



UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Magistrale in Informatica

MAINTENANCE REPORT

INGEGNERIA, GESTIONE ED EVOLUZIONE DEL SOFTWARE

DOCENTI

Prof. Andrea De Lucia

TUTOR

Dott. Stefano Lambiase

Università degli Studi di Salerno

STUDENTI

Francesco Maria Torino

(0522501879)

Francesco Alessandro

Pinto (0522501981)

Stefano Guida

(0522502054)

Anno Accademico 2024-2025

Indice

Elenco delle Figure	iii
Elenco delle Tabelle	iv
1 Introduzione	1
1.1 Processo di Impact Analysis	2
1.2 Metriche di Valutazione dell'Impact Analysis	4
2 Impact Analysis	6
2.1 CR1: Integrazione di TOAD in GUIDO	6
2.1.1 Obiettivi della CR	6
2.1.2 Starting Impact Set (SIS)	6
2.1.3 Candidate Impact Set (CIS)	7
2.1.4 Actual Impact Set (AIS)	7
2.1.5 Metriche di Processo	8
2.2 CR2: Aggiunta di una GUI per utilizzare TOAD all'interno di GUIDO	9
2.2.1 Obiettivi della Change Request	9
2.2.2 Starting Impact Set (SIS)	9
2.2.3 Candidate Impact Set (CIS)	11
2.2.4 Actual Impact Set (AIS)	11
2.2.5 Metriche di Processo	11

2.3	CR3: Estensione della GUI per la visualizzazione del grafo	12
2.3.1	Obiettivi della Change Request	12
2.3.2	Starting Impact Set (SIS)	13
2.3.3	Candidate Impact Set (CIS)	13
2.3.4	Actual Impact Set (AIS)	13
2.3.5	Metriche di Processo	14
2.4	CR4: Dockerizzazione TOAD e GUIDO	15
2.4.1	Obiettivi della Change Request	15
2.4.2	Starting Impact Set (SIS)	15
2.4.3	Candidate Impact Set (CIS)	16
2.4.4	Actual Impact Set (AIS)	16
2.4.5	Metriche di Processo	17
3	Implementazione delle Change Request	18
3.1	CR1 – Integrazione di TOAD in GUIDO	18
3.1.1	Realizzazione del Wrapper	19
3.1.2	Modifiche al Sistema GUIDO	20
3.2	CR2 – Aggiunta di una GUI per utilizzare TOAD all’interno di GUIDO	22
3.2.1	Implementazione della Modifica	22
3.2.2	Altri miglioramenti del tool GUIDO	28
3.3	CR3 - Estensione della GUI per la visualizzazione del grafo	29
3.4	CR4 - Dockerizzazione di TOAD e GUIDO	31

Elenco delle figure

2.1	Esempio di grafo prodotto da TOAD	12
3.1	GUIDO: Class Diagram con il nuovo Tool	21
3.2	Nuova sezione: Community Inspector	22
3.3	Form per l’inserimento della richiesta di analisi TOAD	23
3.4	Esempio di richiesta nello stato <code>PENDING</code>	24
3.5	Esempio di richiesta nello stato <code>STARTED</code>	25
3.6	Esempio di richiesta nello stato <code>FAILED</code> con messaggio di errore . . .	25
3.7	Esempio di richiesta nello stato <code>SUCCESS</code> con accesso ai risultati . . .	25
3.8	Visualizzazione del grafo TOAD all’interno di GUIDO	30

Elenco delle tabelle

1.1	GUIDO: Matrice delle Dipendenze	4
2.1	CR1: Starting Impact Set	7
2.2	CR1: Candidate Impact Set (CIS)	8
2.3	CR2 – Starting Impact Set	10
2.4	CR3 – Starting Impact Set	13
2.5	CR4 – Starting Impact Set	15
2.6	CR4 - Candidate Impact Set (CIS)	16

CAPITOLO 1

Introduzione

Il presente documento ha lo scopo di descrivere nel dettaglio l'attività di manutenzione evolutiva effettuata sul sistema GUIDO, con particolare riferimento all'analisi e all'implementazione delle modifiche richieste attraverso le Change Request (CR).

Per ciascuna CR, verrà presentato il processo di *Impact Analysis*, finalizzato a individuare i componenti coinvolti, le dipendenze esistenti e le potenziali implicazioni sul funzionamento globale del sistema. I risultati di tale analisi costituiranno la base per la pianificazione e l'esecuzione degli interventi necessari.

Successivamente, verranno descritte le strategie adottate per l'implementazione effettiva delle modifiche, con un focus sulle scelte progettuali effettuate, le motivazioni che le hanno guidate e le eventuali difficoltà riscontrate durante lo sviluppo. L'obiettivo è fornire una visione trasparente e tracciabile dell'intero processo di manutenzione, evidenziando l'impatto delle modifiche sul sistema esistente e la coerenza degli interventi con gli obiettivi funzionali prefissati.

1.1 Processo di Impact Analysis

Il processo di *Impact Analysis* adottato per le Change Request (CR) relative alla manutenzione evolutiva del sistema GUIDO si è focalizzato sull'individuazione e sulla valutazione degli elementi del sistema potenzialmente soggetti a modifiche o effetti collaterali a seguito degli interventi pianificati. Per garantire maggiore precisione ed efficacia, l'analisi è stata condotta esclusivamente sugli artefatti di codice sorgente. Tale scelta è stata motivata sia dalla disponibilità limitata di documentazione formale relativa ad altri artefatti (come requisiti, design e test), sia dalla volontà di evitare una propagazione di impatti eccessivamente arbitraria.

Il processo ha previsto la definizione e l'analisi di cinque insiemi fondamentali: *Starting Impact Set* (SIS), *Candidate Impact Set* (CIS), *Actual Impact Set* (AIS), *Discovered Impact Set* (DIS) e *False Positive Impact Set* (FPIS). Di seguito si riporta una descrizione di ciascun insieme e della metodologia utilizzata per il suo calcolo.

Starting Impact Set (SIS)

Il *Starting Impact Set* rappresenta il punto di partenza del processo di analisi e contiene i componenti che, in base alla natura della CR, si presume siano direttamente interessati dalle modifiche. Per la sua individuazione, si è fatto riferimento alle CR descritte nel *Pre-Maintenance Report* e, sulla base di tali informazioni, nonché delle conoscenze acquisite sul sistema attraverso il processo di reverse engineering, sono stati identificati i moduli e i file sorgente potenzialmente soggetti a intervento. Questo insieme costituisce l'input iniziale per la successiva fase di espansione.

Candidate Impact Set (CIS)

Il *Candidate Impact Set* contiene l'insieme dei componenti che potrebbero essere influenzati, direttamente o indirettamente, dai cambiamenti apportati ai componenti del SIS. Per costruirlo, si è fatto uso di apposite **matrici delle dipendenze**, nelle quali ogni cella (i, j) rappresenta il numero di chiamate a funzioni o classi del modulo j effettuate all'interno del modulo i . L'analisi è stata condotta in modo incrementale:

per ciascun componente appartenente al SIS sono state analizzate le sue dipendenze dirette. Ciascuna di esse è stata valutata in termini di probabilità di impatto, e, se ritenuta significativa, è stata inclusa nel CIS. Il processo è stato quindi reiterato su ciascun nuovo elemento incluso nell'insieme. Un esempio concreto di matrice delle dipendenze costruita per il backend di GUIDO è riportato nella Tabella 1.1.

Actual Impact Set (AIS)

L'*Actual Impact Set* rappresenta l'insieme effettivo dei componenti che sono stati modificati nel corso della manutenzione. Esso è stato determinato in maniera empirica a posteriori, confrontando le modifiche introdotte nel codice sorgente con i componenti inizialmente inclusi nel CIS. Questo insieme consente di valutare l'accuratezza e la completezza delle previsioni effettuate in fase di analisi.

