UNIVERSITÀ DEGLI STUDI
DI SALERNO

Analisi Di Manutenzione

CodeFace4Smells

Versione	1.0
Data	27/10/2025
Destinatario	Prof. Andrea De Lucia
Presentato da	Gabriele Santoro Pasquale Sorrentino

Composizione Gruppo	
Gabriele Santoro	0522502066
Pasquale Sorrentino	0522501954

Cronologia Revisioni

Data	Versione	Descrizione	Autori
22/09/2025	0.1	Inizio stesura del documento	Gabriele Santoro Pasquale Sorrentino
22/09/2025	0.2	Suddivisione del documento in capitoli	Gabriele Santoro Pasquale Sorrentino
23/09/2025	0.3	Stesura della panoramica del sistema attuale	Gabriele Santoro Pasquale Sorrentino
24/09/2025	0.4	Stesura dei requisiti funzionali del sistema attuale	Gabriele Santoro Pasquale Sorrentino
25/09/2025	0.5	Stesura dell'attività di manutenzione	Gabriele Santoro Pasquale Sorrentino
30/09/2025	0.6	Stesura delle *change requests*	Gabriele Santoro Pasquale Sorrentino
17/10/2025	0.7	Stesura del sistema modificato	Gabriele Santoro Pasquale Sorrentino
23/10/2025	1.0	Revisione e convalida del documento	Gabriele Santoro Pasquale Sorrentino

Indice

1 Scopo del documento..... 5
2 Panoramica del sistema attuale..... 5
2.1 Reverse Engineering..... 5
2.2 Descrizione del sistema attuale..... 5
2.3 Attori del sistema..... 6
2.4 Requisiti Funzionali..... 6
2.4.1 Gestione Containerizzazione e Deploy..... 6
2.4.2 Analisi ed Elaborazione Dati..... 7
2.4.3 Output e Visualizzazione..... 7
2.4.4 Testing..... 7
2.5 Architettura del Sistema..... 9
2.6 Database..... 9
2.6.1 Modello EER..... 9
2.6.2 Schema Relazionale..... 9
2.7 Testing..... 11
3 Attività di Manutenzione..... 11
3.1 CR_01..... 12
3.1.1 Identificazione analisi e valutazione..... 12
3.1.2 Impact Analysis..... 14
3.1.2.1 Start Impact Set..... 14
3.1.2.2 Candidate Impact Set..... 14
3.1.2.3 Discovered Impact Set..... 14
3.1.2.4 False Positive Impact Set..... 15
3.1.2.5 Actual Impact Set..... 15
3.1.3 Metriche di valutazione..... 16
3.2 CR_02..... 16
3.2.1 Identificazione analisi e valutazione..... 16
3.2.2 Impact Analysis..... 18
3.2.2.1 Start Impact Set..... 18
3.2.2.2 Candidate Impact Set..... 18
3.2.2.3 Discovered Impact Set..... 20
3.2.2.4 False Positive Impact Set..... 20
3.2.2.5 Actual Impact Set..... 20
3.2.3 Metriche di valutazione..... 22
3.3 CR_03..... 23
3.3.1 Identificazione analisi e valutazione..... 23
3.3.2 Impact Analysis..... 24
3.3.2.1 Start Impact Set..... 24

3.3.2.2 Candidate Impact Set..... 25
3.3.2.3 Discovered Impact Set..... 25
3.3.2.4 False Positive Impact Set..... 25
3.3.2.5 Actual Impact Set..... 26
3.3.3 Metriche di valutazione..... 27
3.4 Obiettivi del Refactoring..... 27
3.5 Benefici Attesi..... 28
3.6 Strategie di Debugging..... 29
4 Sistema Modificato..... 29
4.1 Descrizione del sistema attuale..... 29
4.2 Attori del Sistema..... 30
4.3 Requisiti funzionali..... 30
4.3.1 Gestione Containerizzazione e Deploy..... 30
4.3.2 Analisi ed Elaborazione Dati..... 31
4.3.3 Output e Visualizzazione..... 31
4.3.4 Testing..... 31
4.4 Architettura del Sistema..... 33
4.5 Database..... 33
4.5.1 Modello EER..... 33
4.5.2 Schema Relazionale..... 33

1 Scopo del documento

Il presente documento ha l'obiettivo di descrivere le attività di manutenzione e ripristino effettuate sul software ***CodeFace4Smells***. Sarà inoltre presentata l'*impact analysis*, finalizzata a individuare le componenti del sistema coinvolte dalle change request.

2 Panoramica del sistema attuale

Il progetto *Codeface4Smells* si basa su *CodeFace*, uno strumento impiegato per analizzare l'evoluzione e lo sviluppo software. Tuttavia, il sistema originale risultava di difficile comprensione a causa della documentazione limitata. Per questo motivo, il primo obiettivo è stato svolgere un'attività di reverse engineering, volta a ricostruire il funzionamento del sistema e a comprendere il ruolo dei suoi diversi componenti.

2.1 Reverse Engineering

L'attività di reverse engineering ha permesso di ottenere una comprensione più approfondita del sistema, chiarendo lo scopo di ciascun modulo e consentendo la ricostruzione dei requisiti funzionali insieme alla mappatura delle relazioni tra i diversi componenti.
L'approccio seguito si è basato sul modello goal/models/tools, articolato come segue:

- **Goal**: In questa fase l'obiettivo era comprendere il sistema, producendo una documentazione di alto livello contenente i requisiti funzionali e le relazioni tra gli artefatti software.

- **Models**: Sono stati sviluppati due modelli principali:
    - Un elenco dei requisiti funzionali con relativa descrizione.
    - Una matrice di tracciabilità per garantire la gestione delle relazioni forward e backward tra requisiti e codice.

- **Tools**: Gli strumenti impiegati hanno incluso:
    - Le funzionalità di ricerca dell'IDE.
    - La ricerca di elementi sintattici e lessicali all'interno del codice sorgente.

2.2 Descrizione del sistema attuale

Il sistema attuale è una piattaforma di analisi progettata per lo studio dell'evoluzione del software e l'individuazione dei code smells all'interno dei repository di progetti open source. Si tratta di un'estensione e adattamento dello strumento CodeFace, pensato per supportare le attività di manutenzione e refactoring.

La piattaforma consente di:

- Raccogliere e correlare dati provenienti da repository Git, issue tracker e sistemi di tagging.
- Generare matrici di co-evoluzione tra moduli e sviluppatori.
- Identificare code smells tramite metriche e regole basate sul grafo delle dipendenze interne.
- Produrre report in formato PDF e visualizzazioni grafiche relative a dipendenze e cluster.

In questo modo, la piattaforma offre un supporto efficace per l'analisi e lo studio dell'evoluzione di qualsiasi progetto open source.

2.3 Attori del sistema

Gli attori coinvolti nel sistema sono i seguenti:

- **Manutentore software**: È l'utente principale del sistema, che può:
  - Importare e analizzare nuove repository;
  - Consultare metriche software, grafi di dipendenza e code smells;
  - Eseguire confronti tra versioni successive del codice;
  - Generare report in formato PDF e CSV;
  - Personalizzare le soglie e i criteri di rilevamento degli smells.
- **Ricercatore**: Utente con competenze avanzate che può:
  - Configurare analisi su più progetti software contemporaneamente;
  - Esportare dati per ulteriori analisi esterne;
  - Sviluppare, testare e validare nuove tecniche di clustering e analisi.

