Norme di Progetto

Versione: 1.1.3 23/11/2024

Redattori	Malik Giafar Mohamed
Verifica	
Approvazione	
Uso	Interno

nextsoftpadova@gmail.com

Registro dei cambiamenti

Versione	Data	Autore	Descrizione	Verifica
1.1.3	11/05/2025	Malik Giafar Mohamed	Modifica sezione descrizione dei test	
1.1.2	09/05/2025	Malik Giafar Mohamed	Aggiornamento della procedura di modifica delle pull requests	
1.1.1	08/05/2025	Malik Giafar Mohamed	Aggiornamento tecnologie utilizzate e metodologie di testing	
1.1.0	30/04/2025	Malik Giafar Mohamed	Introduzione modifiche in tutto il documento derivanti dalla correzione dell'RTB	
1.0.1	07/03/2025	Malik Giafar Mohamed	Inserimento data di ultimo accesso nelle fonti	

Indice

1 Introduzione

1.1 Scopo del documento

Il presente documento ha lo scopo di definire in modo dettagliato le *best practice*[G] e il *way of working*[G] del nostro gruppo per il progetto al fine di garantire qualità e coerenza nel lavoro svolto. Il documento sarà quindi soggetto a modifiche e integrazioni durante il corso del progetto, in particolare durante le fasi di analisi e progettazione, e quindi non può essere considerato come definitivo.

1.2 Scopo del prodotto

Il prodotto, un *plug-in*[G] per *Visual Studio Code*[G] chiamato *"Requirement Tracker"*[G], è progettato per automatizzare il *tracciamento dei requisiti*[G] nei progetti software complessi, con un focus particolare sull'ambito embedded. L'obiettivo principale è migliorare la qualità e la chiarezza dei requisiti, fornendo suggerimenti basati sull'analisi di un'*intelligenza artificiale*[G], riducendo al contempo i tempi e gli errori legati alla verifica manuale dell'implementazione nel codice sorgente. Il plug-in adotta un'architettura modulare che consente un'estensibilità semplice, rendendolo facilmente adattabile a nuove funzionalità o esigenze future.

1.3 Glossario

I termini ambigui che necessitano di una spiegazione sono contrassegnati da una [G] come apice alla loro prima occorrenza nei documenti. Tutti i termini da glossario sono riportati in ordine alfabetico nell'omonimo documento.

1.4 Riferimenti

Riferimenti Normativi

- Presentazione del capitolato Requirement Tracker - Plug-in VS Code
  - https://www.math.unipd.it/~tullio/IS-1/2024/Progetto/C8.pdf
- Standard ISO/IEC 12207:1995
  - https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf
- *Piano di Qualifica*[G] v1.0.0

Riferimenti Informativi

- Materiale didattico del corso di Ingegneria del Software
  - https://www.math.unipd.it/~tullio/IS-1/2024/Dispense/
- Documentazione *GitHub*[G]:
  - https://help.github.com/en/github ultumo acceso: 05/04/2025 13:06
- Documentazione *git*[G]:

  - https://git-scm.com/docs ultimo acceso: 05/04/2025 13:06
- Documentazione Typst:
  - https://typst.app/docs/ ultimo acceso: 05/04/2025 13:06
- Documentazione Visual Studio Code:
  - https://code.visualstudio.com/docs ultimo acceso: 05/04/2025 13:06
  - https://code.visualstudio.com/api ultimo acceso: 05/04/2025 13:06
- Documentazione API *Ollama*[G]:
  - https://github.com/ollama/ollama/blob/main/docs/api.md ultimo acceso: 07/05/2025 15:50
- Documentazione *NestJS*[G]:
  - https://docs.nestjs.com/ ultimo acceso: 07/05/2025 15:50
- Documentazione *Docker*[G]:
  - https://docs.docker.com/get-started/ ultimo acceso: 07/05/2025 15:50
- Documentazione *StarUML*[G]:
  - https://staruml.io/docs/ ultimo acceso: 07/05/2025 15:50
- Documentazione *npm*[G]:
  - https://docs.npmjs.com/ ultimo acceso: 07/05/2025 15:50
- Documentazione *Typescript*[G]:
  - https://www.typescriptlang.org/docs/ ultimo acceso: 07/05/2025 15:50
- Documentazione *Mocha*[G]:
  - https://mochajs.org/ ultimo acceso: 07/05/2025 15:50
- Documentazione *Jest*[G]:
  - https://jestjs.io/ ultimo acceso: 07/05/2025 15:50
- Documentazione *SuperTest*[G]:
  - https://www.npmjs.com/package/supertest/ ultimo acceso: 07/05/2025 15:50

2 Processi Primari

2.1 Fornitura

2.1.1 descrizione

Secondo lo standard ISO/IEC 12207:1995, il processo di fornitura comprende le attività e i compiti svolti dal fornitore.

Il processo può essere avviato dalla decisione di preparare una proposta che risponda a una richiesta formale da parte dell'acquirente, nel nostro caso quindi l'avvio di tale processo è attribuito all'aggiudicazione dell'appalto dell'azienda proponente da parte del gruppo.

Il processo prosegue con la determinazione delle procedure e delle risorse necessarie per gestire e far avanzare il progetto, inclusa la pianificazione e l'esecuzione dei piani, fino alla consegna del sistema, del prodotto software o del servizio software all'acquirente.

2.1.2 Comunicazione con il proponente

Il proponente[G] metterà a disposizione alcuni canali di comunicazione tramite i quali chiarire eventuali dubbi e programmare nuovi incontri. La cadenza di questi incontri non sarà regolare, ma verrà fissata in base alle necessità del gruppo o dell'azienda. Gli incontri possono essere richieste per discutere i seguenti argomenti:

- Chiarimenti relativi a requisiti o vincoli del *capitolato*[G]
- Dubbi sulla gestione delle tecnologie utilizzate
- Richiesta di feedback su quanto prodotto

Per ogni colloquio con il proponente verrà steso un resoconto nel *Verbale Esterno*[G] che includerà la data in cui tale incontro è avvenuto. Per ogni *baseline*[G], i verbali redatti potranno essere visualizzati in un'apposita cartella nella repository[G] documentale, al percorso: `"NomeBaseline"/Documentazione Esterna/Verbali`.

2.1.3 Documentazione fornita

La documentazione che verrà consegnata al proponente sarà la seguente:

- **Analisi dei Requisiti v2.0.0**: descrive dettagliatamente i requisiti funzionali e non funzionali del progetto. Include una lista dettagliata di requisiti e tutti i *casi d'uso*[G] necessari per descrivere al meglio le esigenze del proponente sul prodotto software. Questo documento è fondamentale per comprendere a fondo le necessità del proponente e per fornire una base solida per la progettazione e lo sviluppo del prodotto.
- **Piano di Qualifica v2.0.0**: definisce le strategie e le metodologie di *verifica*[G] e *validazione*[G] adottate dal gruppo, includendo *metriche*[G] di qualità, *test*[G] pianificati e procedure per garantire che il prodotto software soddisfi tutti i requisiti prefissati. Inoltre, il piano di qualifica fornirà un resoconto sull'avanzamento delle attività descritte, assicurando che i prodotti e i processi siano conformi agli standard di qualità e che tutte le funzionalità richieste siano implementate correttamente nel prodotto software.
- **Piano di Progetto v2.0.0**: descrive la *pianificazione*[G] temporale e le risorse necessarie per completare il progetto. Esso include gli orari sia preventivati che effettivi di tutta la

durata del progetto, e riporterà anche una lista dei rischi possibili o già verificati, insieme alla loro *mitigazione*[G]. Tutto questo sarà riportato in ogni sprint per garantire un monitoraggio costante dell'andamento del progetto.

