SWEvenTeam

E-Mail
sweventeam@outlook.it

NORME DI PROGETTO

Informazioni documento

Versione	2.0.0
Redazione	Alessio Barraco Alessandro Damiani Yuri Lunardon Matteo Mazzotti Valentina Schivo Alessio Turetta
Verifica	Alessio Barraco Alessandro Damiani Alba Hui Larrosa Serrano Yuri Lunardon Matteo Mazzotti Valentina Schivo Alessio Turetta
Approvazione	Valentina Schivo

Storia del documento

Versione	Data	Autori	Verificatori	Descrizione
2.0.0	2025-06-09	Valentina Schivo	-	Approvazione documento
1.3.0	2025-06-06	Alessio Barraco	Alba Hui Larrosa Serrano	Controllo finale
1.2.1	2025-04-29	Matteo Mazzotti	Valentina Schivo	Aggiornamento regole stilistiche
1.2.0	2025-04-25	Yuri Lunardon, Valentina Schivo	Alessio Barraco	Aggiornamento strumenti
1.1.0	2025-04-22	Alessio Barraco, Alessandro Damiani	Alessio Turetta	Aggiornamento procedure
1.0.0	2025-03-20	Valentina Schivo	-	Approvazione per RTB
0.9.0	2025-03-17	Alessio Barraco	Yuri Lunardon, Matteo Mazzotti	Controllo finale
0.8.2	2025-02-11	Valentina Schivo	Alessio Barraco	Revisione e conclusione documento
0.8.1	2025-02-03	Alessandro Damiani	Alessio Barraco	Miglioramento stile documento
0.8.0	2025-01-15	Yuri Lunardon	Alessio Barraco	Miglioramento contenuti
0.7.0	2025-01-08	Alessio Barraco	Valentina Schivo	Scrittura sezione Metriche di qualità
0.6.0	2024-12-28	Valentina Schivo	Matteo Mazzotti	Scrittura sezione Standard per la qualità
0.5.1	2024-12-17	Alessio Barraco	Alessandro Damiani	Miglioramento organizzazione norme
0.5.0	2024-12-10	Yuri Lunardon	Alessio Turetta	Scrittura Processi organizzativi
0.4.0	2024-12-06	Alessio Turetta	Yuri Lunardon	Scrittura sezione Sviluppo
0.3.1	2024-12-02	Matteo Mazzotti	Valentina Schivo	Terminata sezione 3
0.3.0	2024-11-28	Yuri Lunardon	Alessio Turetta	Scrittura Verifica e Validazione
0.2.1	2024-11-23	Matteo Mazzotti	Alessandro Damiani	Scrittura sezione Documentazione (struttura e stile)
0.2.0	2024-11-22	Valentina Schivo	Alessio Barraco	Scrittura sezione Documentazione 3.1 (contenuto)

Versione	Data	Autori	Verificatori	Descrizione
0.1.1	2024-11-20	Matteo Mazzotti	Alessandro Damiani	Stesura sezione Formazione
0.1.0	2024-11-19	Alessio Barraco	Yuri Lunardon	Stesura sezione Fornitura
0.0.2	2024-11-18	Alessandro Damiani	Alessio Turetta	Introduzione documento
0.0.1	2024-11-15	Alessio Turetta	Yuri Lunardon	Definizione struttura

Indice

1 Introduzione

1.1 Scopo del documento

Le **Norme di Progetto** definiscono l'insieme di regole relative al Way of Working$_G$, quindi la messa in pratica da parte del gruppo delle best practice$_G$ da seguire per i processi durante lo svolgimento del progetto$_G$ didattico; le attività$_G$ svolte seguono lo standard ISO$_G$/IEC$_G$ 12207:1995, cardine nella comunità informatica, che definisce un framework$_G$ per i processi di ciclo di vita del software.
La stesura del documento si protrae incrementalmente, ovvero ogni capitolo e sottocapitolo è realizzato a pari passo con nuove aggiunte necessarie per il Way of Working, con opportuni cambiamenti alle regole precedentemente dichiarate secondo le decisioni del gruppo.

1.2 Scopo del progetto

Il progetto prevede lo sviluppo di una WebApp che valuta le performance di diversi Large Language Model$_G$ (LLM$_G$). In particolare l'applicazione si pone l'obiettivo di valutare le capacità di risposta$_G$ dei LLM, ai quali saranno sottoposte varie domande, solitamente considerate difficoltose da elaborare, a cui dovranno rispondere. Le risposte del LLM saranno poi valutate dal sistema$_G$ e verrà restituito un indice di correttezza delle risposte del LLM, in modo tale da capire se le risposte generate sono soddisfacenti o meno. L'applicazione può dunque essere vista come uno strumento di test$_G$ per i LLM.
L'azienda proponente$_G$ intende utilizzare il risultato$_G$ del progetto per verificare e confrontare come le prestazioni dei LLM variano in base alle loro caratteristiche, ad esempio il numero di parametri e l'addestramento ricevuto.

1.3 Glossario

Per mantenere la consistenza nell'uso dei termini specifici al progetto didattico sui documenti, è presente una lista di definizioni dei termini specifici del dominio d'uso su un glossario. Il Glossario risulterà utile fornendo delle definizioni chiare e approvate da tutto il gruppo. La presenza di un termine all'interno del Glossario viene indicata con questo stile$_G$.

1.4 Riferimenti

1.4.1 Riferimenti normativi

- **Capitolato d'appalto C1: ArtificialQI**
  https://www.math.unipd.it/~tullio/IS-1/2024/Progetto/C1.pdf
  Ultima consultazione: 2025-06-06;
- **Standard ISO/IEC 12207:1995**
  https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf
  Ultima consultazione: 2025-06-06.

1.4.2 Riferimenti informativi

- **Glossario v2.0.0**
  https://sweventeam17.github.io/glossario.html
  Ultima consultazione: 2025-06-06;
- **Capitolato d'appalto C1: ArtificialQI**
  https://www.math.unipd.it/~tullio/IS-1/2024/Progetto/C1.pdf
  Ultima consultazione: 2025-06-06;
- **Processi di ciclo di vita**
  https://www.math.unipd.it/~tullio/IS-1/2024/Dispense/T02.pdf
  Ultima consultazione: 2025-06-06.

2 Processi primari

2.1 Fornitura

2.1.1 Descrizione

Secondo lo standard ISO/IEC 12207:1995$_G$, il processo$_G$ di fornitura contiene le attività e i compiti del fornitore$_G$. Questo processo determina anche le procedure e le risorse necessarie per gestire e garantire lo sviluppo del progetto.

2.1.2 Attività

Il processo di fornitura prevede le seguenti attività:

- **Avvio**: vengono individuate le esigenze del cliente e viene condotta un'analisi dei requisiti$_G$;
- **Preparazione della risposta**: viene preparata una risposta dettagliata alle necessità del cliente, vengono individuate le modalità di esecuzione, i costi e i tempi di consegna;
- **Contratto**: viene formalizzato un accordo tra fornitore e cliente;
- **Pianificazione**: vengono pianificate tutte le attività necessarie per soddisfare requisiti, tempi di progettazione, risorse e costi;
- **Esecuzione e controllo**: le attività vengono effettivamente implementate seguendo il piano stabilito e viene effettuato un controllo continuo per monitorare l'avanzamento del progetto, risolvere eventuali problemi e garantire che si stiano rispettando i requisiti concordati;
- **Revisione e valutazione**: viene eseguita una revisione e una valutazione del lavoro svolto e viene verificato l'effettivo raggiungimento degli obiettivi prefissati;
- **Consegna e completamento**: il progetto viene completato e il prodotto viene consegnato al cliente. Si conclude con la verifica finale, la chiusura amministrativa e, se necessario, il supporto post-consegna.

2.1.3 Comunicazioni con il Proponente

Il gruppo SWEvenTeam stabilirà una comunicazione stretta, su richiesta, con l'azienda proponente Zucchetti per ottenere un riscontro sul lavoro svolto e verificare che quanto prodotto soddisfi le esigenze.
L'azienda proponente Zucchetti mette a disposizione un indirizzo email per le comunicazioni e le richieste formali.
Gli incontri vengono richiesti dal gruppo all'azienda tramite comunicazione via email. Come canale di videoconferenza è stato determinato l'utilizzo della piattaforma$_G$ Google Meet$_G$.
Tali incontri permettono la discussione dello stato di avanzamento del progetto e delle difficoltà eventualmente incontrate durante lo sviluppo, oltre che a risultare importanti per la buona riuscita dell'analisi dei requisiti.
Per ogni incontro fissato con l'azienda proponente viene redatto un verbale esterno che riassume argomenti trattati e attività concordate.
Tali verbali esterni sono archiviati all'interno del repository$_G$ documentazione del gruppo SWEvenTeam.

2.1.4 Documentazione

Di seguito vengono elencati i documenti del gruppo SWEvenTeam che verranno consegnati ai committenti Prof. Tullio Vardanega e Prof. Riccardo Cardin e all'azienda proponente Zucchetti.

2.1.4.1 Valutazione dei Capitolati

Il documento Studio dei Capitolati contiene un'analisi svolta dal gruppo SWEvenTeam dei capitolati proposti.