2.4 Requisiti Funzionali

2.4.1 Gestione Containerizzazione e Deploy

RF_CD_1 - Deploy con Vagrant: Il sistema deve supportare il deployment tramite l'uso di Vagrant e di una Virtual Machine apposita installata sul dispositivo - **Priorità Alta**

RF_CD_2 - Installazione Automatica: Il sistema deve supportare l'installazione delle librerie necessarie al suo funzionamento - **Priorità Alta**

RF_CD_3 - Installazione e Setup di MySQL: Il sistema deve supportare l'installazione e la configurazione del database MySQL - **Priorità Alta**

2.4.2 Analisi ed Elaborazione Dati

RF_AD_4 - Import repository: Il sistema deve consentire l'import di repository da diverse sorgenti - **Priorità Alta**

RF_AD_5 - Analisi Multi-Release: Il sistema deve consentire l'analisi focalizzata su release e intervalli temporali specifici - **Priorità Alta**

RF_AD_6 - Rilevamento di Community Smells: Il sistema deve implementare algoritmi per il rilevamento di community smell - **Priorità Alta**

RF_AD_7 - Analisi Temporale: Il sistema deve consentire l'analisi dei commit e tag nel tempo - **Priorità Alta**

RF_AD_8 - Analisi Strutturale: Il sistema deve generare gradi di clusterizzazione degli sviluppatori - **Priorità Alta**

RF_AD_9 - REST API: Il sistema deve esporre delle API REST per la fruizione dei dati - **Priorità Alta**

2.4.3 Output e Visualizzazione

RF_OV_10 - Report Engine: Il sistema deve generare report completi - **Priorità Alta**

RF_OV_11 - Visualizzazione Interattiva: Il sistema deve fornire una dashboard web per l'esplorazione dei risultati in tempo reale - **Priorità Media**

2.4.4 Testing

RF_TE_12 - Esecuzione test: Il sistema supportare l'attività di testing tramite comandi appositi - **Priorità Media**

RF_TE_13 - Logging: Il sistema dovrà produrre file di log dettagliati - **Priorità Media**

Categoria	ID Requisito	Descrizione	Priorità
Gestione Containerizzazione e Deploy	**RF_CD_1**	Il sistema deve supportare il deployment tramite l'uso di Vagrant e di una Virtual Machine apposita installata sul dispositivo	**Alta**
	RF_CD_2	Il sistema deve supportare l'installazione delle librerie necessarie al suo funzionamento	**Alta**
	RF_CD_3	Il sistema deve supportare l'installazione e la configurazione del database MySQL	**Alta**
Analisi ed Elaborazione Dati	**RF_AD_4**	Il sistema deve consentire l'import di repository da diverse sorgenti	**Alta**
	RF_AD_5	Il sistema deve consentire l'analisi focalizzata su release e intervalli temporali specifici	**Alta**
	RF_AD_6	Il sistema deve implementare algoritmi per il rilevamento di community smell	**Alta**
	RF_AD_7	Il sistema deve consentire l'analisi dei commit e tag nel tempo	**Alta**
	RF_AD_8	Il sistema deve generare gradi di clusterizzazione degli sviluppatori	**Alta**
	RF_AD_9	Il sistema deve esporre delle API REST per la fruizione dei dati	**Alta**
Output e Visualizzazione	**RF_OV_10**	Il sistema deve generare report completi	**Alta**

	RF_OV_11	Il sistema deve fornire una dashboard web per l'esplorazione dei risultati in tempo reale	**Media**
Testing	**RF_TE_12**	Il sistema supportare l'attività di testing tramite comandi appositi	**Alta**
	RF_TE_13	Il sistema dovrà produrre file di log dettagliati	**Alta**

2.5 Architettura del Sistema

L'architettura del sistema poggia su una serie di componenti eseguiti in sequenza. I dati vengono analizzati e vengono generati dei report visualizzabili attraverso l'uso di una dashboard.

La componente principale del sistema è l'utilizzo di Vagrant, che consente di creare e gestire ambienti di sviluppo virtualizzati in modo coerente su diverse piattaforme. In questo modo il progetto può essere eseguito in un ambiente isolato e riproducibile, senza la necessità di configurare manualmente una macchina fisica o un sistema operativo dedicato.

I file di installazione vengono eseguiti in sequenza durante la build del progetto, e portano alla creazione di un container con base Ubuntu 12.04, fruibile via ssh.

Per quanto riguarda l'analisi utilizza un'architettura modulare ed eterogenea composta da più livelli e tecnologie, con componenti sviluppati in R, Python e JavaScript. La comunicazione tra moduli avviene tramite REST API, database relazionali (RDBMS) condivisi e file di configurazione YAML.
L'interfaccia utente è una web application visualizzabile via browser, che usa WebSocket per la comunicazione asincrona con il server centrale.

2.6 Database

2.6.1 Modello EER

Vedere file Allegato EER/Diagramma EER.png

2.6.2 Schema Relazionale

Project (*id*, name, analysisMethod, analysisTime*)
Person (*id*, name, projectId↑, email1, email2*, email3*, email4*, email5*)

Issue (*id*, externalId, url*, projectId↑, issueType, title, description*, creationDate, createdBy↑, priority, severity, version*, startDate*, dueDate*, assignedTo↑, estimatedTime*, spentTime, progress, status, resolution*, category*)
ReleaseTimeline (*id*, type, tag, date*, projectId↑)
ReleaseRange (*id*, releaseStartId↑, releaseEndId↑, projectId↑, releaseRCStartId↑)
MailingList (id, projectId↑, name, description)
MailThread (*id*, subject*, createdBy↑, *projectId↑, releaseRangeId↑, mlId↑, mailThreadId, creationDate*, numberOfAuthors, numberOfMessages)
ThreadResponses (who↑, mailThreadId↑, mailDate*)
WatcherList (issueId↑, who↑)
Commit (*id*, commitHash, commitDate, authorDate, author↑, projectId↑, ChangedFiles*, AddedLines*, DeletedLines*, DiffSize*, description*, corrective*, releaseRangeId↑)
CommitCommunication (id, commitId↑, who↑, communicationType)
IssueRelations (id, leftIssueId↑, relation, rightIssueId↑)
AuthorCommitStats (id, authorId↑, releaseRangeId↑, added, deleted*, total*, numcommits*)
Plots (*id*, name, projectId↑, releaseRangeId↑, *labelx*, labely*)
PlotBin (*plotID*↑, type, data)
Cluster (*id*, projectId↑, releaseRangeId↑, clusterNumber*, clusterMethod*, dot↑, *svg*↑)
ClusterUserMapping (*id*, personId↑, clusterId↑)
IssueChange (*id*, changeDate, who↑, issueId↑, comment*)
UrlInfo (*id*, projectId↑, type, url)
TimeSeries (plotId↑, time, value, valuescaled*)
FreqSubjects (*id*, projectId↑, releaseRangeId↑, mlId↑, subject, count)
ThreadDensity (*id*, num, density, type, projectId↑)
PageRank (*id*, releaseRangeId↑, technique, name*)
PageRankMatrix (pageRankId↑, personId↑, rankValue)
EdgeList (clusterId↑, fromId↑, toId↑, weight)
TwoModeEdgeList (releaseRangeId↑, source, mlId↑, fromVert↑, toVert, weight)
TwoModeVertices (releaseRangeId↑, source, mlId↑, name, degree, type)
InitiateResponse (releaseRangeId↑, mlId↑, personId↑, source, responses*, initiations*, responsesreceived*, deg*)
PerClusterStatistics (projectId↑, releaseRangeId↑, clusterId↑, technique, nummembers, added, deleted, total, numcommits, prankavg)
SlocCountTS (plotId↑, time, personmonths, totalcost, schedulemonths, avgdevel)
UnderstandRaw (plotId↑, time, kind, name*, variable, value)
CommitDependency (*id*, commitId↑, file, entityId, entityType, size*, impl*)
Mail (*id*, projectId↑, threadId↑, mlId↑, author↑, subject*, creationDate, creationDateOffset, messageId)