Discovered Impact Set (DIS)

Il *Discovered Impact Set* raccoglie i componenti del sistema che sono stati impattati in fase di implementazione, ma che non erano stati identificati inizialmente nel CIS. Tali componenti possono emergere a seguito di vincoli architetturali, dipendenze implicite o comportamenti dinamici non evidenziati dall'analisi statica. L'identificazione del DIS rappresenta un indicatore importante di incompletezza nella fase di previsione dell'impatto.

False Positive Impact Set (FPIS)

Il *False Positive Impact Set* è composto dai componenti che, pur essendo stati inclusi nel CIS, non hanno subito alcuna modifica né sono risultati realmente affetti dalle CR. Questo insieme è utile per valutare l'accuratezza della strategia predittiva utilizzata e per identificare eventuali sovrastime nella propagazione dell'impatto.

Tabella 1.1: GUIDO: Matrice delle Dipendenze

File	loginGui.py	toolGui.py	cadocs_utils.py	intent_manager.py	cadocs_intent.py	customException.py	intent_resolver.py	tool_selector.py	tool_strategy.py	tools.py	CADOCS.py	model_selector.py	prediction_service.py	oauth.py	cadocs_messages.py	language_handler.py	utils.py	intent_web_service.py
loginGui.py	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
toolGui.py	1	0	0	1	5	0	0	0	0	0	0	0	0	0	3	0	0	1
cadocs_utils.py	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
intent_manager.py	0	0	1	0	4	0	0	0	0	0	0	0	5	0	0	0	0	0
cadocs_intent.py	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
customException.py	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
intent_resolver.py	0	0	0	0	5	0	0	2	0	2	0	0	0	0	0	0	0	0
tool_selector.py	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0
tool_strategy.py	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
tools.py	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
CADOCS.py	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
model_selector.py	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0
prediction_service.py	0	0	0	0	0	0	0	0	0	0	2	2	0	0	0	0	0	0
oauth.py	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
cadocs_messages.py	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0
language_handler.py	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
utils.py	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
intent_web_service.py	0	0	0	1	2	0	1	0	0	0	0	0	0	0	1	0	0	0

1.2 Metriche di Valutazione dell'Impact Analysis

Per valutare l'efficacia e l'affidabilità del processo di *Impact Analysis* condotto per ciascuna Change Request (CR), sono state adottate due metriche fondamentali: **precision** e **recall**. Tali misure consentono di quantificare la qualità della previsione dell'impatto rispetto alle modifiche effettivamente apportate, rappresentate dall'Actual Impact Set (AIS).

Precision

La *precision* misura la proporzione di elementi effettivamente impattati (cioè presenti nell'AIS) rispetto al totale dei componenti che erano stati indicati come potenzialmente impattati nel Candidate Impact Set (CIS). Essa esprime quindi la capacità dell'analisi di evitare falsi positivi, ovvero componenti identificati come a rischio ma che non sono stati realmente toccati dalla modifica.

$$\text{Precision} = \frac{|AIS \cap CIS|}{|CIS|}$$

Recall

La *recall* indica la proporzione di componenti impattati realmente (AIS) che sono stati effettivamente individuati in fase di analisi, ossia inclusi nel CIS. Essa misura la completezza dell'analisi, valutando in che misura i componenti critici sono stati correttamente previsti.

$$\text{Recall} = \frac{|AIS \cap CIS|}{|AIS|}$$

Un valore elevato di *precision* suggerisce che il CIS contiene pochi falsi positivi, mentre una *recall* elevata indica che l'analisi è stata in grado di coprire la maggior parte dei componenti modificati. L'obiettivo è ottenere un buon equilibrio tra le due metriche, riducendo al minimo sia le omissioni (falsi negativi) sia le sovrastime (falsi positivi).

2.1 CR1: Integrazione di TOAD in GUIDO

2.1.1 Obiettivi della CR

L'obiettivo della CR1 è estendere le funzionalità di GUIDO integrando al suo interno lo strumento TOAD. Questa change request riguarda principalmente il backend del sistema e prevede due attività principali:

- **Wrapping TOAD:** creazione di un wrapper per lo strumento TOAD che esponga un'API RESTful, consentendone l'utilizzo tramite richieste HTTP;
- **Aggiornamento del backend di GUIDO:** modifica del backend per gestire le nuove richieste, definire i nuovi endpoint e interfacciarsi con il wrapper di TOAD.

2.1.2 Starting Impact Set (SIS)

Analizzando il contenuto della CR1 e sfruttando le conoscenze acquisite durante il processo di reverse engineering, sono stati identificati i moduli nella Tabella 2.1 come candidati per lo Starting Impact Set (SIS). Si sottolinea che tale insieme comprende

esclusivamente componenti appartenenti a GUIDO: infatti, si prevede di poter realizzare il wrapper RESTful per TOAD senza apportare alcuna modifica ai componenti preesistenti dello strumento.

Componente Coinvolta	Operazioni
<code>intent_web_service.py</code>	Gestisce gli endpoint API esposti da GUIDO tramite Flask. Per l'implementazione della CR sarà necessario estendere tale componente per supportare le nuove richieste rivolte ai servizi offerti da TOAD.
<code>intent_handling.cadocs_intent.py</code>	Contiene gli enum relativi agli intent gestiti da GUIDO. Sarà necessario aggiungere nuovi intent per supportare quelli associati a TOAD.
<code>intent_handling.intent_resolver.py</code>	Seleziona il tool appropriato per ciascun intent. Dovrà essere aggiornato per includere la logica di selezione di TOAD.
<code>intent_handling.tools.py</code>	Contiene le implementazioni concrete dei tool disponibili in GUIDO. Verrà esteso con la nuova classe responsabile della comunicazione con TOAD tramite il wrapper REST.

Tabella 2.1: CR1: Starting Impact Set

2.1.3 Candidate Impact Set (CIS)

Attraverso un'analisi incrementale delle dipendenze dei componenti identificati nel SIS, è stato individuato un ulteriore modulo potenzialmente impattato, non rilevato nella fase iniziale. La Tabella 2.2 ne descrive il ruolo. Per chiarezza espositiva, nella tabella è riportato solo il nuovo modulo identificato, ma si ricorda che il CIS include anche tutti i componenti del SIS.

2.1.4 Actual Impact Set (AIS)

A seguito dell'implementazione della CR1, l'Actual Impact Set (AIS) è risultato coincidente con il Candidate Impact Set (CIS). Di conseguenza, sia il *Discovered Impact Set* (DIS), sia il *False Positive Impact Set* (FPIS) risultano vuoti. Questo risultato suggerisce che l'Impact Analysis è stata condotta in maniera efficace e completa. Tale esito positivo è favorito anche dalla modularità e dalla chiarezza con cui sono

Componente Coinvolta	Operazioni
service.cadocs_messages.py	Gestisce la formattazione delle risposte del chatbot CADDOCS. Sebbene inizialmente non sia stato identificato come candidato per il SIS, si è rilevato necessario includerlo poiché, per i tool non CADDOCS, il modulo inoltra comunque le risposte ricevute. Sarà necessario adeguarne il comportamento per supportare anche TOAD.

Tabella 2.2: CR1: Candidate Impact Set (CIS)

stati progettati i componenti del sistema GUIDO. Inoltre, è stata confermata l'ipotesi iniziale secondo cui la realizzazione del wrapper di TOAD non avrebbe influenzato componenti esistenti, dato che si tratta di un'aggiunta autonoma al sistema.

2.1.5 Metriche di Processo

Le metriche di valutazione del processo di Impact Analysis calcolate per la CR1 risultano le seguenti:

$$\mathbf{Recall} = \frac{|\mathbf{AIS} \cap \mathbf{CIS}|}{|\mathbf{AIS}|} = \frac{5}{5} = 1$$

$$\mathbf{Precision} = \frac{|\mathbf{AIS} \cap \mathbf{CIS}|}{|\mathbf{CIS}|} = \frac{5}{5} = 1$$

I valori massimi di *precision* e *recall* indicano un'ottima capacità predittiva del processo di Impact Analysis, confermando la correttezza delle assunzioni iniziali e l'accuratezza delle dipendenze considerate.

