

Analisi Qualità del Codice

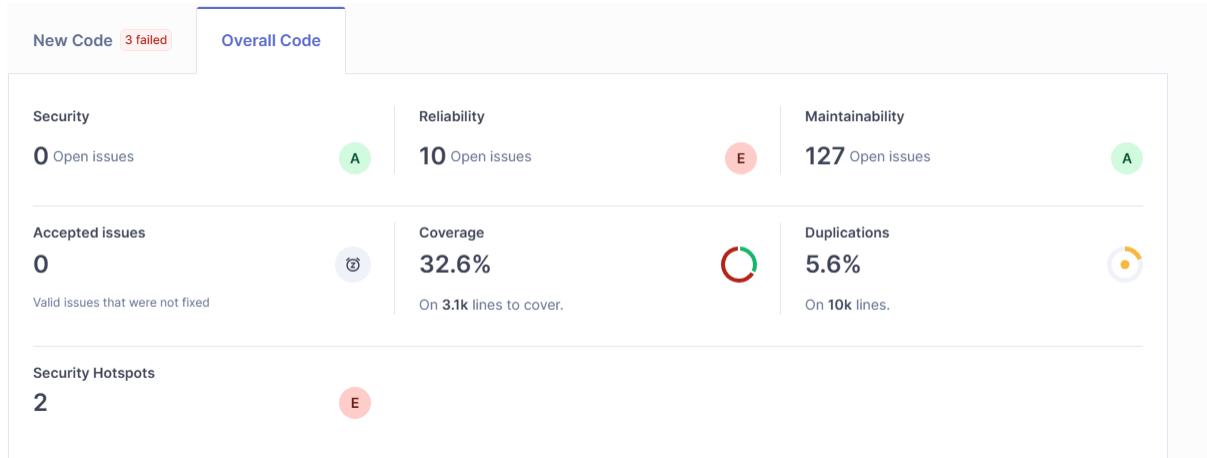
Progetto AIStudyBuddy

Scrum2Milly 2.0

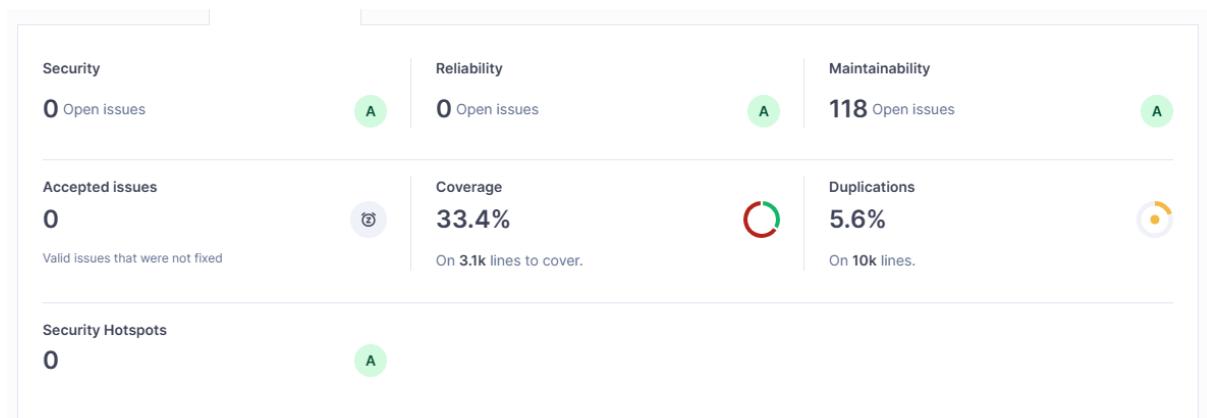
Mario Calipari, Andrea Celestino, Giovanni Giugovaz

1 SonarQube

Prima di effettuare il Refactoring:



Dopo il Refactoring:

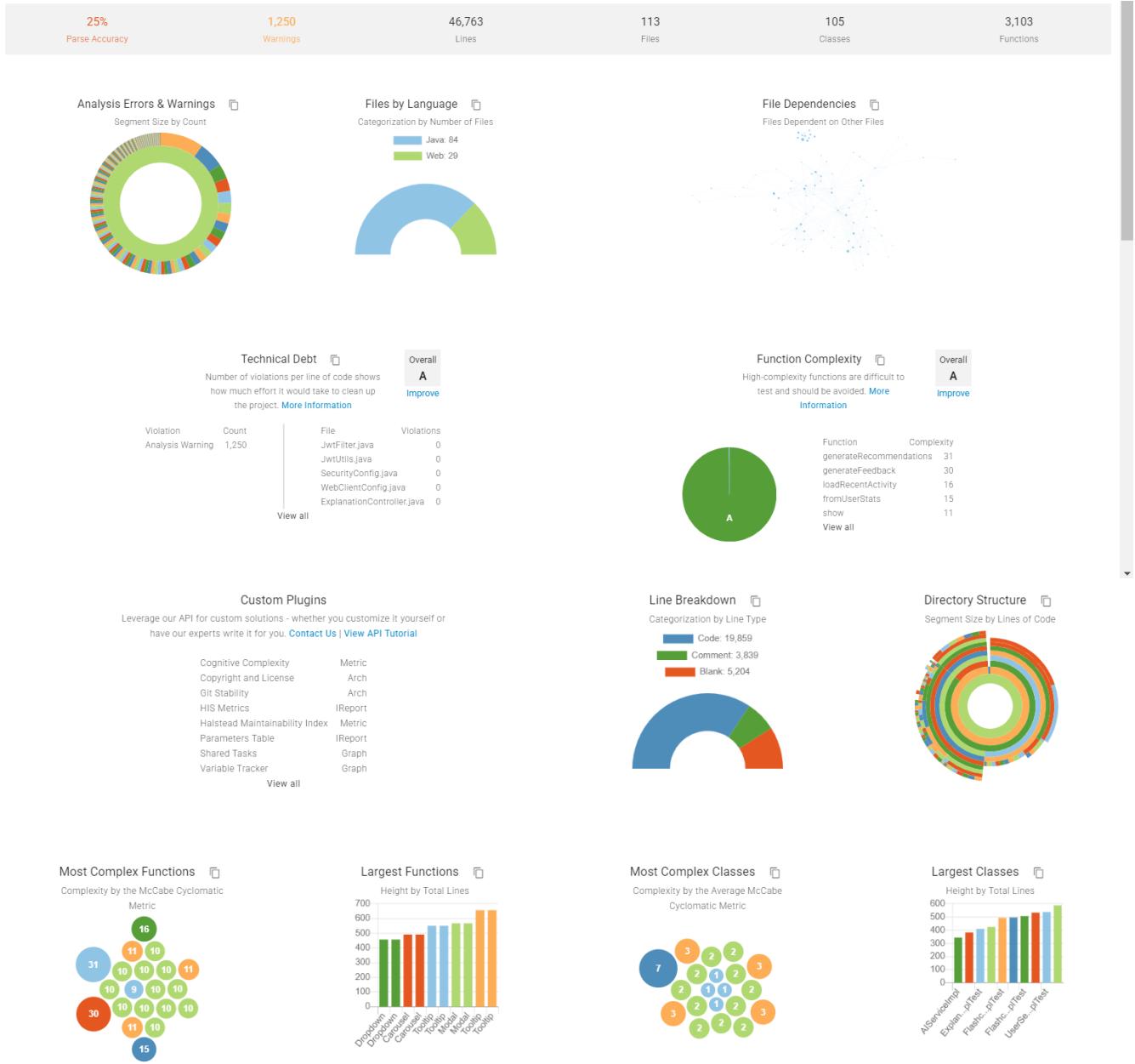


1.1 Problemi Risolti

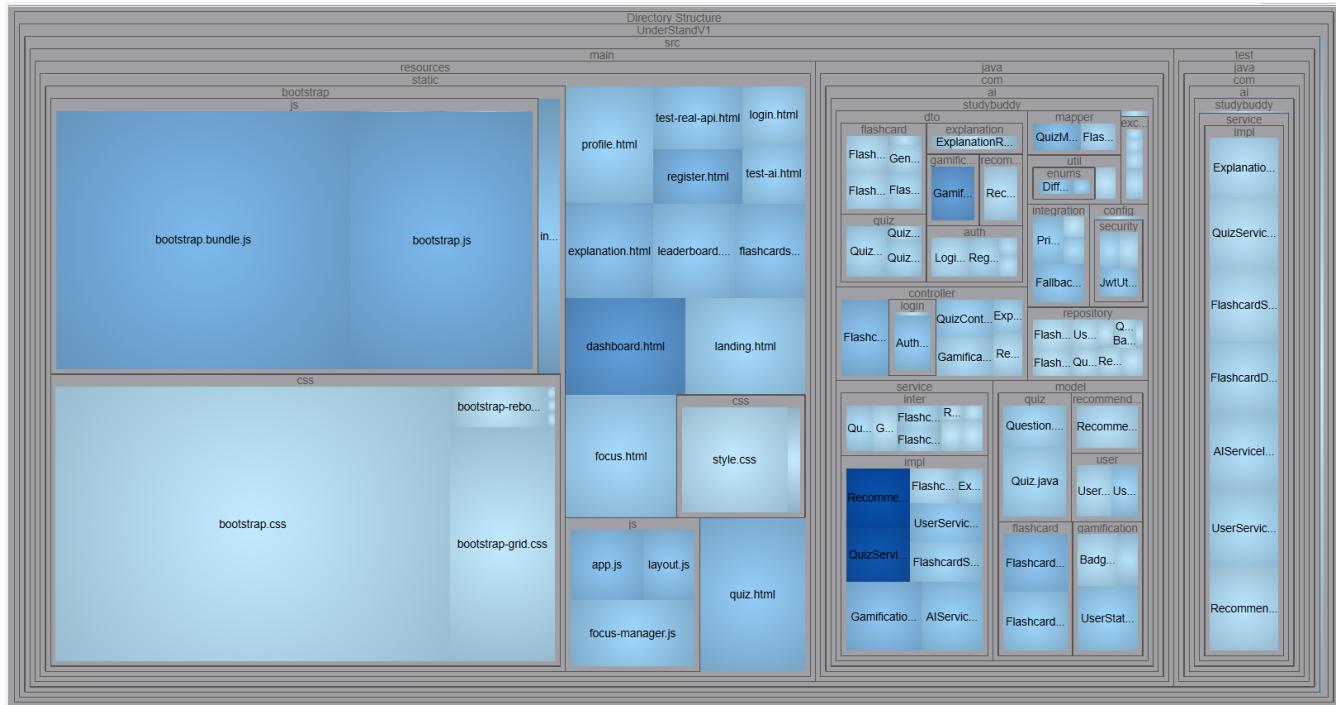
- **Gestione della sicurezza e protezione dati:** (SecurityConfig.java) L'analisi segnalava come errore la disattivazione della protezione CSRF (un sistema che evita l'invio di comandi non autorizzati). Questa protezione serve solo quando si usano i cookie. Il nostro progetto usa i Token JWT, un metodo di riconoscimento più moderno e sicuro che rende questo tipo di attacco impossibile.
- **Miglioramento della struttura dei componenti (Controller e Service):** Il codice è stato riorganizzato per cambiare il modo in cui le varie parti del software, da Field Injection a Constructor Injection. Inizialmente le dipendenze venivano inserite automaticamente in modo poco trasparente, rendendo il programma difficile da testare e meno stabile. Ora, ogni componente riceve tutto ciò di cui ha bisogno nel momento esatto in cui viene creato.
- **Ottimizzazione delle operazioni sul Database:** (es: GamificationServiceImpl.java) Sono state corrette alcune procedure legate al salvataggio e alla lettura dei dati (punteggi, badge e classifiche). È stato corretto un errore tecnico che impediva a Spring di gestire correttamente i salvataggi quando un metodo ne chiamava un altro internamente. Lettura Dati: Per tutte le operazioni che servono solo a visualizzare informazioni (come guardare una classifica), abbiamo attivato una modalità di "sola lettura". Questo rende il caricamento delle pagine più veloce, evita sprechi di memoria e impedisce che il programma vada in crash quando cerca di caricare dati complessi.