- **Informazioni del progetto**: una sezione che contiene una spiegazione riassuntiva dello scopo del capitolato$_G$;
- **Tecnologie previste**: contiene le tecnologie suggerite;
- **Vantaggi e svantaggi**: una lista di considerazioni del gruppo sul capitolato analizzato.

2.1.4.2 Norme di Progetto

Documento che definisce le norme adottate dal gruppo per garantire un lavoro coordinato ed efficiente. Le norme cercano di perseguire le best practice comunemente note all'interno del dominio ingegneria del software.

2.1.4.3 Analisi dei Requisiti

Documento che si pone l'obiettivo di formalizzare i requisiti espliciti e impliciti richiesti dal capitolato. Vengono qui riportati i requisiti funzionali$_G$, di qualità$_G$, di vincolo e prestazionali e il loro tracciamento nei rispettivi casi d'uso.
Il documento è strutturato nel seguente modo:

- **Descrizione del prodotto**: viene descritto quale è lo scopo finale del prodotto e tutte le funzionalità$_G$ principali che è capace di fornire;
- **Analisi e studio degli utenti**: analisi degli utenti che potranno interagire con il prodotto;
- **Lista dei casi d'uso**: identificazione di tutti i possibili casi di scenario nel quale l'utente che decide di utilizzare l'applicazione può imbattersi. In questo modo si delineano tutti i casi possibili e gli sviluppi che si possono verificare, capendo tutte le esigenze da gestire e soddisfare lato software;
- **Lista dei requisiti e tracciamento**: tutte le richieste o vincoli definiti dal proponente o dedotti dal team per la realizzazione del prodotto finale. I requisiti possono essere obbligatori, desiderabili e opzionali.

2.1.4.4 Glossario

Il glossario si pone l'obiettivo di minimizzare le incomprensioni tra i membri del gruppo, l'azienda e i professori, su ciò che viene scritto e in particolare sulla terminologia specifica, utilizzata sia nelle discussioni del gruppo che nella scrittura della documentazione. Ha quindi lo scopo di perseguire chiarezza e unicità.

2.1.4.5 Piano di Progetto

Il Piano di Progetto è un documento che ha lo scopo di pianificare e rendicontare le attività di progetto. Viene redatto lungo il corso di tutto il progetto dal responsabile$_G$ in collaborazione con gli amministratori.
l Piano di Progetto è strutturato nel seguente modo:

- **Analisi dei Rischi**: individuazione dei possibili rischi che potrebbero ostacolare la buona riuscita e il giusto flusso di lavoro durante la durata del progetto. Per questi rischi, sono previste delle soluzioni elaborate dal team, che mirano a mitigare e inibire i rischi. Esempi di rischi possono essere di tipo organizzativo e tecnologico;
- **Scadenze e stima dei costi**: sono riportate le scadenze di consegna previste e la stima dei costi di realizzazione del progetto;
- **Modello di sviluppo**: viene presentato il modello di sviluppo scelto per svolgere il progetto e i motivi che hanno portato a tale decisione;

- **Pianificazione e consuntivo**: vengono riportati tutti i periodi del progetto, includendo informazioni utili come la pianificazione di obiettivi, i rischi attesi, il preventivo$_G$ orario, il consuntivo$_G$, i rischi incontrati e la retrospettiva contenente anche le risorse rimanenti.

2.1.4.6 Piano di Qualifica

Il Piano di Qualifica è un documento che descrive le qualità che i prodotti del progetto devono rispettare e garantire per assicurare conformità alle specifiche richieste e alle aspettative del committente$_G$. Ogni membro del gruppo dovrà tenere in considerazione quanto scritto in questo documento durante la fase di produzione,e il verificatore$_G$, durante il compito di verifica, terrà in considerazione le qualità espresse in questo documento.
Il Piano di Qualifica è strutturato nel seguente modo:

- **Obiettivi metrici di qualità**: stabiliscono le metriche e i parametri per garantire processi e un prodotto finale di alta qualità;
- **Specifica dei test**: descrizione dettagliata dei test utilizzati;
- **Cruscotto di controllo della qualità**: aiuta a comprendere l'andamento del progetto, anche attraverso la visualizzazione di grafici, e a trovare delle misure correttive.

2.1.4.7 Lettera di Presentazione

La lettera di Presentazione è il documento che comunica ai committenti la volontà di partecipare a una delle fasi di revisione del progetto per presentare la documentazione e il prodotto software ai committenti, il Prof. Vardanega e il Prof. Cardin. Il documento conterrà un elenco della documentazione redatta e i termini di consegna del prodotto ultimato, oltre che a un puntatore al repository GitHub$_G$ contenente il PoC$_G$, e un puntatore al sito del gruppo.

2.1.4.8 Specifica Tecnica

Il documento ha come obiettivo quello di illustrare e motivare le scelte architetturali e di design adottate nello sviluppo della web-app. All'interno dovrà descrivere l'architettura$_G$ logica e di deployment$_G$, i design pattern e le tecnologie utilizzate, corredando la trattazione con diagrammi di classe per chiarire la struttura del software. La specifica tecnica sarà strutturata nel seguente modo:

- **Tecnologie**: in questa sezione vengono presentate le tecnologie utilizzate per implementare il prodotto software desiderato dal capitolato;
- **Architettura**: verrà presentata l'architettura di sistema e di deployment, spiegando la struttura del backend$_G$ e del frontend$_G$, elencando anche i design pattern utilizzati. Saranno presentati i diagrammi delle classi ritenuti maggiormente importanti all'interno del sistema.

2.1.4.9 Manuale Utente

Documento che contiene tutte le indicazioni utili all'uso della web-app, assicurando agli utenti una esperienza ottimale e la possibilità di sfruttarne appieno ogni funzionalità. Al suo interno verranno trattate le procedure per poter utilizzare la web-app, guidando l'utente nella fase di deploy, e verranno riportate spiegazioni dettagliate su tutte le funzionalità che può offrire, mostrando come possono essere utilizzate e tutte le possibili interazioni che possono avvenire attraverso l'interfaccia.

2.1.5 Strumenti

Gli strumenti utilizzati per la fase di fornitura sono:

- **LaTeX**: linguaggio di marcatura usato per la preparazione di testi e la scrittura della documentazione;
- **Google Docs**: strumento collaborativo usato per la scrittura della documentazione;
- **Discord**: servizio$_G$ VoIP, messaggistica istantanea e distribuzione digitale, utilizzato dal gruppo per gli incontri interni;

- **Telegram**: servizio di messaggistica istantanea, utilizzata dal gruppo per lo scambio di messaggi e la coordinazione asincrona del lavoro;
- **Google Meet**: servizio di videoconferenza utilizzato per gli incontri con l'azienda proponente.

2.2 Sviluppo

2.2.1 Descrizione

Il processo primario di sviluppo contiene le attività che il fornitore deve completare nella fase di sviluppo effettiva del prodotto software. Si struttura nelle seguenti attività:

- **Implementazione del processo**: vengono scelti il ciclo di vita adatto al progetto, gli standard specifici da seguire, gli strumenti e i linguaggi di programmazione adatti. Prevede l'ideazione di piani dettagliati per orientare il processo di sviluppo.
- **Analisi dei requisiti**: viene svolta un'analisi sull'utilizzo finale del sistema con il proponente allo scopo di trovare i requisiti e bisogni dell'utente finale in relazione a quanto previsto.
- **Progettazione dell'architettura del sistema**: è previsto uno studio macroscopico del sistema per individuare gli elementi hardware e software necessari per soddisfare i requisiti precedentemente individuati, eseguendo anche un tracciamento per verificarne la correttezza.
- **Analisi dei requisiti software**: vengono analizzati i requisiti prestazionali, di sicurezza, il modo in cui i requisiti utente$_G$ sono soddisfatti dal software e l'ambiente dove verrà eseguito.
- **Progettazione dell'architettura del software**: i requisiti vengono convertiti in un'architettura ad alto livello, che presenta come i moduli interagiscono tra di loro, assegnando i requisiti ai componenti predisposti, che verranno definiti nella progettazione di dettaglio. Viene inoltre progettato come il sistema si interfaccerà con il database$_G$ e i sistemi esterni.
- **Progettazione di dettaglio del software**: avviene la progettazione in dettaglio delle componenti descritte nella progettazione generale, andando a definire la struttura fino alle unità più piccole.
- **Codifica e test del software**: vengono sviluppate, secondo le best practice di programmazione, le componenti definite nelle fasi precedenti di progettazione. Vengono sviluppati i test per ciascuna unità, ricercando tracciabilità dei requisiti, qualità e futura integrazione tra le componenti.
- **Integrazione del software**: viene eseguita l'integrazione delle unità di programmazione con gli altri elementi del sistema. Una corretta integrazione viene raggiunta attraverso l'uso di test di integrazione$_G$, che verificano la buona riuscita o meno dell'attività.
- **Test di qualifica del software**: viene verificato che ogni componente software sia conforme ai requisiti di qualità, attraverso specifici test.
- **Integrazione di sistema**: sono eseguiti i test di integrazione per verificare una corretta integrazione di tutte le componenti nel sistema completo.
- **Test di qualifica del sistema**: sono eseguiti i test di sistema$_G$ per verificare la correttezza del sistema sviluppato e delle sue qualità.
- **Installazione del software**: viene effettuato il deploy del sistema, secondo quanto concordato con il proponente, nell'ambiente designato. Viene fornito anche un aiuto nella configurazione del sistema.
- **Supporto all'accettazione del software**: viene verificato da parte del cliente, assieme al fornitore, che il sistema rispetti tutti i requisiti richiesti.