FieldChange (*id*, field, oldValue*, newValue*, issueChangeId↑)
IssueCommit (issueId↑, commitId↑)
RevisionsView (projectId, releaseRangeID, datestart, dateend, datercstart, tag, cycle)
AuthorCommitStatsView (Name, ID, releaseRangeId, added, deleted, total, numcommits)
PerPersonClusterStatisticsView (projectId, releaseRangeId, clusterId, personId, technique, rankValue, added, deleted, total, numcommits)
ClusterUserPageRankView (*id*, personId↑, clusterId↑, technique, rankValue)
PerClusterStatisticsView (projectId, releaseRangeId, clusterId, technique, nummembers, added, deleted, total, numcommits, prankavg)
PageRankView (pageRankId↑, authorId↑, name, rankValue)

2.7 Testing

Prima di procedere con la manutenzione del sistema, è stata effettuata una sessione di testing di sistema, per verificarne il funzionamento nella sua versione originale, e consentire un testing di regressione al termine della manutenzione. Il testing in questione non ha previsto l'uso di strumenti aggiuntivi, ma solamente di una serie di test svolti manualmente tramite riga di comando.

Ciascun test svolto ha consentito di procedere con il debug del progetto, portando alla risoluzione di tutti i vari bug trovati nei vari casi di test monitorati.

3 Attività di Manutenzione

L'analisi del progetto ha mostrato come l'architettura attuale presenta limiti strutturali e funzionali che ne compromettono la manutenibilità, la compatibilità e la stabilità su ambienti di lavoro moderni.
In particolare, il progetto risulta impossibile da eseguire nativamente su dispositivi basati su architettura ARM, a causa dell'assenza di supporto dedicato e delle criticità legate alla portabilità su sistemi non x86.

Un ulteriore svantaggio è rappresentato dalla necessità di ricorrere a macchine virtuali esterne al progetto per assicurare la corretta esecuzione, soluzione che complica il deployment, penalizza la qualità globale e limita notevolmente la semplicità d'utilizzo. Il presente documento propone formalmente alcune Change Request (CR) per avviare un'attività di refactoring profondo del codice esistente, finalizzata ad eliminare questi ostacoli e ad aumentare compatibilità ed efficienza nei contesti operativi più attuali.

3.1 CR_01

3.1.1 Identificazione analisi e valutazione

Project Name	Codeface4Smell		
Project Manager	Gabriele Santoro, Pasquale Sorrentino		
Requested By	Andrea De Lucia		
Submitted By	Gabriele Santoro, Pasquale Sorrentino	**Date Requested**	24/09/2025
Evaluated By	Gabriele Santoro, Pasquale Sorrentino	**Date Evaluated**	25/09/2025
Decided By	Gabriele Santoro, Pasquale Sorrentino	**Date Decided**	25/09/2025

Change Request Identification	
Change Request #	CR_01
Change Title	Sostituzione di Vagrant con Docker
Maintenance Type	Manutenzione Correttiva

Change Request Details	
Description	Sostituire la procedura di build e avvio del sistema basata su Vagrant con una soluzione containerizzata mediante Docker.
Cause	L'utilizzo di Vagrant impone l'esecuzione del sistema all'interno di una macchina virtuale esterna, con conseguente aumento significativo del carico sulle risorse hardware e limitazioni nella compatibilità hardware, escludendo ad esempio i sistemi basati su processori ARM Apple.

Justification	Docker utilizza container leggeri anziché macchine virtuali complete, semplificando notevolmente il processo di setup e avvio del sistema. Questo approccio snellisce la gestione delle risorse, migliora le performance di esecuzione e amplia la compatibilità su diverse piattaforme hardware, inclusi i dispositivi ARM di ultima generazione.
Priority	Alta

Evaluation	
Scope	Le modifiche proposte non snaturano lo scope del progetto, andando semplicemente ad implementare una migliore gestione delle risorse modificando leggermente i requisiti funzionali, in particolare il **RF_CD_1.**
Risk	La modifica della procedura di compilazione ed esecuzione, potrebbe introdurre errori fatali all'interno del sistema, dovuti ad incompatibilità delle nuove procedure. Per questo motivo verrà effettuata un'ampia attività di testing per controllare il corretto comportamento del sistema.
Alternatives and Recomendation	Al momento per raggiungere l'obiettivo non esistono alternative, se non utilizzare altri sistemi basati su containerizzazione.

Implementation Options
Le modifiche verranno implementate andando ad eliminare i riferimenti a Vagrant all'interno del progetto, e sostituendo il Vagrantfile con il Dockerfile. Inoltre gli script di installazione verranno divisi in modo tale da aumentare l'uso del caching di Docker, andando quindi a facilitare la fase di debug del progetto.

3.1.2 Impact Analysis

3.1.2.1 Start Impact Set

Lo *Start Impact Set (SIS)* include i file direttamente modificati nei commit analizzati.
I file identificati sono:

- \VagrantFile
- \README.md

	Artefatti	Descrizione	Grado
CR_01	VagrantFile	Rimozione del file	Forte
	README.md	Modifica di tutti i riferimenti a Vagrant	Debole

3.1.2.2 Candidate Impact Set

Il *Candidate Impact Set (CIS)* è definito considerando la struttura del progetto e le pipeline di elaborazione dei dati, i file che importano o dipendono dai moduli modificati, i test e gli script di orchestrazione coinvolti.
Dunque, i file identificati sono:

- \VagrantFile
- \README.md
- \DockerFile
- \integration-scripts
- \rpackages

	Artefatti	Descrizione	Grado
CR_01	VagrantFile	Rimozione del file	Forte
	README.md	Modifica di tutti i riferimenti a Vagrant	Debole
	DockerFile	Aggiunta del Dockerfile con conseguente procedura di compilazione del sistema	Forte

	Integration Scripts	Aggiunta di ulteriori scripts dedicati all'installazione di pacchetti Linux e Python	Forte
	rpackages	Aggiunta di scripts dedicati all'installazione di pacchetti R	Forte

3.1.2.3 Discovered Impact Set

Attraverso l'analisi statica delle dipendenze e test eseguiti nel contesto progettuale, sono stati rilevati come impattati i seguenti file:

- \packages.r
- \vagrant

	Artefatti	Descrizione
CR_01	packages.R	Il file conteneva le procedure di installazione di tutti i pacchetti R richiesti. Non più necessario in seguito alla creazione di script appositi, più aggiornati.
	Directory Vagrant	Eliminazione di tutti i riferimenti a qualsiasi directory /vagrant all'interno del progetto

3.1.2.4 False Positive Impact Set

Tutte le componenti del CIS sono state modificate, quindi il FPIS è vuoto.

