Final Report Document

Versione 1.0

CodeSmile

Team Members:
Matteo Ercolino matr. 0522501462
Simone Silvestri matr. 0522501419

Repository: GitHub link

Anno Accademico 2024/2025

Indice

1 Introduzione

Il presente **Final Report Document** sintetizza l'intero ciclo di vita del progetto *CodeSmile*, dalla concezione iniziale alla consegna finale a valle delle tre *Change Request* approvate. Obiettivi principali del documento:

- fornire una vista organica dell'evoluzione del sistema e delle sue architetture;
- illustrare le modifiche introdotte, l'analisi di impatto svolta e gli esiti del testing;
- dimostrare il rispetto dei requisiti di manutenibilità, testabilità e qualità del software.

Tutti i contenuti qui riportati derivano dai documenti prodotti durante il progetto (*Initial Report*, *Change Requests Document*, *Impact Analysis*, *Master Test Plan*, *Pre-Modifications System Testing* e *Post-Modification & Regression Testing*).

1.1 Descrizione sintetica del Sistema

CodeSmile è uno strumento di *static code analysis* specializzato nell'individuazione di **ML-specific code smells** nei progetti Python. Il sistema è articolato in:

Core CLI/GUI per l'analisi locale di file o interi repository ;

Web App basata su architettura *gateway–services* (AI Analysis, Static Analysis, Report);

Rule Engine con 16 regole, suddivise in *API-specific* e *generic smells*;

Il design a pacchetti autonomi (`cli`, `components`, `detection_rules`, ...) riduce l'accoppiamento e facilita la manutenzione, condizione necessaria per integrare agilmente le evoluzioni descritte nei capitoli successivi.

2 Change Requests implementate

2.1 CR-01 — Dashboard interattiva nella Web App (perfettiva)

Introduzione di una *dashboard* React con grafici dinamici che mostra:

- numero e tipologia di code smell nel tempo;
- trend per progetto e file;
- filtri avanzati per severità e categoria.

Stato: **Approvata** e rilasciata.

2.2 CR-02 — Integrazione con CI/CD (adattiva)

Aggiunta di una `GitHub Actions` pipeline che:

1. esegue CodeSmile su ogni `push`/`PR`;
2. crea automaticamente issue con i risultati.

Stato: **Approvata** e rilasciata.

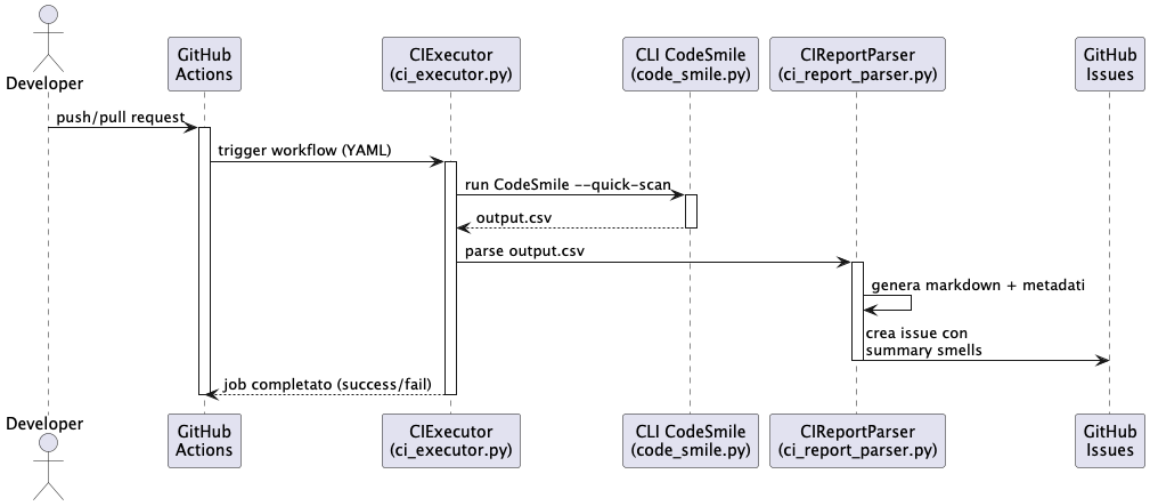

Figura 2.1: Sequence Diagram - CI

2.3 CR-03 — Modalità “Quick Scan” (perfettiva)

Nuova opzione `--quick-scan` che analizza soltanto i file modificati negli ultimi N commit, riducendo i tempi d’esecuzione e/o esecuzioni ripetitive.
Stato: **Approvata** e rilasciata.

3 Analisi di Impatto

3.1 Metodologia

Per ciascuna CR è stata applicata l'**Impact Analysis** basata su *Call Graph* e *Reachability Matrix*, con la classificazione degli insiemi: *SIS*, *CIS*, *AIS*, *FPIS*, *DIS*. Le metriche di *precision* e *recall* hanno misurato l'accuratezza delle previsioni.

CR	**Precision**	**Recall**	**FPIS**	**DIS**
CR-01	0.85	0.46	gateway_-main.py	`components/ChartContainer.tsx`, `components/PDFDownloadButton.tsx`, `components/ProjectSidebar.tsx`, `components/SmellDensityCard.tsx`, `hooks/useReportData.ts`, `components/Modal.tsx`, `utils/dataFormatters.ts`
CR-02	–	–	–	–
CR-03	1.0	1.0	–	–

4 Testing

4.1 Master Test Plan

Il **Master Test Plan** ha definito quattro livelli di verifica: Unità, Integrazione, Sistema ed End-to-End (webapp). Obiettivo di *branch coverage*: $\geq 80\%$.

4.2 Pre-Modifications System Testing

Prima delle CR sono stati eseguiti 21 test di sistema secondo il *Category Partition Method*. Tutti gli oracoli sono stati soddisfatti, stabilendo la *baseline* di regressione.

4.3 Post-Modification & Regression Testing

- **Unit Test**: 166/166 superati – nuove classi coperte al 100 %.
- **Integration Test**: 8/8 superati, incluse interfacce Gateway→Report e CIExecutor.
- **System Test**: scenari baseline rieseguiti con successo; aggiunti 4 test Quick Scan.
- **E2E WebApp**: 20/20 scenari Cypress superati, dashboard renderizzata correttamente.
- **Coverage**: branch coverage globale 87 %, con picchi >90 % sui package `components`, `detection_rules`.

5 Conclusioni

Il progetto *CodeSmile* ha dimostrato:

1. un'architettura modulare capace di assorbire evoluzioni (CR-01, 02, 03) senza regressioni;
2. un processo di *Impact Analysis* affidabile (precision/recall $\geq$1.0 nel caso più critico);
3. una pipeline di testing solida che mantiene l'*quality gate* sopra l'80 % di coverage;
4. benefici concreti per gli utenti: tempi di analisi ridotti (Quick Scan) e feedback continuo (CI/CD).

Tutti gli obiettivi fissati all'inizio del processo sono stati pienamente raggiunti. Il sistema è ora pronto per il rilascio pubblico e per future estensioni (*e.g.* nuovi ML smells o supporto a linguaggi addizionali).