- ***MVP***[G]: il Minimum Viable Product, ovvero un prodotto software funzionante che implementi tutte le funzionalità di base essenziali, il quale verrà presentato al proponente. Esso dovrà essere realizzato secondo quanto scritto nel documento di specifica tecnica e soprattutto dovrà essere accettato dall'azienda proponente per essere considerato soddisfacente. Il suo rilascio determinerà la fine del progetto.
- **Glossario v2.0.0**: un documento che contiene la definizione dei termini tecnici e delle sigle utilizzate nel progetto. Questo documento è essenziale per garantire che tutti i membri del gruppo e le parti interessate abbiano una comprensione comune dei termini utilizzati nel progetto.
- **Manuale Utente v1.0.0**: in questo documento saranno presenti informazioni su come installare, configurare e utilizzare il plug-in. Questo documento è fondamentale per garantire che gli utenti siano in grado di utilizzare il prodotto in modo efficace e per massimizzare il valore del software.
- **Specifica Tecnica v1.0.0**: un documento che fornisce dettagli tecnici sulla struttura e sul funzionamento del prodotto software. Esso include diagrammi UML dettagliati dell'architettura del sistema in ogni sua componente, con una descrizione testuale correlata. Sono presenti inoltre l'elenco delle tecnologie utilizzate e delle scelte progettuali adottate. Questo documento è essenziale per garantire che il prodotto sia sviluppato in modo coerente e che tutte le parti interessate abbiano una comprensione chiara delle scelte tecniche effettuate.

2.1.4 Strumenti

Il nostro processo di fornitura prevede l'utilizzo dei seguenti strumenti:

- **Google Sheets**[G]: utilizzato per la gestione di fogli di calcolo tra i membri del gruppo. Viene usato principalmente per la creazione di grafici per la registrazione dell'andamento delle metriche di qualità, e di tabelle per la pianificazione temporale del progetto e la gestione dei requisiti.
- **Discord**[G]: utilizzato come piattaforma di comunicazione per le discussioni interne al gruppo. È utile per organizzare riunioni, discutere problemi tecnici e coordinare le attività in modo informale.
- **GitLab**[G]: piattaforma che permette la gestione di repository Git, verrà utilizzato per ospitare il codice sorgente dell'MVP al rilascio del prodotto.

- **GitHub**[G]: un'altra piattaforma che permette la gestione di repository Git, verrà utilizzato per la gestione della repository documentale e del codice sorgente dei prodotti software non ancora rilasciati.
- **Typst**[G]: È un linguaggio di markup simile a Markdown, che consente di creare documenti strutturati e ben formattati in modo semplice ed efficiente. Esso verrà utilizzato principalmente per la stesura della documentazione.
- **Whatsapp**[G]: utilizzato come piattaforma di comunicazione per le discussioni interne al gruppo. È utile per organizzare riunioni, discutere problemi tecnici e coordinare le attività in modo informale.

2.2 Sviluppo

2.2.1 Descrizione

Secondo lo standard ISO/IEC 12207:1995, il processo di sviluppo comprende le attività e i compiti di analisi, progettazione, codifica, integrazione, testing, installazione e accettazione relativi ai prodotti software.

Le principali attività del processo di sviluppo del nostro progetto sono:

- **Analisi dei Requisiti**: definizione e documentazione dei requisiti e dei casi d'uso del sistema, svolta dagli *Analisti*[G].
- **Progettazione**: definizione dell'architettura del sistema e delle scelte tecnologiche, a cura dei *Progettisti*[G].
- **Codifica**: implementazione del prodotto software basata sulla progettazione, realizzata dai *Programmatori*[G].

2.2.2 Analisi dei Requisiti

Durante l'attività di *analisi dei requisiti*[G], svolta dagli *Analisti*[G], vengono definiti e riportati nel relativo documento i requisiti e i casi d'uso del sistema. I requisiti rappresentano le caratteristiche e le funzionalità che il sistema deve possedere per soddisfare le esigenze degli utenti e degli *stakeholder*[G]. I casi d'uso, invece, descrivono le interazioni tra gli utenti e il sistema, specificando come il sistema deve comportarsi in risposta a determinate azioni degli utenti.

2.2.2.1 Casi d'uso

Ogni caso d'uso verrà riportato con una denominazione chiara e univoca che lo identifichi, il formato adottato sarà il seguente:

UC_[Numero caso d'uso](.[Numero sottocaso o scenario alternativo])* - Titolo

Mentre essi saranno strutturati nel seguente modo :

- **Denominazione**: codice identificativo del caso d'uso, stabilito come enunciato sopra
- **Diagramma UML**: diagramma per rappresentare graficamente il caso d'uso (opzionale per i sottocasi d'uso).
- **Attori P principali**: entità esterne al sistema che interagiscono con esso
- **Precondizioni**: descrivono lo stato del sistema prima del verificarsi del caso d'uso
- **Postcondizioni**: descrivono lo stato del sistema dopo che si è verificato il caso d'uso
- **Scenario principale**: elenco puntato che descrive il flusso degli eventi del caso d'uso
- **Estensioni**: elenco puntato di sottocasi d'uso, o di specializzazioni del caso d'uso principale, che descrivono il flusso degli eventi dopo un evento imprevisto che lo ha deviato dal caso principale. Possono non esserci o possono essercene varie.

2.2.2.2 Denominazione dei requisiti

Ogni requisito verrà riportato nel seguente formato:

R[Classificazione][Tipologia][Identificativo]

Dove:

- **R**: indica che si tratta di un requisito
- **Classificazione**: specifica la categoria del requisito
  - **F**: *requisito funzionale*[G]
  - **P**: *requisito prestazionale*[G]
  - **Q**: *requisito qualitativo*[G]
  - **V**: *vincolo*[G]
- **Tipologia**: descrive la natura del requisito
  - **D**: *requisito desiderabile*[G]
  - **O**: *requisito obbligatorio*[G]
  - **F**: *requisito facoltativo*[G]
- **Identificativo**: numero intero progressivo e univoco del requisito

2.2.3 Progettazione

2.2.3.1 Descrizione

L'attività di progettazione è legata al ruolo dei *Progettisti*[G], consiste nel definire l'architettura della soluzione considerando i requisiti derivanti dall'analisi fatta in precedenza. La progettazione si articola in due fasi principali:

- **Progettazione logica**: in questa fase si definisce l'architettura generale del sistema, identificando i componenti principali, le interazioni tra di essi e con chi utilizzerà il prodotto. In questa fase vengono documentati i requisiti nel relativo documento e vengono creati i diagrammi UML per rappresentare graficamente le interazioni con il sistema da parte dell'utente.
- **Progettazione di dettaglio**: in questa fase si approfondiscono i dettagli tecnici di ciascun componente, definendo design patterns, best practices e diagrammi UML per descrivere l'architettura delle classi. Tutte le scelte riportate dovranno poi essere implementate nella programmazione e riportate nel documento di specifica tecnica.

