Analisi dei Requisiti

Versione: 0.3.0 04/12/2024

Redattori	Malik Giafar Mohamed
Verifica	Ion Cainareanu Maria Fuensanta Trigueros Hernandez
Approvazione	
Uso	Esterno

nextsoftpadova@gmail.com

Registro dei cambiamenti

Versione	Data	Autore	Descrizione	Verifica
0.3.0	06/12/2024	Ion Cainareanu	Stesura degli Use Case	
0.2.0	30/12/2024	Ion Cainareanu	Stesura dell'Introduzione e Descrizione	Stefano Baso, Malik Giafar Mohamed
0.1.1	04/12/2024	Luca Parise	Aggiunta indice e creazione struttura tabella per use case	Malik Giafar Mohamed
0.1.0	23/11/2024	Malik Giafar Mohamed	Creazione Documento	Ion Cainareanu, Maria Fuensanta Trigueros Hernandez

Indice

1 Introduzione

1.1 Scopo del documento

Lo scopo del presente documento è fornire una descrizione completa e dettagliata degli obiettivi, delle funzionalità e delle caratteristiche tecniche del progetto **Requirement Tracker - VS Code Plug-in**, con particolare attenzione all'utilizzo dell'UML per la modellazione dei casi d'uso. Il documento funge da riferimento per tutti gli stakeholder coinvolti, descrivendo il contesto operativo, i requisiti funzionali e non funzionali, nonché le linee guida tecnologiche necessarie per lo sviluppo del plug-in. I casi d'uso saranno descritti utilizzando una struttura standardizzata, che includerà il nominativo del caso, gli attori principali, le precondizioni, le postcondizioni, lo scenario principale e gli eventuali scenari alternativi o sottocasi. Questa struttura garantisce chiarezza e coerenza, facilitando la comprensione e la tracciabilità delle funzionalità principali del sistema. Il documento intende inoltre fornire una visione condivisa del progetto, ponendo le basi per una pianificazione e un'implementazione efficaci.

1.2 Scopo del prodotto

Lo scopo di **Requirement Tracker - VS Code Plug-in** è affrontare il problema della complessità nella gestione e nel tracciamento dei requisiti nei progetti software di grandi dimensioni. Nei codebase estesi, la verifica manuale della copertura e dell'implementazione dei requisiti è un processo lungo e soggetto a errori, spesso complicato dalla qualità insufficiente con cui i requisiti stessi vengono definiti. Questo può portare a malintesi e problemi durante l'implementazione, compromettendo l'allineamento tra specifiche e funzionalità sviluppate. Il plug-in mira a risolvere queste difficoltà automatizzando il tracciamento dei requisiti nel codice sorgente, migliorando la qualità della loro definizione e semplificando l'identificazione delle aree di mancata o errata implementazione. In particolare, offre strumenti per integrare requisiti tecnici derivati da manuali e datasheet di componenti hardware, fornendo analisi automatizzate e suggerimenti per rendere i requisiti più chiari, specifici e strutturati. Grazie a questo, sviluppatori potranno garantire una gestione più efficace dei requisiti, riducendo errori e aumentando la coerenza tra specifiche e implementazione.

1.3 Glossario

Per evitare ambiguità relative alle terminologie utilizzate è stato creato un documento denominato **Glossario**. Questo documento comprende tutti i termini tecnici scelti dai membri del gruppo e utilizzati nei vari documenti con le relative definizioni. Tutti i termini inclusi in questo glossario vengono segnalati all'interno del documento con l'apice [G] accanto alla parola.

2 Descrizione

2.1 Obiettivi del prodotto

L'obiettivo del progetto è realizzare un plug-in per Visual Studio Code che consenta di tracciare e verificare l'implementazione dei requisiti di progetto, basandosi su analisi automatizzate del codice sorgente e sui requisiti tecnici espressi in documenti di riferimento, mediante l'utilizzo di tecnologie avanzate come modelli LLM[G] di AI[G]. Il plug-in sarà supportato da API REST[G] che si interfacciano con Ollama[G], fornendo un'infrastruttura flessibile e scalabile per l'integrazione di modelli di AI e garantendo un'elaborazione efficiente e sicura delle analisi richieste.

2.2 Funzionalità del prodotto

Il plug-in sarà utilizzato dal programmatore per analizzare i requisiti implementati nel codice sorgente. Sia i requisiti che il codice saranno analizzati da vari modelli LLM reperibili attraverso la piattaforma di Ollama, grazie alle API REST che interagiscono con Ollama.
Le funzionalità implementate nell'applicazione includono:

- Importazione del file dei requisiti in formato CSV[G]
- Richiesta di analisi dei requisiti tramite un modello LLM;

- Valutazione qualitativa dei requisiti;
- Visualizzazione grafica dei risultati dell'analisi;
- Filtraggio dei risultati dell'analisi;
- Possibilità di ripetere l'analisi di un requisito specifico;
- Esportazione dei risultati dell'analisi in formato CSV;
- Verifica dell'implementazione dei requisiti nel codice sorgente;
- Analisi semantica dei requisiti e del codice sorgente;
- Suggerimenti per migliorare la qualità dei requisiti e del codice.

