_SYSTE2 _SYSTE3 _SYSTE4 _SYSTE5 _SYSTE6 _SYSTE7 _SYSTE8 _SYSTE9 _SYSTE10 _SYSTE11 _SYSTE12 _SYSTE13 _SYSTE14 _SYSTE15 _SYSTE16 _SYSTE17

UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Magistrale in Informatica

TEST CASE SPECIFICATION

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

Elenco delle tabelle

CAPITOLO 1

Introduzione

Il presente documento è dedicato alla **Test Case Specification**, ovvero alla descrizione dettagliata dei casi di test sviluppati per la validazione del requisito funzionale **RF5: Get Request** del sistema **GUiDO**. Questa attività si basa sulle analisi condotte nel capitolo precedente, relativo al **Test Design**, in cui è stata impiegata la tecnica formale del **Category Partitioning** per strutturare e suddividere il dominio di input in categorie e scelte significative.

Ogni caso di test specificato in questa sezione è costruito a partire da un particolare *Test Frame*, ovvero una combinazione coerente di categorie e scelte input/output, in modo da coprire tutte le principali configurazioni operative del sistema, comprese situazioni normali e condizioni limite o di errore.

Per ciascun caso, vengono documentati i seguenti elementi:

- **Test Case Id**: identificativo univoco del test;
- **Obiettivo**: descrizione sintetica della condizione che il test vuole verificare;
- **Test Frame**: rappresentazione delle scelte derivate dal Category Partitioning, con indicazione delle categorie rilevanti per quel caso;

- **Ambiente di test**: configurazione tecnica (browser, sistema, risoluzione, strumenti);
- **Pre-condizione**: stato iniziale richiesto del sistema per eseguire il test;
- **Input**: valori in input inseriti dall'utente (es. URL e data fine);
- **Flusso di eventi**: sequenza di azioni che l'utente compie durante l'esecuzione del test;
- **Oracolo**: risultato atteso, specificato come risposta del sistema osservabile.

Ogni test è pensato per essere autonomo e ripetibile, contribuendo in modo strutturato alla verifica completa del comportamento del sistema in relazione al requisito **RF5**. Le tabelle seguenti illustrano, per ciascun test case, le caratteristiche sopra elencate.

CAPITOLO 2

Test case specification

2.1 System Test: Get request functionality [RF5]

2.1.1 Validazione URL GitHub (TEST_SYSTE1 - URL non valido)

Tabella 2.1: Validazione URL GitHub (TEST_SYSTE1)

Test Case Id	TEST_SYSTE1
Obiettivo	Verificare che, in caso di URL GitHub malformato e data correttamente popolata nel passato, venga mostrato l'errore corretto nel form di richiesta.

Test Frame	Funzionalità: insertRequest • URL = Non valido: senza autore e repository [RA1] • URL = Valido: con https://github.com/ [RH2] • URL $\neq \epsilon$ [RE2] • Data Fine = Valida [DF3] • Commit non validi (Repository) (Non elaborato) [CO3] • Milestone validi (Repository) (Non elaborato) [MI3] • Membri validi (Repository) (Non elaborato) [ME3] • Geodispersione community valida (Repository) (Non elaborato) [GE3] • Esecuzione conclusa con successo (Non elaborato) [EC4] • Non ci sono altri problemi legati alla repository (Non elaborato) [OT3]

Ambiente di test	• Browser: Firefox • Sistema: localhost • Risoluzione: 1290x828 • Versione Selenium: v4+
Pre-Condizione	L'utente ha installato GUIDO e si trova sulla pagina iniziale
Input	• Repository: `http://github.com/user` • Data fine: `29/06/2025`
Flusso di eventi	1. L'utente clicca sulla sezione "Community Inspector" 2. L'utente clicca su "Insert Request" 3. L'utente inserisce in input la repository 4. L'utente inserisce in input la data fine 5. Clicca su "Aggiungi richiesta"
Oracolo	Il sistema mostra il messaggio di errore: *"Please enter a valid GitHub repository URL (https://github.com/username/repo)"*, posizionato all'interno del form.

