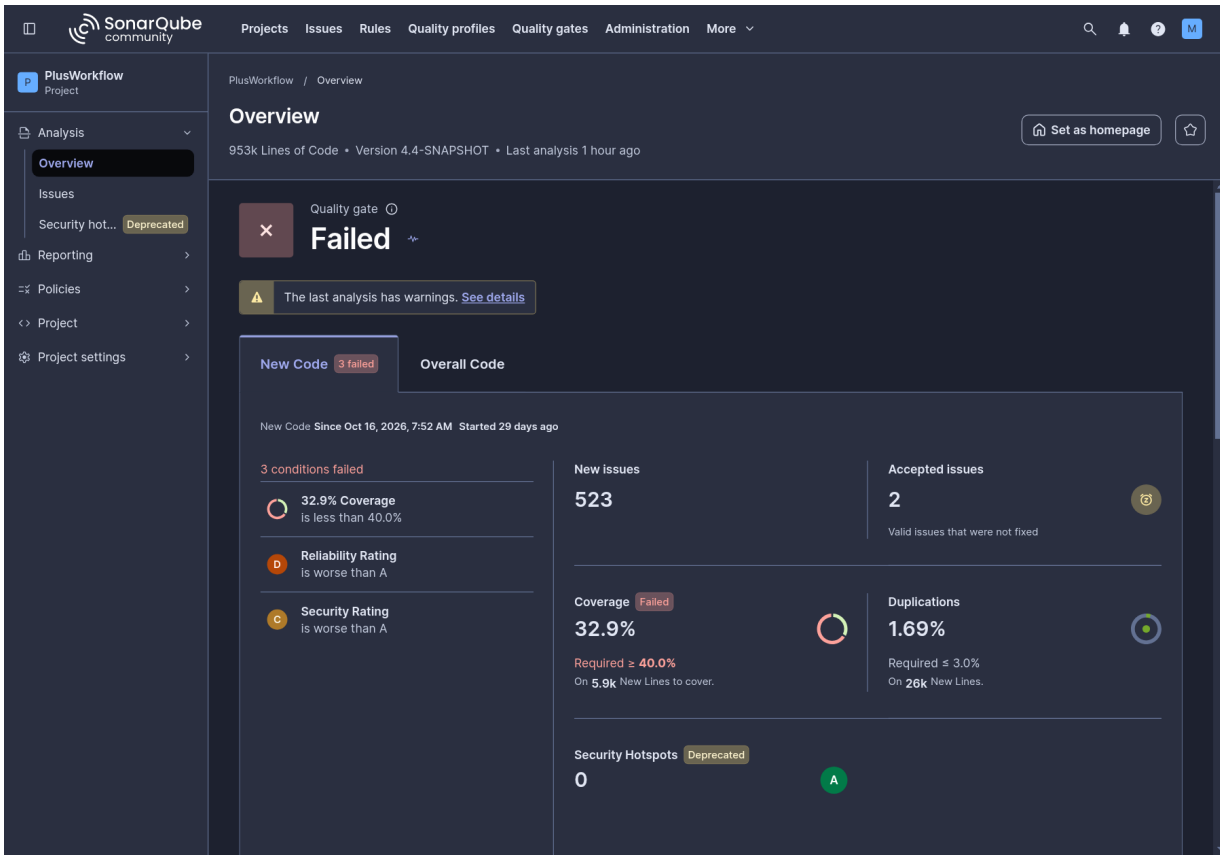
Raport projektu jest pod adresem https://sonar.suncode.local/dashboard?id=PlusWorkflow (wyma

ga zalogowania). Zrzuty pochodzą z 14.11.2026 — wartości na nich są przykładowe.

1. Overview — stan bramki

Strona startowa projektu. Zakładka New Code (domyślna) pokazuje to, co decyduje o bramce:

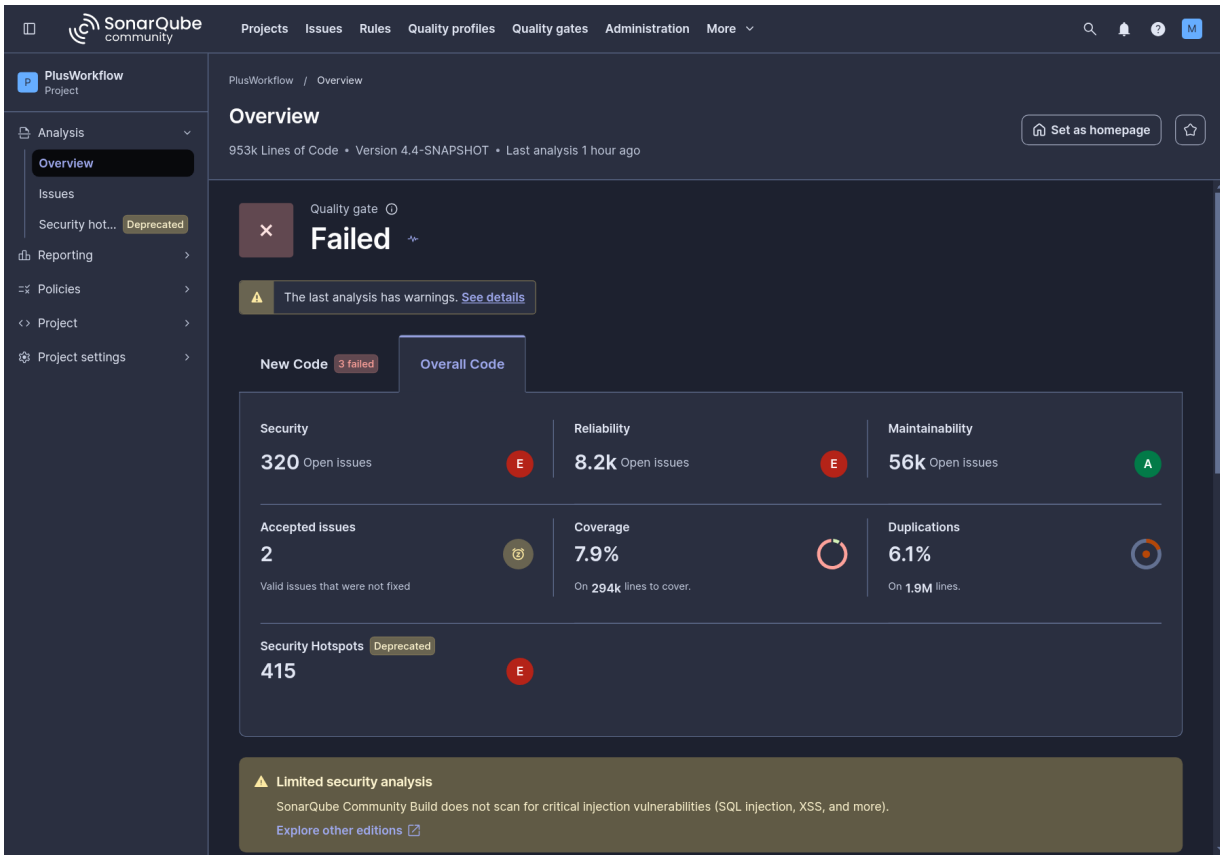
- Quality gate — wynik ostatniej analizy: Passed albo Failed.
- New Code Since … — od kiedy liczy się nowy kod (tu od 16.10.2026).
- N conditions failed — lista niespełnionych warunków z wartością i progiem. Każdy jest klikalny i prowadzi do szczegółów.
- Kafelki po prawej (New issues, Coverage, Duplications, Security Hotspots) — metryki nowego kodu. Kafelek z etykietą Failed to warunek, który nie przechodzi.



2. Overall Code — cały kod

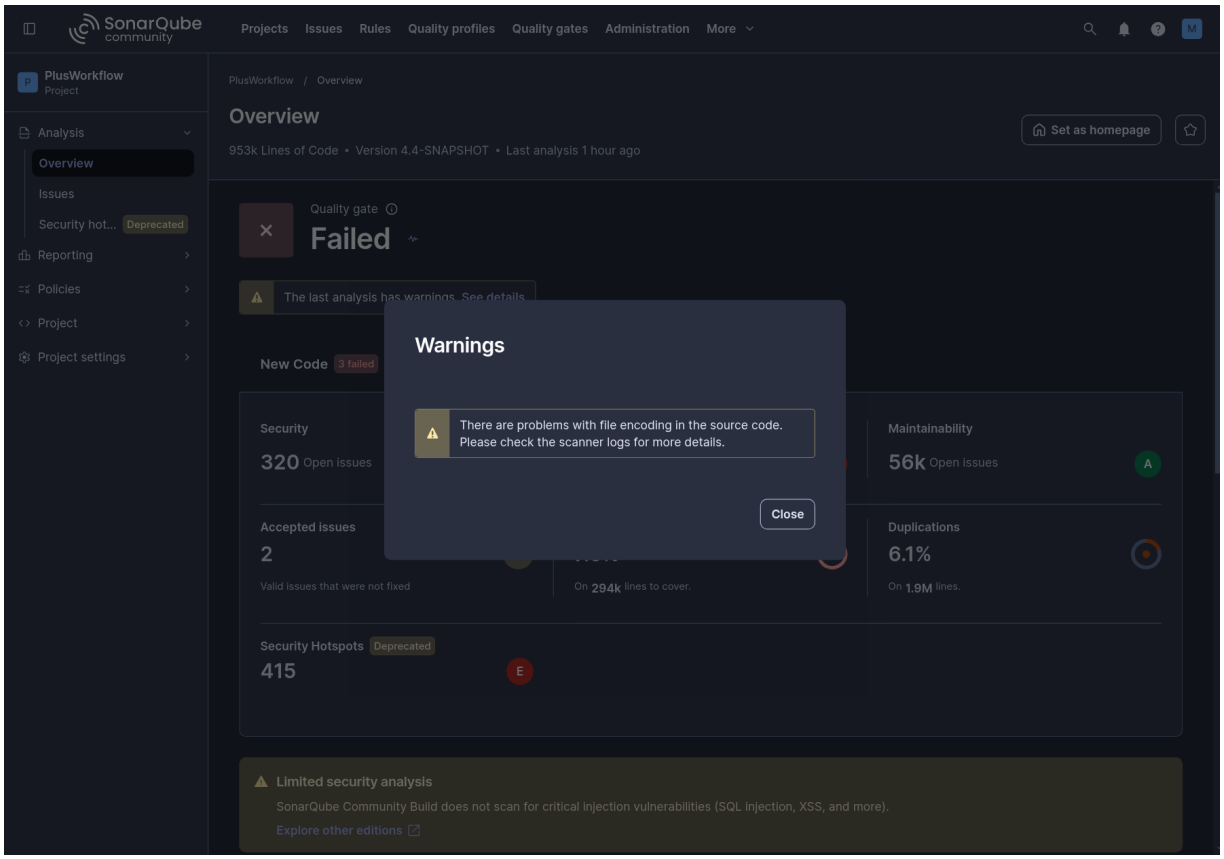
Zakładka Overall Code pokazuje stan całego projektu, razem z kodem sprzed okresu nowego kodu. Bramki nie blokuje — oceny E i niskie pokrycie wynikają tu z długu technicznego w starym kodzie. Przydaje się do planowania porządków, nie do oceny bieżących zmian.