3.1.2.5 Actual Impact Set

Di conseguenza l'*Actual Impact Set (AIS)* definito come: $AIS = (SIS \cup DIS) - FPIS$
Risulta essere:

- \VagrantFile
- \vagrant
- \README.md
- \DockerFile
- \integration-scripts

- \rpackages
- packages.r

	Artefatti	Descrizione
CR_01	VagrantFile	Rimozione del file
	README.md	Modifica di tutti i riferimenti a Vagrant
	DockerFile	Aggiunta del Dockerfile con conseguente procedura di compilazione del sistema
	Integration Scripts	Aggiunta di ulteriori scripts dedicati all'installazione di pacchetti Linux e Python
	rpackages	Aggiunta di scripts dedicati all'installazione di pacchetti R
	packages.R	Il file conteneva le procedure di installazione di tutti i pacchetti R richiesti. Non più necessario in seguito alla creazione di script appositi, più aggiornati.
	Directory Vagrant	Eliminazione di tutti i riferimenti a qualsiasi directory /vagrant all'interno del progetto

3.1.3 Metriche di valutazione

Insieme	Cardinalità
SIS	2
CIS	5
DIS	2
FPIS	0
AIS	7

Metrica	Formula	Valore
Recall	Recall = TP / (TP + FN) = |CIS ∩ AIS| / |AIS|	71.4%
Precision	Precision = TP / (TP + FP) = |CIS ∩ AIS| / |CIS|	100%

3.2 CR_02

3.2.1 Identificazione analisi e valutazione

Project Name	Codeface4Smell		
Project Manager	Gabriele Santoro, Pasquale Sorrentino		
Requested By	Andrea De Lucia		
Submitted By	Gabriele Santoro, Pasquale Sorrentino	**Date Requested**	24/09/2025
Evaluated By	Gabriele Santoro, Pasquale Sorrentino	**Date Evaluated**	25/09/2025
Decided By	Gabriele Santoro, Pasquale Sorrentino	**Date Decided**	25/09/2025

Change Request Identification	
Change Request #	CR_02
Change Title	Aggiornamento utilizzo di Python nel sistema
Maintenance Type	Manutenzione Perfettiva

Change Request Details	
Description	Aggiornare tutte le funzioni e dipendenze Python obsolete, non compatibili con la versione 3.6 e successive, sostituendo il

	codice e le librerie che utilizzavano sintassi o funzionalità presenti solo in Python 2.7.
Cause	L'upgrade del sistema operativo a Ubuntu 22.04 ha reso necessario l'uso di una versione più recente di Python, eliminando la compatibilità con Python 2.7 e rendendo indispensabile l'adeguamento del codice applicativo.
Justification	L'adozione di Python 3 non solo garantisce compatibilità con gli ambienti moderni e futuri aggiornamenti, ma consente anche di migliorare la robustezza, le performance e la sicurezza del sistema, riducendo le esigenze di manutenzione e il rischio di vulnerabilità derivanti dall'uso di software deprecato.
Priority	Alta

Evaluation	
Scope	Le modifiche proposte non snaturano lo scope del progetto.
Risk	La migrazione del sistema Codeface verso Python 3 comporta un rischio significativo di malfunzionamenti improvvisi, principalmente dovuti all'utilizzo di sintassi e funzionalità non più supportate rispetto alla precedente versione 2.7. Diverse porzioni di codice e librerie legacy potrebbero risultare incompatibili o generare errori runtime se non adeguatamente revisionate e aggiornate, rendendo fondamentale un approccio graduale e una fase di testing approfondita per garantire la stabilità dell'applicativo durante la transizione.
Alternatives and Recomendation	Al momento per raggiungere l'obiettivo non esistono alternative.

Implementation Options

Le modifiche verranno implementate in maniera graduale, verificando i punti in cui il programma va in errore, per poter intervenire a modificare le porzioni di codice non più supportate.

3.2.2 Impact Analysis

3.2.2.1 Start Impact Set

Lo *Start Impact Set (SIS)* include i file direttamente modificati nei commit analizzati.
I file identificato è: \setup.py

	Artefatti	Descrizione	Grado
CR_02	setup.py	Configurazione dei pacchetti python riadattata alla nuova versione, andando a sostituire i pacchetti deprecati, e aggiornando le versioni non più supportate	Forte

3.2.2.2 Candidate Impact Set

Il *Candidate Impact Set (CIS)* è definito considerando la struttura del progetto e le pipeline di elaborazione dei dati, i file che importano o dipendono dai moduli modificati, i test e gli script di orchestrazione coinvolti.
Dunque, i file identificati sono:

- codeface\cli.py
- codeface\cluster\cluster.py
- codeface\cluster\codeBlock.py
- codeface\cluster\idManager.py
- codeface\cluster\PersonInfo.py
- codeface\commit_analysis.py
- codeface\configuration.py
- codeface\dbmanager.py
- codeface\fileCommit.py
- codeface\logger.py
- codeface\project.py

- codeface\sourceAnalysis.py
- codeface\ts.py
- codeface\util.py
- codeface\VCS.py
- codeface\test\integration\example_projects.py
- codeface\test\integration\gitproject.py
- codeface\test\integration\test_exampleprojects.py
- codeface\test\integration\test_features.py
- codeface\test\integration\test_logger.py
- codeface\test\unit\test_batchjob.py
- codeface\test\unit\test_cppstats_works.py
- codeface\test\unit\test_getFeatureLines.py
- codeface\test\unit\test_logger.py
- codeface\test\unit\test_R_code.py
- id_service\test.py

	Artefatti	Descrizione	Grado
CR_02	cli.py	Modifica di sintassi di “print” ed import	Forte
	cluster.py	Modifica degli import	Medio
	codeBlock.py	Modifica degli import	Medio
	idManager.py	Modifica degli import	Medio
	PersonInfo.py	Modifica di metodi di gestione delle stringhe	Forte
	commit_analysis.py	Modifica degli import	Medio
	configuration.py	Modifica degli import	Medio
	dbmanager.py	Nuovi metodi per la gestione del db	Forte
	fileCommit.py	Modifica degli import	Medio
	logger.py	Modifica degli import	Medio
	project.py	Modifica degli import	Medio
	sourceAnalysis.py	Modifica degli import, e conseguente modifiche al codice	Forte

	ts.py	Gestione delle eccezioni più efficace	Medio
	util.py	Modifica degli import	Medio
	VCS.py	Modifica degli import	Medio
	example_projects.py	Modifica degli import	Medio
	gitproject.py	Modifiche sull'apertura dei file, e sull'iterazione delle liste.	Forte
	test_exampleprojects.py	Modifiche degli import e iterazione delle liste	Forte
	test_features.py	Modifiche degli import	Medio
	test_logger.py	Modifiche degli import e conseguente modifica del codice	Forte
	test_batchjob.py	Rimozione di alcuni import, e aggiunta di gestione di exception	Medio
	test_cppstats_works.py	Modifiche degli import	Medio
	test_getFeatureLines.py	Modifiche degli import	Medio
	unit/test_logger.py	Modifiche degli import e conseguente modifica del codice	Forte
	test_R_code.py	Modifiche degli import	Medio
	test.py	Modifiche degli import e conseguente modifica del codice	Forte