2.2.3.2 Requirements and Technology Baseline

Lo scopo della *Requirements and Technology Baseline*[G] è di fornire una base solida per la progettazione e lo sviluppo del prodotto, dimostrando la fattibilità delle soluzioni proposte e garantendo che le tecnologie scelte siano adeguate per soddisfare i requisiti del proponente.

In questa *baseline*[G] dovranno essere forniti:

- ***Proof of Concept***[G]: un prodotto software temporaneo creato per dimostrare la fattibilità del capitolato. Verrà realizzato senza seguire le linee guida di progettazione e codifica definite in questo documento, ma con l'obiettivo di dimostrare che il prodotto finale sarà realizzabile.
- **Scelte tecnologiche**: verranno raccolti requisiti, casi d'uso e scelte tecnologiche che verranno utilizzate per la realizzazione del prodotto finale. Le scelte tecnologiche comprendono le tecnologie, i linguaggi di programmazione e gli strumenti che verranno utilizzati per lo sviluppo e per il testing del prodotto.
- **Diagrammi dei casi d'uso**: verranno utilizzati per rappresentare graficamente le interazioni tra gli utenti e il sistema. Questi diagrammi sono fondamentali per comprendere le esigenze degli utenti e per garantire che il prodotto soddisfi i requisiti richiesti.

2.2.3.3 Product Baseline

Lo scopo della Product Baseline è di fornire un prodotto software che verrà valutato come MVP. Esso dovrà soddisfare i requisiti obbligatori definiti nel capitolato e nel documento di analisi dei requisiti, dovrà essere implementato secondo quanto scritto nel documento di specifica tecnica e soprattutto dovrà essere accettato dall'azienda proponente per essere considerato soddisfacente.

Inoltre, durante l'avanzamento di questa baseline, verranno progettati:

- **Diagrammi delle classi**: verranno utilizzati per descrivere l'architettura logica e di dettaglio dell'MVP.
- ***Design Patterns***[G]: soluzioni di programmazione che consentono di risolvere problemi ricorrenti in modo rapido ed efficace. I design pattern sono schemi riutilizzabili di progettazione illustrati con diagrammi che ne mostrano la struttura
- ***Test***[G]: ovvero la definizione, l'implementazione e il resoconto dei test eseguiti per verificare che il funzionamento del sistema siano corretti e conformi ai requisiti.

2.2.4 Codifica

2.2.4.1 Descrizione

L'attività di *codifica*[G] è legata al ruolo dei *Programmatori*[G], consiste nell'implementazione del prodotto software tenendo conto della progettazione.

2.2.4.2 Metodi

I metodi di un programma verranno considerati accettabili solamente se brevi. Risulta essere una buona pratica in quanto porta notevoli vantaggi quali:

- **Manutenibilità**: sono più facili da mantenere rispetto a metodi lunghi e complessi in quanto il codice è più leggibile e comprensibile. Si ottiene così un codice robusto e meno suscettibile a errori.
- **Leggibilità**: il codice è reso più accessibile a tutti gli sviluppatori che potrebbero doverlo leggere, comprendere o modificare in futuro.
- **Debugging**: se un metodo è breve e semplice, è più facile identificare eventuali *bug*[G] o problemi nel codice. Questo rende il processo di *debugging*[G] più efficiente.

Per la dichiarazione dei metodi si fa riferimento alla convenzione "*CamelCase*[G]" per la scelta del nome del metodo.

2.2.4.3 Univocità dei nomi

Tutte le variabili, i metodi e le classi dovranno avere un nome che li distingua univocamente, per evitare di creare confusione durante la stesura e l'analisi.

2.2.4.4 Strumenti

Per la codifica del prodotto, i principali strumenti adottati saranno:

- **Visual Studio Code**: in quanto è l'*IDE*[G] più consigliato per realizzare proprie estensioni.
- **Typescript**: linguaggio di programmazione che verrà utilizzato per la realizzazione del plug-in. È un linguaggio di programmazione open source, sviluppato da Microsoft, che estende JavaScript aggiungendo tipizzazione statica e funzionalità orientate agli oggetti.
- **Visual Studio Code Extension API**: utilizzata per lo sviluppo del plug-in, fornisce un set di API che consentono di estendere le funzionalità di Visual Studio Code.
- **NestJS**: framework per lo sviluppo di applicazioni server-side, utilizzato per la realizzazione del backend del prodotto.
- **npm**: gestore di pacchetti per nodejs, utilizzato per installare e gestire le dipendenze del progetto.
- **Visual Studio Code**: IDE utilizzato per la scrittura del codice sorgente e per il debugging del prodotto.
- **StarUML**: per la creazione dei *diagrammi UML*[G]
- **Ollama API**: utilizzata per la comunicazione tra il modello LLM locale di Ollama e il backend del plugin.
- **Docker**: piattaforma per lo sviluppo, la distribuzione e l'esecuzione di applicazioni in container, verrà utilizzato per il deployment del server API del backend.

3 Processi di Supporto

3.1 Documentazione

3.1.1 Descrizione

Secondo lo standard ISO/IEC 12207:1995, il processo di documentazione è un processo per registrare le informazioni prodotte da un processo o attività del ciclo di vita. Il processo comprende un insieme di attività che pianificano, progettano, sviluppano, producono, modificano, distribuiscono e mantengono quei documenti necessari a tutte le parti interessate, come manager, ingegneri e utenti del sistema o del prodotto software.

3.1.2 Ciclo di vita del documento

Il ciclo di vita di un documento è suddiviso nelle seguenti fasi:

- **Creazione**: viene definita una struttura di base per il documento, che include le sezioni principali, poi viene svolta un'analisi sulle informazioni di un processo o di un'attivita

da includere nel documento, una volta identificate le informazioni da includere si procede con la stesura del documento.

- **Stesura**: Una volta raccolte le informazioni da aggiungere o da modificare, si procede al loro inserimento all'interno del documento. Durante questa fase, è importante seguire le norme tipografiche e di stile definite in questo documento per garantire coerenza e chiarezza nel contenuto. La stesura può essere effettuata da uno o più redattori, a seconda della complessità del documento e delle informazioni da includere.
- **Verifica**: una volta completata la stesura, il documento viene sottoposto a verifica da parte di uno o più verificatori. Quest'ultimi controllano la correttezza e la completezza del contenuto, segnalando eventuali errori o incongruenze. Eventualmente è possibile un ritorno alla fase di stesura in caso la verifica abbia esito negativo e sia necessario modificare il contenuto del documento.
- **Approvazione**: una volta verificata una versione finale del documento, l'approvazione di esso deve essere effettuata dal *Responsabile di Progetto*[G]. Una volta approvato, il documento viene considerato ufficiale e può essere distribuito o utilizzato come riferimento.

Ogni fase del ciclo di vita del documento è fondamentale per garantire la qualità e l'accuratezza delle informazioni contenute. È importante seguire rigorosamente queste fasi per assicurare che il documento finale soddisfi tutti i requisiti e gli standard previsti.

