
UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA

Esame di Ingegneria, Gestione ed Evoluzione del Software

PROGETTO CODESMILE

# Change Requests

TEAM MEMBERS

Dario Mazza - 0522501553

Nicolò Delogu - 0522501556

REPOSITORY

`https://github.com/xDaryamo/smell_ai`

VERSIONE

1.0

Anno Accademico 2024-2025

# 1 Modifica Architetturale: Transizione al Design Orientato agli Oggetti

- **Priorità:** Alta
- **Stato Attuale:** Codice procedurale per il rilevamento degli smells
- **Proposta di Modifica:** Rifattorizzare in codice orientato agli oggetti; implementare classi che incapsulano le funzionalità attualmente realizzate.
- **Requisiti:**
  - **R1:** Supportare la suddivisione delle funzionalità in unità distinte e riutilizzabili
  - **R2:** Organizzare il sistema in modo che le modifiche a una parte del codice non abbiano impatti imprevisti su altre parti
  - **R3:** Assicurare che ogni componente si occupi esclusivamente di una responsabilità
- **Giustificazione:** Migliora la modularità e la manutenibilità del codice, semplifica l'aggiunta di nuovi smells, migliora il testing grazie a funzionalità incapsulate

# 2 Interfaccia Grafica: Web-App

- **Priorità:** Media
- **Stato Attuale:** Interfaccia grafica desktop
- **Proposta di Modifica:** Implementare una Web-App per consentire interazioni via web
- **Requisiti:**
  - **R4:** Consentire agli utenti di caricare file da analizzare tramite un'interfaccia web
  - **R5:** Permettere agli utenti di configurare le opzioni di analisi tramite l'interfaccia
  - **R6:** Visualizzare i risultati dell'analisi in modo chiaro e accessibile
  - **R7:** Non mantenere uno storico degli utenti o delle analisi effettuate da questi ultimi
- **Giustificazione:** Migliorare l'esperienza utente e l'accessibilità. Consentire agli utenti di eseguire analisi senza conoscenze della riga di comando. La decisione di non mantenere uno storico è motivata dalla necessità di semplificare il sistema, ridurre i costi di storage, garantire un elevato livello di privacy e minimizzare i rischi legati alla gestione dei dati utente

# 3 Uso di LLM per un Injector di smell nei codebase

- **Priorità:** Alta
- **Stato Attuale:** Assenza di un metodo automatizzato per generare dataset con smell
- **Proposta di Modifica:** Utilizzare Modelli di Linguaggio di Grandi Dimensioni (LLM) per creare dataset con smell iniettati, utilizzare modelli come GPT di OpenAI o modelli open-source come LLAMA
- **Requisiti:**
  - **R8:** Consentire la generazione di varianti di code snippets iniettando smells specifici.
- **Giustificazione:** Generare dati di addestramento in assenza di dataset etichettati nel mondo reale

# 4 Addestramento di un Modello di IA per il rilevamento degli ML-specific smells

- **Priorità:** Alta
- **Stato Attuale:** Assenza di un modello di IA per il rilevamento degli smells
- **Proposta di Modifica:** Addestrare un modello di IA per il rilevamento automatico degli smells, utilizzare il dataset generato dagli LLM con smell iniettati e valutare con metriche come accuratezza, precisione, richiamo e F1 score
- **Requisiti:**
  - **R9:** Creare un modulo in grado di apprendere a identificare ML-specific smells nel codice
  - **R10:** Consentire la valutazione delle capacità del modulo attraverso metriche standard
  - **R11:** Permettere l'utilizzo del modulo per analizzare code snippets e classificare gli smells automaticamente
- **Giustificazione:** Studiare da un punto di vista di fattibilità e di performance, l'applicazione di LLM rispetto ai task di identificazione e di classificazione di ML-specific code smells.

# 5 Funzionalità di Refactoring Automatico

- **Priorità:** Bassa
- **Stato Attuale:** Non è presente una funzionalità che supporti il refactoring automatico del codice per eliminare gli smells individuati
- **Proposta di Modifica:** Implementare una funzionalità per eseguire il refactoring automatico del codice al fine di correggere gli smells rilevati
- **Requisiti:**
  - **R12:** Identificare gli smells presenti nel codice e suggerire refactoring appropriati
  - **R13:** Consentire all'utente di selezionare quali refactoring applicare tra quelli suggeriti
  - **R14:** Applicare automaticamente le modifiche al codice mantenendo la correttezza e la coerenza
  - **R15:** Fornire un meccanismo per il rollback delle modifiche apportate in caso di necessità
- **Giustificazione:** Migliorare la qualità del codice riducendo la presenza di smells in modo proattivo. Risparmiare tempo e risorse degli sviluppatori, che possono concentrarsi su attività più critiche. Garantire coerenza nelle modifiche grazie a refactoring automatizzati. Offrire agli utenti maggiore controllo sul processo di refactoring, mantenendo comunque la possibilità di annullare modifiche indesiderate.