3.2.2.3 Discovered Impact Set

Attraverso l'analisi statica delle dipendenze e test eseguiti nel contesto progettuale, sono stati rilevati come impattati i seguenti file:

- codeface\conway.py
- codeface\DBAnalysis.py
- cppstats\cppstats.py

	Artefatti	Descrizione
CR_02	conway.py	Nuovo modulo per effettuare l'analisi tramite Jira
	DBAnalysis.py	Nuovo modulo per l'analisi del DB
	cppstats	Il modulo cppstats è stato importato in locale e modificato, per essere adattato al nuovo Python

3.2.2.4 False Positive Impact Set

Tutte le componenti del CIS sono state modificate, quindi il FPIS è vuoto.

3.2.2.5 Actual Impact Set

Di conseguenza l'*Actual Impact Set (AIS)* definito come: $AIS = (SIS \cup DIS) - FPIS$
Risulta essere:

- \setup.py
- codeface\cli.py
- codeface\cluster\cluster.py
- codeface\cluster\codeBlock.py
- codeface\cluster\idManager.py
- codeface\cluster\PersonInfo.py
- codeface\commit_analysis.py
- codeface\configuration.py
- codeface\dbmanager.py
- codeface\fileCommit.py
- codeface\logger.py
- codeface\project.py
- codeface\sourceAnalysis.py
- codeface\ts.py
- codeface\util.py
- codeface\VCS.py
- codeface\test\integration\example_projects.py
- codeface\test\integration\gitproject.py
- codeface\test\integration\test_exampleprojects.py
- codeface\test\integration\test_features.py

- codeface\test\integration\test_logger.py
- codeface\test\unit\test_batchjob.py
- codeface\test\unit\test_cppstats_works.py
- codeface\test\unit\test_getFeatureLines.py
- codeface\test\unit\test_logger.py
- codeface\test\unit\test_R_code.py
- id_service\test.py
- codeface\conway.py
- codeface\DBAnalysis.py
- cppstats\cppstats.py

	Artefatti	Descrizione
CR_02	setup.py	Configurazione dei pacchetti python riadattata alla nuova versione, andando a sostituire i pacchetti deprecati, e aggiornando le versioni non più supportate
	cli.py	Modifica di sintassi di “print” ed import
	cluster.py	Modifica degli import
	codeBlock.py	Modifica degli import
	idManager.py	Modifica degli import
	PersonInfo.py	Modifica di metodi di gestione delle stringhe
	commit_analysis.py	Modifica degli import
	configuration.py	Modifica degli import
	dbmanager.py	Nuovi metodi per la gestione del db
	fileCommit.py	Modifica degli import
	logger.py	Modifica degli import
	project.py	Modifica degli import
	sourceAnalysis.py	Modifica degli import, e conseguente modifiche al codice

	ts.py	Gestione delle eccezioni più efficace
	util.py	Modifica degli import
	VCS.py	Modifica degli import
	example_projects.py	Modifica degli import
	gitproject.py	Modifiche sull'apertura dei file, e sull'iterazione delle liste.
	test_exampleprojects.py	Modifiche degli import e iterazione delle liste
	test_features.py	Modifiche degli import
	test_logger.py	Modifiche degli import e conseguente modifica del codice
	test_batchjob.py	Rimozione di alcuni import, e aggiunta di gestione di exception
	test_cppstats_works.py	Modifiche degli import
	test_getFeatureLines.py	Modifiche degli import
	unit/test_logger.py	Modifiche degli import e conseguente modifica del codice
	test_R_code.py	Modifiche degli import
	test.py	Modifiche degli import e conseguente modifica del codice
	conway.py	Nuovo modulo per effettuare l'analisi tramite Jira
	DBAnalysis.py	Nuovo modulo per l'analisi del DB
	cppstats	Il modulo cppstats è stato importato in locale e modificato, per essere adattato al nuovo Python

3.2.3 Metriche di valutazione

Insieme	Cardinalità
SIS	1
CIS	26
DIS	3
FPIS	0
AIS	30

3.3 CR_03

3.3.1 Identificazione analisi e valutazione

Project Name	Codeface4Smell		
Project Manager	Gabriele Santoro, Pasquale Sorrentino		
Requested By	Andrea De Lucia		
Submitted By	Gabriele Santoro, Pasquale Sorrentino	**Date Requested**	24/09/2025
Evaluated By	Gabriele Santoro, Pasquale Sorrentino	**Date Evaluated**	25/09/2025
Decided By	Gabriele Santoro, Pasquale Sorrentino	**Date Decided**	25/09/2025

Change Request Identification	
Change Request #	CR_03
Change Title	Aggiornamento librerie R nel sistema
Maintenance Type	Manutenzione Perfettiva

Change Request Details	
Description	Aggiornare tutte le librerie e funzioni di R deprecate, in seguito al passaggio ad una versione di R più aggiornata.
Cause	L'aggiornamento del Sistema Operativo alla versione 22.04, ha portato alla necessità di installare nuove versioni di R, e delle sue librerie, andando ad eliminare o aggiornare le librerie deprecate.
Justification	L'aggiornamento di R non solo garantisce compatibilità con gli ambienti moderni e futuri aggiornamenti, ma consente anche di migliorare la robustezza, le performance e la sicurezza del sistema.
Priority	Alta

3.3.2 Impact Analysis

3.3.2.1 Start Impact Set

Lo *Start Impact Set (SIS)* include i file direttamente modificati nei commit analizzati.
I file identificati sono:

- \rpackages
- \Dockerfile

	Artefatti	Descrizione	Grado
CR_03	packages.R	Lo script monolitico per l'installazione di tutti i pacchetti R necessari al progetto, è stato rimosso, per far posto ad una ripartizione migliore basata sulla tipologia di pacchetti.	Forte
	packages.minimal.R	Lo script serviva a fare un'installazione minima e rapida delle dipendenze essenziali. Rimosso per lo stesso motivo del file pakcages.R	Medio

	Dockerfile	Il file di configurazione di Docker è stato modificato per integrare i nuovi script che provvederanno a sostituire quelli eliminati.	Forte

3.3.2.2 Candidate Impact Set

Il *Candidate Impact Set (CIS)* è definito considerando la struttura del progetto e le pipeline di elaborazione dei dati, i file che importano o dipendono dai moduli modificati, i test e gli script di orchestrazione coinvolti.
Dunque, i file identificati sono:

- integration-scripts\install_codeface_R.sh
- \rpackages

	Artefatti	**Descrizione**	**Grado**
CR_03	install_codeface_R.sh	Lo script per l'installazione delle dipendenze di sistema necessarie per R, che gestisce anche l'installazione di alcuni pacchetti tramite "apt", più rapido rispetto all'installazione classica di R.	Forte
	packages.R	Lo script monolitico per l'installazione di tutti i pacchetti R necessari al progetto, è stato rimosso, per far posto ad una ripartizione migliore basata sulla tipologia di pacchetti.	Forte

	packages.minimal.R	Lo script serviva a fare un'installazione minima e rapida delle dipendenze	Forte

3.3.2.3 Discovered Impact Set

Attraverso l'analisi statica delle dipendenze e test eseguiti nel contesto progettuale, sono stati rilevati come impattati i seguenti file:

- rpackages\install_bioc_packages.R
- rpackages\install_cran_packages.R
- rpackages\install_cran_packages.sh
- rpackages\install_devtools.R
- rpackages\install_github_packages.R
- rpackages\install_shiny_packages.R
- rpackages\utils.R

	Artefatti	Descrizione
CR_03	install_bioc_packages.R	Nuovo modulo che sostituisce l'installazione dei pacchetti Bioconductor.
	install_cran_packages.R	Nuovo modulo che sostituisce l'installazione di pacchetti CRAN
	install_cran_packages.sh	Nuovo modulo che gestisce l'installazione di pacchetti CRAN, disponibili via installazione apt-get
	install_devtools.R	Installazione del pacchetto di devtools
	install_github_packages.R	Installazione di progetti GitHub specifici
	utils.R	File che contiene funzioni di supporto per l'installazione e disinstallazione di pacchetti

3.3.2.4 False Positive Impact Set

Tutte le componenti del CIS sono state modificate, quindi il FPIS è vuoto.

3.3.2.5 Actual Impact Set

Di conseguenza l'*Actual Impact Set (AIS)* definito come: $AIS = (SIS \cup DIS) - FPIS$
Risulta essere:

- \rpackages
- \Dockerfile
- integration-scripts\install_codeface_R.sh
- rpackages\install_bioc_packages.R
- rpackages\install_cran_packages.R
- rpackages\install_cran_packages.sh
- rpackages\install_devtools.R
- rpackages\install_github_packages.R
- rpackages\install_shiny_packages.R
- rpackages\utils.R

	Artefatti	Descrizione
CR_03	install_codeface_R.sh	Lo script per l'installazione delle dipendenze di sistema necessarie per R, che gestisce anche l'installazione di alcuni pacchetti tramite "apt", più rapido rispetto all'installazione classica di R.
	install_bioc_packages.R	Nuovo modulo che sostituisce l'installazione dei pacchetti Bioconductor.
	install_cran_packages.R	Nuovo modulo che sostituisce l'installazione di pacchetti CRAN
	install_cran_packages.sh	Nuovo modulo che gestisce l'installazione di pacchetti CRAN, disponibili via installazione apt-get
	install_devtools.R	Installazione del pacchetto di devtools
	install_github_packages.R	Installazione di progetti GitHub specifici
	utils.R	File che contiene funzioni di supporto per l'installazione e disinstallazione di pacchetti

	packages.R	Lo script monolitico per l'installazione di tutti i pacchetti R necessari al progetto, è stato rimosso, per far posto ad una ripartizione migliore basata sulla tipologia di pacchetti.
	packages.minimal.R	Lo script serviva a fare un'installazione minima e rapida delle dipendenze essenziali. Rimosso per lo stesso motivo del file pakcages.R
	Dockerfile	Il file di configurazione di Docker è stato modificato per integrare i nuovi script che provvederanno a sostituire quelli eliminati.

3.3.3 Metriche di valutazione

Insieme	Cardinalità
SIS	3
CIS	3
DIS	6
FPIS	0
AIS	10

3.4 Metriche

Per valutare l'efficacia dell'impact analysis condotta su ciascuna Change Request, sono state impiegate tre metriche statistiche complementari: *Recall*, *Precision* e *F-measure*.
Queste metriche forniscono una valutazione multidimensionale della qualità dell'analisi di impatto.

3.4.1 Recall

Il Recall misura la completezza dell'analisi di impatto, ovvero la capacità di identificare correttamente tutti gli artefatti che dovranno essere effettivamente modificati durante l'implementazione della Change Request.

La formula è la seguente:

$$Recall = \frac{TP}{(TP + FN)} = \frac{|\, CIS \cap AIS \,|}{|\, AIS \,|} \times 100$$

Dove:

- TP (True Positives): Artefatti correttamente previsti come impattati
- FN (False Negatives): Artefatti impattati ma non previsti

Un Recall alto indica che l'analisi ha coperto efficacemente tutti gli artefatti rilevanti, minimizzando il rischio di scoperta tardiva di componenti trattanti.

3.4.2 Precision

La Precision misura l'accuratezza delle previsioni dell'analisi di impatto, ovvero la capacità di evitare sovrastime e falsi allarmi.

$$Precision = \frac{TP}{(\, TP + FP \,)} = \frac{|\, CIS \cap AIS \,|}{|\, CIS \,|} \times 100$$

Una Precision alta indica che ogni artefatto previsto è stato effettivamente modificato, evitando lo spreco di risorse su componenti non rilevanti. Una Precision bassa, invece, significa che il team ha dedicato tempo e risorse ad elementi che non avevano effettivamente bisogno di modifiche.

3.4.3 F-Measure

L'F-Measure è la media armonica tra Recall e Precision, il che significa che da uguale peso ad entrambe le metriche. L'uso della media armonica assicura che l'F-Measure rimanga basso se una delle due metriche è significativamente inferiore all'altra, forzando il raggiungimento di un equilibrio tra completezza e accuratezza.

$$F - Measure = 2 \times \frac{Precision * Recall}{Precision + Recall} \times 100$$

3.4.4 Risultati

	Recall	Precision	F-Measure
CR_01	$\frac{5}{7}\% = 71.4\%$	$\frac{5}{5}\% = 100\%$	$2 \times \frac{1 \times 0.714}{1 + 0.714}\% = 83\%$
CR_02	$\frac{26}{30}\% = 86.7\%$	$\frac{26}{26}\% = 100\%$	$2 \times \frac{1 \times 0.867}{1 + 0.867}\% = 93\%$
CR_03	$\frac{3}{10}\% = 30\%$	$\frac{3}{3}\% = 100\%$	$2 \times \frac{1 \times 0.3}{1 + 0.3}\% = 46\%$

3.5 Obiettivi del Refactoring

L'attività di refactoring ha come obiettivi principali il superamento delle criticità strutturali e funzionali rilevate nel progetto Codeface, al fine di migliorarne la qualità, la manutenibilità e la compatibilità con ambienti tecnologici moderni. In particolare, si propone di:

- Risolvere la deprecazione delle librerie R fondamentali, sostituendo i pacchetti obsoleti (es. reshape, plyr, RJSONIO, wordnet, shinybootstrap2, shinyGridster) con alternative moderne e attivamente mantenute, per garantire stabilità, aggiornabilità e migliori performance.

- Ridurre gli errori sistemici e i comportamenti inattesi che si manifestano in fasi cruciali dell'esecuzione (come la generazione di time series e la gestione delle boundaries), intervenendo sulle logiche di gestione dati e aumentando la robustezza del codice.