2.1.2 Validazione URL GitHub (TEST_SYSTE2 - URL non valido)

Tabella 2.2: Validazione URL GitHub (TEST_SYSTE2)

Test Case Id	TEST_SYSTE2
Obiettivo	Verificare che, in caso di URL GitHub malformato e data correttamente popolata nel passato, venga mostrato l'errore corretto nel form di richiesta.

Test Frame	Funzionalità: insertRequest • URL = Valido: con autore e repository [RA2] • URL = Non valido: senza https://github.com/ [RH1] • URL $\neq \epsilon$ [RE2] • Data Fine = Valida [DF3] • Commit non validi (Repository) (Non elaborato) [CO3] • Milestone validi (Repository) (Non elaborato) [MI3] • Membri validi (Repository) (Non elaborato) [ME3] • Geodispersione community valida (Repository) (Non elaborato) [GE3] • Esecuzione conclusa con successo (Non elaborato) [EC4] • Non ci sono altri problemi legati alla repository (Non elaborato) [OT3]

Ambiente di test	• Browser: Firefox • Sistema: localhost • Risoluzione: 1290x828 • Versione Selenium: v4+
Pre-Condizione	L'utente ha installato GUIDO e si trova sulla pagina iniziale
Input	• Repository: `http://google.com/user/repo` • Data fine: `29/06/2025`
Flusso di eventi	1. L'utente clicca sulla sezione "Community Inspector" 2. L'utente clicca su "Insert Request" 3. L'utente inserisce in input la repository 4. L'utente inserisce in input la data fine 5. Clicca su "Aggiungi richiesta"
Oracolo	Il sistema mostra il messaggio di errore: *"Please enter a valid GitHub repository URL (https://github.com/username/repo)"*, posizionato all'interno del form.

2.1.3 Validazione URL GitHub (TEST_SYSTE3 - URL non valido)

Tabella 2.3: Validazione URL GitHub (TEST_SYSTE3)

Test Case Id	TEST_SYSTE3
Obiettivo	Verificare che, in caso di URL GitHub malformato e data correttamente popolata nel passato, venga mostrato l'errore corretto nel form di richiesta.
Test Frame	Funzionalità: insertRequest • URL = Valido: con autore e repository [RA2] • URL = Valido: con https://github.com/ [RH2] • URL = ϵ [RE1] • Data Fine = Valida [DF3] • Commit non validi (Repository) (Non elaborato) [CO3] • Milestone validi (Repository) (Non elaborato) [MI3] • Membri validi (Repository) (Non elaborato) [ME3] • Geodispersione community valida (Repository) (Non elaborato) [GE3] • Esecuzione conclusa con successo (Non elaborato) [EC4] • Non ci sono altri problemi legati alla repository (Non elaborato) [OT3]

Ambiente di test	• Browser: Firefox • Sistema: localhost • Risoluzione: 1290x828 • Versione Selenium: v4+
Pre-Condizione	L'utente ha installato GUIDO e si trova sulla pagina iniziale
Input	• Repository: ϵ • Data fine: `29/06/2025`
Flusso di eventi	1. L'utente clicca sulla sezione "Community Inspector" 2. L'utente clicca su "Insert Request" 3. L'utente inserisce in input la repository 4. L'utente inserisce in input la data fine 5. Clicca su "Aggiungi richiesta"
Oracolo	Il sistema mostra il messaggio di errore: *"Please enter a valid GitHub repository URL (https://github.com/username/repo)"*, posizionato all'interno del form.

2.1.4 Validazione URL GitHub (TEST_SYSTE4 - Data & URL Validi)

Tabella 2.4: Validazione URL GitHub (TEST_SYSTE4)

Test Case Id	TEST_SYSTE4
Obiettivo	Verificare che, in caso di URL GitHub ben formato e data correttamente popolata nel passato, la richiesta venga accettata.