Na dole strony jest stała informacja Limited security analysis: nasza wersja SonarQube Community Build nie wykrywa podatności typu wstrzyknięcie (SQL injection, XSS). Brak takich zgłoszeń w Sonarze nie oznacza, że ich w kodzie nie ma.



3. Ostrzeżenia analizy

Komunikat The last analysis has warnings → See details pokazuje problemy samej analizy, nie kodu. Obecnie jest to ostrzeżenie o kodowaniu znaków w części plików źródłowych — szczegóły są w logach skanera w zadaniu CI, które uruchamia analizę. Pliki, których skaner nie odczytał poprawnie, mogą mieć niepełne wyniki.

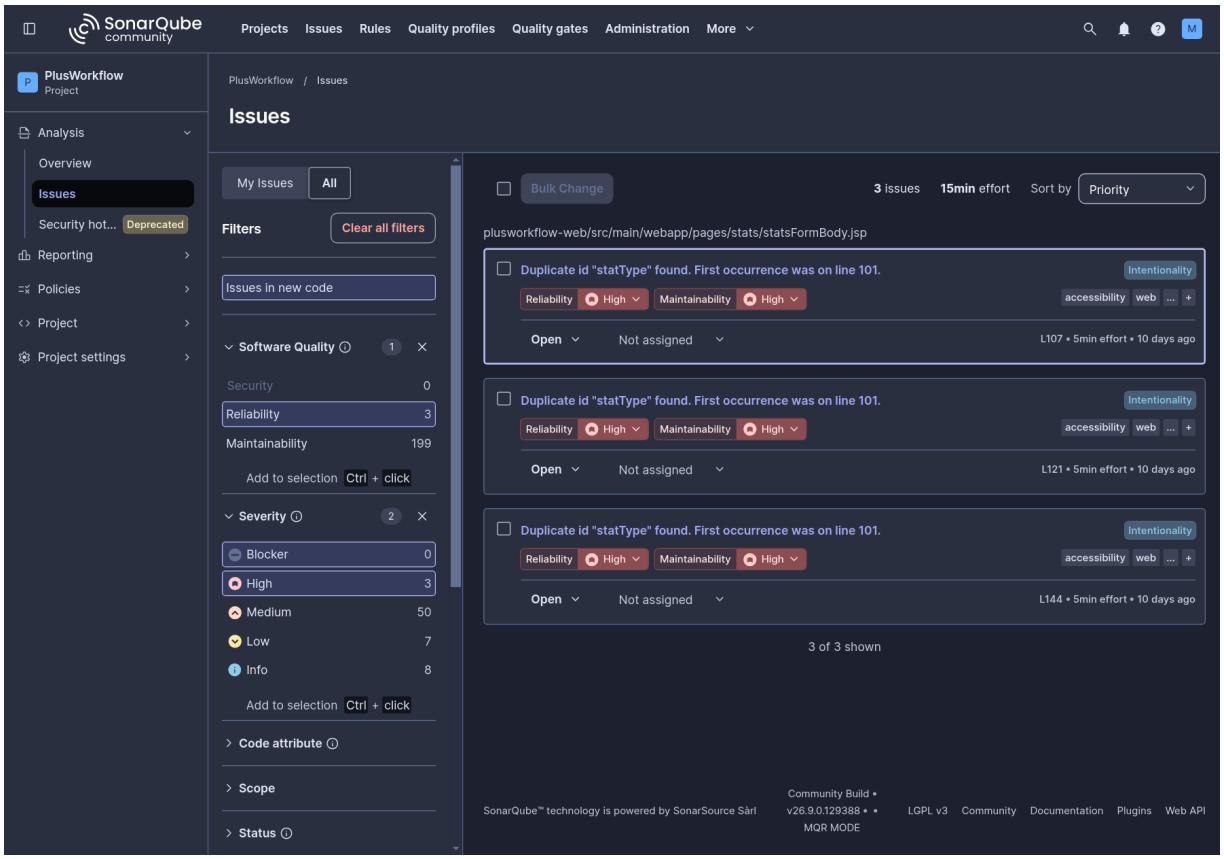


4. Issues — które zgłoszenia psują bramkę

Zakładka Issues z filtrami po lewej stronie. Żeby znaleźć zgłoszenia obniżające ocenę bramki, ustaw:

- New code — tylko nowy kod,
- Software Quality — Reliability albo Security (ocena, która nie przechodzi),
- Severity — od najwyższej wagi, bo ocena wynika z najpoważniejszego zgłoszenia (strona o ocenach A–E wyżej),
- Issue status — Open i Confirmed.