- Migliorare la capacità di debug e la qualità del ciclo di sviluppo introducendo un framework strutturato di testing e validazione del codice, che consenta di individuare e correggere tempestivamente anomalie e regressioni.

- Adeguare il codice Python esistente alla versione 3, abbandonando il supporto a Python 2, per assicurare compatibilità con piattaforme aggiornate, migliorare la sicurezza e sfruttare le nuove funzionalità e ottimizzazioni del linguaggio.

Attraverso questi interventi, il refactoring mira a rendere Codeface un sistema più modulare, estensibile e affidabile, capace di evolvere agevolmente nel tempo in risposta alle nuove esigenze degli utenti e del mercato.

3.6 Benefici Attesi

- **Miglioramento della stabilità e dell'affidabilità**: La sostituzione di librerie R deprecate con alternative moderne ridurrà errori sistemici e malfunzionamenti, incrementando la

robustezza complessiva del sistema.

- **Aumento della manutenibilità**: Un codice più modulare e meno accoppiato facilita future estensioni, modifiche o sostituzioni di singoli componenti senza compromettere l'intero sistema.

- **Maggiore facilità di debug e testing**: L'introduzione di un framework strutturato per il controllo qualità consentirà di identificare e risolvere anomalie più velocemente, riducendo i tempi di sviluppo e migliorando la qualità del software.

- **Compatibilità e sicurezza migliorate**: L'aggiornamento del codice Python a Python 3 garantirà il funzionamento su ambienti moderni, diminuendo i rischi legati all'uso di versioni obsolete, e permettendo di sfruttare nuove funzionalità e ottimizzazioni del linguaggio.

- **Ottimizzazione delle performance**: L'adozione di pacchetti aggiornati e codice più efficiente contribuirà a ridurre i colli di bottiglia e a migliorare i tempi di esecuzione, rendendo Codeface più reattivo e performante.

- **Facilitazione della collaborazione**: Codice più chiaro e leggibile agevolerà il lavoro del team di sviluppo, riducendo il debito tecnico e promuovendo uno sviluppo più sostenibile e coordinato.

- **Longevità del progetto**: Questi interventi garantiranno che Codeface possa adattarsi facilmente ai cambiamenti tecnologici futuri, preservandone valore e utilità nel tempo.

3.7 Strategie di Debugging

Sono state utilizzate molteplici tecniche per identificare e risolvere problemi all'interno del sistema.

- Separazione delle fasi di installazione, per garantire la possibilità di un caching efficace in fase di build

- Inserimento di log visibili a console, per valutare l'avanzamento dell'analisi

- Analisi manuale del codice che presenta bug o incompatibilità con le nuove versioni di OS e Python installate

- Analisi dello schema del DB, adattandolo alle nuove esigenze, e controlli dei valori inseriti in seguito ad un'analisi di un progetto Git

4 Sistema Modificato

4.1 Descrizione del sistema modificato

Pur essendo un'operazione di refactoring e ripristino del sistema, il progetto ha mantenuto tutte le sue funzionalità originarie.

Di conseguenza, il sistema continua a rappresentare una piattaforma automatizzata per l'analisi dell'evoluzione di software open source scelto dall'utente, offrendo inoltre capacità avanzate di identificazione e valutazione dei code smells.

Il sistema offre le seguenti funzionalità principali:

- Raccogliere e correlare informazioni provenienti da repository Git, issue tracker e sistemi di tagging;
- Generare matrici di co-evoluzione tra moduli e sviluppatori, evidenziando le interdipendenze più rilevanti;
- Individuare code smells utilizzando metriche specifiche e regole basate sull'analisi del grafo delle dipendenze interne al codice;
- Produrre report in formato PDF e visualizzazioni grafiche chiare delle dipendenze e dei cluster all'interno del progetto.

La piattaforma è progettata per facilitare lo studio dell'evoluzione del software da parte di ricercatori e manutentori, permettendo un'analisi dettagliata delle modifiche nel tempo e identificando le aree soggette a degrado strutturale.

Per favorire la collaborazione e la replicabilità delle analisi, CodeFace4Smells mette a disposizione una suite di script in R e Python che automatizzano i processi di analisi rendendo la piattaforma flessibile e adattabile a molteplici contesti software.

La miglioria principale introdotta consiste nell'adozione di Docker, che consente la build del progetto per un utilizzo multipiattaforma, sostituendo la necessità di Virtual Machine esterne con container più convenienti e ottimizzati. Grazie a Docker, CodeFace4Smell può ora essere eseguito anche su sistemi dotati di processori ARM di ultima generazione, garantendo maggiore efficienza, semplicità d'uso e compatibilità con piattaforme hardware moderne.

4.2 Attori del Sistema

Gli attori coinvolti nel sistema sono rimasti invariati, e sono:

- **Manutentore software**: È l'utente principale del sistema, che può:
    - Importare e analizzare nuove repository;
    - Consultare metriche software, grafi di dipendenza e code smells;
    - Eseguire confronti tra versioni successive del codice;
    - Generare report in formato PDF e CSV;
    - Personalizzare le soglie e i criteri di rilevamento degli smells.

- **Ricercatore**: Utente con competenze avanzate che può:
    - Configurare analisi su più progetti software contemporaneamente;
    - Esportare dati per ulteriori analisi esterne;
    - Sviluppare, testare e validare nuove tecniche di clustering e analisi.

4.3 Requisiti funzionali

4.3.1 Gestione Containerizzazione e Deploy

RF_CD_1 - Deploy con Docker: Il sistema deve supportare il deployment automatico di tutti i servizi attraverso Docker Compose con gestione delle dipendenze - **Priorità Alta**

RF_CD_2 - Installazione Automatica: Il sistema deve supportare l'installazione delle librerie necessarie al suo funzionamento - **Priorità Alta**

RF_CD_3 - Installazione e Setup di MySql: Il sistema deve supportare l'installazione e la configurazione del database MySQL - **Priorità Alta**

4.3.2 Analisi ed Elaborazione Dati

RF_AD_4 - Import repository: Il sistema deve consentire l'import di repository da diverse sorgenti - **Priorità Alta**

RF_AD_5 - Analisi Multi-Release: Il sistema deve consentire l'analisi focalizzata su release e intervalli temporali specifici - **Priorità Alta**

RF_AD_6 - Rilevamento di Community Smells: Il sistema deve implementare algoritmi per il rilevamento di community smell - **Priorità Alta**

RF_AD_7 - Analisi Temporale: Il sistema deve consentire l'analisi dei commit e tag nel tempo - **Priorità Alta**
RF_AD_8 - Analisi Strutturale: Il sistema deve generare gradi di clusterizzazione degli sviluppatori - **Priorità Alta**

RF_AD_9 - REST API: Il sistema deve esporre delle API REST per la fruizione dei dati - **Priorità Alta**

4.3.3 Output e Visualizzazione

RF_OV_10 - Report Engine: Il sistema deve generare report completi - **Priorità Alta**

RF_OV_11 - Visualizzazione Interattiva: Il sistema deve fornire una dashboard web per l'esplorazione dei risultati in tempo reale - **Priorità Media**

4.3.4 Testing

RF_TE_12 - Esecuzione test: Il sistema supportare l'attività di testing tramite comandi appositi - **Priorità Media**