Test Frame	Funzionalità: insertRequest • URL = Valido: con autore e repository [RA2] • URL = Valido: con https://github.com/ [RH2] • URL $\neq \epsilon$ [RE2] • Data Fine = Valida [DF3] • Commit non validi (Repository) (Non elaborato) [CO3] • Milestone validi (Repository) (Non elaborato) [MI3] • Membri validi (Repository) (Non elaborato) [ME3] • Geodispersione community valida (Repository) (Non elaborato) [GE3] • Esecuzione conclusa con successo (Non elaborato) [EC4] • Non ci sono altri problemi legati alla repository (Non elaborato) [OT3]
Ambiente di test	• Browser: Firefox • Sistema: localhost • Risoluzione: 1290x828 • Versione Selenium: v4+

Pre-Condizione	L'utente ha installato GUIDO e si trova sulla pagina iniziale
Input	• Repository: `https://github.com/rubygems/bundler` • Data fine: `01/01/2019`
Flusso di eventi	1. L'utente clicca sulla sezione "Community Inspector" 2. L'utente clicca su "Insert Request" 3. L'utente inserisce in input la repository 4. L'utente inserisce in input la data fine 5. Clicca su "Aggiungi richiesta"
Oracolo	Il sistema mostra il messaggio di conferma di accettazione della richiesta: *"Request sent successfully!"*, posizionato in una modale di successo.

2.1.5 Validazione Data fine (TEST_SYSTE5 - Data non valida)

Tabella 2.5: Validazione Data fine (TEST_SYSTE5)

Test Case Id	TEST_SYSTE5
Obiettivo	Verificare che, in caso di URL GitHub ben formato e data malformata, venga mostrato l'errore corretto nel form di richiesta.
Test Frame	Funzionalità: insertRequest • URL = Valido: con autore e repository [RA2] • URL = Valido: con https://github.com/ [RH2] • URL $\neq \epsilon$ [RE2] • Data = Non valida [DF1] • Commit non validi (Repository) (Non elaborato) [CO3] • Milestone validi (Repository) (Non elaborato) [MI3] • Membri validi (Repository) (Non elaborato) [ME3] • Geodispersione community valida (Repository) (Non elaborato) [GE3] • Esecuzione conclusa con successo (Non elaborato) [EC4] • Non ci sono altri problemi legati alla repository (Non elaborato) [OT3]

Ambiente di test	• Browser: Firefox • Sistema: localhost • Risoluzione: 1290x828 • Versione Selenium: v4+
Pre-Condizione	L'utente ha installato GUIDO e si trova sulla pagina iniziale
Input	• Repository: `https://github.com/rubygems/bundler` • Data fine: `29/06/2099`
Flusso di eventi	1. L'utente clicca sulla sezione "Community Inspector" 2. L'utente clicca su "Insert Request" 3. L'utente inserisce in input la repository 4. L'utente inserisce in input la data fine 5. Clicca su "Aggiungi richiesta"
Oracolo	Il sistema mostra il messaggio di errore: *"The date must be today or a date in the past"*, posizionato all'interno del form.

2.1.6 Validazione Data fine (TEST_SYSTE6 - Data non valida)

Tabella 2.6: Validazione Data fine (TEST_SYSTE6)

Test Case Id	TEST_SYSTE6
Obiettivo	Verificare che, in caso di URL GitHub ben formato e data assente, venga mostrato l'errore corretto nel form di richiesta.
Test Frame	Funzionalità: insertRequest • URL = Valido: con autore e repository [RA2] • URL = Valido: con https://github.com/ [RH2] • URL $\neq \epsilon$ [RE2] • Data Fine = Non valida [DF2] • Commit non validi (Repository) (Non elaborato) [CO3] • Milestone validi (Repository) (Non elaborato) [MI3] • Membri validi (Repository) (Non elaborato) [ME3] • Geodispersione community valida (Repository) (Non elaborato) [GE3] • Esecuzione conclusa con successo (Non elaborato) [EC4] • Non ci sono altri problemi legati alla repository (Non elaborato) [OT3]