3.1.3 Tipologie di documenti

3.1.3.1 Documenti ad uso Iinterno

I documenti ad uso interno sono destinati all'uso esclusivo dei membri del gruppo di progetto e del *committente*[G]. I principali documenti ad uso interno sono:

- Norme di Progetto
- Verbali interni

3.1.3.2 Documenti ad uso esterno

I documenti ad uso esterno sono destinati all'uso del committente e agli stakeholders. I principali documenti ad uso esterno sono:

- Analisi dei requisiti
- Piano di qualifica
- Piano di progetto
- Specifica tecnica
- Documento di accettazione

- Manuale utente
- Glossario
- Verbali esterni

3.1.4 Template

Sono stati creati dei *template*[G] per facilitare la stesura dei documenti. Ogni template contiene uno stile unico utilizzato da tutti i documenti che lo importano. Possono esistere molteplici template contemporaneamente; attualmente i principali template di documenti sono quello della Candidatura e quello comune per RTB e PB.

3.1.5 Struttura del documento

3.1.5.1 Frontespizio

La prima pagina di ogni documento contiene le seguenti informazioni:

- Logo del gruppo
- Titolo del documento
- Sottotitolo del documento (opzionale)
- Data di creazione del documento
- Versione del documento
- Nome e cognome dei redattori
- Nome e cognome dei verificatori
- Nome e cognome del responsabile di progetto che ha approvato quel documento
- Uso del documento (interno/esterno)
- Email del gruppo

Tutti i seguenti elementi sono disposti al centro del documento, versione e data di creazione sono disposti in linea; mentre il responsabile di progetto, i redattori e i verificatori sono disposti in righe separate di una tabella creata *ad hoc*[G].

3.1.5.2 Registro dei cambiamenti

Il registro delle modifiche contiene la cronologia delle modifiche apportate ai documenti scritti in maniera incrementale. È situato alla seconda pagina del documento sopra all'indice e contiene le seguenti informazioni:

- Versione del documento
- Data della modifica
- Nome e cognome dei redattori

- Breve descrizione della modifica
- Nome e cognome dei verificatori

Le informazioni sono disposte in colonne separate di una tabella creata ad hoc.

3.1.5.3 Indice

L'indice del documento è disposto dopo il frontespizio e prima del corpo del documento, contiene la lista dei capitoli e delle sezioni presenti nel documento, con i relativi numeri di pagina.

3.1.5.4 Corpo del documento

In ogni pagina del documento sono presenti i seguenti elementi:

- **Intestazione**
  - Nome del gruppo: posizionato in alto a sinistra della pagina
  - Titolo del documento: posizionato in alto al centro della pagina
  - Versione del documento: posizionato in alto a destra della pagina
- **Contenuto**
  - Testo del documento
- **Piè di pagina**
  - Numero di pagina: posizionato in basso a destra della pagina
  - Indirizzo email del gruppo: posizionato in basso a sinistra della pagina

3.1.5.5 Verbali

I verbali sono documenti che riassumono una riunione. Essi differiscono leggermente nello stile grafico del frontespizio degli altri documenti, e non contengono il registro dei cambiamenti.

3.1.6 Norme Tipografiche

3.1.6.1 Nome del documento

Ogni documento ha una denominazione omogenea. I verbali come nome file hanno la data del relativo incontro in formato YYYY-MM-DD, mentre gli altri documenti saranno denominati nel seguente modo:

`Nome_File_vX.Y.Z`

Per la denominazione dei file si fa riferimento alla convenzione *CamelCase*[G], le parole verranno separate dal carattere "_" (underscore) e la versione sarà indicata con la lettera "v" seguita dal numero di versione, specificato a sua volta nel paragrafo del Versionamento.

Per questioni di praticità, questo tipo di nomenclatura vale solo per i documenti ad uso interno ed esterno, non viene applicata ai template, alle immagini e altri tipi di documento. La nomenclatura dei documenti senza versionamento utilizza sempre gli underscore al posto degli spazi, ma senza utilizzare il CamelCase.

I verbali invece, riportano come nome del file la data dell'incontro in formato YYYY-MM-DD. Ad esempio: `2024-04-05` per il verbale del 5 aprile 2024.

3.1.6.2 Stile del testo

Il testo dei documenti deve essere scritto in lingua italiana, e utilizzerà i seguenti formati di testo:

- **Grassetto**: per evidenziare i titoli di sezioni
- ***Corsivo***: per far riferimento alla prima occorrenza di un termine del glossario o ad un altro documento
- **`Monospace`**: per riportare nomi di file, cartelle o elementi che richiamano alla stesura di codice

3.1.6.3 Elenchi

Gli elenchi seguono le seguenti norme:

- Il simbolo che scandisce ogni elemento di un elenco puntato è il pallino (•), al secondo e terzo livello si trovano rispettivamente il trattino(-) e il triangolo nero (‣); mentre per nelle liste numerate si utilizzano i numeri arabi per il primo livello, poi i numeri romani e infine le lettere minuscole
- Le voci iniziano per lettera maiuscola
- Le liste del tipo "Termine: descrizione" presentano il termine in grassetto con la prima lettera in maiuscolo

3.1.6.4 Sigle

Nella documentazione verranno utilizzate delle sigle per facilitare l'identificazione di un documento, ruolo o revisione senza doverne scrivere il nome per intero. Queste si sono rivelate utili soprattutto nell'utilizzo di tabelle.

Le sigle utilizzate sono le seguenti:

- Sigle relative ai documenti:
  - **AdR**: Analisi dei Requisiti
  - **NdP**: Norme di Progetto
  - **PdQ**: Piano di Qualifica

  - **PdP**: Piano di Progetto
  - **ST**: Specifica Tecnica
  - **MU**: Manuale Utente
  - **G**: Glossario
  - **MVP**: Minimum Valuable Product
  - **LdP**: Lettera di Presentazione
  - **VIXXXX-XX-XX**: Verbale Interno del XXXX-XX-XX
    - dove **XXXX-XX-XX** è la data del verbale in formato YYYY-MM-DD
  - **VEXXXX-XX-XX**: Verbale Esterno del XXXX-XX-XX
    - dove **XXXX-XX-XX** è la data del verbale in formato YYYY-MM-DD
  - **VI**: Verbale Interno
  - **VE**: Verbale Esterno
- Sigle relative alle revisioni:
  - **RTB**: Requirement and Technology Baseline
  - **PB**: Product Baseline
  - **CA**: Customer Acceptance
- Sigle relative ai ruoli:
  - **Re**: Responsabile
  - **Am**: Amministratore
  - **An**: Analista
  - **Prj**: Progettista
  - **Ve**: Verificatore
  - **Prg**: Programmatore

3.1.6.5 Elementi grafici

Vengono seguite le seguenti norme per utilizzare immagini, grafici e tabelle:

- **Immagini**: sono centrate e accompagnate da una didascalia
- **Diagrammi dei casi d'uso**: anch'essi verranno inseriti come immagine nel documento; verranno centrati, numerati e accompagnati da una didascalia
- **Tabelle**: come per i diagrammi, anch'esse sono centrate e numerate, con una didascalia che le descrive. Non è stato definito uno stile grafico standard per le tabelle, dunque è possibile che esso possa essere diverso per i vari documenti