2.2.2 Analisi dei requisiti

Data l'importanza che questa attività ricopre all'interno della prima fase di revisione del progetto, l'RTB$_G$, il gruppo desidera approfondire e ampliare la descrizione di tale attività, fornendo ulteriori dettagli e informazioni.
L'Analisi dei requisiti ha l'obiettivo di individuare tutti i requisiti fondamentali che il sistema deve rispettare, andando a definire così i punti cardine su cui i progettisti e i programmatori andranno a lavorare, i primi cercando di progettare architetture e componenti in grado di soddisfare i requisiti emersi, i secondi andando a implementare quanto determinato dai progettisti. Risulta chiara l'importanza di questa attività, che si pone alla base delle altre, e si capisce dunque quanto sia importante svolgere un analisi dei requisiti completa e corretta, per poi completare la progettazione e codifica in modo efficace ed esatto. Non è un caso che uno dei motivi più comuni per cui i progetti di ingegneria del software falliscono, sia una errata o insufficiente analisi dei requisiti. I risultati dell'attività di analisi dei requisiti sono raccolti nell'omonimo documento di Analisi dei Requisiti che dovrà includere le seguenti informazioni:

- **Descrizione del prodotto**: Scopo finale del prodotto e funzionalità principali offerte;
- **Analisi degli utenti**: Studio degli utenti che potranno interagire con il sistema;
- **Lista dei casi d'uso**: Identificazione di tutti i possibili scenari d'uso dell'applicazione. Questo permette di delineare le varie situazioni che l'utente può incontrare e di comprendere le esigenze da gestire a livello software;
- **Lista dei requisiti**: Identificazione di tutti i requisiti, che andranno soddisfatti, emersi dal capitolato, da decisioni interne e dalle discussioni con il proponente.

I casi d'uso verranno strutturati nel seguente modo:

- **Titolo**: Identificativo numerico (ad esempio, UC 1) e titolo breve descrittivo;
- **Attori**: Attori principali che avviano il caso d'uso;
- **Attori secondari**: Eventuali attori coinvolti nello svolgimento del caso d'uso;
- **Obiettivi**: Descrizione dettagliata del caso d'uso e del suo obiettivo;
- **Precondizioni**: Stato del sistema prima dell'esecuzione del caso d'uso;
- **Scenario**: Sequenza di passi svolti dall'attore$_G$ risultanti dall'interazione con il sistema;
- **Postcondizioni**: Stato del sistema al termine del caso d'uso;
- **Esclusioni**: Eventuali esclusioni inerenti al caso d'uso, ovvero casi d'uso alternativi che deviano il normale corso del caso d'uso.

I requisiti possono essere classificati dal gruppo, in obbligatori, desiderabili o opzionali, in base alla loro importanza, e dovranno includere:

- **Codice alfanumerico**, con formato R[T]-[N], dove T indica la tipologia del requisito$_G$ (F = funzionale, Q = qualità, V = vincolo, P = prestazionale) ed N il numero progressivo del requisito per quella tipologia;
- **Descrizione generale del requisito**, secondo quanto stabilito dal capitolato e dalle decisioni interne del gruppo, discusse con l'azienda;
- **Classificazione dell'importanza**, secondo quanto stabilito dal capitolato e dalle decisioni interne del gruppo, discusse con l'azienda;
- **Fonte**, da cui è stato estrapolato il requisito.

Una volta sviluppati, i risultati dell'analisi dei requisiti devono essere sottoposti a revisione da parte dell'azienda. Il feedback ottenuto darà un riscontro chiaro sulla aderenza di quanto elaborato con le volontà dell'azienda, in particolare verificando l'esattezza e completezza dei casi d'uso e dei requisiti, dei quali verrà valutata anche la correttezza della loro classificazione in base all'importanza.

2.2.3 Progettazione

La progettazione ha lo scopo di modellare la soluzione realizzativa ottimale che soddisfi i requisiti precedentemente analizzati.
Il primo passo è l'individuazione delle tecnologie da utilizzare, dopo un'attenta valutazione dei pro e contro. Fatto questo, viene realizzata l'architettura del prodotto: partendo da una visione ad alto livello e, successivamente, andando nel dettaglio.
Risulta fondamentale un'architettura che sia aperta al cambiamento, chiave è la modularità e la riduzione delle dipendenze. Un forte aiuto è dato dai design pattern che presentano soluzioni allo stato dell'arte a problemi frequenti nella progettazione di un prodotto software. La loro adozione migliora l'architettura ed implicitamente riduce la probabilità di incorrere in errori.
Parte integrante della progettazione è la definizione dei test che consentono così di sviluppare un prodotto aderente alle specifiche e di individuare i problemi prima di eseguire il rilascio.

2.2.4 Codifica

La codifica prevede l'uso dei seguenti linguaggi: Python per la parte di backend e JavaScript per la parte di frontend. Le regole per lo stile di codifica sono le seguenti:

- Preferire funzioni brevi e ben definite;
- Evitare, se possibile, l'uso di variabili globali;
- Commentare usando la lingua italiana;
- Ogni funzione necessita di un commento che descriva il funzionamento;
- Evitare più istruzioni per linea di codice.

Per ciascun linguaggio di programmazione sono disponibili degli strumenti (indicati nella sezione sottostante) che permettono di seguire e verificare il rispetto della sintassi e dello stile, oltre che ad individuare potenziali bug$_G$.

2.2.5 Strumenti

Durante lo svolgimento dei compiti vengono utilizzati i seguenti strumenti:

- **StarUML**: strumento di ingegneria del software per la modellazione di sistemi che utilizza UML$_G$;
- **Ollama**: piattaforma che consente di eseguire modelli linguistici (LLM) localmente. Permette di avere accesso a modelli già pre-addestrati e offre un'interfaccia per interagire con essi;
- **LangChain**: software per facilitare l'integrazione di grandi LLM all'interno di un prodotto software;
- **SQLite**: database relazionale leggero che viene spesso usato in applicazioni che non richiedono un database server complesso;
- **Next.js**: framework di React$_G$, per la realizzazione dell'interfaccia front end;
- **Django Rest Framework**: framework web per lo sviluppo del back end;
- **Docker**: piattaforma open-source che consente di sviluppare, distribuire ed eseguire applicazioni all'interno di container$_G$;
- **Gemini**: famiglia di modelli di intelligenza artificiale sviluppata da Google DeepMind;

- **ESLint**: strumento di analisi statica per JavaScript e TypeScript che identifica e corregge automaticamente errori di sintassi, problemi di stile e potenziali bug;
- **Pylint**: strumento di analisi statica per Python che rileva errori di codice, violazioni di stile e potenziali bug;
- **Prettier**: formattatore di codice automatizzato per JavaScript, TypeScript, HTML, CSS e altri linguaggi;
- **Black**: formattatore di codice per Python che applica automaticamente uno stile di formattazione coerente;
- **Visual Studio Code**: editor di testo per la codifica del prodotto.

3 Processi di supporto

3.1 Documentazione

3.1.1 Introduzione

La documentazione è lo strumento di supporto testuale dedito a fornire, in modo chiaro e conciso, informazioni utili a tutti gli stakeholders.

3.1.2 Scopo

La documentazione ha lo scopo di perseguire tali obiettivi:

- Raccogliere le norme predisposte dal team affinché tutti possano adattarsi a quanto scritto senza rischio$_G$ di incomprensioni;
- Porre dei limiti chiari e rispettati da tutti, chiarendo cosa può essere fatto e cosa no e quali sono i limiti da rispettare all'interno del progetto;
- Analisi totale del prodotto attraverso un metodo comunicativo chiaro a tutti;
- Riportare tutti i risultati dei processi e delle attività svolte lungo tutto il ciclo di vita;
- Confermare la bontà di quanto prodotto, perseguendo aspettative di qualità dichiarate in documentazione.

Nelle prossime sezioni verrà definito l'insieme di regole che i redattori e i verificatori, dovranno seguire quando si apprestano a scrivere, o verificare, delle sezioni di documentazione. Il rispetto di tali norme permetterà alla documentazione di essere uniforme, chiara e ben strutturata, fornendo così un contenuto di qualità. Tutti i membri del team sono tenuti a rispettare le norme definite con lo scopo di raggiungere il miglior risultato possibile.