Ambiente di test	• Browser: Firefox • Sistema: localhost • Risoluzione: 1290x828 • Versione Selenium: v4+
Pre-Condizione	L'utente ha installato GUIDO e si trova sulla pagina iniziale
Input	• Repository: `https://github.com/rubygems/bundler` • Data fine: ϵ
Flusso di eventi	1. L'utente clicca sulla sezione "Community Inspector" 2. L'utente clicca su "Insert Request" 3. L'utente inserisce in input la repository 4. L'utente inserisce in input la data fine 5. Clicca su "Aggiungi richiesta"
Oracolo	Il sistema mostra il messaggio di errore: *"The date must be today or a date in the past"*, posizionato all'interno del form.

2.1.7 Validazione Repository (TEST_SYSTE7 - Repository valida)

Tabella 2.7: Validazione Repository (TEST_SYSTE7)

Test Case Id	TEST_SYSTE7
Obiettivo	Verificare che, in caso di richiesta accettata, nel caso in cui la repository rispetti tutti i criteri di correttezza il sistema mostri all'utente la richiesta elaborata con successo.
Test Frame	Funzionalità: getRequest • URL = Valido: con autore e repository [RA2] • URL = Valido: con https://github.com/ [RH2] • URL $\neq \epsilon$ [RE2] • Data Fine = Valida [DF3] • Commit non validi (Repository) (> 100) [CO2] • Milestone validi (Repository) ($>$ 0) [MI2] • Membri validi (Repository) ($\geq$ 2) [ME2] • Geodispersione community valida (Repository) [GE2] • Esecuzione conclusa con successo [EC3] • Non ci sono altri problemi legati alla repository [OT2]

Ambiente di test	• Browser: Firefox • Sistema: localhost • Risoluzione: 1290x828 • Versione Selenium: v4+
Pre-Condizione	L'utente ha installato GUIDO e si trova sulla pagina iniziale.
Input	• Repository: `https://github.com/rubygems/bundler` • Data fine: `01/01/2019`
Flusso di eventi	1. L'utente clicca sulla sezione "Community Inspector" 2. L'utente clicca su "Insert Request" 3. L'utente inserisce in input la repository 4. L'utente inserisce in input la data fine 5. Clicca su "Aggiungi richiesta" 6. L'utente entra nell'area "View your requests"
Oracolo	Il sistema mostra la richiesta conclusa con successo nell'apposita sezione.

2.1.8 Validazione Repository (TEST_SYSTE8 - Repository con numero di commit non validi)

Tabella 2.8: Validazione Repository (TEST_SYSTE8)

Test Case Id	TEST_SYSTE8
Obiettivo	Verificare che, in caso di richiesta accettata, nel caso in cui la repository abbia #commit ≤ 100, venga mostrato l'errore corretto nell'apposita sezione.
Test Frame	Funzionalità: getRequest • URL = Valido: con autore e repository [RA2] • URL = Valido: con https://github.com/ [RH2] • URL $\neq \epsilon$ [RE2] • Data Fine = Valida [DF3] • Commit non validi (Repository) (≤ 100) [CO1] • Milestone validi (Repository) (> 0) [MI2] • Membri validi (Repository) (≥ 2) • Geodispersione community valida (Repository) [GE2]

Ambiente di test	• Browser: Firefox • Sistema: localhost • Risoluzione: 1290x828 • Versione Selenium: v4+
Pre-Condizione	L'utente ha installato GUIDO e si trova sulla pagina iniziale.
Input	• Repository: `https://github.com/StefanoLambiase/GUIDO` • Data fine: `01/01/2019`
Flusso di eventi	1. L'utente clicca sulla sezione "Community Inspector" 2. L'utente clicca su "Insert Request" 3. L'utente inserisce in input la repository 4. L'utente inserisce in input la data fine 5. Clicca su "Aggiungi richiesta" 6. L'utente entra nell'area "View your requests"
Oracolo	Il sistema mostra il messaggio di errore: *"Invalid Repository: There must be at least 100 commits!"*, nell'apposita sezione.