2.2 CR2: Aggiunta di una GUI per utilizzare TOAD all'interno di GUIDO

2.2.1 Obiettivi della Change Request

La Change Request **CR2** ha come obiettivo la modifica dell'interfaccia grafica di **GUIDO**, per consentire agli utenti l'inserimento delle richieste e la gestione delle analisi fornite dal tool **TOAD** in modo interattivo. Tale modifica ha lo scopo principale di integrare le funzionalità offerte da TOAD direttamente all'interno dell'interfaccia di **GUIDO**. Oltre all'introduzione delle nuove sezioni per la gestione delle richieste e dei risultati, abbiamo previsto ulteriori interventi sull'interfaccia complessiva della piattaforma, volti a migliorarne l'usabilità e la coerenza grafica.

La modifica è classificata come **ad alta priorità**, con uno sforzo stimato di livello medio. La mancata implementazione della CR2 comporterebbe una significativa limitazione della fruibilità del sistema, rendendo i dati prodotti da TOAD inaccessibili. Abbiamo stimato che le modifiche previste avranno un impatto localizzato sulla logica di presentazione e interazione lato client.

2.2.2 Starting Impact Set (SIS)

Analizzando dettagliatamente la CR2 e sfruttando le conoscenze acquisite durante il processo di reverse engineering, sono stati identificati come candidati per lo **Starting Impact Set (SIS)** i file presenti nella Tabella 2.3.

Componente Coinvolta	Operazioni
App.css	Gestisce lo stile globale dell'applicazione: è plausibile che debba essere aggiornato per integrare gli stili delle nuove sezioni grafiche.
App.js	Contiene la configurazione del routing principale: l'aggiunta di nuove pagine comporta la necessità di definire nuovi percorsi.
Menu.js	Gestisce il menu di navigazione: l'inserimento di nuove sezioni richiede un'estensione delle voci di menu disponibili.
menu_style.css	Gestisce lo stile del menu: eventuali modifiche strutturali (es. menu sticky) potrebbero impattare direttamente su questo file.
package.json	File di configurazione delle dipendenze del progetto: nuove funzionalità lato frontend potrebbero richiedere librerie aggiuntive.
index.html	File statico principale: il <code>title</code> dell'applicazione potrebbe essere aggiornato per riflettere correttamente il nome "GUIDO".
ChatbotPage.jsx	Componente UI esistente che utilizza pulsanti e form: potrebbe essere utile per uniformare lo stile con le nuove interfacce.
CountrySelector.jsx	Componente UI esistente che utilizza pulsanti e form: potrebbe essere utile per uniformare lo stile con le nuove interfacce.

Tabella 2.3: CR2 – Starting Impact Set

2.2.3 Candidate Impact Set (CIS)

Analizzando in maniera incrementale le dipendenze dei componenti identificati nello *Starting Impact Set*, non sono emerse ulteriori componenti da includere nel *Candidate Impact Set*. Pertanto, il **CIS** coincide esattamente con lo **SIS**, suggerendo che l'impatto previsto della modifica dovrebbe essere confinato ai moduli inizialmente individuati.

2.2.4 Actual Impact Set (AIS)

Al termine dell'implementazione della **CR2**, l'*Actual Impact Set (AIS)* è risultato pienamente sovrapponibile al *Candidate Impact Set (CIS)*. Di conseguenza:

- Il **Discovered Impact Set (DIS)** risulta vuoto: non sono emerse modifiche a componenti non precedentemente identificate.
- Il **False Positive Impact Set (FPIS)** risulta vuoto: tutte le componenti previste come potenzialmente impattate sono effettivamente state modificate.

Questo risultato conferma l'efficacia dell'analisi dell'impatto condotta in fase preliminare. Il merito è attribuibile, in larga parte, alla buona modularità e alla chiarezza architetturale dei componenti *front-end* della piattaforma **GUIDO**, che hanno permesso di circoscrivere con precisione l'impatto della modifica.

2.2.5 Metriche di Processo

Calcolando le Metriche del processo di Impact Analysis effettuato per la CR2, abbiamo ottenuto i seguenti risultati:

$$\mathbf{Recall} = \frac{|\mathbf{AIS} \cap \mathbf{CIS}|}{|\mathbf{AIS}|} = \frac{8}{8} = 1$$

$$\mathbf{Precision} = \frac{|\mathbf{AIS} \cap \mathbf{CIS}|}{|\mathbf{CIS}|} = \frac{8}{8} = 1$$

2.3 CR3: Estensione della GUI per la visualizzazione del grafo

2.3.1 Obiettivi della Change Request

La Change Request **CR3** ha come obiettivo il potenziamento della visualizzazione del grafo collaborativo fornito dal tool **TOAD**, già restituito come output strutturato in risposta alle richieste utente. In particolare, si intende trasformare tale grafo in un componente **interattivo, tridimensionale e navigabile**, al fine di migliorare l'efficacia comunicativa e la comprensione delle relazioni sociali tra i membri della community di sviluppo. L'approccio attuale adottato da TOAD si limita a una visualizzazione statica, spesso **congesta, poco leggibile e non esplorabile**, che rende difficile l'identificazione di informazioni rilevanti. A titolo esemplificativo, nella Figura 2.1, si riporta una rappresentazione standard prodotta da TOAD, da cui si evince la difficoltà di interpretazione causata dalla densità e sovrapposizione dei nodi.

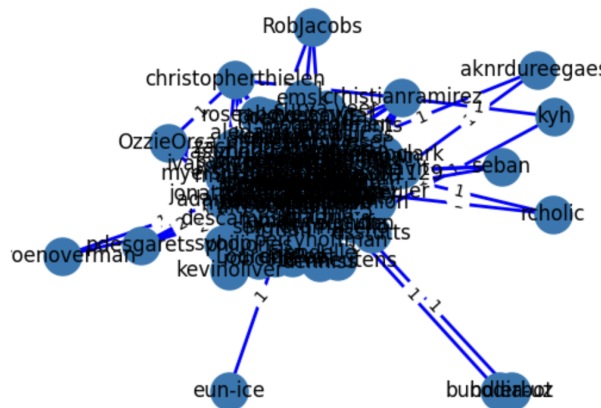


Figura 2.1: Esempio di grafo prodotto da TOAD

Attraverso l'introduzione di una rappresentazione grafica avanzata, la CR3 mira a:

- rendere visivamente evidenti le connessioni tra gli sviluppatori;
- agevolare l'identificazione di anomalie strutturali o punti di debolezza;
- aumentare l'accessibilità e l'intuitività del dato anche per utenti non tecnici.

Il nuovo componente verrà progettato per supportare interazioni dirette con i nodi e gli archi (ad es. selezione, ingrandimento, evidenziazione di cluster), sfruttando tecnologie di visualizzazione 3D e librerie grafiche dinamiche. Le modifiche interesseranno esclusivamente la logica di presentazione lato **frontend**, lasciando inalterata la struttura dati restituita da TOAD e gli altri elementi funzionali già introdotti nella precedente Change Request.

2.3.2 Starting Impact Set (SIS)

Analizzando dettagliatamente la CR3, sono stati identificati come candidati per lo **Starting Impact Set (SIS)** i moduli presentati nella Tabella 2.4.

Componente Coinvolta	Operazioni
App.js	Contiene la configurazione del routing principale: l'aggiunta di nuove pagine comporta la necessità di definire nuovi percorsi.
package.json	File di configurazione delle dipendenze del progetto: nuove funzionalità lato frontend potrebbero richiedere librerie aggiuntive.

Tabella 2.4: CR3 – Starting Impact Set

2.3.3 Candidate Impact Set (CIS)

Analizzando in maniera incrementale le dipendenze dei componenti identificati nello *Starting Impact Set*, non sono emerse ulteriori componenti da includere nel *Candidate Impact Set*. Pertanto, il **CIS** coincide esattamente con lo **SIS**, suggerendo anche in questo caso che l'impatto previsto della modifica è confinato ai moduli inizialmente individuati.