3.1.3 Ciclo di vita dei documenti

Il ciclo di vita di un documento è contraddistinto da 4 stati in cui si può trovare e ogni stato ha delle proprie attività:

- **Preparazione**: a un certo punto dell'avanzamento del progetto emerge la necessità di redigere della documentazione, dunque si procede a costruire o ad adattare un template che sia consono alla struttura e formattazione del documento. Viene poi svolta un'attività di pianificazione del documento, dove si determina sommariamente ciò che verrà scritto, facendo emergere così una struttura chiara, che viene esplicitata dalla suddivisione in sezioni e sottosezioni, accompagnate da titoli che ne specificano l'argomento. Viene inoltre svolta una preliminare discussione su ciò che verrà scritto all'interno di ogni sezione e sottosezione, in modo da avere chiara la composizione del documento una volta che si inizierà l'attività successiva;

- **Redazione**: inizialmente i redattori del documento si incontrano e scrivono un primo abbozzo del documento, che specifica nel concreto i punti che verranno trattati in ogni sezione, definendo così una traccia comune e condivisa da tutti. In seguito ogni redattore$_G$ procede alla scrittura effettiva del documento, dove vengono trattati più nel dettaglio e riformulati i punti discussi nella fase precedente di analisi delle sezioni, raggiungendo un testo completo e soddisfacente;
- **Verifica**: terminata la fase di redazione del documento o di una sua sezione, si entra nella fase di verifica, nella quale un verificatore controlla che quanto scritto dai redattori sia corretto e che rispetti le Norme di Progetto$_G$. Nel caso in cui quanto scritto non vada bene, il verificatore riporterà cosa non va al redattore, che dovrà dunque correggere e riformulare quanto scritto, altrimenti la fase di verifica produrrà esito positivo e quella parte di documento viene giudicata verificata. La verifica finale del documento prende il nome di approvazione e viene svolta dal responsabile. La sua approvazione sancisce il rilascio della versione finale del documento;
- **Caricamento sul sito**: quando il documento è pronto per il suo rilascio finale (al termine della RTB o PB), viene caricato sul sito.
  Tale operazione viene fatta caricando il documento nella repository documentazione e successivamente eseguendo la seguente procedura$_G$:
  1. Andare nella repository del sito;
  2. Portarsi all'interno della cartella di collegamento a documentazione pdf@;
  3. Digitare il comando:
     ```
     \href{https://sweventeam17.github.io/glossario.html\#git}{\textcolor{black}{\underline
     ```
  4. Digitare il comando:
     ```
     cd ..
     ```
  5. Digitare il comando:
     ```
     git submodule update --recursive --remote
     ```
  6. Eseguire il push$_G$ delle modifiche.

  A questo punto, tramite le GitHub Actions (nell'arco di qualche minuto), l'automazione provvederà ad aggiornare il sito.
- **Manutenzione**: si può verificare il caso che un documento abbia la necessità di essere modificato, dunque verrà ritrattato e rivisto secondo le regole di versionamento$_G$, ricominciando ad avanzare tra gli stati precedentemente descritti.

3.1.4 Ruoli

Nel processo di documentazione si distinguono 4 ruoli:

- **Redattore**: colui che si occupa della stesura del contenuto di un documento o di una sezione di un documento;
- **Verificatore**: colui che si occupa di verificare la correttezza di quanto scritto dai redattori. Se il verificatore non approva il lavoro del redattore, lascia dei feedback contenenti i problemi rilevati;
- **Responsabile**: colui che si occupa di decidere quando debba essere scritto un documento o una sua sezione, definendo anche le relative scadenze e i compiti di coloro che si appresteranno a lavorarci. Inoltre ha potere di approvazione finale, potendo scegliere se accettare o richiedere modifica di quanto svolto dai redattori e dai verificatori, nel periodo di competenza del suo incarico;
- **Amministratore**: colui che ha il compito di gestire gli strumenti tecnologici utili ad assegnare ad ogni componente la propria mansione. Si occupa principalmente della gestione dei documenti e delle issues su GitHub.

3.1.5 Sistema di composizione tipografica

Per la composizione tipografica dei documenti, il nostro gruppo ha deciso di utilizzare **LaTeX**, riconoscendone tali vantaggi:

- **Potenza e flessibilità**: LaTeX$_G$ è ideale per creare documenti complessi e strutturati;
- **Qualità tipografica elevata**: garantisce una formattazione professionale, particolarmente adatta a documenti tecnici.

L'utilizzo di LaTeX garantisce una gestione precisa della formattazione e della struttura del documento, liberando i redattori dalla necessità di concentrarsi sulla resa grafica e assicurando coerenza e professionalità nella documentazione del progetto.

Il template che è stato scritto e utilizzato è presente nella repository docs, nella cartella template.

Per la stesura dei documenti viene usato un documento Google, con lo scopo di raggiungere una collaborazione efficace tra i vari redattori.

3.1.6 Struttura del documento

Ogni documento viene prodotto secondo una determinata struttura, distinta dalle seguenti sezioni:

3.1.6.1 Intestazione

La prima pagina ha lo scopo di intestazione del documento e contiene le seguenti informazioni:

- **Logo e nome del team**:
- **E-mail del team**:
- **Nome del documento**:
- **Versione attuale del documento**:
- **Redattori che hanno contribuito**:
- **Verificatori che hanno contribuito**:
- **Approvazione**: chi ha dato l'approvazione finale al documento.

3.1.6.2 Testa di pagina

Ogni pagina seguente all'intestazione contiene in testa il logo del gruppo e il nome del documento.

3.1.6.3 Storia del documento

La seconda sezione è dedicata alla storia del documento. Le informazioni sono organizzate in una tabella e permettono di tenere traccia dei cambiamenti subiti dal documento.
La tabella riporta i seguenti dati:

- **Versione**: il numero di versione del documento;
- **Data**: data di modifica del documento;
- **Autori**: chi ha effettuato le modifiche;
- **Verificatori**: chi ha approvato le modifiche;
- **Descrizione**: una breve descrizione.

3.1.6.4 Indice

In seguito è presente l'indice che elenca le sezioni e le sottosezioni del contenuto del documento. Ogni sezione è raggiungibile tramite click. Se nel documento saranno presenti immagini o tabelle, verranno forniti indici separati per entrambi.

3.1.6.5 Corpo del documento

Segue all'indice il corpo del documento, ovvero il contenuto stesso, diviso in capitoli, sezioni e sottosezioni.

3.1.6.6 Documenti del progetto

Verranno prodotti i seguenti documenti di progetto:

- Norme di Progetto;
- Piano di Progetto;
- Piano di Qualifica;
- Analisi dei Requisiti;
- Glossario;
- Verbali Interni;
- Verbali Esterni.

3.1.6.7 Verbali

I verbali sono documenti redatti a seguito dei meeting tra i membri del gruppo e costituiscono un resoconto accurato dell'attività comunicativa. Essi riportano in dettaglio gli argomenti trattati, le decisioni prese, le discussioni su scelte precedenti e le attività da svolgere in futuro.
Si distinguono verbali interni, destinati a registrare i contenuti degli incontri interni al gruppo, e verbali esterni, destinati a registrare i contenuti degli incontri svolti dal gruppo con enti esterni, come l'azienda.

La struttura dei verbali differisce dagli altri documenti per una serie di cambiamenti applicati, che aggiungono maggiori informazioni inerenti al meeting.

Il primo cambiamento è la sezione informazioni riunione, una sezione scritta in seguito al intestazione del documento, che contiene informazioni come:

- **Luogo**: luogo fisico o virtuale dove è stato svolto l'incontro;
- **Data**: data di svolgimento dell'incontro;
- **Orario**: orario in cui si è svolto l'incontro;
- **Presenti**: elenco dei presenti alla riunione;
- **Assenti**: elenco degli assenti alla riunione.

Per i verbali esterni si aggiungono le seguenti sezioni:

- **Esterni**: elenco dei membri esterni al gruppo presenti in riunione;
- **Presa visione esterni**: contiene, se necessaria, la firma dei presenti esterni a fondo pagina.

Un altro cambiamento è la sezione contenuto documento, che si distingue tra verbali interni e verbali esterni.
Per i verbali interni è presente la sezione "Ordine del giorno", che contiene gli argomenti trattati, espressi da un titolo, e le relative discussioni e scelte prese a riguardo. Contiene poi la sezione di "Attività da svolgere", contenente le attività da svolgere decise in riunione, per il prossimo periodo.
Per i verbali esterni è presente il "Resoconto dell'incontro", che presenta il motivo della richiesta d'incontro con l'azienda; in seguito vengono presentati gli argomenti trattati nella riunione, utilizzando sempre titoli descrittivi del tema e una relativa descrizione approfondita dei risultati della discussione.
Infine è presente la sezione "Conclusioni", che tira le somme dell'incontro e riassume i punti trattati, le scelte discusse e pone l'attenzione sulle attività da svolgere.

3.1.7 Regole tipografiche

3.1.7.1 Nome file dei documenti progetto

Il nome che viene dato ai file dei documenti prodotti deve avere la prima lettera di ogni sostantivo maiuscola. Per i verbali va inserita la data di svolgimento della riunione, mentre per gli altri documenti, deve contenere l'indicazione della attuale versione del documento. In particolare devono seguire la seguente convenzione di scrittura:

- **Verbale interno**: Verbale_Interno_YYYY-MM-DD;
- **Verbale esterno**: Verbale_Esterno_YYYY-MM-DD;
- **Norme di Progetto**: Norme_di_Progetto_vX.Y.Z;
- **Analisi dei Requisiti**: Analisi_dei_Requisiti_vX.Y.Z;
- **Piano di Progetto**: Piano_di_Progetto_vX.Y.Z;
- **Piano di Qualifica**: Piano_di_Qualifica_vX.Y.Z;
- **Glossario**: Glossario_vX.Y.Z;
- **Manuale Utente**: Manuale_Utente_vX.Y.Z;
- **Specifica Tecnica**: Specifica_Tecnica_vX.Y.Z.