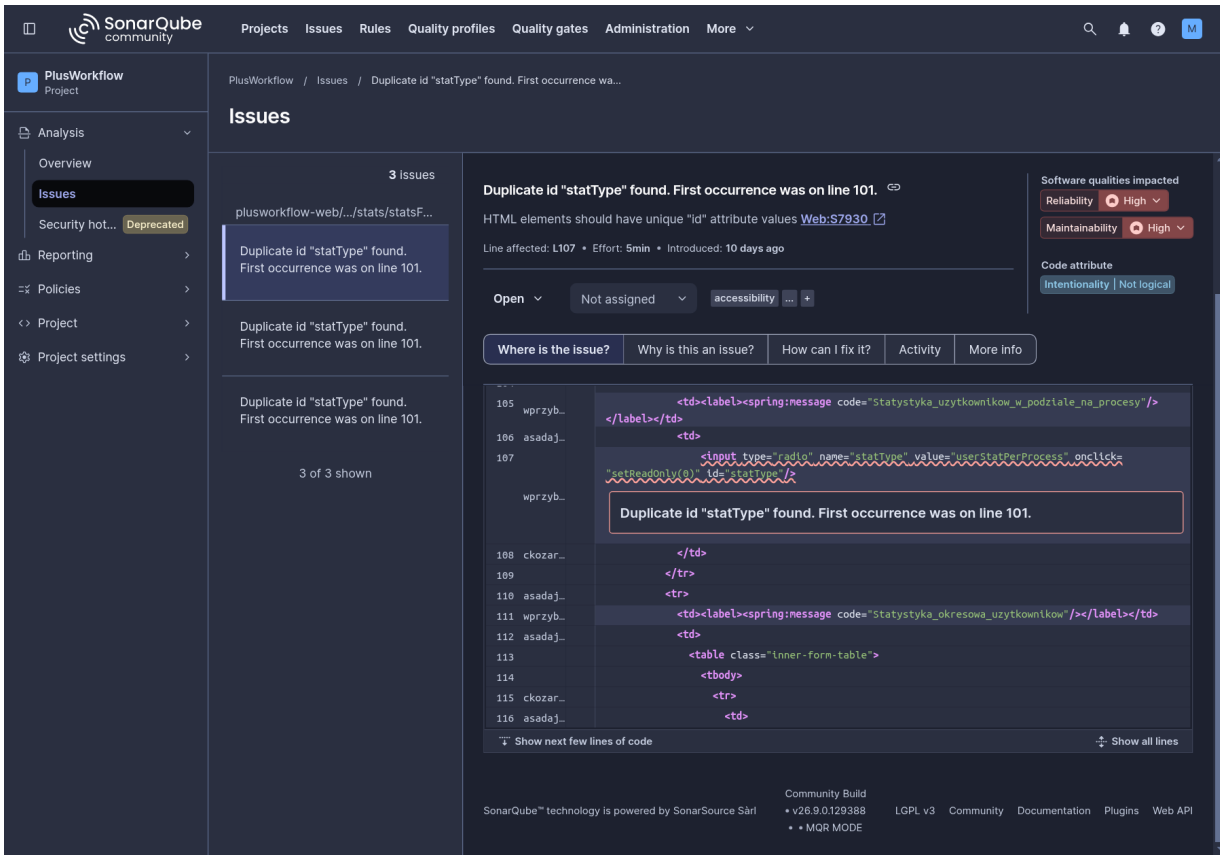
Lista jest pogrupowana według plików. Przydatne są też filtry Author (autor zmiany według git blame) i Rule. Link z ustawionymi filtrami można skopiować i wkleić do zgłoszenia w Jirze.