2.3 Utenti e caratteristiche

In seguito a un incontro con il proponente, è stato discusso come il plug-in possa essere utilizzato principalmente da un utente che ricopre il ruolo di programmatore. Di conseguenza, si è deciso di focalizzare le funzionalità del plug-in per rispondere alle esigenze di questa categoria di utenti. È stato inoltre specificato che non devono essere fatte assunzioni sulle competenze tecniche dell'utente riguardo all'uso di Visual Studio Code. Pertanto, il plug-in deve essere progettato per essere il più intuitivo possibile, con un processo di installazione semplice e accessibile.

3 Use Case

3.1 Obiettivi

Questa sezione si propone di identificare e descrivere i casi d'uso derivati dall'analisi del capitolato d'appalto selezionato dal gruppo. In particolare, vengono definiti gli attori principali e le funzionalità ad essi associate.

3.2 Attori

L'applicazione è progettata con un unico attore, il **Programmatore**, che ha accesso a tutte le funzionalità del sistema.

3.3 UC_1 - Importazione del file dei requisiti in formato CSV

Attori: Programmatore.

Precondizioni:

- L'utente ha aperto un progetto in Visual Studio Code.
- Il file dei requisiti deve essere in formato CSV valido.
- Il plug-in deve essere installato e attivo in VS Code.

Postcondizioni:

- I requisiti dal file CSV sono stati importati correttamente e sono visualizzabili nel sistema.

Scenario principale:

1. L'utente seleziona l'opzione "Importa file CSV".
2. Il sistema apre un file explorer.
3. L'utente seleziona il file CSV da importare.
4. Il sistema verifica la validità del file.
5. I requisiti vengono importati e mostrati in una vista strutturata.

Estensioni:

- **UC_1.1:** File CSV non valido: il sistema notifica l'errore all'utente e richiede di selezionare un file corretto.

3.4 UC_1.1 - File CSV non valido

Attori: Programmatore.

Precondizioni:

- L'utente ha selezionato un file CSV non valido o malformato.

Postcondizioni:

- L'utente viene informato che il file non è valido.

Scenario principale:

1. L'utente seleziona un file CSV da importare.
2. Il sistema verifica il file e rileva che è malformato o non valido.
3. Il sistema mostra un messaggio di errore esplicativo e richiede di selezionare un file valido.

3.5 UC_2 - Analisi automatizzata dei requisiti

Attori: Programmatore.

Precondizioni:

- Il file dei requisiti è stato importato con successo.
- La connessione con le API REST di Ollama è attiva.

Postcondizioni:

- Il sistema fornisce una valutazione dei requisiti.

Scenario principale:

1. L'utente seleziona "Analizza requisiti".
2. Il sistema invia i requisiti a un modello LLM tramite API REST.
3. Il modello restituisce i risultati dell'analisi.
4. Il sistema visualizza i risultati nella UI con suggerimenti.

Estensioni:

- **UC_2.1:** Errore di connessione alle API: il sistema informa l'utente e permette di riprovare.
- **UC_2.2:** Il sistema sta sottoperformando per connessione lenta o modello troppo grande: il sistema informa l'utente con messaggi esplicativi.
- **UC_2.3:** Verifica dell'implementazione dei requisiti.

3.6 UC_2.1 - Errore di connessione alle API

Attori: Programmatore.

Precondizioni:

- La connessione internet è instabile o assente.

Postcondizioni:

- Il sistema informa l'utente dell'errore e fornisce indicazioni per ripristinare la connessione.

Scenario principale:

1. L'utente avvia l'analisi dei requisiti.
2. Il sistema tenta di connettersi alle API REST.
3. La connessione fallisce.
4. Il sistema mostra un messaggio di errore e suggerisce di verificare la connessione.

3.7 UC_2.2 - Sistema in sottoperformance

Attori: Programmatore.

Precondizioni:

- La connessione è lenta o il modello richiede risorse computazionali elevate.

Postcondizioni:
- Il sistema notifica all'utente il problema.

Scenario principale:
1. L'utente avvia l'analisi dei requisiti.
2. Il sistema rileva che il tempo di risposta delle API è eccessivo o il modello richiede troppo tempo per l'elaborazione.
3. Il sistema informa l'utente del ritardo.

3.8 UC_2.3 - Verifica dell'implementazione dei requisiti

Attori: Programmatore.