3.1.7.2 Regole stilistiche

Vengono utilizzate le seguenti regole stilistiche:

- I titoli delle sezioni iniziano con la lettera maiuscola;
- Ciascuna voce di un elenco (ad eccezione degli elenchi di attributi/metodi di classi o di rotte API$_G$) deve essere seguita da un punto e virgola, ad eccezione dell'ultima che deve terminare con un punto;
- Le date vengono scritte nel formato YYYY-MM-DD (anno-mese-giorno);
- Quando si fa riferimento a informazioni contenute in un documento diverso da quello attuale, bisogna indicare la versione del documento e, se necessario, specificare anche la sezione pertinente (Ex: Analisi dei Requisiti v.1.0.0 sez. 1.1.1).

Si utilizzerà il grassetto per evidenziare:

- **Titoli** delle sezioni;
- Parole di rilievo in **elenchi puntati**.

Si utilizzerà il sottolineato e la "G" a pedice per:

- **Termini di Glossario**.

Si utilizzerà il colore blu di LaTeX per:

- **Gli indirizzi mail**;
- **I link**.

3.1.8 Abbreviazioni

Nella stesura del documento potrebbero apparire delle abbreviazioni di termini noti che per comodità potrebbero essere accorciati:

- **AdR**: Analisi dei Requisiti;
- **CA**: Customer Acceptance;
- **CI**: Configuration Item;
- **ITS**: Issue Tracking System$_G$;
- **PB**: Product Baseline$_G$;
- **PoC**: Proof of Concept$_G$;
- **PdP**: Piano di Progetto;
- **PdQ**: Piano di Qualifica;
- **RTB**: Requirements and Technology Baseline$_G$.

3.1.9 Strumenti

I seguenti strumenti sono stati scelti dal gruppo per la realizzazione della documentazione:

- **GitHub**: utilizzato per il versionamento, la collaborazione e la distribuzione dei documenti;
- **LaTeX**: utilizzato per la redazione dei documenti per ottenere una formattazione testuale adeguata e creare documenti di alta qualità;
- **Google Docs**: utilizzato per la stesura dei documenti che richiedono un'alta collaborazione.

3.2 Verifica

3.2.1 Descrizione

È un processo che segue l'intero ciclo di vita del software. Il processo serve a garantire che ciascun prodotto rispetti le aspettative. È fondamentale quindi dotarsi di linee guida e procedure. Compito del verificatore è quello di analizzare e valutare la correttezza del prodotto rispetto alle specifiche.

3.2.2 Analisi statica

Viene condotta senza necessità di esecuzione dell'oggetto di verifica. Sfrutta metodi di lettura che possono essere manuali o automatici.
I principali sono:

- **Walkthrough**: lavoro svolto tra verificatore e autore che prevede un esame dell'oggetto studiando ogni singola parte senza presupposti, in quanto non è ancora chiaro dov'è la maggior probabilità di trovare difetti;
- **Inspection**: lavoro svolto dal verificatore che, sapendo cosa cercare, può fare una lettura mirata dell'oggetto di verifica. La definizione migliore di cosa va verificato consente maggiormente l'uso di automazioni.

3.2.3 Analisi dinamica

Necessita l'esecuzione dell'oggetto di verifica. Si svolge tramite l'esecuzione di test. L'esito del test deve essere decidibile in modo automatico. È opportuno quindi definire attentamente i casi ed essere più specifici possibili. Ideale è trovare il numero minimo di test necessari per verificare nella maniera più esaustiva possibile un oggetto.

- **Test di unità**: verificano le più piccole unità del codice. Rappresentano la maggioranza dei test. Sono automatizzati e sono implementati durante la progettazione. Sono in grado di rilevare la maggior parte dei difetti;
- **Test di integrazione**: verificano la corretta integrazione fra le unità già testate. Aggiungendo componenti a insiemi di componenti già verificati permette di individuare facilmente il difetto, rendendo così reversibile l'integrazione fatta;
- **Test di sistema**: verificano il corretto funzionamento dell'intero sistema. Precedono il collaudo;
- **Test di regressione**: verificano che le nuove modifiche apportate non introducano nuovi difetti al resto del sistema;
- **Test di accettazione**: verificano che il prodotto soddisfi i requisiti richiesti dal committente.

3.2.4 Strumenti

Per il processo di verifica sono stati selezionati i seguenti strumenti:

- **pytest**: strumento per il test di codice in linguaggio Python che è in grado di eseguire test di unità$_G$, integrazione, end-to-end, funzionali;
- **Cypress**: strumento per testare il frontend del prodotto tramite test end-to-end;
- **ESLint**: strumento di analisi statica per JavaScript e TypeScript che identifica e corregge automaticamente errori di sintassi, problemi di stile e potenziali bug;
- **Pylint**: strumento di analisi statica per Python che rileva errori di codice, violazioni di stile e potenziali bug.

3.3 Validazione

3.3.1 Scopo ed Importanza

La validazione è il processo che conferma, attraverso dimostrazioni oggettive come test, prove o altri metodi, che il prodotto software soddisfa i requisiti specificati e risponde alle aspettative degli utenti finali. Risponde alla domanda "Did I build the right thing?", cioè: "Ho costruito ciò che era effettivamente richiesto?". Questo processo, definito nello standard ISO/IEC 12207:1995, è fondamentale per garantire che il prodotto finale:

- Soddisfi integralmente i requisiti specificati dal cliente;
- Funzioni correttamente nel suo ambiente operativo;
- Sia intuitivo, di facile utilizzo e conforme alle logiche di progettazione;
- Dimostri efficacia$_G$ nel rispondere alle necessità degli utenti finali.

Per una buona validazione è importante un confronto diretto con il proponente e il committente, che permetterà di raccogliere opinioni e di comprendere se è presente un effettiva corrispondenza tra il prodotto sviluppato e quello richiesto. La validazione è necessaria per poter rilasciare il proprio prodotto software, perché assicura la sua adeguatezza a quanto richiesto.

3.3.2 Procedura di Validazione

Il processo di validazione è combinato al processo di verifica e utilizza anch'esso test, come quelli di verifica precedentemente descritti. Questi strumenti consentono un'analisi approfondita e oggettiva delle funzionalità e dei requisiti del prodotto, supportando l'intero ciclo di validazione. Il test di accettazione$_G$ determinerà la conclusione e la buona riuscita della validazione del prodotto. Tali test hanno l'obiettivo di valutare il soddisfacimento dei casi d'uso ipotizzati, la conformità dei requisiti obbligatori e dei requisiti eventualmente stabiliti con l'azienda proponente.

3.4 Gestione della configurazione

Lo scopo è quello di identificare, organizzare e controllare le modifiche apportate a documenti e prodotti. Viene utilizzato durante l'intero ciclo di vita di un prodotto.

3.4.1 Versionamento

Il formato del versionamento adottato è X.Y.Z con:

- **X**: viene aumentato quando il prodotto è pronto per una delle 3 fasi del progetto (RTB, PB, CA);
- **Y**: viene aumentato in caso di modifiche significative;
- **Z**: viene aumentato per modifiche minori che non portano ad un impatto significativo.

L'aumento di un valore porta all'azzeramento dei valori meno significativi.

3.4.2 Strumenti

Per i repository si utilizza **git** come strumento di controllo di versione e **GitHub** come servizio di hosting$_G$.
Il gruppo utilizza 4 repository pubblici:

- **docs**: contiene i sorgenti della documentazione;
- **Documentazione**: contiene i PDF del progetto;
- **SWEvenTeam17.github.io**: serve per gestire la pagina web tramite GitHub Pages;
- **ArtificialQI**: contiene il codice del MVP$_G$.

La repository **Documentazione** si divide in 3 cartelle:

- **Candidatura**:
  - Lettera di Presentazione;
  - Dichiarazione degli Impegni;
  - Studio dei Capitolati;
  - Cartella **Verbali**, che contiene i verbali divisi tra interni ed esterni.

- Le due cartelle, **RTB** e **PB**, strutturate in questo modo:
  - **Documenti interni**: cartella contenente i documenti destinati alla consultazione interna al gruppo;
  - **Documenti esterni**: cartella contenente i documenti destinati alla consultazione da parte di membri esterni al gruppo;
  - **Verbali**: cartella contenente due sottocartelle, Verbali interni, che contiene al suo interno tutti i verbali interni redatti, e la cartella Verbali esterni, che contiene al suo interno tutti i verbali esterni redatti.

La repository **ArtificialQI** si divide in 5 cartelle:

- **.github/workflows**: contenente i file di configurazione delle GitHub Actions;
- **client**: contenente il frontend del MVP;
- **microservices**: contenente la logica di gestione dei LLM;
- **ollama**: contenente la containerizzazione di Ollama$_G$;
- **server**: contenente il backend del core del MVP.