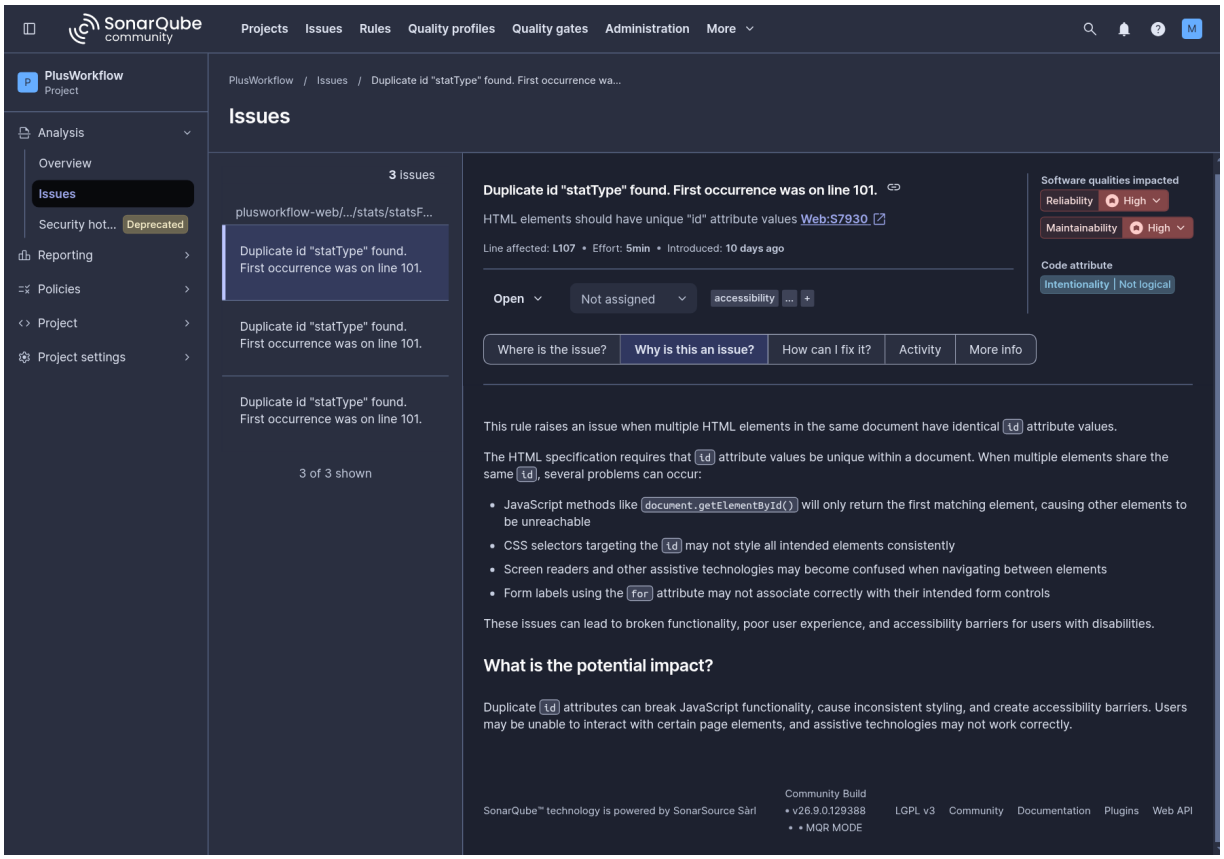
5. Szczegóły zgłoszenia

Kliknięcie zgłoszenia otwiera panel szczegółów:

- nazwa reguły z kluczem (np. `Web:S7930`), linia (Line affected), szacowany czas naprawy (Effort) i kiedy zgłoszenie powstało (Introduced),
- po prawej Software qualities impacted — na którą ocenę wpływa i z jaką wagą,
- zakładka Where is the issue? — fragment kodu z zaznaczonym miejscem.



Zakładka Why is this an issue? wyjaśnia regułę i jej skutki, a How can I fix it? pokazuje przykłady poprawki. Zanim uznasz zgłoszenie za fałszywy alarm, przeczytaj opis reguły — często wymienia sytuacje, w których kod jest zgodny z regułą.

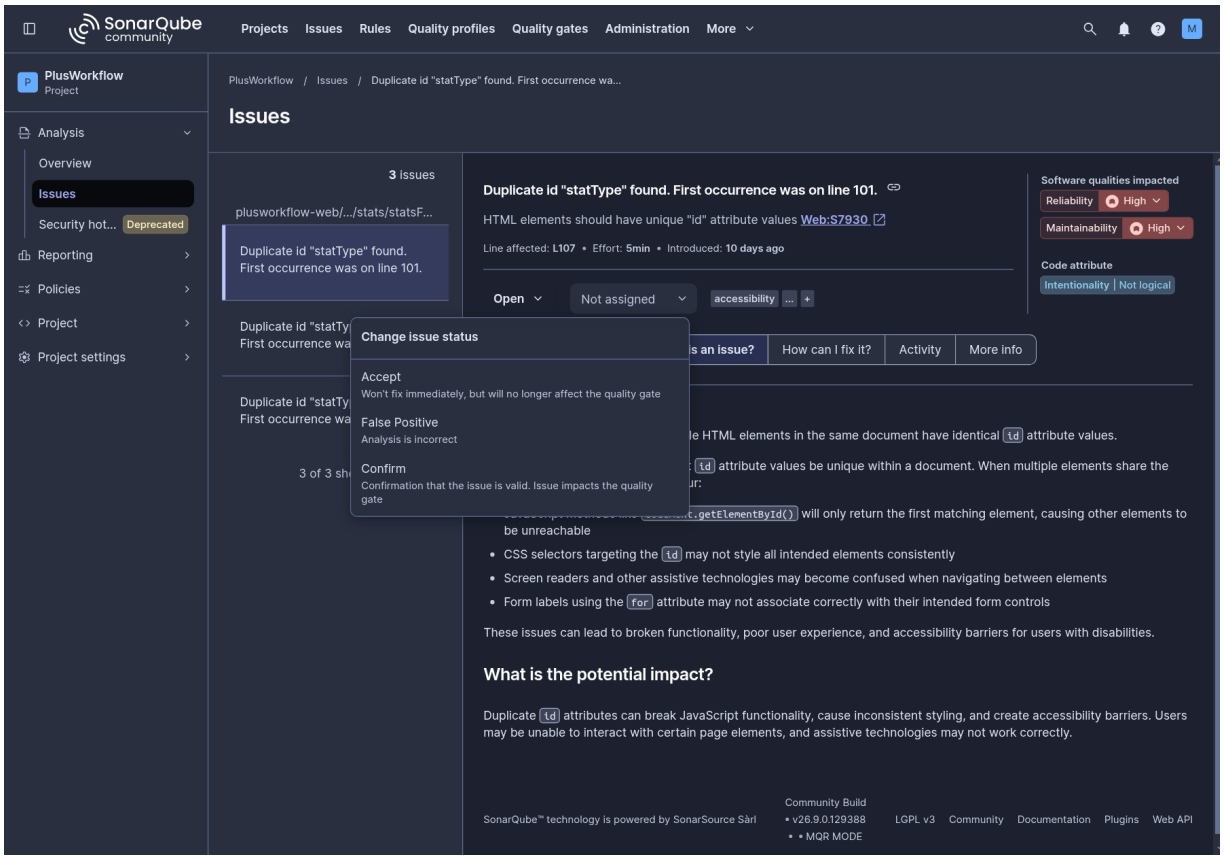


6. Zmiana statusu

Przycisk ze statusem (Open) otwiera menu:

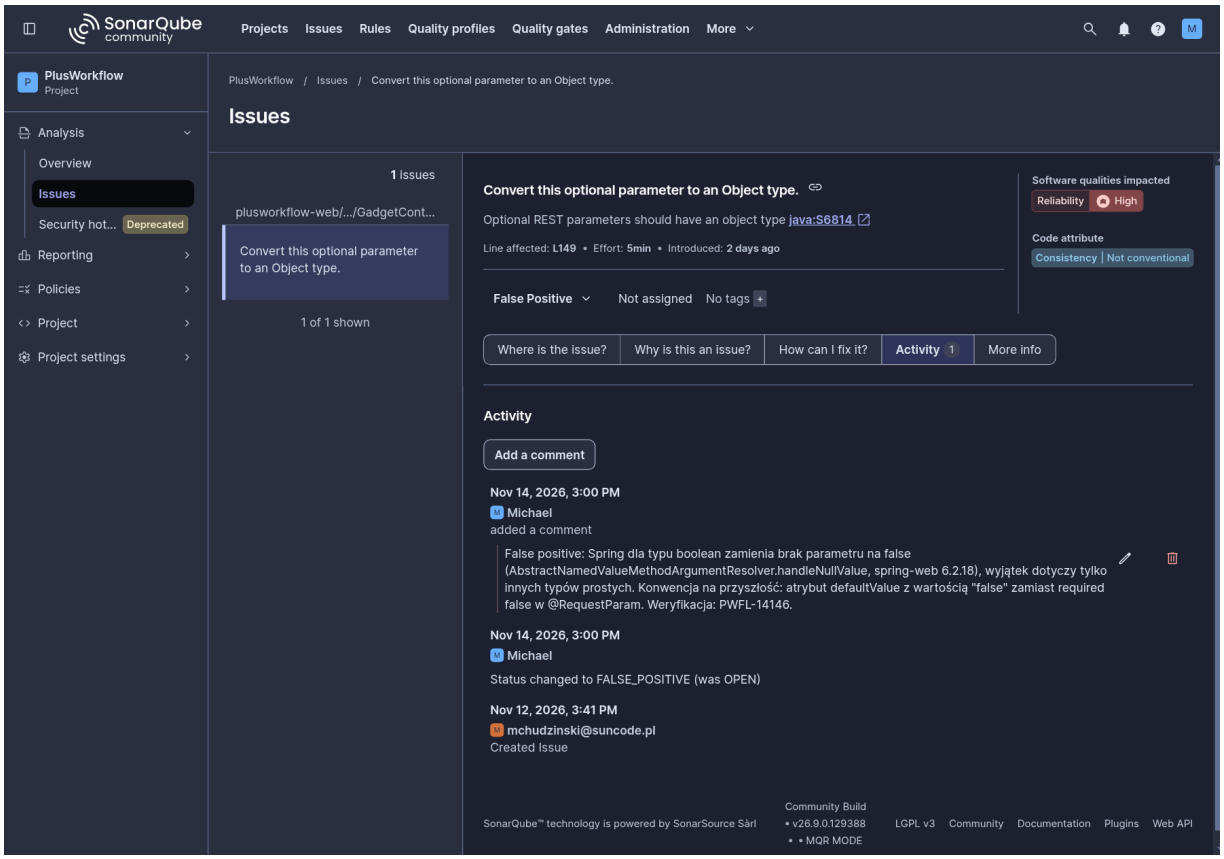
- Accept — problem jest prawdziwy, ale świadomie go nie poprawiamy. Przestaje wpływać na bramkę.
- False Positive — reguła się myli. Przestaje wpływać na bramkę.
- Confirm — problem potwierdzony, nadal wpływa na bramkę.

Kiedy wybrać Accept, a kiedy False Positive, opisuje strona SonarQube - fałszywe alarmy i oznaczanie zgłoszeń. Po zmianie statusu dodaj komentarz z uzasadnieniem.