RF_TE_13 - Logging: Il sistema dovrà produrre file di log dettagliati - **Priorità Media**

Categoria	ID Requisito	Descrizione	Priorità
Gestione Containerizzazione e Deploy	**RF_CD_1**	Il sistema deve supportare il deployment automatico di tutti i servizi attraverso Docker Compose con gestione delle dipendenze	**Alta**
	RF_CD_2	Il sistema deve supportare l'installazione delle librerie necessarie al suo funzionamento	**Alta**
	RF_CD_3	Il sistema deve supportare l'installazione e la configurazione del database MySQL	**Alta**
Analisi ed Elaborazione Dati	**RF_AD_4**	Il sistema deve consentire l'import di repository da diverse sorgenti	**Alta**
	RF_AD_5	Il sistema deve consentire l'analisi focalizzata su release e intervalli temporali specifici	**Alta**

	RF_AD_6	Il sistema deve implementare algoritmi per il rilevamento di community smell	**Alta**
	RF_AD_7	Il sistema deve consentire l'analisi dei commit e tag nel tempo	**Alta**
	RF_AD_8	Il sistema deve generare gradi di clusterizzazione degli sviluppatori	**Alta**
	RF_AD_9	Il sistema deve esporre delle API REST per la fruizione dei dati	**Alta**
Output e Visualizzazione	**RF_OV_10**	Il sistema deve generare report completi	**Alta**
	RF_OV_11	Il sistema deve fornire una dashboard web per l'esplorazione dei risultati in tempo reale	**Media**
Testing	**RF_TE_12**	Il sistema supportare l'attività di testing tramite comandi appositi	**Alta**
	RF_TE_13	Il sistema dovrà produrre file di log dettagliati	**Alta**

4.4 Architettura del Sistema

L'architettura del sistema CodeFace4Smell è organizzata secondo un paradigma a microservizi containerizzati, in cui componenti specializzati (container MySQL, worker Python, engine R, server API) vengono orchestrati tramite Docker Compose e comunicano attraverso interfacce REST standardizzate.

I dati vengono elaborati in modalità real-time e batch ibrida, con risultati accessibili immediatamente tramite API REST e visualizzabili attraverso dashboard web interattive e report generati dinamicamente.

L'architettura è suddivisa logicamente nei seguenti quattro strati principali:

- Strato di Containerizzazione e Deploy: Componenti come Docker Compose, health checks automatici e script di provisioning gestiscono l'intero ciclo di vita dell'infrastruttura, garantendo scalabilità orizzontale e isolamento dei servizi.

- Strato di Elaborazione Distribuita: Worker Python specializzati e engine R statistici operano in parallelo per l'analisi multi-dimensionale (strutturale, temporale, community smell detection), comunicando attraverso code message-based e database condiviso.

- Strato di Integrazione e API: Server REST centralizzato espone endpoint standardizzati per import repository, configurazione analisi e recupero risultati, abilitando integrazioni con pipeline CI/CD esterne e sistemi enterprise.

- Strato di Presentazione Interattiva: Dashboard web responsive e report engine multi-formato forniscono visualizzazioni real-time, drill-down interattivi e export personalizzabili, sostituendo i report statici con esperienze utente dinamiche.

4.5 Database

4.5.1 Modello EER

Vedere file Allegato EER/Diagramma EER.png

4.5.2 Schema Relazionale

Project (*id*, name, analysisMethod, analysisTime*)
Person (*id*, name, projectId↑, email1, email2*, email3*, email4*, email5*)
Issue (*id*, externalId, url*, projectId↑, issueType, title, description*, creationDate, createdBy↑, priority, severity, version*, startDate*, dueDate*, assignedTo↑, estimatedTime*, spentTime, progress, status, resolution*, category*)
ReleaseTimeline (*id*, type, tag, date*, projectId↑)
ReleaseRange (*id*, releaseStartId↑, releaseEndId↑, projectId↑, releaseRCStartId↑*)*
MailingList (id, projectId↑, name, description)
MailThread (*id*, subject*, createdBy↑, *projectId↑, releaseRangeId↑, mlId↑, mailThreadId, creationDate*, numberOfAuthors, numberOfMessages)
ThreadResponses (who↑, mailThreadId↑, mailDate*)
WatcherList (issueId↑, who↑)
Commit (*id*, commitHash, commitDate, authorDate, author↑, projectId↑, ChangedFiles*, AddedLines*, DeletedLines*, DiffSize*, description*, corrective*, releaseRangeId↑*)*
CommitCommunication (id, commitId↑, who↑, communicationType)
IssueRelations (id, leftIssueId↑, relation, rightIssueId↑)
AuthorCommitStats (id, authorId↑, releaseRangeId↑, added, deleted*, total*, numcommits*)

Plots (*id*, name, projectId↑, releaseRangeId↑, *labelx*, labely*)
PlotBin (*plotID*↑, type, data)
Cluster (*id*, projectId↑, releaseRangeId↑, clusterNumber*, clusterMethod*, dot↑, *svg*↑)
ClusterUserMapping (*id*, personId↑, clusterId↑)
IssueChange (*id*, changeDate, who↑, issueId↑, comment*)
UrlInfo (*id*, projectId↑, type, url)
TimeSeries (plotId↑, time, value, valuescaled*)
FreqSubjects (*id*, projectId↑, releaseRangeId↑, mlId↑, subject, count)
ThreadDensity (*id*, num, density, type, projectId↑)
PageRank (*id*, releaseRangeId↑, technique, name*)
PageRankMatrix (pageRankId↑, personId↑, rankValue)
EdgeList (clusterId↑, fromId↑, toId↑, weight)
TwoModeEdgeList (releaseRangeId↑, source, mlId↑, fromVert↑, toVert, weight)
TwoModeVertices (releaseRangeId↑, source, mlId↑, name, degree, type)
InitiateResponse (releaseRangeId↑, mlId↑, personId↑, source, responses*, initiations*, responsesreceived*, deg*)
PerClusterStatistics (projectId↑, releaseRangeId↑, clusterId↑, technique, nummembers, added, deleted, total, numcommits, prankavg)
SlocCountTS (plotId↑, time, personmonths, totalcost, schedulemonths, avgdevel)
UnderstandRaw (plotId↑, time, kind, name*, variable, value)
CommitDependency (*id*, commitId↑, file, entityId, entityType, size*, impl*)
Mail (*id*, projectId↑, threadId↑, mlId↑, author↑, subject*, creationDate, creationDateOffset, messageId)
FieldChange (*id*, field, oldValue*, newValue*, issueChangeId↑)
IssueCommit (issueId↑, commitId↑)
RevisionsView (projectId, releaseRangeID, datestart, dateend, datercstart, tag, cycle)
AuthorCommitStatsView (Name, ID, releaseRangeId, added, deleted, total, numcommits)
PerPersonClusterStatisticsView (projectId, releaseRangeId, clusterId, personId, technique, rankValue, added, deleted, total, numcommits)
ClusterUserPageRankView (*id*, personId↑, clusterId↑, technique, rankValue)
PerClusterStatisticsView (projectId, releaseRangeId, clusterId, technique, nummembers, added, deleted, total, numcommits, prankavg)
PageRankView (pageRankId↑, authorId↑, name, rankValue)