- **Collegamenti ipertestuali**: sono riportati nel documento di colore blu, più precisamente del colore #005d88

3.1.7 Strumenti

Per la stesura dei documenti finora sono stati utilizzati i seguenti strumenti:

- **Typst**: linguaggio di markup simile a Markdown utilizzato per la stesura di documenti
- **Visual Studio Code**: IDE utilizzato per la scrittura del codice sorgente dei documenti
- **Typst.app**: sito web utilizzato come alternativa per la stesura dei documenti
- **GitHub Actions**: utilizzate per la generazione automatica dei file pdf derivanti dal codice sorgente dei documenti
- **Star UML**: utilizzato per la creazione dei diagrammi UML, i quali verranno poi esportati in formato PNG o SVG all'interno dei documenti

3.2 Gestione della configurazione

3.2.1 Descrizione

Secondo lo standard ISO/IEC 12207:1995, il processo di gestione della configurazione è un processo che applica procedure amministrative e tecniche durante l'intero ciclo di vita del software per:

- identificare, definire e stabilire baseline per gli elementi software di un sistema
- controllare le modifiche e i rilasci degli elementi
- registrare e riportare lo stato degli elementi e delle richieste di modifica
- garantire la completezza, la coerenza e la correttezza degli elementi
- controllare l'archiviazione, la gestione e la consegna degli elementi.

3.2.2 Versionamento

Per poter capire lo stato di avanzamento di un prodotto derivante dalle attività del progetto è necessario un identificatore. Il formato del numero di versione utilizzato è il seguente:

X . Y . Z

dove:

- **X**, **Y** e **Z** sono numeri interi positivi
- **X** corrisponde ad una versione approvata dal Responsabile di Progetto. La numerazione parte da 0.
- **Y** indica l'aggiunta di un incremento, senza però essere arrivati ad avere una versione rilasciabile del prodotto. La numerazione parte da 0 e si azzera ad ogni incremento di **X**.

- **Z** viene incrementato ad ogni piccola modifica o correzione, questo tipo di versione assume il nome di *minor*. La numerazione parte da 0 e si azzera ad ogni incremento di **X** o **Y**.
- Per evitare un eccessivo overhead burocratico, le versioni **Z** possono essere verificate insieme alle **Y**, mentre le versioni **X** devono sempre essere verificate prima di nuove modifiche.

3.2.3 Struttura delle repository

3.2.3.1 Documenti

La repository utilizzata dal gruppo per la creazione dei documenti è strutturata nel seguente modo:

- `Candidatura`: contiene i documenti relativi alla candidatura del gruppo per il capitolato.
  - `src`: contiene il codice sorgente documenti relativi alla candidatura.
  - `Verbali`: contiene i verbali esterni ed interni delle riunioni relativi alla candidatura.
- `RTB`: contiene i documenti relativi alla milestone RTB, quindi le norme di progetto, il piano di progetto e di qualifica, l'analisi dei requisiti e il glossario.
  - `Documentazione Esterna`: contiene i documenti ad uso interno relative alla milestone RTB.
    - `src`: contiene il codice sorgente dei documenti esterni relativi all'RTB.
      - `img`: contiene i file delle immagini a supporto dei documenti, ogni insieme di immagini è raggruppato in una cartella denominata con la sigla del documento al quale essa è associata.
    - `Verbali`: contiene i verbali esterni fatti con il proponente per discutere del capitolato e del *PoC*[G].
  - `Documentazione Interna`: contiene i verbali delle riunioni relative alla milestone RTB.
    - `src`: contiene il codice sorgente dei documenti interni relativi all'RTB.
    - `Verbali`: contiene i verbali delle riunioni relative alla milestone RTB.
- `PB`: contiene i documenti relativi alla milestone PB, quindi la specifica tecnica, il manuale utente, e la versione di rilascio successiva a quella della documentazione dell'RTB.
  - `Documentazione Esterna`: contiene i documenti ad uso interno relativi alla milestone PB.
    - `src`: contiene il codice sorgente dei documenti esterni relativi alla PB.

        - `img`: contiene i file delle immagini a supporto dei documenti, ogni insieme di immagini è raggruppato in una cartella denominata con la sigla del documento al quale essa è associata.
      - `Verbali`: contiene i verbali esterni fatti con il proponente per discutere del capitolato e dell'MVP.
    - `Documentazione Interna`: contiene i verbali delle riunioni relative alla milestone PB.
      - `src`: contiene il codice sorgente dei documenti interni relativi alla PB.
      - `Verbali`: contiene i verbali delle riunioni relative alla milestone PB.
- `assets`: contiene tutto ciò che è di supporto alla documentazione, come:
  - file di template per i documenti.
  - il logo del gruppo, utilizzato per tutta la repository.
- `README.md`: file che contiene le informazioni principali della repository, come la sua e la lista dei membri del gruppo.

La repository è pubblica e si può facilmente trovare al seguente link:
- https://github.com/nextsoftPD/Documenti

3.2.3.2 PoC

La repository del Proof of Concept è strutturata nel seguente modo:
- `API`: contiene il codice sorgente dell'API utilizzate dal plugin.
- `Plugin`: contiene il codice sorgente dell'estensione di Visual Studio Code.
- `TestProject`: contiene un progetto di test utilizzato per verificare il funzionamento del Proof of Concept.
- `README.md`: file che contiene le informazioni principali della repository, come la sua struttura e le istruzioni per l'utilizzo del Proof of Concept.

La repository è pubblica e si può facilmente raggiungere al seguente link:
- https://github.com/nextsoftPD/PoC

3.2.3.3 MVP

La repository dell'MVP è stata resa disponinibile solo al referente dell'azienda proponente, dunque non avrebbe senso spiegarne la struttura o includere il link.

3.2.4 Branch

Tutte le repository del gruppo si compongono di più *branch*[G], suddivisi ad hoc, in modo da garantire una separazione tra ciò che è stabile e verificato e ciò che è in fase di sviluppo.

Ogni membro può fare delle modifiche in un branch specifico a patto che esse siano relative solo ai *configuration items*[G] modificabili in quello specifico branch.

La suddivisione dei branch varia in base allo scopo della repository.

In nessuna repository è consentito modificare direttamente il branch principale, poiché porterebbe ad un elevato rischio di incongruenze e *merge conflicts*[G]. Si potranno applicare modifiche solo tramite il meccanismo di *pull request*[G], con verifica obbligatoria da parte di un verificatore, in modo da garantire che sia sempre presente una versione verificata e corretta del documento, anche se incompleta. Nel caso di una *minor*[G] invece, la verifica può essere svolta nel momento stesso in cui viene aggiunta una versione "stabile".

Per quanto riguarda cambiamenti minimali (punteggiatura, errori ortografici, ecc.) è permessa la modifica autonoma da parte dei verificatori, a patto che sia solo allo scopo di risolvere errori ortografici o per migliorare la comprensibilità di alcune frasi senza modificarne il significato logico. Questa decisione è stata presa al fine di evitare la creazione di numerose minor e di poter proseguire più velocemente con la stesura dei documenti.

La repository dei documenti è suddivisa in più *branch*[G] così definiti:

- **`main`**: il branch principale, che contiene l'ultima versione verificata di ogni documento
- **`nome_documento`**: branch che assume il nome del documento a cui fa riferimento, qui sarà possibile modificare solo il documento in questione.
- **`bugfix`**: branch secondario utilizzato per la correzione di errori di vario genere.