2.3.4 Actual Impact Set (AIS)

Al termine dell'implementazione della **CR3**, anche in questo caso, l'*Actual Impact Set (AIS)* è risultato pienamente sovrapponibile al *Candidate Impact Set (CIS)*. Di conseguenza:

- Il **Discovered Impact Set (DIS)** risulta vuoto: non sono emerse modifiche a componenti non precedentemente identificate.
- Il **False Positive Impact Set (FPIS)** risulta vuoto: tutte le componenti previste come potenzialmente impattate sono effettivamente state modificate.

Questo risultato è attribuibile principalmente al fatto che l'implementazione della CR2 ha permesso già di acquisire un'ottima conoscenza della struttura del Frontend di GUIDO, permettendo di identificare facilmente i componenti impattati dalla modifica. Inoltre per la CR3 sono state necessarie per lo più delle aggiunte di nuovi componenti dell'interfaccia grafica (poche modifiche a quelle esistenti), rendendo quindi l'impact analysis non troppo complessa.

2.3.5 Metriche di Processo

Calcolando le Metriche del processo di Impact Analysis effettuato per la CR3, abbiamo ottenuto i seguenti risultati:

$$\mathbf{Recall} = \frac{|\mathbf{AIS} \cap \mathbf{CIS}|}{|\mathbf{AIS}|} = \frac{2}{2} = 1$$

$$\mathbf{Precision} = \frac{|\mathbf{AIS} \cap \mathbf{CIS}|}{|\mathbf{CIS}|} = \frac{2}{2} = 1$$

2.4 CR4: Dockerizzazione TOAD e GUIDO

2.4.1 Obiettivi della Change Request

La Change Request CR4 mira a facilitare l'integrazione del tool TOAD all'interno del sistema GUIDO già esistente, sfruttando l'ambiente Docker in cui quest'ultimo era già configurato. A differenza di GUIDO, TOAD non era precedentemente dockerizzato: ciò rendeva l'utilizzo congiunto dei due strumenti complesso, richiedendo l'avvio separato di TOAD in locale. L'obiettivo della CR4 è stato dunque quello di integrare TOAD (insieme al suo wrapper sviluppato nella CR1) direttamente nel container di GUIDO, garantendo un'installazione semplificata, un avvio unificato e un utilizzo più fluido del sistema complessivo.

2.4.2 Starting Impact Set (SIS)

Analizzando dettagliatamente la CR4, sono stati identificati come candidati per lo **Starting Impact Set (SIS)** i moduli presentati nella Tabella 2.5.

Componente Coinvolta	Operazioni
Dockerfile (GUIDO)	File contenente le istruzioni necessarie alla costruzione dell'immagine Docker di GUIDO: verrà modificato per clonare tramite Git la repository del tool TOAD, installarne i requisiti e definire le variabili d'ambiente necessarie per l'integrazione.
docker-compose.yml (GUIDO)	File di configurazione dei container Docker da lanciare: sarà aggiornato per definire i servizi necessari per lanciare TOAD, Redis nello specifico, e per gestire l'interconnessione tra i container.
tasks.py (TOAD)	Modulo Python contenente le definizioni dei task per TOAD: verrà modificato per adattare il nome del task e alcuni percorsi relativi.
globe_data_reader.py (TOAD)	Modulo Python destinato a leggere informazioni utili al calcolo della geodispersione dei membri del team: verrà modificato per adattare il percorso relativo del file

Tabella 2.5: CR4 – Starting Impact Set

2.4.3 Candidate Impact Set (CIS)

Attraverso un'analisi incrementale delle dipendenze dei componenti identificati nello SIS, sono stati individuati ulteriori moduli potenzialmente impattati che non erano stati rilevati nella fase iniziale. La Tabella 2.6 ne descrive il ruolo. Per chiarezza espositiva, nella tabella sono riportati solo i nuovi moduli identificati, ma si ricorda che il CIS include anche tutti i componenti dello SIS.

Componente Coinvolta	Operazioni
dispersion_processor.py (TOAD)	Modulo Python responsabile del calcolo di metriche di dispersione. È stato identificato come impattato in quanto dipendente da <code>in_globe_data_reader.py</code> per l'accesso ai dati geografici.
geographical_retriever.py (TOAD)	Modulo Python di che si occupa del recupero dei dati geografici. Anche questo modulo è stato incluso nel CIS a causa della sua dipendenza da <code>in_globe_data_reader.py</code> .

Tabella 2.6: CR4 - Candidate Impact Set (CIS)

2.4.4 Actual Impact Set (AIS)

Al termine dell'implementazione della CR4, l'*Actual Impact Set (AIS)* ha mostrato una parziale sovrapposizione con il *Candidate Impact Set (CIS)*.

- Il **Discovered Impact Set (DIS)** non è risultato vuoto: è emersa la necessità di modificare il file `requirements.txt` a causa di un **conflitto di versione tra i pacchetti** richiesti dalle nuove configurazioni Dockerizzate di TOAD e GUIDO. Questo file non era stato precedentemente identificato nel SIS o nel CIS.
- Il **False Positive Impact Set (FPIS)** è risultato vuoto: tutte le componenti previste come potenzialmente impattate nel CIS sono state effettivamente modificate.

2.4.5 Metriche di Processo

Calcolando le Metriche del processo di Impact Analysis effettuato per la CR4, abbiamo ottenuto i seguenti risultati:

$$\mathbf{Recall} = \frac{|\mathbf{AIS} \cap \mathbf{CIS}|}{|\mathbf{AIS}|} = \frac{6}{7} \approx 0.857$$

$$\mathbf{Precision} = \frac{|\mathbf{AIS} \cap \mathbf{CIS}|}{|\mathbf{CIS}|} = \frac{6}{6} = 1$$

A differenza delle Change Request precedenti, non è stato possibile ottenere valori massimi per entrambe le metriche, in particolare per il recall, che risulta leggermente inferiore a 1. Ciò indica che una componente modificata durante l'implementazione non era stata precedentemente identificata nell'analisi di impatto. Questo risultato è principalmente dovuto alla limitata esperienza iniziale del team con la tecnologia Docker. In particolare, non era stata prevista l'incompatibilità tra la versione di una libreria utilizzata da TOAD e l'ambiente Docker in uso, situazione che ha richiesto la modifica del file requirements.txt durante l'integrazione.

Questo imprevisto ha mostrato i limiti dell'approccio adottato, evidenziando che la gestione delle dipendenze in ambienti containerizzati può introdurre complessità difficili da rilevare con la sola analisi statica del codice sorgente. L'esperienza ha rappresentato un'importante occasione di apprendimento, utile per migliorare le future analisi e prevedere con maggiore precisione l'impatto di modifiche infrastrutturali.

Implementazione delle Change Request

3.1 CR1 – Integrazione di TOAD in GUIDO

L'integrazione dello strumento TOAD all'interno di GUIDO ha richiesto la progettazione e lo sviluppo di un componente intermedio, ovvero un *wrapper*, in grado di esporre le funzionalità analitiche di TOAD tramite una API RESTful.

A causa della natura computazionalmente intensiva di TOAD, che in alcuni casi può richiedere anche più di 20 minuti per completare un'analisi, si è reso necessario progettare il sistema per supportare richieste asincrone. Per soddisfare tale esigenza, è stato adottato uno stack tecnologico composto da:

- **FastAPI**: framework leggero ed efficiente per la costruzione dell'interfaccia REST;
- **Celery**: sistema di task queue per l'esecuzione asincrona delle analisi in background;
- **Redis**: utilizzato sia come broker per Celery, sia come backend per la memorizzazione dello stato dei task.