2.1.9 Validazione Repository (TEST_SYSTE9 - Repository con numero di membri non validi)

Tabella 2.9: Validazione Repository (TEST_SYSTE9)

Test Case Id	TEST_SYSTE9
Obiettivo	Verificare che, in caso di richiesta accettata, nel caso in cui la repository abbia #membri < 2, venga mostrato l'errore corretto nell'apposita sezione.
Test Frame	Funzionalità: getRequest • URL = Valido: con autore e repository [RA2] • URL = Valido: con https://github.com/ [RH2] • URL $\neq \epsilon$ [RE2] • Data Fine = Valida [UR4] • Commit Validi (Repository) (> 100) [CO2] • Milestone validi (Repository) (> 0) [MI2] • Membri non validi (Repository) (< 2) [ME1] • Geodispersione community valida (Repository) [GE2] • Esecuzione conclusa con errore [EC1] • Non ci sono altri problemi legati alla repository [OT2]

Ambiente di test	• Browser: Firefox • Sistema: localhost • Risoluzione: 1290x828 • Versione Selenium: v4+
Pre-Condizione	L'utente ha installato GUIDO e si trova sulla pagina iniziale.
Input	• Repository: `https://github.com/legames/legames` • Data fine: `01/01/2019`
Flusso di eventi	1. L'utente clicca sulla sezione "Community Inspector" 2. L'utente clicca su "Insert Request" 3. L'utente inserisce in input la repository 4. L'utente inserisce in input la data fine 5. Clicca su "Aggiungi richiesta" 6. L'utente entra nell'area "View your requests"
Oracolo	Il sistema mostra il messaggio di errore: *"Invalid Repository: Not enough members (min. 2)!"*, nell'apposita sezione.

2.1.10 Validazione Repository (TEST_SYSTE10 - Repository con numero di milestones non validi)

Tabella 2.10: Validazione Repository (TEST_SYSTE10)

Test Case Id	TEST_SYSTE10
Obiettivo	Verificare che, in caso di richiesta accettata, nel caso in cui la repository abbia #milestones < 2, venga mostrato l'errore corretto nell'apposita sezione.
Test Frame	Funzionalità: getRequest • URL = Valido: con autore e repository [RA2] • URL = Valido: con https://github.com/ [RH2] • URL $\neq \epsilon$ [RE2] • Data Fine = Valida [DF3] • Commit Validi (Repository) (> 100) [CO2] • Milestone non validi (Repository) (< 1) [MI1] • Membri validi (Repository) (≥ 2) [ME2] • Geodispersione community valida (Repository) [GE2] • Esecuzione conclusa con errore [EC1] • Non ci sono altri problemi legati alla repository [OT2]

Ambiente di test	• Browser: Firefox • Sistema: localhost • Risoluzione: 1290x828 • Versione Selenium: v4+
Pre-Condizione	L'utente ha installato GUIDO e si trova sulla pagina iniziale.
Input	• Repository: `ttps://github.com/FrancescoTorino1999/angular` • Data fine: `01/01/2019`
Flusso di eventi	1. L'utente clicca sulla sezione "Community Inspector" 2. L'utente clicca su "Insert Request" 3. L'utente inserisce in input la repository 4. L'utente inserisce in input la data fine 5. Clicca su "Aggiungi richiesta" 6. L'utente entra nell'area "View your requests"
Oracolo	Il sistema mostra il messaggio di errore: *"Invalid Repository: No milestones found (min. 1)!"*, nell'apposita sezione.

2.1.11 Validazione Repository (TEST_SYSTE11 - Repository con geodispersione non valida)

Tabella 2.11: Validazione Repository (TEST_SYSTE11)

Test Case Id	TEST_SYSTE11
Obiettivo	Verificare che, in caso di richiesta accettata, nel caso in cui la repository non abbia informazioni riguardo la geodispersione sufficienti, venga mostrato l'errore corretto nell'apposita sezione.
Test Frame	Funzionalità: getRequest • URL = Valido: con autore e repository [RA2] • URL = Valido: con https://github.com/ [RH2] • URL $\neq \epsilon$ [RE2] • Data Fine = Valida [DF3] • Commit Validi (Repository) (> 100) [CO2] • Milestone validi (Repository) ($>$ 0) [MI2] • Membri validi (Repository) ($\geq$ 2) [ME2] • Geodispersione community non valida (Repository) [GE1] • Esecuzione conclusa con errore [EC1] • Non ci sono altri problemi legati alla repository [OT2]