2 Understand

2.1 Analisi Generale



2.2 Metrics Treemap



2.3 Metriche Generali - AIStudyBuddy

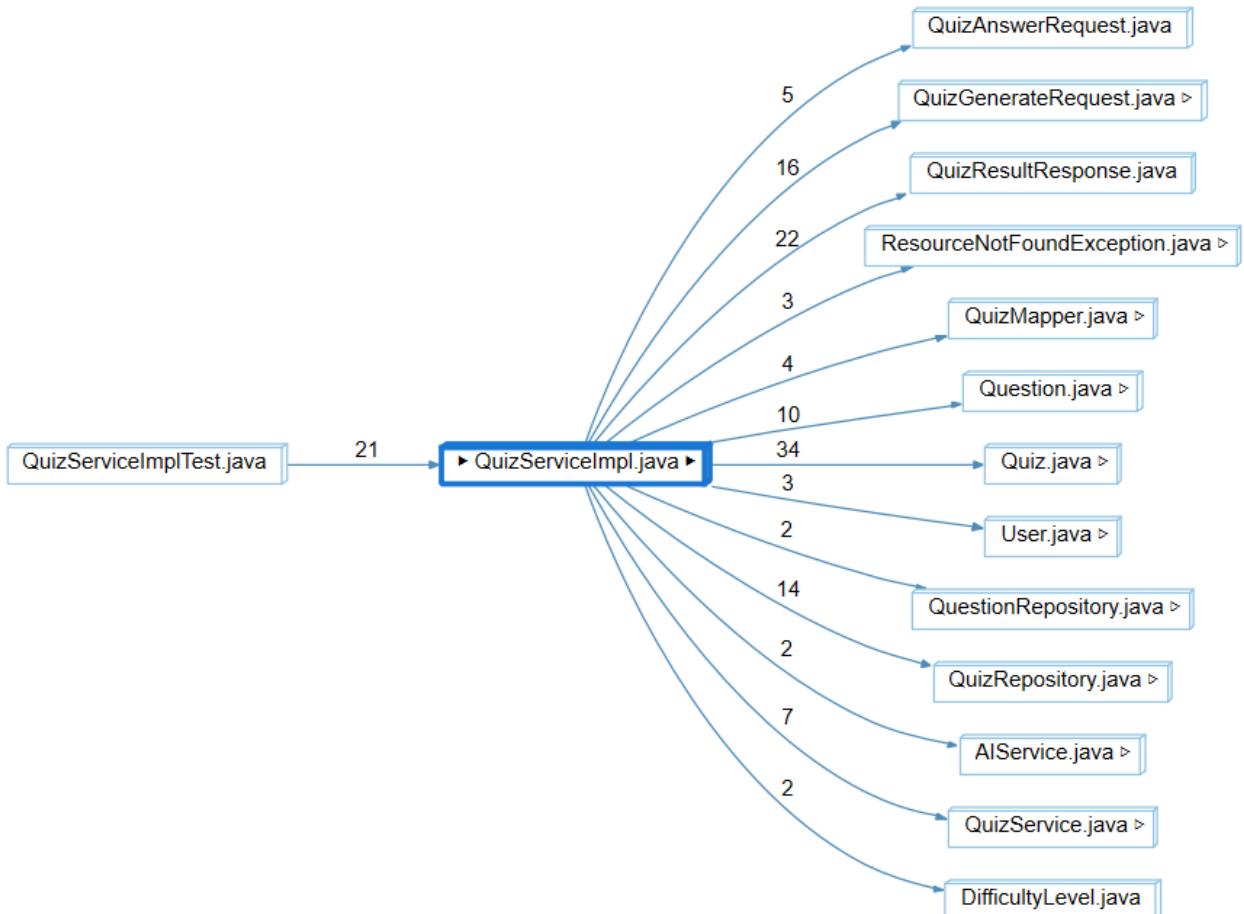
- **Java File:** 84
- **Web File:** 29
- **Blank Lines:** 5204
- **Classes:** 105
- **Code Lines:** 19859
- **CodeCheck Violation Density by Code Lines:** 0,00
- **CodeCheck Violation Density by Lines:** 0,00
- **CodeCheck Violation Types:** 0
- **CodeCheck Violation:** 0
- **Comment Lines:** 3839
- **Comment to Code Ratio:** 0,19
- **Declarative Statements:** 5665
- **Executable Statements:** 10208
- **Files:** 113
- **Functions:** 3103
- **Lines:** 46763