3.1.1 Realizzazione del Wrapper

Il componente FastAPI sviluppato espone tre endpoint principali:

- `POST /analyze`: avvia un'analisi TOAD in background. Riceve in input l'autore, la repository e la data di fine analisi. La richiesta genera un task Celery e restituisce un `job_id` univoco;
- `GET /status/{job_id}`: consente di monitorare lo stato del task associato a un determinato job;
- `GET /result/{job_id}`: restituisce i risultati dell'analisi (pattern, metriche, grafo) una volta completata con successo, oppure un messaggio di errore in caso di fallimento.

All'avvio di una richiesta, il task Celery crea una struttura di input che simula l'interazione utente prevista da TOAD. In particolare, genera dinamicamente:

- un file `input.csv` contenente le informazioni sulla repository;
- un file `toad_stdin.txt` che simula l'input da tastiera per eseguire lo script `pattern_detection.py` di TOAD.

L'analisi viene eseguita tramite una chiamata a `subprocess.run()` ed è soggetta a un timeout massimo di 60 minuti. Durante l'esecuzione, Celery aggiorna lo stato del task attraverso i seguenti stati standard:

- `PENDING`: il task è in coda ma non è ancora stato avviato;
- `STARTED`: il task è in esecuzione;
- `SUCCESS`: il task è terminato con successo e i risultati sono disponibili;
- `FAILED`: il task è terminato ma con un errore.

TOAD, in caso di errore, non restituisce codici d'uscita specifici ma stampa messaggi descrittivi in console. Per intercettare tali situazioni, senza modificare il codice di TOAD, stdout ed stderr vengono catturati e analizzati alla ricerca di stringhe note (ad esempio: `There must be at least 100 commits`). In caso di rilevamento

di uno di questi messaggi, il task viene considerato fallito e lo stato aggiornato di conseguenza.

Al termine dell'analisi, se non vengono rilevati errori, il wrapper raccoglie i risultati prodotti da TOAD e memorizzati all'interno di file temporanei, tra cui i community pattern rilevati, le metriche calcolate e la struttura del grafo sociale. I dati vengono convertiti in formato JSON (eventualmente arricchiti con informazioni aggiuntive) e restituiti tramite l'endpoint `/result`. È prevista inoltre una pulizia automatica delle cartelle e dei file temporanei generati durante il processo.

3.1.2 Modifiche al Sistema GUIDO

Per permettere l'interazione tra GUIDO e il wrapper di TOAD, è stato necessario introdurre due nuovi intent, ovvero `CommunityInspectorAnalyze` (per richiedere l'avvio di un'analisi) e `CommunityInspectorResults` (per richiedere i risultati di un'analisi). L'integrazione è avvenuta attraverso l'estensione dell'enum `CadocsIntents`, che raccoglie gli intent riconosciuti da GUIDO. Sono state inoltre apportate modifiche al componente `intent_web_service.py` per consentire la gestione completa delle nuove richieste, distinguendo correttamente tra la fase di avvio dell'analisi e quella di interrogazione del risultato (oltre che permettere a GUIDO di continuare a soddisfare richieste relative agli altri intent già presenti).

L'architettura di GUIDO utilizza un *Strategy Design Pattern* per la gestione dei tool, che ha facilitato l'integrazione del nuovo strumento. In particolare:

- `Tool` è l'interfaccia astratta che definisce il metodo `execute_tool()`;
- `ToolSelector` è il contesto responsabile della selezione e attivazione del tool corretto;
- `CommunityInspectorTool` è la nuova implementazione concreta dell'interfaccia, dedicata alla comunicazione con il wrapper TOAD.

Nella Figura 3.1 è possibile osservare la porzione del Class Diagram modificata a seguito dell'integrazione del nuovo tool `CommunityInspectorTool`

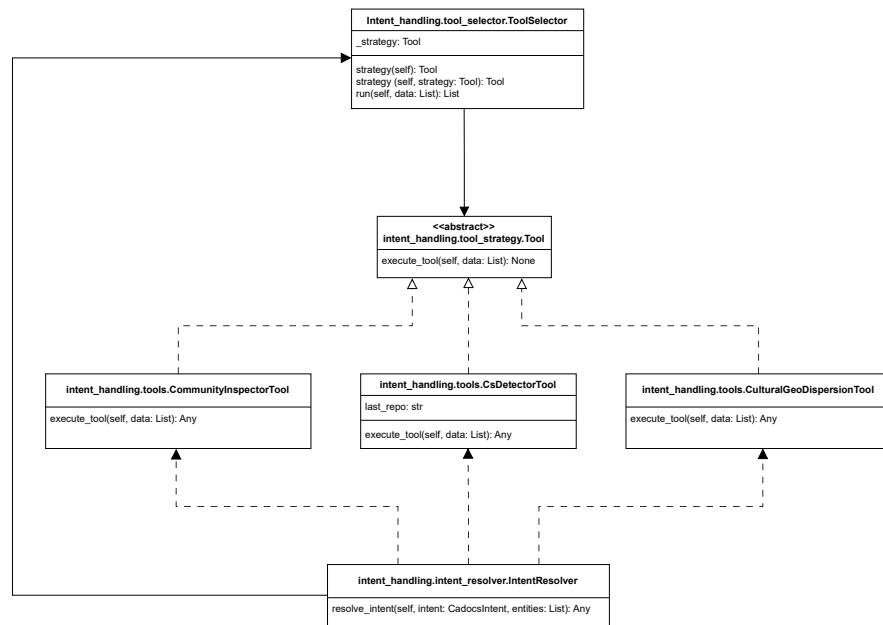


Figura 3.1: GUIDO: Class Diagram con il nuovo Tool

La classe `CommunityInspectorTool` gestisce entrambi gli intent:

- In presenza dell'intent `CommunityInspectorAnalyze`, la classe effettua una richiesta POST all'endpoint `/analyze` del wrapper, inviando i parametri richiesti (autore, repository, data di fine). Il wrapper restituisce un `job_id` che viene inoltrato all'utente.
- In presenza dell'intent `CommunityInspectorResults`, la classe interroga lo stato del job tramite GET `/status/{job_id}`. Se il task risulta completato, viene eseguita una seconda richiesta GET `/result/{job_id}` per recuperare i risultati; in caso contrario, viene restituito lo stato corrente.

Infine, sono stati aggiornati anche i moduli responsabili della risoluzione e gestione degli intent, assicurando che il nuovo comportamento si integrasse perfettamente nell'architettura esistente, senza alterare il funzionamento dei tool già presenti.

3.2 CR2 – Aggiunta di una GUI per utilizzare TOAD all'interno di GUIDO

3.2.1 Implementazione della Modifica

L'implementazione della Change Request **CR2** ha previsto un'estensione funzionale e grafica della piattaforma **GUIDO**, al fine di integrare le analisi di TOAD direttamente all'interno dell'interfaccia utente. L'obiettivo era quello di consentire agli utenti di interagire con il sistema in maniera semplice ed intuitiva.

Per rendere accessibile questa nuova funzionalità, è stato aggiornato il menù principale della piattaforma, includendo una nuova voce di navigazione che consente l'accesso a una sezione dedicata all'analisi delle community chiamata "Community inspector". All'interno di quest'area, l'utente ha la possibilità di scegliere tra due operazioni principali:

- **Inserimento della richiesta:** per avviare una nuova analisi TOAD fornendo i parametri richiesti;
- **Consultazione delle richieste:** per monitorare lo stato delle analisi avviate e accedere ai risultati disponibili.