Precondizioni:
- I requisiti sono stati importati.
- È disponibile il codice sorgente da analizzare.

Postcondizioni:
- Il sistema verifica e visualizza il grado di implementazione dei requisiti nel codice.

Scenario principale:
1. L'utente seleziona "Analizza requisiti".
2. Il sistema analizza il codice sorgente utilizzando modelli semantici.
3. I risultati dell'analisi vengono presentati con indicazioni sulle discrepanze.

Estensioni:
- **UC_2.3.1:** Codice sorgente non disponibile: il sistema notifica l'errore e richiede di configurare il progetto.

3.9 UC_2.3.1 - Codice sorgente non disponibile

Attori: Programmatore.

Precondizioni:
- Il progetto non contiene file sorgente o non è configurato correttamente.

Postcondizioni:
- Il sistema informa l'utente e guida nella configurazione del progetto.

Scenario principale:
1. L'utente avvia la verifica dell'implementazione dei requisiti.
2. Il sistema verifica la presenza del codice sorgente.
3. Il sistema rileva che non è configurato correttamente o non è presente.
4. Il sistema mostra un messaggio di errore.

3.10 UC_3 - Esportazione dei risultati in formato CSV

Attori: Programmatore.

Precondizioni:
- I risultati di un'analisi devono essere disponibili.

Postcondizioni:
- I risultati vengono esportati correttamente in un file CSV.

Scenario principale:
1. L'utente seleziona "Esporta risultati".

2. Il sistema apre un file explorer per scegliere il percorso di salvataggio.
3. L'utente conferma la posizione e il nome del file.
4. Il sistema salva i risultati nel formato CSV.

Estensioni:
- **UC_3.1:** Mancato salvataggio: il sistema notifica l'errore e permette di riprovare.

3.11 UC_3.1 - Mancato salvataggio

Attori: Programmatore.

Precondizioni:
- L'utente tenta di esportare i risultati ma il salvataggio fallisce.

Postcondizioni:
- Il sistema informa l'utente dell'errore e consente di riprovare l'operazione.

Scenario principale:
1. L'utente seleziona "Esporta risultati".
2. Il sistema tenta di salvare il file.
3. Si verifica un errore durante il salvataggio (es. mancanza di permessi o spazio insufficiente).
4. Il sistema mostra un messaggio di errore.
5. L'utente ha la possibilità di riprovare o cambiare il percorso di salvataggio.

3.12 UC_4 - Visualizzazione grafica dei risultati dell'analisi

Attori: Programmatore.

Precondizioni:
- I risultati dell'analisi dei requisiti sono stati generati.

Postcondizioni:
- I risultati vengono rappresentati graficamente per migliorare la comprensione.

Scenario principale:
1. L'utente seleziona la voce dei requisiti.
2. Il sistema espande un menu ad albero con i risultati dell'analisi.
3. L'utente può selezionare ed espandere i risultati sui requisiti di suo interesse.

Estensioni:
- **UC_4.1:** Nessun risultato disponibile: il sistema informa l'utente che non ci sono requisiti analizzabili.

3.13 UC_4.1 - Nessun risultato disponibile

Attori: Programmatore.

Precondizioni:
- Non ci sono risultati generati dall'analisi.

Postcondizioni:
- Il sistema notifica l'utente dell'assenza di risultati e fornisce indicazioni per avviare una nuova analisi.

Scenario principale:
1. L'utente accede alla visualizzazione dei risultati.
2. Il sistema rileva che non ci sono dati da mostrare.
3. Il sistema visualizza un messaggio informativo e suggerisce di controllare i requisiti o avviare una nuova analisi.

3.14 UC_5 - Filtraggio dei risultati dell'analisi

Attori: Programmatore.

Precondizioni:

- I risultati dell'analisi sono stati generati e visualizzati.

Postcondizioni:

- I risultati vengono filtrati in base ai criteri scelti dall'utente.

Scenario principale:

1. L'utente utilizza la barra di ricerca.
2. Il sistema filtra la lista dei requisiti in base al testo del requisito, nome del requisito o percorso del codice associato al requisito.
3. Il sistema applica la ricerca e aggiorna la visualizzazione.

Estensioni:

- **UC_5.1:** Nessun risultato corrisponde ai filtri: il sistema informa l'utente e consente di modificare i criteri.

3.15 UC_5.1 - Nessun risultato corrisponde ai filtri

Attori: Programmatore.

Precondizioni:

- Non ci sono risultati che soddisfano i criteri di filtro applicati dall'utente.

Postcondizioni:

- Il sistema informa l'utente che nessun risultato corrisponde ai criteri e consente di modificare il filtro.