Mentre le repository contenenti del codice verranno suddivise nel seguente modo:

- **`main`**: il branch principale, che contiene l'ultima versione verificata del codice sorgente
- **`feature`**: branch secondario utilizzato per lo sviluppo di nuove funzionalità, il nome del branch sarà lo stesso della feature sviluppata e sarà eliminato una volta che la funzionalità è stata implementata e verificata.

3.2.5 Strumenti

Per il versionamento e la gestione del codice sorgente, si è scelto di utilizzare i seguenti strumenti:

- **Git**: software di controllo versione distribuito, utilizzato per tracciare le modifiche al codice sorgente.
- **GitHub**: piattaforma per la gestione dei repository Git, che consente la collaborazione tra i membri del gruppo e l'hosting del codice sorgente.

- **Visual Studio Code Source Control**: funzionalità integrata in Visual Studio Code per interagire con Git, che permette di gestire le modifiche, effettuare commit, creare branch e risolvere conflitti direttamente dall'IDE.

3.3 Gestione della Qualità

3.3.1 Descrizione

Il processo di gestione della qualità ha lo scopo di garantire che il prodotto software sviluppato sia funzionale, affidabile, sicuro, manutenibile e compatibile con l'ambiente in cui deve essere eseguito. Tale processo definisce un insieme di attività che vengono eseguite all'interno di un progetto per garantire la qualità sia sui prodotti che sui processi, in modo che il prodotto finale risulti conforme agli standard e alle norme in vigore.

3.3.2 Piano di Qualifica

Il documento del piano di qualifica verrà utilizzato per il *controllo della qualità*[G] su processi e prodotti dell'intero progetto, esso definisce le strategie e le metodologie di verifica e validazione adottate dal gruppo, verrà quindi utilizzato come riferimento per garantire che il processo di gestione della qualità raggiunga l'*economicità*[G].

3.3.3 Ciclo di Deming

Per mantenere un'alta qualità di lavoro si è stabilito l'utilizzo del *ciclo di Deming*[G] (o PDCA),un approccio continuo e costante al miglioramento della qualità dei processi e dei prodotti, basandosi su uno schema sistematico e iterativo che consiste di quattro punti:

- **Plan**: Si identificano gli obiettivi di miglioramento e si pianificano le azioni necessarie per raggiungerli. Si analizzano i processi attuali, si raccolgono dati e si individuano le aree che necessitano di miglioramenti. Il principale strumento utilizzato per la decisione e la pianificazione delle modifiche saranno i verbali interni.
- **Do**: si implementano le azioni pianificate nella fase precedente. Si eseguono le attività necessarie per apportare i miglioramenti, raccogliendo dati e documentando i risultati ottenuti.
- **Check**: si verificano i risultati ottenuti tramite risultati di test o tramite un'analisi con gli obiettivi stabiliti nella fase di pianificazione. Infine si analizzano i dati raccolti per determinare se le azioni intraprese hanno portato ai miglioramenti desiderati.
- **Act**: si decide se standardizzare le nuove pratiche integrandole nel way of working o apportare ulteriori modifiche. Se i risultati della fase di verifica dimostrano che le azioni intraprese hanno portato a miglioramenti significativi, queste diventano il nuovo

standard. Altrimenti, si ritorna alla fase di pianificazione per individuare nuove azioni correttive.

3.3.4 Strumenti

Per la gestione della qualità i principali strumenti adottati saranno le metriche definite nel documento di piano di qualifica.

3.3.5 Denominazione Metriche e Obiettivi

Metriche e obiettivi sono strumenti importanti per valutare la qualità del prodotto e dei processi. Le metriche sono misurazioni quantitative che forniscono informazioni sui prodotti e sui processi, mentre gli obiettivi sono traguardi specifici che il gruppo si propone di raggiungere. Entrambi sono fondamentali per monitorare le prestazioni e identificare aree di miglioramento. Le metriche e gli obiettivi vengono definiti in modo da essere misurabili e quantificabili, in modo da poter monitorare i progressi nel tempo. Le metriche possono riguardare vari aspetti del prodotto, come la qualità del codice, la copertura dei test, il numero di bug trovati e risolti, ecc. Gli obiettivi possono riguardare aspetti come la riduzione dei tempi di sviluppo, l'aumento della soddisfazione del cliente o il miglioramento della qualità del prodotto.

La denominazione delle metriche e degli obiettivi nei vari documenti segue questo formato:

[Sigla][Categoria][TipoProdotto][Numero]

dove:

- **[Sigla]**: indica che si tratta di una metrica o un obiettivo
  - **M**: per indicare una metrica
  - **O**: per indicare un obiettivo
- **[Categoria]**: indica a quale categoria appartiene la metrica, e può assumere i seguenti valori:
  - **PD**: per indicare i prodotti
  - **PC**: per indicare i processi
  - **TS**: per indicare i test
- **[TipoProdotto]**: indica se si riferisce a documenti o software ed è presente solo nel caso sia una metrica di prodotto. Può assumere i seguenti valori:
  - **D**: per indicare i documenti
  - **S**: per indicare i prodotti software
- **[Numero]**: rappresenta il codice numerico identificativo della metrica o dell'obiettivo, inizia da 1 per ogni tipologia di categoria e prodotto

3.4 Verifica

3.4.1 Descrizione

Secondo lo standard ISO/IEC 12207:1995, il processo di verifica è un processo per determinare se i prodotti software di un'attività soddisfano i requisiti o le condizioni imposte nelle attività precedenti. Per garantire efficacia in termini di costi e prestazioni, la verifica dovrebbe essere integrata, il prima possibile, con il processo che la utilizza (fornitura, sviluppo, ecc). La verifica del software verrà eseguita principalmente tramite analisi ed esecuzione di test.

3.4.2 Analisi statica

L'analisi statica è una tecnica di verifica del software che si effettua senza eseguire il codice. Si basa sull'esame del codice sorgente, della documentazione e di altri *artefatti*[G] del progetto per individuare errori, violazioni di standard e altre problematiche. Nell'analisi statica si distinguono due tecniche principali:

- **Walkthrough**: consiste in una ricerca generale di errori, analizzando l'insieme di tutti i configuration items. Durante un walkthrough, i membri del gruppo esaminano il codice o i documenti per identificare problemi evidenti, errori di logica, violazioni degli standard di codifica e altre anomalie.
- **Inspection**: consiste nell'eseguire una lettura mirata di una specifica unità di testing. Le inspection sono solitamente più brevi rispetto ai walkthrough. Durante un'inspection, un gruppo di revisori esamina il codice o alcuni paragrafi di documenti, utilizzando una *checklist*[G] di criteri predefiniti per identificare difetti. Ogni difetto trovato viene documentato e discusso, e vengono proposte soluzioni per correggerlo.

3.4.3 Analisi dinamica

L'analisi dinamica è applicabile solo al prodotto software in quanto prevede l'esecuzione di test, cioè prove sul codice in esecuzione. Un test definito correttamente deve:

- Essere ripetibile: dato un determinato input si deve ottenere sempre lo stesso output per ogni prova effettuata
- Specificare l'ambiente di esecuzione
- Identificare input e output richiesti
- Fornire informazioni utili sui risultati dell'esecuzione

Esistono diverse categorie di test, ognuno con scopo e oggetto di verifica differente.