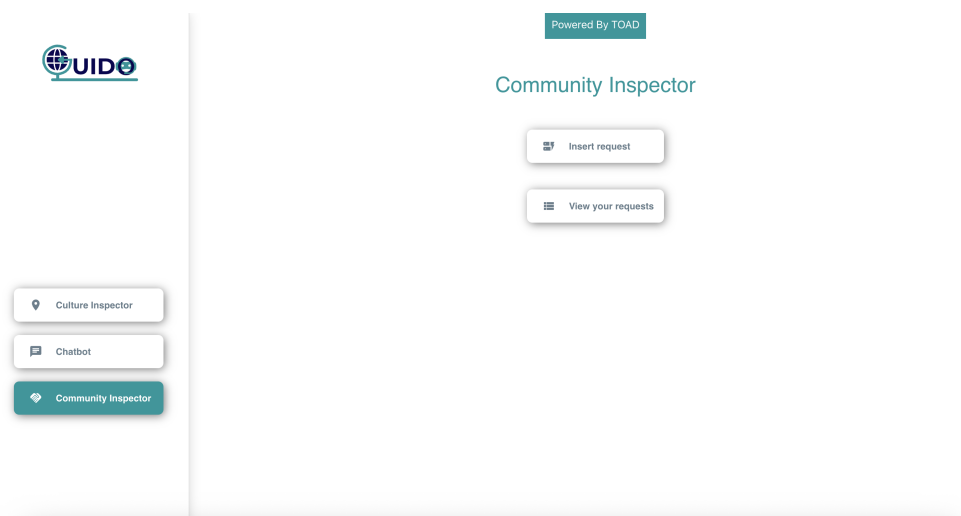


Figura 3.2: Nuova sezione: Community Inspector

Sezione: Inserimento della Richiesta ("Insert request")

È stata sviluppata una nuova sezione dedicata all'inserimento delle richieste di analisi TOAD. Tramite un form strutturato, l'utente può specificare:

- **Repository GitHub:** URL completo della repository da analizzare;
- **Data di fine:** utilizzata dal sistema per calcolare automaticamente l'intervallo di analisi (3 mesi precedenti).

Prima dell'invio della richiesta al sistema di backend (descritto nella **CR1**), vengono effettuati opportuni controlli di validazione sugli input. In particolare:

- viene verificato che l'URL inserito corrisponda al formato previsto per i repository GitHub (tramite espressioni regolari);
- viene verificato che la data selezionata sia una data valida e non futura.

Una volta superati i controlli, l'utente può avviare l'analisi. Se il backend restituisce un identificativo `job_id` valido, questo viene automaticamente salvato nella sessione dell'utente. A partire da quel momento, la sezione di inserimento viene disabilitata, evitando l'avvio di nuove richieste finché quella precedente non si è conclusa.

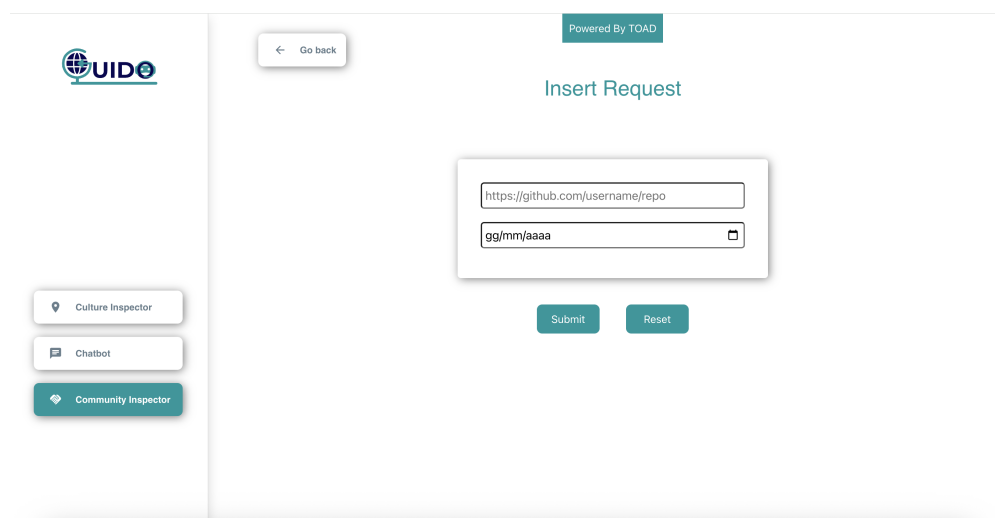
The screenshot shows the 'Insert Request' form within the GUIDO application. On the left is a sidebar with the GUIDO logo and three buttons: 'Culture Inspector', 'Chatbot', and 'Community Inspector'. The main area has a 'Go back' button at the top left and a 'Powered By TOAD' badge at the top right. The title 'Insert Request' is centered. Below it is a form with two input fields: the first contains 'https://github.com/username/repo' and the second contains 'gg/mm/aaaa' with a calendar icon. At the bottom of the form are 'Submit' and 'Reset' buttons.

Figura 3.3: Form per l'inserimento della richiesta di analisi TOAD

Sezione: Consultazione delle Richieste ("View your requests")

Una seconda sezione consente all'utente di visualizzare lo stato e i risultati delle richieste precedentemente effettuate. Per ogni richiesta associata al `job_id` salvato in sessione, viene mostrata una **card** che ne riassume lo stato attuale. I possibili stati sono:

- **PENDING:** la richiesta è stata accodata, in attesa di elaborazione;
- **STARTED:** l'elaborazione è attualmente in corso;
- **FAILED:** si è verificato un errore, mostrato direttamente nella card;
- **SUCCESS:** l'elaborazione è completata e i risultati sono disponibili.