Ambiente di test	• Browser: Firefox • Sistema: localhost • Risoluzione: 1290x828 • Versione Selenium: v4+
Pre-Condizione	L'utente ha installato GUIDO e si trova sulla pagina iniziale.
Input	• Repository: `https://github.com/SimpleMobileTools/Simple-Calculator` • Data fine: `01/03/2023`
Flusso di eventi	1. L'utente clicca sulla sezione "Community Inspector" 2. L'utente clicca su "Insert Request" 3. L'utente inserisce in input la repository 4. L'utente inserisce in input la data fine 5. Clicca su "Aggiungi richiesta" 6. L'utente entra nell'area "View your requests"
Oracolo	Il sistema mostra il messaggio di errore: *"Invalid Repository: Insufficient geographical data!"*, nell'apposita sezione.

2.1.12 Validazione Repository (TEST_SYSTE12 - Repository generic error)

Tabella 2.12: Validazione Repository (TEST_SYSTE12)

Test Case Id	TEST_SYSTE12
Obiettivo	Verificare che, in caso di richiesta accettata, qualora la repository non esista o l'utente non abbia i permessi di accesso, venga mostrato il messaggio di errore corretto nell'apposita sezione.
Test Frame	Funzionalità: getRequest • URL = Valido: con autore e repository [RA2] • URL = Valido: con https://github.com/ [RH2] • URL $\neq \epsilon$ [RE2] • Data Fine = Valida [DF3] • Commit Validi (Repository) (> 100) [CO2] • Milestone validi (Repository) ($>$ 0) [MI2] • Membri validi (Repository) ($\geq$ 2) [ME2] • Geodispersione community valida (Repository) [GE2] • Esecuzione conclusa con errore [EC1] • Non ci sono altri problemi legati alla repository [OT1]

Ambiente di test	• Browser: Firefox • Sistema: localhost • Risoluzione: 1290x828 • Versione Selenium: v4+
Pre-Condizione	L'utente ha installato GUIDO e si trova sulla pagina iniziale.
Input	• Repository: `https://github.com/netty/netty` • Data fine: `01/06/2025`
Flusso di eventi	1. L'utente clicca sulla sezione "Community Inspector" 2. L'utente clicca su "Insert Request" 3. L'utente inserisce in input la repository 4. L'utente inserisce in input la data fine 5. Clicca su "Aggiungi richiesta" 6. L'utente entra nell'area "View your requests"
Oracolo	Il sistema mostra il messaggio di errore: *"An Error Occurred during the analysis... Please try again later!"*, nell'apposita sezione.

2.1.13 Validazione Repository (TEST_SYSTE13 - Richieste in stato pending)

Tabella 2.13: Richieste pending (TEST_SYSTE13)

Test Case Id	TEST_SYSTE13
Obiettivo	Verificare che, in caso di richiesta accettata, nella sezione View your requests appaia la richiesta in stato di elaborazione.
Test Frame	Funzionalità: getRequest • URL = Valido: con autore e repository [RA2] • URL = Valido: con https://github.com/ [RH2] • URL $\neq \epsilon$ [RE2] • Data Fine = Valida [DF3] • Commit Validi (Repository) (> 100) [CO2] • Milestone validi (Repository) ($>$ 0) [MI2] • Membri validi (Repository) ($\geq$ 2) [ME2] • Geodispersione community valida (Repository) [GE2] • Esecuzione in stato pending [EC2] • Non ci sono altri problemi legati alla repository [OT2]

Ambiente di test	• Browser: Firefox • Sistema: localhost • Risoluzione: 1290x828 • Versione Selenium: v4+
Pre-Condizione	L'utente ha installato GUIDO e si trova sulla pagina iniziale.
Input	• Repository: `https://github.com/rubygems/bundler` • Data fine: `01/01/2019`
Flusso di eventi	1. L'utente clicca sulla sezione "Community Inspector" 2. L'utente clicca su "Insert Request" 3. L'utente inserisce in input la repository 4. L'utente inserisce in input la data fine 5. Clicca su "Aggiungi richiesta" 6. L'utente entra nell'area "View your requests"
Oracolo	L'utente visualizza la richiesta fatta in stato PENDING.