3.4.3.1 Test di unità

I test di unità verificano la correttezza di una piccola parte di software testabile, chiamata unità, per stabilirne il corretto funzionamento rispetto alle attese. Le unità da noi definite per i test sono rappresentate dalle singole classi presenti nel prodotto software. I nostri test di unità mireranno a verificare il corretto funzionamento di ogni metodo definito in una specifica unità. Nel caso una classe comprenda delle dipendenze esterne, si utilizzeranno dei *mock*[G] per simulare il comportamento di tali dipendenze. Ogni test di unità viene riportato all'interno del documento del piano di qualifica.

3.4.3.2 Test di integrazione

I test di integrazione verificano il corretto funzionamento tra due o più unità del prodotto software. Questi test sono mirati a verificare che lo scambio di dati tra le istanze delle classi del prodotto software avvenga in modo corretto. Siccome questi test comprenderanno un gruppo di unità, è comunque possibile che vi siano dipendenze esterne inutili al fine della verifica del test. Di conseguenza abbiamo deciso di adottare comunque l'utilizzo di mock per simulare il comportamento di tali dipendenze. Ogni test di integrazione viene riportato all'interno del documento del piano di qualifica.

3.4.3.3 Test di sistema

I test di sistema verificano il sistema completo del prodotto software, prendendo in considerazione tutti i componenti e interfacce con altri sistemi. Nel nostro caso, verrà verificata la corretta integrazione tra frontend, backend e il server di Ollama, basandosi sui requisiti esistenti sul documento di *Analisi dei requisiti v2.0.0*. Tali test verranno eseguiti manualmente, in quanto non è possibile automatizzarli. Ogni test di sistema viene riportato all'interno del documento del piano di qualifica, e perchè si consideri superato, il sistema dovrà avere la capacità di soddisfare i requisiti associati a quel determinato test.

3.4.4 Strumenti

Gli strumenti principali a supporto del processo di verifica sono i seguenti:

- **Jest**: framework di testing JavaScript, utilizzato per eseguire test unitari e di integrazione nel backend.
- **Mocha**: framework di testing utilizzato per l'esecuzione dei test di integrazione nel frontend, è stato scelto in quanto supportato nativamente da Visual Studio Code per i test di unità e di integrazione.
- **Supertest**: utilizzato per il testing delle richieste API. È stato utilizzato per i test di integrazione riguardanti il funzionamento del server API.

- **NestJS**: è stato utilizzato sia per i test di integrazione sia per quelli di unità, in quanto riesce a fornire dei mock di alcuni moduli utilizzati come dipendenze esterne. Il suo utilizzo come strumento per i test è anche dovuto al fatto che il framework è stato utilizzato per la realizzazione del backend stesso.

3.4.5 Codice identificativo dei test

I test vengono identificati dal codice

T[Tipo]-[ID]

dove:

- **T**: indica che si tratta di un test
- **Tipo**: indica il tipo di test e può assumere i seguenti valori
  - **U** per i test di unità
  - **I** per i test d'integrazione
  - **S** per i test di sistema
  - **A** per i test di accettazione
- **ID**: codice numerico progressivo che inizia da 1 per ogni tipo di test.

3.4.6 Verifica della Documentazione

Per la verifica della documentazione, si utilizzeranno solamente le tecniche di inspection definite in precedenza. Quando verrà aggiornato un documento, verrà eseguita una lettura mirata ad un singolo paragrafo o ai cambiamenti registrati dal sistema di versionamento, per una verifica dei documenti più veloce. La verifica della documentazione si dice aver esito positivo quando:

- Il contenuto dei documenti vari documenti è coerente, completo ed esaustivo per tutti gli argomenti trattati da esso.
- I documenti sono stati redatti secondo le norme di progetto e soddisfano i criteri qualitativi del piano di qualifica.
- La gestione della configurazione dei documenti segue le procedure specificate nei paragrafi addietro.

3.5 Validazione

3.5.1 Descrizione

Il processo di validazione è necessario per determinare se il prodotto finale è pronto per l'utilizzo, è quindi necessario il superamento di tutti i test definiti nella fase di verifica per

assicurare che il prodotto implementi correttamente tutte le funzionalità richieste dal proponente, in modo da essere ritenuto accettabile come MVP.

3.5.1.1 Test di Accettazione

Il test di accettazione o collaudo è un'attività esterna, supervisionata dal committente e consiste nel dimostrare il soddisfacimento dei requisiti. Ogni test da eseguire sarà definito in un apposito documento, il quale riporterà la descrizione del test da effettuare insieme alla sua eventuale accettazione da parte del committente. Tale documento dovrà essere firmato dal rappresentante dell'azienda proponente per essere considerato valido.

4 Processi Organizzativi

4.1 Gestione dei Processi

4.1.1 Descrizione

Secondo lo standard ISO/IEC 12207:1995, il processo di gestione dei processi identifica le attività e i compiti di progetto necessari per gestire efficacemente i processi utilizzati dal gruppo. Le attività di gestione dei processi definite dallo standard sono le seguenti:

- Assegnazione dei ruoli di progetto
- Assegnazione delle task
- Gestione delle comunicazioni

4.1.2 Ruoli di progetto

Avendo adottato un modello agile per lo sviluppo, i ruoli di progetto sono assegnati ad ogni task, in modo da garantire una maggiore flessibilità e adattabilità alle esigenze del progetto. Ogni membro del gruppo avrà un ruolo definito in base ad accordi presi sulla ripartizione di determinate *task*[G] per quello sprint.

I ruoli che ciascun membro dovrà ricoprire sono i seguenti:

Responsabile di Progetto

Rappresenta l'intero progetto, punto di riferimento sia per il committente sia per il proponente, con lo scopo di mediare tra le due parti. Si assume la responsabilità delle scelte del gruppo dopo averle approvate. In particolare, si occupa di:

- Approvare l'emissione della documentazione
- Approvare l'offerta economica sottoposta al committente
- Pianificare e coordinare le attività di progetto
- Gestire le risorse umane

Amministratore di Progetto
Esegue delle procedure di controllo e amministrazione dell'ambiente di lavoro. In particolare, si occupa di:

- Risolvere problemi legati alla gestione dei processi
- Gestire la documentazione di progetto
- Automatizzare i processi
- Individuare punti di miglioramento nei processi
- Redigere e attuare i piani e le procedure per la gestione della qualità

Analista
Possiede maggiori competenze riguardo il dominio applicativo del problema. Si occupa di:

- Studiare il problema e il relativo contesto applicativo
- Comprendere il problema e definire la complessità e i requisiti
- Svolgere l'analisi dei requisiti e redigere il relativo documento

Progettista
Gestisce gli aspetti tecnologici e tecnici del progetto. In particolare si occupa di:

- Effettuare scelte riguardanti gli aspetti tecnici e tecnologici del progetto, favorendone l'efficacia e l'*efficienza*[G]
- Definire un'architettura del prodotto da sviluppare che miri all'economicità e alla manutenibilità a partire dal lavoro svolto dall'analista
- Documentare l'architettura descritta in precedenza nel documento di specifica tecnica
- Redigere la parte pragmatica del piano di qualifica

Verificatore
Verifica il lavoro svolto dagli altri componenti del gruppo, sulla base delle proprie competenze tecniche, esperienza e conoscenza delle norme. In particolare si occupa di:

- Esaminare i prodotti in fase di revisione, con l'ausilio delle tecniche e degli strumenti definiti nelle norme di progetto
- Verificare la conformità dei prodotti ai requisiti funzionali e di qualità
- Segnalare eventuali errori bloccando il processo di verifica

Programmatore
Svolge la fase di codifica del progetto e delle componenti di supporto che verranno utilizzate per eseguire prove di verifica e validazione sul prodotto. In particolare, si occupa di:

- Scrivere un codice che segui le best practices e gli standard di codifica definiti nel piano di qualifica e in questo documento

- Realizzare i test di unità, di integrazione, basandosi sull'architettura definita nella specifica tecnica e sulle norme definite in questo documento.