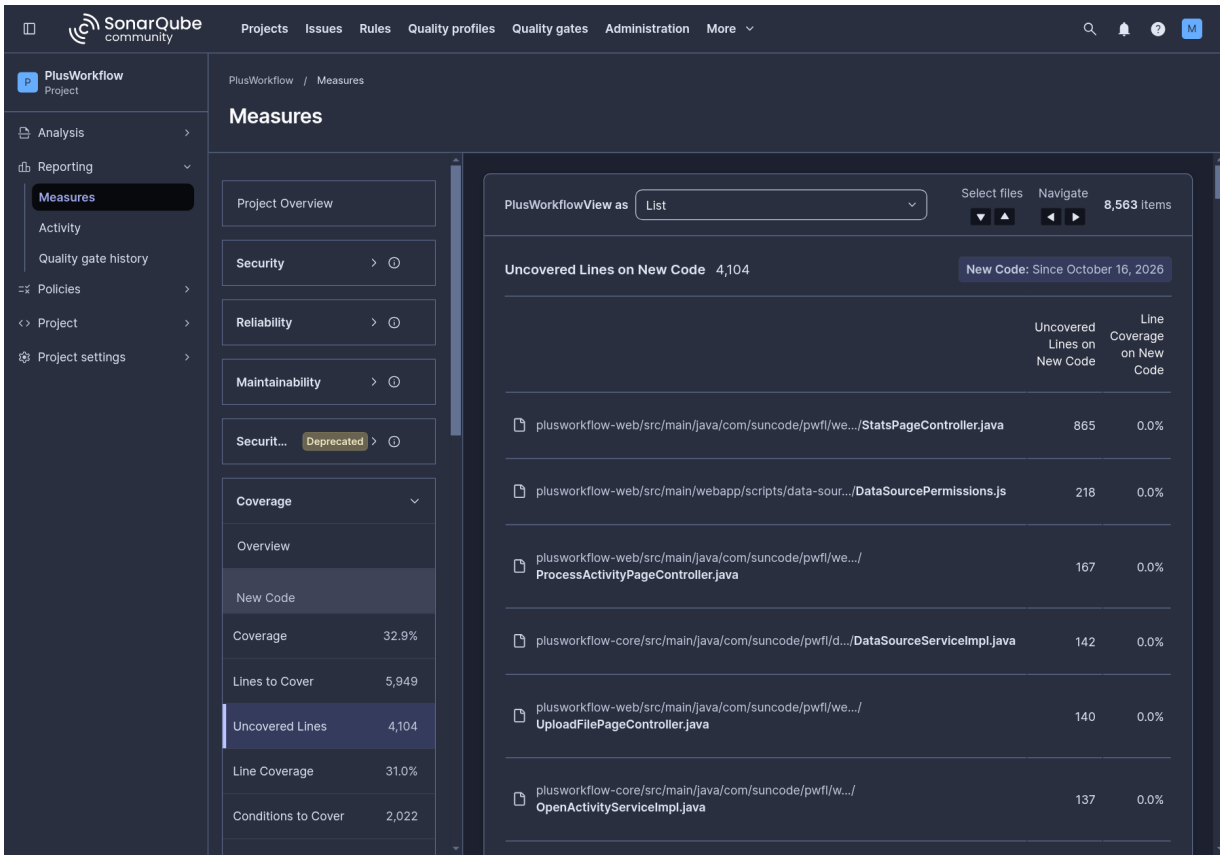
7. Activity zgłoszenia — kto i dlaczego zmienił status

Zakładka Activity w szczegółach zgłoszenia pokazuje historię zmian statusu i komentarze. Tu sprawdzisz, kto oznaczył zgłoszenie i z jakim uzasadnieniem. Komentarze obsługują ograniczony Markdown — unikaj znaku = na początku zdania i odwrotnych apostrofów, bo psują formatowanie.