2.1.14 Validazione Repository (TEST_SYSTE14 - Visualizzazione grafo community)

Tabella 2.14: Visualizzazione grafo (TEST_SYSTE14)

Test Case Id	TEST_SYSTE14
Obiettivo	Verificare che, in caso di richiesta accettata, nella sezione di dettaglio della richiesta (una volta conclusa l'elaborazione), venga visualizzato il grafo rappresentante la community costruito dal Tool TOAD.
Test Frame	Funzionalità: getRequest • URL = Valido: con autore e repository [RA2] • URL = Valido: con https://github.com/ [RH2] • URL $\neq \epsilon$ [RE2] • Data Fine = Valida [DF3] • Commit Validi (Repository) (> 100) [CO2] • Milestone validi (Repository) ($>$ 0) [MI2] • Membri validi (Repository) ($\geq$ 2) [ME2] • Geodispersione community valida (Repository) [GE2] • Esecuzione conclusa con successo [EC3] • Non ci sono altri problemi legati alla repository [OT2]

Ambiente di test	• Browser: Firefox • Sistema: localhost • Risoluzione: 1290x828 • Versione Selenium: v4+
Pre-Condizione	L'utente ha installato GUIDO e si trova sulla pagina iniziale.
Input	• Repository: `https://github.com/rubygems/bundler` • Data fine: `01/01/2019`
Flusso di eventi	1. L'utente clicca sulla sezione "Community Inspector" 2. L'utente clicca su "Insert Request" 3. L'utente inserisce in input la repository 4. L'utente inserisce in input la data fine 5. Clicca su "Aggiungi richiesta" 6. L'utente entra nell'area "View your requests" 7. L'utente clicca sulla richiesta in stato "SUCCESS"

Oracolo	L'utente visualizza il grafo della community riguardante la richiesta fatta ed elaborata dal Tool TOAD.

2.1.15 Validazione Repository (TEST_SYSTE15 - Visualizzazione tabella Community pattern)

Tabella 2.15: Visualizzazione Smells (TEST_SYSTE15)

Test Case Id	TEST_SYSTE15
Obiettivo	Verificare che, in caso di richiesta accettata, nella sezione di dettaglio della richiesta (una volta conclusa l'elaborazione), venga visualizzata la tabella degli smells trovati nella repository.
Test Frame	Funzionalità: getRequest • URL = Valido: con autore e repository [RA2] • URL = Valido: con https://github.com/ [RH2] • URL $\neq \epsilon$ [RE2] • Data Fine = Valida [DF3] • Commit Validi (Repository) (> 100) [CO2] • Milestone validi (Repository) ($>$ 0) [MI2] • Membri validi (Repository) ($\geq$ 2) [ME2] • Geodispersione community valida (Repository) [GE2] • Esecuzione conclusa con successo [EC3] • Non ci sono altri problemi legati alla repository [OT2]

Ambiente di test	• Browser: Firefox • Sistema: localhost • Risoluzione: 1290x828 • Versione Selenium: v4+
Pre-Condizione	L'utente ha installato GUIDO e si trova sulla pagina iniziale.
Input	• Repository: `https://github.com/rubygems/bundler` • Data fine: `01/01/2019`
Flusso di eventi	1. L'utente clicca sulla sezione "Community Inspector" 2. L'utente clicca su "Insert Request" 3. L'utente inserisce in input la repository 4. L'utente inserisce in input la data fine 5. Clicca su "Aggiungi richiesta" 6. L'utente entra nell'area "View your requests" 7. L'utente clicca sulla richiesta in stato "SUCCESS" 8. L'utente clicca sul pulsante "Smells"

Oracolo	L'utente visualizza la tabella che riporta i community pattern della repository trovati da TOAD.