3.4.3 Flusso di lavoro

Per quel che riguarda la gestione del codice, si utilizza una versione più snella della rinomata strategia di branching Git Flow. Nel nostro caso si utilizzano solo 3 tipi di branch$_G$:

- **Main**: è il branch principale, considerato di produzione;
- **Develop**: è il branch di sviluppo, utilizzato per l'integrazione continua del prodotto;
- **Feature**: in seguito alla decisione di aggiungere una nuova funzionalità viene aperto un feature branch partendo da develop. La funzionalità implementata viene fusa con il branch develop.

3.4.4 Comandi

Sono riportati i comandi di maggior utilità:

- Aprire una nuova feature:

  ```
  git checkout develop
  git checkout -b nomefeature
  ```

  Il primo push va eseguito nel seguente modo:

  ```
  git push -u origin nomefeature
  ```

- Caricare le modifiche apportate:

  ```
  git fetch
  git merge
  ```

  Risolvere eventuali conflitti e poi fare il commit$_G$:

  ```
  git add .
  git commit -m "descrizione commit"
  git push nomefeature
  ```

- Per ogni branch feature va stabilita la frequenza con cui va aggiornato al branch develop. Tale operazione viene svolta effettuando l'aggiornamento del branch develop

  ```
  git checkout develop
  git fetch
  git merge
  ```

  e successivamente eseguire il merge con nomefeature

  ```
  git checkout nomefeature
  git merge main
  ```

  risolvere eventuali conflitti e procedere al commit e push di nomefeature.
  Questo è un processo essenziale per ridurre le possibilità di conflitto durante una pull request$_G$.

3.5 Gestione della qualità

È il processo che fornisce garanzie che il prodotto software e i processi siano conformi ai requisiti. La sua implementazione parte dalla definizione di una strategia per certificare la qualità. Il piano di qualità identifica le risorse e le specifiche necessarie per garantire che il prodotto aderisca agli standard stabiliti. Tramite delle verifiche continue viene accertato se il prodotto rispetta quanto stabilito.
Queste attività sono discusse nel Piano di Qualifica che descrive quali misurazioni e verifiche effettuare per assicurare la qualità. L'uso delle metriche permette di ottenere dei riscontri oggettivi sull'andamento del progetto e del rispetto degli obiettivi prefissati.

4 Processi organizzativi

4.1 Gestione dei processi

4.1.1 Introduzione

La gestione dei processi si occupa di definire, attuare e ottimizzare i processi necessari alla realizzazione del software, con l'obiettivo di conseguire i risultati desiderati e soddisfare le aspettative degli stakeholder$_G$.

4.1.2 Scopo

La gestione dei processi definisce i compiti che i membri del gruppo dovranno compiere permettendo lo svolgimento delle attività in maniera efficace.

4.1.3 Descrizione

Le attività del processo di gestione sono:

- **Inizio e definizione dei processi**: stabilire i requisiti dei processi e verificarne la fattibilità verificando che le risorse necessarie siano disponibili;
- **Pianificazione**: stabilire obiettivi, scadenze, risorse e responsabilità per ogni processo;
- **Esecuzione e controllo**: vengono eseguite le attività e intraprese azioni correttive qualora l'analisi evidenzi problemi;
- **Revisione e valutazione**: valutare se quanto prodotto soddisfa i requisiti;
- **Conclusione**: quando il prodotto, le attività e i compiti sono completati viene verificato che i processi siano conclusi in base alle regole stabilite.

4.1.4 Pianificazione

4.1.4.1 Descrizione

La pianificazione ha lo scopo di produrre e coordinare una programmazione efficace delle attività durante il ciclo di vita del software. È compito del responsabile occuparsi della pianificazione gestendo ruoli, risorse e scadenze affinché vengano rispettati gli obiettivi stabiliti. La pianificazione viene inserita all'interno del Piano di Progetto.

4.1.4.2 Ruoli

Nel corso del progetto, i membri del team ricopriranno sei ruoli diversi:

4.1.4.2.1 Responsabile di Progetto

Il responsabile coordina il gruppo e funge da punto di riferimento sia per il committente che per il team.
I suoi principali compiti sono:

- **Pianificare le attività**: dopo aver analizzato l'andamento del progetto, definisce quali attività svolgere, gli assegnatari, i costi, i rischi e le tempistiche;
- **Coordinare il gruppo**;
- **Curare i rapporti con soggetti esterni**;
- **Approvare la documentazione**.

4.1.4.2.2 Amministratore di Progetto

L'amministratore$_G$ si occupa del controllo e della gestione dell'ambiente di lavoro, assicurando il rispetto delle norme di progetto.
I suoi principali compiti sono:

- Gestire e risolvere le problematiche legate ai processi;
- Gestire il versionamento della documentazione;
- Redigere la documentazione;
- Gestire la configurazione del prodotto;
- Migliorare l'ambiente di lavoro fornendo gli strumenti necessari.

4.1.4.2.3 Analista

L'analista$_G$, presente nelle fasi iniziali del progetto, ha il compito di identificare, documentare e comprendere a fondo le esigenze del progetto trasformandole in requisiti chiari e dettagliati.
I suoi principali compiti sono:

- Analizzare il problema e individuare gli obiettivi;
- Definire i requisiti, dopo aver valutato la complessità del problema;
- Raccogliere e analizzare le richieste, anche implicite, dei committenti;
- Redigere il documento Analisi dei Requisiti.

4.1.4.2.4 Progettista

Il progettista$_G$ è il referente per le scelte tecniche e progettuali: basandosi sul lavoro dell'analista, prende decisioni riguardo l'architettura del prodotto. Segue il processo di sviluppo del pro
dotto supervisionandolo, senza però occuparsi della relativa manutenzione$_G$. I suoi principali compiti sono:

- Progettare l'architettura del prodotto seguendo gli standard allo stato dell'arte;
- Operare le scelte che consentano di arrivare ad un prodotto che, pur soddisfando i criteri di affidabilità ed efficienza$_G$, risulti economico e conforme ai requisiti;
- Assicurare un basso grado di accoppiamento nel sistema;
- Redigere il documento di Specifica Tecnica.

4.1.4.2.5 Programmatore

Il programmatore$_G$ è responsabile della codifica del software, implementando le specifiche fornite dall'analista e l'architettura sviluppata dal progettista.

- Scrivere codice manutenibile che rispetti le Specifiche Tecniche e sia conforme alle Norme di Progetto;
- Codificare le componenti dell'architettura seguendo le linee guida del progettista;
- Creare test per verificare e validare il codice;
- Redigere il Manuale Utente$_G$.

4.1.4.2.6 Verificatore

Compito del verificatore è controllare che il lavoro svolto dall'intero team risulti conforme agli standard prefissati.

- Basandosi sugli strumenti e le tecniche inseriti nelle Norme di Progetto, verificare la conformità dei prodotti;
- Identificare e segnalare eventuali errori;
- Assicurarsi che i prodotti soddisfino i requisiti funzionali e di qualità.

4.1.4.3 Ticketing

Viene utilizzato l'Issue$_G$ Tracking System (ITS$_G$) di GitHub come processo di gestione e tracciamento delle attività. Il responsabile si occupa di creare i task ed assegnarli ai vari membri del gruppo, il cui stato di avanzamento è consultabile all'interno della Board$_G$.
Quando le issues vengono create sono caratterizzate da:

- **Titolo**: identifica il compito da svolgere;
- **Descrizione**: note per migliorare la comprensione;
- **Assegnatario**: chi deve svolgere la issue;
- **Label**: tag per identificare il tipo di issue;
- **Milestone**: a quale milestone$_G$ appartiene;
- **Stato**: avanzamento della issue;
- **Project**: il progetto a cui viene associata la issue.

Per terminare il ciclo di vita di una issue è necessario seguire la seguente procedura:

1. Il responsabile crea una nuova issue, in stato "to do", e la assegna;
2. Quando inizia il lavoro la issue cambia stato ad "in progress" e viene creato un branch per la issue;
3. Una volta terminato il lavoro sulla issue viene aperta una pull request, assegnata ad un verificatore;
4. Se il verificatore approva la issue, conferma la pull request e GitHub fa il merge sul branch develop e la issue viene considerata completata. In caso contrario, il lavoro sulla issue verrà modificato secondo i suggerimenti del verificatore.

4.1.5 Coordinamento

4.1.5.1 Descrizione

Il coordinamento è l'attività che gestisce le comunicazioni e gli incontri tra le parti coinvolte nel progetto: membri del gruppo, proponente e committenti. Serve per assicurarsi che le parti interessate siano informate e che eventuali dubbi siano risolti, riducendo quindi il rischio di incomprensioni e conseguenti ritardi. Obiettivo fondamentale del coordinamento è proprio quello di favorire la collaborazione nel gruppo ottimizzando l'efficienza e la puntualità del lavoro.

4.1.5.2 Comunicazioni

Il gruppo, in base alle necessità, fa uso di comunicazioni sincrone o asincrone.

4.1.5.2.1 Comunicazioni sincrone

Le comunicazioni sincrone interne sono tenute sulla piattaforma Discord$_G$. Grazie alla creazione del server, è possibile effettuare videochiamate e condividere contenuti multimediali o informazioni aggiuntive.
Le comunicazioni sincrone esterne sono tenute su Google Meet (preferito anche dall'azienda proponente) in quanto non richiede l'installazione di alcun client.