4.1.3 Issue Tracking System

Un *issue tracking system*[G] permette di tenere traccia delle attività e dei problemi legati al progetto. Esso consente di monitorare lo stato di avanzamento delle attività, assegnare compiti ai membri del gruppo e gestire le richieste di modifica. Il gruppo adotta *GitHub Projects*[G] per gestione semplice e chiara dei compiti da svolgere. Lo stato di avanzamento è segnato tramite le seguenti colonne:

- `To do`: contiene la lista delle task create dal responsabile.
- `In progress`: contiene le task che sono in fase di svolgimento.
- `In review`: contiene le task completate e in attesa di verifica.
- `Done`: contiene le task completate e verificate.

Ogni volta che è necessario portare a termine un compito, si segue questa procedura:

1. Il responsabile crea e assegna la task su *GitHub Issues*[G]
    i. una task può essere creata da chiunque, ma deve essere assegnata al responsabile per essere inserita nel progetto
    ii. una *issue*[G] può corrispondere a più task o ad una singola parte di una task
2. L'incaricato si assegna la task spostando la issue nella colonna `In progress`, segnando l'inizio del lavoro di produzione
    i. Sono possibili spostamenti in avanti di più colonne, aggiungendo un commento che ne spiega il motivo.
3. Finito il lavoro di produzione, viene aperta la pull request su GitHub per fare il merge dal branch sul quale si stava lavorando al branch principale
    i. Una pull request può corrispondere a più issues e quindi incorporare più task purché siano correlate tra loro
4. La task viene marcata come `In review` su GitHub Issues.
5. Il verificatore verifica la presenza di errori, incongruenze o bug all'interno del lavoro svolto.
    i. Se la verifica ha esito positivo:
        a. Il verificatore approva la pull request su GitHub con una review, che può essere richiesta da chi ha aperto la pull request
        b. Il cambiamento viene integrato sul branch principale, facendo il merge della pull request
        c. La issue viene marcata come `Done` su GitHub Issues

ii. Se la verifica ha esito negativo:
    a. Il verificatore richiede dei cambiamenti sul *commit*[G] della pull request
    b. Si ritorna al punto precedente

Considerazioni aggiuntive:

- Una pull request può essere approvata dalla persona che l'ha creata, oppure dall'amministratore di progetto in caso sia rimasta aperta per troppo tempo.
- Nel caso della verifica dei documenti, il verificatore aggiunge il suo nome nella lista dei verificatori del documento o nella colonna di verifica del versionamento
- Nel caso un branch contenga troppi errori, o ci siano state molte modifiche ad una pull request molto vecchia, il verificatore può richiedere all'amministratore di chiudere quella pull-request con un commento che ne spiega il motivo
- È possibile che il nome e il contenuto di una pull request possa essere modificato in un secondo momento, nel caso in cui si volessero utilizzare una pr già aperta per aggiungervi più task completate da verificare
- Nel caso in cui la verifica abbia esito negativo, quando verranno effettuate le modifiche, non sarà necessario aumentare di una versione, poiché il configuration item verrà ritenuto valido quando "stabile" e di conseguenza privo di errori o incongruenze
- I cambiamenti possono essere richiesti tramite un commento su github o possono essere accordati in una riunione o una breve chiamata

4.1.4 Gestione delle riunioni

Le riunioni sono un momento fondamentale per coordinare le attività del gruppo e prendere decisioni importanti. Esse si dividono in:

Riunioni interne

Riservate ai membri del gruppo di progetto e con lo scopo di:

- Coordinare le attività del gruppo
- Discutere e risolvere eventuali problemi
- Pianificare le attività future
- Verificare lo stato di avanzamento del progetto

Riunioni esterne

Che coinvolgono anche il Proponente o altre parti interessate e hanno lo scopo di:

- Presentare lo stato di avanzamento del progetto
- Discutere e approvare le decisioni importanti
- Ricevere feedback dal Proponente

La frequenza delle riunioni dipende dalle esigenze del progetto e può variare nel tempo. Nel nostro caso, le riunioni con il proponente vengono fissate quando si registrano avanzamenti significativi nel progetto o quando sorgono dubbi sulla progettazione o sull'implementazione del prodotto software.

Per ogni riunione viene redatto un verbale che riporta:
- Data e ora della riunione
- Argomenti trattati
- Decisioni prese
- Retrospettiva del precedente periodo produttivo
- Azioni da intraprendere
- Analisi delle criticità riscontrate

I verbali delle riunioni vengono archiviati e resi disponibili a tutti i membri del gruppo per garantire la trasparenza e la tracciabilità delle decisioni prese.

4.1.4.1 Gestione delle comunicazioni

Le comunicazioni all'interno del gruppo avvengono principalmente attraverso i seguenti canali:
- **Whatsapp**: app di messaggistica utilizzata per fissare le riunioni
- **Discord**: piattaforma di comunicazione utilizzata per discutere eventuali problemi e definire delle azioni da intraprendere di conseguenza

Per le comunicazioni formali con il proponente o altre parti interessate, si utilizzano principalmente:
- **Gmail**: client di posta utilizzato dal gruppo per fissare riunioni o per lo scambio di risorse e documenti con l'azienda proponente
- **Zoom**: per ottenere feedback sull'avanzamento del progetto e discutere eventuali dubbi

4.2 Strumenti

Tutti gli strumenti di supporto ai processi organizzativi utilizzati sono stati i seguenti:
- **GitHub**: per la gestione della repository e il versionamento del codice
- **GitHub Projects**: per il tracking delle issue e la gestione delle task
- **Zoom**: per le riunioni con il proponente e altre parti interessate
- **Whatsapp**: app di messaggistica utilizzata per fissare le riunioni interne
- **Discord**: per discutere eventuali problemi e definire delle azioni da intraprendere

- **Gmail**: client di posta utilizzato dal gruppo per fissare riunioni o per lo scambio di risorse e documenti con l'azienda proponente

4.3 Formazione del Personale

Ogni membro del gruppo sarà responsabile della propria formazione e dell'aggiornamento delle proprie competenze. Il gruppo si impegna a fornire supporto e risorse per la formazione continua, in modo da garantire che tutti i membri siano in grado di svolgere le proprie attività in modo efficace e efficiente.