2.1.16 Validazione Repository (TEST_SYSTE16 - Visualizzazione tabella Metrics)

Tabella 2.16: Visualizzazione Metrics (TEST_SYSTE16)

Test Case Id	TEST_SYSTE16
Obiettivo	Verificare che, in caso di richiesta accettata, nella sezione di dettaglio della richiesta (una volta conclusa l'elaborazione), venga visualizzata la tabella delle metriche calcolate sulla community che ha lavorato nella repository.
Test Frame	Funzionalità: getRequest • URL = Valido: con autore e repository [RA2] • URL = Valido: con https://github.com/ [RH2] • URL $\neq \epsilon$ [RE2] • Data Fine = Valida [DF3] • Commit Validi (Repository) ($>$ 100) [CO2] • Milestone validi (Repository) ($>$ 0) [MI2] • Membri validi (Repository) ($\geq$ 2) [ME2] • Geodispersione community valida (Repository) [GE2] • Esecuzione conclusa con successo [EC3] • Non ci sono altri problemi legati alla repository [OT2]

Ambiente di test	• Browser: Firefox • Sistema: localhost • Risoluzione: 1290x828 • Versione Selenium: v4+
Pre-Condizione	L'utente ha installato GUIDO e si trova sulla pagina iniziale.
Input	• Repository: `https://github.com/rubygems/bundler` • Data fine: `01/01/2019`
Flusso di eventi	1. L'utente clicca sulla sezione "Community Inspector" 2. L'utente clicca su "Insert Request" 3. L'utente inserisce in input la repository 4. L'utente inserisce in input la data fine 5. Clicca su "Aggiungi richiesta" 6. L'utente entra nell'area "View your requests" 7. L'utente clicca sulla richiesta in stato "SUCCESS" 8. L'utente clicca sul pulsante "Metrics"

Oracolo	L'utente visualizza la tabella che riporta le metriche della community che ha lavorato alla repository trovate da TOAD.

2.1.17 Validazione Repository (TEST_SYSTE17 - Visualizzazione tabella Members)

Tabella 2.17: Visualizzazione Members (TEST_SYSTE17)

Test Case Id	TEST_SYSTE17
Obiettivo	Verificare che, in caso di richiesta accettata, nella sezione di dettaglio della richiesta (una volta conclusa l'elaborazione), venga visualizzata la tabella dei membri della community.
Test Frame	Funzionalità: getRequest • URL = Valido: con autore e repository [RA2] • URL = Valido: con https://github.com/ [RH2] • URL $\neq \epsilon$ [RE2] • Data Fine = Valida [DF3] • Commit Validi (Repository) ($>$ 100) [CO2] • Milestone validi (Repository) ($>$ 0) [MI2] • Membri validi (Repository) ($\geq$ 2) [ME2] • Geodispersione community valida (Repository) [GE2] • Esecuzione conclusa con successo [EC3] • Non ci sono altri problemi legati alla repository [OT2]

Ambiente di test	• Browser: Firefox • Sistema: localhost • Risoluzione: 1290x828 • Versione Selenium: v4+
Pre-Condizione	L'utente ha installato GUIDO e si trova sulla pagina iniziale.
Input	• Repository: `https://github.com/rubygems/bundler` • Data fine: `01/01/2019`
Flusso di eventi	1. L'utente clicca sulla sezione "Community Inspector" 2. L'utente clicca su "Insert Request" 3. L'utente inserisce in input la repository 4. L'utente inserisce in input la data fine 5. Clicca su "Aggiungi richiesta" 6. L'utente entra nell'area "View your requests" 7. L'utente clicca sulla richiesta in stato "SUCCESS" 8. L'utente clicca sul pulsante "Members"

Oracolo	L'utente visualizza la tabella che riporta i membri della community.

Bibliografia