2.4 Individuazione Code smell classi più critiche

- **RecommendationServiceImpl**
 - **File totale:** 328 linee di codice
 - **MaxCyclomatic:** 31 (file-level) - *High Cyclomatic Complexity*
 - **AvgCyclomatic:** 5.13
 - **MaxNesting:** 4
 - **RatioCommentToCode:** 0.07 (7%) - *Low Comment Density*
 - **Numero metodi:** 8
 - **Analisi:** La classe presenta un'architettura squilibrata con il 68% della complessità totale concentrata in un singolo metodo (`generateRecommendations()`), identificabile come **God Method**.
- **QuizServiceImpl** (Situazione simile)
 - **File totale:** 300 linee di codice
 - **MaxCyclomatic:** 30 (file-level) - *High Cyclomatic Complexity*
 - **AvgCyclomatic:** 2.86
 - **MaxNesting:** 3
 - **RatioCommentToCode:** 0.01 (1%) - *Low Comment Density*
 - **Numero metodi:** 20
 - **Analisi:** Il metodo problematico è `generateFeedback()`, identificabile come **God Method**.

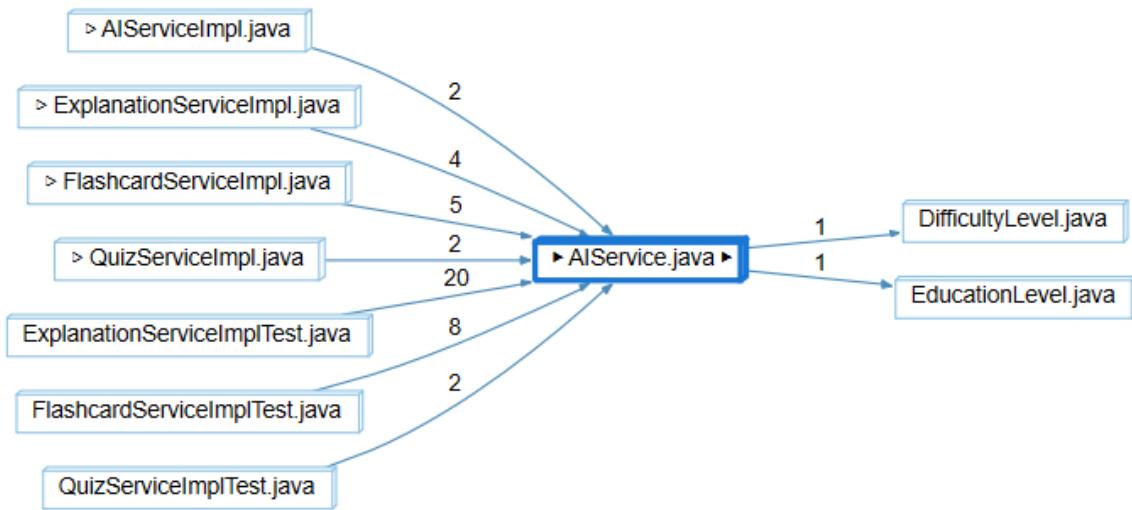
2.5 Antipattern Strutturali individuati

- External Hub:



`QuizServiceImpl` è la classe che presenta il maggior numero di dipendenze. Presenta dipendenze verso 13 classi diverse. Il numero più alto di riferimenti è 34, verso la classe appartenente al package model, “`Quiz.java`”. Rispetto al numero medio di dipendenze delle classi del progetto può essere considerato un potenziale “*External Hub*”.

- External Butterfly:



`AIService` (interfaccia) ha 7 classi dipendenti con un totale di 43 riferimenti. I dipendenti sono distribuiti in package diversi e la modifica dell’interfaccia comporta un impatto globale. Di conseguenza, `AIService` rappresenta una potenziale “*External Butterfly*”.