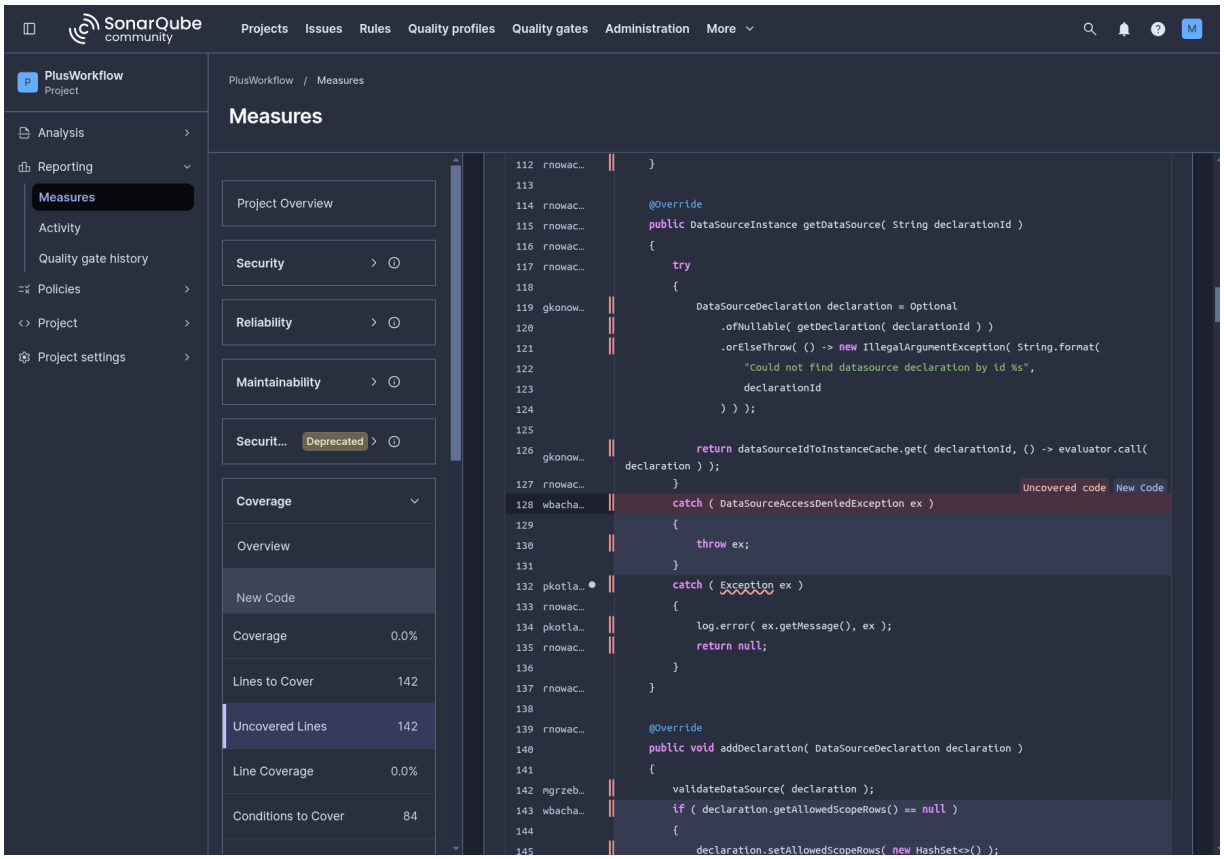


8. Pokrycie testami nowego kodu

Reporting → Measures → Coverage → Uncovered Lines z widokiem listy (lub kliknięcie warunku pokrycia na Overview). Pliki są posortowane według liczby niepokrytych linii w nowym kodzie, więc od góry widać, gdzie dopisanie testów najbardziej podniesie wynik.

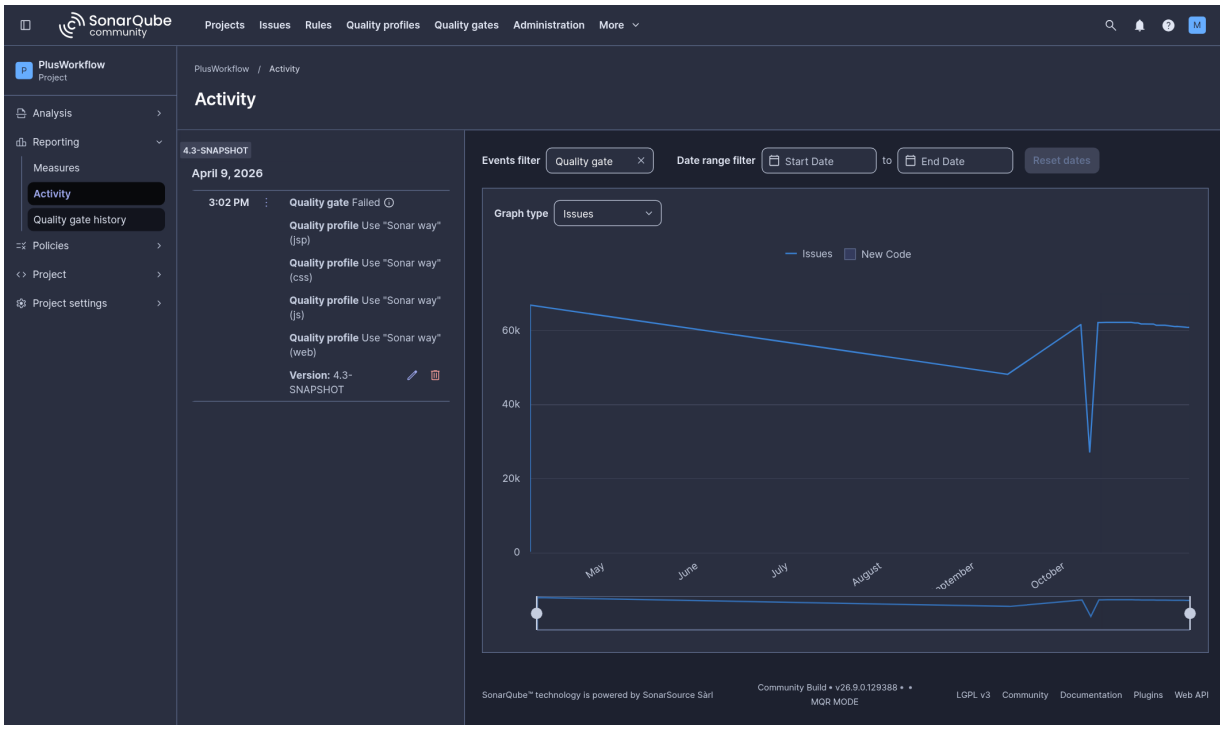


Po kliknięciu pliku Sonar pokazuje kod z oznaczeniami na marginesie: czerwony pasek to linia niepokryta testami, zielony — pokryta, żółty — warunek pokryty tylko częściowo. Linie należące do nowego kodu są wyróżnione tłem, a najechanie na znacznik pokazuje etykiety Uncovered code i New Code. Plik na zrzucie nie ma żadnych testów (pokrycie 0%), dlatego widać tylko czerwone znaczniki.



9. Activity projektu — historia analiz

Reporting → Activity pokazuje historię analiz z wykresem (liczba zgłoszeń, pokrycie, duplikacje) i zdarzeniami. Filtr Events filter → Quality gate zostawia tylko zmiany stanu bramki, co pomaga ustalić, od której analizy bramka nie przechodzi.



Strona Quality gate history w tym samym menu jest u nas pusta: działa tylko przy okresie nowego kodu ustawionym na „Previous version”, a my używamy konkretnej analizy bazowej.