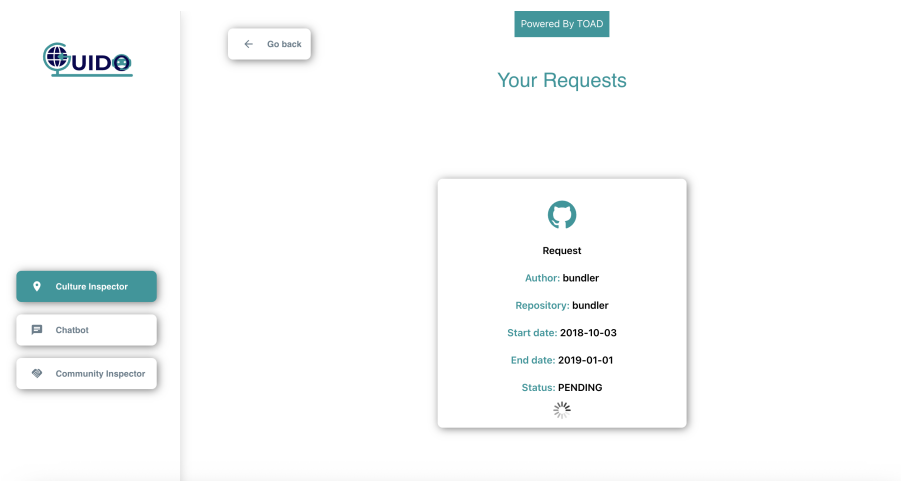


Figura 3.4: Esempio di richiesta nello stato PENDING

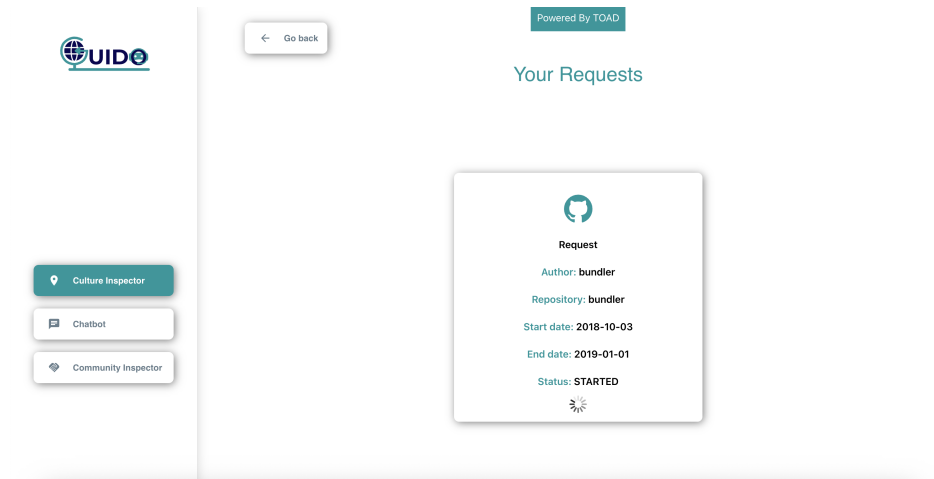


Figura 3.5: Esempio di richiesta nello stato STARTED

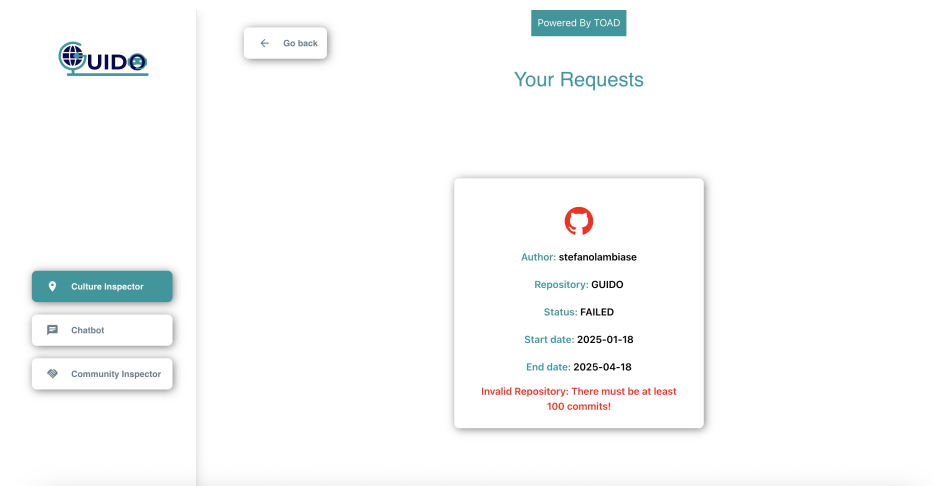


Figura 3.6: Esempio di richiesta nello stato FAILED con messaggio di errore

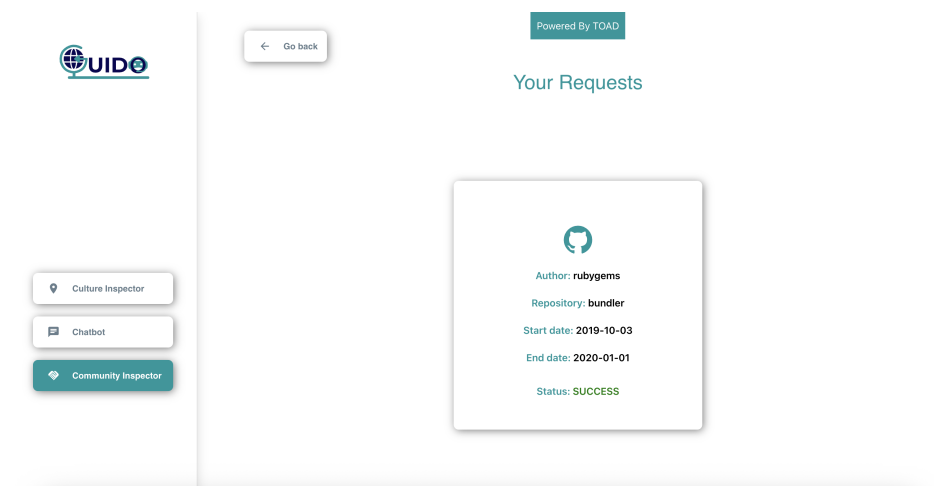


Figura 3.7: Esempio di richiesta nello stato SUCCESS con accesso ai risultati

Gestione dell'aggiornamento asincrono

Il sistema implementa un **polling automatico ogni 10 secondi** nella sezione di consultazione ("View your requests"), con lo scopo di verificare se lo stato della richiesta sia cambiato. Questo permette all'utente di ricevere aggiornamenti in tempo reale sull'avanzamento della propria richiesta elaborata da TOAD, senza dover ricaricare manualmente la pagina. Nel caso in cui una richiesta sia ancora in stato `PENDING` o `STARTED`, l'inserimento di nuove richieste viene temporaneamente disabilitato. In questo modo si garantisce la corretta serializzazione delle analisi asincrone e si evita la sovrapposizione di elaborazioni non gestite.

Tabelle Pattern e Metrics

Una volta conclusa l'elaborazione della richiesta dal tool TOAD, nella sezione di dettaglio della richiesta sarà quindi possibile consultare una serie di informazioni organizzate in delle **tabelle sovrapposte in trasparenza**, pensate per integrare le informazioni visive con dati strutturati e facilmente consultabili. Questi pannelli informativi includono:

- **Community Patterns:** una tabella strutturata mostra le configurazioni sociali rilevate all'interno della community analizzata, basandosi sulla tassonomia fornita da TOAD. Per ciascun pattern vengono riportati:
 - **Pattern:** il nome del modello sociale rilevato o atteso;
 - **Descrizione:** una definizione sintetica del pattern;
 - **Detected:** indicazione binaria (`Yes/No`) dell'effettivo rilevamento del pattern nella community.

I pattern considerati includono:

- **Informal Community (IC):** insiemi di persone parte della stessa organizzazione, unite da interessi comuni, con interazioni prevalentemente informali anche a distanza;

- **Community of Practice (CoP)**: gruppi che condividono problemi o passioni e sviluppano competenze interagendo frequentemente in una stessa geolocalizzazione;
 - **Formal Network (FN)**: membri selezionati formalmente e organizzati secondo strategie aziendali;
 - **Social Network (SN)**: macrostrutture organizzative che possono comprendere tutti gli altri pattern sociali;
 - **Informal Network (IN)**: reti deboli di contatti tra individui nello stesso contesto, basate sulla forza dei legami;
 - **Network of Practice (NoP)**: reti collaborative che connettono più Community of Practice, spesso geograficamente distanti;
 - **Formal Group (FG)**: gruppi organizzativi imposti da enti aziendali per finalità operative, tipicamente locali e strutturati;
 - **Project Team (PT)**: team con competenze complementari e obiettivi comuni, operanti secondo linee guida organizzative.
- **Metriche Analitiche**: sezione che raccoglie i principali indicatori numerici relativi alla struttura e alla dinamica della community. Le metriche includono:
 - **Dispersione (Dispersion)**: misura la distribuzione dei membri nei sottogruppi, indicando quanto la community sia frammentata;
 - **Formalità (Formality)**: valuta la struttura gerarchica delle comunicazioni, misurando quanto l'organizzazione sia centralizzata;
 - **Engagement**: rappresenta il livello medio di interazione tra i membri;
 - **Struttura (Structure)**: esprime la coesione del grafo in termini topologici (densità, clustering coefficient);
 - **Longevità (Longevity)**: calcola la stabilità dei legami sociali in un arco temporale di tre mesi.

Questi contenuti, accessibili con immediatezza attraverso un'interfaccia coerente e integrata, completano il processo di visualizzazione delle analisi TOAD, migliorando

la comprensione e l'usabilità del sistema anche per utenti privi di conoscenze tecniche avanzate.

3.2.2 Altri miglioramenti del tool GUIDO

Oltre all'integrazione con il backend TOAD, sono stati apportati diversi miglioramenti grafici all'interfaccia generale della piattaforma, tra cui:

- il menù principale è stato reso *sticky*, in modo da rimanere sempre visibile durante la navigazione;
- l'aspetto e il comportamento dei pulsanti sono stati uniformati in tutto il tool, al fine di garantire coerenza grafica e usabilità;
- il `title` della pagina, inizialmente impostato su *"React App"*, è stato modificato in GUIDO per mantenere coerenza con il nome del tool.

3.3 CR3 - Estensione della GUI per la visualizzazione del grafo

L'implementazione della **Change Request CR3** ha previsto un'estensione funzionale dell'interfaccia di **GUiDO**, volta a migliorare la comprensione e l'usabilità del grafo prodotto dal tool TOAD. In particolare, è stato deciso di rendere il grafo **interattivo e tridimensionale**, al fine di facilitare la lettura delle relazioni tra i membri della community analizzata. Per ottenere questo risultato è stata integrata nella piattaforma la libreria `ForceGraph3D`, una soluzione JavaScript ad alte prestazioni che consente di costruire grafi tridimensionali dinamici e personalizzabili.