4.1.5.2.2 Comunicazioni asincrone

Le comunicazioni interne asincrone sono gestite attraverso il servizio di messaggistica di Telegram$_G$, accessibile da qualsiasi dispositivo, in un gruppo dedicato.
Per le comunicazioni asincrone esterne, il responsabile è incaricato di gestire il dialogo attraverso l'apposito indirizzo email: sweventeam@outlook.it.

4.1.5.3 Riunioni

4.1.5.3.1 Riunioni interne

Si è scelto di svolgere i meeting interni ogni due settimane, normalmente al termine e al contestuale inizio di uno sprint$_G$. Generalmente le riunioni sono programmate il giovedì pomeriggio. In caso di necessità, è possibile richiedere riunioni straordinarie durante la settimana tramite il canale dedicato su Telegram, con data e orario stabiliti attraverso un sondaggio. Tutte le riunioni online si svolgono nel canale Discord designato.

Relativamente ai meeting interni, il responsabile:

- definisce i punti all'ordine del giorno;
- comunica le eventuali variazioni orarie al gruppo;
- nomina un segretario che prenda appunti su quanto detto;
- pianifica le attività da svolgere;
- assegna i task a ciascun membro del gruppo;
- approva il relativo verbale.

I partecipanti hanno il dovere di essere presenti e puntuali alle riunioni. Qualora uno dei membri fosse assente, il responsabile si occuperà di riassumere ciò che è stato deciso durante il meeting nel canale apposito di Telegram. Se più di qualche membro del gruppo non potesse partecipare alla riunione nella data e nell'orario stabiliti, si procede programmando un nuovo incontro, concordando data e ora tramite un sondaggio sullo stesso canale.

4.1.5.3.2 Riunioni esterne

Hanno lo scopo di illustrare a proponente e/o committenti lo stato di avanzamento del progetto, raccogliere feedback e chiarire eventuali dubbi. È compito del responsabile individuare le date da proporre, dopo aver condotto un sondaggio all'interno del gruppo nel tentativo di massimizzare la partecipazione dei componenti. Tutti i componenti del gruppo, ove possibile, sono tenuti a garantire la presenza, salvo impegni improrogabili.
Nella mail di conferma della riunione è necessario includere il link di Google Meet per la partecipazione.

4.1.5.4 Verbali

I verbali hanno lo scopo di riassumere i punti trattati, le decisioni prese e le attività assegnate durante una riunione. Inoltre concorrono a mantenere traccia delle discussioni e garantire che tutti i membri del team abbiano una chiara comprensione degli accordi raggiunti.

4.1.5.4.1 Verbali interni

Le riunioni interne hanno l'obiettivo di analizzare il periodo precedente, discutere i punti all'ordine del giorno e pianificare le attività future. Il compito di redigere il verbale è affidato a un redattore, il quale deve assicurarsi di includere tutte le informazioni rilevanti discusse, seguendo quanto indicato in questo documento nella sezione 3.1 Documentazione e sfruttando i template predisposti.

4.1.5.4.2 Verbali esterni

Anche per le riunioni esterne viene redatto un verbale seguendo le indicazioni già descritte in "Verbali interni" e assicurandosi di inserire tra i partecipanti al meeting anche i committenti e/o il proponente.

4.2 Miglioramento

4.2.1 Obiettivo

Lo standard ISO/IEC 12207:1995 definisce il miglioramento come il processo che stabilisce, valuta, misura, controlla e migliora un processo del ciclo di vita del software.

4.2.2 Definizione dei processi

Vengono stabiliti e documentati una serie di processi organizzativi per tutti i processi del ciclo di vita del software. Inoltre vengono implementati meccanismi di controllo per poter sviluppare, monitorare e migliorare i processi.

4.2.3 Valutazione dei processi

Per procedere al miglioramento, occorre documentare, sviluppare e applicare una procedura di valutazione del processo. A tal fine, ciclicamente vengono condotte revisioni ed analisi dei processi allo scopo di migliorarli. Dati, come le metriche, forniscono una valida fonte per poter condurre la valutazione dei processi.

4.2.4 Miglioramento dei processi

Vengono apportati miglioramenti ai processi che, dopo essere stati valutati, hanno dimostrato la necessità di essere modificati. La documentazione del processo deve essere aggiornata per riflettere le modifiche apportate ai processi organizzativi. Raccogliere i dati e la valutazione dei processi funge da feedback e orienta verso un continuo miglioramento, evitando il rischio di regressione.

4.3 Formazione

4.3.1 Introduzione

Per lavorare in maniera efficace e corretta al progetto è necessario che i componenti abbiano compreso in maniera chiara tutto ciò che sta alla base del progetto. Serve che tutti i componenti riescano a completare, senza riscontrare difficoltà eccessive, i compiti che sono legati a determinati ruoli e per colmare le mancanze che ogni studente avrà in qualche ambito, sono previste attività di formazione.
In seguito viene spiegato come verranno acquisite e approfondite le conoscenze richieste per un buon avanzamento del progetto al interno del gruppo, ricercando un metodo che permetta di raggiungere risultati formativi rapidi ed efficaci.

4.3.2 Descrizione

I membri del gruppo necessitano di una formazione adeguata in tre ambiti principali. In primo luogo, è fondamentale acquisire conoscenze relative all'ingegneria del software. A tal fine, il team potrà avvalersi delle lezioni offerte dal corso di studi e, se necessario, approfondire specifici argomenti particolarmente rilevanti per il progetto.

Successivamente, sarà essenziale comprendere i temi legati al capitolato, in particolare il funzionamento degli LLM e la loro interazione con le altre componenti del sistema. Poiché nel percorso di studi non sono previsti corsi obbligatori dedicati specificamente agli LLM, sarà necessaria una formazione preliminare sull'argomento. È probabile che alcuni membri del gruppo abbiano già una conoscenza più approfondita rispetto ad altri; per questo motivo, il progetto favorirà un approccio collaborativo, attraverso riunioni in cui i più esperti potranno condividere il loro sapere o tramite la condivisione di materiali utili, come documenti validati dal team, per agevolare un apprendimento più mirato.

Infine, il gruppo dovrà acquisire competenze tecniche relative agli strumenti e alle tecnologie che verranno utilizzati nel progetto. Non è detto che tali strumenti siano già noti ai membri, pertanto sarà necessaria una fase di studio continua, con un'attenzione particolare alle fasi iniziali. L'obiettivo sarà comprendere gli aspetti fondamentali delle tecnologie adottate, concentrandosi sulle funzionalità più utili ai fini del progetto. Anche in questo contesto, la collaborazione sarà fondamentale per ottimizzare i tempi di apprendimento: momenti di studio individuale si alterneranno a sessioni di condivisione delle conoscenze, con il supporto dei membri più esperti, che potranno guidare il gruppo e facilitare la suddivisione del materiale di studio.

5 Standard per la qualità

La valutazione della qualità del software e dei relativi processi richiede un approccio metodico, supportato da standard internazionali riconosciuti. Per garantire risultati accurati e coerenti, faremo riferimento a due standard fondamentali: ISO/IEC 9126 e ISO/IEC 12207, che forniranno un quadro solido per misurare la qualità del prodotto software e per gestire l'intero ciclo di vita del progetto.

5.1 Standard ISO/IEC 9126

Lo standard ISO/IEC 9126 è un documente di specifiche internazionale che definisce i criteri per valutare la qualità di un software in modo completo, garantendo che il prodotto finale soddisfi le esigenze dell'utente e del destinatario finale. Lo standard si concentra su aspetti cruciali per la qualità del software, fornendo un modello per misurare e analizzare vari attributi.
Tra i principali criteri di valutazione, troviamo:

- Funzionalità;
- Affidabilità;
- Usabilità;
- Efficienza;
- Manutenibilità$_G$;
- Portabilità.

5.1.1 Funzionalità

Questo criterio riguarda la capacità del software di fare ciò che ci si aspetta che faccia, offrendo funzionalità che soddisfano sia i requisiti specificati che quelli impliciti.
Rispetta le seguenti caratteristiche:

- **Adeguatezza**: il software deve essere in grado di svolgere le attività per le quali è stato creato;

- **Accuratezza**: ogni funzione deve essere eseguita correttamente, senza errori (in linea con le aspettative);
- **Interoperabilità**: il software deve essere in grado di interagire con altri sistemi.

5.1.2 Affidabilità

Questo criterio riguarda la capacità del software di funzionare correttamente e senza interruzioni nel tempo, anche in condizioni di utilizzo non sempre ideali. Rispetta le seguenti caratteristiche:

- **Maturità**: il software deve mantenere una stabilità durante l'esecuzione e non deve incorrere in risultati scorretti o errori;
- **Tolleranza agli errori**: il software deve essere in grado di gestire in modo efficace le situazioni inaspettate, come input errati o malfunzionamenti, senza compromettere l'intero sistema;
- **Recuperabilità**: il software deve essere in grado di tornare a lavorare ad un alto livello di prestazioni in seguito ad un malfunzionamento.