Scenario principale:

1. L'utente applica un filtro utilizzando la barra di ricerca.
2. Il sistema verifica i risultati rispetto ai criteri inseriti.
3. Nessun risultato corrisponde ai criteri.
4. Il sistema mostra un messaggio informativo indicando che non ci sono risultati disponibili.
5. L'utente ha la possibilità di modificare i criteri di filtro o di rimuoverli completamente.

3.16 UC_6 - Ripetizione dell'analisi di un requisito specifico

Attori: Programmatore.

Precondizioni:

- Il requisito da analizzare è stato selezionato.

Postcondizioni:

- Viene fornito un nuovo risultato per il requisito selezionato.

Scenario principale:

1. L'utente seleziona un requisito dalla lista.
2. L'utente clicca su "Ripeti analisi".
3. Il sistema invia il requisito al modello LLM per una nuova analisi.
4. I risultati aggiornati vengono visualizzati per il requisito selezionato.

Estensioni:

- **UC6_1:** Errore nella comunicazione con il modello: il sistema notifica l'utente e consente di riprovare.

3.17 UC_6.1 - Errore nella comunicazione con il modello

Attori: Programmatore.

Precondizioni:
- Il sistema non riesce a stabilire una connessione con il modello LLM per l'analisi del requisito.

Postcondizioni:
- Il sistema informa l'utente dell'errore e permette di riprovare l'operazione.

Scenario principale:
1. L'utente avvia la ripetizione dell'analisi di un requisito.
2. Il sistema tenta di inviare il requisito al modello LLM.
3. La comunicazione con il modello fallisce.
4. Il sistema mostra un messaggio di errore dettagliato con opzioni per riprovare o controllare la configurazione della connessione.

3.18 UC_7 - Importazione del codice sorgente per l'analisi

Attori: Programmatore.

Precondizioni:
- Il progetto deve essere configurato in Visual Studio Code.

Postcondizioni:
- Il codice sorgente è pronto per essere analizzato dal sistema.

Scenario principale:
1. L'utente seleziona "Analisi requisiti".
2. Il sistema verifica i file del progetto aperto in Visual Studio Code.
3. Il sistema carica i file necessari per l'analisi dei requisiti e del codice associato.
4. Il sistema fa partire l'analisi dei requisiti inviando la richiesta all'API.

Estensioni:
- **UC7_1:** Progetto non configurato correttamente: il sistema fornisce un errore e suggerisce di configurare il progetto.

3.19 UC_7.1 - Progetto non configurato correttamente

Attori: Programmatore.

Precondizioni:
- Il progetto aperto in Visual Studio Code non è configurato in modo corretto o mancano file necessari (ad esempio l'uso del linguaggio di programmazione non supportato).

Postcondizioni:
- Il sistema informa l'utente e fornisce una guida per configurare correttamente il progetto.

Scenario principale:
1. L'utente avvia l'importazione del codice sorgente per l'analisi.
2. Il sistema verifica la configurazione del progetto.
3. Il sistema rileva che la configurazione è incompleta o errata.
4. Il sistema visualizza un messaggio di errore con una guida dettagliata per configurare correttamente il progetto.
5. L'utente può correggere la configurazione e riprovare l'operazione.

3.20 UC_8 - Configurazione dei path da ignorare per la ricerca dei requisiti

Attori: Programmatore.

Precondizioni:
- Il progetto è stato configurato in Visual Studio Code.

- Il plug-in è attivo e funzionante.

Postcondizioni:
- I path specificati nel file .ignore vengono esclusi dalla ricerca dei requisiti nel codice sorgente.

Scenario principale:
1. L'utente crea o modifica un file .ignore nel progetto.
2. L'utente inserisce nel file .ignore i path o pattern relativi ai file o directory da escludere.
3. Il sistema rileva automaticamente le modifiche al file .ignore.
4. Durante la ricerca dei requisiti nel codice sorgente, il sistema esclude i path specificati nel file .ignore.
5. L'utente avvia l'analisi e i path ignorati non vengono considerati.

Estensioni:
- **UC_8.1:** Path specificato non valido: il sistema notifica l'utente e ignora l'entry errata mantenendo valide le altre.

3.21 UC_8.1 - Path specificato non valido
Attori: Programmatore.

Precondizioni:
- L'utente inserisce un path o pattern non valido nel file .ignore.

Postcondizioni:
- Il sistema ignora il path non valido e prosegue con le configurazioni valide.

Scenario principale:
1. L'utente modifica il file .ignore e inserisce un path o pattern non valido.
2. Il sistema rileva l'entry non valida durante la verifica del file.
3. Il sistema notifica l'utente dell'errore e fornisce dettagli sul path non valido.
4. Il sistema ignora l'entry non valida e considera solo i path configurati correttamente.

4 Elenco delle immagini

5 Elenco delle tabelle