Funzionalità offerte

La visualizzazione tramite `ForceGraph3D` offre all'utente le seguenti funzionalità interattive:

- **Zoom in/out:** possibilità di ingrandire o ridurre la visualizzazione del grafo con il mouse o il trackpad;
- **Rotazione libera:** possibilità di ruotare l'intero grafo lungo i tre assi spaziali per esplorare meglio le connessioni;
- **Pan dinamico:** spostamento del grafo nell'area di visualizzazione senza perdere il contesto;
- **Tooltip interattivi:** passando con il cursore su un nodo è possibile visualizzare informazioni contestuali (es. nome utente GitHub).

Questa scelta tecnologica ha migliorato notevolmente la leggibilità del grafo, risolvendo le criticità precedentemente riscontrate nella versione statica fornita da TOAD, che risultava complessa da interpretare.

Gestione della distanza tra i nodi

Durante lo sviluppo dell'interfaccia grafica tridimensionale per la visualizzazione del grafo, si è evidenziata la necessità di migliorare la leggibilità della struttura collabo-

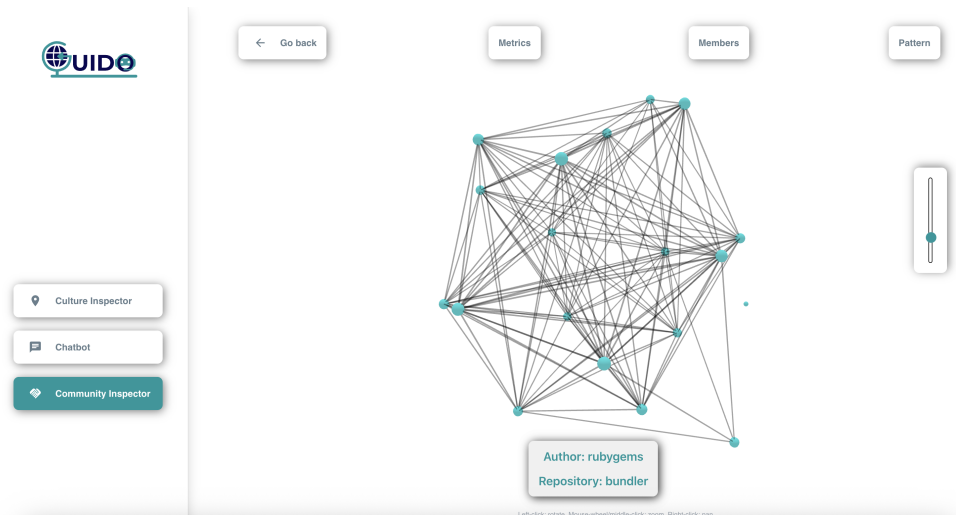


Figura 3.8: Visualizzazione del grafo TOAD all'interno di GUIDO

rativa tra i membri della community. La libreria `ForceGraph3D`, pur offrendo una visualizzazione interattiva avanzata (supportando operazioni di zoom, rotazione e panoramica), non include nativamente la possibilità di modificare dinamicamente la distanza tra i nodi.

Per rispondere a tale esigenza, è stato integrato un meccanismo custom che consente all'utente di regolare in tempo reale il parametro di attrazione/repulsione tra i nodi. Attraverso un'apposita interfaccia di controllo, è possibile aumentare o diminuire la distanza relativa, migliorando così l'esplorazione visiva del grafo in contesti con un alto numero di connessioni o nodi vicini. Questa funzionalità ha contribuito a rendere la rappresentazione della community più comprensibile e fruibile, permettendo di evidenziare cluster, sottogruppi e configurazioni strutturali che altrimenti risulterebbero sovrapposte o difficilmente interpretabili.

Membri della Community

Per completare tutte le informazioni fornite dall'analisi, oltre al grafo è stata aggiunta una finestra che permette di visualizzare l'elenco degli sviluppatori attivi nella repository analizzata, in cui per ciascun membro sono mostrati il nome utente GitHub e un collegamento ipertestuale diretto al rispettivo profilo.

3.4 CR4 - Dockerizzazione di TOAD e GUIDO

L'implementazione della Change Request CR4 ha avuto come obiettivo primario l'integrazione di TOAD, inizialmente non dockerizzato, all'interno dell'ambiente containerizzato già esistente di GUIDO. Questa operazione è stata fondamentale per migliorare la **portabilità, la scalabilità e la riproducibilità** degli ambienti di sviluppo e di esecuzione di entrambi i sistemi. Incapsulando i due tool in **container Docker**, abbiamo isolato le dipendenze e garantito un comportamento coerente su diverse piattaforme, eliminando le problematiche legate all'installazione manuale e alla gestione delle configurazioni.

Architettura della Dockerizzazione e integrazione

Per integrare TOAD nell'ambiente di GUIDO e allinearci alla gestione degli altri tool esterni (come `csDetector` e `culture-inspector`), abbiamo adottato un approccio che prevede l'inclusione di TOAD direttamente nel container di GUIDO. Questo ha richiesto diverse modifiche chiave:

- Il **Dockerfile di GUIDO** è stato modificato per incorporare TOAD direttamente nell'immagine. Questo ha incluso la clonazione del repository di TOAD tramite Git, l'installazione delle sue dipendenze specifiche e la definizione delle variabili d'ambiente necessarie per il suo corretto funzionamento. Inoltre, sono stati modificati gli script di lancio per avviare sia Celery che FastAPI, componenti essenziali per il wrapper di TOAD.
- Il file **`docker-compose.yml`** è stato aggiornato per gestire i servizi dei container. In particolare, è stato aggiunto Redis come servizio separato, indispensabile per il funzionamento di TOAD, e configurato sulla stessa rete condivisa con il backend di GUIDO per garantire una comunicazione fluida.

Problematiche riscontrate

Durante il processo di dockerizzazione abbiamo incontrato alcune criticità che hanno rallentato lo sviluppo e messo alla prova le nostre capacità di adattamento:

- **Imprevisti legati ai server Ubuntu:** una difficoltà inattesa ha riguardato l'indisponibilità temporanea di alcuni server Ubuntu da cui Docker scarica i pacchetti durante la fase di build dell'immagine. Questo problema, esterno al nostro controllo, ha causato il blocco delle attività per quasi un'intera giornata. Si è trattato di un evento non preventivato nella pianificazione iniziale, che ci ha fatto riflettere sull'importanza di prevedere margini di sicurezza anche per fattori infrastrutturali esterni.
- **Gestione di ambienti virtuali separati:** per garantire la compatibilità delle dipendenze, è stato necessario configurare due ambienti virtuali distinti all'interno dello stesso container. Infatti, TOAD richiede Python 3.11, mentre GUIDO è progettato per operare con Python 3.8. Questo approccio ha permesso di mantenere isolate le due configurazioni e ha evitato conflitti tra le dipendenze dei due strumenti, seppur al prezzo di una maggiore complessità nella configurazione e nella gestione dell'ambiente.
- **Incompatibilità non previste:** durante l'integrazione di TOAD nel container di GUIDO, abbiamo riscontrato un'incompatibilità tra una specifica libreria di TOAD e l'ambiente Docker adottato. Questo tipo di problema non era stato previsto in fase di Impact Analysis, soprattutto a causa della nostra limitata esperienza con Docker. Fortunatamente, l'aggiornamento della versione della libreria in questione ha risolto il problema senza introdurre regressioni nel comportamento di TOAD, permettendoci di completare l'integrazione senza compromettere la stabilità dello strumento.

Nonostante le difficoltà incontrate, l'integrazione di TOAD nell'ambiente Docker di GUIDO è stata completata con successo, risultando in un sistema complessivamente più stabile, portabile e agevole da gestire.