5.1.3 Usabilità

Questo criterio riguarda quanto sia facile e intuitivo per gli utenti interagire con il software.
Rispetta le seguenti caratteristiche:

- **Comprensibilità**: indica quanto sia semplice per l'utente capire l'interfaccia del software, le informazioni presentate e le modalità per compiere le operazioni necessarie;
- **Apprendibilità**: rappresenta la facilità con cui un nuovo utente può imparare ad usare agilmente il software;
- **Operabilità**: descrive quanto facilmente gli utenti possono utilizzarlo e controllarlo;
- **Attrattività**: riguarda l'aspetto estetico dell'interfaccia e l'impressione complessiva che il design lascia sugli utenti;
- **Aderenza all'usabilità**: indica quanto il software segue le linee guida e le best practice di usabilità.

5.1.4 Efficienza

Questo criterio riguarda l'ottimizzazione dell'uso delle risorse del software, come tempo, memoria e potenza di calcolo, per eseguire le operazioni in modo rapido e senza sprechi.
Rispetta le seguenti caratteristiche:

- **Prestazioni**: la velocità con cui il software risponde alle richieste dell'utente;
- **Uso delle risorse**: quanto il software ottimizza memoria, CPU e altre risorse per eseguire le funzioni senza rallentamenti o consumi eccessivi.

5.1.5 Manutenibilità

Questo criterio riguarda la facilità con cui il software può essere modificato, aggiornato o corretto dopo il rilascio, per risolvere problemi, migliorare prestazioni o adattarsi a nuove esigenze.
Rispetta le seguenti caratteristiche:

- **Modificabilità**: la facilità con cui il software può essere modificato in fasi avanzate;
- **Analizzabilità**: la capacità di individuare rapidamente le cause di eventuali problemi nel software;
- **Stabilità**: la capacità del software di mantenere prestazioni coerenti anche dopo modifiche, senza introdurre nuovi errori;
- **Testabilità**: la facilità con cui il prodotto software e le sue componenti possono essere testate e verificate.

5.1.6 Portabilità

Questo criterio riguarda la capacità del software di essere facilmente trasferito e utilizzato su diverse piattaforme o ambienti senza perdere le sue prestazioni e funzionalità.
Rispetta le seguenti caratteristiche:

- **Adattabilità**: la facilità con cui il software può essere adattato a nuove piattaforme hardware o sistemi operativi;
- **Installabilità**: la semplicità con cui il software può essere installato su diversi ambienti senza difficoltà;
- **Sostituibilità**: la facilità con cui il software può sostituire o essere sostituito da altre applicazioni nello stesso ambiente.

5.2 Suddivisione standard ISO/IEC 12207

5.2.1 Processi primari

I processi primari si concentrano sulla creazione effettiva del prodotto software lungo tutto il suo ciclo di vita, dall'avvio e la presa in carica del progetto, fino alla consegna e la manutenzione. Rappresenta il processo più importante che produce effettivamente il prodotto e viene accompagnato dagli altri processi, che aiutano quello primario lungo tutta la sua realizzazione. Si distinguono attività fondamentali ad esempio la raccolta dei requisiti, la progettazione e la codifica.

5.2.2 Processi di supporto

I processi di supporto aiutano e facilitano le attività principali durante l'intero ciclo di vita del software. Comprendono attività come la stesura della documentazione, la gestione della configurazione e la qualità del software. L'obiettivo è assicurare che i processi primari possano svolgersi in modo efficace e corretto.

5.2.3 Processi organizzativi

I processi organizzativi definiscono il contesto e l'ambiente in cui si svolgono le attività di sviluppo software. Questi processi includono la pianificazione, la formazione, il miglioramento dei processi nel ciclo di vita e la gestione delle risorse necessarie per supportare le attività di sviluppo.

6 Metriche di qualità

6.1 Metriche di qualità del processo

6.1.1 Processi primari

6.1.1.1 Fornitura

- **MPC-01 - Budget Variance**: misura la percentuale di differenza tra il budget pianificato di un periodo e quello effettivamente utilizzato.

$$BV = \left(\frac{PV - AC}{PV}\right) \times 100$$

- **MPC-02 - Earned Value (EV)**: valore del lavoro effettivamente svolto alla fine di un periodo.

$$EV = \%\text{ di completamento} \times BAC$$

  dove "% di completamento" indica il rapporto tra le ore effettive e le ore totali a disposizione.

- **MPC-03 - Actual Cost**: misura i costi effettivamente sostenuti dall'inizio del progetto fino al periodo attuale.

- **MPC-04 - Planned Value (PV)**: rappresenta il totale dei costi pianificati per periodo e viene calcolata prima che esso inizi.

$$PV = \%\text{ di completamento} \times BAC$$

dove "% di completamento" indica il rapporto tra le ore pianificate e le ore totali a disposizione.

- **MPC-05 - Cost Variance (CV)**: rappresenta lo scostamento dai costi pianificati.

$$CV = EV - AC$$

- **MPC-06 - Schedule Variance (SV)**: indica di quanto si è in anticipo o in ritardo con le attività pianificate.

$$SV = EV - PV$$

- **MPC-07 - Cost Performance Index (CPI)**: misura il rapporto tra il valore del lavoro effettivamente svolto ed il costo reale del lavoro fino al periodo di riferimento.

$$CPI = \frac{EV}{AC}$$

- **MPC-08 - Estimate to Complete (ETC)**: indica la stima dei costi rimanenti per completare il progetto.

$$ETC = EAC - AC$$

- **MPC-09 - Estimate at Completion (EAC)**: rappresenta il costo totale alla fine del progetto in base all'andamento attuale.

$$EAC = \left(\frac{BAC - EV}{CPI}\right) + AC$$

6.1.1.2 Sviluppo

- **MPC-10 - Variazione dei requisiti**: misura la variazione dei requisiti durante il progetto.

6.1.2 Processi di supporto

6.1.2.1 Documentazione

- **MPC-11 - Indice Gulpease**: stima la comprensibilità del testo.

$$\text{Gulpease} = 89 + \left(\frac{300 \times \text{numero delle frasi} - 10 \times \text{numero delle lettere}}{\text{numero delle parole}}\right)$$

- **MPC-12 - Correttezza ortografica**: misura la presenza di errori ortografici nel testo.

6.1.2.2 Verifica

- **MPC-13 - Code Coverage**: stima la percentuale di codice attraversato dai test rispetto al totale.
- **MPC-14 - Percentuale di test case superati**: percentuale di casi dei test superati.

$$\left(\frac{\text{test superati}}{\text{test totali}}\right) \times 100$$

- **MPC-15 - Percentuale di test case falliti**: percentuale di casi test falliti.

$$\left(\frac{\text{test falliti}}{\text{test totali}}\right) \times 100$$

6.1.2.3 Gestione della qualità

- **MPC-16 - Percentuale di metriche soddisfatte**: misura quante metriche sono state effettivamente adottate e soddisfatte, rispetto a quelle definite.

6.1.3 Processi organizzativi

6.1.3.1 Gestione dei processi

- **MPC-17 - Rischi inattesi**: misura il numero di rischi non previsti nel corso del progetto.

6.2 Metriche di qualità del prodotto

6.3 Funzionalità

- **MPD-01 - Copertura requisiti obbligatori**: indica la percentuale dei requisiti obbligatori soddisfatti.

$$\left(\frac{\text{requisiti obbligatori soddisfatti}}{\text{requisiti obbligatori totali}}\right) \times 100$$

- **MPD-02 - Copertura requisiti desiderabili**: indica la percentuale dei requisiti desiderabili soddisfatti.

$$\left(\frac{\text{requisiti desiderabili soddisfatti}}{\text{requisiti desiderabili totali}}\right) \times 100$$

- **MPD-03 - Copertura requisiti opzionali**: indica la percentuale dei requisiti opzionali soddisfatti.

$$\left(\frac{\text{requisiti opzionali soddisfatti}}{\text{requisiti opzionali totali}}\right) \times 100$$

6.3.1 Affidabilità

- **MPD-04 - Code coverage**: stima la percentuale di codice attraversato dai test rispetto al totale;
- **MPD-05 - Branch coverage**: indica la percentuale dei rami decisionali del codice coperti dai test;
- **MPD-06 - Statement coverage**: indica la percentuale degli statement del codice coperti dai test;
- **MPD-07 - Densità errori**: misura la frequenza con cui si verificano guasti o difetti del prodotto.

6.3.2 Manutenibilità

- **MPD-08 - Complessità ciclomatica**: misura la complessità del codice in termini di percorsi logici;
- **MPD-09 - Coefficiente di accoppiamento**: misura il livello di dipendenza tra moduli, classi o funzioni;
- **MPD-10 - Code smell**: indica la potenziale presenza di problemi nel codice sorgente.

6.3.3 Usabilità

- **MPD-11 - Facilità di utilizzo**: misura l'usabilità di un sistema software;
- **MPD-12 - Tempo di apprendimento**: misura il tempo massimo richiesto per apprendere l'utilizzo del prodotto.

6.3.4 Efficienza

- **MPD-13 - Utilizzo delle risorse**: misura quanto efficientemente vengono utilizzate le risorse disponibili;
- **MPD-14 - Tempo medio di risposta**: misura quanto è efficiente e reattivo un sistema software.